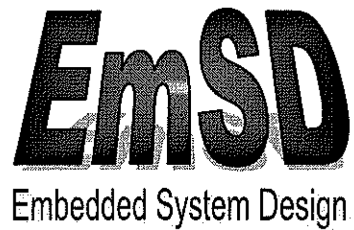
HOGESCHOOL VOOR WETENSCHAP & KUNST
DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER

Digitale Synthese

Bachelor in de Industriële Wetenschappen: elektronica-ICT
3e jaar, 5e semester

3Ba E

Ba3.EL.06



ir. J. Meel

april 2008

Bachelor in de Industriële Wetenschappen: elektronica-ICT
3e jaar, 5e semester

Digitale Synthese 1

Ba3.EL.06 (SPBa3.EL.06)

ir. J. Meel

1 Doelstelling

De student kan zelfstandig 'complexe' digitale systemen op een systematische wijze (bottom-up) ontwerpen op basis van:

- ***de VHDL-beschrijving van de functionaliteit van het systeem volgens een synchrone ontwerpmethodologie***
- ***de simulatie en verificatie van de functionaliteit beschreven in VHDL***
- ***de logische synthese van de functionaliteit***
- ***de implementatie in een FPGA***
- ***de verificatie door systeemtesten.***

2 Competenties

Bachelorcompetentie	Ba Comp-Matrix
kan de specificatie van een nieuwe schakeling uit toegepast wetenschappelijk onderzoek analyseren	IV1, OV1, OV2, SV1
kan een digitale schakeling bottom-up ontwerpen op basis van (herbruikbare) modules	KI3, AV2, IV3, OV1, OV2, OV3, SV3
kan de functionaliteit van digitale schakelingen met VHDL beschrijven	KI3, AV1, AV2, IV3
kan synchrone ontwerpmethodologie voor digitale hardware toepassen	KI3, AV2, IV3
kan de VHDL beschrijving van digitale schakelingen verifiëren met simulatie a.h.v. test-benches (kan de gepaste inputvectoren opstellen en de resulterende outputvectoren analyseren)	KI3, AV2, OV1
kan de VHDL beschrijving van digitale schakelingen met logische synthese omzetten naar een implementeerbare beschrijving	KI3, AV2
kan een digitaal ontwerp implementeren in FPGA-architecturen	KI3, AV2
kan het kritische pad van een geïmplementeerde digitale schakeling bepalen	
kan een digitaal systeem in een hedendaagse EDA-omgeving ontwikkelen op architectuurniveau (Modelsim, ISE, EDK): beschrijven, simuleren, synthetiseren (logische synthese) en implementeren (FPGA)	KI3, IV2, IV3
heeft inzicht in het gedrag van elementaire digitale bouwstenen	KI3
kan zijn activiteiten plannen, rekening houdend met opgegeven deadlines	SV3
kan de informatie, ideeën, problemen en oplossingen communiceren met vakgenoten	AV3

3 Inhoud

	Kennis	Ontwerp-vaardigheid		Taal	Architectuur
Ba sem 5	Ontwerptechnieken ▪ synchroon ontwerp ▪ ontwerp voor hoge snelheid ▪ fixed-point ontwerp ▪ architecturen Implementatie ▪ programmatietechnologie ▪ FPGA architectuur	systeemconcept analyseren, module-integratie	bottom-up	VHDL	FPGA

Digitale Modules

1. Rekenkundige Combinatorische Functies (ALU, shifter, comparator, multiplier, MAC)
2. Tellers voor hoge snelheid
3. Direct Digital Synthesis

Ontwerpmethodologie

4. Synchroon Ontwerp voor hoge snelheid
5. Input/output conditionering voor synchroon ontwerp
6. Ontwerp voor testbaarheid

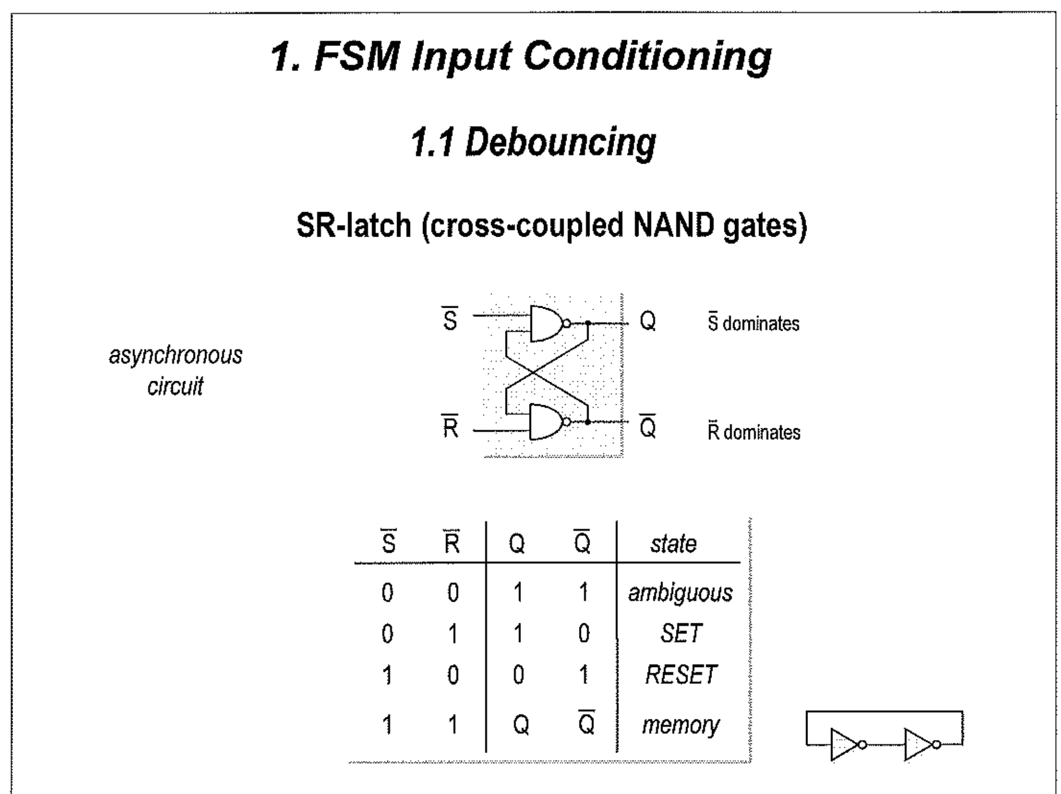
Technologie

7. Field Programmable Gate Array (FPGA)

HOGESCHOOL VOOR WETENSCHAP & KUNST | DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER

Digitale Synthese
FSM: Input/Output

ir. J. Meel
nov 2006



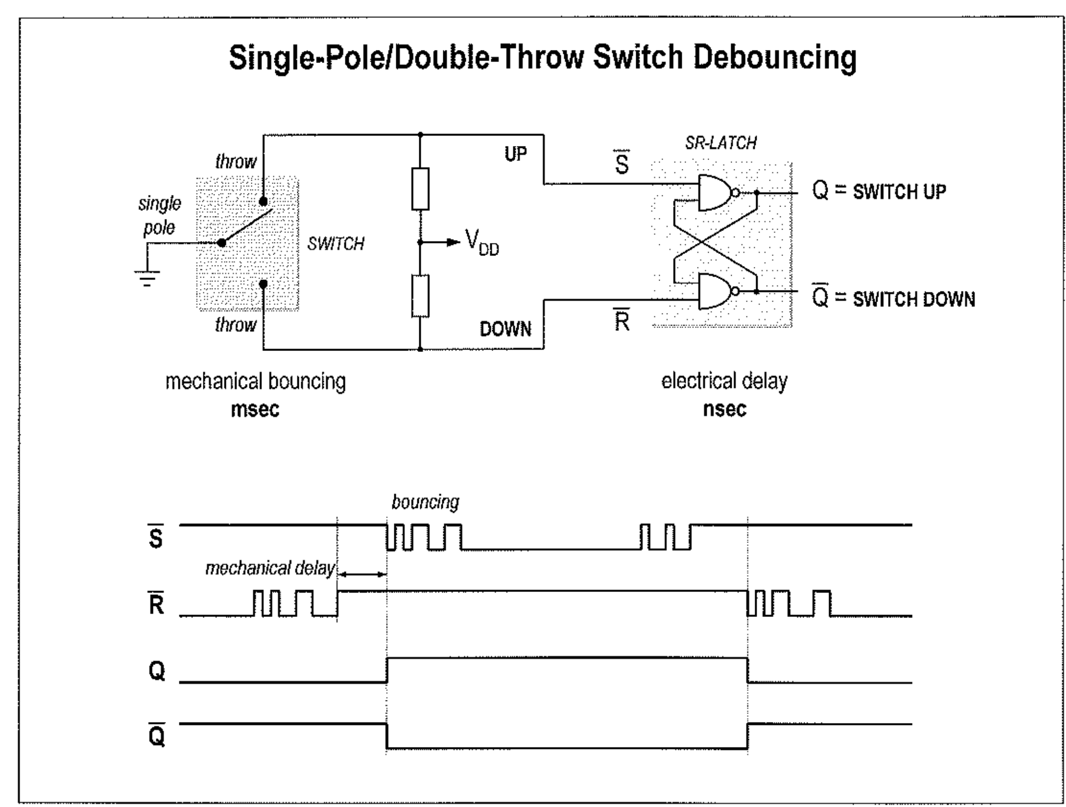
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	state
0	0	1	1	*ambiguous*
0	1	1	0	*SET*
1	0	0	1	*RESET*
1	1	Q	$\overline{Q}$	*memory*

The simplest memory element in digital design is the SR latch, which consists of two cross-coupled NAND gates. The SR-latch has two input signals, the set signal S' and the reset signal R'. It also has two output signals, Q and Q'. Finally, it has two states, a set state when Q = 1 (Q' = 0) and a reset state when Q = 0 (Q' = 1). As long as both input signals S' and R' are equal to 1, the SR latch persists in the same state. For example, if Q = 0, the output of the bottom NAND will be equal to 1, which, in turn, will keep the output of the top NAND at 0, because also S'= 1. Similarly, if Q = 1, the output of the bottom NAND will be equal to 0 because also R'= 1, which will make the output of the top NAND equal to 1.

If the S' input (R' input) becomes equal to 0, while keeping R'(S') at the value 1, the SR-latch goes to the set (reset) state.

If input signals S' and R' are both equal to 1, both output signals, Q and Q', must be equal to 1. This are ambiguous outputs, since they state of the SR-latch cannot be labeled as either set or reset. If one of the input signals is disasserted earlier than the other in this ambiguous state, the latch will end up in the state forced by the signal that was disasserted later. One problem inherent in the SR latch is the fact that if both S and Rare disasserted at the same time, the latch state cannot be predicted.

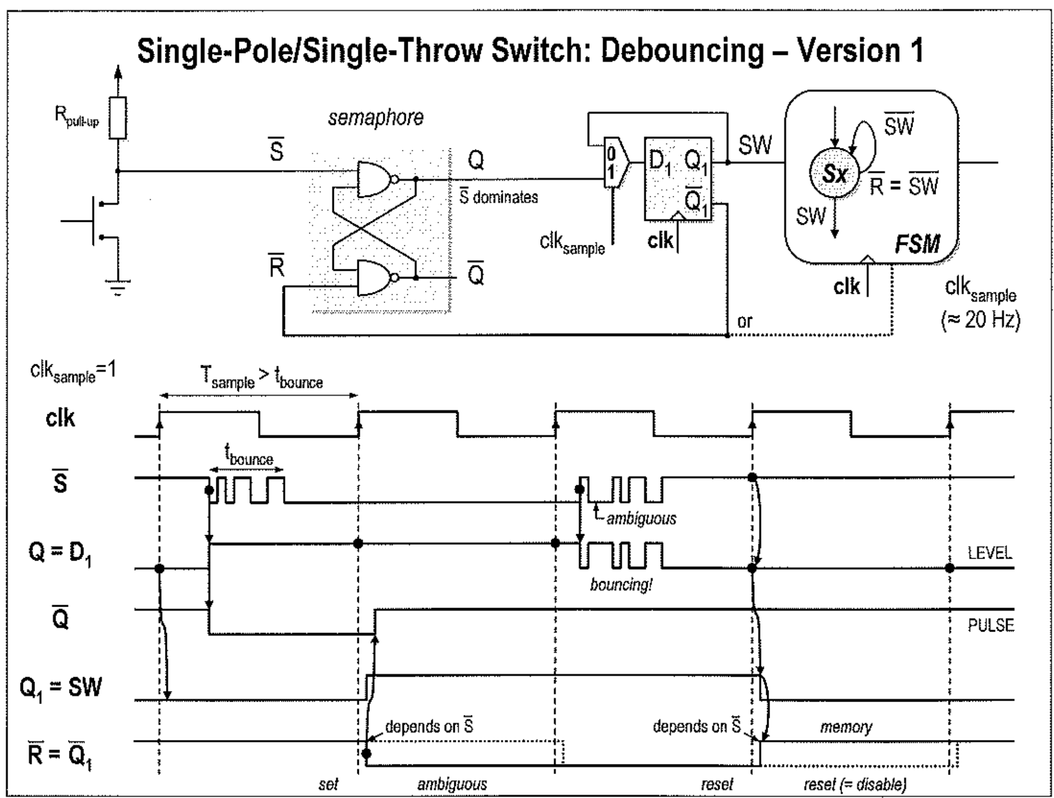
Examining cross-coupled NAND gate cell shows that the basic property for a sequential circuit is preserved, which is feedback.



A common problem in digital system design is to provide human interface to the system. The use of push-button switches is a typical example. Asynchronous input signals from push-button switches often produce a phenomenon called *contact- bounce* that derives from the mechanical structure of the switch and the physical nature of the contact surfaces. This contact-bounce is the result of the spring-like operation of the switch mechanism and the microscopic irregularity of the switch contacts. Multiple open/close transitions may occur immediately following the depression or release of a button switch, or any mechanical switch for that matter. It can take several milliseconds to die out. Serious problems can result in an FSM if a high-frequency clock catches the bounce signals produced by a mechanical switch. This is equivalent to the introduction of false data.

The single-pole/double-throw (SPDT) switch can be debounced very easily and precisely by using a SR-latch. The debouncing circuit for a SPDT switch is shown. Two pull-up resistors generate a logic one for the gates; the switch pulls one of the inputs to ground. Notice that when the switch button is in the up contact position the SR-latch is set, and when it is in the down contact position it is reset. Furthermore, in an off-contact position the SR-latch is forced to hold the previous output. What this means is that the first contact bounce to cross the switching threshold of the SR-latch on an up or down position of the switch will set or reset the SR-latch, respectively. All subsequent bounces are ignored. That is, any contact bounce that is produced following the first can do nothing but hold the SR-latch in either a set or reset condition.

The switch moves a rather long way between contacts. It may bounce around a bit, but will never bang all the way back to the other contact. Thus, the SR-latch's output is guaranteed bounce-free.



Shown is a simple normally closed single-pole/single-throw (SPST) mechanical switch, and the contact noise (bounce) that occurs as a result of opening or closing the switch.

Only one of the inputs of the SR-latch can be driven by the switch. When the switch is closed, the SR-latch is set and all subsequent bounces are ignored. But also the return of the switch to an open state will be ignored. Therefore the SR-latch must be re-armed on regular time intervals.

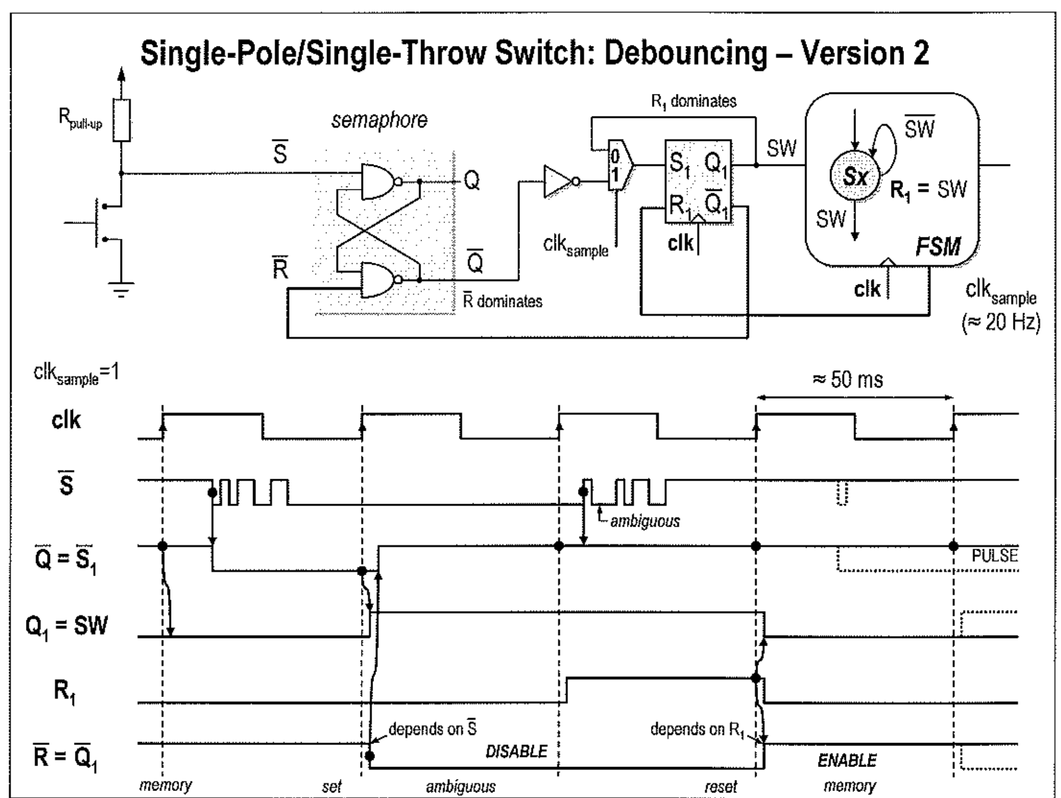
The maximum speed with which a mechanical switch can switch state is about f_{switch} = 10 Hz (100 ms period). Therefore, the state of the SR-latch is sampled on a frequency f_{sample} higher than f_{switch}, but lower than the oscillating frequency of the contact-bounce (typically more than 1 kHz) to prevent that the SR-latch is sampled more than once during the bouncing of the switch.

The output Q of the SR-latch is loaded in a synchronizing data flip-flop at the sampling rate f_{sample}. Once the state of the SR-latch is sampled, the input status can be processed by the FSM.

The SR-latch can be reset by the sampling flip-flop (this is the fastest way), when it is certain that the FSM will not 'miss' this status of the switch. When the FSM must perform an action, caused by the closing of the switch, it is needed that the FSM is in a state where a transition or an output depends on SW, the sampled status of the switch. When this is not the case, the SR-latch is kept in the set state until the FSM is in a state where it can react. In this state the FSM can also reset the SR-latch, as an acknowledge that the closing of the switch is detected and processed.

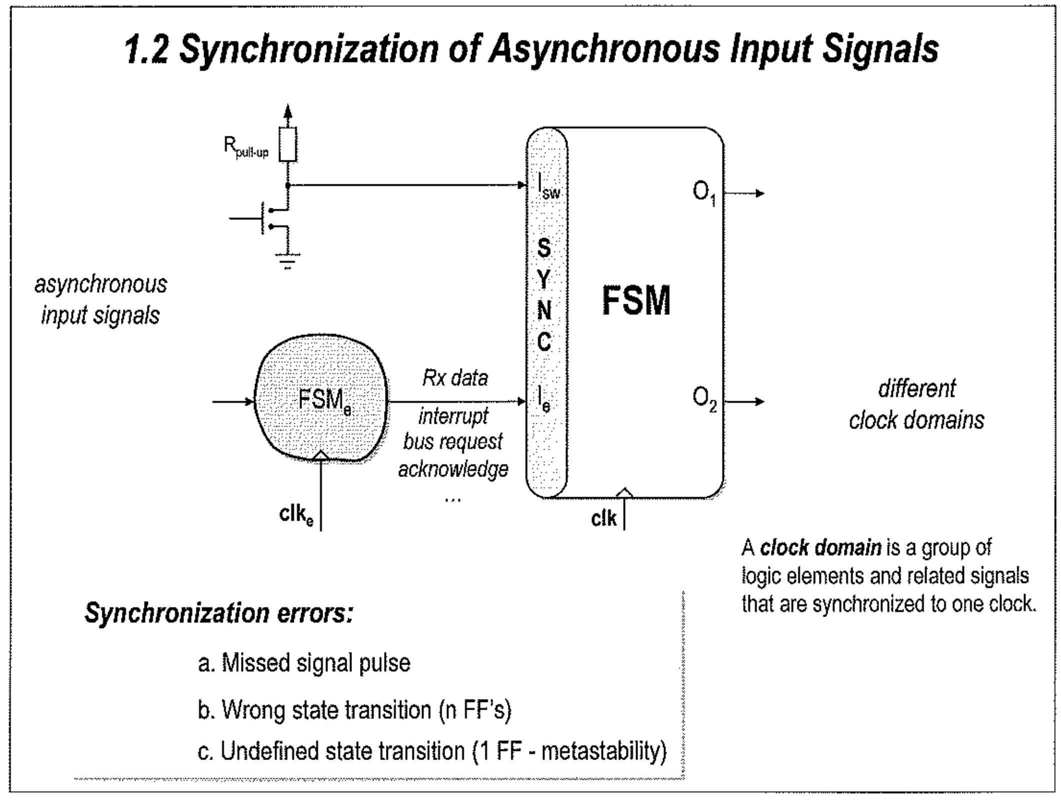
If the switch is still closed, while resetting, the SR-latch comes in the ambiguous state. In this case the bouncing, while opening the switch, can still be seen at the output of the SR-latch.

When the reset is released (after 1 sample clock cycle) the SR-latch is re-armed. A new closing condition of the switch can now be detected.



The status of the SR-latch can also be sampled on the Q' output, which is dominated by the value of the R' input. During reset (R'= 0) of the SR-latch, the Q' output remains high, independent of the value of the S' input.

To prevent that the FSM will 'miss' the detection of a closing of the switch, due to an early reset of the SR-latch, a SR flip-flop is used to 'catch' the closing state of the switch. This SR flip-flop remains high until it is reset by the FSM. In this way the switch and the FSM are synchronized to each other.



Asynchronous inputs don't have a timing relation with the system clock clk of the FSM.

Typical asynchronous inputs are: interrupt, bus request, transmission acknowledge, mechanical switch, proximity switch, analog sensor with DA converter, ... Also signals originating from other synchronous FSM's that operate on a different clock (other clock domain).

Asynchronous inputs can change logic level at any time, also in the sampling interval $[-T_S,T_H]$ around the active clock edge. Violation of the set-up and hold requirements are possible. Proper transitions cannot generally be guaranteed. Simply stated, a synchronous FSM may not function properly if asynchronous inputs are present.

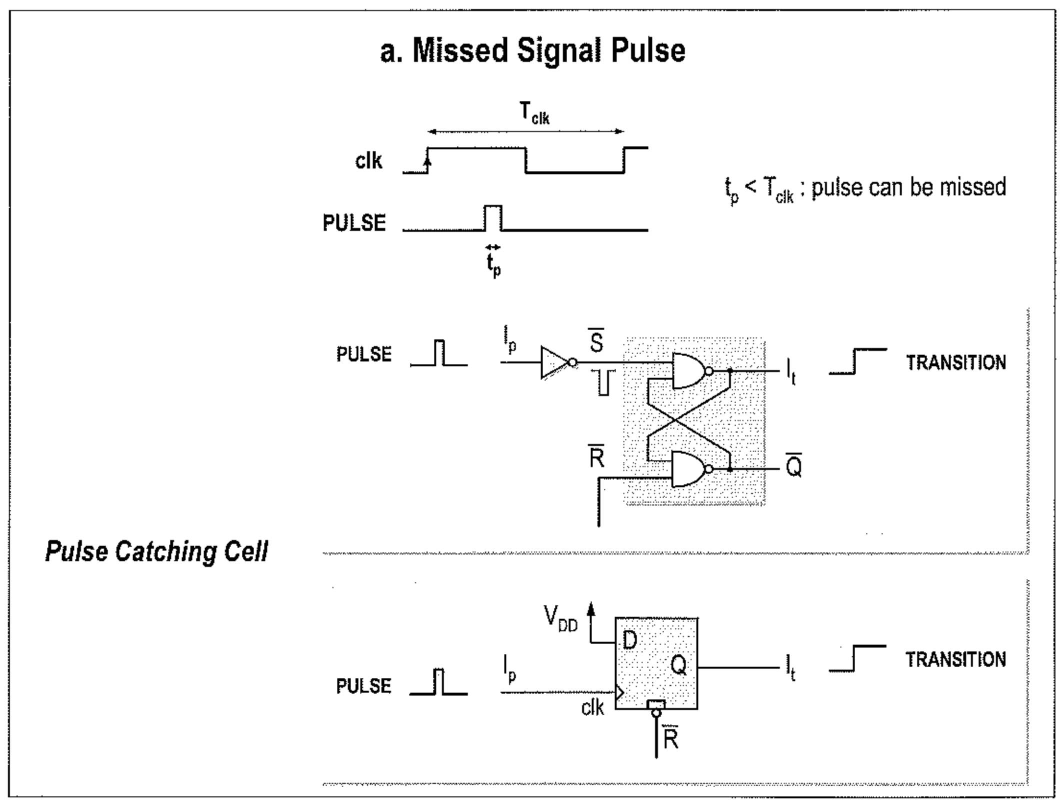
There three types of synchronization errors:

a. Missed signal pulse

b. Wrong state transition (change of multiple FFs)

c. Undefined state transition (single FF changes and becomes metastable)

Proper synchronization is needed to handle this synchronization problems.



Data can arrive as asynchronous pulses of duration t_p less than that of the clock period T_{clk}. The FSM must detect this pulses. It can be impractical to increase the frequency of the system clock to a frequency higher than $2/t_p$ (Nyquist sampling criterion).

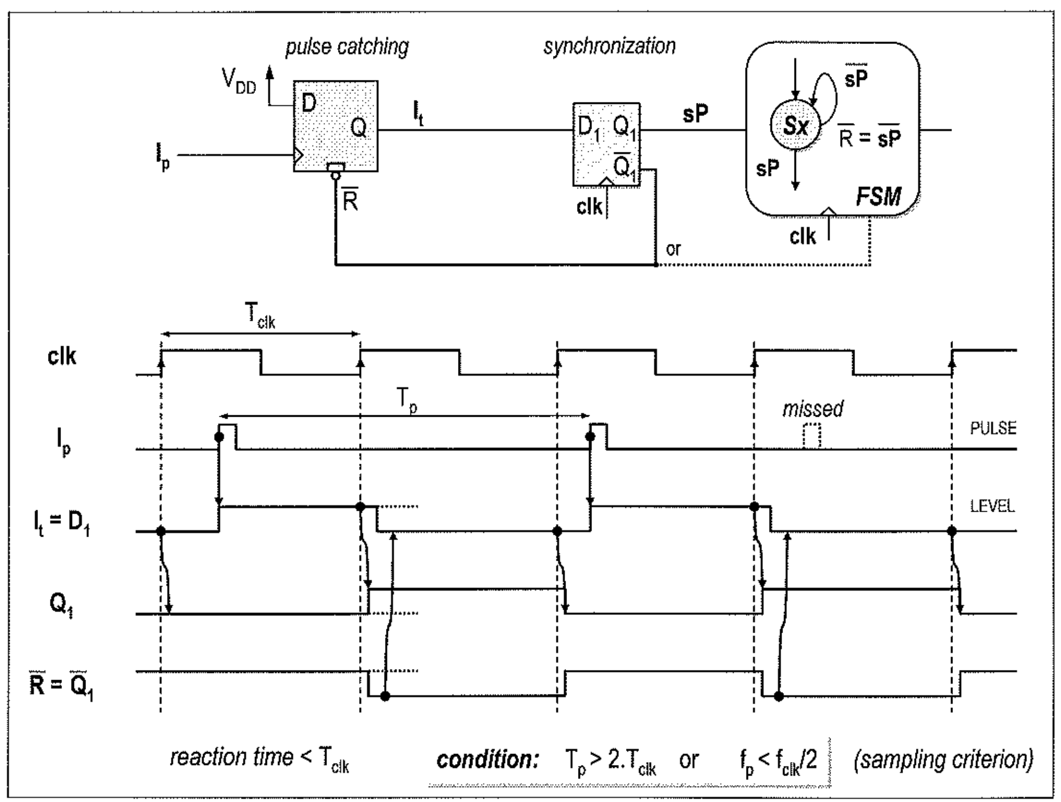
A means must be sought to stretch as well as synchronize the data signals. A scheme is needed to catch the input pulse and hold it until the FSM can service the input. If this can be done then the input is synchronized.

Two effective schemes for accomplishing the stretching/catching operation (pulse catching cell) are shown:

. set-dominant SR-latch: the pulse drives the set input and sets the SR-latch

. data flip-flop: the pulse drives the edge sensitive clock edge and sets the data flip-flop

The narrow asynchronous pulse is transformed into a transition. The change in level can then be detected on the next active clock edge of the synchronous system (synchronization).



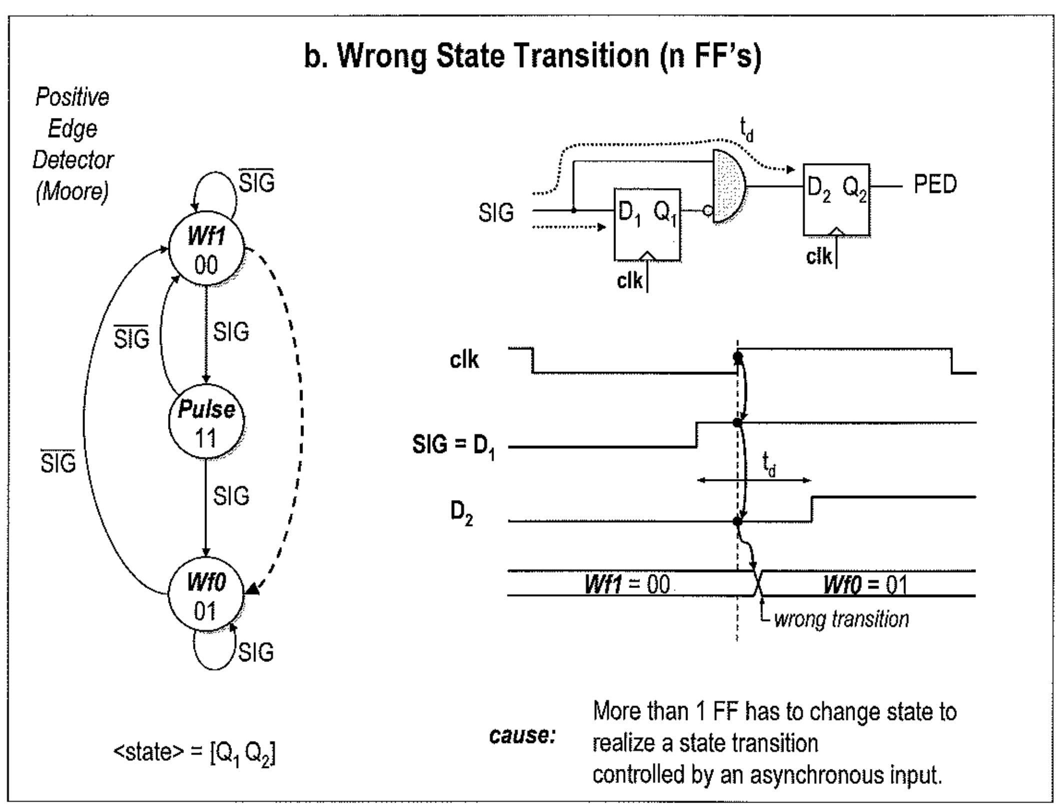
The actions of the pulse catching and synchronization are illustrated

The pulse is first transformed to a level by the pulse catching cell, then synchronized by the synchronizer (data flip-flop). The active low output of the synchronizer is fed back to the R' input of the pulse catching cell to reset it in readiness for the next narrow pulse. Notice that the output of the synchronizer, sP, is both stretched and synchronized, thereby providing a reliably detectable signal to the FSM regardless of the pulse duration. it is needed that the FSM is in a state where a transition or an output depends on sP, the synchronized level of the pulse. When this is not the case, the pulse catching cell is kept in the set state until the FSM is in a state where it can react. In this state the FSM can also reset the pulse catching cell, as an acknowledge that the pulse is detected and processed.

In the timing diagram, it is observed that not all narrow pulses can be caught by the synchronizer and presented to the FSM. If a second pulse appears during the reset of the pulse catching cell, it cannot be picked up by the pulse catching cell as a discrete data pulse. Consequently, a second narrow pulse having a leading edge separated by less than $2.T_{clk}$ from the leading edge of the first pulse cannot be guaranteed to be caught by the FSM, and a second leading pulse edge separated by less than T_{clk} from the first can never be caught. These limiting conditions are based on the assumption that the setup and hold times are negligibly small compared to the clock period, usually a valid assumption.

This condition is an application of the Nyquist sampling criterion:

$$T_p > 2.T_{clk}$$



If the next states of a single state whose branching is controlled by an asynchronous variable are not unit-distance coded, there exists a finite possibility that a wrong (undefined) transition to a state can be induced.

To illustrate the possible fault created by the improper state assignment to the next states of a state whose branching is controlled by an asynchronous input, the state assignment and resulting circuit of a positive edge detector (Moore FSM) are used.

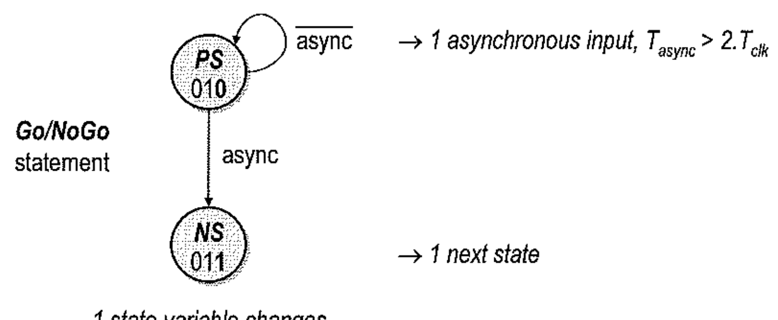
The signal SIG is an asynchronous input. A change of SIG before the active clock edge (a time larger than the set-up time of the data flip-flop) will be detected by data flip-flop Q_1. But due to the delay of the AND gate, the transition arrives at the input D_2 of the second data flip-flop after the active clock edge. This delay can cause an erroneous transition to state Wf0 rather than state Pulse if the input change is timed just right. This anomaly is catastrophic and one that is totally unacceptable. This is particularly true because of its intermittent occurrence.

The origin of the problem of wrong transition lies in the fact that the change of the asynchronous input must be detected by more than one flip-flop and that the combinatorial delay from the asynchronous input to the inputs of the different flip-flops are not equal.

FSM rules to prevent wrong state transitions

cause: synchronous inputs controlling state transitions

1. Only *1 asynchronous input* can control a conditional state transition.
2. Only *1 state variable* (=flip-flop) can change during a conditional state transition. Present state and next state are logically adjacent (unit-distance rule).
3. Only *1 next state.* Only 1 flip-flop can change, independent of the level of the input.
4. The *active level* of the asynchronous signal must last at least *2 clock periods.*



Wrong state transitions due to asynchronous inputs can be prevented by the following state assignment rules:

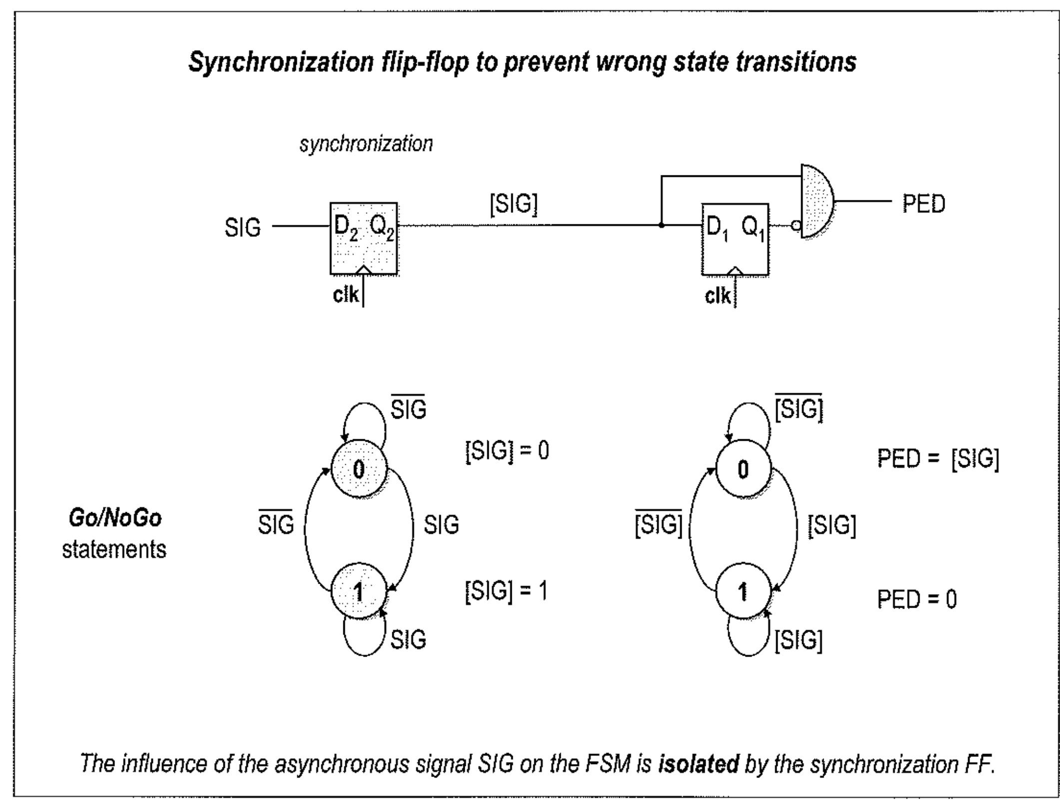
1. The branching conditions for any state should never be controlled by more than one asynchronous input.

2. The next states should be given unit-distance assignments, that is only one state variable should be affected, implying that only one flip-flop in the present state register should be scheduled to change on a conditional asynchronous input variable. So, the next state from a single state whose branching is controlled by an asynchronous input must be given unit-distance state assignments.

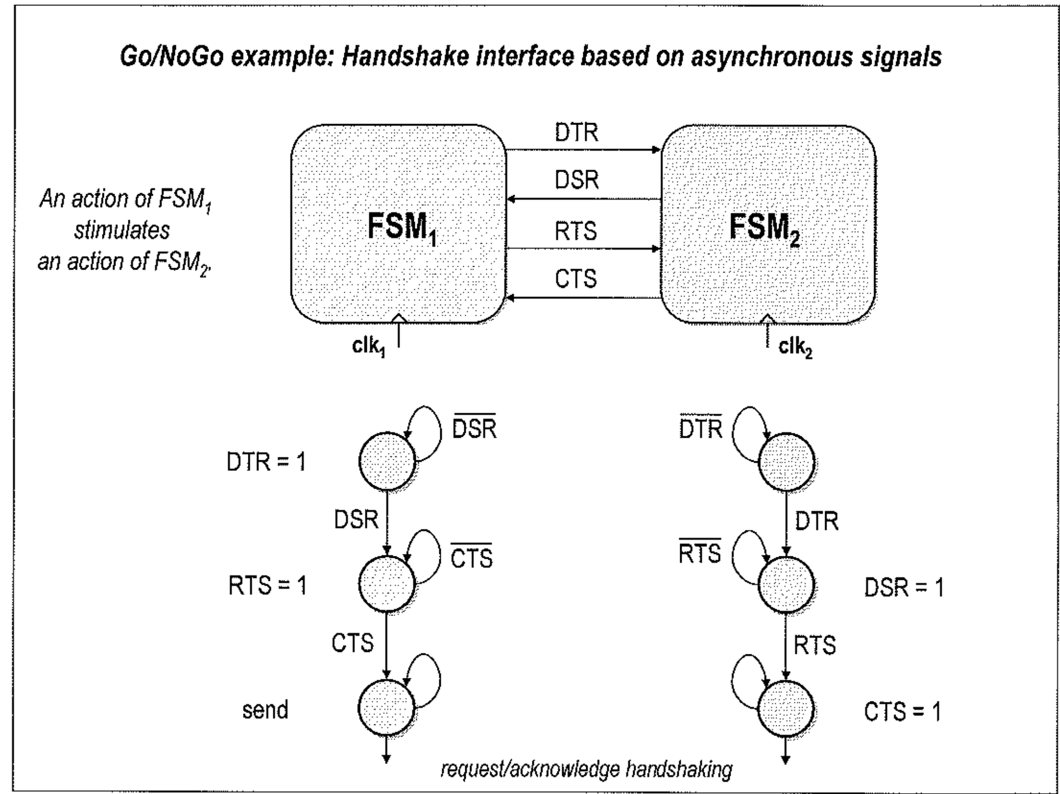
3. Only one next state is allowed. Avoid multi-way branches, because the change of the asynchronous input must than be propagated to more than one flip-flop. So, there are only two transitions allowed: a loop back transition onto the same state, and one next state with a unit-distance assignment.

4. To prevent the 'missing' of an asynchronous pulse, the active level must last at least 2 clock periods.

These rules result in a go/nogo statement, which is the only acceptable state diagram construction that will not result in an anomalous transition. The FSM is designed to remain in state PS until the asynchronous input is definitely found asserted. Then and only then will the transition to state NS be effected. If one edge of the clock missed the transition of the asynchronous input, the next edge will catch it and it then will be processed by the FSM.



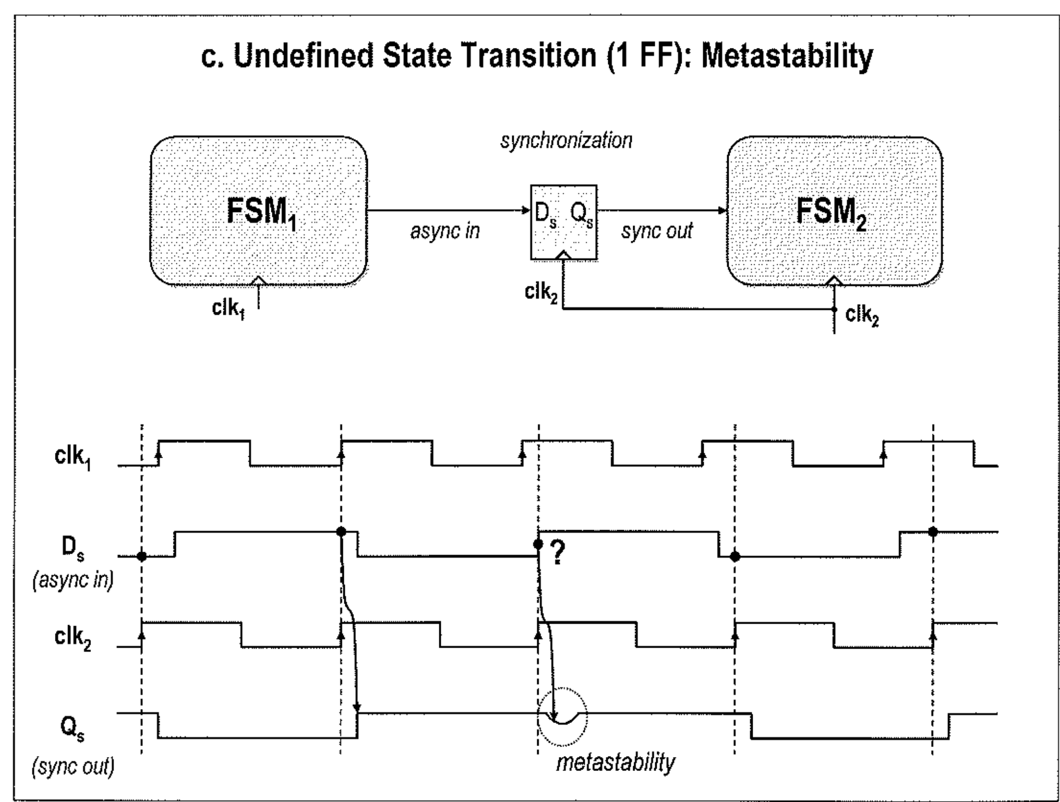
Synchronizing the asynchronous input signal with a data flip-flop automatically results in an implementation of a go/nogo statement. Also the FSM that uses the output of the synchronized signal [SIG] uses a go/nogo statement to handle the signal.



To partly eliminate the potential problems related to asynchronous inputs, the handshake interface technique has been developed. A handshake implementation is a technique whereby some action is taken by one party (system) that serves to stimulate some action by the other party (other system). This second party action is to signal the first party that the initial action was acknowledged and that the next transmission can now proceed. This handshake concept is symbolized by a sample block diagram and a section of a state diagram shown in the figure.

FSM_1 moves into a state and issues DTR and waits for the FSM_2 to return with DSR. The FSM_2 is slaved to the FSM_1 and responds accordingly.

Thus the handshake is a sort of mutual agreement to synchronize, but nonetheless the two systems still must treat the other's input as an asynchronous input. By using this scheme and by making a proper state assignment (go/nogo statement), the erratic behavior can be avoided.



Metastable is a Greek word meaning "in between." Metastability is an undesirable output condition of a flip-flop caused by inputs violating the flip-flops' minimum set-up and hold times.

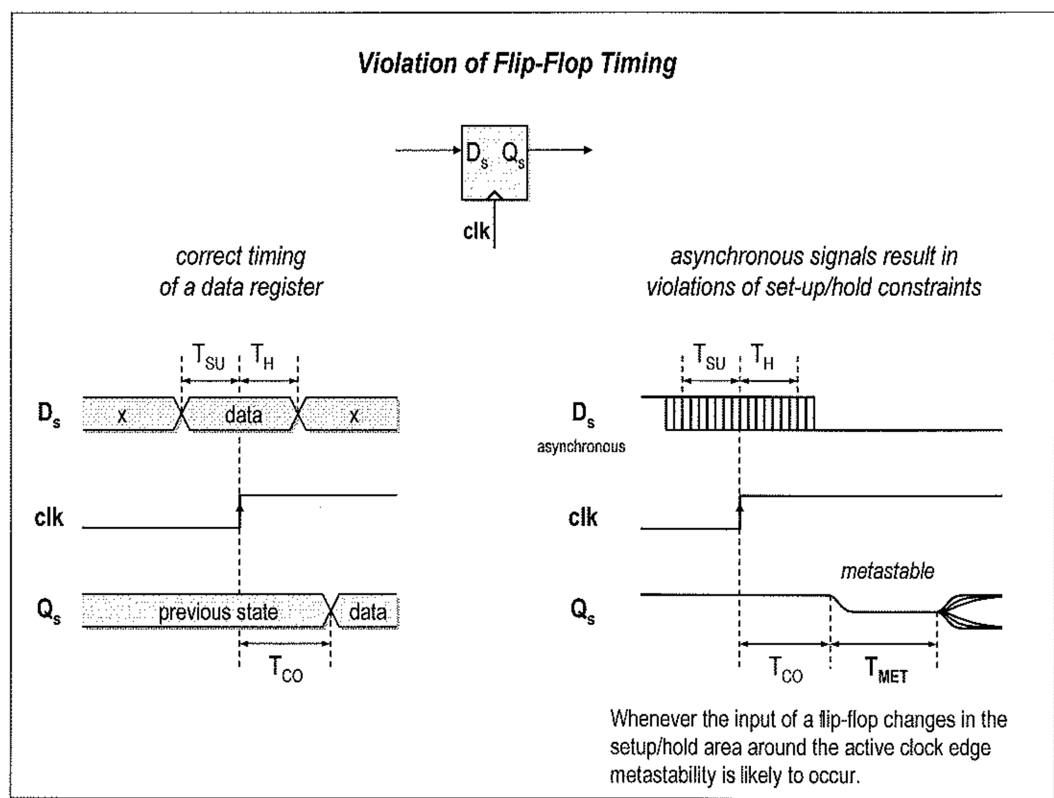
Metastability is seen as an output voltage level in the area between a logic HIGH and a logic LOW. Although systems have been designed that did not account for metastability, its effects have taken their toll on many of those systems.

In most digital systems, violation of the input set-up and hold times of storage elements does not occur. These systems are designed as synchronous systems that meet or exceed their components' worst-case specifications. Totally synchronous design is not possible for systems that impose no fixed relationship between input signals and the local system dock. This includes systems with asynchronous bus arbitration signal, telecommunications data, I/O interfaces like buttons or switches, and anything not synchronized to the system clock. Asynchronous means that the signal is generated using another clock signal. Although the external signal might be synchronous to another clock signal it is an asynchronous event for this clock domain (clk_2).

For digital systems to function properly, it is necessary to synchronize the incoming asynchronous signals with the local system dock before using them. But, this asynchronous inputs will inevitably cause input setup or input hold violations of the flip-flop using this signal as input.

A simple synchronizing data flip-flop is shown. The asynchronous input (D_s) comes from outside the local system. The synchronizer operates with a system dock clk_2 of the local system. On each rising edge of this system dock, the synchronizer attempts to capture the state of the asynchronous input. Most of the time, this synchronizer performs as desired.

Digital systems are supposed to function properly all the time, however. But because there is no direct relationship between the asynchronous input (D_s) and the system dock (clk_2), at some point the two signals will both be in transition at very nearly the same instant. When this input condition occurs the synchronizer can become metastable. The output might later transition to a HIGH or a LOW. But the synchronizer will not make a decision one way or the other in its specified clock-to-output time.



When other components in the local system sample the synchronizer's metastable output, they might also become metastable. A potentially worse problem can occur if two or more components sample the metastable signal and yield different results. This situation can easily corrupt data or cause a system failure.

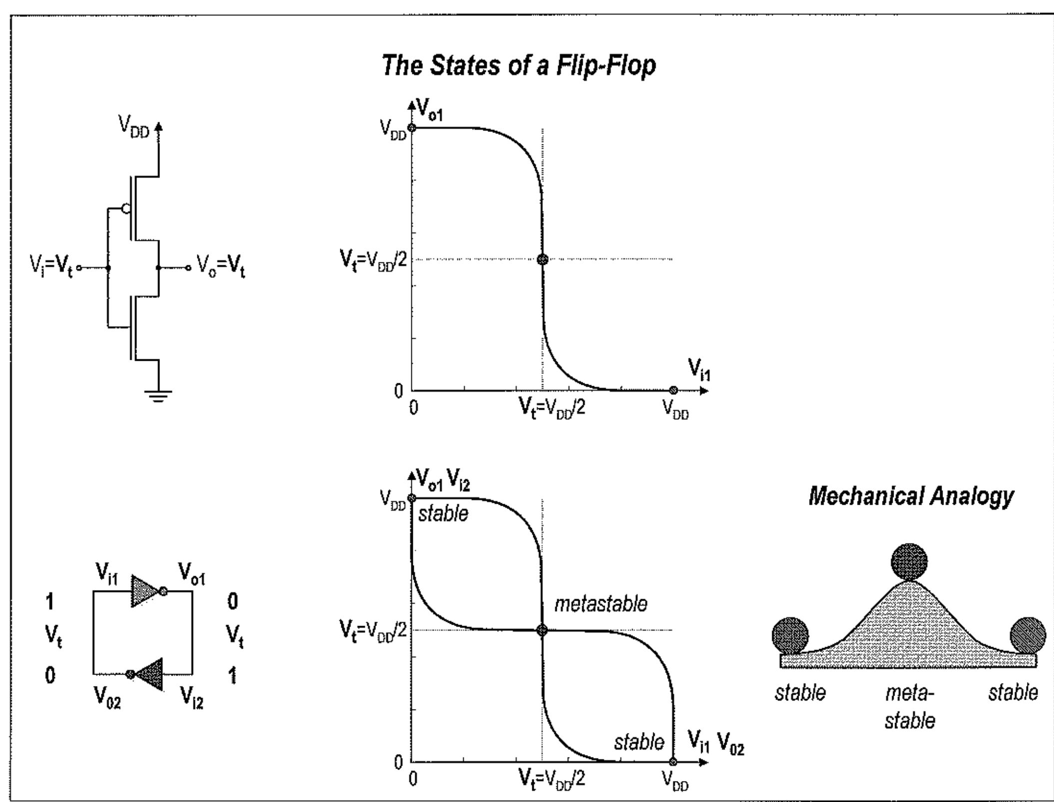
As system speeds increase and as more systems utilize inputs from asynchronous external sources, metastability-induced failures become an increasingly significant portion of the total possible system failures. So far, no known method totally eliminates the possibility of metastability. However design techniques can be used that make its probability relatively small compared with other failure modes.

Asynchronous inputs, like buttons, switches, and anything not synchronized to the system clock will inevitably cause input setup or input hold violations of the flip-flop using this signal as input..

In a flip-flop, a metastable output is undefined with a level between HIGH and LOW for an indefinite time. This anomalous flip-flop behavior results when data inputs violate the specified set-up and hold times with respect to the clock.

In the case of a D-type flip-flop, the data must be stable at the device's D input before the clock edge by a time known as the set-up time, T_{SU}. This data must remain stable after the clock edge by a time known as the hold time, T_H. The data signal must satisfy both the set-up and hold times to ensure that the storage device (register, flip-flop) stores valid data and to ensure that the outputs present valid data after a maximum specified clock-to-output delay T_{CO}. This T_{CO} refers to the interval from the clock's rising edge to the time the data is valid on the outputs.

If the data violates either the set-up or hold specifications, the flip-flop output might go to an anomalous state for a time greater than T_{CO}. The additional time it takes the outputs to reach a valid level can range from a few hundred picoseconds to microseconds. The amount of additional time beyond T_{CO} required for the outputs to reach a valid logic level is known as the metastable walk-out time T_{MET}. This walk-out time, while statistically predictable, is not deterministic.



A flip-flop, like any other bistable device, has two minimum-potential energy levels, separated by a maximum-energy potential.

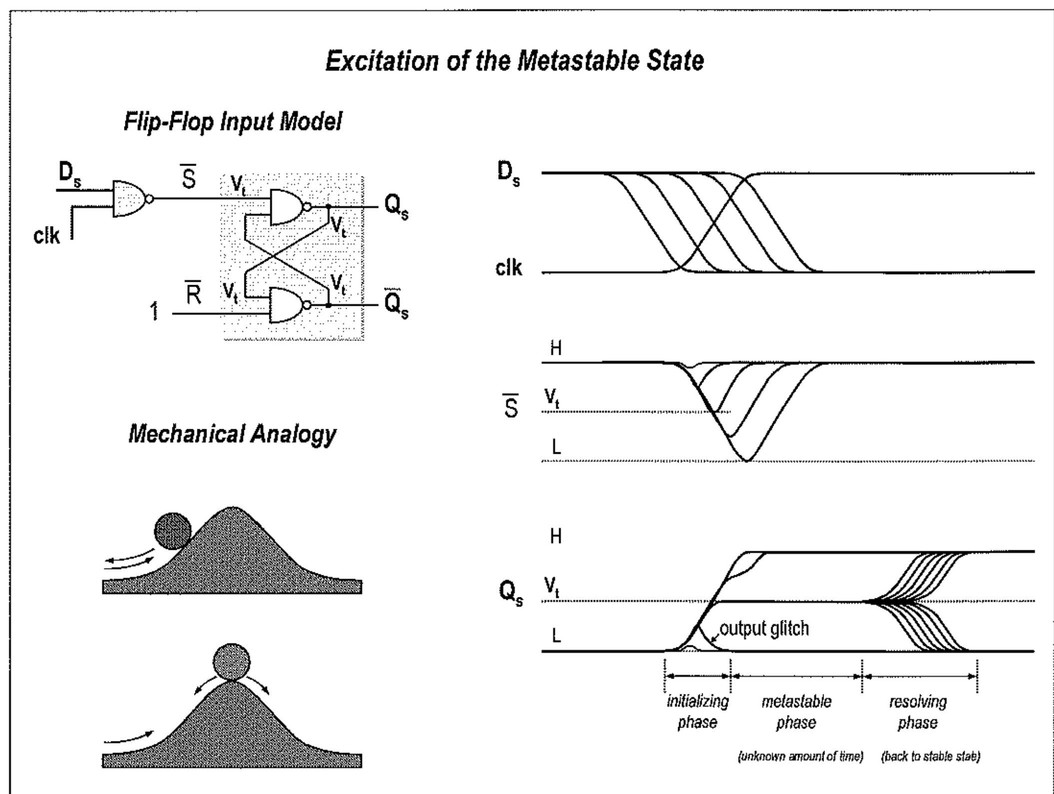
A typical static memory element contains two cross-coupled inverters. A logic 1 at the input of the first inverter becomes a 0 at its output. The 0 is mapped to a logic 1 at the output of the second stage, which reinforces the value at the first inverter's input. A similar argument holds for a 0 at the input. Once a value (0 or 1) is inserted at the input, it can be held indefinitely by the circuit. This are the two stable states of the memory element.

A bistable system has stability at either of the two minimum-energy points. The system can also have temporary stability, metastability, at the energy maximum. This state can roughly be defined as 'half set and half reset'. If nothing pushes the system from the maximum-energy point, the system remains at this point indefinitely.

For the two cross-coupled inverters, when a value $V_t = V_{DD}/2$ is presented at the input of the first inverter this becomes V_t at its output. The V_t is mapped to V_t at the output of the second stage, which reinforces the value at the first inverter's input. So, both outputs are at the threshold level. Because the inverters act as high gain amplifiers around the metastability voltage V_t, internal noise, thermal disturbances, asymmetries will be amplified and bring the memory element back to a stable state.

Mechanical Analogy

A hill with valleys on either side is another bistable system. A ball placed on top of the hill tends to roll toward one of the minimum-energy levels. If left undisturbed at the top, the ball can remain there for an indeterminate amount of time. As this figure indicates, the characteristics of the top of the hill as well as natural factors affect how long the ball stays there. The steepness of the hill is analogous to the gain-bandwidth product of the flip-flop's input stage.

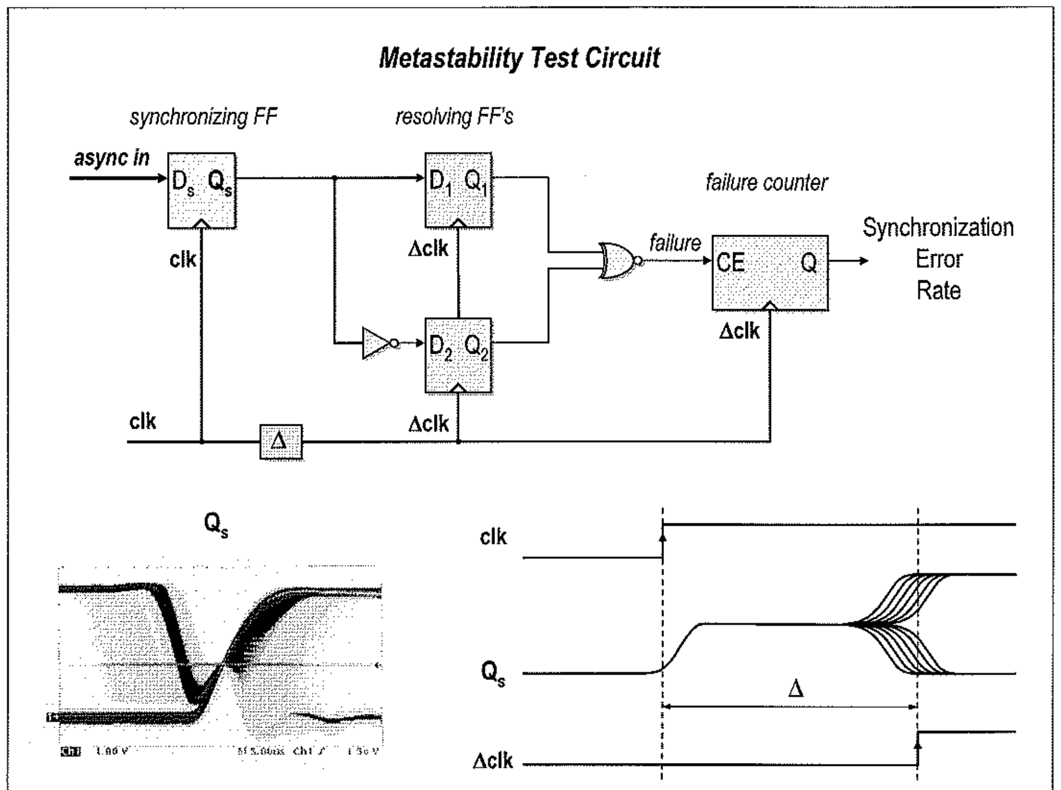


The excitation of the metastable state of a flip-flop can be explained by the simple input model of a flip-flop as shown.

When the data input changes to a low value at a the rising (active edge) of the clock, there exist a finite probability that the input nand-gate outputs a runt pulse (reduced pulse), with an amplitude depending on the relative timing of the data and clock signals. When the amplitude of the runt pulse equals V_t ($V_{DD}/2$ for CMOS), this is the threshold voltage of the cross-coupled nand-gates, then the flip-flop can enter the metastable state. This is called the *initializing phase.* The Q_s and $\overline{Q_s}$ outputs move simultaneously from their existing level to the metastable voltage V_t.

Once a flip-flop has entered the *metastable phase*, the device can stay metastable for an indeterminate length of time (statistical period of time that cannot be predicted). The probability that the flip-flop will stay metastable for an unusually long period of time is zero, however, due to factors such as noise, temperature imbalance within the chip, transistor differences, and variance in input timing. The output remains in metastable state for an unknown amount of time. During this second phase of metastability, for very small deviations around the metastable voltage, V_t, the flip-flop behaves like two cross-coupled amplifier stages. The gain of the cross-coupled loop exceeds unity.

The *resolving phase* occurs when the outputs once again drift toward stable voltages. Noise or imbalance is amplified in the cross-coupled loop and increases exponentially with time. The length of time the flip-flop takes to resolve cannot be exactly determined. The probability that the flip-flop will resolve within a specific length of time, however, can be predicted. This probability depends on the electrical parameters of the flip-flop acting as an amplifier around the metastability voltage. However, it cannot be predicted which logic level (set or reset) will emerge following exit from the metastable state.



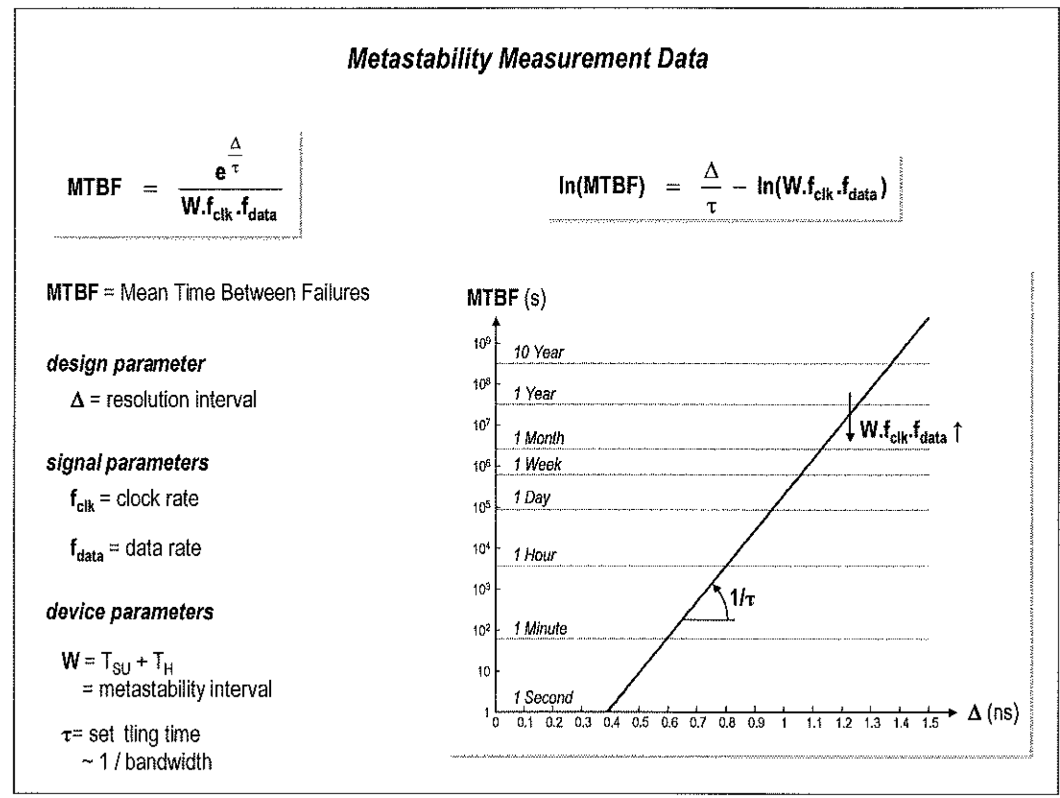
A typical metastability test circuit is shown.

The asynchronous input is first sampled by the synchronizing flip-flop. Then two resolving flip-flops look at he output of the synchronizing flip-flop a time Δ later. The inverter amplifies the level of Q_s to a stable value. During normal operation, the exclusive-NOR on these outputs produces a LOW level. This indicates either that metastability has not occurred within the device or that metastability that has occurred has resolved before the sampling instance.

If a metastable event cannot resolve before the next sampling moment, the exclusive-NOR produces a HIGH level. In this case, the resolving flip-flops have interpreted the signal from the synchronization register differently, indicating that unresolved metastability has occurred.

This test circuit does not catch all metastable events. Specifically, it does not record metastable events that resolve before the next sampling moment. But metastability causes an error only when it has not resolved by the time the signal is needed. The tests thus reveal the information designers need to know: how often metastability creates an error in the system.

When the clock is common to all flip-flops ($\Delta=T_{clk}$), the test circuit also includes the ability to check the maximum operating frequency of the flip-flop under test. At each dock edge, the first toggle flip-flop's output toggles. When the device reaches its maximum operating frequency, the resolving flip-flops cannot resolve the changing signal fast enough to produce a valid output. At this speed, one register might resolve the signal correctly and one might not, or both might produce invalid signal resolutions. In any case, when the exclusive-NOR of the two resolving registers results in a LOW, the system's maximum operating frequency is exceeded.



The MTBF (Mean Time Between Failures) value shows how metastability affects the reliability of a design dealing with asynchronous signals.

The MTBF formula provides an estimate of the mean time between the probable occurrence of two successive metastable events for a synchronizing flip-flop.

Δ = time delay after the active clock edge for the metastability to resolve itself, the additional time allowed by the system for the flip-flop to settle to a stable state

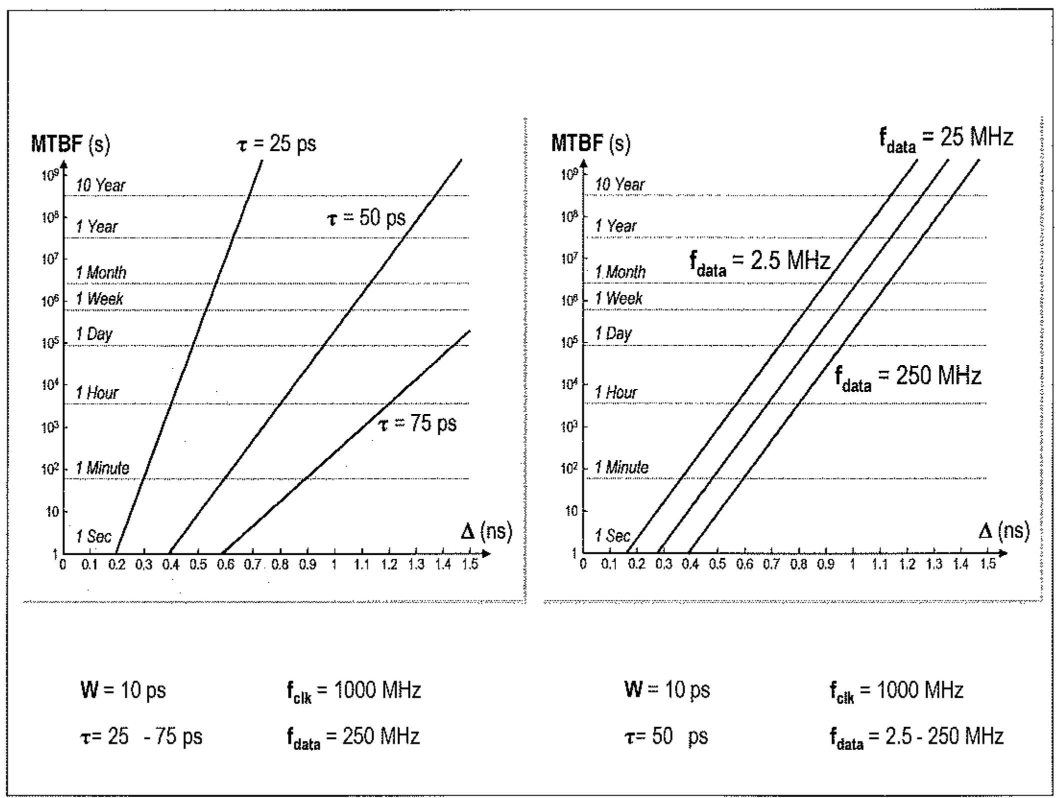
f_{clk} = clocking frequency of the synchronizing flip-flop

f_{data} = data frequency

W = constant representing the metastability catching window (setup time before and hold time after the active clock edge)

τ = the settling time with which the speed with which the metastable condition is resolved

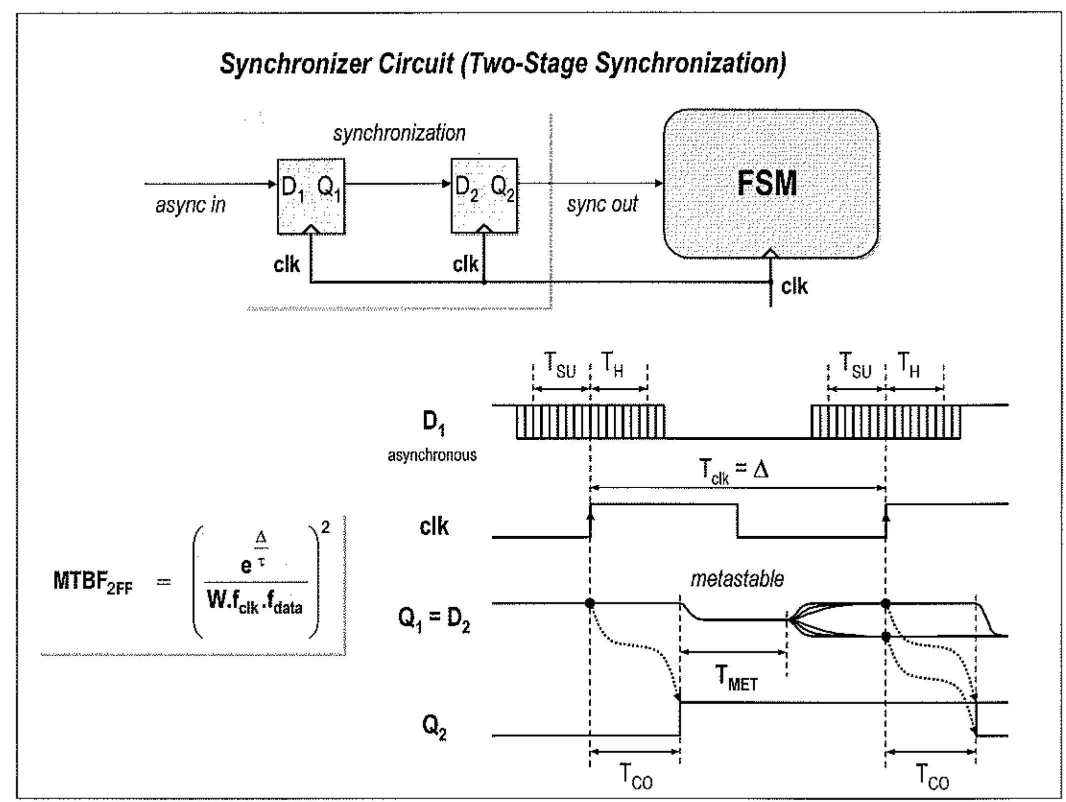
The MTBF equation is usually plotted with Δ (the resolving time allowed for metastable events) on the X axis and the natural log of the MTBF plotted on the Y axis. Because the metastability equation is plotted on a semi-log scale, the graph of Δ vs ln(MTBF) becomes a line, with slope $1/\tau$.



There is no cure for metastability. What you can do is trade latency of your system for higher MTBF. The MTBF is in an exponential way sensitive to the resolving time Δ. The reliability of the synchronization process can be increased dramatically by increasing this resolving time. This gives the synchronizing flip-flop more time to make its decision and enter a stable state.

Nothing improves the MTBF of a metastable synchronizer better than just waiting longer. Not clocking the intermediate signal on the negative clock edge. Not voting. Not threshold testing. Not adding noise. Not predicting circuits. Not circuits designed to bias the outcome to either 1 or 0. Not clocking it twice as fast through twice as many flip flops. Nothing.

Reducing the clock rate and the data rate of the asynchronous input result in a proportional decrease of the MTBF.



An accepted solution to handle asynchronous input signals is the concatenation of an additional flip-flop after the first synchronizing flip-flop. (The output of the first synchronization register is connected to the input of the second synchronization register.) This added flip-flop does not totally remove the problem but does improve reliability. This solution is still in wide use today.

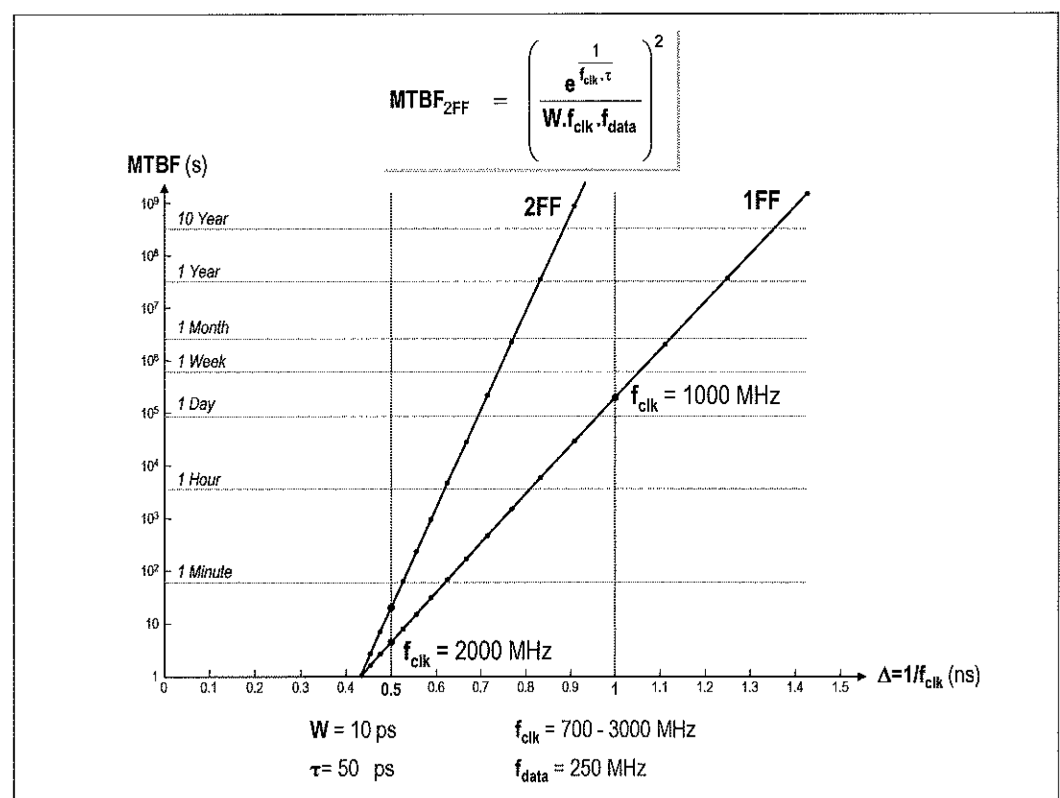
Recovery from metastability is probabilistic. In the improved synchronizer, the first flip-flop's output might still be in a metastable state at the end of the sample clock period. Because the flip-flops are sequential, the probability of propagating a metastable condition from the second flip-flop stage is the square of the probability of the first flip-flop remaining metastable for its sample clock period (product of probabilities: both flip-flops must become metastable). This type of synchronizer does have the drawback of adding one clock cycle of latency, which might be unacceptable in some systems.

When implementing a two-stage synchronizer, the probability that a synchronizer is metastable after the second stage of synchronization is the square of the probability that a synchronizer is metastable after the first stage of synchronization.

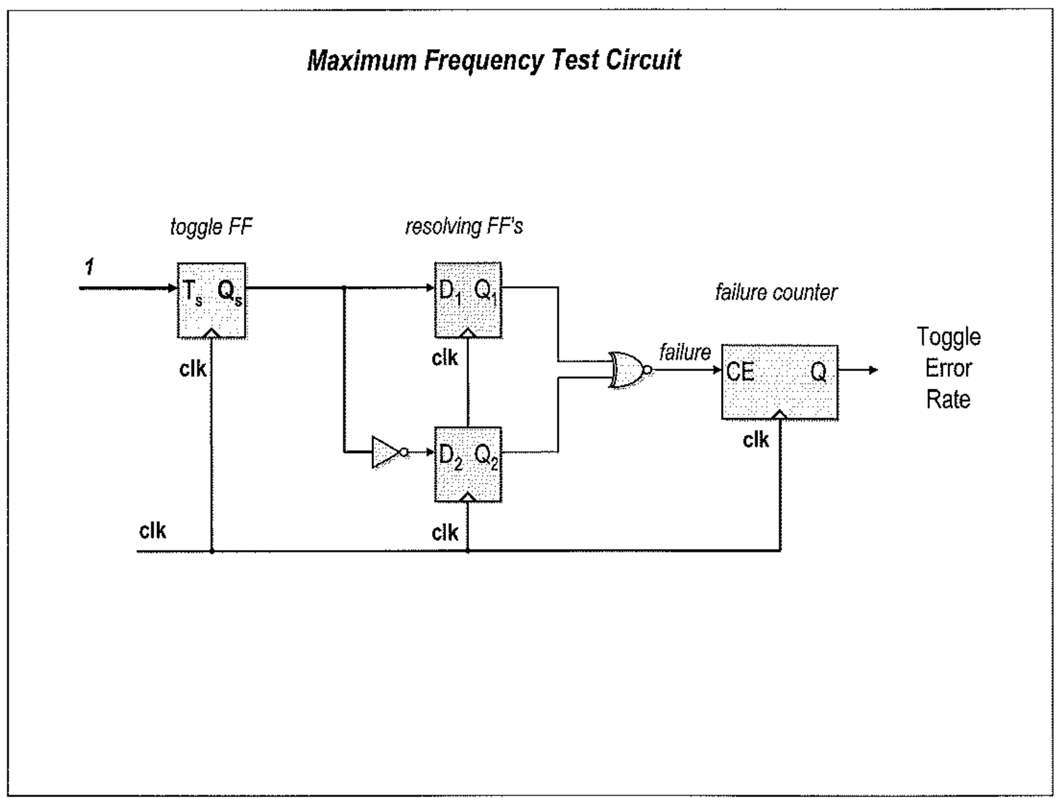
It must be remembered that because metastability is a statistical phenomenon and is unpredictable, no synchronizer 'fix-it' scheme can be devised that will eliminate entirely the possible occurrence of the metastable state. All that can be done is to reduce the probability for metastability occurrence to acceptable levels for a given application.

Evaluation:

- Synchronizer circuits add delay (latency).
- Synchronizer circuits are not perfect guarantees.
  - With MTBF calculations the time between synchronization failures can be determined.
- When a signal comes on-chip, synchronize it once and then fan signal out as required.
  - Do not fan out and then synchronize at multiple places.
  - Variations in timing can create different results.

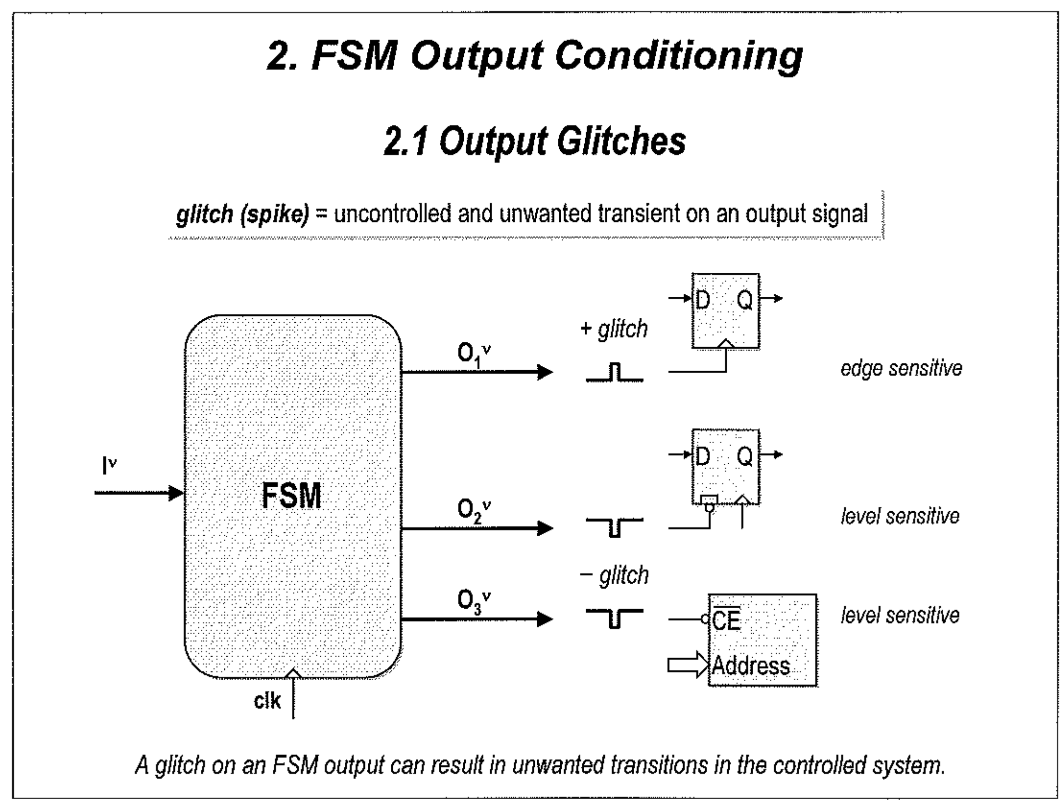


When implementing a two-stage synchronizer, the probability that a synchronizer is metastable after the second stage of synchronization is the square of the probability that a synchronizer is metastable after the first stage of synchronization.



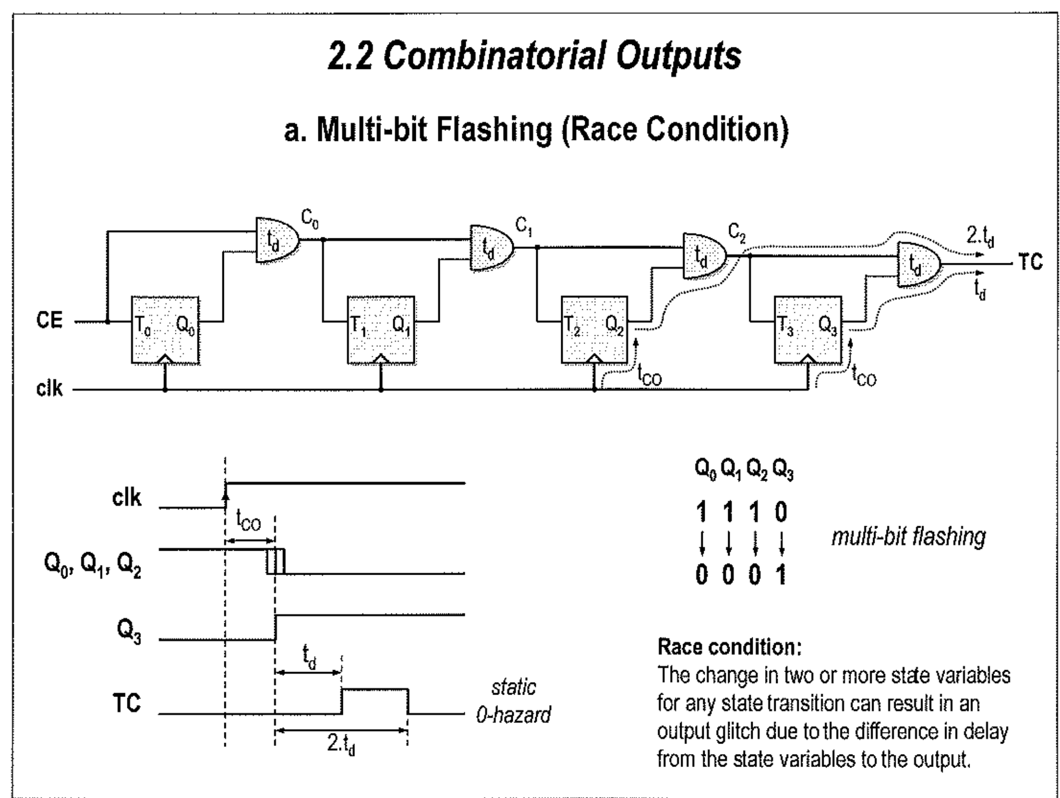
A typical maximum frequency test circuit is shown. It is similar to the metastability test circuit.

When the clock is common to all flip-flops ($\Delta=T_{clk}$), the test circuit also includes the ability to check the maximum operating frequency of the flip-flop under test. At each dock edge, the first toggle flip-flop's output toggles. When the device reaches its maximum operating frequency, the resolving flip-flops cannot resolve the changing signal fast enough to produce a valid output. At this speed, one register might resolve the signal correctly and one might not, or both might produce invalid signal resolutions. In any case, when the exclusive-NOR of the two resolving registers results in a LOW, the system's maximum operating frequency is exceeded



A 'glitch' on an FSM output is an unwanted 'spike' or transient that drives asynchronous control signals of the controlled system: it clocks or resets a flip-flop, or enables during a short time an IC, for instance a memory chip, and results in unwanted actions in the system that is controlled by the FSM.

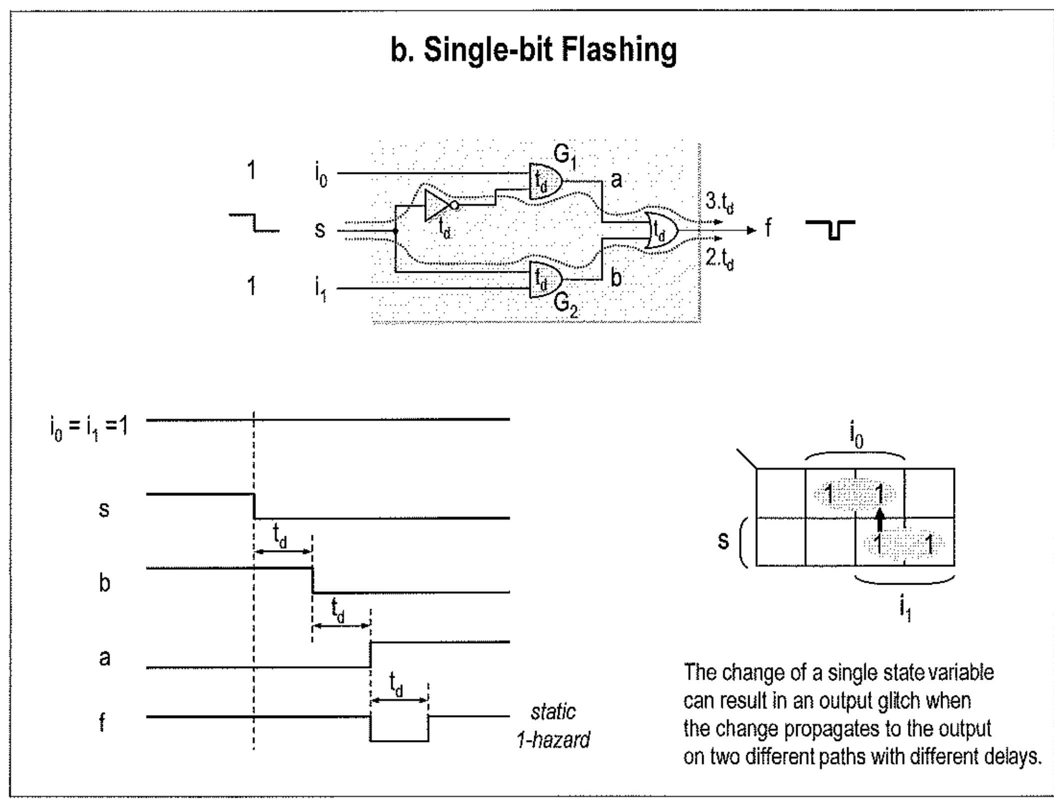
In this cases it is important that the FSM is designed to issue signals free of glitches.



A glitch can occur as a result of two or more state variable changes during a state-to-state transition (multi-bit flashing). But no two actions happen simultaneously as noted by an extension of the Heisenberg Uncertainty Principle, which says that even though a bank of flip-flops are all triggered from the same clock, they will not all change state simultaneously but in fact will change one by one. One might have to measure the time interval between these changes in ever-increasing smaller increments, as set forth by the Heisenberg Uncertainty Principle, but rest assured that no two actions happen exactly in coincidence.

Even if the state variables should change simultaneously, the difference in the combinatorial propagation path to the output can result in output glitches. This is illustrated for a binary counter with a serial carry. The change of the most significant bit will be seen first on the TC output, because it has the smallest delay.

Thus decoding the present state variables (outputs of flip-flops) using a combinational decoder can cause transient outputs unless the decoder is specifically designed to suppress the glitch. One glitch suppression method disables the complete output decoder just prior to a state change and maintains this disabled condition for some prescribed period of time after the state change, allowing the transient conditions to settle out. This is typically used in address decoders, that use a strobe signal that indicates after a change of the address bits when they are valid.

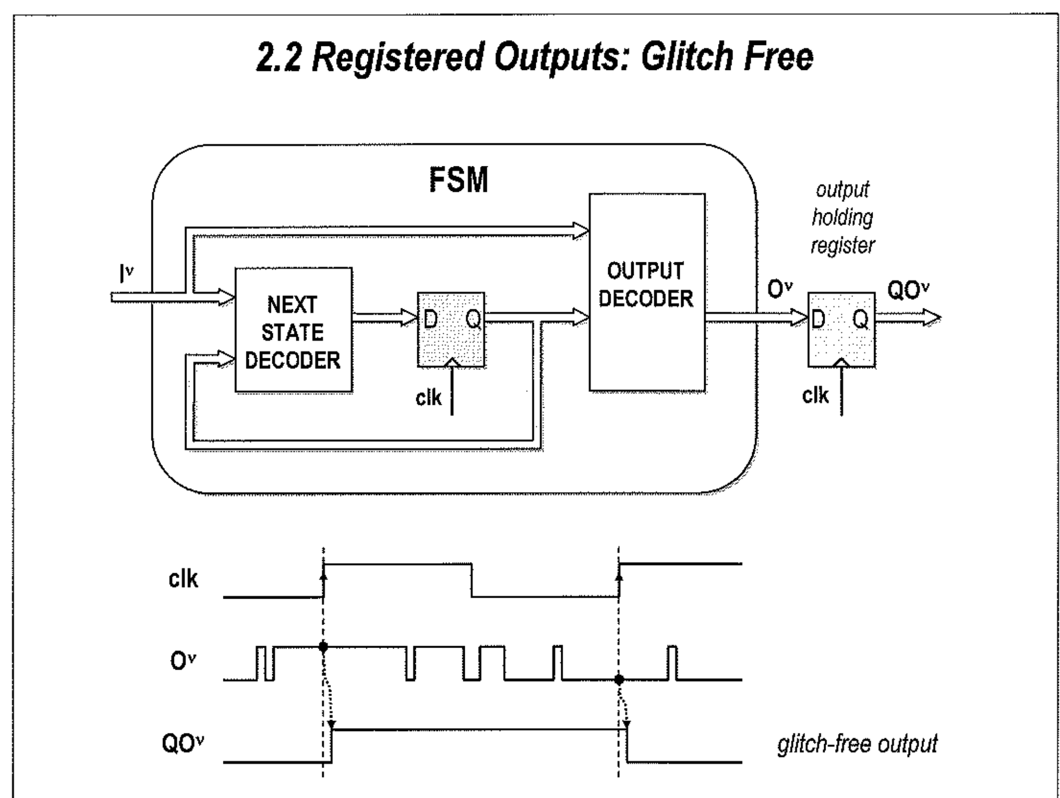


Even if only one state variable changes, a glitch can occur on the output as a result of two or more paths from that variable to the output with different delays.

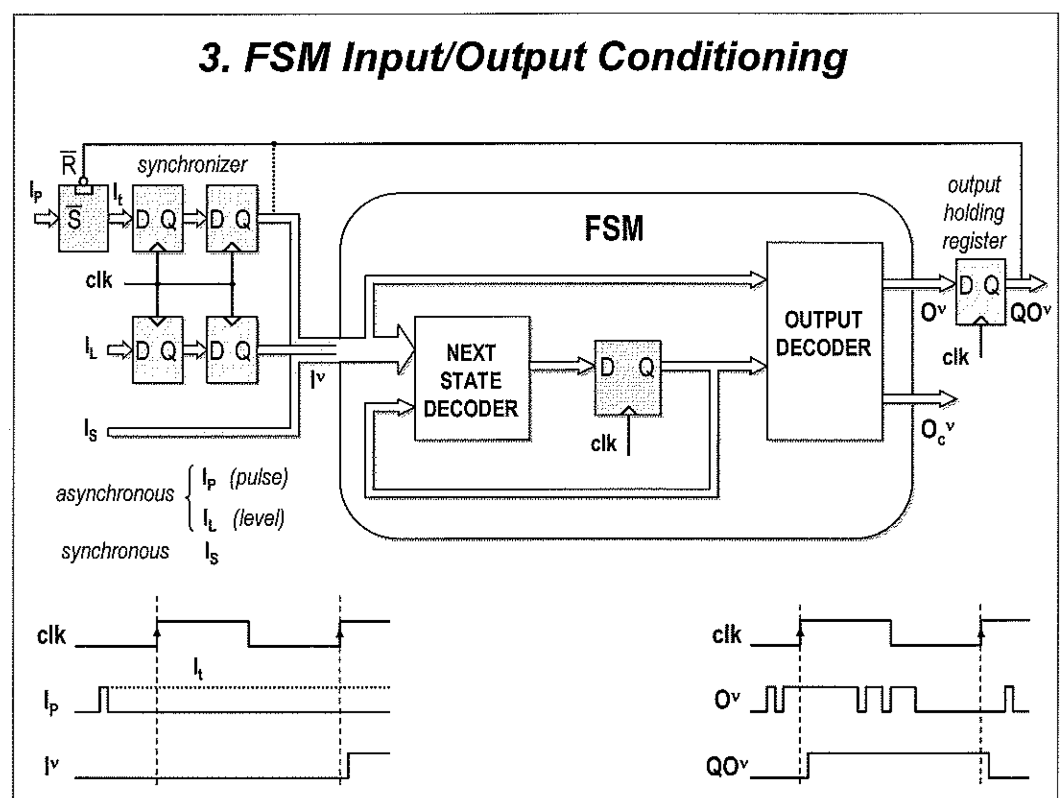
This is illustrated for a multiplexer. The two data inputs i_0 and i_1 are given a high level (1). Changing the selection input s from 1 to 0, should not change the output f. This changes is propagated to the output along two paths with a different combinatorial delay. This results in a static-1-hazard. This is a glitch that occurs in an otherwise steady-state 1 output signal.

More generally, a combinatorial circuit has a static-1-hazard, if there are two product terms (AND gates) that differ in only one variable, but are not covered by a common product term in a sum-of-products (SOP) implementation. This variable has two asymmetric paths to the output.

This hazard can be eliminated by including an additional prime implicant (minterm) in the SOP form.



As indicated in the figure, data flip-flops (output holding register) eliminate the glitch by loading and holding (sampling) the outputs of the output decoder after each active clock edge. Further, by making use of the holding capability of the data flip-flops, those outputs that must remain asserted for multiple consecutive states also remain glitch-free. The timing diagram illustrates the typical timing considerations related to the use of an output holding register. An extra delay is introduced.



The interfacing of an FSM with the external world (inputs and outputs) must be considered carefully. Additional registers can solve or mitigate several interfacing problems.

INPUTS:

. asynchronous inputs with levels that last longer than one clock period (I_L)

→ a synchronizer (cascade of 2 data flip-flops) reduces the probability of metastability

. asynchronous inputs with pulses shorter than one clock period (I_P)

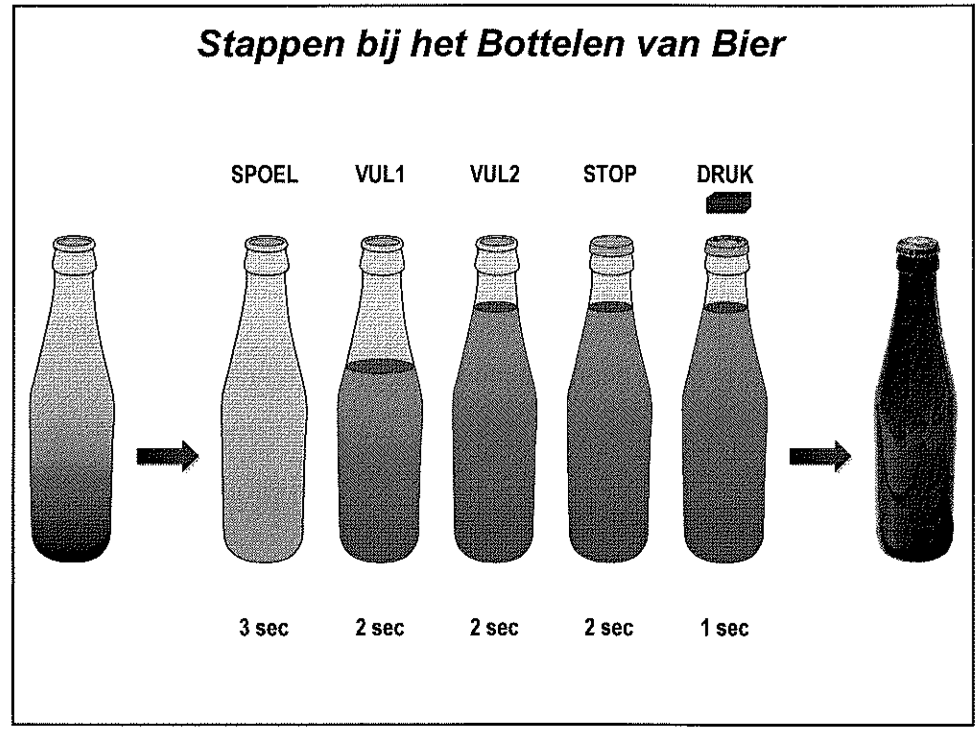
→ a pulse catching cell transforms a pulse into a level that can be synchronized to the system clock with a synchronizer

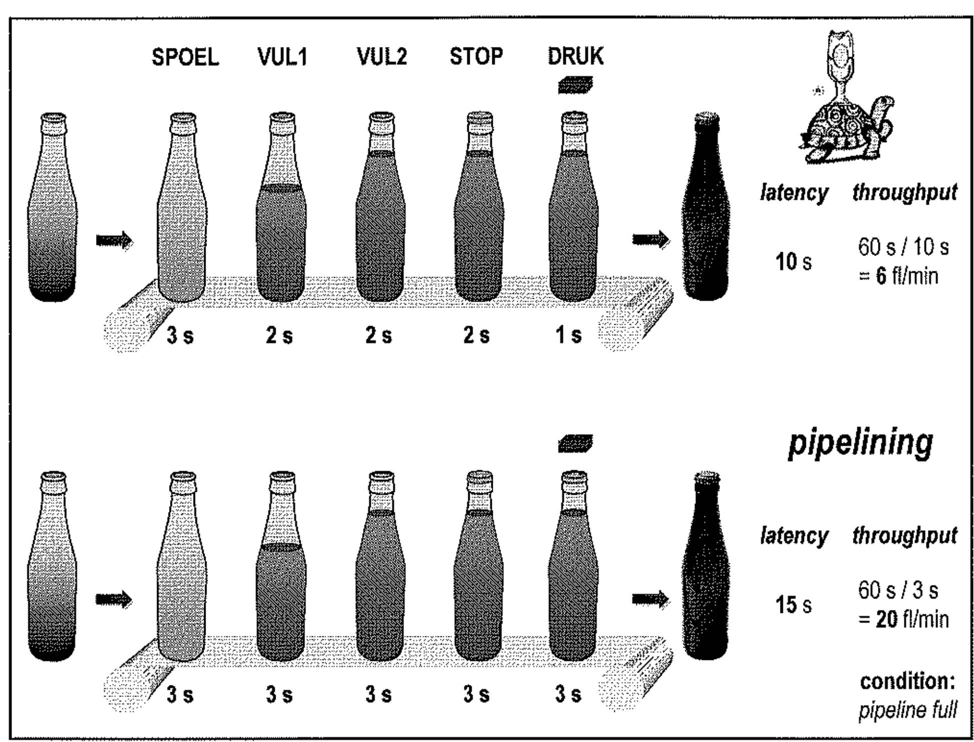
OUTPUTS:

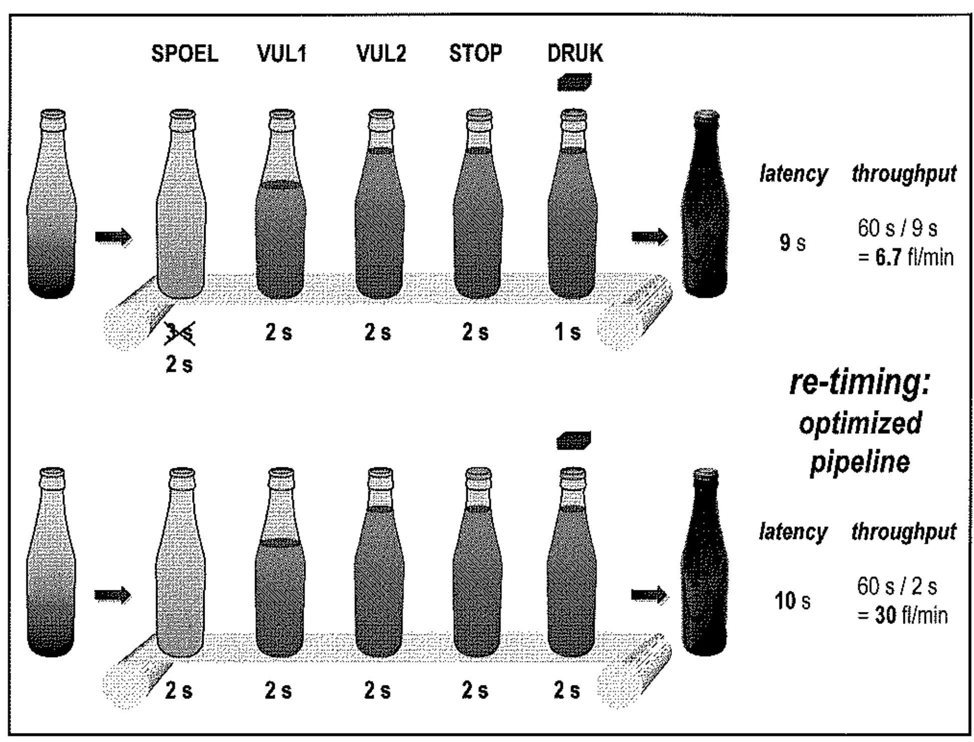
. outputs driving asynchronous (edge or level sensitive) inputs

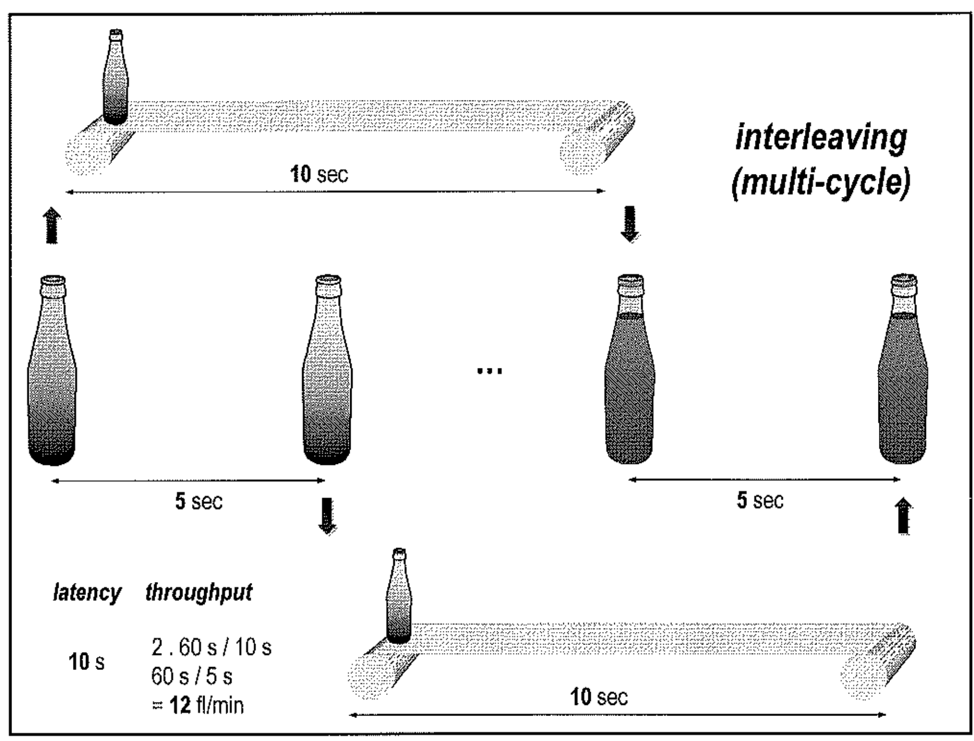
→ an output holding register makes the output glitch-free

HOGESCHOOL VOOR WETENSCHAP & KUNST
DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER
Digitale Synthese
High Throughput Design
Analogie
EmSD
Embedded System Design
ASSOCIATIE K.U.LEUVEN
ir. J. Meel
april 2008

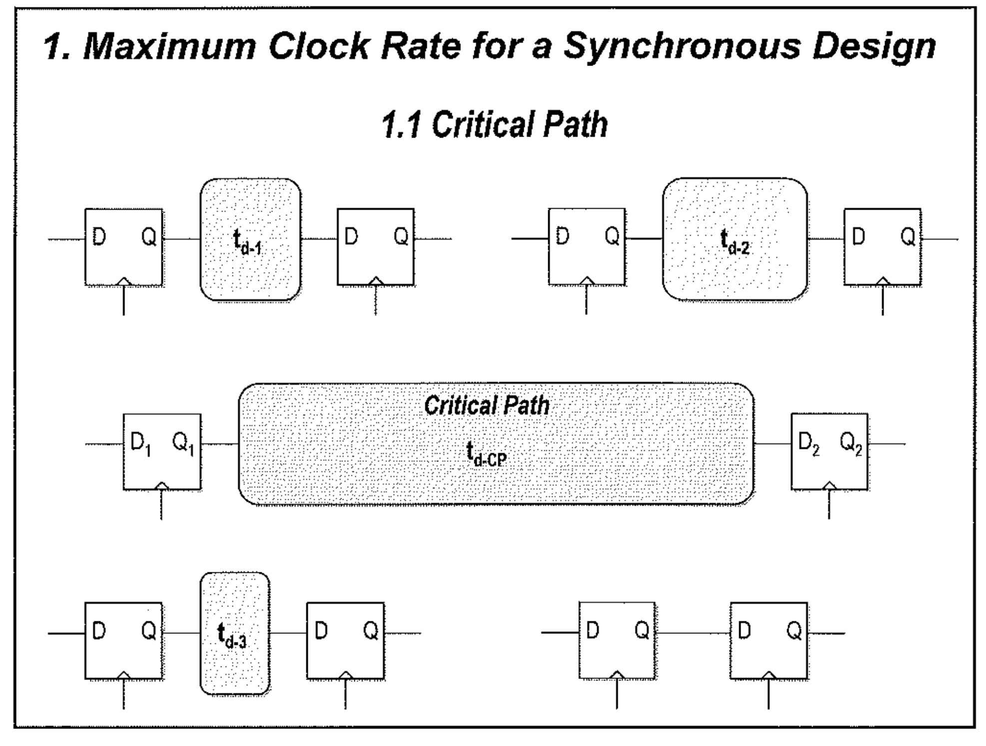

Stappen bij het Bottelen van Bier
SPOEL
VUL1
VUL2
STOP
DRUK
3 sec
2 sec
2 sec
2 sec
1 sec

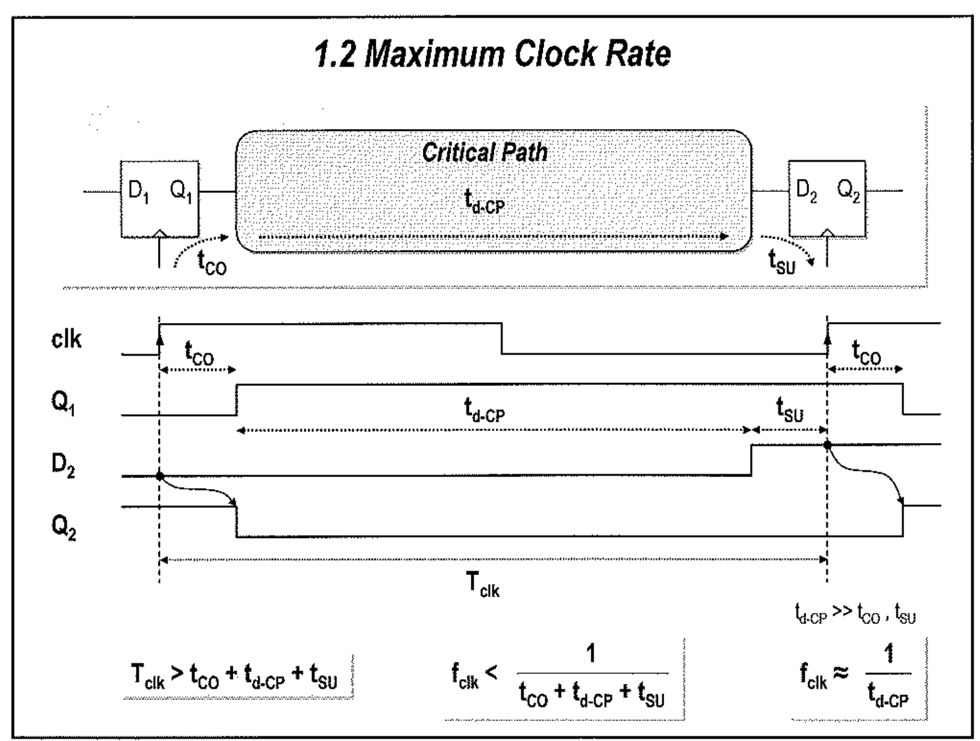

SPOEL
VUL1
VUL2
STOP
DRUK
latency
throughput
10 s
60 s / 10 s
= 6 fl/min
3 s
2 s
2 s
2 s
1 s
pipelining
latency
throughput
15 s
60 s / 3 s
= 20 fl/min
condition:
pipeline full
3 s
3 s
3 s
3 s
3 s

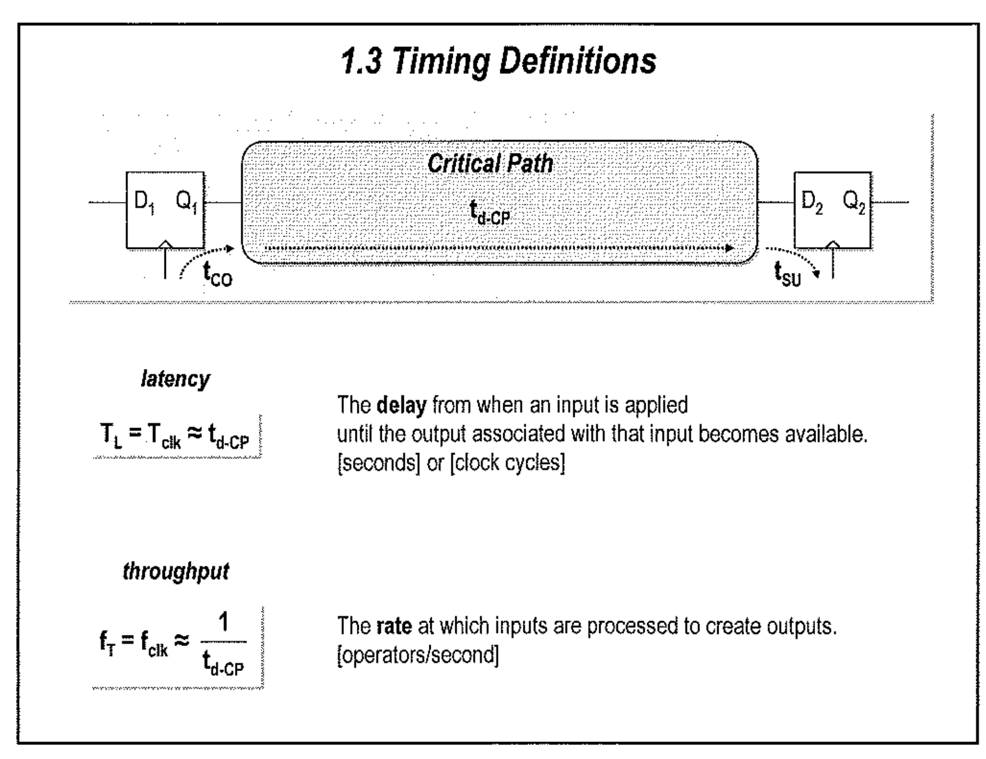

SPOEL
VUL1
VUL2
STOP
DRUK
latency
throughput
9 s
60 s / 9 s
= 6.7 fl/min
3 s
2 s
2 s
2 s
2 s
1 s
re-timing:
optimized
pipeline
latency
throughput
10 s
60 s / 2 s
= 30 fl/min
2 s
2 s
2 s
2 s
2 s

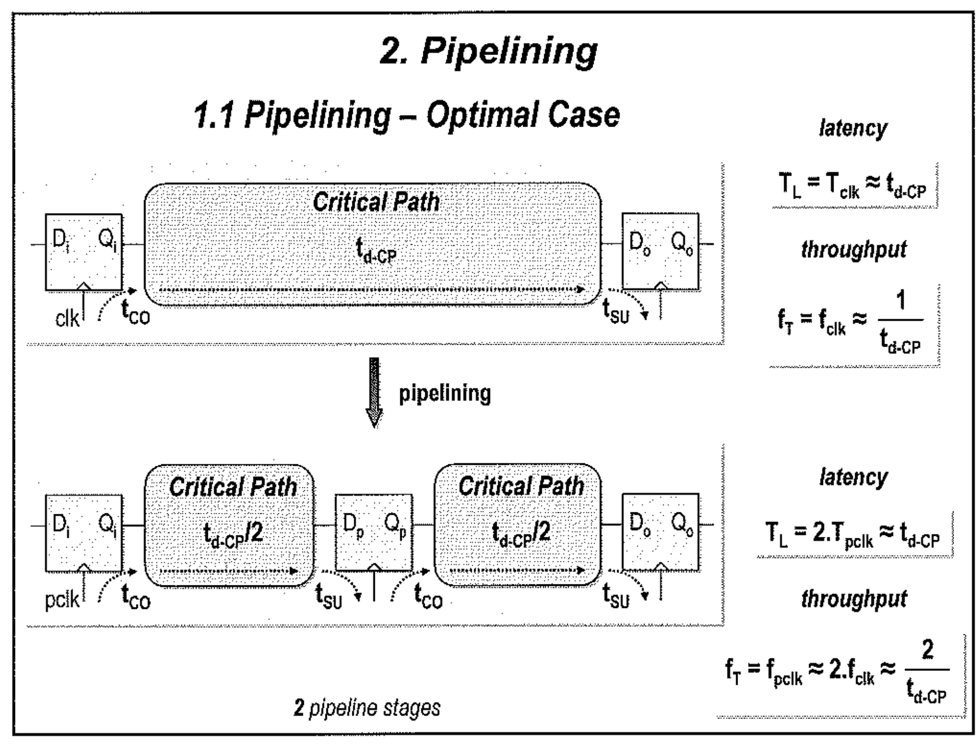

interleaving
(multi-cycle)
10 sec
5 sec
5 sec
latency throughput
10 s
2 . 60 s / 10 s
60 s / 5 s
= 12 fl/min
10 sec

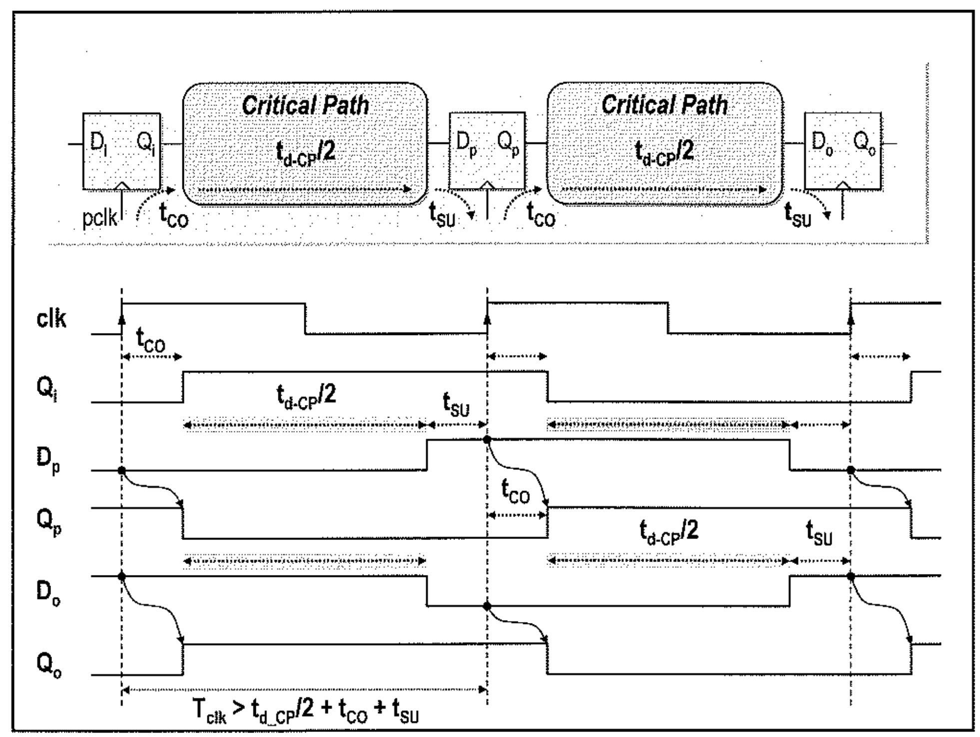
HOGESCHOOL VOOR WETENSCHAP & KUNST
DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER
Digitale Synthese
High Throughput Design
EmSD
Embedded System Design
ASSOCIATIE K.U.LEUVEN
ir. J. Meel
april 2008

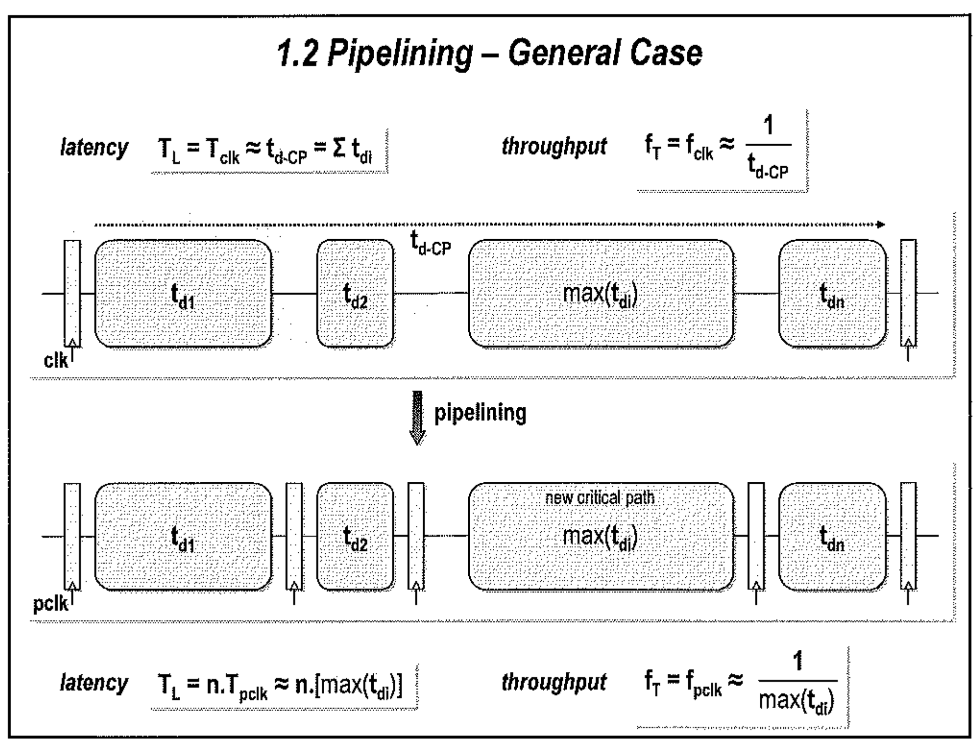

1. Maximum Clock Rate for a Synchronous Design
1.1 Critical Path
D Q
t_d-1
t_d-2
Critical Path
t_d-CP
D1 Q1
D2 Q2
t_d-3

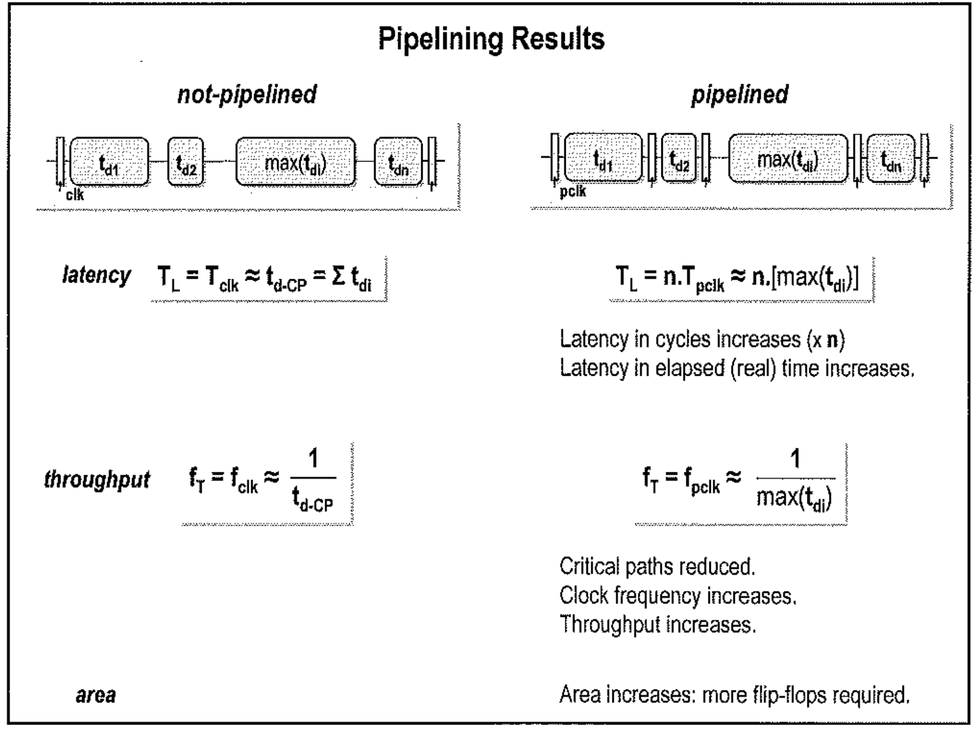

1.2 Maximum Clock Rate
Critical Path
$t_{d\text{-}CP}$
D_1 Q_1
D_2 Q_2
t_{CO}
t_{SU}
clk
Q_1
D_2
Q_2
T_{clk}
$t_{d\text{-}CP} >> t_{CO}, t_{SU}$
$T_{clk} > t_{CO} + t_{d\text{-}CP} + t_{SU}$
$f_{clk} < \frac{1}{t_{CO} + t_{d\text{-}CP} + t_{SU}}$
$f_{clk} \approx \frac{1}{t_{d\text{-}CP}}$

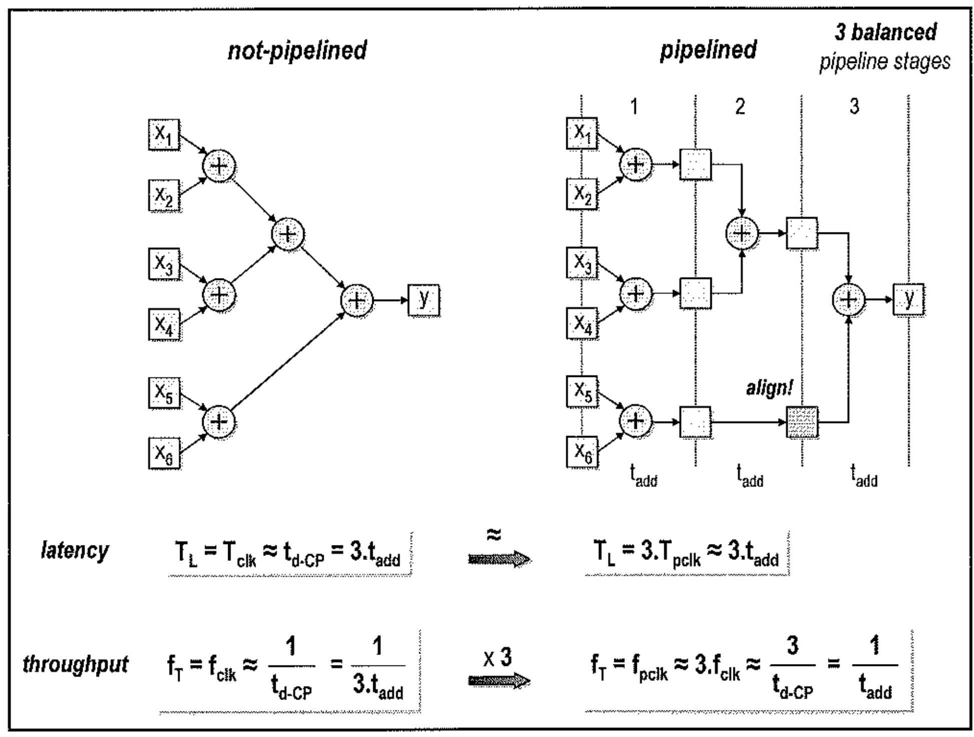

1.3 Timing Definitions
Critical Path
$t_{d\text{-}CP}$
D_1 Q_1
D_2 Q_2
t_{CO}
t_{SU}
latency
$T_L = T_{clk} \approx t_{d\text{-}CP}$
The delay from when an input is applied
until the output associated with that input becomes available.
[seconds] or [clock cycles]
throughput
$f_T = f_{clk} \approx \frac{1}{t_{d\text{-}CP}}$
The rate at which inputs are processed to create outputs.
[operators/second]

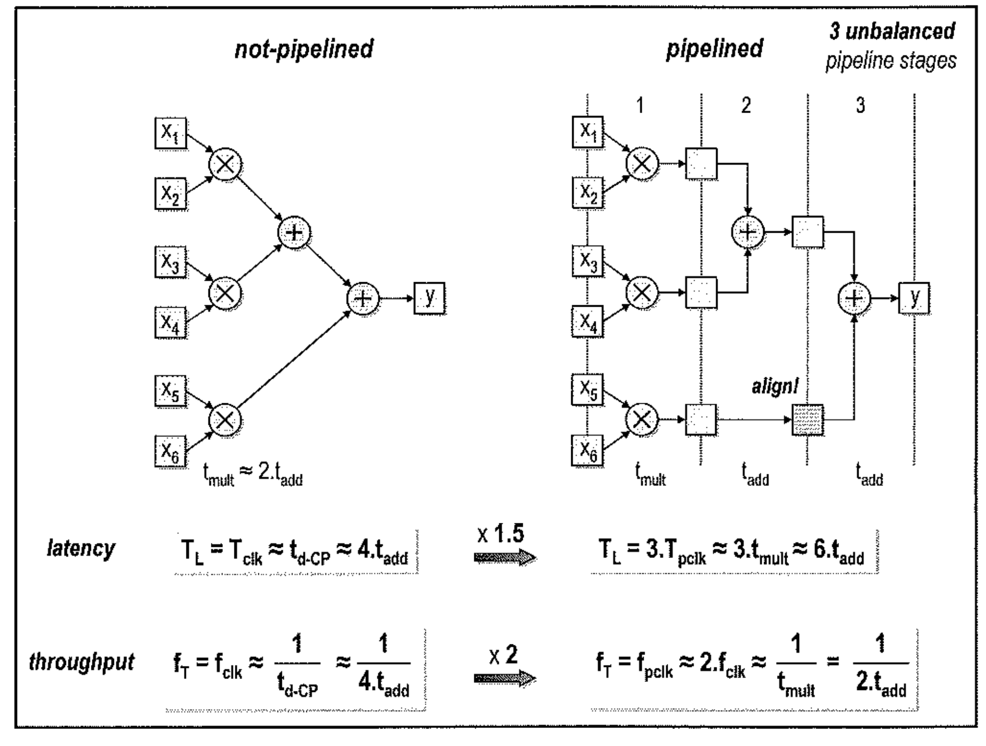

2. Pipelining
1.1 Pipelining – Optimal Case
Critical Path
t_d-CP
D_i Q_i
D_o Q_o
clk
t_CO
t_SU
latency
T_L = T_clk ≈ t_d-CP
throughput
f_T = f_clk ≈ 1/t_d-CP
pipelining
Critical Path
t_d-CP/2
D_p Q_p
pclk
latency
T_L = 2.T_pclk ≈ t_d-CP
throughput
f_T = f_pclk ≈ 2.f_clk ≈ 2/t_d-CP
2 pipeline stages

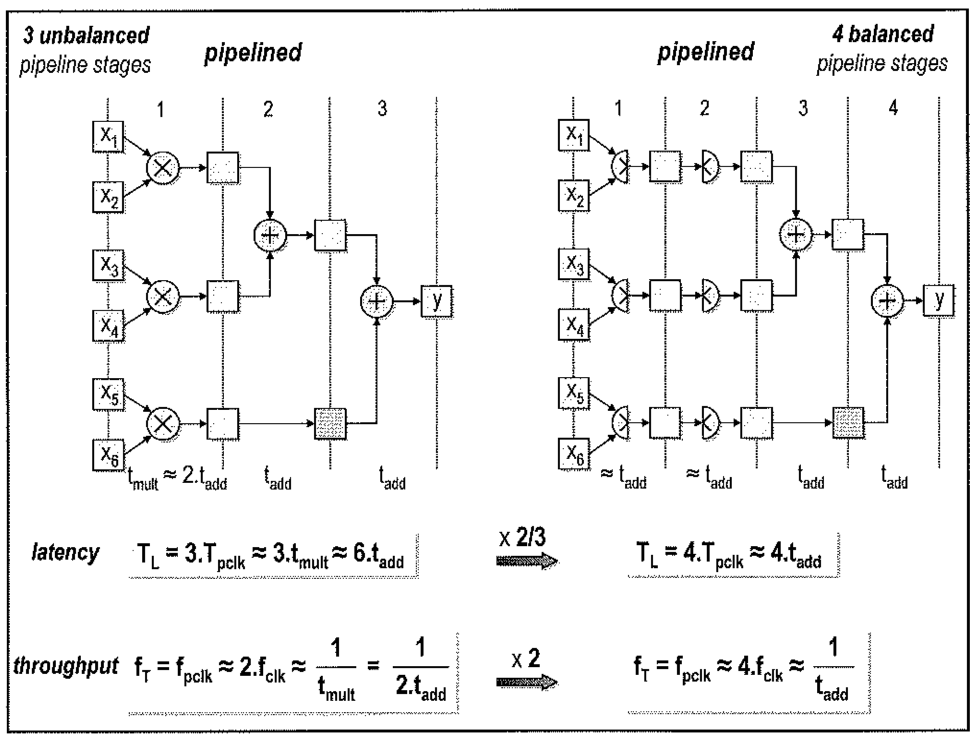

Critical Path
t_d-CP/2
D_i Q_i
D_p Q_p
D_o Q_o
pclk
t_CO
t_SU
clk
Q_i
D_p
Q_p
D_o
Q_o
T_clk > t_d_CP/2 + t_CO + t_SU

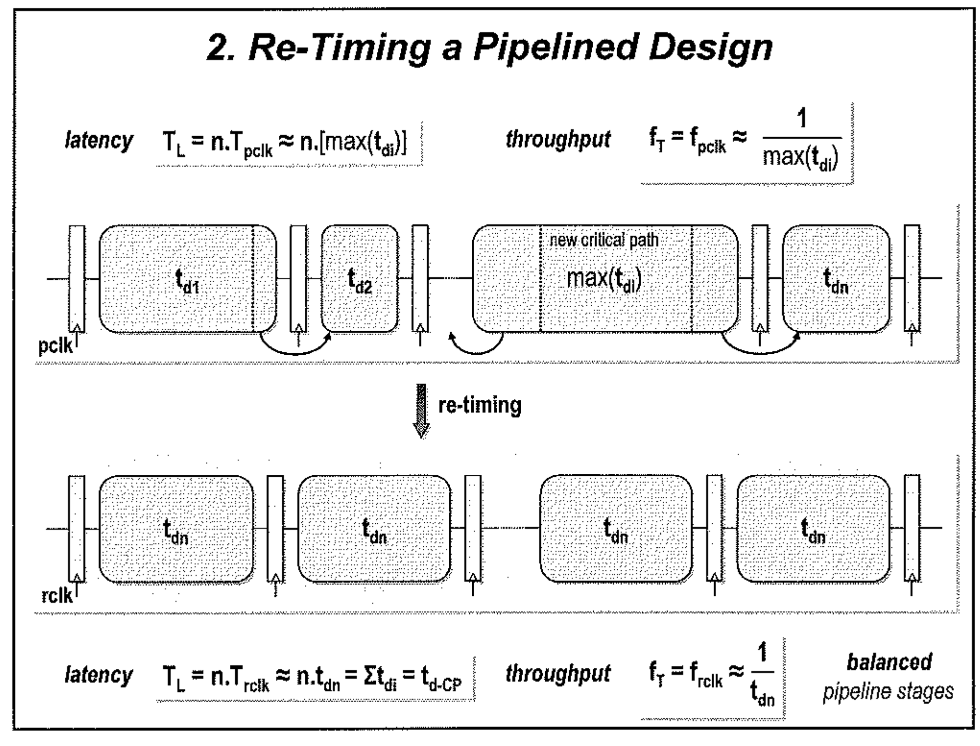

1.2 Pipelining – General Case
latency $T_L = T_{clk} \approx t_{d\text{-}CP} = \Sigma\, t_{di}$
throughput $f_T = f_{clk} \approx \frac{1}{t_{d\text{-}CP}}$
$t_{d\text{-}CP}$
t_{d1}
t_{d2}
$max(t_{di})$
t_{dn}
clk
pipelining
new critical path
pclk
latency $T_L = n.T_{pclk} \approx n.[max(t_{di})]$
throughput $f_T = f_{pclk} \approx \frac{1}{max(t_{di})}$

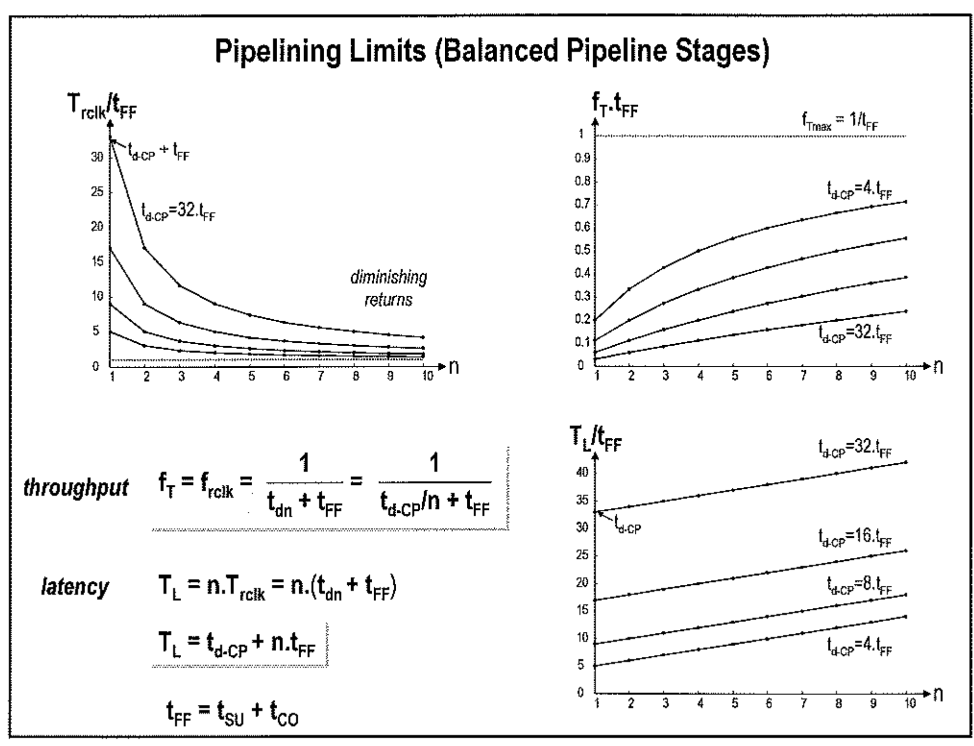

Pipelining Results
not-pipelined
pipelined
clk
pclk
latency $T_L = T_{clk} \approx t_{d\text{-}CP} = \Sigma\, t_{di}$
$T_L = n.T_{pclk} \approx n.[max(t_{di})]$
Latency in cycles increases (x n)
Latency in elapsed (real) time increases.
throughput $f_T = f_{clk} \approx \frac{1}{t_{d\text{-}CP}}$
$f_T = f_{pclk} \approx \frac{1}{max(t_{di})}$
Critical paths reduced.
Clock frequency increases.
Throughput increases.
area
Area increases: more flip-flops required.

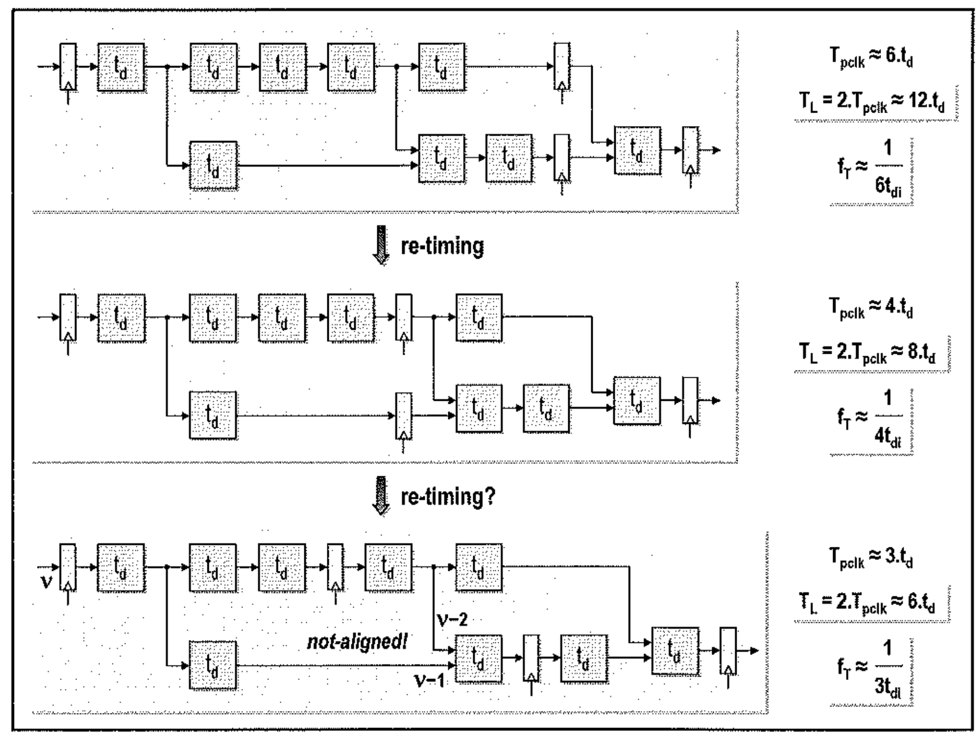

not-pipelined
pipelined
3 balanced
pipeline stages
1
2
3
align!
latency
throughput
$T_L = T_{clk} \approx t_{d\text{-}CP} = 3.t_{add}$
≈
$T_L = 3.T_{pclk} \approx 3.t_{add}$
$f_T = f_{clk} \approx \frac{1}{t_{d\text{-}CP}} = \frac{1}{3.t_{add}}$
x 3
$f_T = f_{pclk} \approx 3.f_{clk} \approx \frac{3}{t_{d\text{-}CP}} = \frac{1}{t_{add}}$

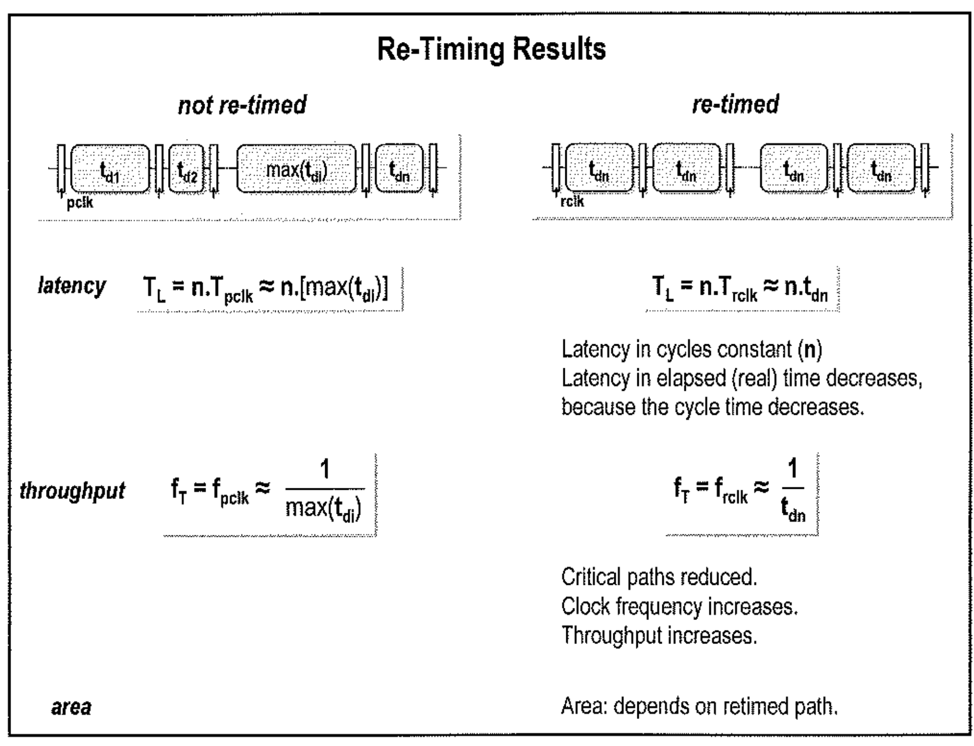

not-pipelined
pipelined
3 unbalanced
pipeline stages
1
2
3
align!
$t_{mult} \approx 2.t_{add}$
latency
throughput
$T_L = T_{clk} \approx t_{d\text{-}CP} \approx 4.t_{add}$
x 1.5
$T_L = 3.T_{pclk} \approx 3.t_{mult} \approx 6.t_{add}$
$f_T = f_{clk} \approx \frac{1}{t_{d\text{-}CP}} \approx \frac{1}{4.t_{add}}$
x 2
$f_T = f_{pclk} \approx 2.f_{clk} \approx \frac{1}{t_{mult}} = \frac{1}{2.t_{add}}$

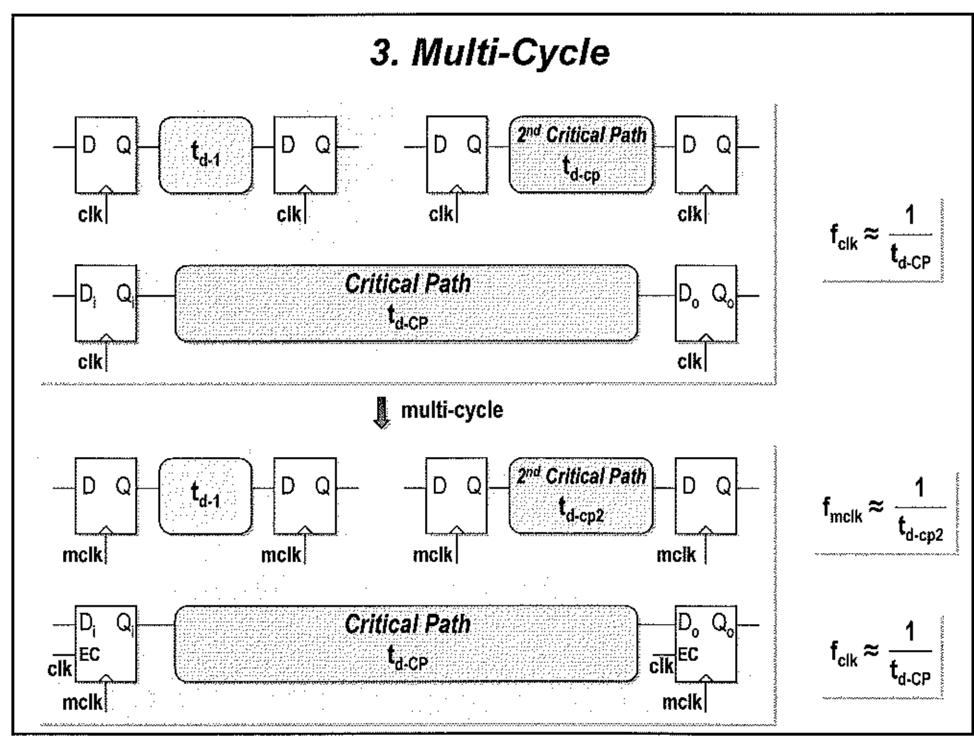

3 unbalanced pipeline stages
pipelined
pipelined
4 balanced pipeline stages
1
2
3
4
X_1
X_2
X_3
X_4
X_5
X_6
y
$t_{mult} \approx 2.t_{add}$
t_{add}
$\approx t_{add}$
latency
$T_L = 3.T_{pclk} \approx 3.t_{mult} \approx 6.t_{add}$
x 2/3
$T_L = 4.T_{pclk} \approx 4.t_{add}$
throughput
$f_T = f_{pclk} \approx 2.f_{clk} \approx \frac{1}{t_{mult}} = \frac{1}{2.t_{add}}$
x 2
$f_T = f_{pclk} \approx 4.f_{clk} \approx \frac{1}{t_{add}}$

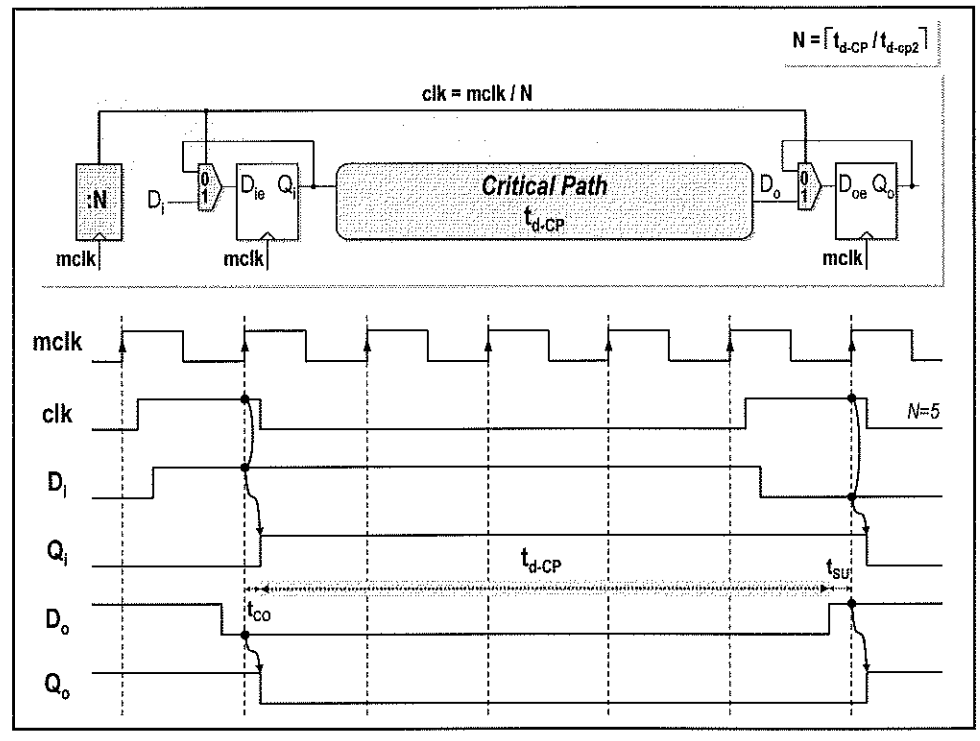

2. Re-Timing a Pipelined Design
latency
$T_L = n.T_{pclk} \approx n.[\max(t_{di})]$
throughput
$f_T = f_{pclk} \approx \frac{1}{\max(t_{di})}$
t_{d1}
t_{d2}
new critical path
$\max(t_{di})$
t_{dn}
pclk
re-timing
rclk
latency
$T_L = n.T_{rclk} \approx n.t_{dn} = \Sigma t_{di} = t_{d\text{-}CP}$
throughput
$f_T = f_{rclk} \approx \frac{1}{t_{dn}}$
balanced pipeline stages

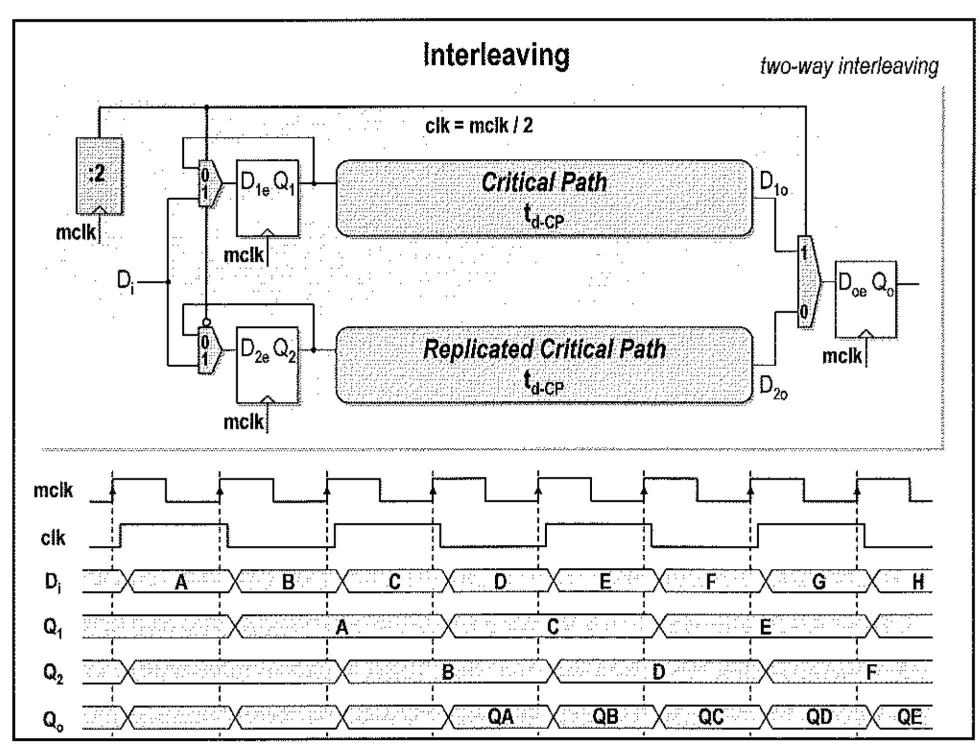

Pipelining Limits (Balanced Pipeline Stages)
T_{rclk}/t_{FF}
$t_{d\text{-}CP} + t_{FF}$
$t_{d\text{-}CP}=32.t_{FF}$
diminishing returns
$f_T.t_{FF}$
$f_{Tmax} = 1/t_{FF}$
$t_{d\text{-}CP}=4.t_{FF}$
$t_{d\text{-}CP}=32.t_{FF}$
T_L/t_{FF}
$t_{d\text{-}CP}=32.t_{FF}$
$t_{d\text{-}CP}$
$t_{d\text{-}CP}=16.t_{FF}$
$t_{d\text{-}CP}=8.t_{FF}$
$t_{d\text{-}CP}=4.t_{FF}$
n
throughput $f_T = f_{rclk} = \frac{1}{t_{dn} + t_{FF}} = \frac{1}{t_{d\text{-}CP}/n + t_{FF}}$
latency $T_L = n.T_{rclk} = n.(t_{dn} + t_{FF})$
$T_L = t_{d\text{-}CP} + n.t_{FF}$
$t_{FF} = t_{SU} + t_{CO}$

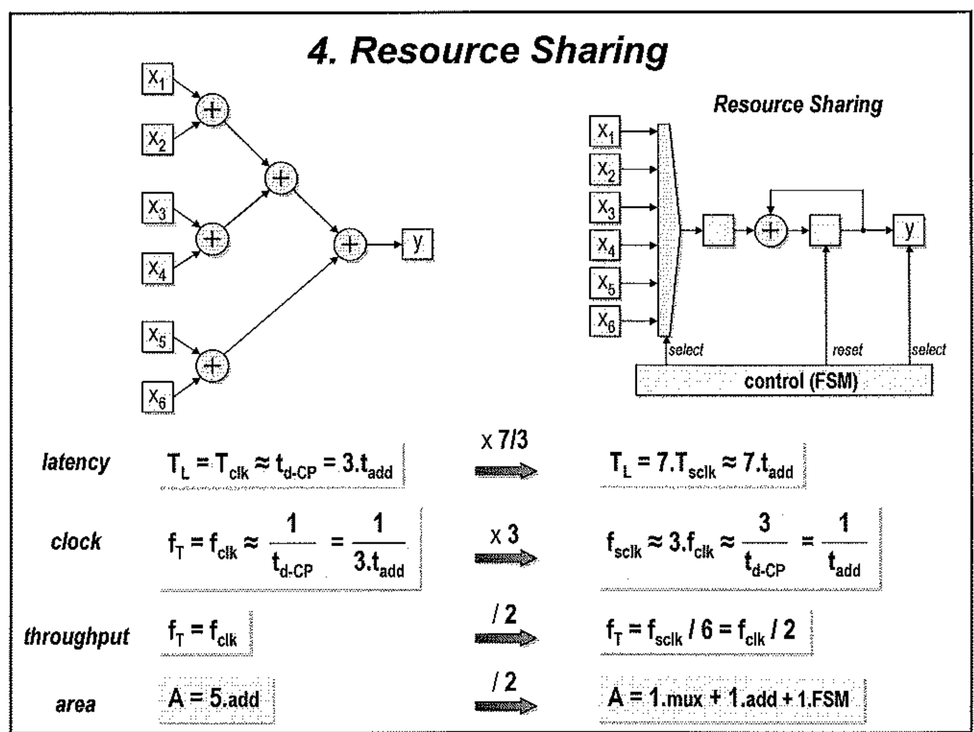

$T_{pclk} \approx 6.t_d$
$T_L = 2.T_{pclk} \approx 12.t_d$
$f_T \approx \frac{1}{6t_{di}}$
re-timing
$T_{pclk} \approx 4.t_d$
$T_L = 2.T_{pclk} \approx 8.t_d$
$f_T \approx \frac{1}{4t_{di}}$
re-timing?
$T_{pclk} \approx 3.t_d$
$T_L = 2.T_{pclk} \approx 6.t_d$
$f_T \approx \frac{1}{3t_{di}}$
not-aligned!
ν
ν−2
ν−1
t_d


Re-Timing Results
not re-timed
re-timed
pclk
rclk
latency
$T_L = n.T_{pclk} \approx n.[\max(t_{di})]$
$T_L = n.T_{rclk} \approx n.t_{dn}$
Latency in cycles constant (n)
Latency in elapsed (real) time decreases, because the cycle time decreases.
throughput
$f_T = f_{pclk} \approx \frac{1}{\max(t_{di})}$
$f_T = f_{rclk} \approx \frac{1}{t_{dn}}$
Critical paths reduced.
Clock frequency increases.
Throughput increases.
area
Area: depends on retimed path.


3. Multi-Cycle
2nd Critical Path
Critical Path
multi-cycle
clk
mclk
EC
$f_{clk} \approx \frac{1}{t_{d\text{-}CP}}$
$f_{mclk} \approx \frac{1}{t_{d\text{-}cp2}}$


N = ⌈t_d-CP / t_d-cp2⌉
clk = mclk / N
:N
mclk
D_i
D_ie Q_i
Critical Path
t_d-CP
D_o
D_oe Q_o
mclk
clk
N=5
D_i
Q_i
t_d-CP
t_SU
D_o
t_CO
Q_o

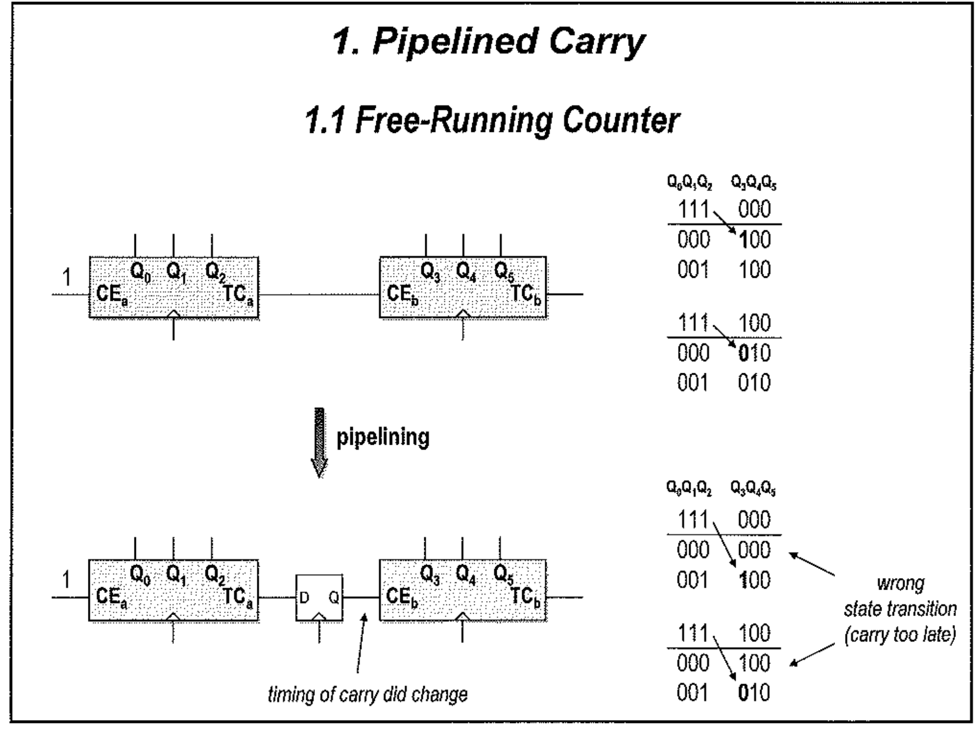

Interleaving
two-way interleaving
clk = mclk / 2
:2
mclk
D_1e Q_1
Critical Path
t_d-CP
D_1o
D_i
D_2e Q_2
Replicated Critical Path
t_d-CP
D_2o
D_oe Q_o
mclk
clk
D_i
A B C D E F G H
Q_1
A C E
Q_2
B D F
Q_o
QA QB QC QD QE

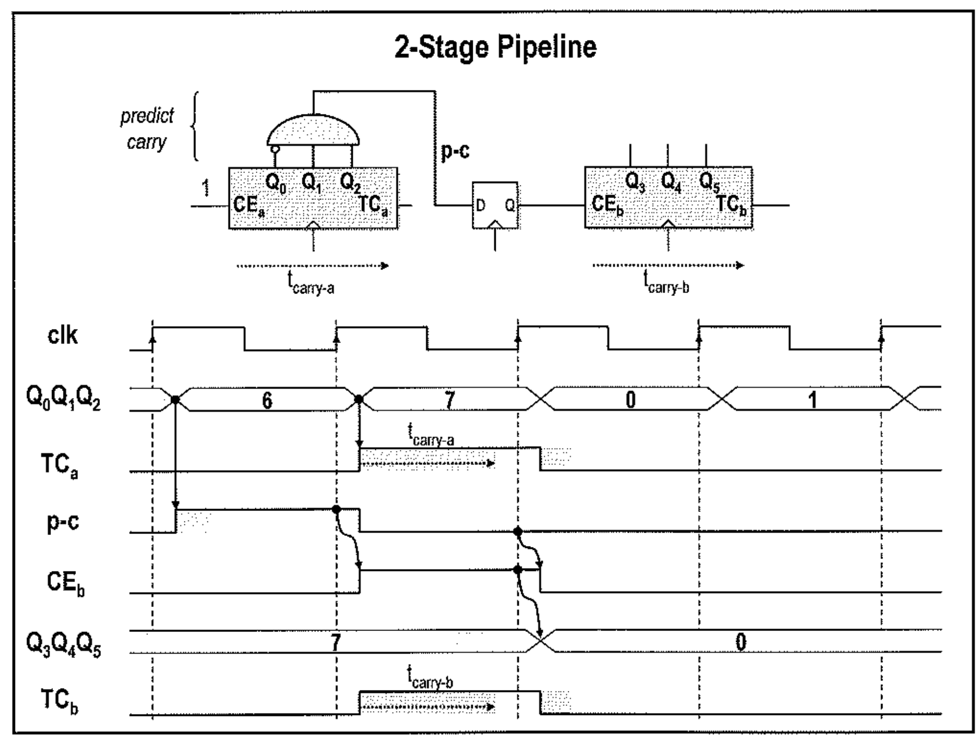


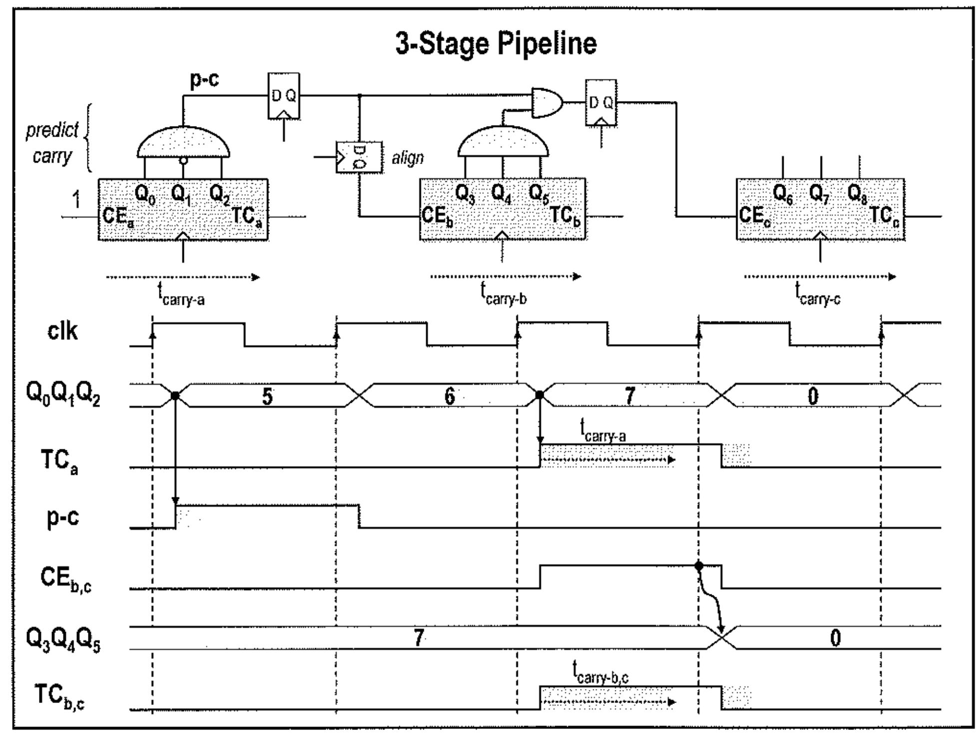
More problems can arise:
– Pipelining data hazards occur when a computation depends on the result of a previous computation still in the pipeline.
– Pipelining control hazards occur when a computation in the pipeline changes the selection of inputs to the design.

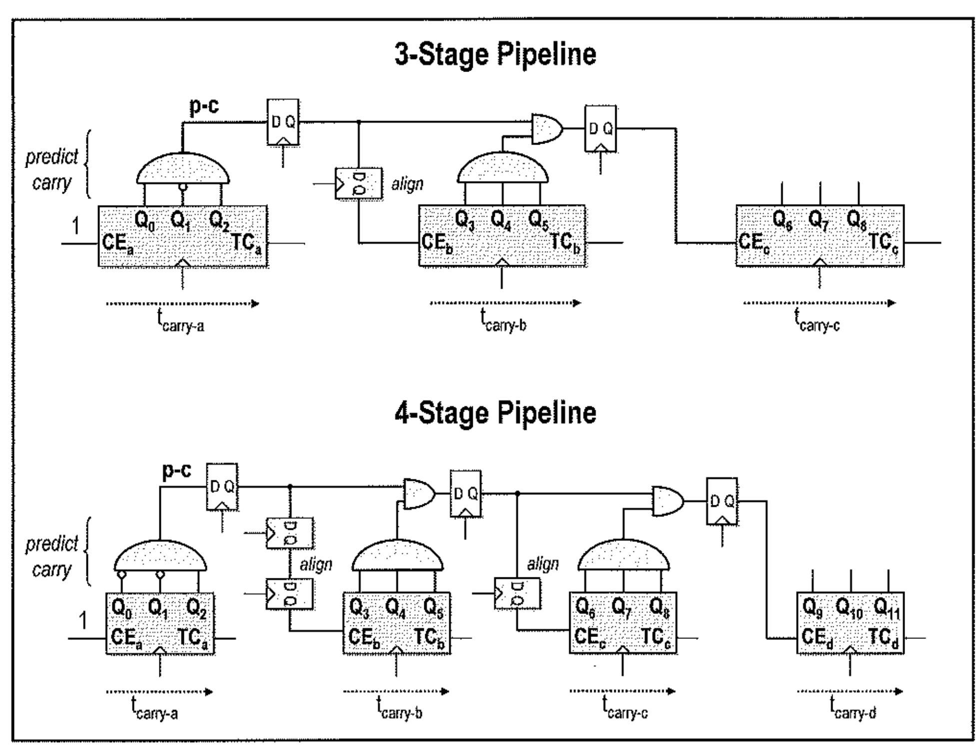
Carry ketting!

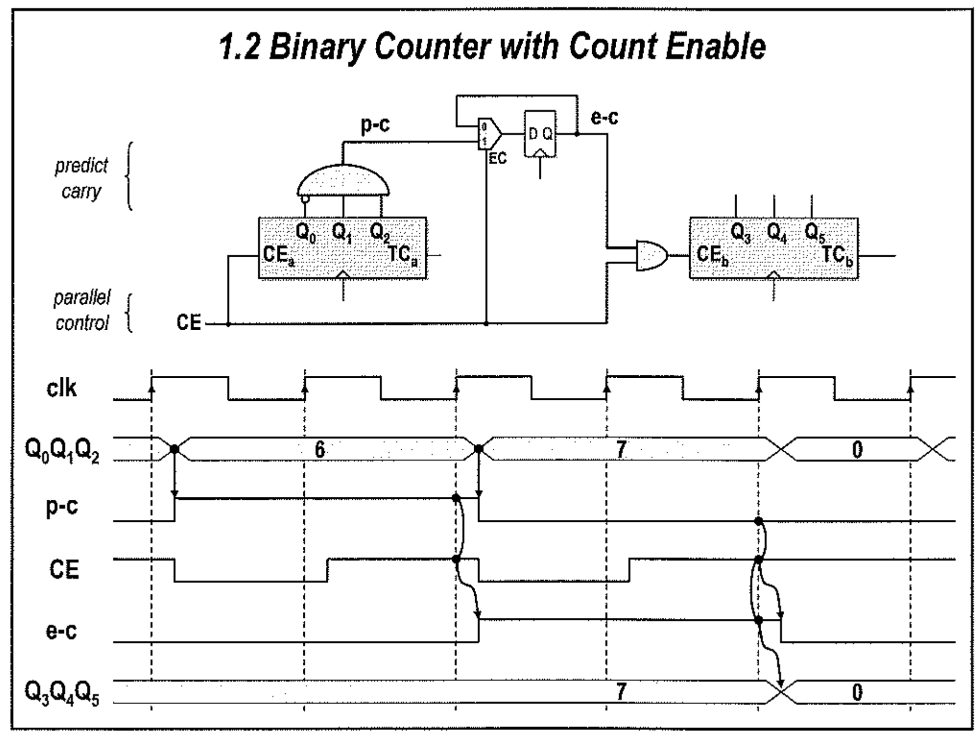
HOGESCHOOL VOOR WETENSCHAP & KUNST
DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER
Digitale Synthese
High Speed Counters
EmSD
Embedded System Design
ASSOCIATIE K.U.LEUVEN
ir. J. Meel
april 2008

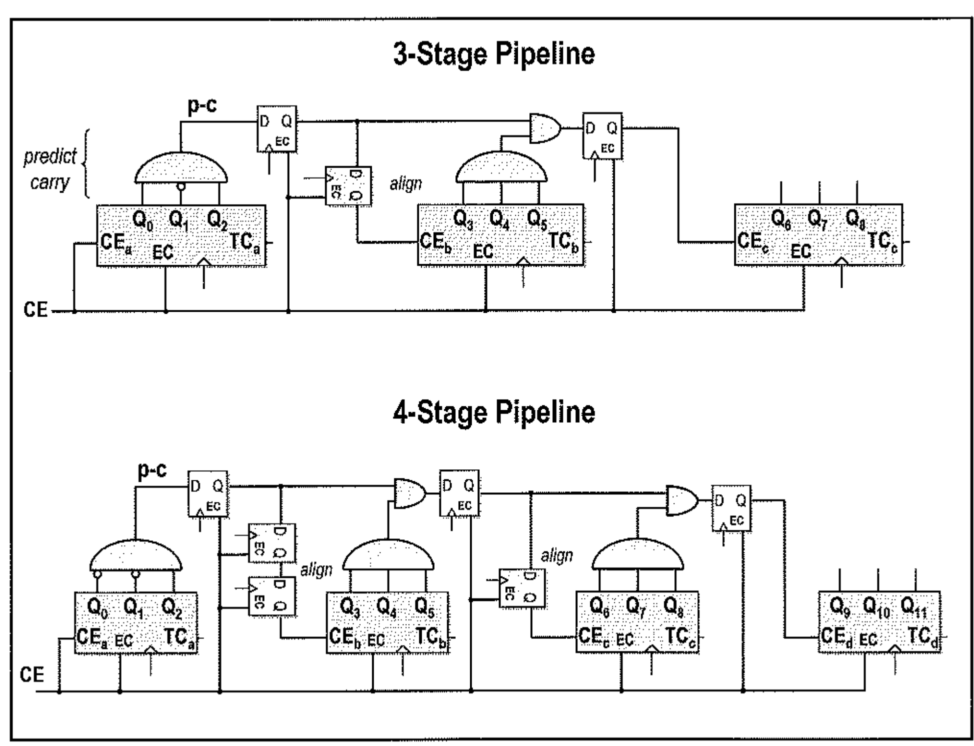

1. Pipelined Carry
1.1 Free-Running Counter
pipelining
timing of carry did change
wrong
state transition
(carry too late)

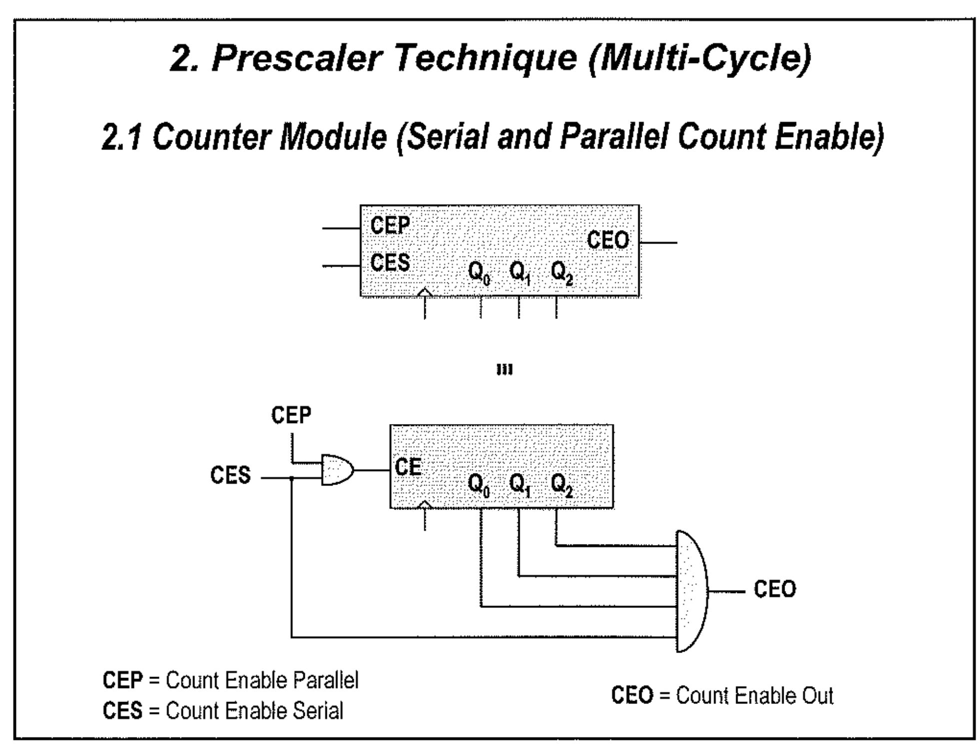

2-Stage Pipeline
predict carry
p-c
1
CEa Q0 Q1 Q2 TCa
D Q
CEb Q3 Q4 Q5 TCb
tcarry-a
tcarry-b
clk
Q0Q1Q2
6
7
0
1
TCa
p-c
CEb
Q3Q4Q5
TCb

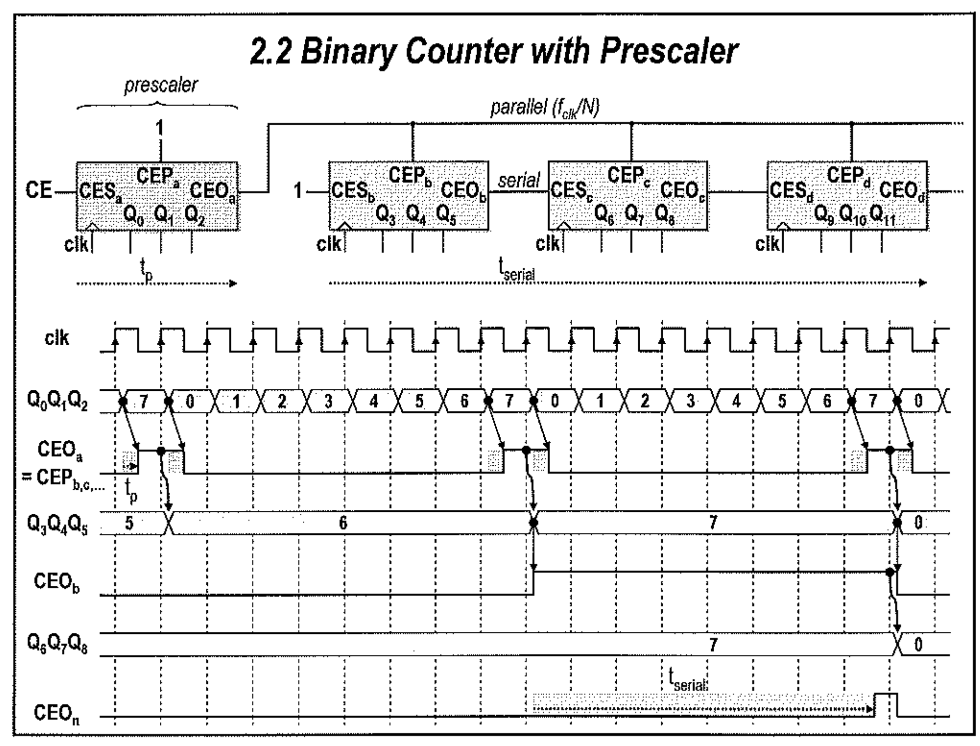

3-Stage Pipeline
p-c
predict carry
align
1
CEa Q0 Q1 Q2 TCa
CEb Q3 Q4 Q5 TCb
CEc Q6 Q7 Q8 TCc
tcarry-a
tcarry-b
tcarry-c
clk
Q0Q1Q2
5
6
7
0
TCa
p-c
CEb,c
Q3Q4Q5
TCb,c
tcarry-b,c

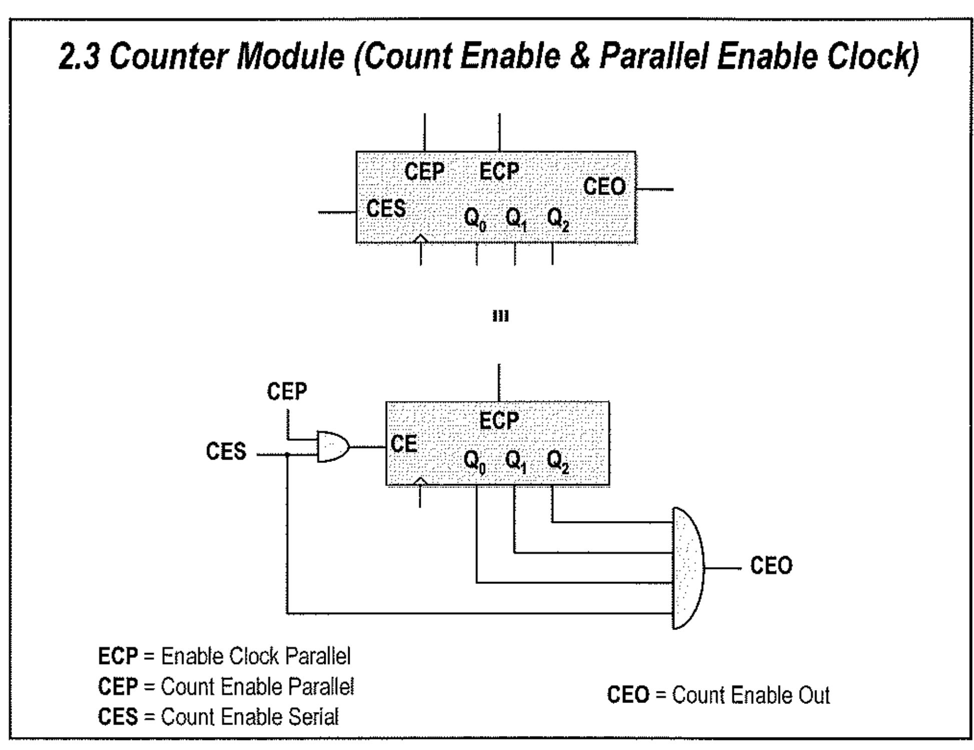

3-Stage Pipeline
p-c
predict carry
align
1
CEa Q0 Q1 Q2 TCa
CEb Q3 Q4 Q5 TCb
CEc Q6 Q7 Q8 TCc
tcarry-a
tcarry-b
tcarry-c
4-Stage Pipeline
p-c
predict carry
align
align
1
CEa Q0 Q1 Q2 TCa
CEb Q3 Q4 Q5 TCb
CEc Q6 Q7 Q8 TCc
CEd Q9 Q10 Q11 TCd
tcarry-a
tcarry-b
tcarry-c
tcarry-d

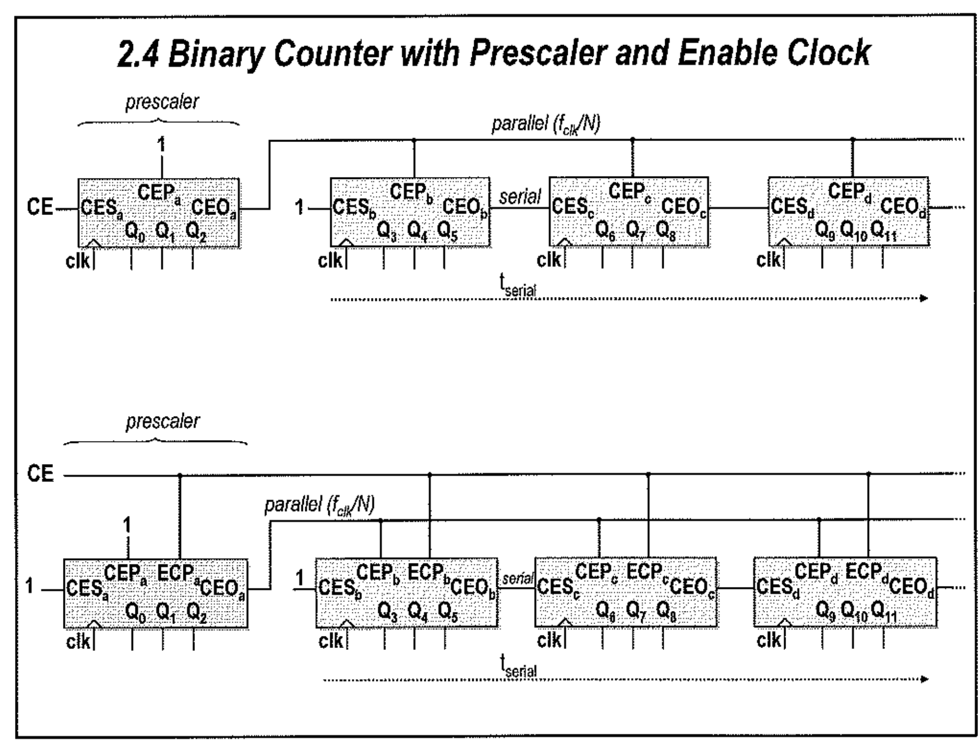

1.2 Binary Counter with Count Enable
p-c
e-c
EC
predict carry
CEa Q0 Q1 Q2 TCa
CEb Q3 Q4 Q5 TCb
parallel control
CE
clk
Q0Q1Q2
6
7
0
p-c
CE
e-c
Q3Q4Q5
7
0

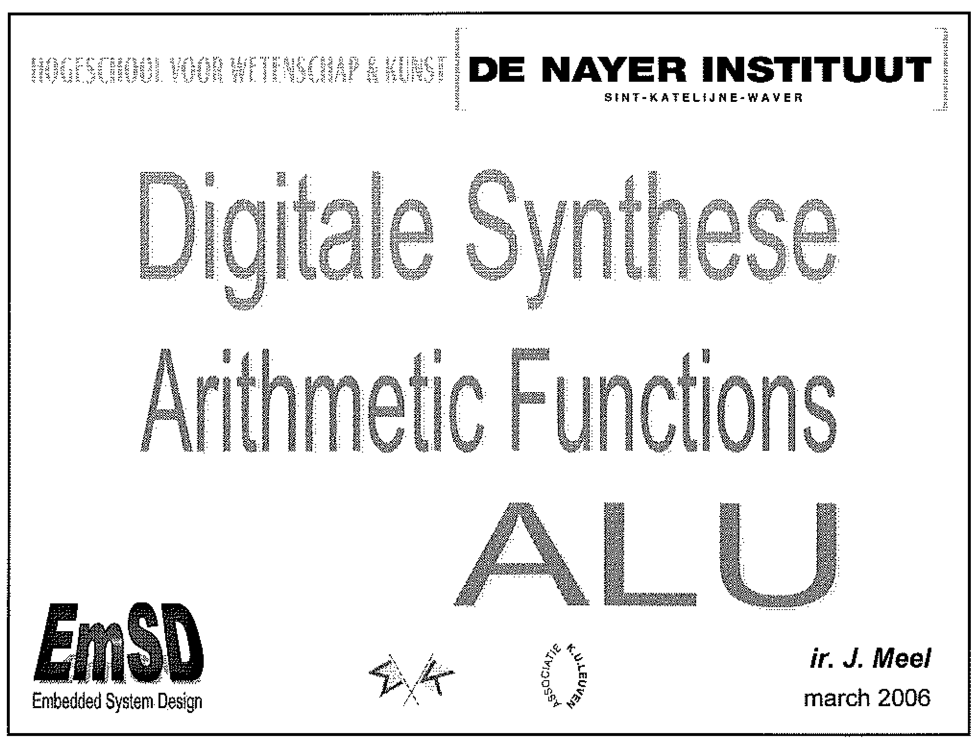

3-Stage Pipeline
p-c
predict carry
D Q
EC
align
Q_0 Q_1 Q_2
CE_a EC TC_a
Q_3 Q_4 Q_5
CE_b EC TC_b
Q_6 Q_7 Q_8
CE_c EC TC_c
CE
4-Stage Pipeline
Q_9 Q_{10} Q_{11}
CE_d EC TC_d

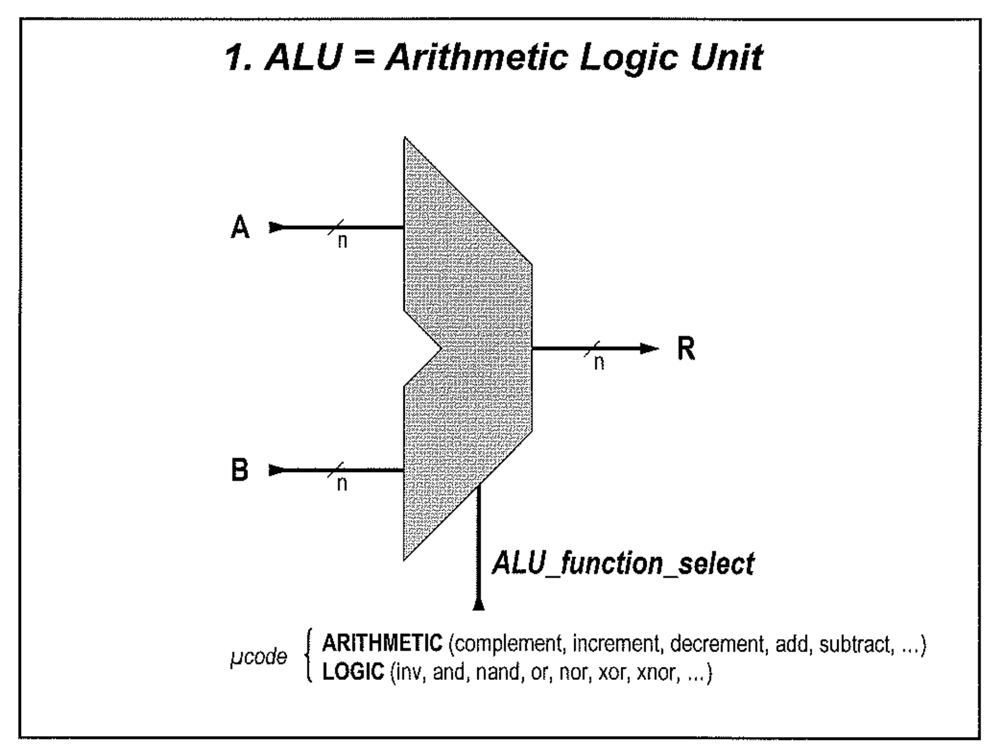

2. Prescaler Technique (Multi-Cycle)
2.1 Counter Module (Serial and Parallel Count Enable)
CEP
CES
CEO
Q_0 Q_1 Q_2
≡
CEP
CES
CE
Q_0 Q_1 Q_2
CEO
CEP = Count Enable Parallel
CES = Count Enable Serial
CEO = Count Enable Out

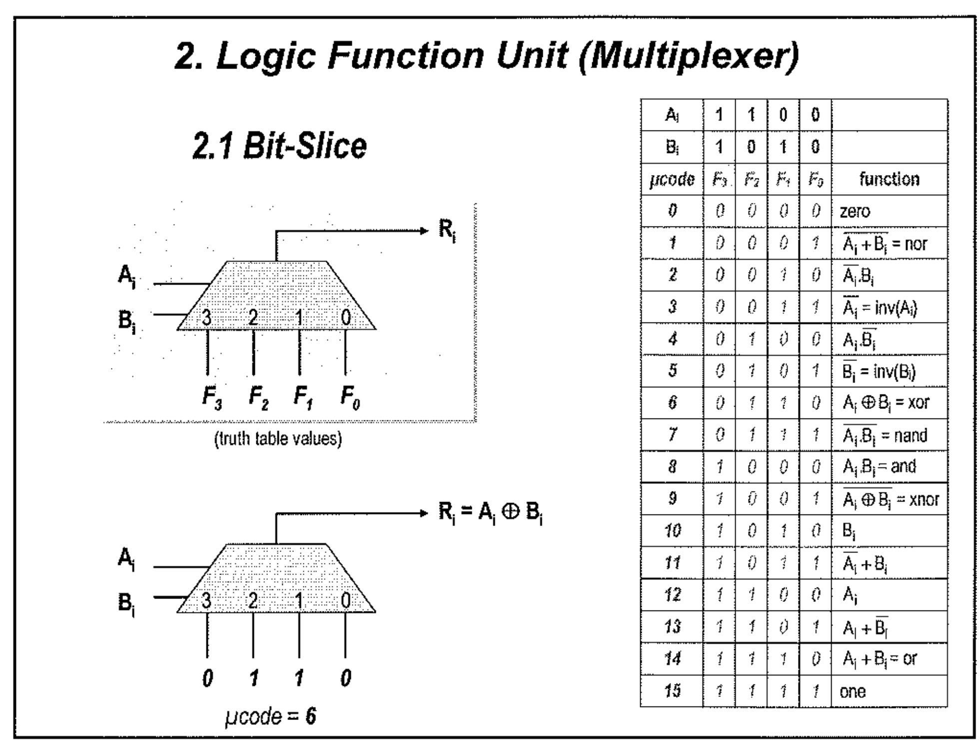

2.2 Binary Counter with Prescaler
prescaler
parallel (f_{clk}/N)
serial
CE
CES
CEP
CEO
clk
t_p
t_{serial}
$Q_0Q_1Q_2$
CEO_a = $CEP_{b,c,...}$
$Q_3Q_4Q_5$
CEO_b
$Q_6Q_7Q_8$
CEO_n


2.3 Counter Module (Count Enable & Parallel Enable Clock)
CEP
ECP
CEO
CES
Q_0 Q_1 Q_2
CE
ECP = Enable Clock Parallel
CEP = Count Enable Parallel
CES = Count Enable Serial
CEO = Count Enable Out

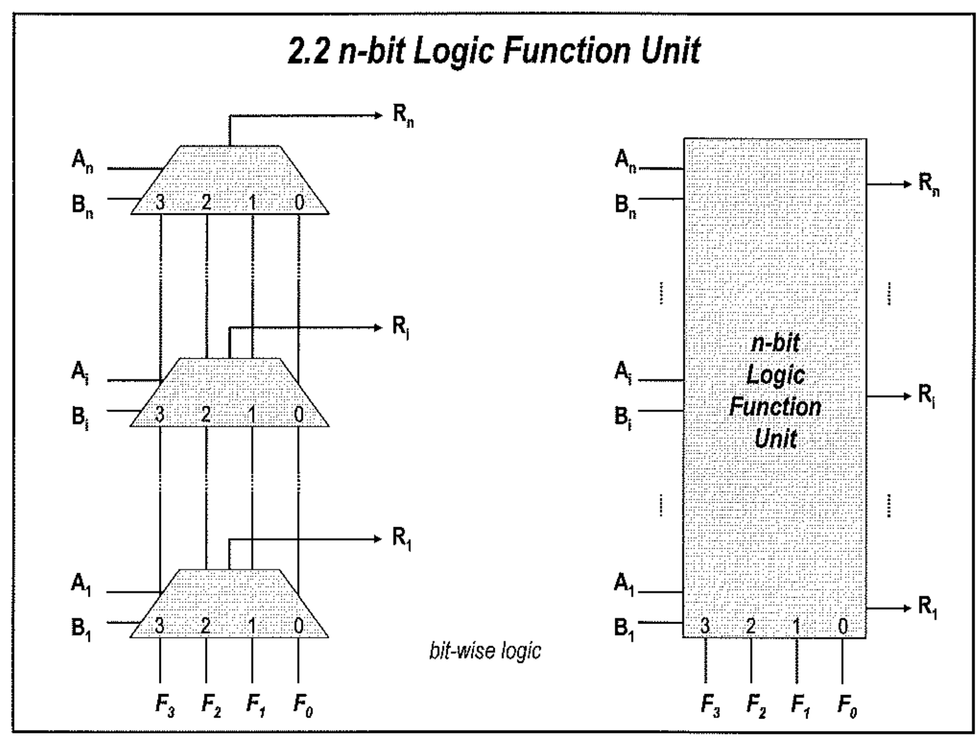

2.4 Binary Counter with Prescaler and Enable Clock
prescaler
parallel (f_clk/N)
CE
1
CES_a CEP_a CEO_a
Q_0 Q_1 Q_2
clk
CES_b CEP_b CEO_b
Q_3 Q_4 Q_5
serial
CES_c CEP_c CEO_c
Q_6 Q_7 Q_8
CES_d CEP_d CEO_d
Q_9 Q_10 Q_11
t_serial
prescaler
CE
parallel (f_clk/N)
CES_a CEP_a ECP_a CEO_a
CES_b CEP_b ECP_b CEO_b
serial
CES_c CEP_c ECP_c CEO_c
CES_d CEP_d ECP_d CEO_d
t_serial

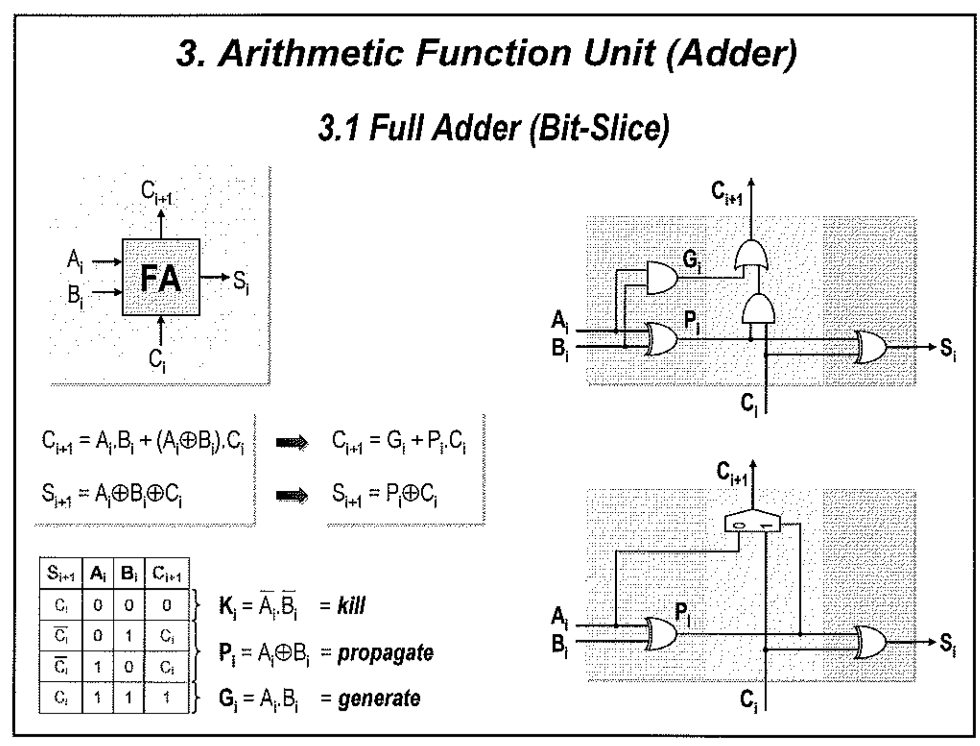

HOGESCHOOL VOOR WETENSCHAP & KUNST
DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER
Digitale Synthese
Arithmetic Functions
ALU
EmSD
Embedded System Design
ASSOCIATIE K.U.LEUVEN
ir. J. Meel
march 2006

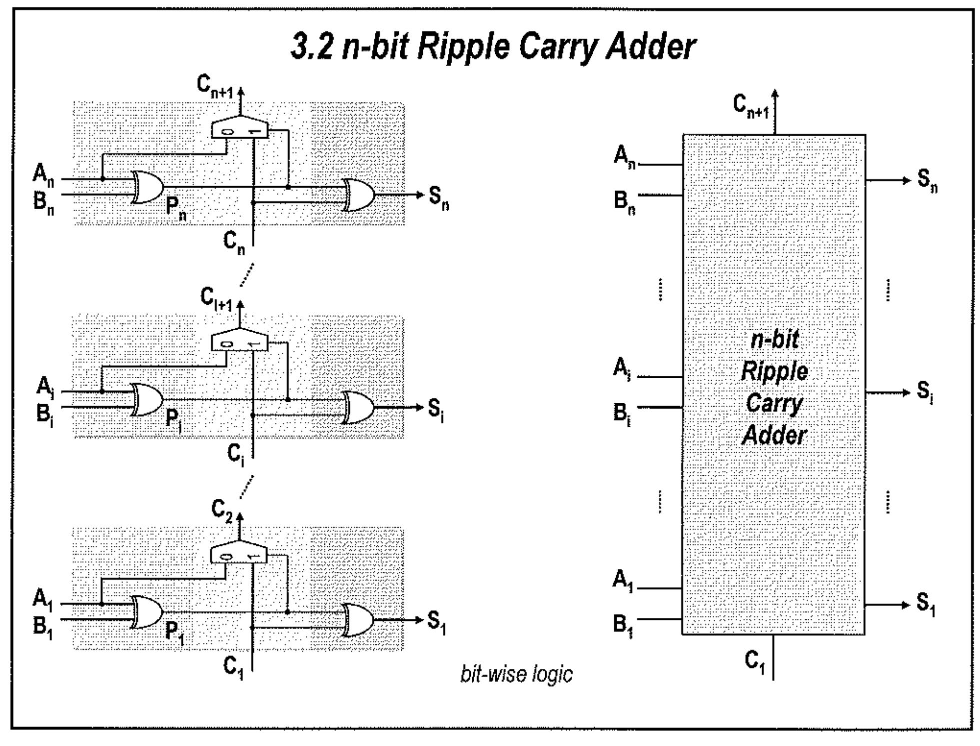

1. ALU = Arithmetic Logic Unit
A
n
B
n
R
n
ALU_function_select
µcode
ARITHMETIC (complement, increment, decrement, add, subtract, ...)
LOGIC (inv, and, nand, or, nor, xor, xnor, ...)

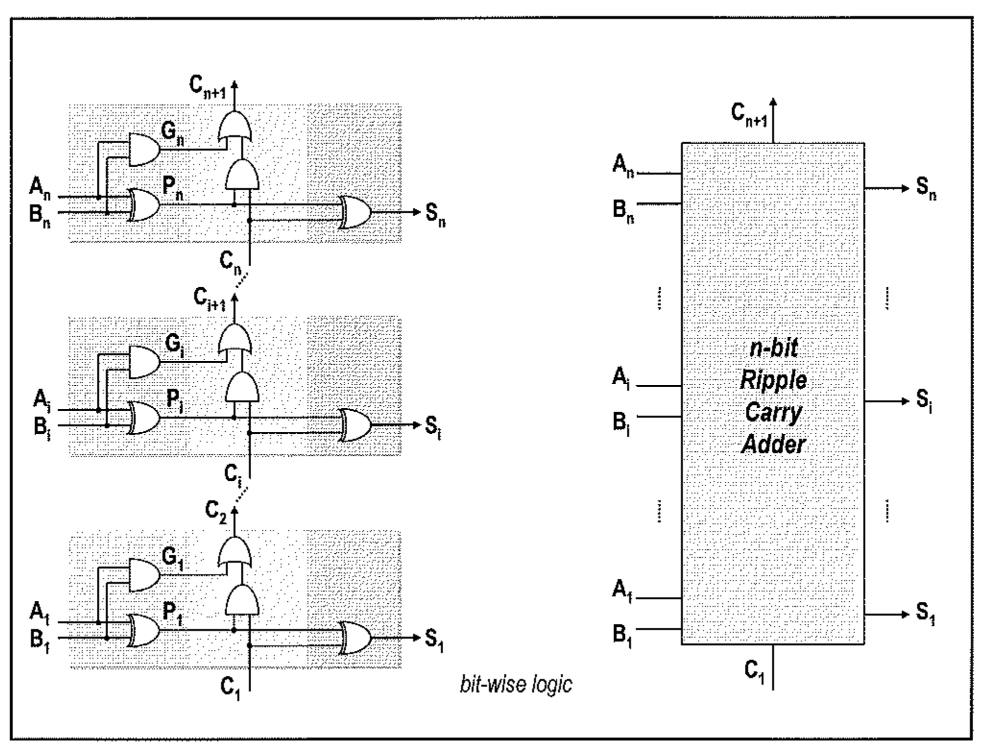


A_i	1	1	0	0	
B_i	1	0	1	0	
μcode	F_3	F_2	F_1	F_0	function
0	0	0	0	0	zero
1	0	0	0	1	$\overline{A_i + B_i}$ = nor
2	0	0	1	0	$\overline{A_i}.B_i$
3	0	0	1	1	$\overline{A_i}$ = inv(A_i)
4	0	1	0	0	$A_i.\overline{B_i}$
5	0	1	0	1	$\overline{B_i}$ = inv(B_i)
6	0	1	1	0	$A_i \oplus B_i$ = xor
7	0	1	1	1	$\overline{A_i.B_i}$ = nand
8	1	0	0	0	$A_i.B_i$ = and
9	1	0	0	1	$\overline{A_i \oplus B_i}$ = xnor
10	1	0	1	0	B_i
11	1	0	1	1	$\overline{A_i} + B_i$
12	1	1	0	0	A_i
13	1	1	0	1	$A_i + \overline{B_i}$
14	1	1	1	0	$A_i + B_i$ = or
15	1	1	1	1	one

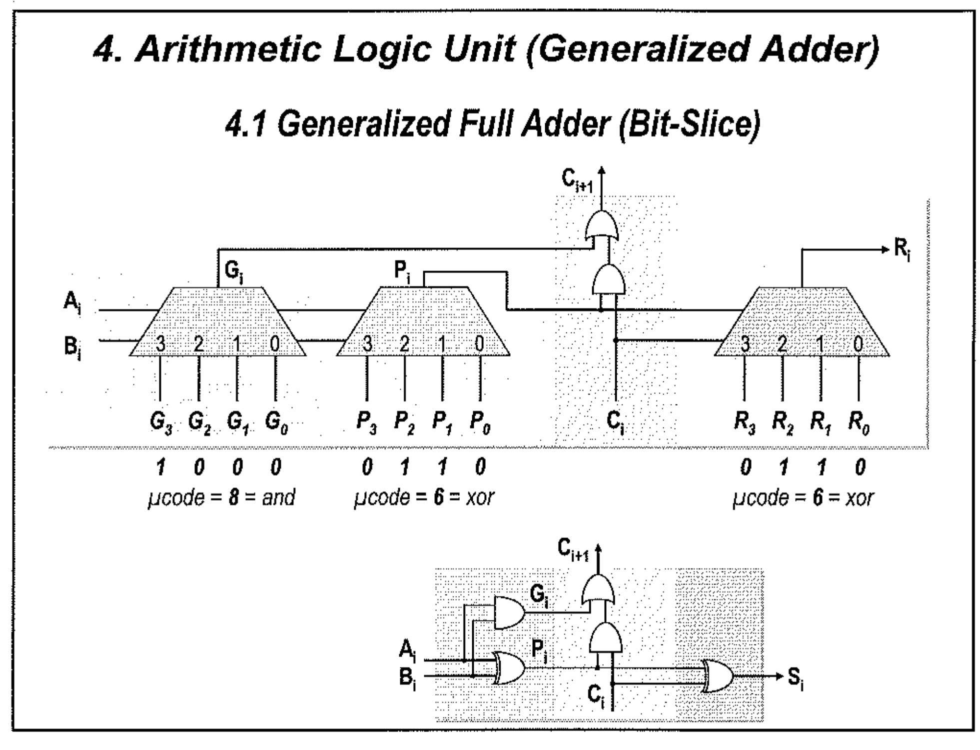



3. Arithmetic Function Unit (Adder)
3.1 Full Adder (Bit-Slice)
FA
A_i
B_i
S_i
C_i
C_{i+1}
G_i
P_i
$C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i$ ⟹ $C_{i+1} = G_i + P_i.C_i$
$S_{i+1} = A_i \oplus B_i \oplus C_i$ ⟹ $S_{i+1} = P_i \oplus C_i$
S_{i+1} | A_i | B_i | C_{i+1}
C_i | 0 | 0 | 0
$\overline{C_i}$ | 0 | 1 | C_i
$\overline{C_i}$ | 1 | 0 | C_i
C_i | 1 | 1 | 1
$K_i = \overline{A_i}.\overline{B_i}$ = kill
$P_i = A_i \oplus B_i$ = propagate
$G_i = A_i.B_i$ = generate

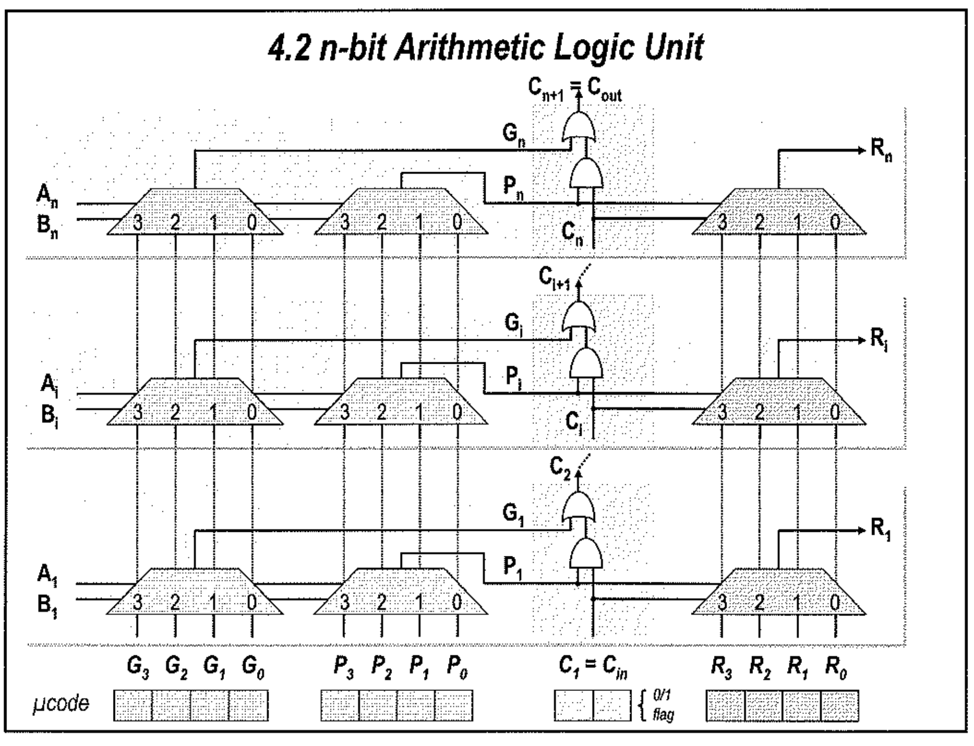

3.2 n-bit Ripple Carry Adder
C_{n+1}
A_n
B_n
P_n
S_n
C_n
C_{i+1}
A_i
B_i
P_i
S_i
C_i
C_2
A_1
B_1
P_1
S_1
C_1
bit-wise logic
n-bit Ripple Carry Adder

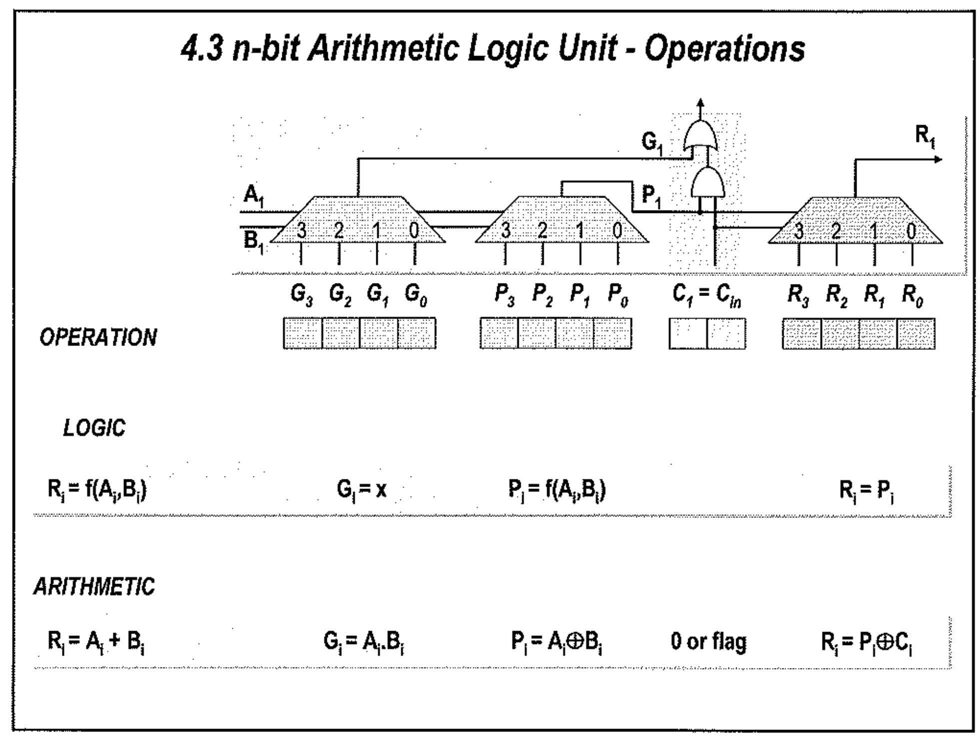


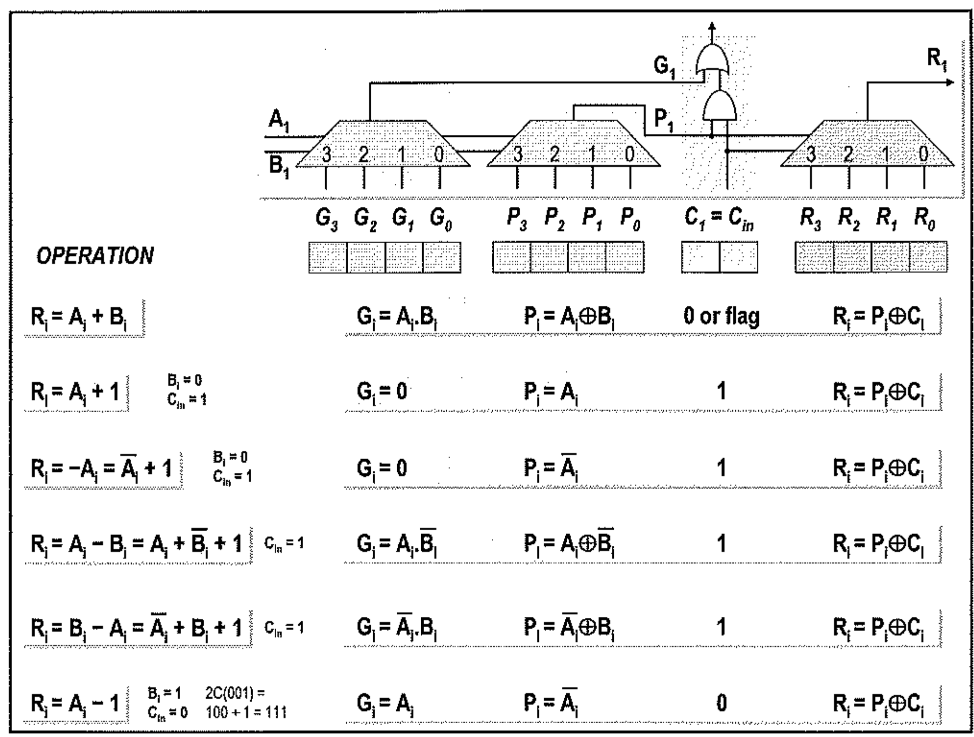
C_{n+1}
G_n
P_n
A_n
B_n
S_n
C_n
C_{i+1}
G_i
P_i
A_i
B_i
S_i
C_i
C_2
G_1
P_1
A_1
B_1
S_1
C_1
bit-wise logic
n-bit Ripple Carry Adder

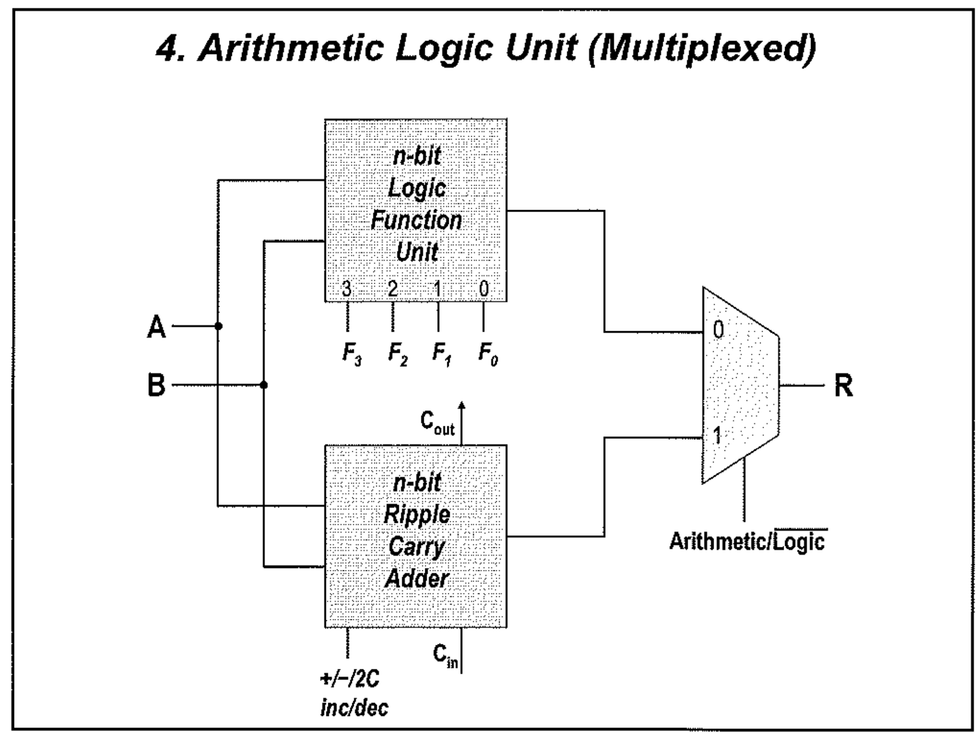


4. Arithmetic Logic Unit (Generalized Adder)
4.1 Generalized Full Adder (Bit-Slice)
C_{i+1}
G_i
P_i
R_i
A_i
B_i
3 2 1 0
G_3 G_2 G_1 G_0
P_3 P_2 P_1 P_0
C_i
R_3 R_2 R_1 R_0
1 0 0 0
μcode = 8 = and
0 1 1 0
μcode = 6 = xor
0 1 1 0
μcode = 6 = xor
S_i


4.2 n-bit Arithmetic Logic Unit
C_{n+1} = C_out
G_n
P_n
C_n
A_n
B_n
R_n
C_{i+1}
G_i
P_i
C_i
A_i
B_i
R_i
C_2
G_1
P_1
A_1
B_1
R_1
G_3 G_2 G_1 G_0
P_3 P_2 P_1 P_0
C_1 = C_in
R_3 R_2 R_1 R_0
µcode
0/1 flag


4.3 n-bit Arithmetic Logic Unit - Operations
G_1
R_1
A_1
P_1
B_1
G_3 G_2 G_1 G_0
P_3 P_2 P_1 P_0
C_1 = C_in
R_3 R_2 R_1 R_0
OPERATION
LOGIC
R_i = f(A_i,B_i)
G_i = x
P_i = f(A_i,B_i)
R_i = P_i
ARITHMETIC
R_i = A_i + B_i
G_i = A_i.B_i
P_i = A_i⊕B_i
0 or flag
R_i = P_i⊕C_i

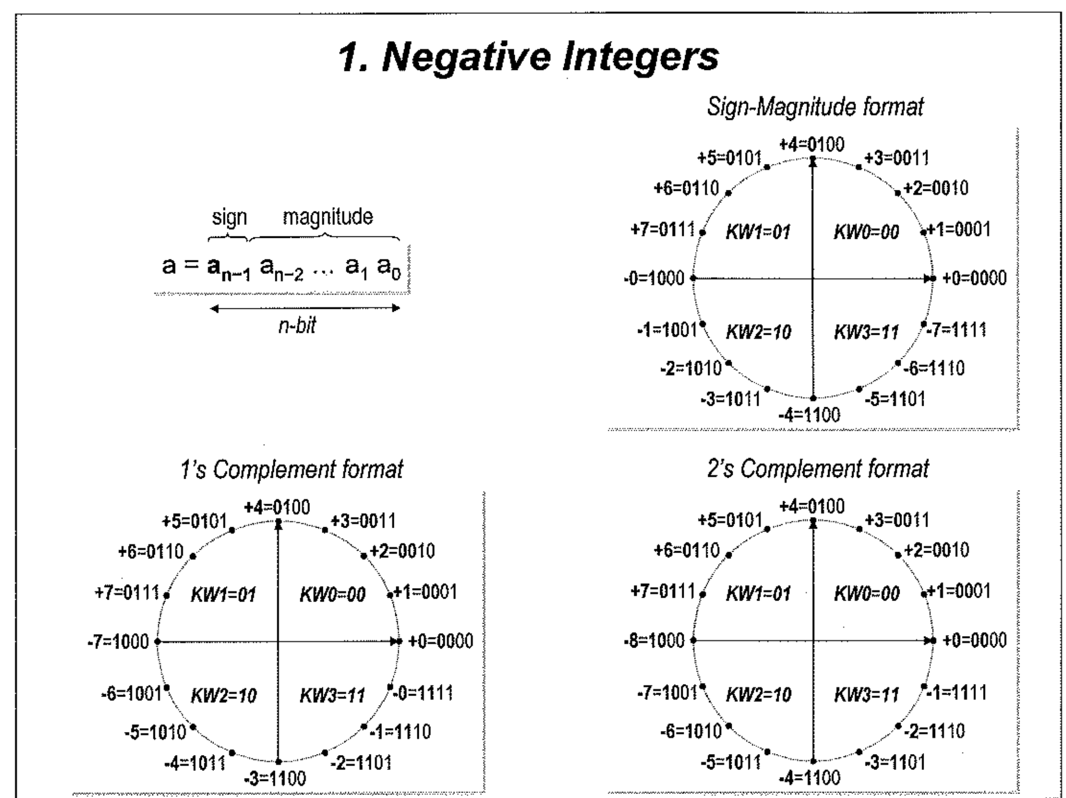

G_1
R_1
A_1
P_1
B_1
3 2 1 0
3 2 1 0
3 2 1 0
G_3 G_2 G_1 G_0
P_3 P_2 P_1 P_0
$C_1 = C_{in}$
R_3 R_2 R_1 R_0
OPERATION
$R_i = A_i + B_i$ | $G_i = A_i.B_i$ | $P_i = A_i \oplus B_i$ | 0 or flag | $R_i = P_i \oplus C_i$
$R_i = A_i + 1$ $B_i = 0$ $C_{in} = 1$ | $G_i = 0$ | $P_i = A_i$ | 1 | $R_i = P_i \oplus C_i$
$R_i = -A_i = \overline{A_i} + 1$ $B_i = 0$ $C_{in} = 1$ | $G_i = 0$ | $P_i = \overline{A_i}$ | 1 | $R_i = P_i \oplus C_i$
$R_i = A_i - B_i = A_i + \overline{B_i} + 1$ $C_{in} = 1$ | $G_i = A_i.\overline{B_i}$ | $P_i = A_i \oplus \overline{B_i}$ | 1 | $R_i = P_i \oplus C_i$
$R_i = B_i - A_i = \overline{A_i} + B_i + 1$ $C_{in} = 1$ | $G_i = \overline{A_i}.B_i$ | $P_i = \overline{A_i} \oplus B_i$ | 1 | $R_i = P_i \oplus C_i$
$R_i = A_i - 1$ $B_i = 1$ $C_{in} = 0$ 2C(001) = 100 + 1 = 111 | $G_i = A_i$ | $P_i = \overline{A_i}$ | 0 | $R_i = P_i \oplus C_i$


4. Arithmetic Logic Unit (Multiplexed)
n-bit Logic Function Unit
3 2 1 0
F_3 F_2 F_1 F_0
A
B
C_{out}
n-bit Ripple Carry Adder
C_{in}
+/–/2C inc/dec
0
1
R
Arithmetic/$\overline{\text{Logic}}$

HOGESCHOOL VOOR WETENSCHAP & KUNST | DE NAYER INSTITUUT
SINT-KATELIJNE-WAVER

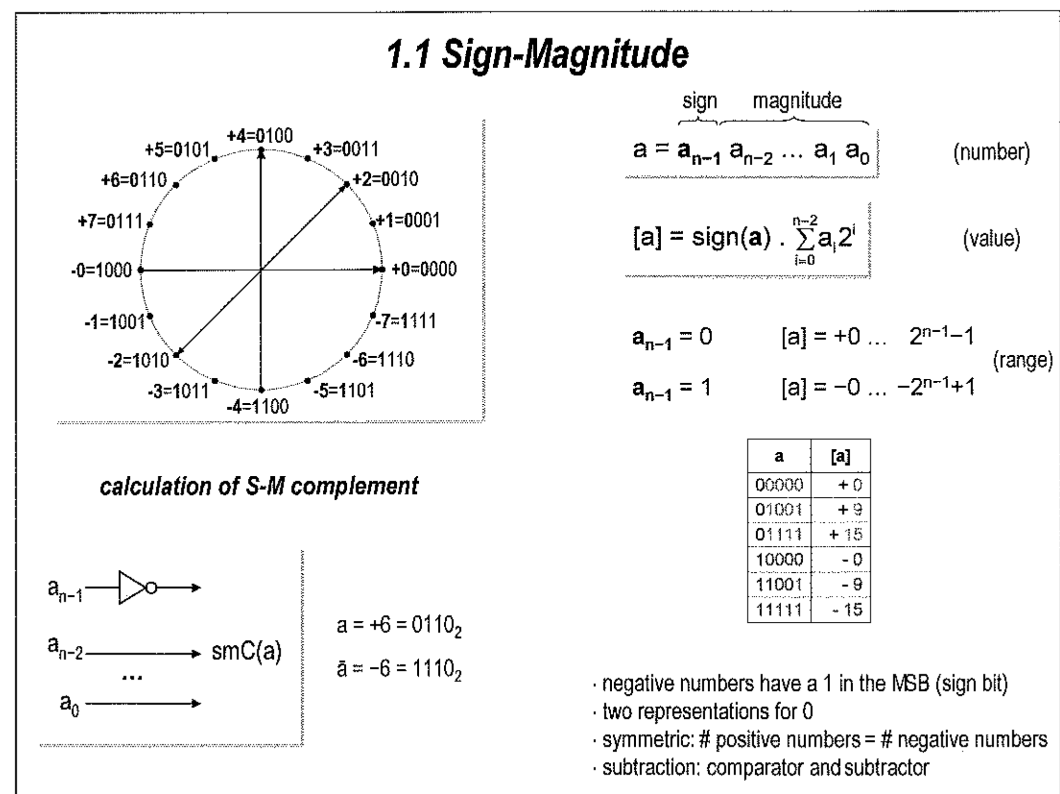
Digitale Synthese Arithmetic Functions MAC

ir. J. Meel
march 2007



In mathematics, there are infinitely many positive and negative integers. However, in a practical hardware system only a fixed number of integers can be represented, based on the number of bits allocated to the representation.

As shown, 4-bit binary quantities can represent 16 unique binary numbers. An *overflow* occurs when an arithmetic operation results in a number outside the range of those that can be represented. Roughly half of these will represent positive numbers and zero, while the remainder will be negative numbers. Each of the three representation schemes (sign-magnitude, ones complement, twos complement) handles negative numbers slightly differently.



a	[a]
00000	+0
01001	+9
01111	+15
10000	-0
11001	-9
11111	-15

In *sign-magnitude* systems, the most significant bit (a_{n-1}) represents the number's sign, while the remaining bits represent its absolute value as an unsigned binary magnitude. If the sign bit a_{n-1} is a 0, the number is positive. If the sign bit a_{n-1} is a 1, the number is negative.

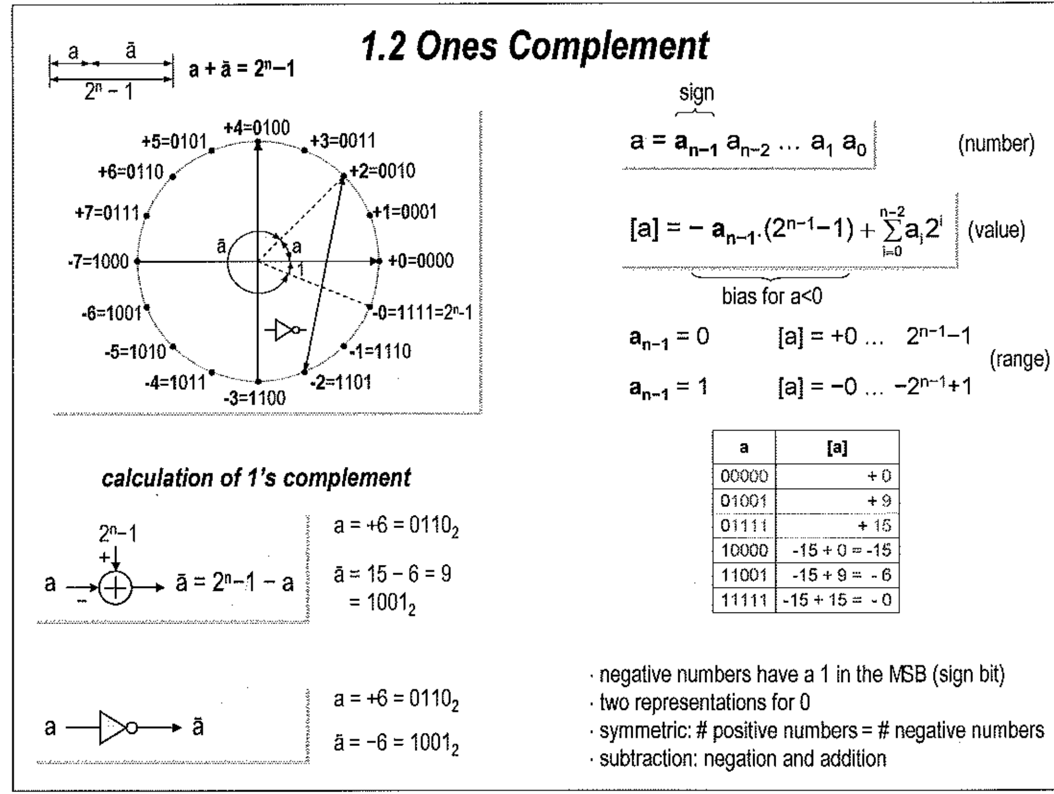
Number wheel

A 4-bit sign-magnitude number system is represented on a number wheel. The figure shows the binary numbers and their decimal integer equivalents, assuming that the numbers are interpreted as sign and magnitude. The largest positive number that can be represent in three data bits is $+7 = 2^3 - 1$. By a similar calculation, the smallest negative number is -7. There is an equal number of positive and negative numbers (symmetric).

Zero has two different representations, even though +0 and -0 don't make much sense mathematically.

Calculation of the sign-magnitude complement

Negation of a number is done simply by replacing the sign bit with its complement. The magnitude value stays unchanged.



a	[a]
00000	+0
01001	+9
01111	+15
10000	-15 + 0 = -15
11001	-15 + 9 = -6
11111	-15 + 15 = -0

A ones complement approach represents the positive numbers just as in the sign-magnitude representation. The only difference is in how it represents negative numbers.

Number wheel

The number wheel representation of the 4-bit ones complement number system is shown. All negative numbers have a 1 in their sign bit, making it easy to distinguish between positive and negative numbers.

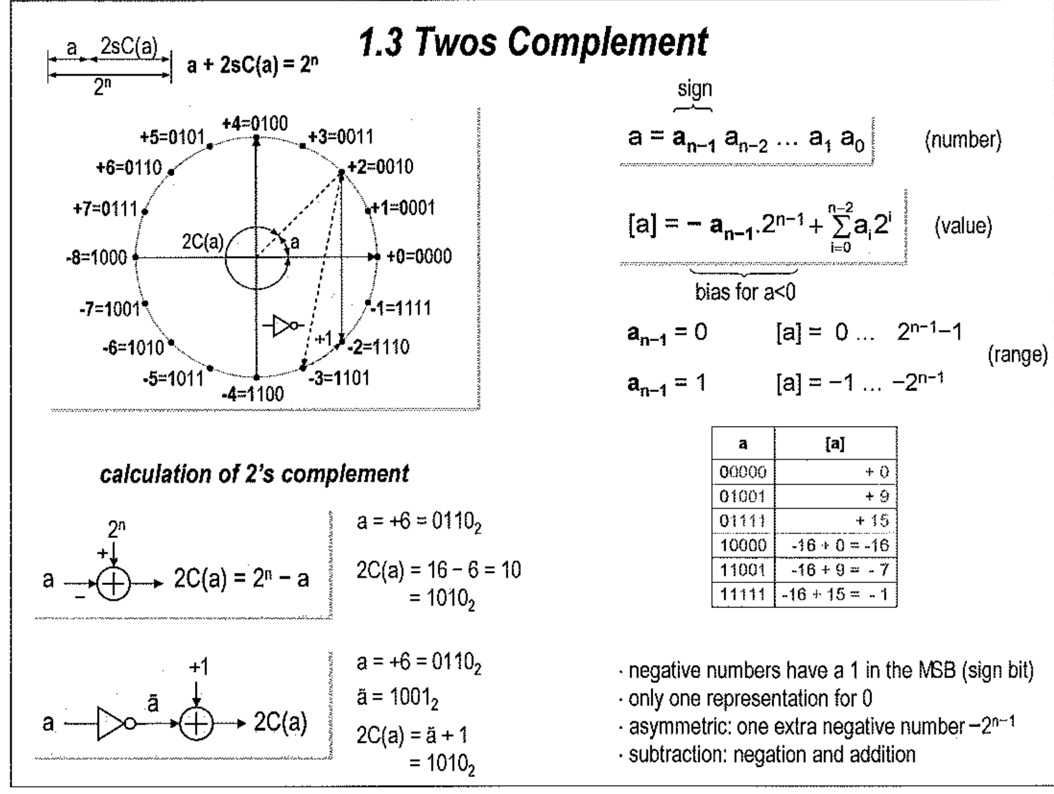
There are still two different representations of zero.

Calculation of the ones complement

The first procedure to derive a negative ones complement integer, denoted ā, from a positive integer, denoted a. If the word length is n bits (n = 4 in the case shown), then $\bar{a} = (2^n - 1) - a$. For example, in a 4-bit system, a=+7 is represented as 0111. The ones complement ā=−7 can be computed as:

$2^n=2^4$	10000
subtract 1	- 0001
	1111
subtract 7	- 0111
	1000 representation of −7

This rather complicated method is just one way to compute the negative of a ones complement number. A simpler method forms the ones complement by taking the number's bitwise complement (bitwise inversion). Thus, +7 = 0111 and −7 = 1000, +4 = 0100 and −4 = 1011, and so on.



A twos complement approach represents the positive numbers just as in the sign-magnitude representation. The only difference is in how it represents negative numbers.

Number wheel

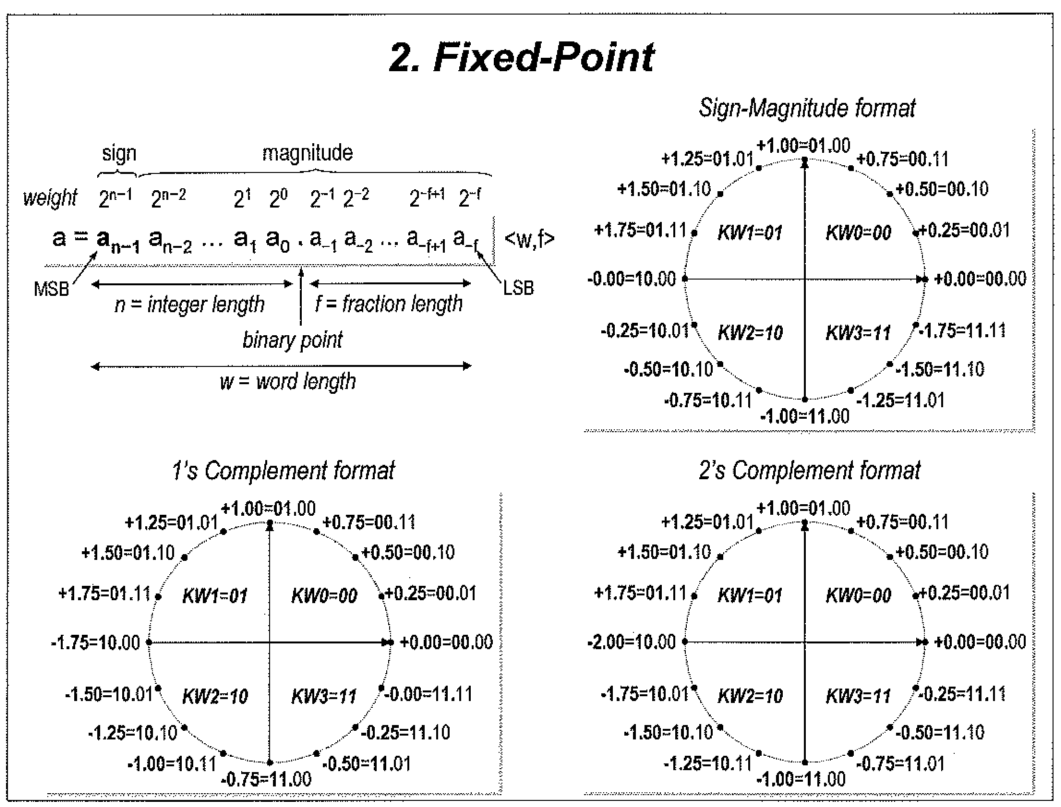
The twos complement scheme is similar to ones complement. The twos complement numbers can be derived from the ones complement representation by shifting the negative numbers one position in the clockwise direction. By this operation there is only one representation for zero.
This operation also results in one extra negative number that can now be represented: -2^{n-1} (−8 in the 4-bit case). This makes the twos complement representation asymmetric.
The negative numbers still have a 1 in their highest-order bit, the sign bit.

Calculation of the twos complement

A twos complement negative number, denoted 2C(a) is derived from its positive number a, by the equation $2C(a) = 2^n - a$, where n is the number of bits in the representation. This equation omits the ones complement step that subtracts 1 from 2^n.
For example the twos complement of +7, can be calculated as:

$2^n=2^4$	10000	
subtract 7	- 0111	
	1001	representation of −7

The same shortcut used to find ones complement numbers also applies for the twos complement system, but with a twist. The number wheel suggests the scheme. Simply form the bitwise complement of the number and then add one to form its twos complement.
For example, +7 = 0111_2, its bitwise complement is 1000_2. plus 1 is 1001_2. This is the same twos complement representation of −7 as derived by the last calculation.
For the number −4, represented as 1100_2, its bitwise complement is 0011_2, plus 1 is 0100_2, This is exactly the twos complement representation of +4.
A third method two calculate the twos complement. Leave all least significant 0's and the first 1 unchanged, then bit-wise complement the remaining bits.

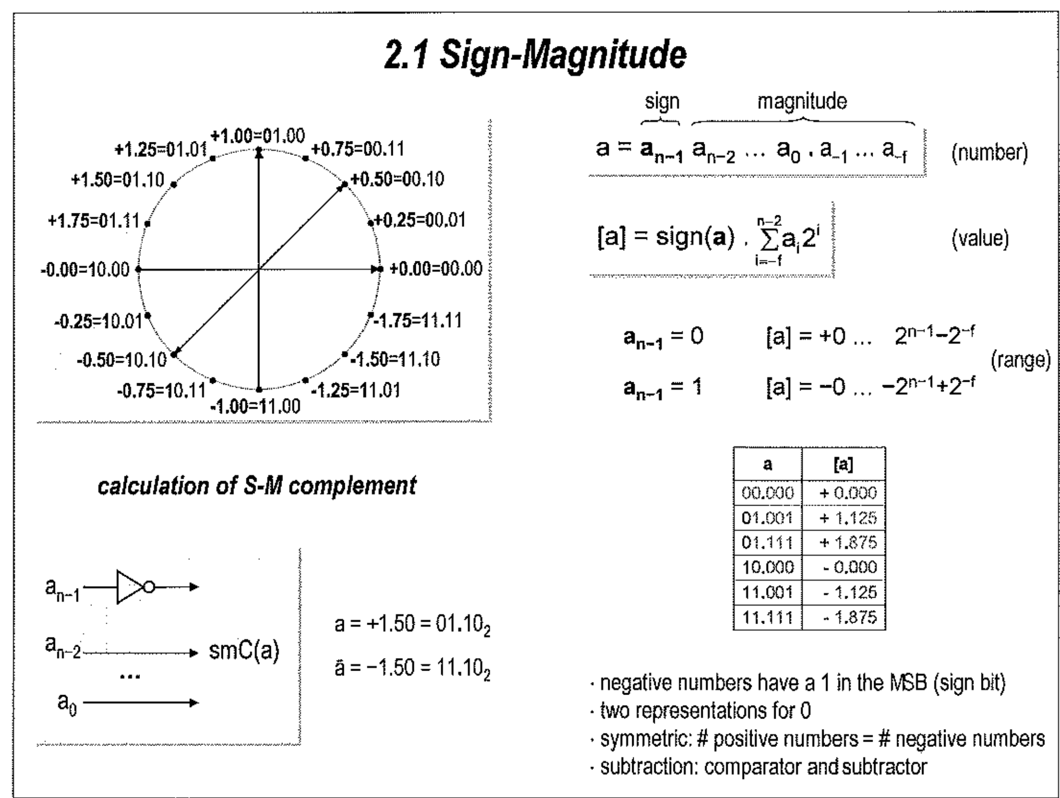


A binary number comes with a binary point. By moving the implicit "binary" point at the right of an integer number, fractions can be represented too.

The portion to the left represents an integer and the portion to the right represents fractions less than one. In fixed-point notation, the binary point is constrained to lie at a fixed position in the bit pattern, as shown.

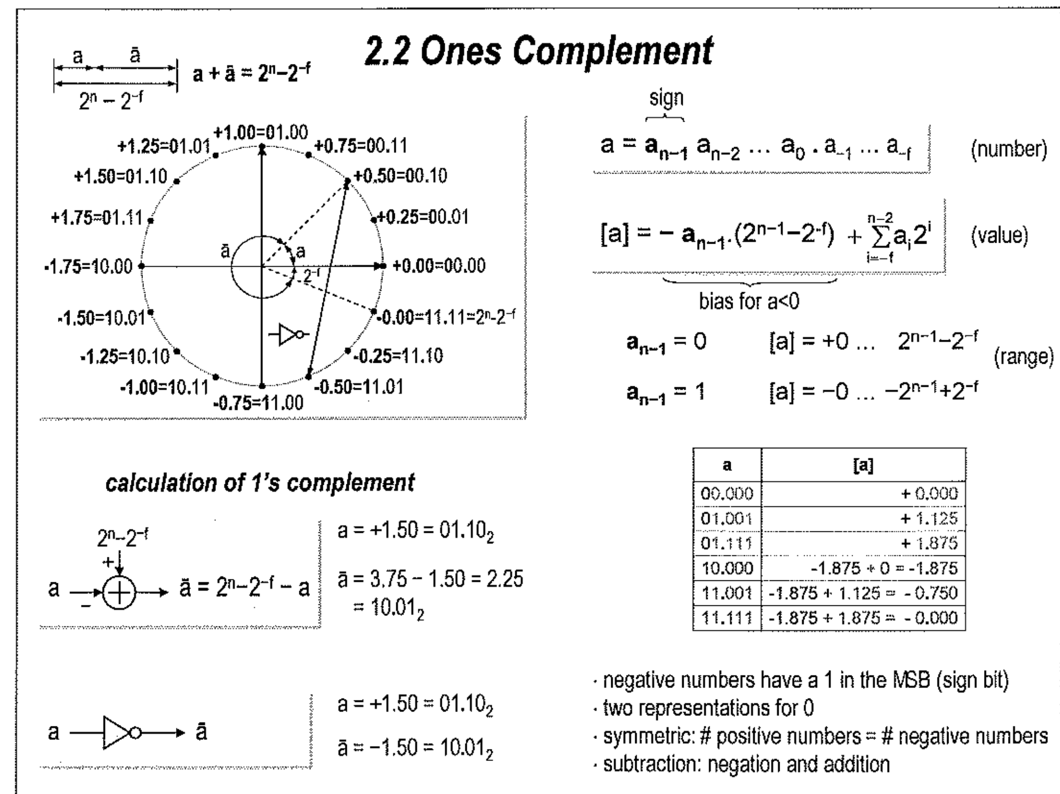
The first bit to the left is called the sign bit. The precision of the number system is defined as the increment between two consecutive numbers and is determined by the value of the least significant bit (LSB).

Within the subclass of fixed-point representations, there are tree commonly used methods to represent binary numbers: sign-magnitude, one's complement and two's complement representation. They differ in the way they handle negative numbers.



a	[a]
00.000	+ 0.000
01.001	+ 1.125
01.111	+ 1.875
10.000	- 0.000
11.001	- 1.125
11.111	- 1.875

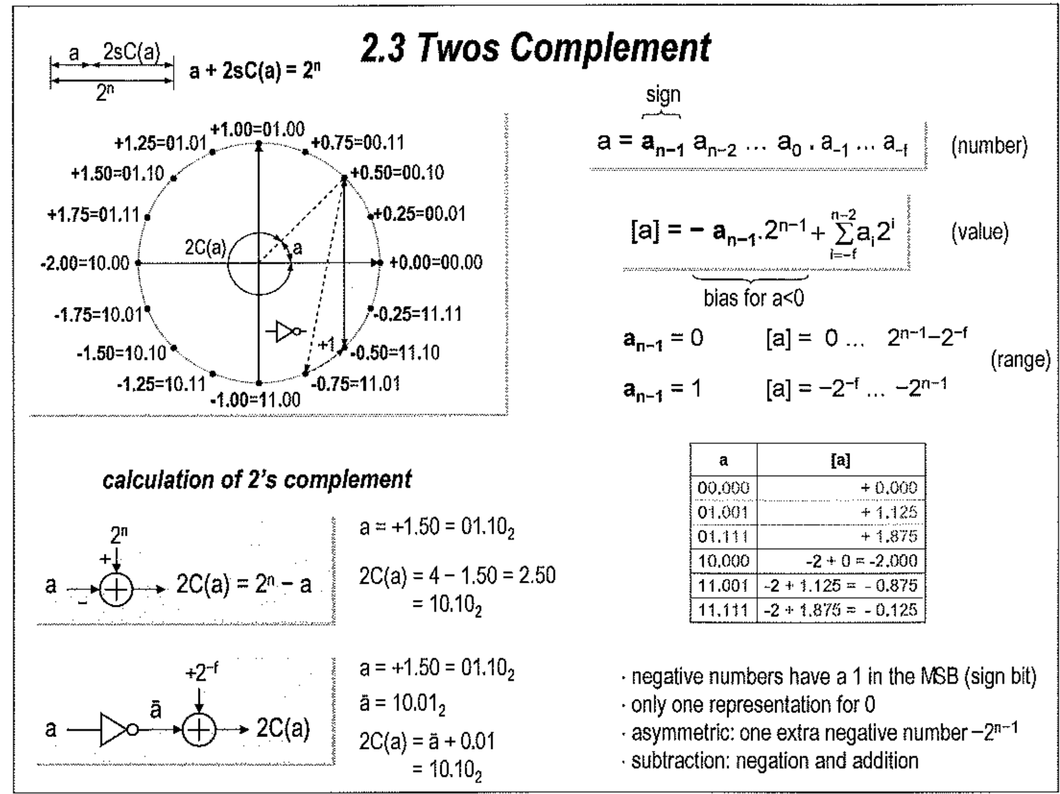
In the *sign-magnitude* representation, the most significant bit (a_{n-1}) represents the number's sign, while the remaining bits represent its absolute value as an unsigned binary magnitude. If the sign bit a_{n-1} is a 0, the number is positive. If the sign bit a_{n-1} is a 1, the number is negative.



a	[a]
00.000	+ 0.000
01.001	+ 1.125
01.111	+ 1.875
10.000	-1.875 + 0 = -1.875
11.001	-1.875 + 1.125 = - 0.750
11.111	-1.875 + 1.875 = - 0.000

The *ones complement* representation is identical to the sign-magnitude representation for positive numbers. The only difference is in how it represents negative numbers. A negative number is formed by complementing its corresponding positive number representation.

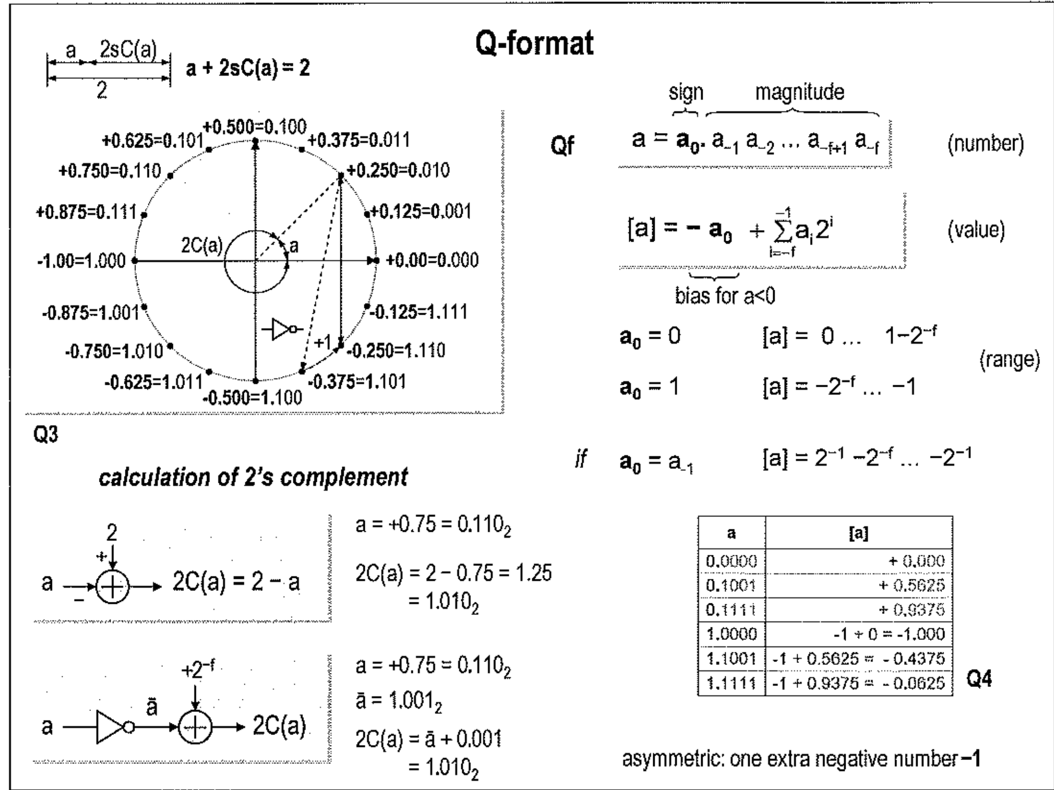
Note that zero is now represented by 00...0.0...00 or 11...1.1...11, which is an undesired ambiguity.



a	[a]
00.000	+ 0.000
01.001	+ 1.125
01.111	+ 1.875
10.000	-2 + 0 = -2.000
11.001	-2 + 1.125 = - 0.875
11.111	-2 + 1.875 = - 0.125

In the *twos complement* representation a positive numbers is identical as in the sign-magnitude representation and ones complement representation. The only difference is in how it represents negative numbers. The negative number is obtained by subtracting the corresponding positive number from 2^n.

If a two's-complement representation is used for signed fixed-point numbers, the same binary addition procedure will work for adding both signed and unsigned numbers.



The Q-format is a special case of the twos complement fixed-point representation. It is a pure *fractional representation.*

Qf-format is a popular format in DSP. The most significant bit is the sign bit followed by an imaginary binary point, followed by f bits of fraction. The Qf number has a range between -1 and $1 - 2^{-f}$.

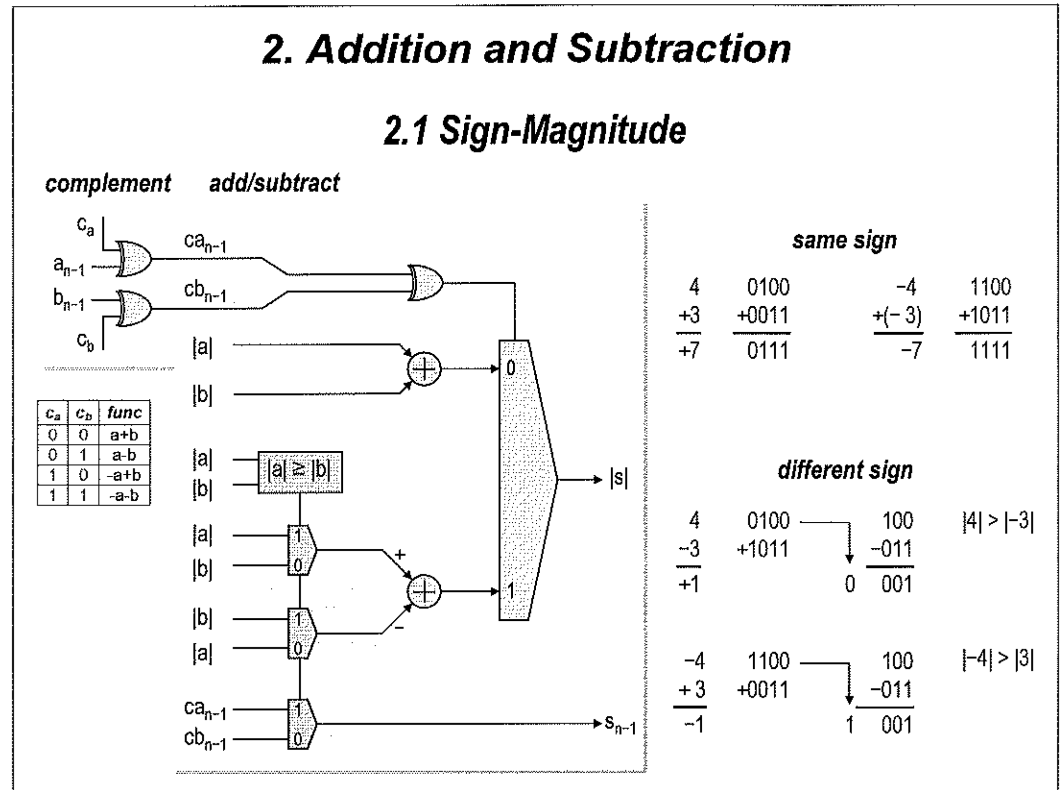
For example, the Q15 format contains 16 bit: 1 sign bit and 15 fraction bits.

$$0.111111111111111 = 1 - 2^{-15} = 0.99996948242188$$

$$0.100000000000000 = 0.5$$

$$1.000000000000000 = -1$$

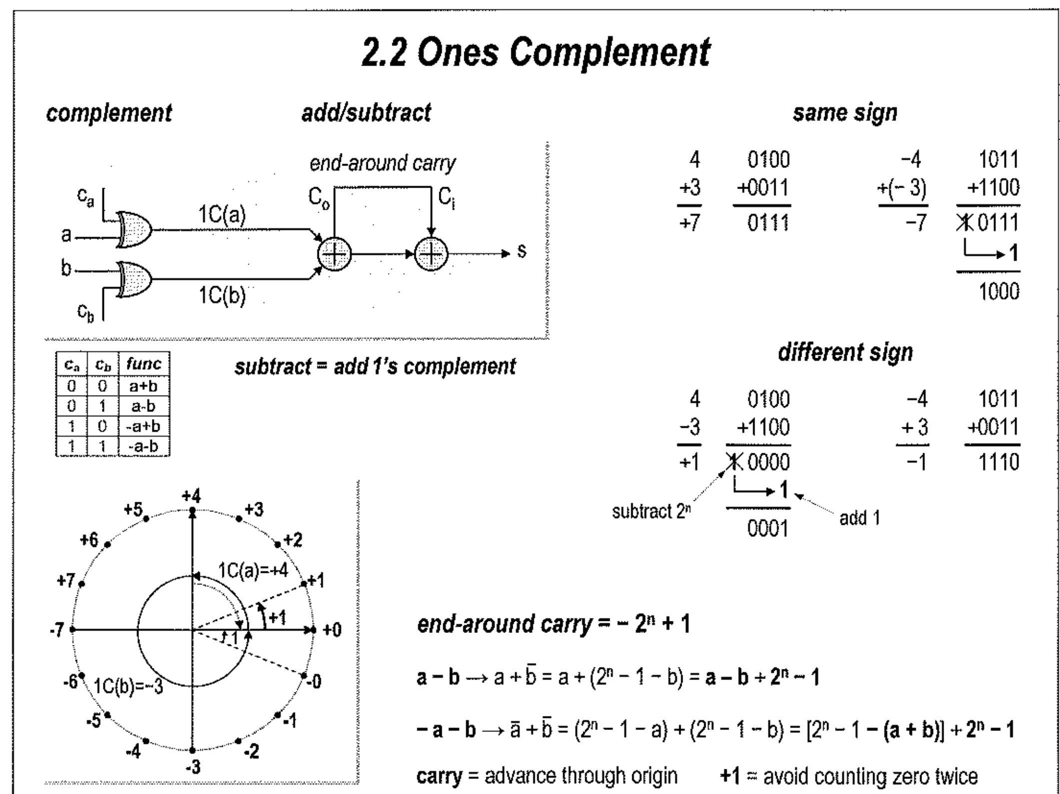
A negative number is obtained by subtracting its positive number from 2, hence the name 2's complement.



Adding two positive or two negative numbers is straightforward. Simply perform the addition and assign the result the same sign as the original operands. When the signs of the two operands are not the same, addition becomes more complex. In this case, the smaller magnitude should be subtracted from the larger. The resulting sign is the same as that of the number with the larger magnitude.

This is what makes arithmetic operations with sign-magnitude numbers so cumbersome. Any adder circuit must also include a subtractor and a comparator. The adder is used when the signs are the same, the subtractor when they differ. Subtraction is just as complicated.

Because of this burdensome complexity, hardware designers have proposed other schemes for representing negative numbers.



The advantage of ones complement numbers is the ease with which negative numbers can be computed. Subtraction is implemented by a combination of addition and negation: A − B = A + (−B). Thus, a separate subtractor circuit is not needed. However, addition is still complicated by the two zeros. This problem is solved with the twos complement representation.

End-around carry

In ones complement, −4 is represented as 1011_2 and −3 as 1100_2. When these two numbers are added, a carry-out of the high-order bit position is generated. Whenever this occurs, the carry bit must be added to the result of the sum. $1011_2 + 1100_2$ yields 1 0111_2. Adding the carry-out as an LSB to the 4-bit sum, gives $0111_2 + 1 = 1000_2$. This is the representation of −7 in ones complement.

The end-around carry also happens for the sum of $4(0100_2)$ and $-3(0011_2)$. This yields 1 0000_2. Adding in the carry gives 0001_2, the ones complement representation of -1.

Why does the end-around carry scheme work?

Intuitively, the carry-out of 1 means that the resulting addition advances through the origin of the number wheel. In effect, the result must be advanced by 1 to avoid counting zero twice.

More formally, the operation of the end-around carry is the equivalent of subtracting 2^n and adding 1. Consider the computation of the sum b + (−a) where b > a:

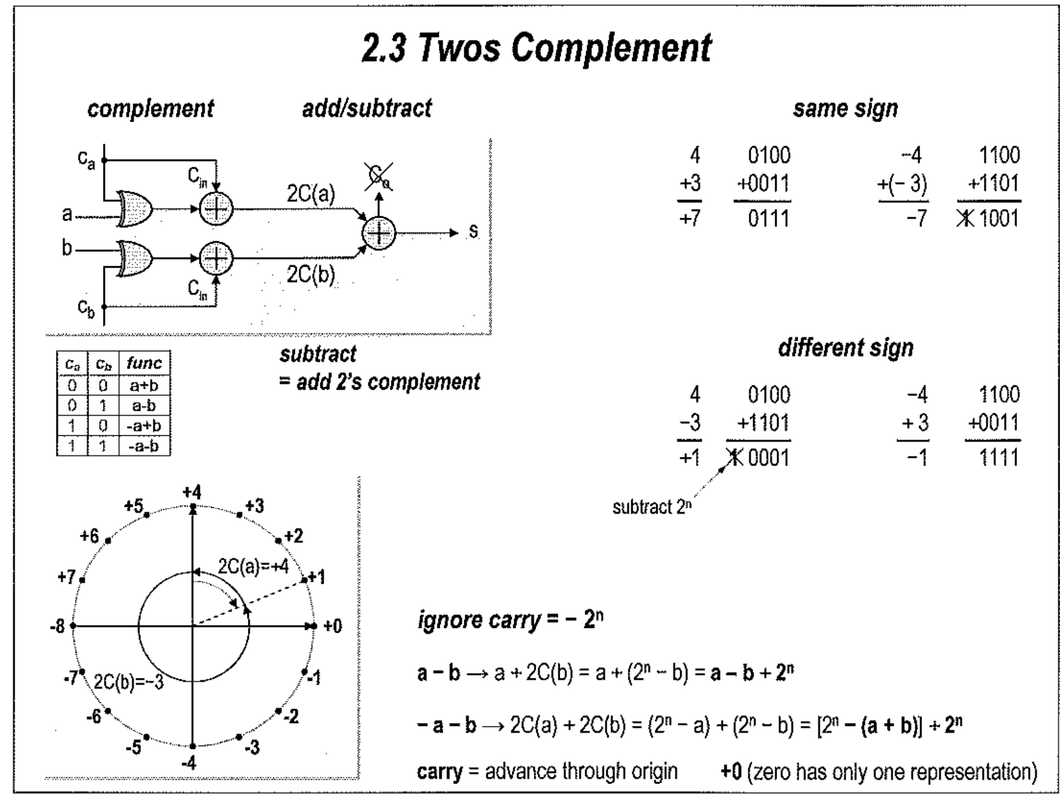
$$b - a = b + \bar{a} = b + (2^n - 1 - a) = (b - a) + 2^n - 1$$

This is exactly the situation of last example. The end-around carry subtracts off 2^n and adds 1, yielding the desired result of b − a.

Now consider the case shown in the first example. The sum to be formed is −a + (−e), where a + e is less than $2^n - 1$. This results in the following sequence of equations:

$$-a + (-e) = \bar{a} + \bar{e} = (2^n - 1 - a) + (2^n - 1 - e) = [2^n - 1 - (a + e)] + 2^n - 1$$

After the end-around carry (subtract off 2^n and add 1), the result of the sum becomes $[\, 2^n - 1 - (a + e)\,]$. This is the correct form for representing −(a + e) in ones complement form.



Twos complement calculations behave very much like the ones complement method, but without the end-around carry. Subtraction is handled as before: negate the operand and perform addition. Carry-outs can still occur, but in twos complement arithmetic they are ignored.

Ignore carry-out

Summing two positive numbers, is identical to the two previous representation schemes. Summing two negative numbers is also straightforward. Simply perform binary addition, ignoring any carry-outs. Since there are no longer two representations for zero, there is no need to worry about correcting the summation. Mixed addition of positive and negative numbers is handled exactly like the other cases.

Why is it all right to ignore the carry-out?

Consider the sum b + (−a) where b > a. This can be rewritten as:

$$b + 2C(a) = b + (2^n - a) = 2^n + (b - a)$$

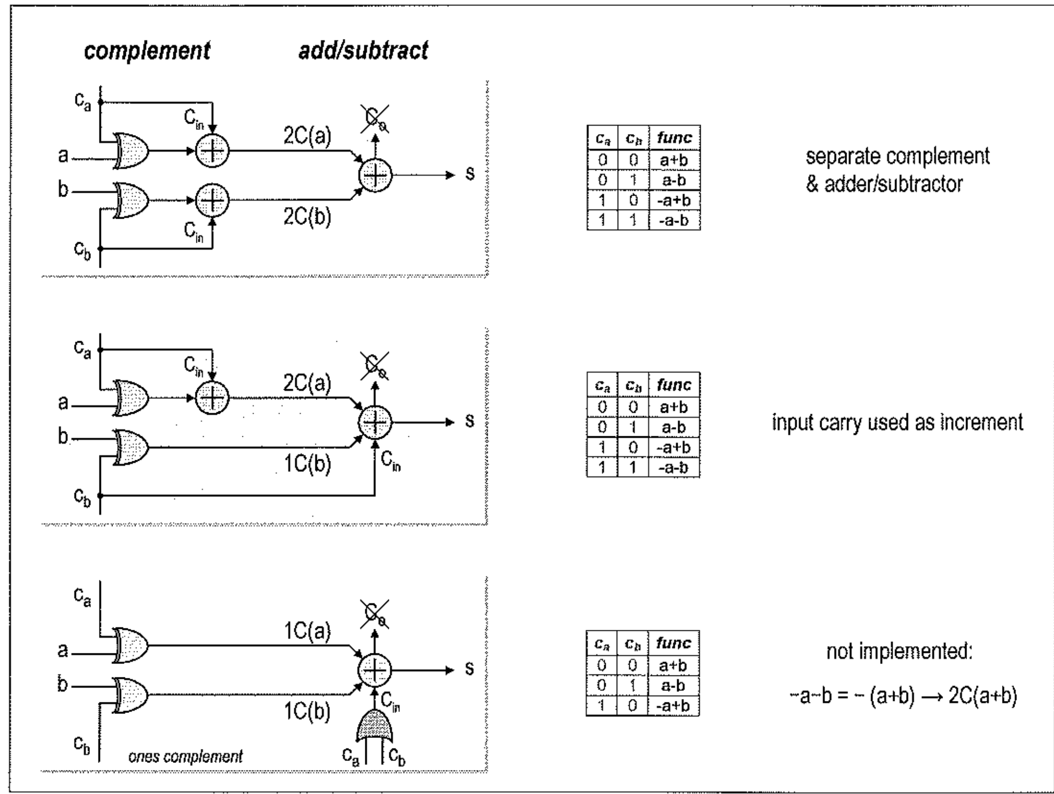
Ignoring the carry-out is equivalent to subtracting 2^n. Doing this to the foregoing expression yields the result b − a.

Consider the sum: (−a) + (−b) where a+ b is less than or equal to 2^n -1. This can be rewritten as:

$$2C(a) + 2C(b) = (2^n - a) + (2^n - b) = [2^n - (a + b)] + 2^n$$

By subtracting 2^n, the resulting form is exactly the representation of 2C(a + b), the desired twos complement representation of −(a + b).

The trade-off between twos complement and ones complement arithmetic should now be clearer. In the twos complement case, addition is simple but negation is more complex. For the ones complement system, it is easy to perform negation but addition becomes more complicated. Because twos complement only has one representation for zero, it is preferred for most digital systems.

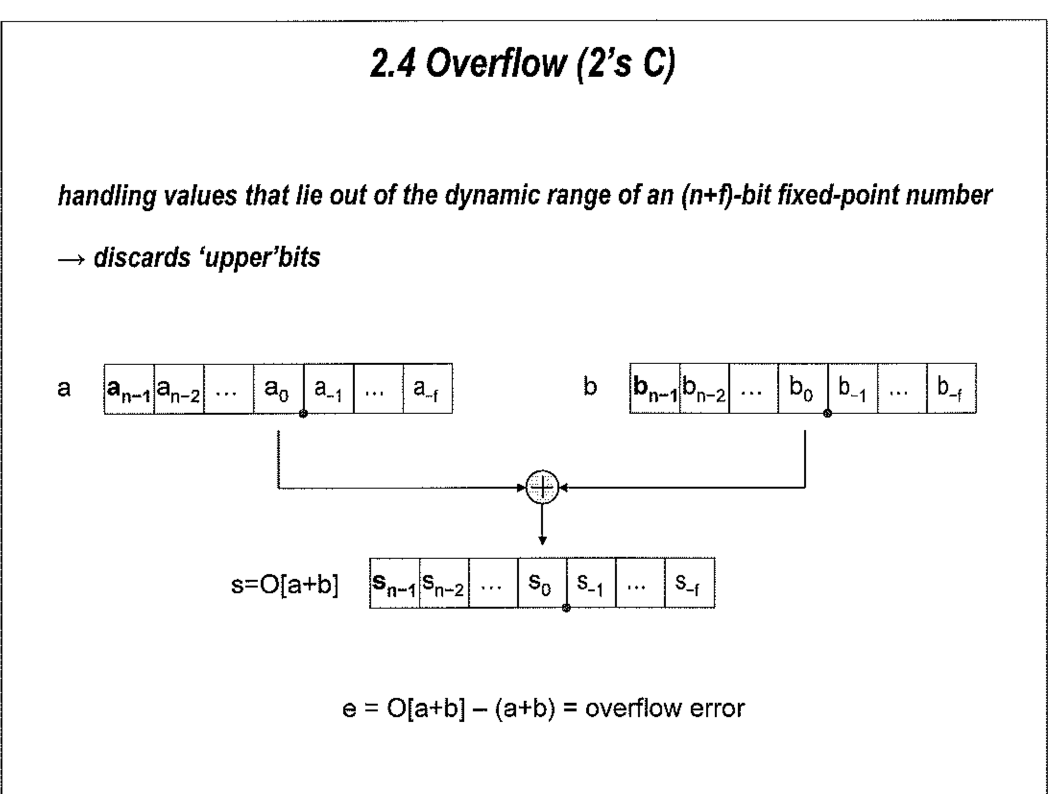


c_a	c_b	func
0	0	a+b
0	1	a-b
1	0	-a+b
1	1	-a-b

c_a	c_b	func
0	0	a+b
0	1	a-b
1	0	-a+b
1	1	-a-b

c_a	c_b	func
0	0	a+b
0	1	a-b
1	0	-a+b

Complementing a twos complement number requires the negating of each bit (EXOR) and adding an LSB. When only one of the numbers (a or b) must be complemented, this addition of an LSB can be implemented as setting the input carry of the final addition to '1'. This results in hardware savings.



Overflow occurs when the number of bits required to represent a number exceed the number of bits available. So, an overflow occurs when the value of a signal exceeds the dynamic range available. The dynamic range is the range of numbers which can be represented within the arithmetic used. In twos complement fixed-point arithmetic, this range is given by:

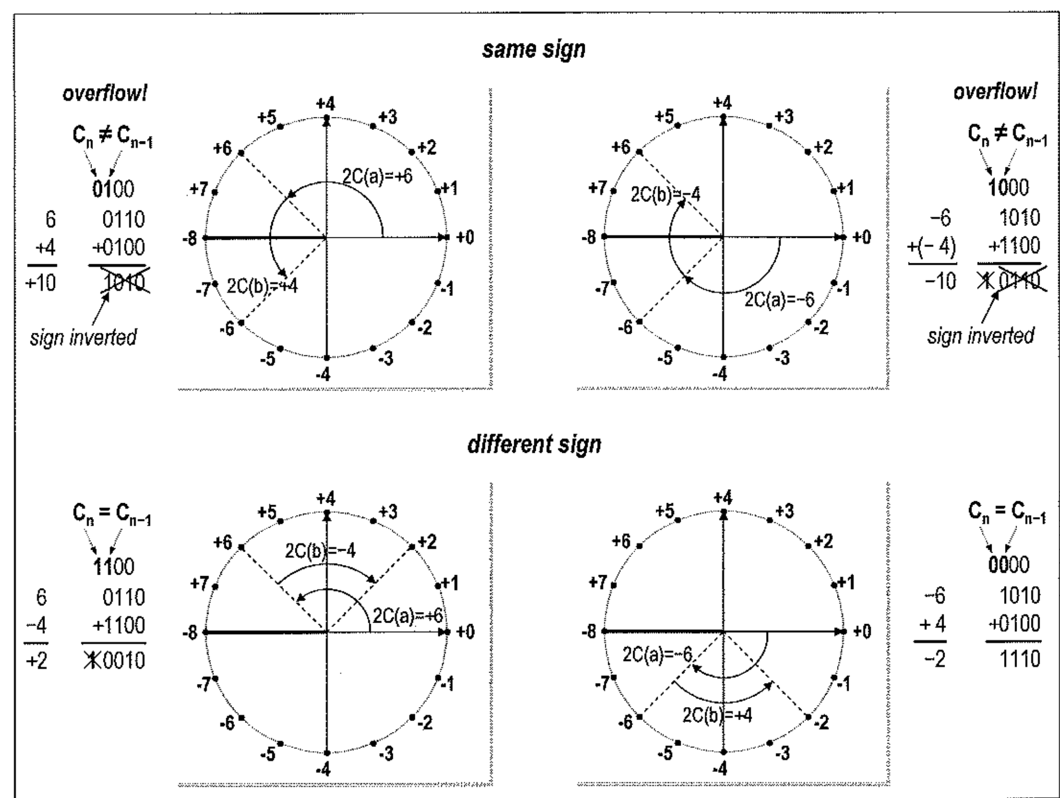
$$[\min, \max] = [\,-2^{n-1}, 2^{n-1} - 2^{n-f}\,]$$

Obviously, overflow errors are bad because they introduce big errors.

Overflow occurs whenever the sum of two positive numbers yields a negative result or when two negative numbers are summed and the result is positive.

Remark:

Scaling is the process of readjusting some internal gain parameters to avoid overflows. For a fixed total number of bits, there is a trade-off between decreasing the probability of overflow and increasing the relative quantisation error. Therefore, scaling is usually applied only to minimize the probability of overflow to a reasonable extent and not to preclude it entirely. When an overflow occurs, the resulting distortion is minimized by using overflow arithmetic.



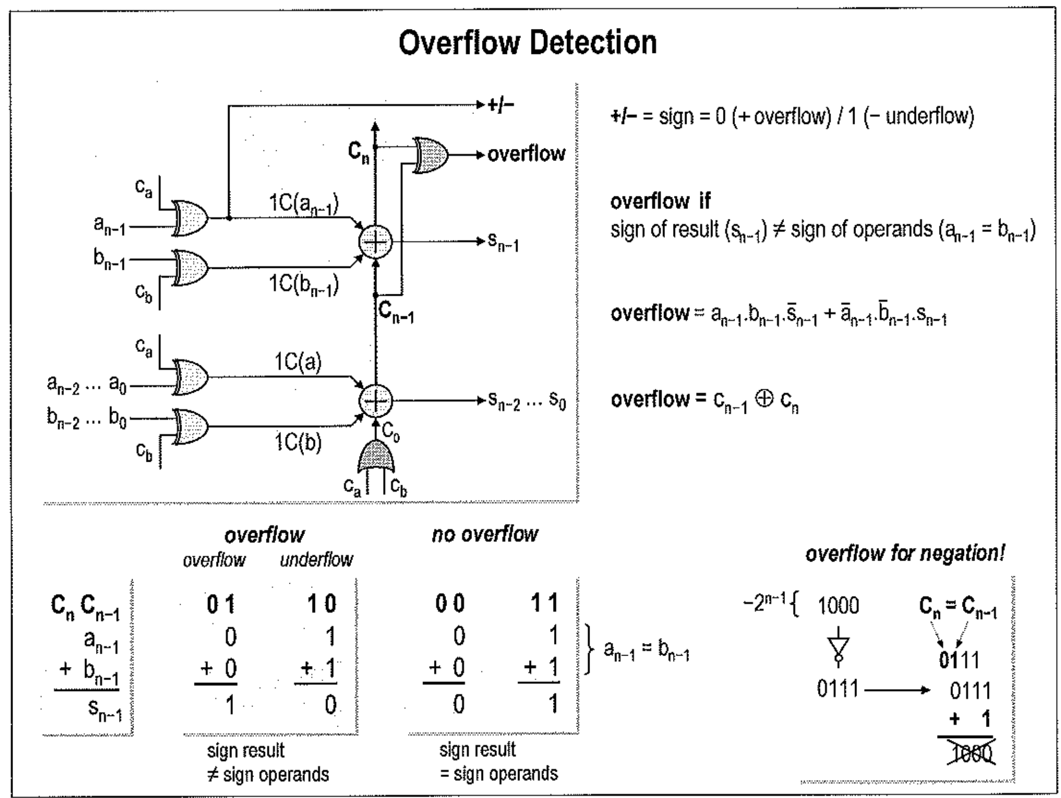
The number wheel can be used to illustrate overflow. Think of addition as moving clockwise around the number wheel. Subtraction moves counter clockwise. Using the twos complement number representation, the number wheel can be divided into two halves, one representing positive numbers (and zero), the other representing the negative numbers. Whenever addition or subtraction crosses the positive/negative line, an overflow has occurred.

This concept is illustrated in the figure, with the two example calculations 6 + 4 and −6 + (−4).

For the sum 6 + 4, start on the number wheel with the representation for +6, advance for numbers in the clockwise direction. This yields −6; an overflow has occurred.

Similarly for subtraction. For the sum −6 + (−4), start with the representation for −6, move for numbers in the counter clockwise direction, obtaining the representation for +6. Once again, an overflow has occurred.

In general, overflow occurs when the carry-in and carry-out of the sign bit are different.



An overflow can be detected in two ways.

First, an overflow has occurred whenever the signs of the operands are the same and the sign of the sum does not agree with the signs of the operands. In an n-bit adder, overflow can be defined as:

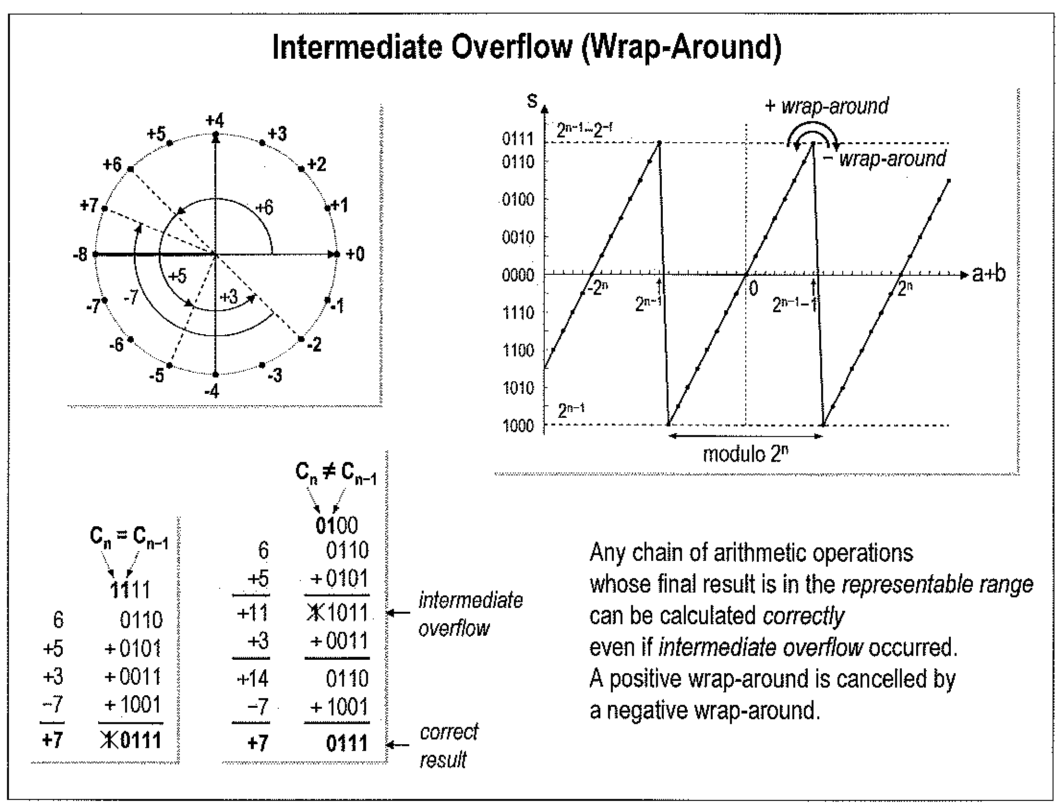
$$\text{overflow} = a_{n-1} \cdot b_{n-1} \cdot \bar{s}_{n-1} + \bar{a}_{n-1} \cdot \bar{b}_{n-1} \cdot s_{n-1}$$

So, overflows occurs if the two operands are positive and the sum is negative, or if the two operands are negative and the sum is positive.

Secondly, if the carry-out of the high-order numeric (magnitude) position of the sum and the carry-out of the sign position of the sum agree, the sum is satisfactory; if they disagree, an overflow has occurred:

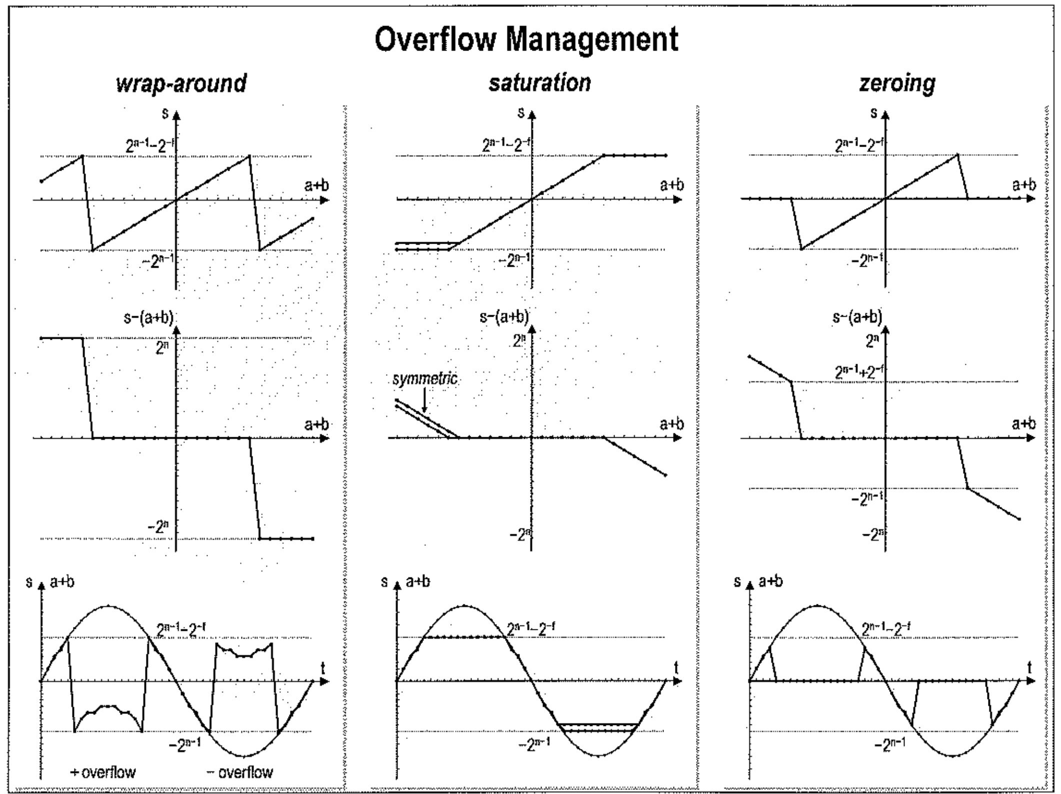
$$\text{overflow} = c_{n-1} \oplus c_n$$

A parallel adder adds the two operands, including the sign bits. An overflow from the magnitude part will tend to change the sign of the sum.



Two's complement wrap-around introduces large errors, but it has the advantage that if the sum of several numbers is within the dynamic range [min, max], the final result will be correct, even if intermediate sums overflow!

However, wrap-around overflow leaves IIR systems susceptible to zero-input large-scale limit cycles.



In twos complement fixed-point arithmetic, the dynamic range is given by:

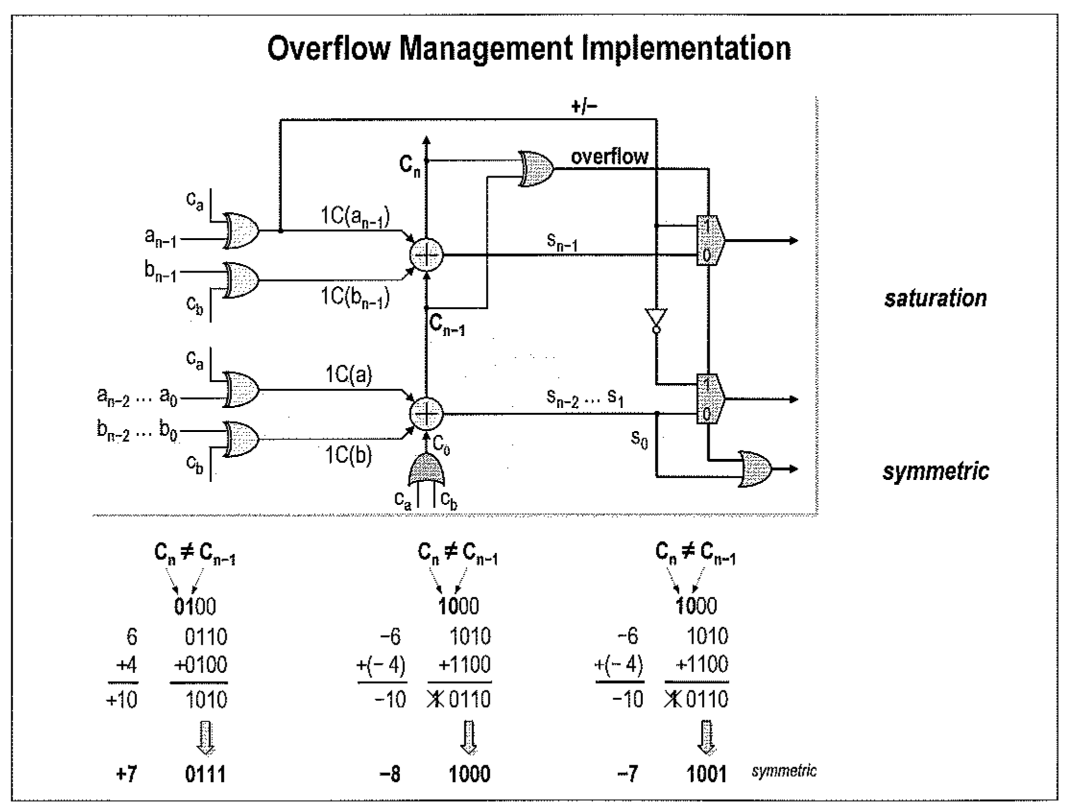
$$[\min, \max] = [-2^{n-1}, 2^{n-1} - 2^{n-f}]$$

Saturation (or clipping)

When the result of an arithmetic operation exceeds the range of the destination type, saturation chooses the nearest representable value (as opposed to a nonsense wrap-around value obtained by dropping upper bits).

The overflow result is substituted by the values min or max, according to its sign.

Two's complement wrap-around introduce more error than saturation, but is easier in terms of hardware.



An overflow has occurred if the carry-out of the high-order numeric (magnitude) position of the sum and the carry-out of the sign position of the sum disagree :

$$\text{overflow} = c_{n-1} \oplus c_n$$

The resulting sum is not satisfactory.

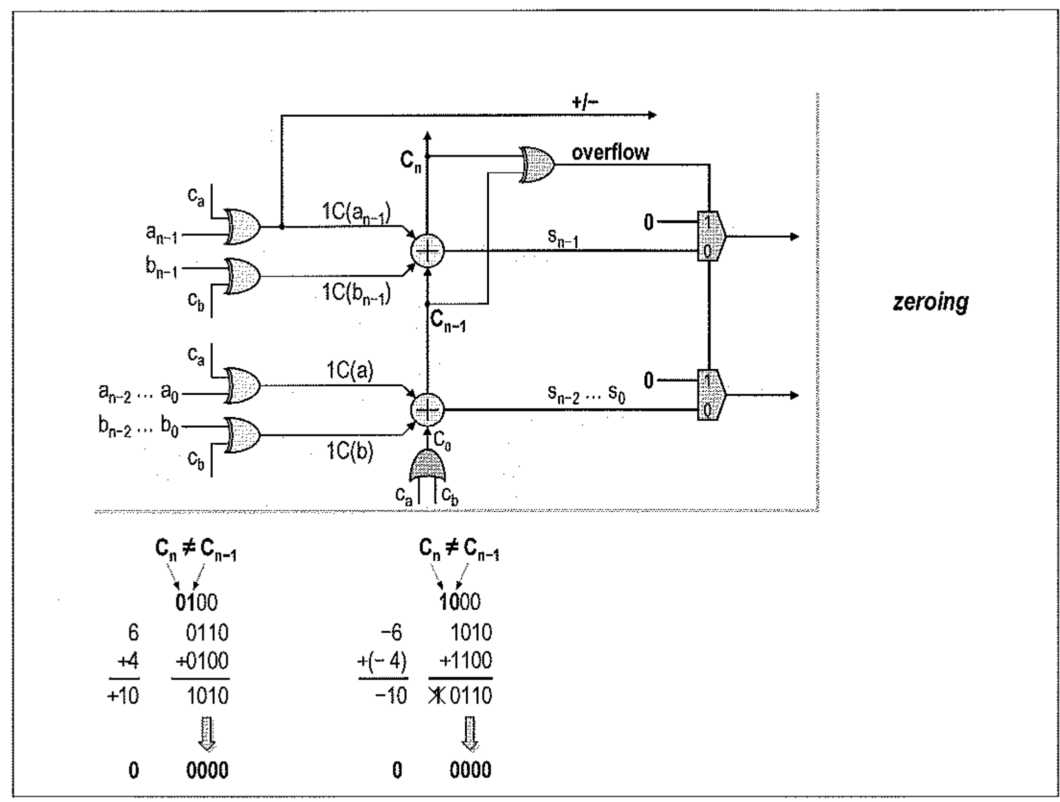
The saturating logic is activated when overflow is detected:

positive overflow: the sum is replaced by 011...11

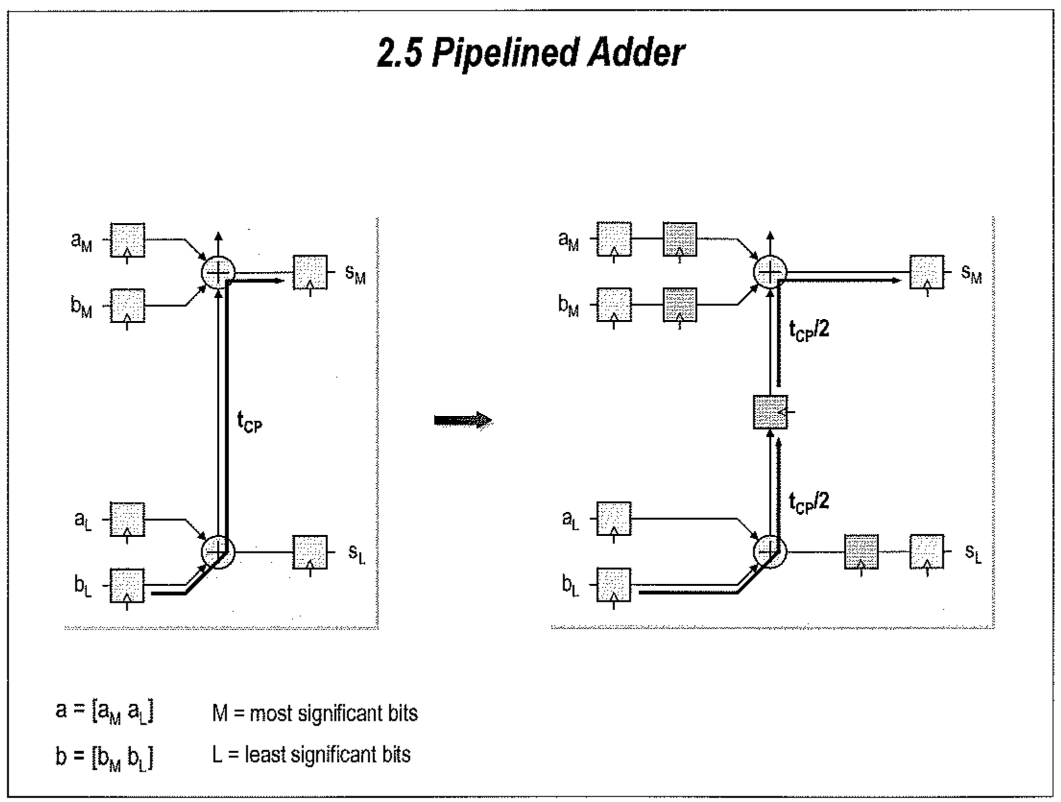
negative overflow: the sum is replaced by 100...00 (asymmetric saturation)

100...01 (symmetric saturation)

Remark: For the case that the result is 100...00, no overflow occurs. For symmetric saturation, a special detection circuit is needed to replace the result by 100...01. This circuit is not shown.

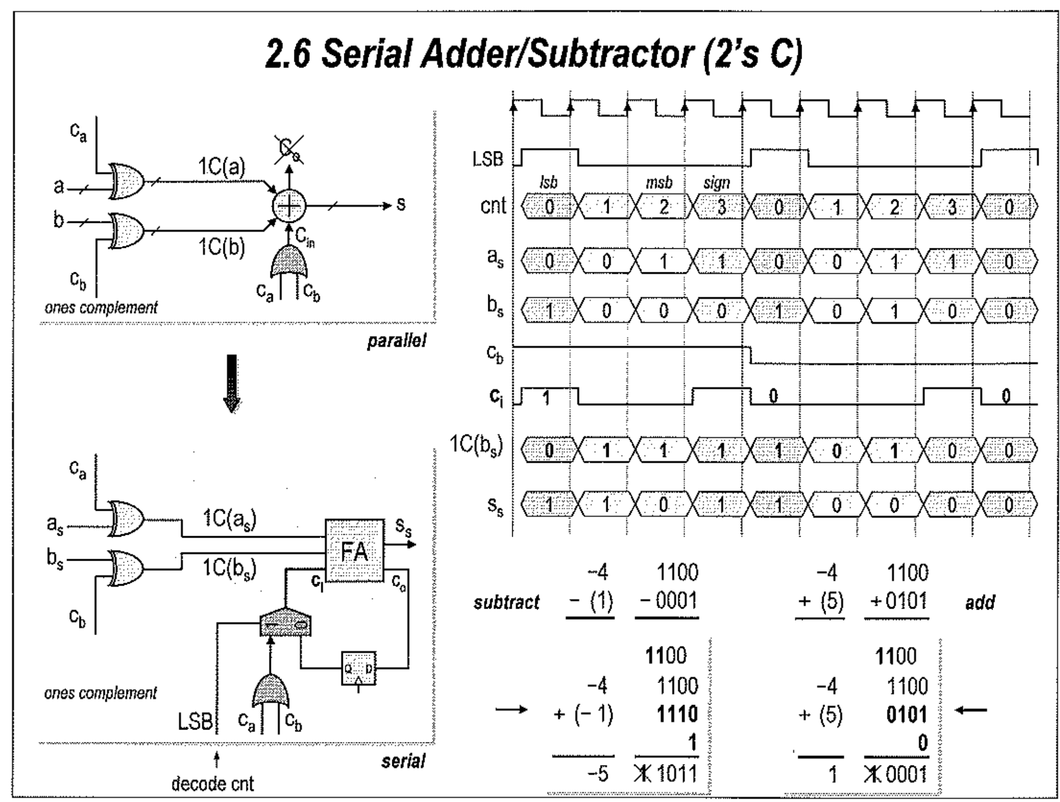


The zeroing logic is activated when overflow is detected. The sum is replaced by 000…00.



The critical path of an adder is the carry chain wit a delay t_{CP}.

This delay can be divided by two, by placing a pipeline register in the middle of the carry chain. Extra registers are inserted to align the most (M) and least (L) significant parts of the operands (a and b) and the sum (s). This is called retiming .



A bit-serial adder/subtractor can reduce the needed hardware but with a more complex control structure.

Adder

The two serial inputs a_s and b_s are assumed to be n-bit twos complement numbers and are synchronized to a clock signal, with the least significant bits (LSB's) first. The full adder is a combinational logic circuit which adds the two present input bits and a carry bit to produce a sum bit and an output carry bit. The output carry is stored in a one-bit delay or flip-flop (D) to be input for the full adder during the next clock cycle. After the most significant bits and sign bits are added to complete the addition, a new addition may begin immediately in the next clock cycle. Since a nonzero carry bit may still be stored in D from the previous addition, a clear (LSB) timing signal, which is zero during each LSB clock cycle, forces the input carry to zero via the multiplexer.

Assuming that the data samples are (n+1)-bit twos-complement numbers, an addition requires n + 1 clock cycles. This is one clock cycle longer than the serial inputs (operands). Several methods can be used to make operands and result equal in length: sign extension of the operands, rounding of the result, limiting the range of the operands so that the result can be represented with n bits (as was done in the figure).

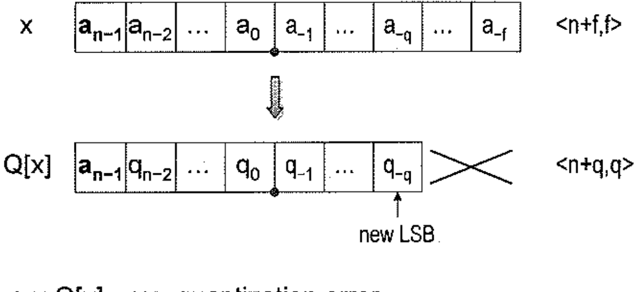
Subtractor

Subtraction is the negation of the subtrahend (b_s in the example shown), followed by addition. Negation of a twos-complement number is implemented by complementing all the bits of the number and adding '1' in the LSB position. Hence in the subtractor, an inverter complements all bits of the subtrahend, and the carry loop is modified to cause an initial carry input of '1' during the LSB cycle.

3. Quantization

reduction of the precision of a binary fixed-point number from n+f bits to n+q bits

→ discards 'lower'bits



e = Q[x] – x = quantization error

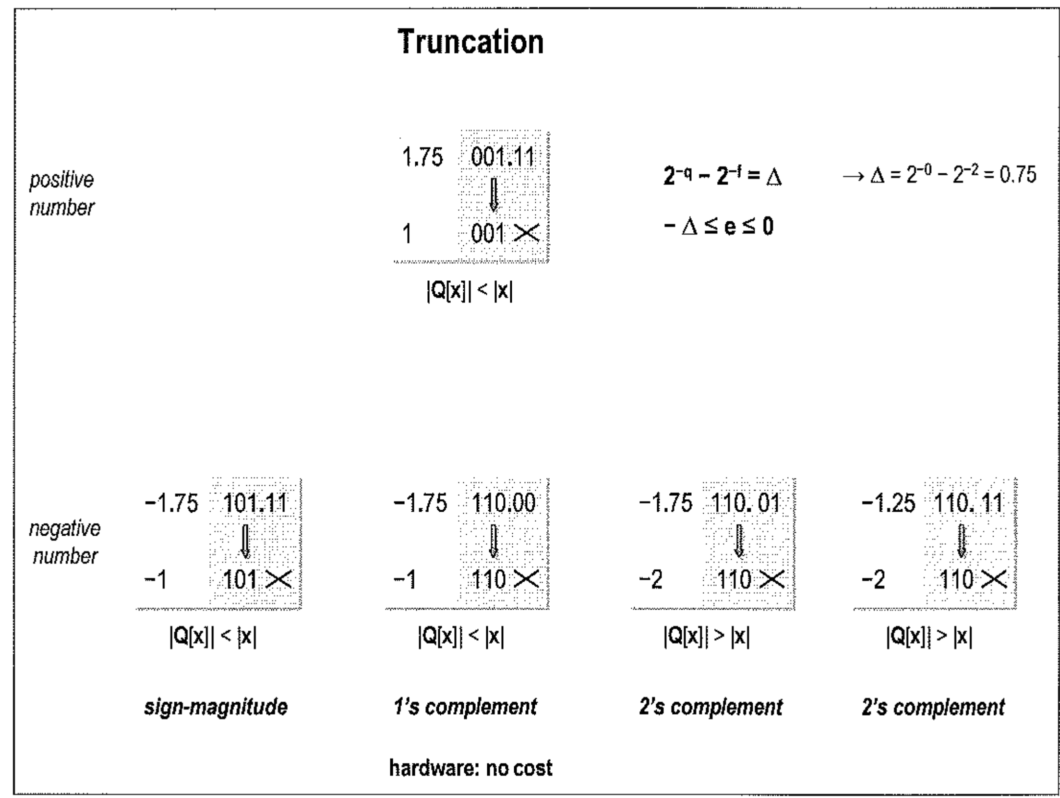
In performing fixed-point operations such as multiplications, it is often necessary to quantize a binary fixed-point number x, to another number Q[x], reducing the precision (number of fractional bits) from f to q, as shown. This can be done via truncation or rounding.

The effect of this reduction of precision is to introduce an error whose power depends on the values of f and q and whose statistical behaviour depends on the type of truncation or rounding used. This error is referred to by the generic name quantisation error and is given by:

$$e = Q[x] - x$$

Overflow handling versus Quantization

- Overflow handling discards upper bits.
- Quantization discards lower bits.
- Overflow handling is concerned with values that lie out of the range of the data type.
- Quantization is concerned with values that are more precise than the data type (i.e., between values in the data type).



In truncation, the least significant bits $a_{-q-1} \ldots a_{-f}$, are simply dropped, regardless of the sign and the convention to represent the negative numbers.

If x is a *positive number*, the 3 binary representations are identical.
If all the truncated bits are 1, the maximum quantisation error occurs, with absolute value Δ :

$$\Delta = 2^{-f} - 2^{-q}$$

The resulting number Q[x] is always smaller than the original number x. The corresponding quantisation error is always negative and limited:

$$-\Delta \leq e \leq 0 \quad \text{and} \quad Q[x]| \leq |x|$$

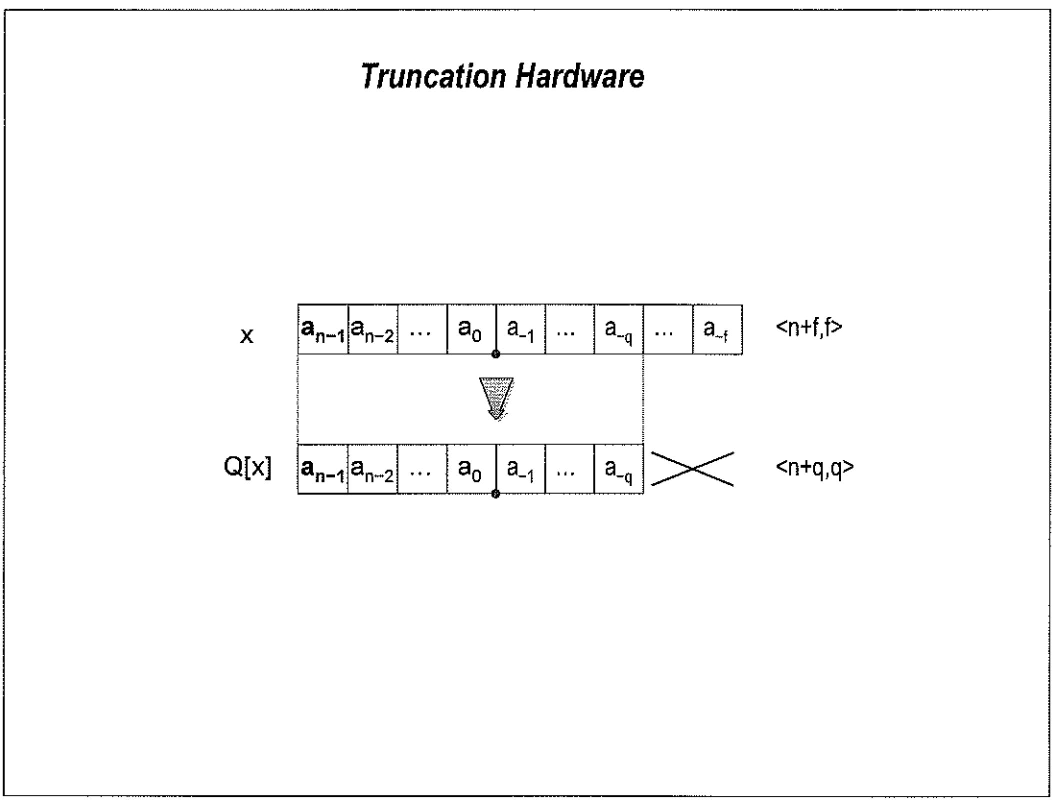
If x is a *negative number*, the error depends on which binary representation is used.

• sign-magnitude truncation and ones complement truncation:

$$0 \leq e \leq \Delta \quad \text{and} \quad |Q[x]| \leq |x|$$

• twos complement truncation:

$$-\Delta \leq e \leq 0 \quad \text{and} \quad |Q[x]| \geq |x|$$

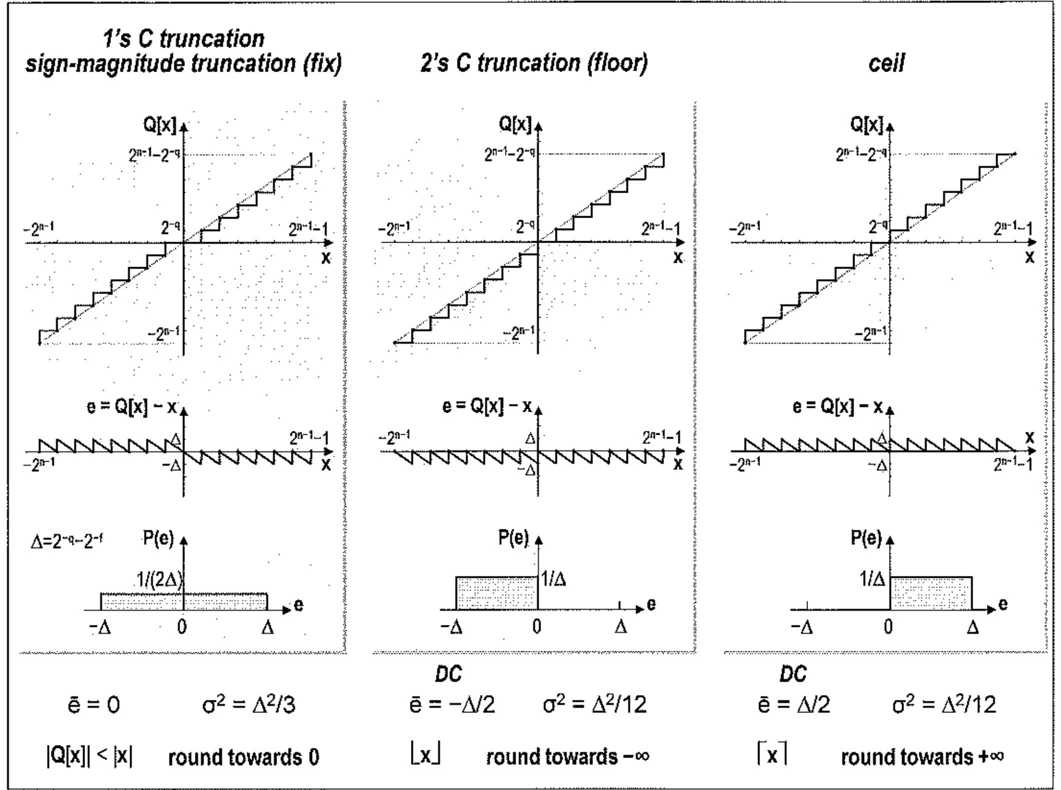


Truncation

- Simply discards the low (f-q) bits
- Quite imprecise

Example:

0000|1111 becomes 0 by truncation of the lower 4 bits.



The quantisation curve, the quantization error and the statistical behaviour of the quantisation error (error probability density function) for truncation and ceil are shown.

Round towards zero (1's C truncation, sign-magnitude truncation = fix)

The sign of the quantisation error depends on the sign of the number to be quantised. The mean value for random numbers is zero.

Notice that with round towards zero, the probability density function is twice as wide as the others. For this reason, the variance is four times that of the others.

Round towards minus infinity (2's C truncation = floor)

Floor is often called truncation when used with integers and fixed-point numbers that are represented in two's complement. It is the most common quantisation mode of DSP processors because it requires no hardware to implement.

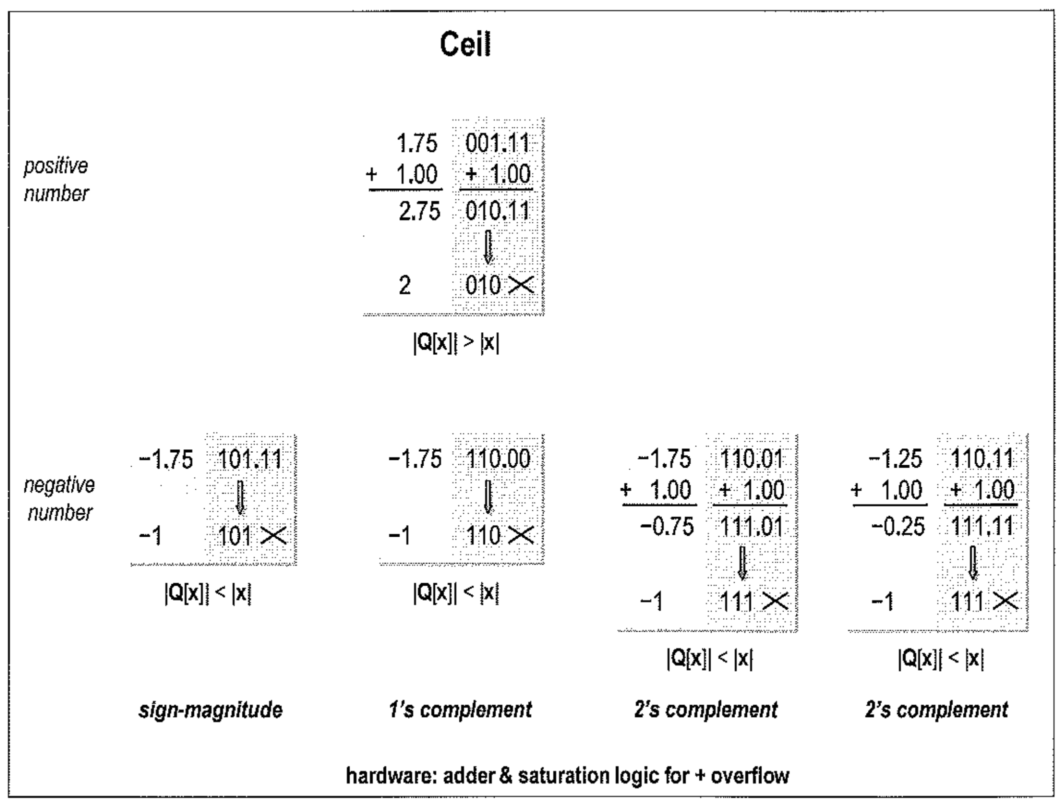
The sign of the quantisation error is always negative. So for random numbers the mean value is also negative. This results in a DC shift in digital filters implemented with 2's C truncation!

Floor does not produce quantized values that are as close to the true values as round will, but it has the same variance, and small signals that vary in sign will be detected, whereas in round they will be lost.

Round towards plus infinity (ceil)

The sign of the quantisation error is always positive. So for random numbers the mean value is also positive. This results in a DC shift in digital filters!

Ceil does not produce quantized values that are as close to the true values as round will, but it has the same variance, and small signals that vary in sign will be detected, whereas in round they will be lost.



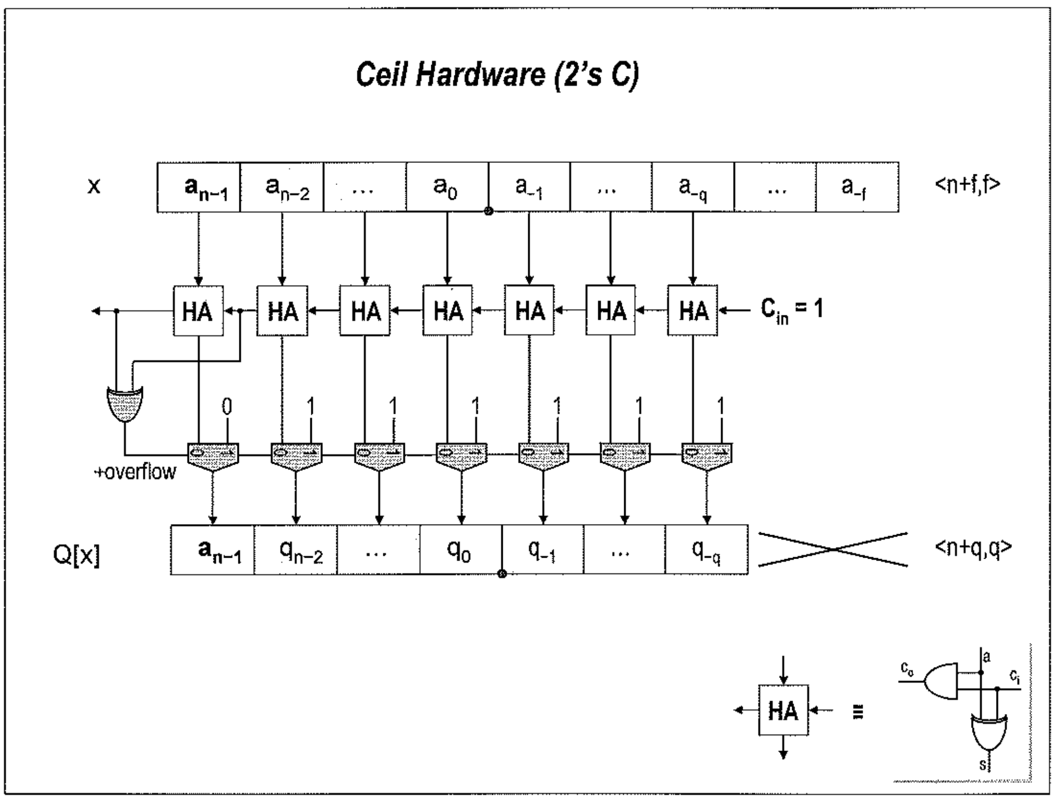
Round towards minus infinity (2's C truncation = floor)

2's C truncation requires no hardware to implement. The least significant bits are simply ignored.

Round towards plus infinity (ceil)

2's C ceiling requires extra hardware, without reducing the quantisation error.

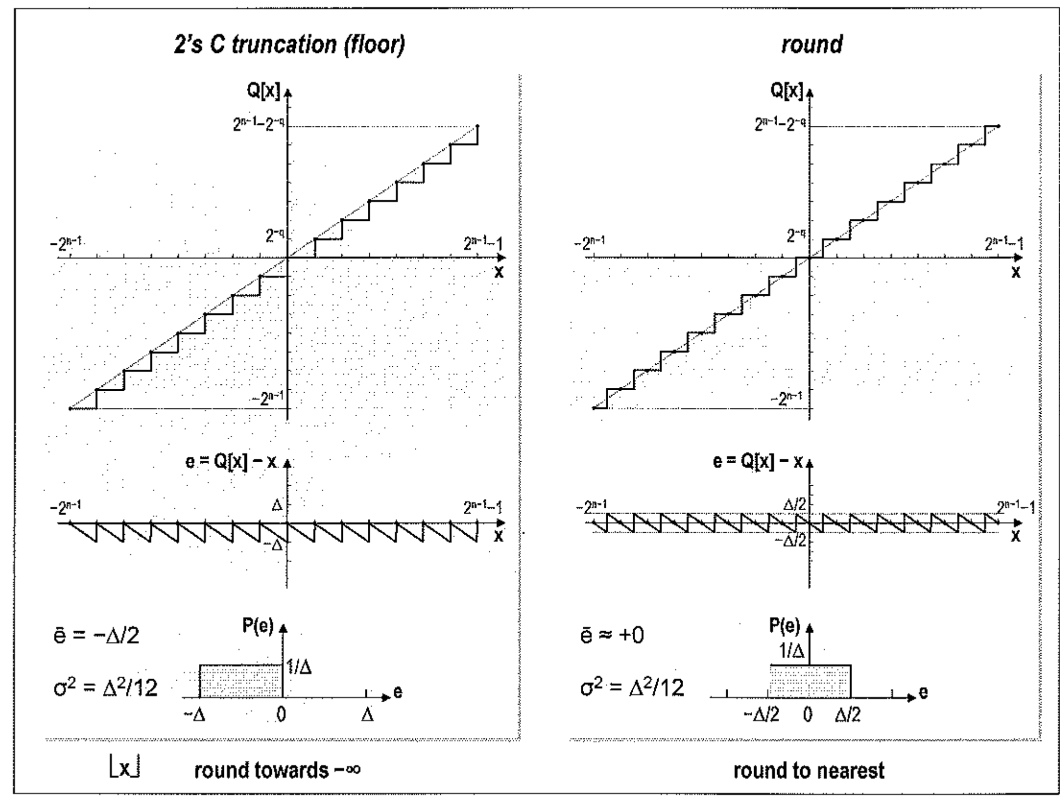
First, a '1' has to be added at the LSB position of the new (quantized) number before the least significant bits can be ignored. Due to this addition, overflow is possible.



Round towards plus infinity (ceil)

2's C ceiling requires extra hardware.

- First, a '1' is added at the LSB position (bit a_{-q}) of the new (quantized) number.
- Then, the least significant bits can be ignored.
- When positive overflow occurs due to the addition, saturation logic can be activated.



The quantisation curve, the quantization error and the statistical behaviour of the quantisation error (error probability density function) for truncation and rounding are shown.

Floor = Round towards minus infinity

The sign of the truncation error is always negative. So for random numbers the mean value is also negative. This results in a DC shift in digital filters implemented with 2's C truncation!

The worst case truncation error is twice as large as the worst case rounding error. Also, the truncation error is always negative, so on average it has a non-zero mean.

Round = Round-to-nearest. In a tie, round to plus infinity

The sign of the quantisation error is alternating, depending on the precise magnitude of the number. A small bias is introduced by ordinary "round" caused by always rounding the tie in the same direction. So for random numbers the mean value is nearly zero (slightly positive).

Round is more accurate than floor, but all values smaller than 2^{-q} get rounded to zero and so are lost.

Round

2's C truncation

	decimal	binary
positive number	1.75	001.11
	⇓	⇓
	1	001 ✕
negative number	−1.75	110.01
	⇓	⇓
	−2	110 ✕

2's complement

hardware: no cost

2's C round

1.75	001.11	1.50	001.10	1.25	001.01
+ 0.50	+ 0.10	+ 0.50	+ 0.10	+ 0.50	+ 0.10
2.25	010.01	2.00	010.00	1.75	001.11
2	010 ✕	2	010 ✕	1	001 ✕

TIE ↑

−1.75	110.01	−1.50	110.10	−1.25	110.11
+ 0.50	+ 0.10	+ 0.50	+ 0.10	+ 0.50	+ 0.10
−1.25	110.11	−1.00	111.00	−0.75	111.01
−2	110 ✕	−1	111 ✕	−1	111 ✕

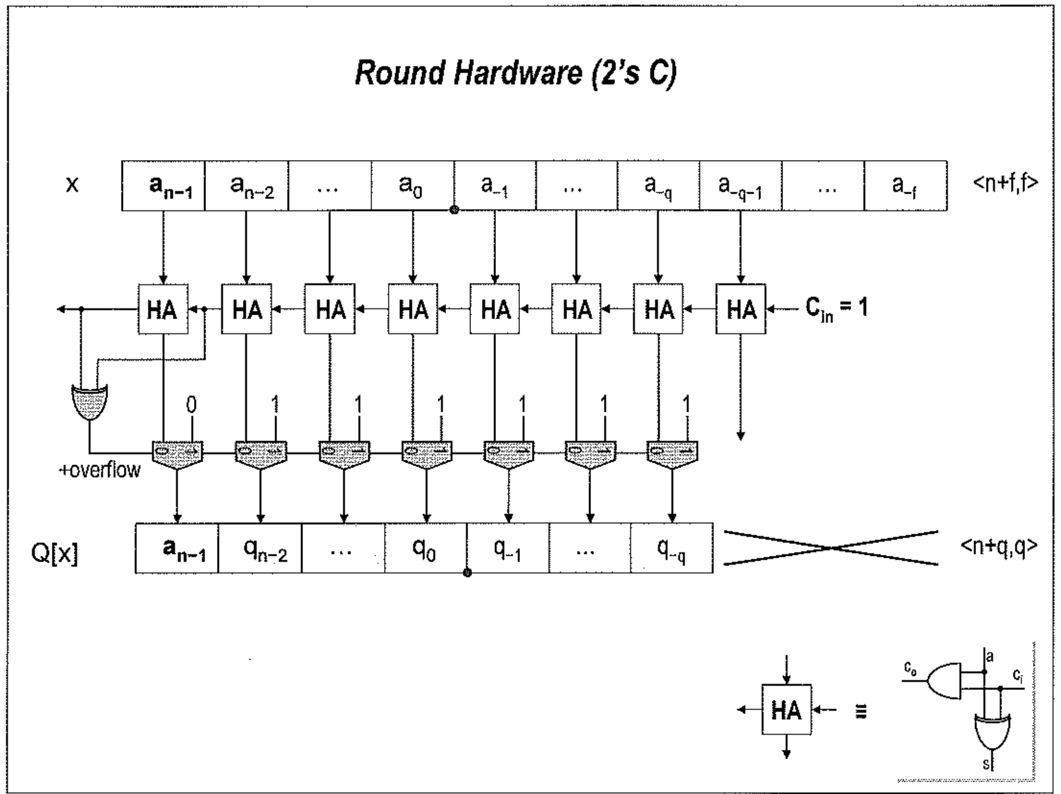
2's complement

hardware: adder & saturation logic for + overflow

In rounding, the value Q[x] is taken to be the nearest possible number to x. The quantisation error is limited by:

$$-(2^{-f} - 2^{-q})/2 < e \leq -(2^{-f} - 2^{-q})/2$$

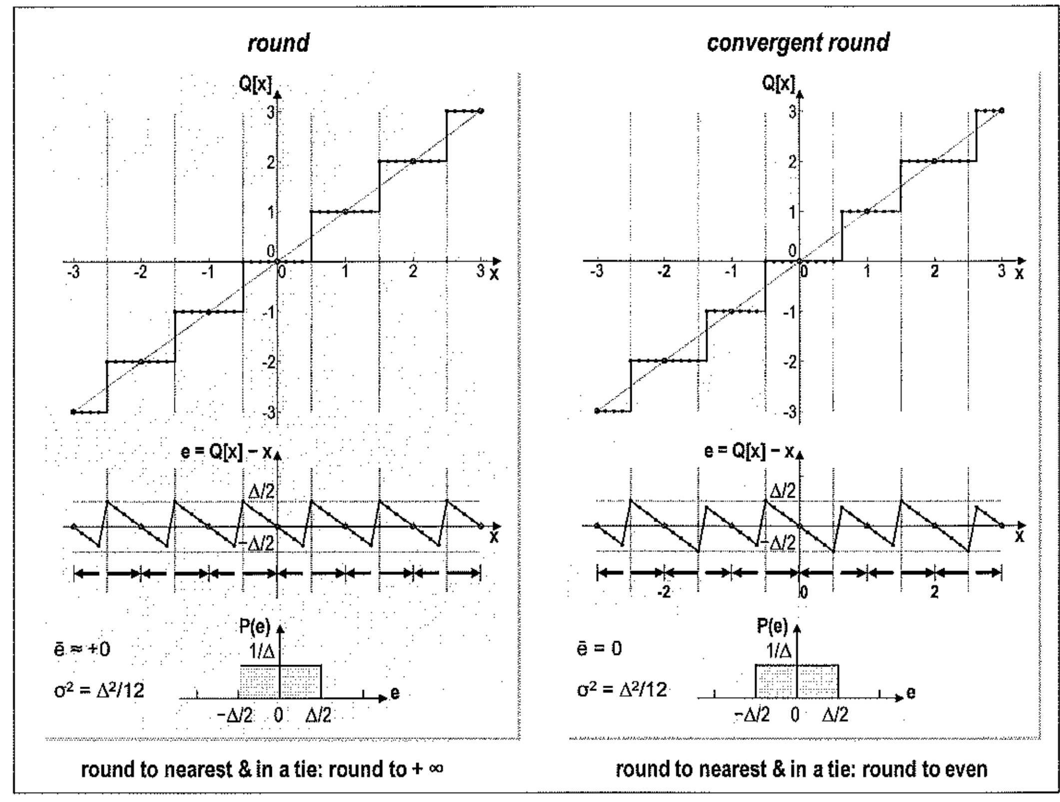
Rounding is more accurate than truncation, but requires more effort in its implementation.



Round = Round-to-nearest. In a tie, round to plus infinity

2's C rounding requires extra hardware.

- First, a '1' is added at the LSB−1 position (bit a_{-q-1}) of the new (quantized) number.
- Then, the least significant bits can be ignored.
- When positive overflow occurs due to the addition, saturation logic can be activated.



Convergent rounding eliminates the bias introduced by ordinary "round" caused by always rounding the tie in the same direction.

Round = Round-to-nearest. In a tie, round to plus infinity

- Chooses the nearest representable value.
- In case of a tie, rounds up.
- Sometimes called *biased* rounding.
- The error probability density function:

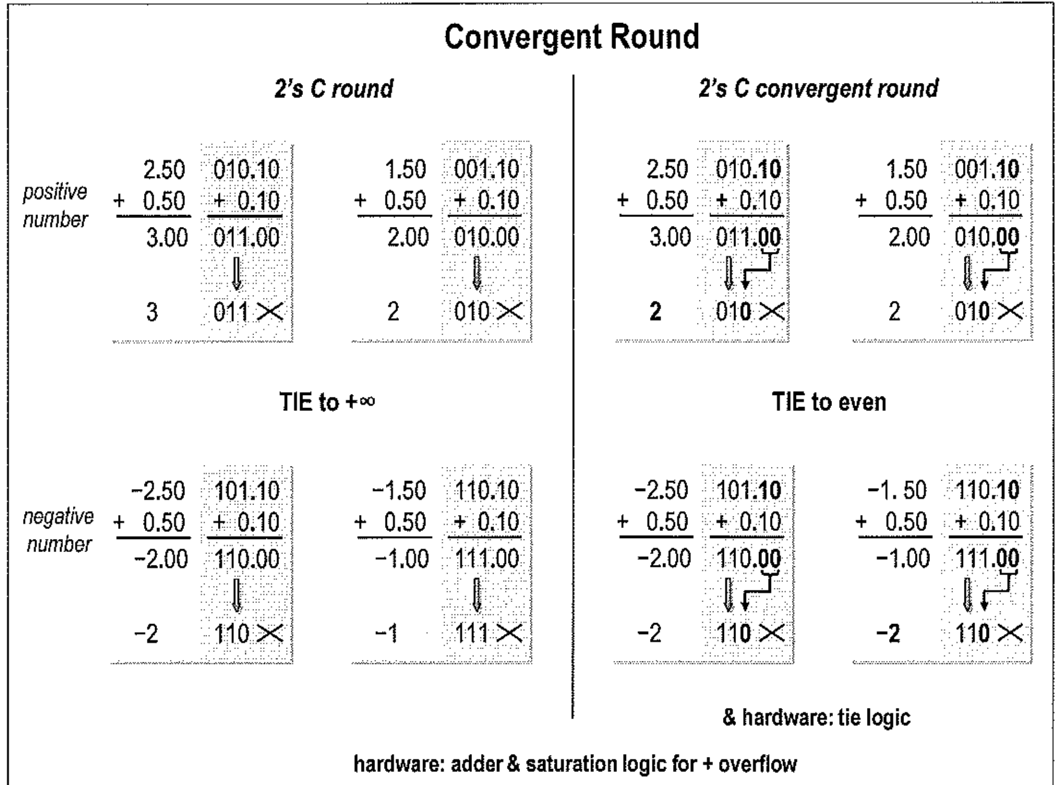
$$P(e) = 1/\Delta \quad \text{for} \quad -\Delta/2 < e \leq \Delta/2$$
$$= 0 \quad \text{otherwise}$$

Convergent rounding = Round-to-nearest. In a tie, round to even

- Also chooses nearest representable value.
- In case of tie, chooses nearest *even* number.
- Sometimes called *unbiased* rounding.
- The error probability density function:

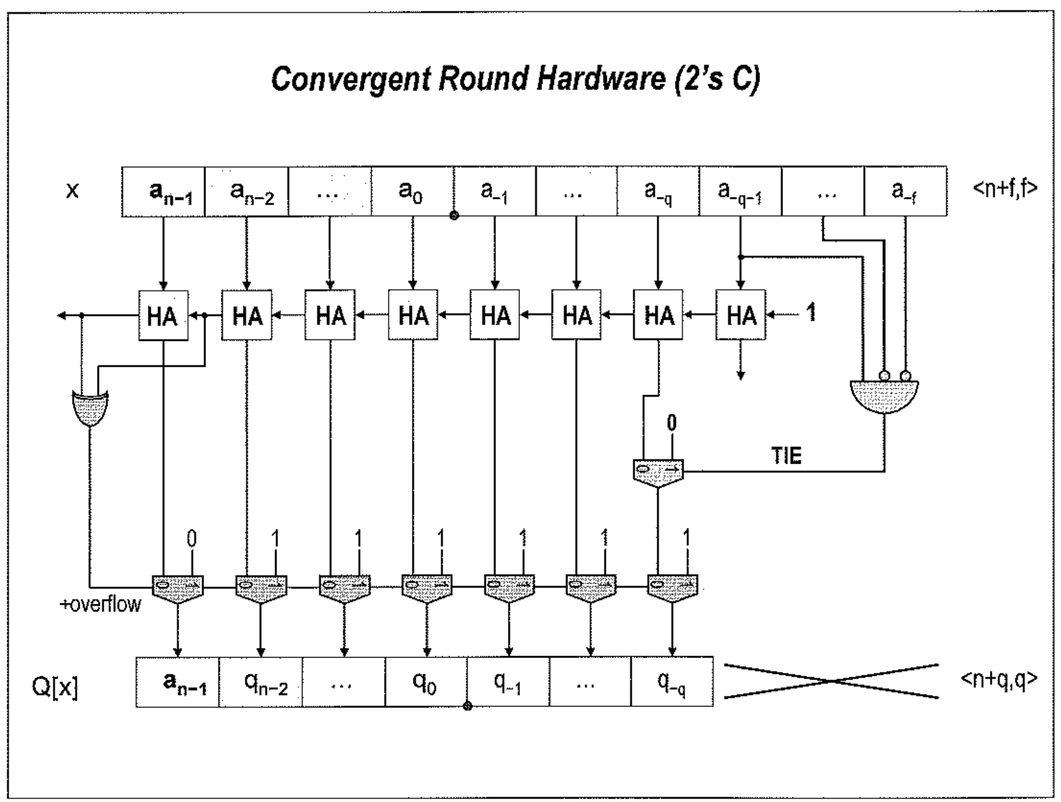
$$P(e) = 1/\Delta \quad \text{for} \quad -\Delta/2 \leq e \leq \Delta/2$$
$$= 0 \quad \text{otherwise}$$

The error probability density function for convergent rounding is difficult to distinguish from that of round-to-nearest by looking at the plot. Note that the error p.d.f. of convergent is symmetric, while round is slightly biased towards the positive. The only difference is the direction of rounding in a tie.



The conventional rounding rounds up any value above one-half the quantisation interval and rounds down any value below one-half. Because one-half is always rounded up, the result Q[x] will be biased in upward direction.

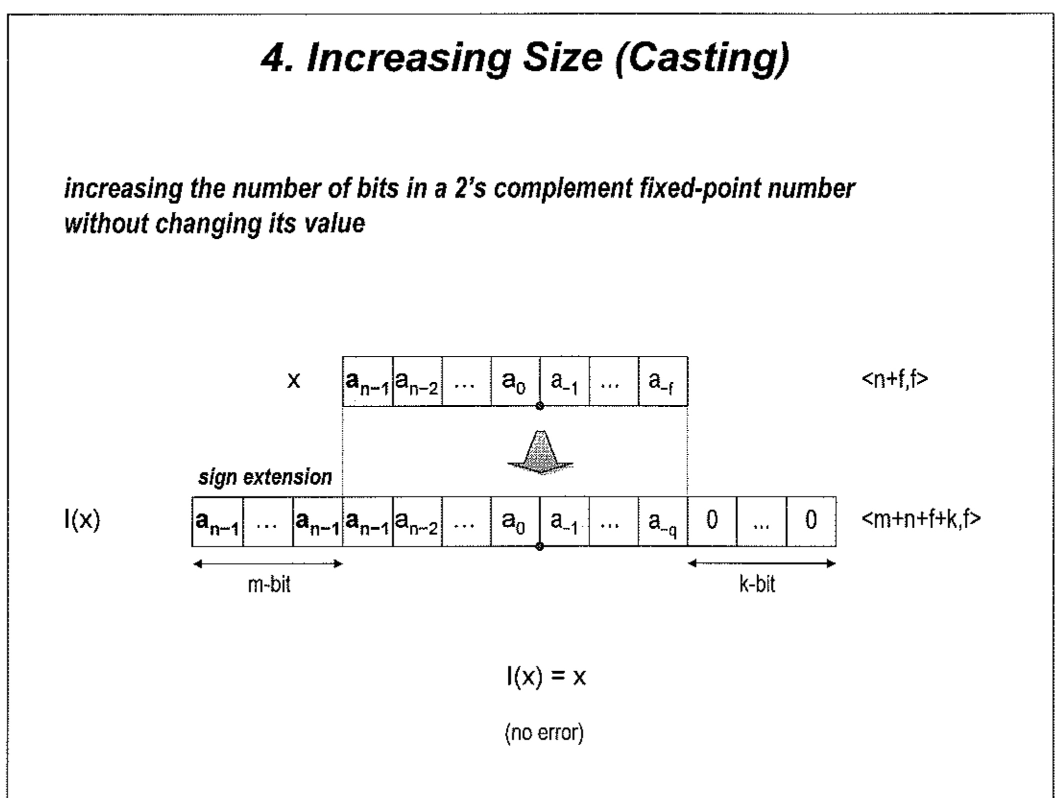
Convergent rounding solves the problem by rounding down if the number is odd and rounding up if the number is even.



Round = Round-to-nearest. In a tie, round to plus infinity

2's C rounding requires extra hardware.

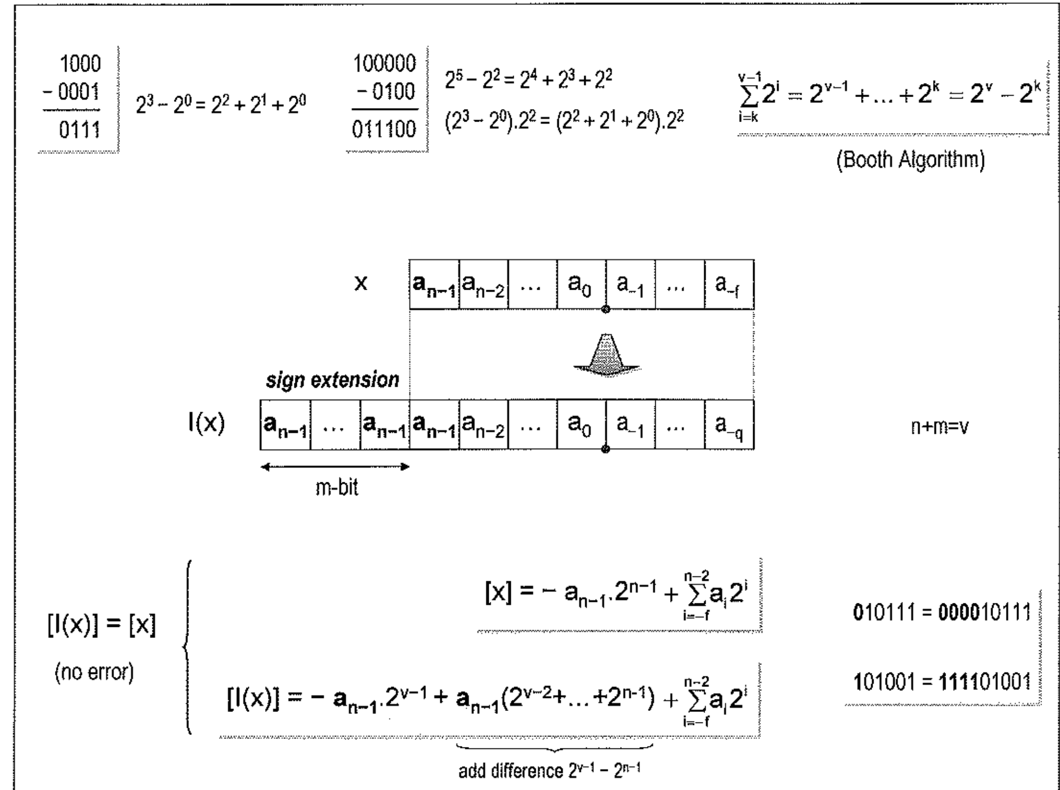
- First, a '1' is added at the LSB−1 position (bit a_{-q-1}) of the new (quantized) number.
- When a tie is detected (all bits that will be ignored are zero except the bit a_{-q-1}), the LSB position of the new (quantized) number is set to 0.
- Then, the least significant bits can be ignored.
- When positive overflow occurs due to the addition, saturation logic can be activated.



casting = changing the number of bits in a twos complement representation.

The number of bits in a twos complement representation can be increased, without changing its value.

- *To the right of the LSB*
  Append positions to the right of the LSB and fill them with 0's.
- *To the left of the MSB* = sign extension
  Replicates the sign bit in an uninterrupted sequence.



When a twos complement number with a n-bit integer part is extended to a number with a v-bit integer part, the complementation constant (= weight of the sign bit when the value is calculated) increases from 2^{n-1} to 2^{v-1}.

The difference of the two constants:

$$2^{v-1} - 2^{n-1}$$

must be added to the representation of any negative number. Using the Booth algorithm:

$$2^{v-1} - 2^{n-1} = 2^{v-2} + 2^{v-3} + \ldots + 2^{n-1}$$

this is equal to (v−n−1) 1's followed by (n−1) 0's.

Example:

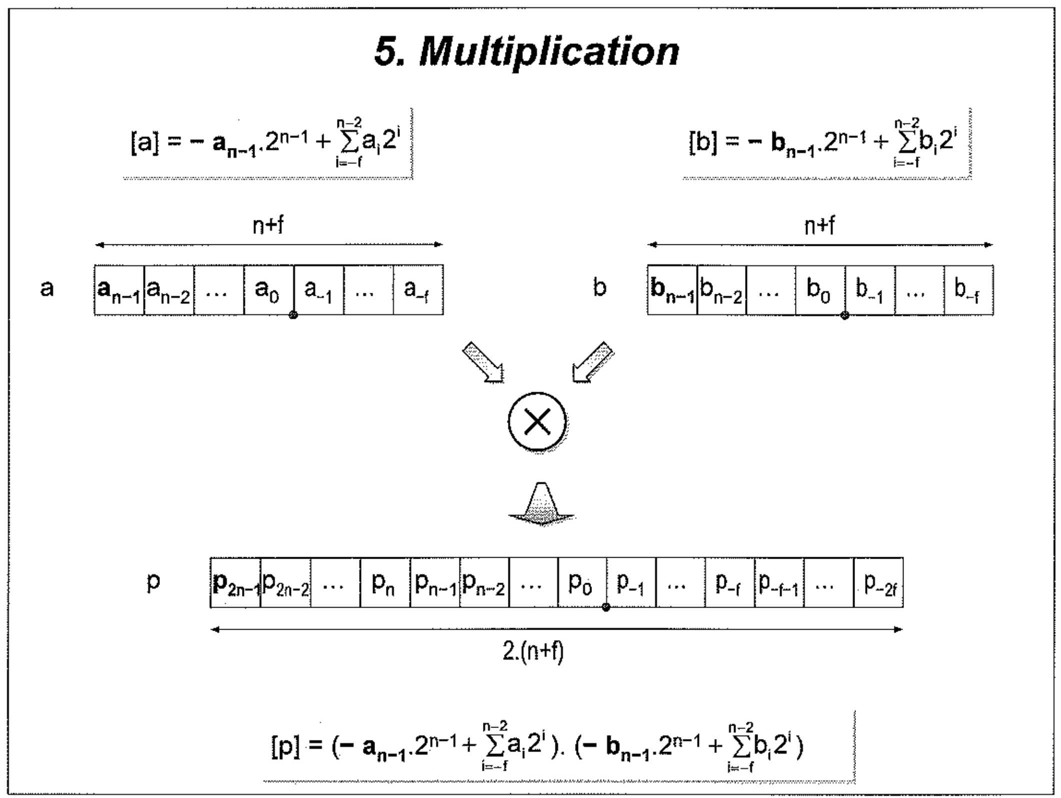
876543210		
100000000	2^8	256
- 000010000	- 2^4	- 16
011110000	$2^7 + 2^6 + 2^5 + 2^4$	240 = 128 + 64 + 32 + 16

To extend the integer part of a twos complement number with p=v-n bits, the sign bit must be extended (replicated) p times.

Example (p=3):

010111 = **000**010111

101001 = **111**101001



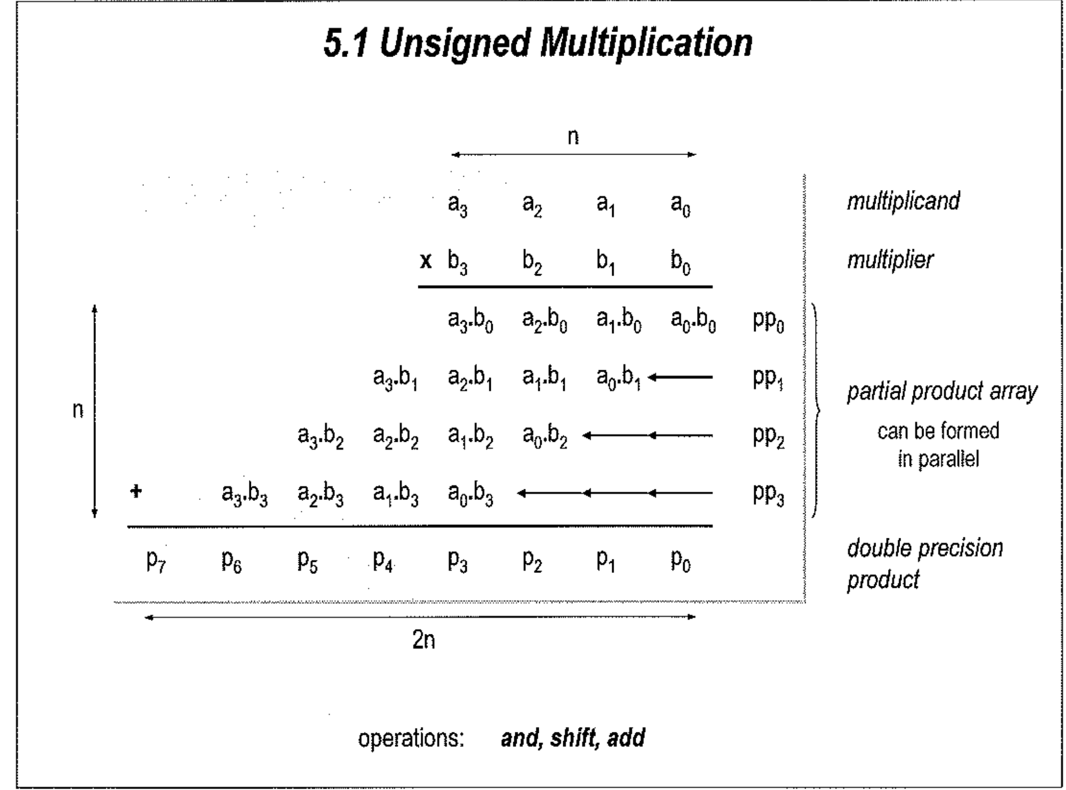
Multiplications are expensive and slow operations. The performance of many computational problems often is dominated by the speed at which a multiplication operation can be executed. This observation has, for instance, prompted the integration of complete multiplication units in state-of-the-art digital signal processors and microprocessors. Multipliers are, in effect, complex adder arrays. The analysis of the multiplier gives some further insight into how to optimize the performance (or the area) of complex circuit topologies.

Consider two signed fixed-point binary numbers a and b that are n+f bits wide. To introduce the multiplication operation, it is useful to express a and b in the binary representation:

$$[a] = -a_{n-1}.2^{n-1} + \sum_{i=-f}^{n-2} a_i.2^i \qquad [b] = -b_{n-1}.2^{n-1} + \sum_{i=-f}^{n-2} b_i.2^i$$

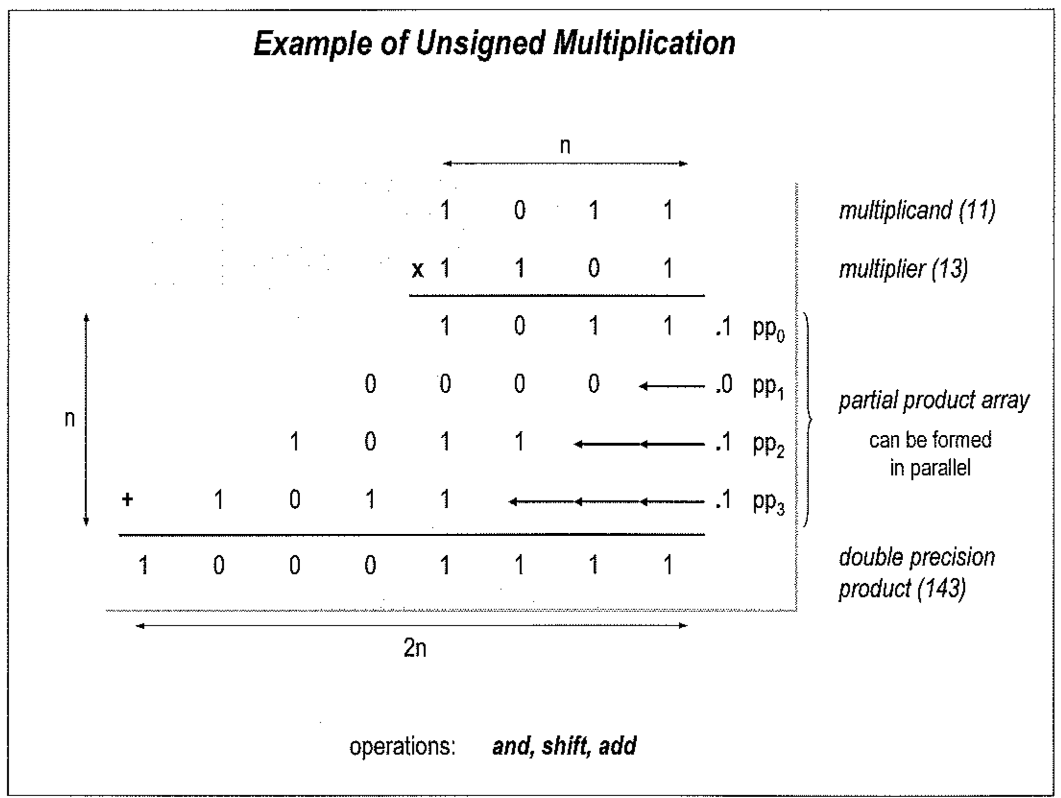
with a_i, bj $\in \{0,1\}$. The multiplication operation is then defined as follows:

$$[p] = (-a_{n-1}.2^{n-1} + \sum_{i=-f}^{n-2} a_i.2^i).(-b_{n-1}.2^{n-1} + \sum_{i=-f}^{n-2} b_i.2^i)$$



Multiplication can be defined as repeated addition. The number to be added is the multiplicand, the number of times that its shifted version is added is the multiplier, and the result is the product. Each step of the addition generates a partial product. In most computers, the operands usually contain the same number of bits. When the operands are interpreted as integers, as in fixed-point arithmetic, the product is usually twice the length of the operands in order to preserve the information content.

The simplest way to perform a multiplication is to use a single two-input adder. For inputs that are n bits wide, the multiplication takes n cycles, using an n-bit adder. This shift-and-add algorithm for multiplication adds together n partial products. Each partial product is generated by multiplying the multiplicand with a bit of the multiplier - which, essentially, is an AND operation - and by shifting the result on the basis of the multiplier bit's position. A faster way to implement multiplication is to resort to an approach similar to manually computing a multiplication. All the partial products are generated at the same time and organized in an array. A multi-operand addition is applied to compute the final product. This set of operations can be mapped directly into hardware. The resulting structure is called an array multiplier and combines the following three functions: partial-product generation (and), partial-product shift, partial-product accumulation, and final addition.



The multiplication of two unsigned n-bit numbers results in a double precision product (2n-bit):

n bits x n bits = 2n bit product

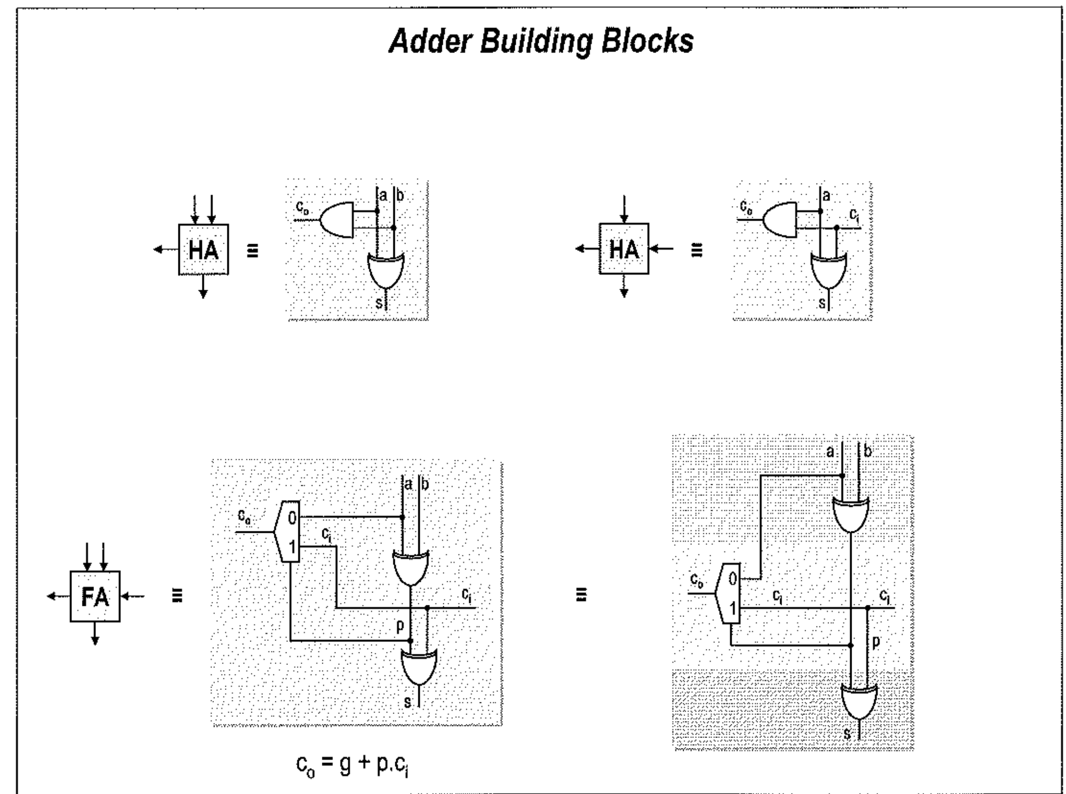
The binary representation of numbers makes the multiplication process easier:

0 → place 0 (0 x multiplicand)

1 → place a shifted copy (1 x multiplicand)

Partial products result from the logical AND of multiplicand a with a multiplier bit b_i. Each row in the partial-product array is either a copy of the multiplicand or a row of zeroes. Careful optimization of the partial-product generation can lead to some substantial delay and area reductions. Note that in most cases the partial-product array has many zero rows that have no impact on the result and thus represent a waste of effort when added. In the case of a multiplier consisting of all ones, all the partial products exist, while in the case of all zeros, there is none. Based on this observation the mean number of generated partial products is n/2. *Booth recoding* is a technique that skips over 0's to reduce the number of additions in the multiplication process.

After the partial products are generated, they must be summed. This accumulation is essentially a *multi-operand addition*. A straightforward way to accumulate partial products is by using a number of adders that will form an array - hence, the name, array multiplier. A more sophisticated procedure performs the addition in a tree format (*Wallace*).



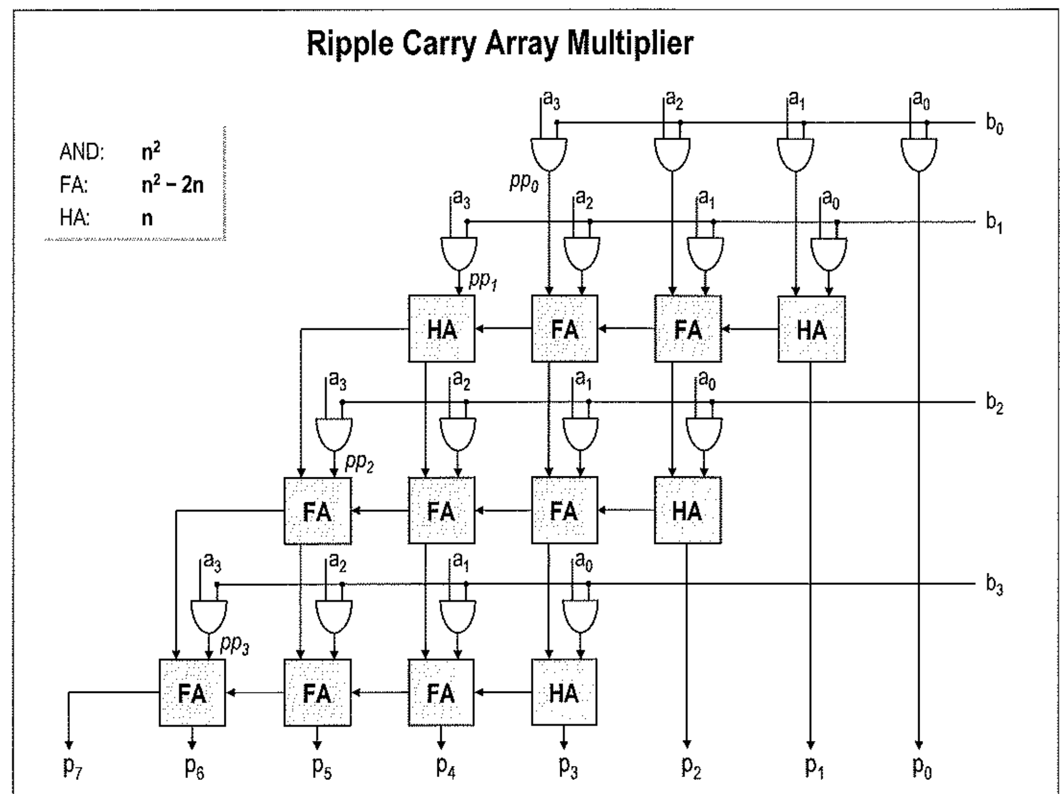
The basic building blocks for binary addition are the half adder (HA) and full adder (FA).

Half Adder (HA)

The half adder is the most primitive arithmetic circuit involved in the addition process. It has two inputs (the bits a and b to be added) and two outputs (the sum s and the output carry c_o).

Full Adder (FA)

In a multi-bit addition, the carry-out of a previous stage must be added to the sum of the present stage. This requires a third input: input carry c_i.

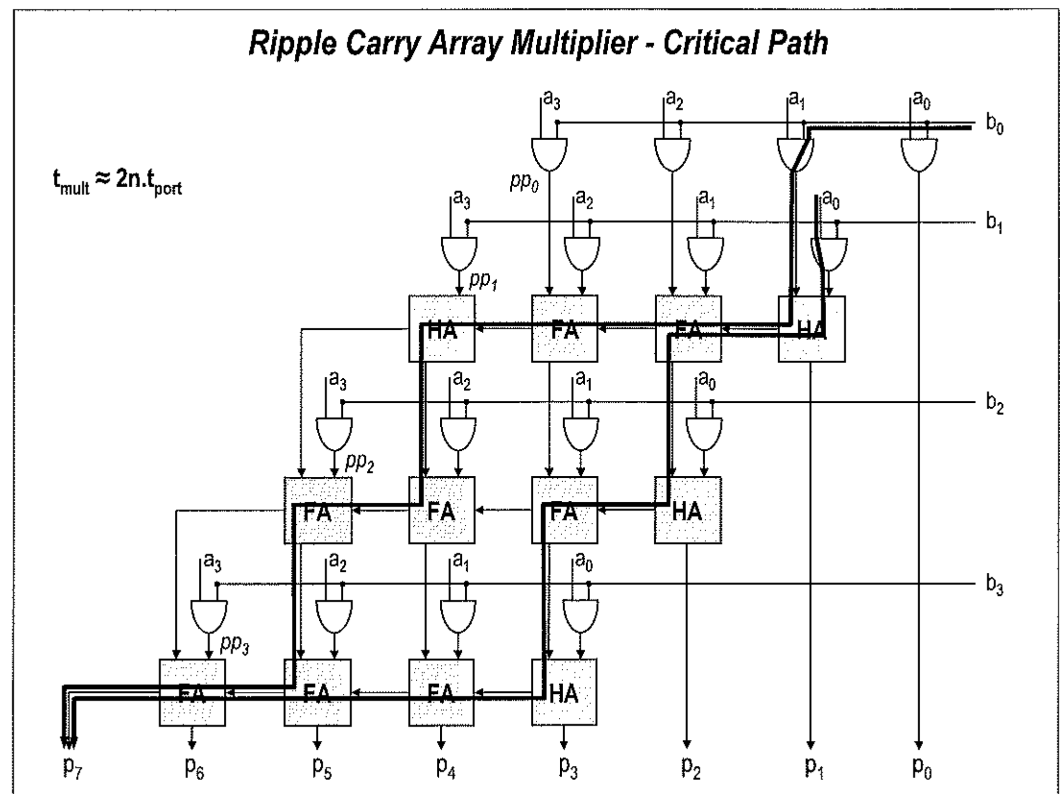


Ripple Carry Array Multiplier

A ripple carry array multiplier (also called row ripple form) is an unrolled embodiment of the classic shift-add multiplication algorithm. The illustration shows the adder structure used to combine all the partial products in a 4x4 multiplier. The partial products are the logical and of the bits of the multiplicand a witch each bit of the multiplier b.

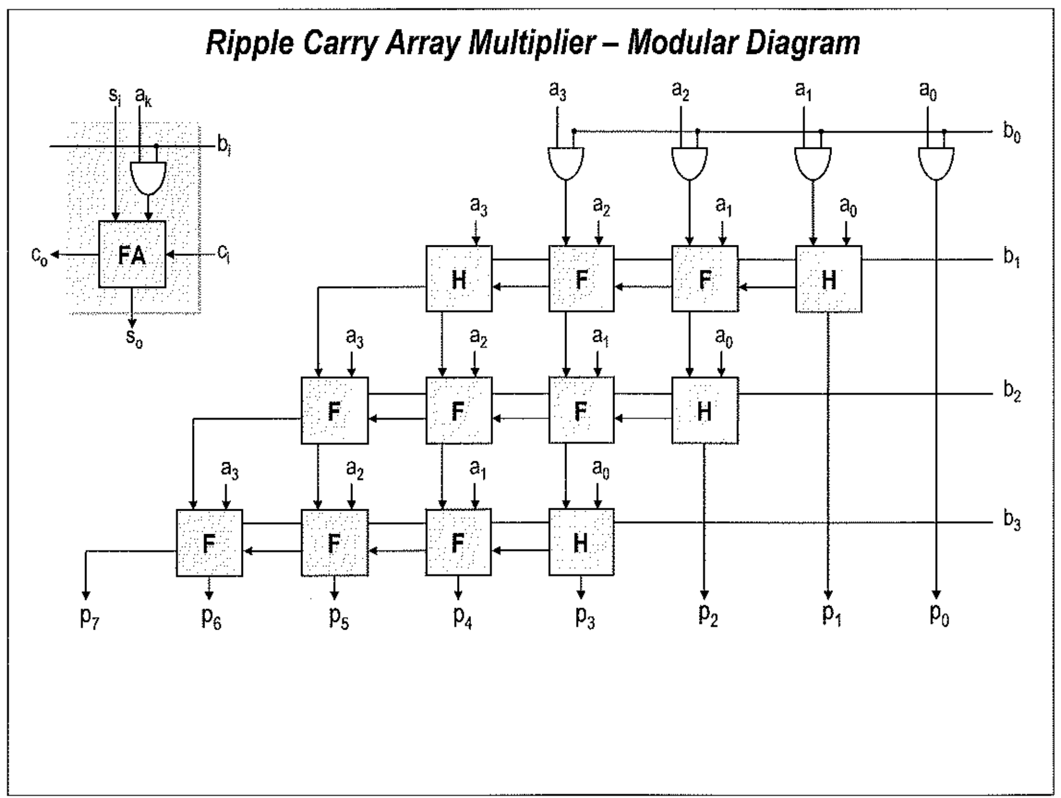
The hardware composition of an array multiplier is shown. There is a one-to-one topological correspondence between this hardware structure and the manual multiplication. The generation of n partial products requires n^2 two-bit AND gates. Most of the area of the multiplier is devoted to the adding of the n partial products, which requires (n−1) n−bit adders, of which n HA's and $(n-1)^2-1$ FA's. The shifting of the partial products for their proper alignment is performed by simple routing and does not require any logic.

This basic structure is simple to implement in FPGA's, but does not make efficient use of the logic in many FPGA's, and is therefore larger and slower than other implementations.



Due to the array organization, determining the propagation delay of this circuit is not straightforward. Consider the implementation shown. The partial sum adders are implemented as ripple-carry structures. Performance optimization requires that the *critical timing path* be identified first. This turns out to be nontrivial. In fact, a large number of paths of almost identical length can be identified. Two of those are highlighted in the figure. The maximum delay is the path from either LSB input (a_0 or b_0) to the MSB of the product (p_7), and is the same (ignoring routing delays) regardless of the path taken. The delay is approximately $2.n.t_{port}$.

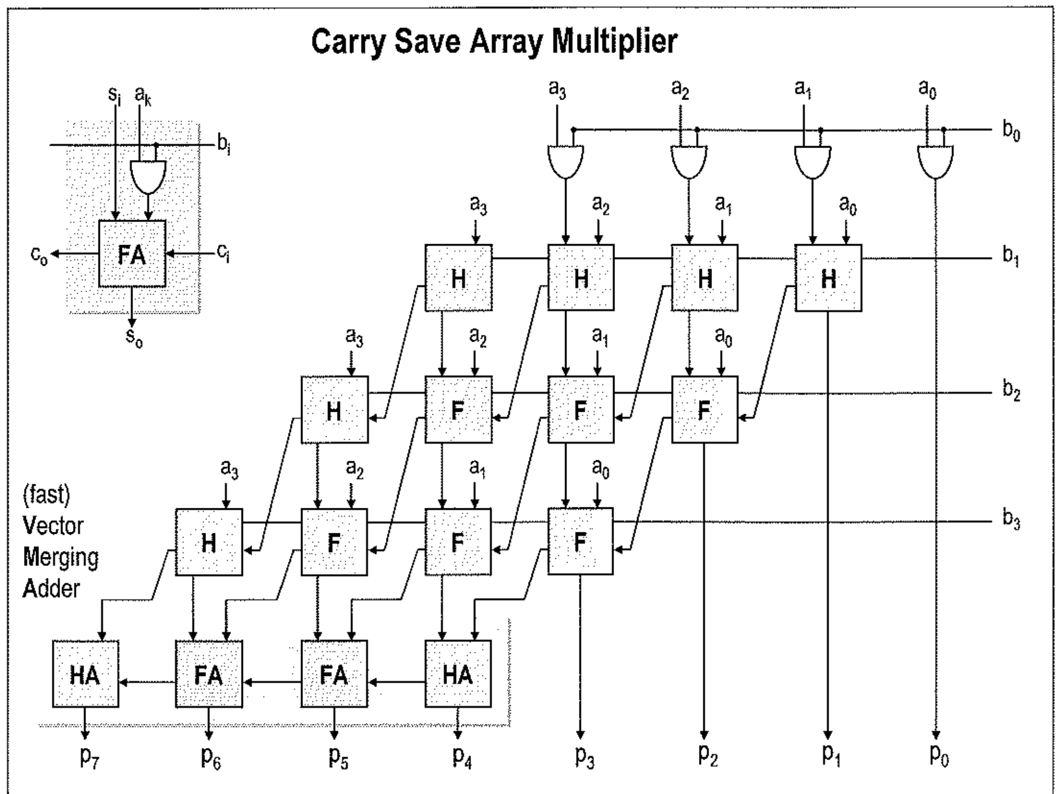
Since all critical paths have the same length, speeding up just one of them for instance, by replacing one adder by a faster one, does not make much sense from a design standpoint. All critical paths have to be attacked at the same time.



In the modular diagram of the ripple carry array multiplier, the partial product generation (and) and addition (HA or FA) are combined in a single block to simplify the structure of the multiplier.

Ripple Carry Array Multiplier (conclusion):

- row ripple form
- unrolled shift-add algorithm
- delay is proportional to n
- regular routing pattern

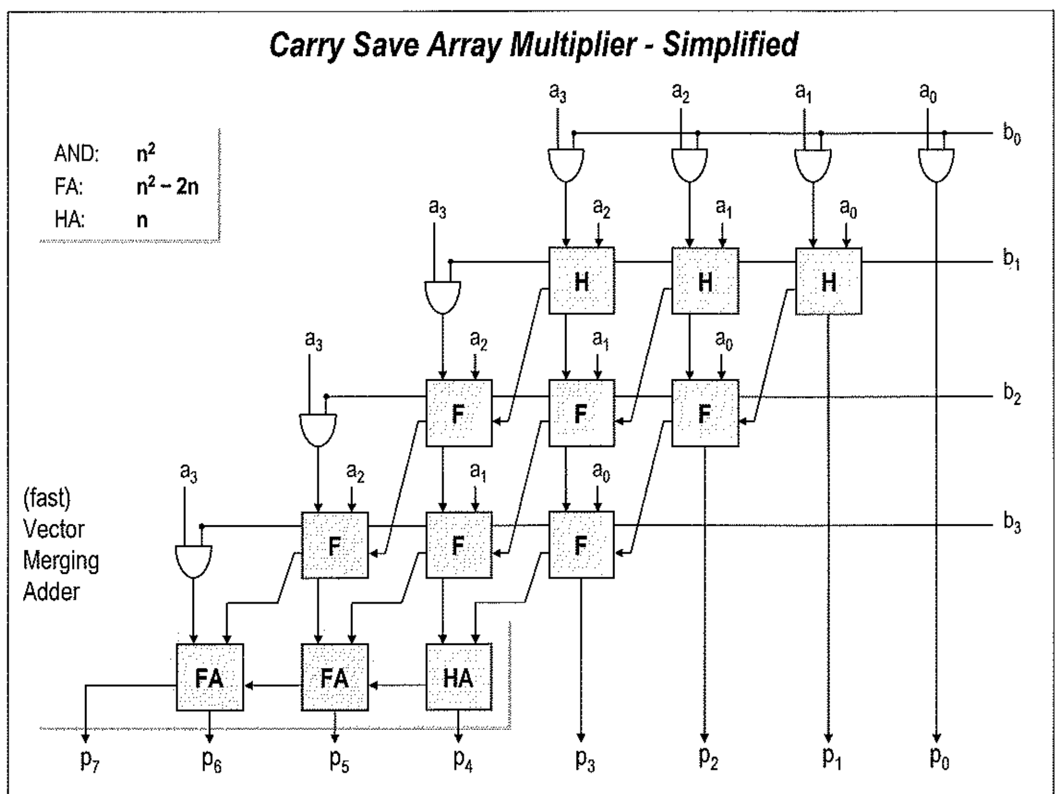


Carry Save Adder (CSA)

Multi-operand adder that does not propagate the carry to the following stage. Instead, it saves the carry propagation until all additions are completed and then take a final cycle to complete the carry propagation for all additions.

Carry Save Array Multiplier

Due to the large number of almost identical critical paths, increasing the performance of the structure of the ripple carry array through optimisation of the FA's yields marginal benefits. A more efficient realization can be obtained by noticing that the multiplication result does not change when the output carry bits are passed diagonally downwards instead of only to the right, as shown in the figure. An extra adder is included to generate the final result. This is called a Vector Merging Adder (VMA). The resulting multiplier is called a *carry save multiplier*, because the carry bits are not immediately added, but rather are "saved" for the next adder stage. In the final stage, carries and sums are merged in a fast carry-propagate (e.g., carry-look-ahead) adder stage: the Vector Merging Adder (VMA).



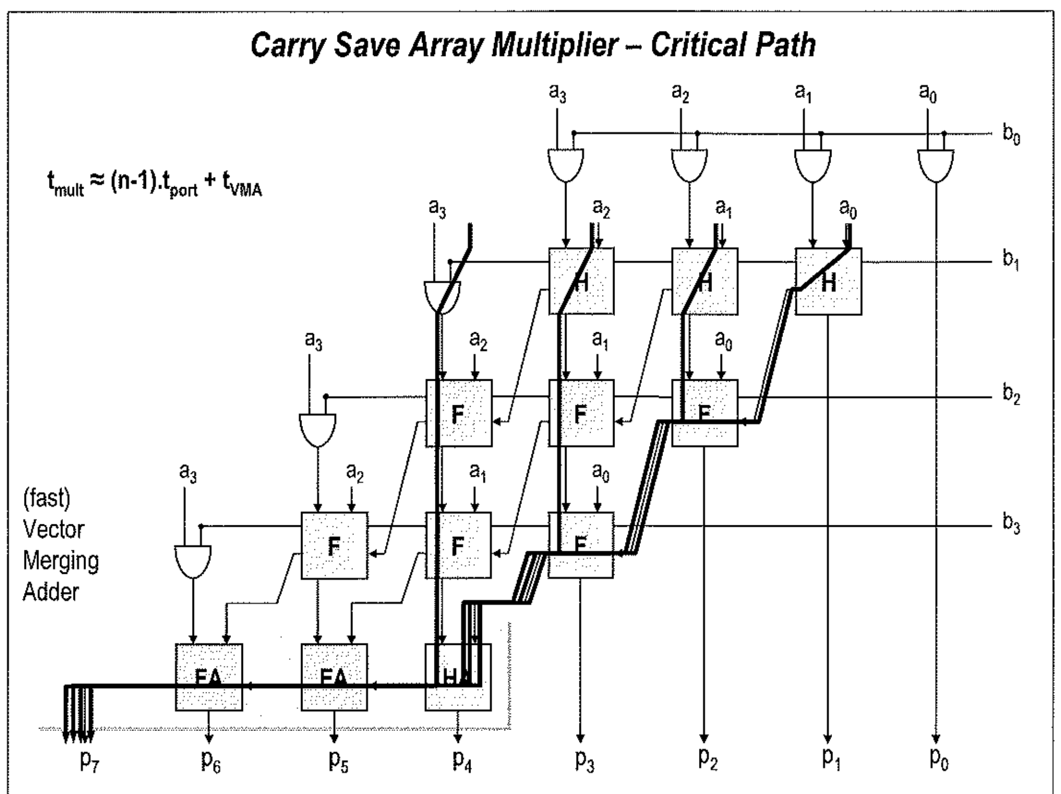
The half adders blocks (H) having the MSB of the multiplicand a as input only perform a partial product generation (no input carry, no output carry). They are replaced in the simplified structure by their equivalent and gate.

The carry save array multiplier (column ripple) has the same gate count as the carry ripple multiplier (row ripple):

AND: $n.n = n^2$

FA: $(n-1)^2-1 = n^2-2n$

HA: n

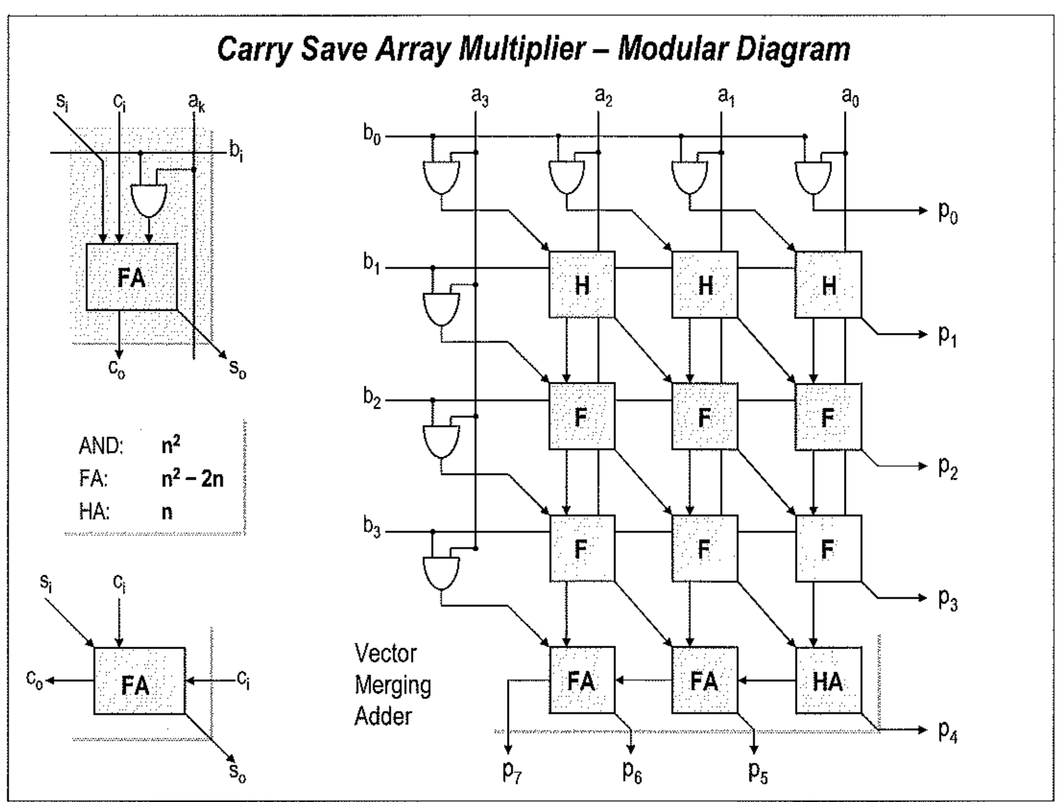


While the structure of the carry save multiplier has the same area cost, it has the advantage that its worst case critical path is shorter and uniquely defined, as highlighted in the figure and is expressed as:

$$t_{mult} \approx (n\text{-}1).t_{port} + t_{VMA}$$

This delay is almost the same as that from the carry ripple multiplier.

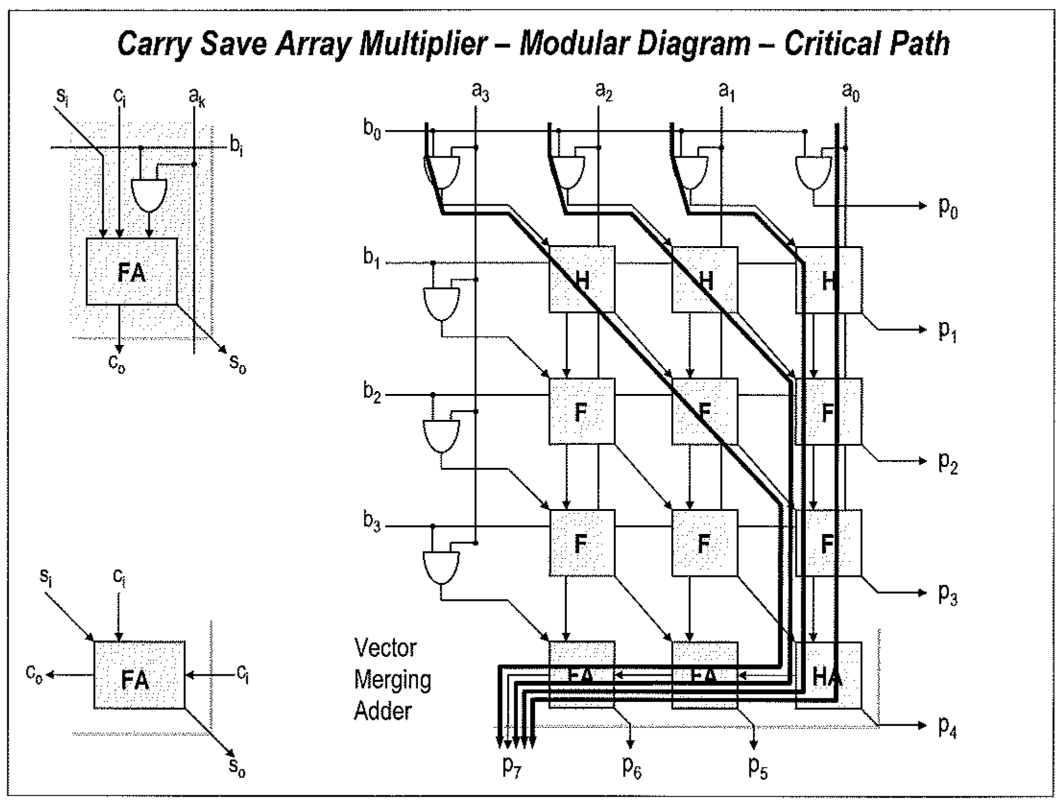
Because all the critical paths contain the delay of the Vector Merging Adder t_{VMA}, the critical path of the multiplier can be reduced by increasing only the speed of this Vector Merging Adder (carry look ahead, tree adder).



When mapping the carry save multiplier onto silicon, one has to take into account some other topological considerations. To ease the integration of the multiplier into the rest of the chip, it is advisable to make the outline of the module approximately rectangular. The overall structure can easily be compacted into a rectangle, resulting in a very efficient layout. A floor plan for the carry save multiplier that achieves this goal is shown in the figure. Observe the regularity of the topology. This makes the generation of the structure amenable to automation.

The AND gates generate the partial products. Full adders and half adders add the generated partial products. Sum outputs are connected diagonally and carry outputs are connected vertically. The last row of adders (Vector Merging Adder) which are connected from left to right, generates the n most significant product bits.

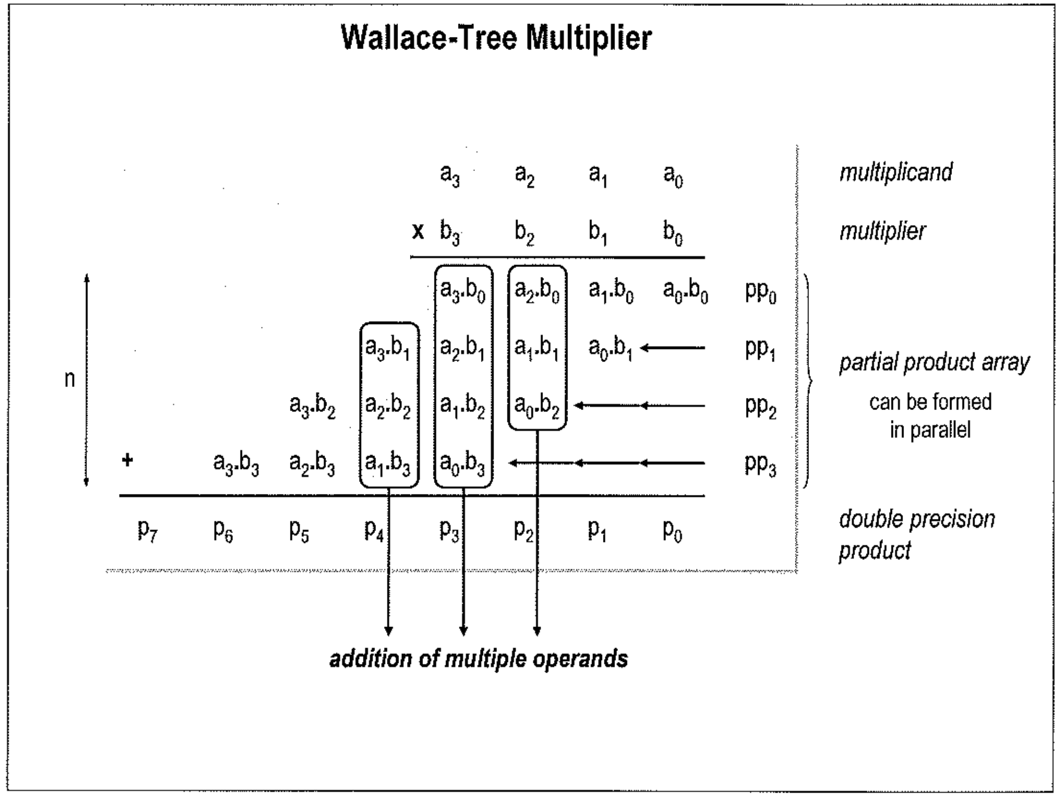
An n by n unsigned array multiplier uses n^2 AND gates, (n^2 – 2n) FA's and n HA's.



All the critical paths contain the delay of the Vector Merging Adder t_{VMA}. Almost half of the latency is due to the bottom row of adders. Therefore, the critical path of the multiplier can be reduced by increasing only the speed of this Vector Merging Adder (carry look ahead, tree adder). Although this decreases the overall delay it has a negative impact on the designs regularity.

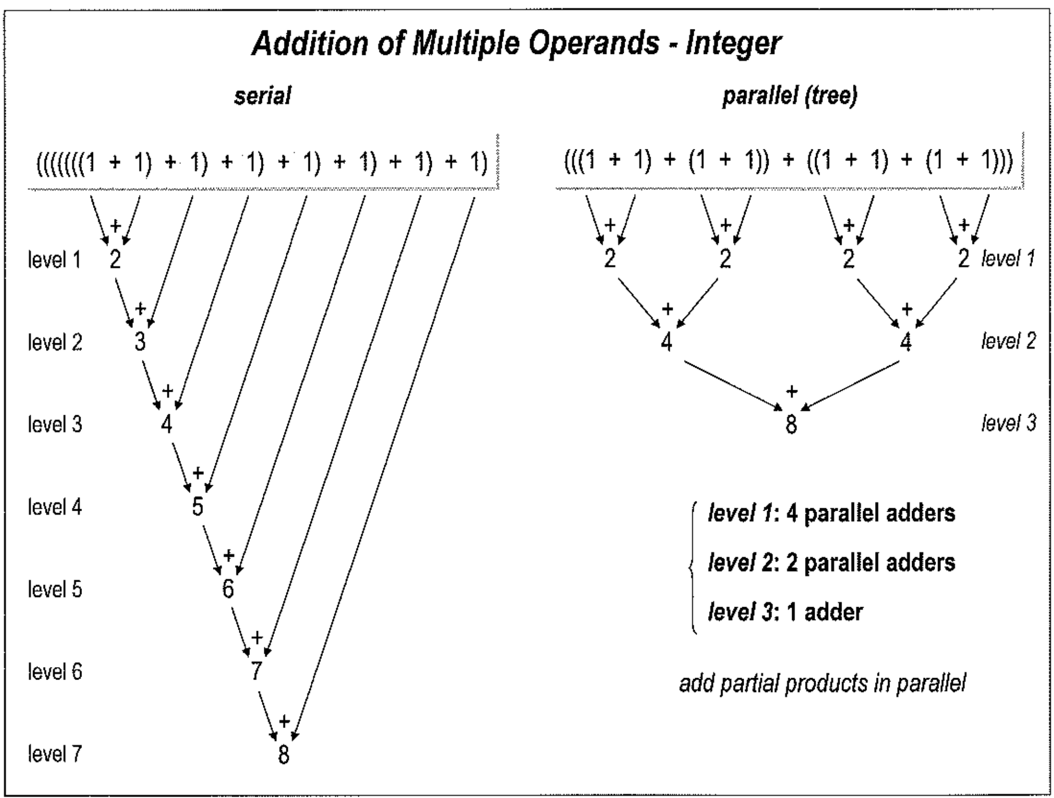
Carry Save Array Multiplier (conclusion):

- column ripple form
- fundamentally same delay and gate count as row ripple form
- ripple adder can be replaced with faster carry look ahead or carry tree adder
- regular routing pattern

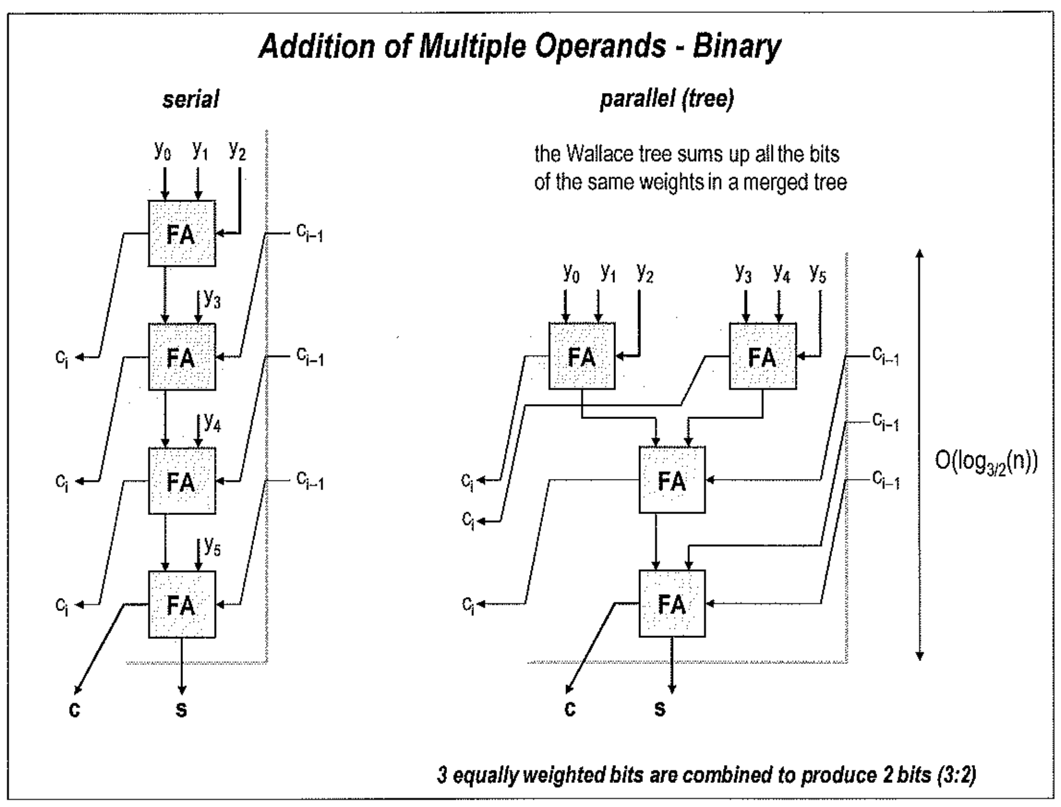


Wallace-Tree Multiplier

The partial-sum adders can also be rearranged in a treelike fashion, reducing both the critical path and the number of adder cells needed. Consider the simple example of four partial products each of which is four bits wide, as shown in the figure. The full adders needed for this operation can be rearranged by observing that only column 3 in the array has to add four bits. All other columns are somewhat less complex.

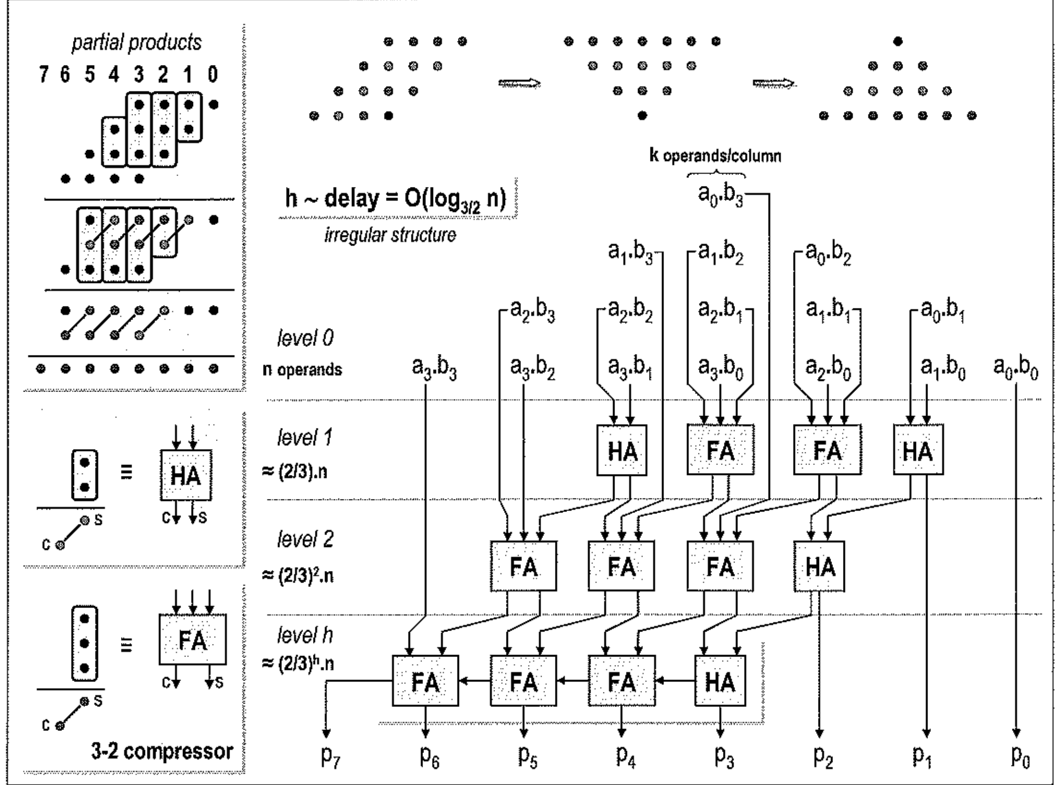


The addition is commutative and associative. By a reorganisation of the additions of multiple operands, the number of levels can be reduced (not the number of additions).



One can reduce the number of series carry save adders, by adding more of the partial products in parallel. A tree of adders is used.

A Wallace tree is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder does, the Wallace tree sums up all the bits of the same weights in a merged tree. Usually full adders are used, so that *3 equally weighted bits are combined to produce 2 bits*: one (the carry) with weight of k+1 and the other (the sum) with weight k. Each layer of the tree therefore reduces the number of vectors by a factor of 3:2 (Another popular scheme obtains a 4:2 reduction using a different adder style that adds little delay in an ASIC implementation). The tree has as many layers as is necessary to reduce the number of vectors to two (a carry and a sum). A conventional adder is used to combine these to obtain the final product. The structure of the tree is shown. For a multiplier, this tree is pruned because the input partial products are shifted by varying amounts.

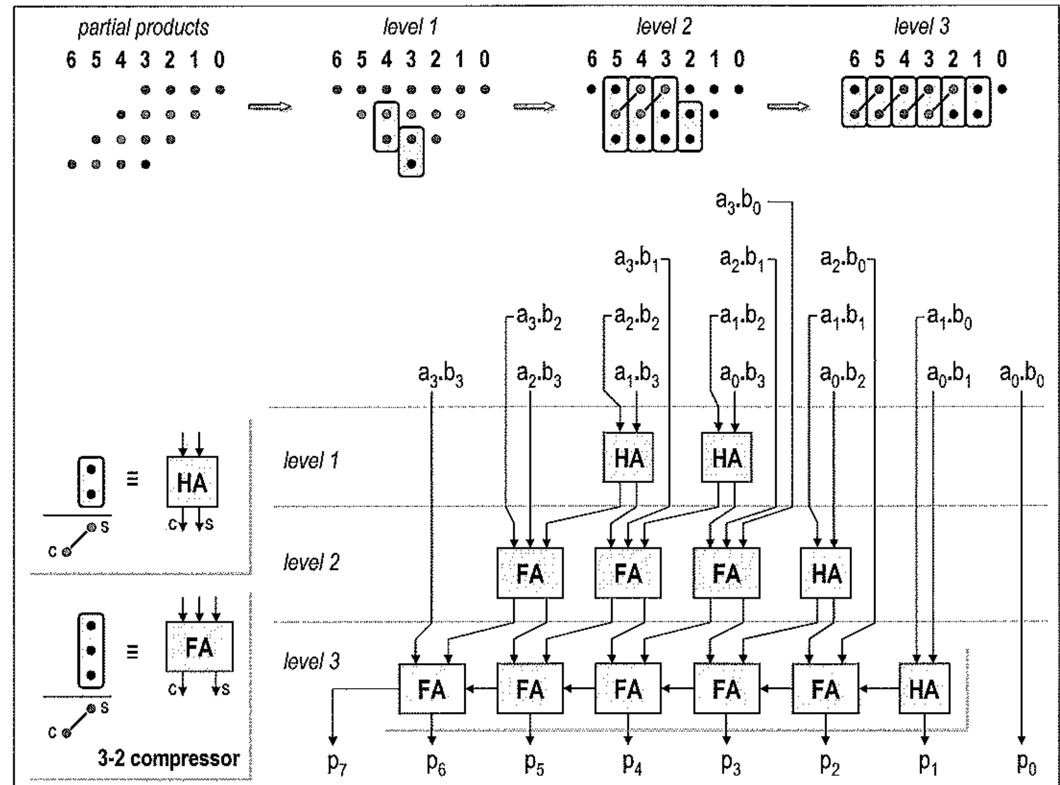


Tree multipliers generate all partial products in parallel using a tree of counters to reduce the partial products to sum and carry vectors and then sum these vectors using a fast carry-propagate adder. Tree multipliers offer a delay proportional to the logarithm of the operand length (n).

In the figure, the original matrix of partial products is reorganized into a tree shape to visually illustrate its varying depth. The challenge is to realize the complete matrix with a minimum depth and a minimum number of adder elements. The first type of operator that can be used to cover the array is a full adder, which takes three inputs and produces two outputs: the sum, located in the same column and the carry, located in the next one. For this reason, the FA is called a 3-2 *compressor.* It is denoted by a circle covering three bits. The other operator is the half-adder, which takes two input bits in a column and produces two outputs. The HA is denoted by a circle covering two bits.

The final step for completing the multiplication is to combine the result in the final adder. This can be a simple two-input adder but performance of this "vector-merging" operation is of key importance.

The presented structure is called the *Wallace-Tree multiplier,* and its implementation is shown. The tree multiplier realizes hardware savings for larger multipliers. The propagation delay is reduced as well. In fact, it can be shown that the propagation delay through the tree is equal to $O(\log_{3/2}(n))$ instead of O(n) for a carry save multiplier. While faster than the carry save structure for large multiplier word lengths, the Wallace multiplier has the disadvantage of being very *irregular,* which complicates the task of coming up with an efficient layout. Each tree has a different number of partial products and a different number of input carries. This irregularity is visible even in the four-bit implementation shown.



There are numerous other ways to accumulate the partial-product tree. A number of compression circuits has been proposed in the literature. They are all based on the concept that when full adders are used as 3:2 compressors, the number of partial products is reduced by two-thirds per multiplier stage. One can even go a step further and devise a 4-2 (or higher order) compressor.

The tree shown in the figure is iteratively covered with FA's and HA's, starting from its densest part (level 1). In a first step, HA's are introduced in columns 4 and 3. The reduced tree is shown (level 2). A second round of reductions creates a tree of depth 2 (level 3).

The final step for completing the multiplication is to combine the result in the final adder. This can be a simple two-input adder but performance of this "vector-merging" operation is of key importance.

A Wallace -Tree is often slower than a ripple carry multiplier in an FPGA

Many FPGA's have a highly optimized ripple carry chain connection. Regular logic connections are several times slower than the optimized carry chain, making it nearly impossible to improve on the performance of the ripple carry adders for reasonable data widths (at least 16 bits). Even in FPGA's without optimized carry chains, the delays caused by the complex routing can overshadow any gains attributed to the Wallace-Tree structure. For this reason, a Wallace-Tree multiplier does not provide any advantage over ripple carry multipliers in many FPGA's. In fact due to the irregular routing, they may actually be slower and are certainly more difficult to route.

5.2 Direct 2's Complement Integer Multiplication

$$a = \mathbf{a_{n-1}}\, a_{n-2} \ldots a_1\, a_0 \qquad [a] = -\,\mathbf{a_{n-1}}.2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

$$\times \qquad b = \mathbf{b_{n-1}}\, b_{n-2} \ldots b_1\, b_0 \qquad [b] = -\,\mathbf{b_{n-1}}.2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i$$

$$a \times b = p = \mathbf{p_{2n-1}}\, p_{2n-2} \ldots p_1\, p_0 \qquad [p] = -\,\mathbf{p_{2n-1}}.2^{2n-1} + \sum_{i=0}^{2n-2} p_i 2^i$$

$$[p] = +\,\mathbf{a_{n-1}}.\mathbf{b_{n-1}}.2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} a_i b_j 2^{i+j}$$

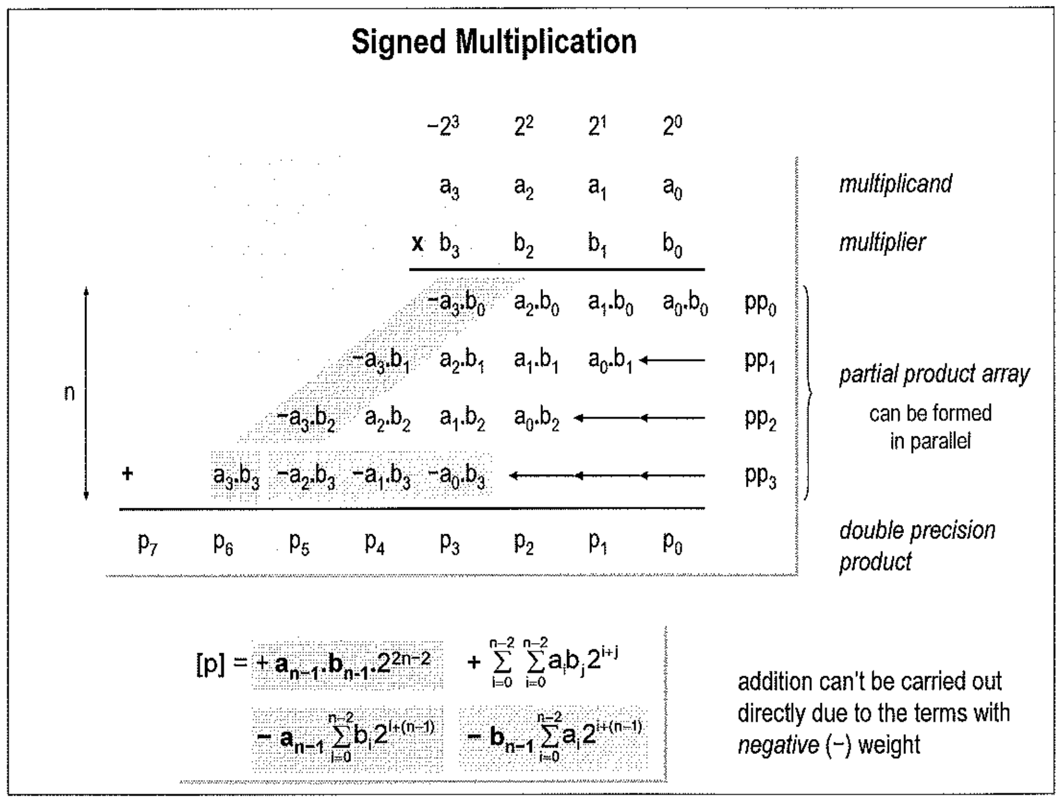
$$-\,\mathbf{a_{n-1}}\sum_{i=0}^{n-2} b_i 2^{i+(n-1)} - \mathbf{b_{n-1}}\sum_{i=0}^{n-2} a_i 2^{i+(n-1)} \quad \Big\} \; \mathbf{a_{n-1}}.b_i,\ \mathbf{b_{n-1}}.a_i$$

negative terms

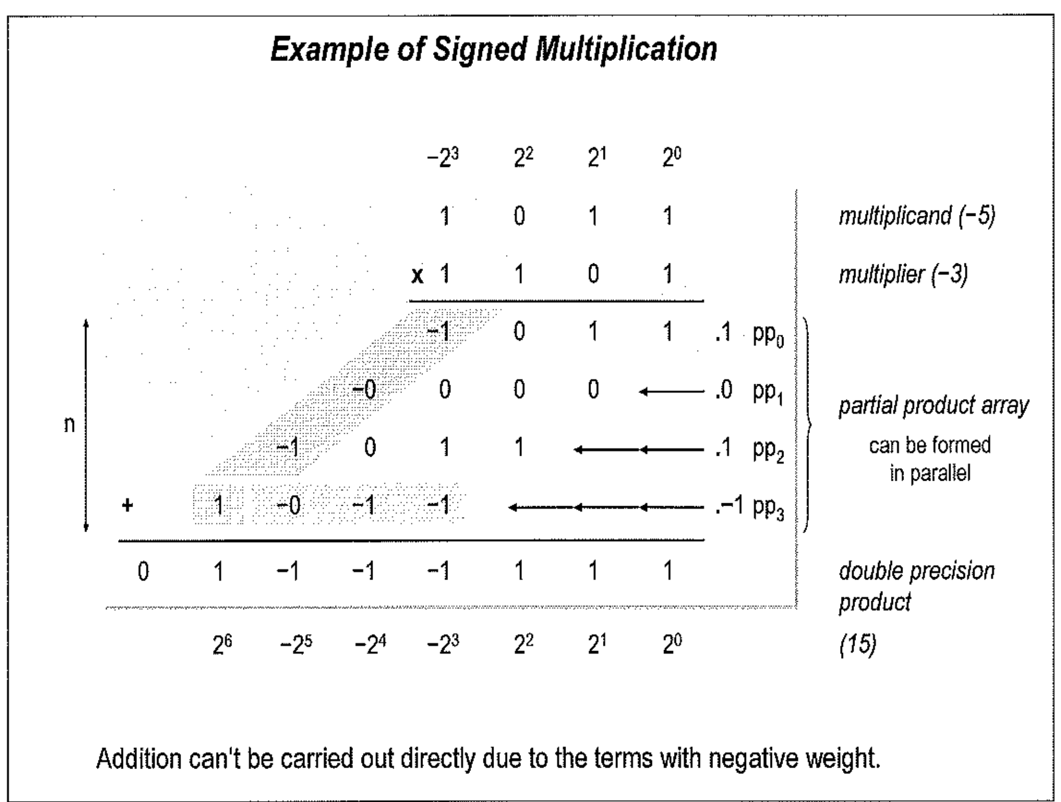
↓

subtractors

Multiplication of two's complement numbers generates signed partial products as shown in the figure. Since $a_{n-1}.b_i$ and $b_{n-1}.a_i$ have negative weights they should be subtracted rather than added. This makes the design difficult to implement because it requires adder and subtractor cells. Consequently several techniques have been proposed to handle partial products with negative and positive weight such as the Baugh-Wooley Algorithm.



Since $a_{n-1}.b_i$ and $b_{n-1}.a_i$ have negative weights, they should be subtracted rather than added. This makes the design difficult to implement because it requires adder and subtractor cells.



This example of the multiplication of two's complement operands illustrates the problem of the terms with negative weight. They cannot be handled directly with conventional full adders.

Two's Complement Negative Number

$$[a] = -\mathbf{a_{n-1}}.2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

$$[-a] = \mathbf{a_{n-1}}.2^{n-1} - \sum_{i=0}^{n-2} a_i 2^i$$

$$= \mathbf{a_{n-1}}.2^{n-1} - 2^{n-1} + 2^{n-1} - \sum_{i=0}^{n-2} a_i 2^i$$

(Booth Algorithm)

$$= -(1-\mathbf{a_{n-1}}).2^{n-1} + 2^{n-1} - \sum_{i=0}^{n-2} a_i 2^i$$

$$\sum_{i=k}^{v-1} 2^i = 2^{v-1} + \ldots + 2^k = 2^v - 2^k$$

$$= -(1-\mathbf{a_{n-1}}).2^{n-1} \left(+1 + \sum_{i=0}^{n-2} 2^i\right) - \sum_{i=0}^{n-2} a_i 2^i$$

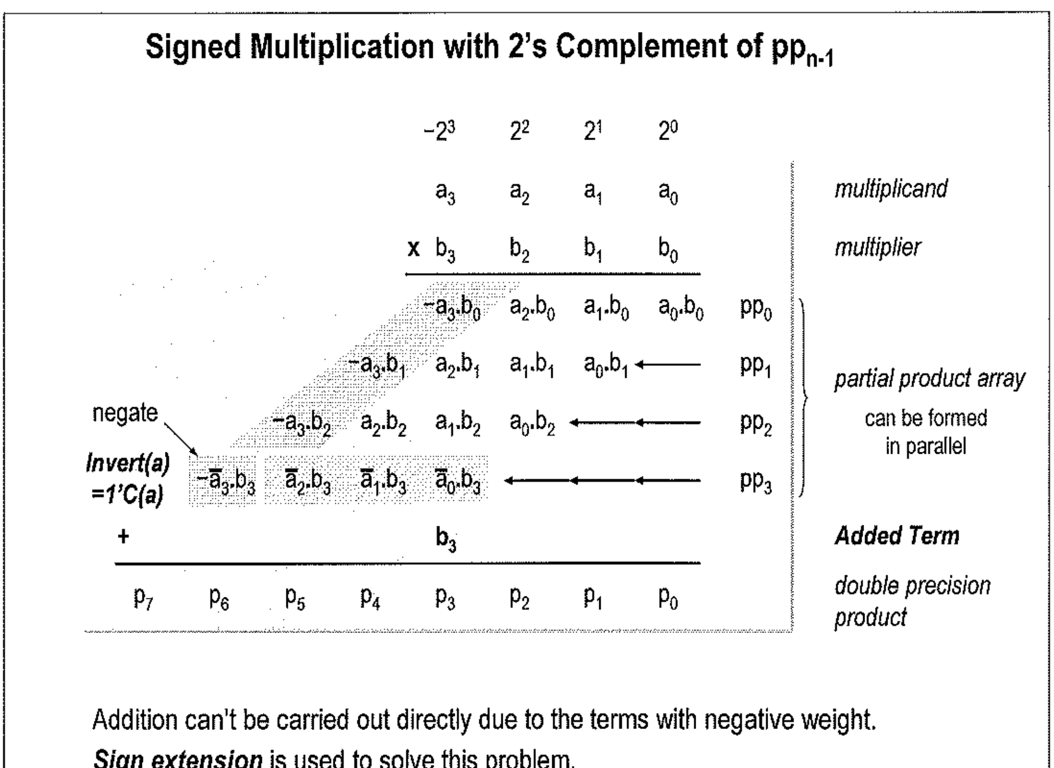
$$\leftarrow \sum_{i=0}^{n-2} 2^i = 2^{n-2} + \ldots + 1 = 2^{n-1} - 1$$

$$= -(1-\mathbf{a_{n-1}}).2^{n-1} + \sum_{i=0}^{n-2} (1-a_i).2^i + 1$$

$$= \underbrace{-(\bar{a}_{n-1}).2^{n-1} + \sum_{i=0}^{n-2} (\bar{a}_i).2^i}_{\text{1's complement}} + 1$$

$$a = \overline{(-a)} + 1$$

The complement –a of a two's complement number a can be found by inverting the bits and adding an LSB bit.



The last partial product results from the multiplication of the sign bit of the multiplier (b_3) with the multiplicand a.

$$pp_3 = a.(-b_3.2^3) = (-a).b_3.2^3$$

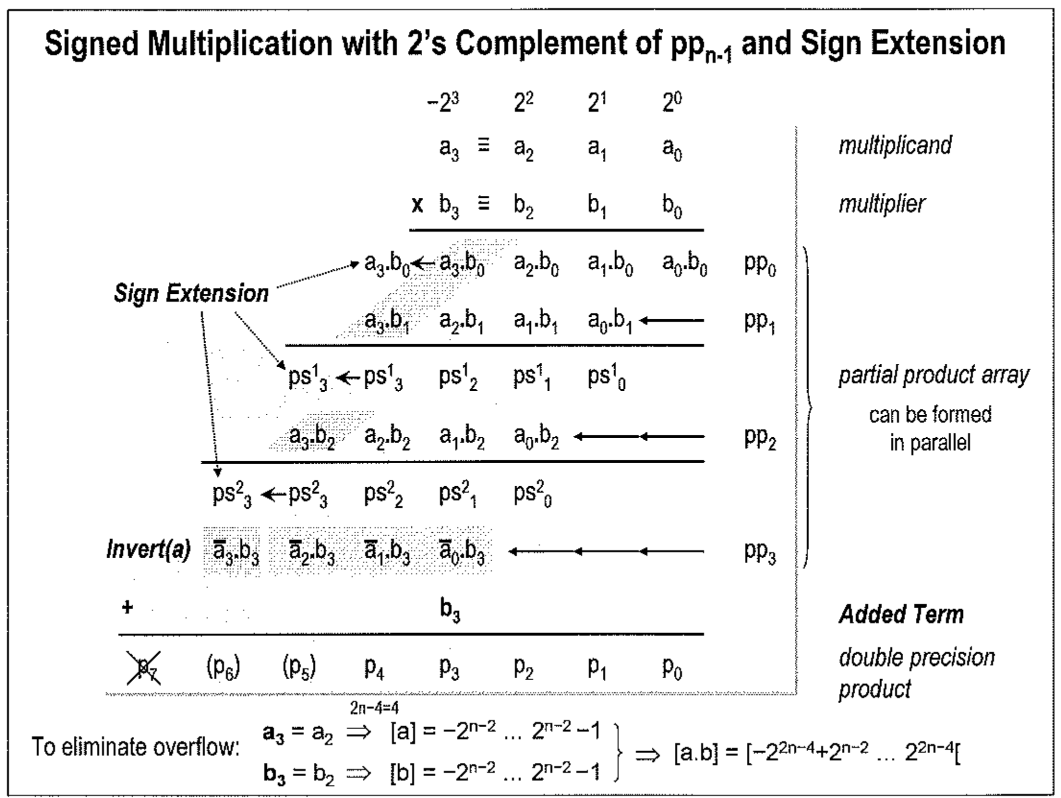
The complement –a of the multiplicand a is taken:

$$2'sC(a) = 1'sC(a) + 1$$

The complemented partial product becomes:

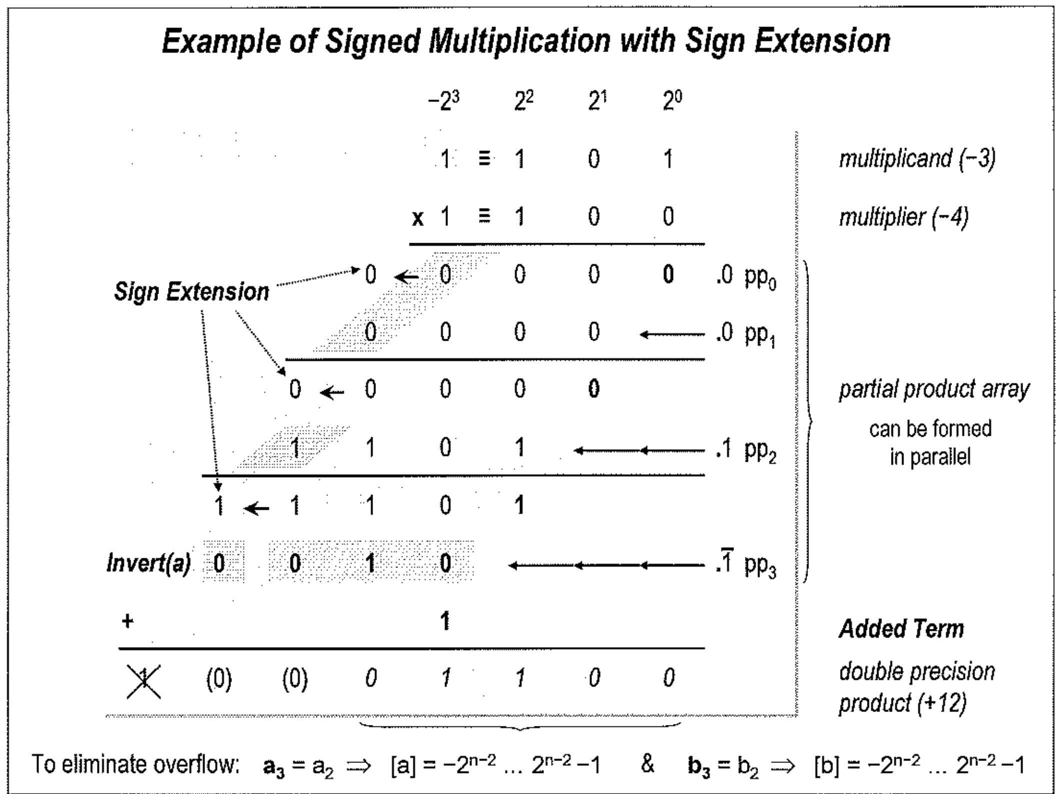
$$2'sC(pp_3) = (1'sC(a) + 1).b_3.2^3 = 1'sC(a).b_3.2^3 + b_3.2^3$$

This can be realised by inverting the a_i bits in the last partial product and adding the term $b_3.2^3$.



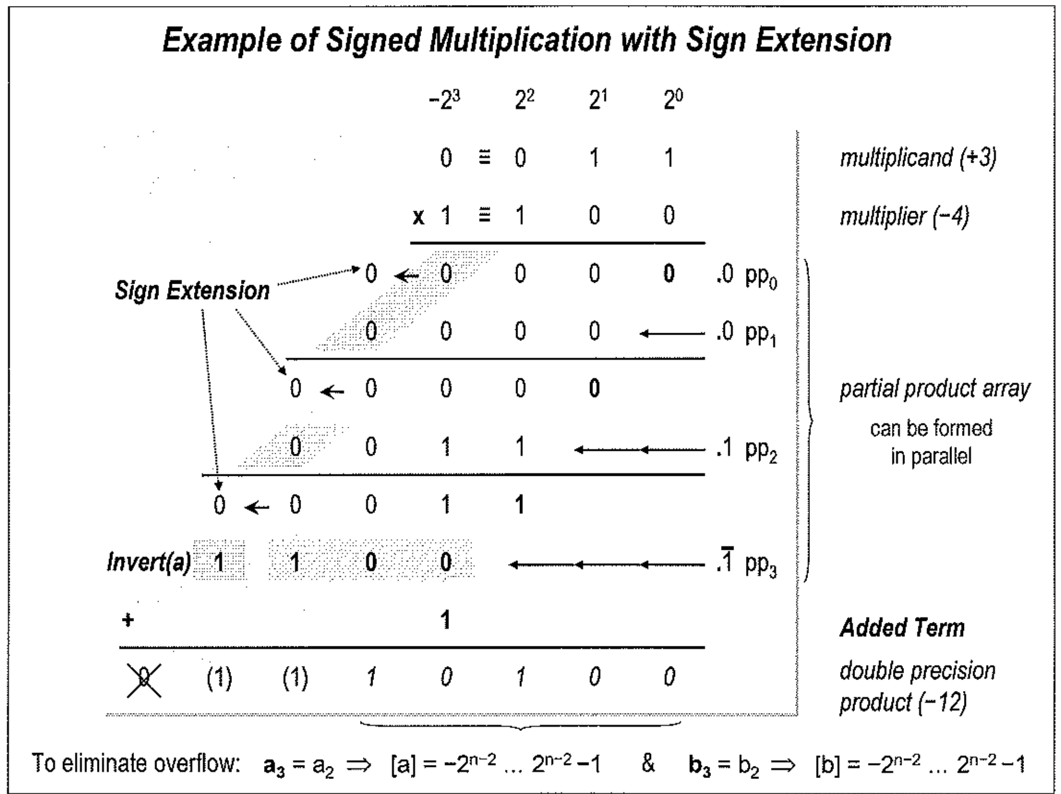
The partial products are not aligned. Due to the sign bits with negative weight, sign extension of the twos complement numbers is used. To limit the length of the partial products the are only extended with one bit. This reduces the hardware needed but also limits the range of the multiplicand and the multiplier.

To eliminate overflow, the MSB of the n-bit multiplier, b_{n-1}, is defined as a guard bit and is set equal to b_{n-2}. The multipliers must now lie in the range $[-2^{n-2}, 2^{n-2}-1[$. Also the multiplicand has a guard bit, so $a_{n-1} = a_{n-2}$.

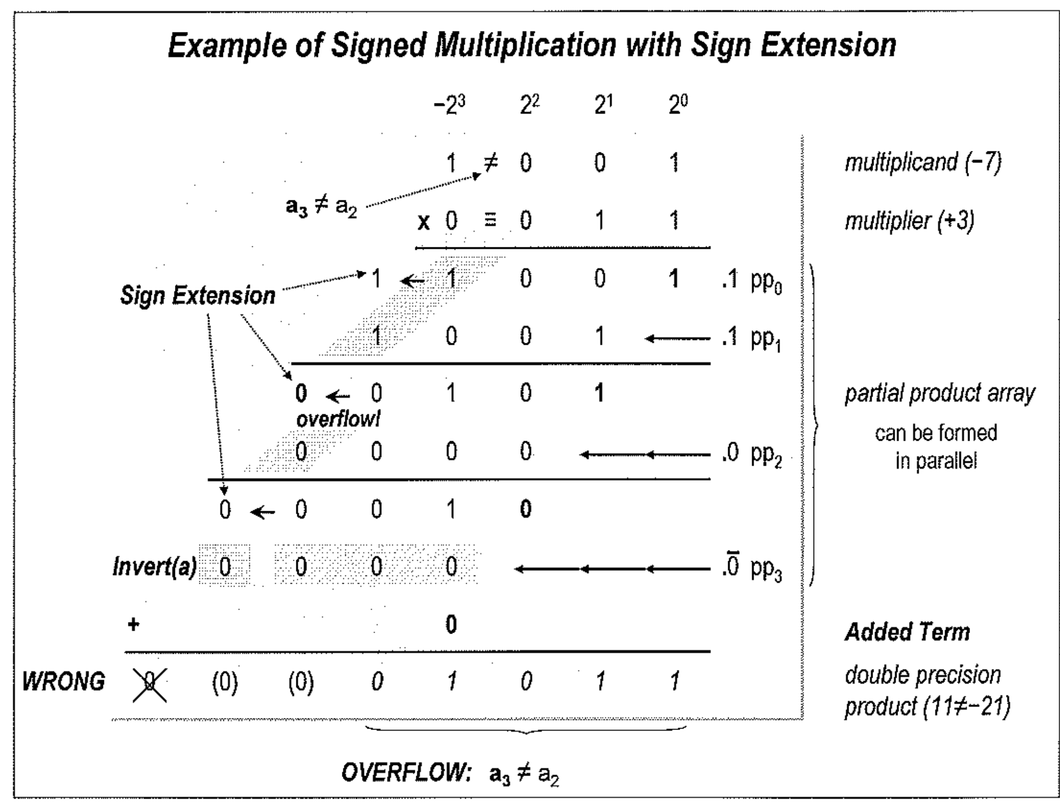


To eliminate overflow, the MSB of the n-bit multiplicand, a_{n-1}, is defined as a guard bit and is set equal to a_{n-2}. The multiplicands must now lie in the range $[-2^{n-2}, 2^{n-2}-1[$. Also $b_{n-1} = b_{n-2}$ for the multiplier.

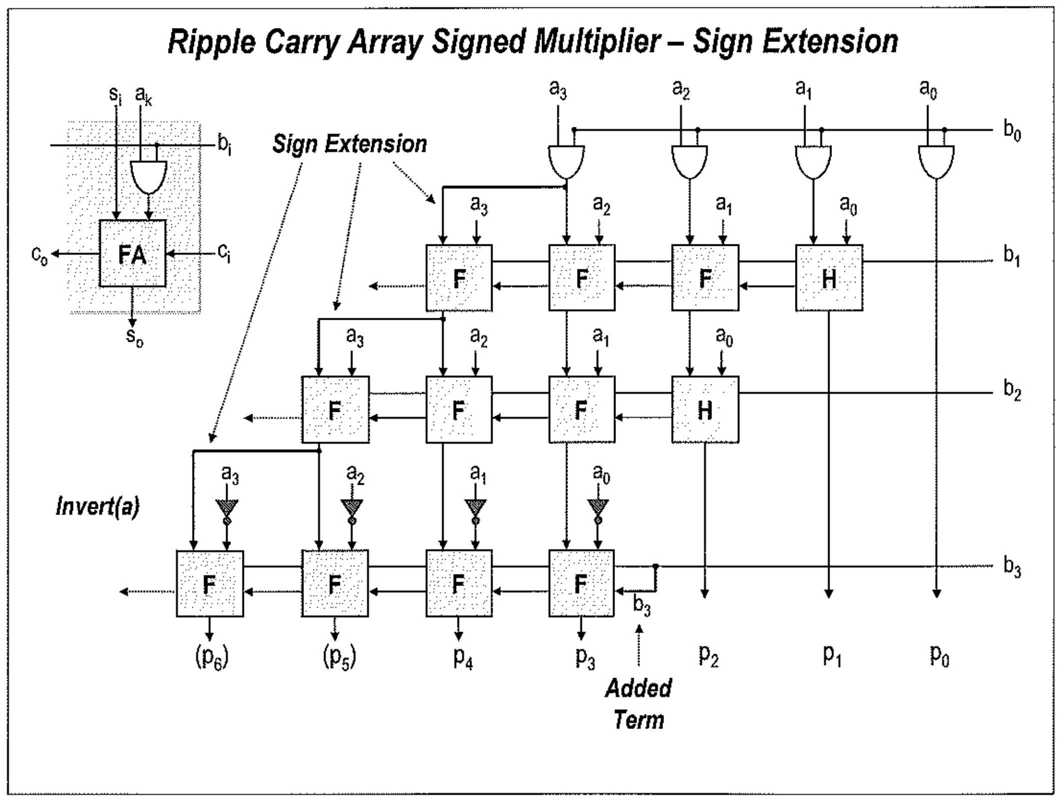
The example illustrates the principle of the complemented last partial product and the sign extension of the other partial products to implement the direct multiplication of signed integers.



To eliminate overflow, the MSB of the n-bit multiplicand, a_{n-1}, is defined as a guard bit and is set equal to a_{n-2}. The multiplicands must now lie in the range $[-2^{n-2}, 2^{n-2}-1[$. Also $b_{n-1} = b_{n-2}$ for the multiplier.



When the multiplicand (and/or multiplier) lies outside the range $[-2^{n-2}, 2^{n-2}-1[$, overflow occurs due to the limited sign extension (only one bit per partial product).



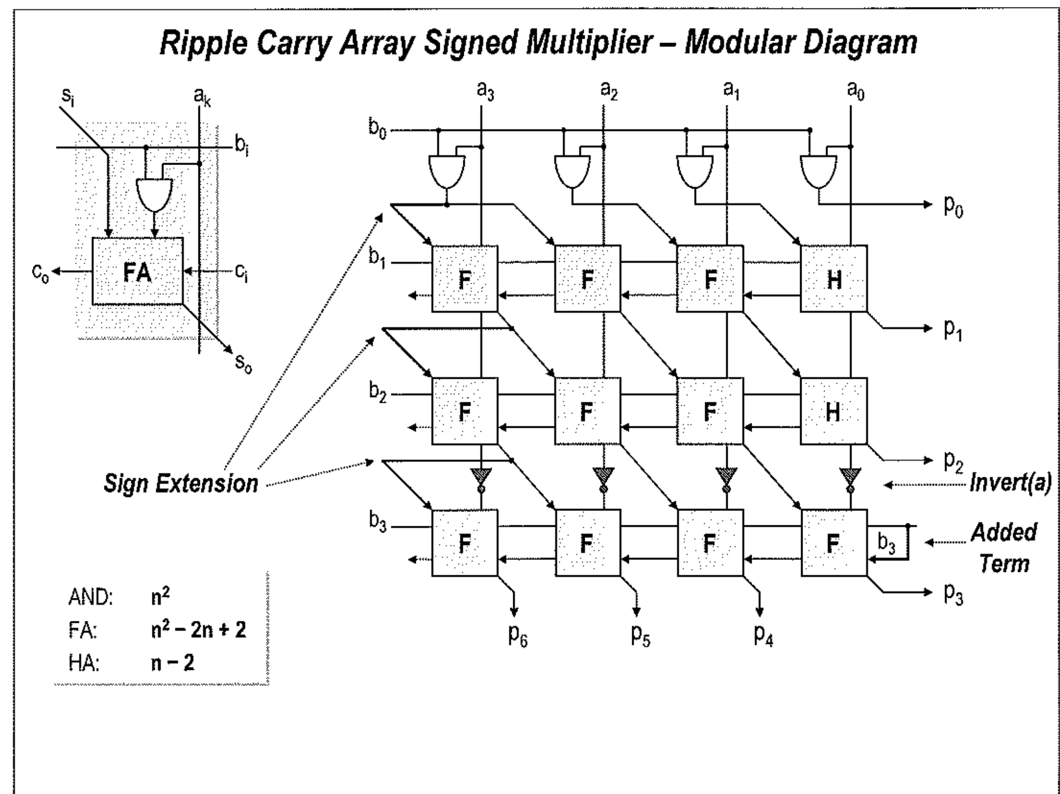
The array multiplier for signed numbers based on the complement of the last partial product, is illustrated.

The complemented partial product:

$$2'sC(pp_3) = (1'sC(a) + 1).b_3.2^3 = 1'sC(a).b_3.2^3 + b_3.2^3$$

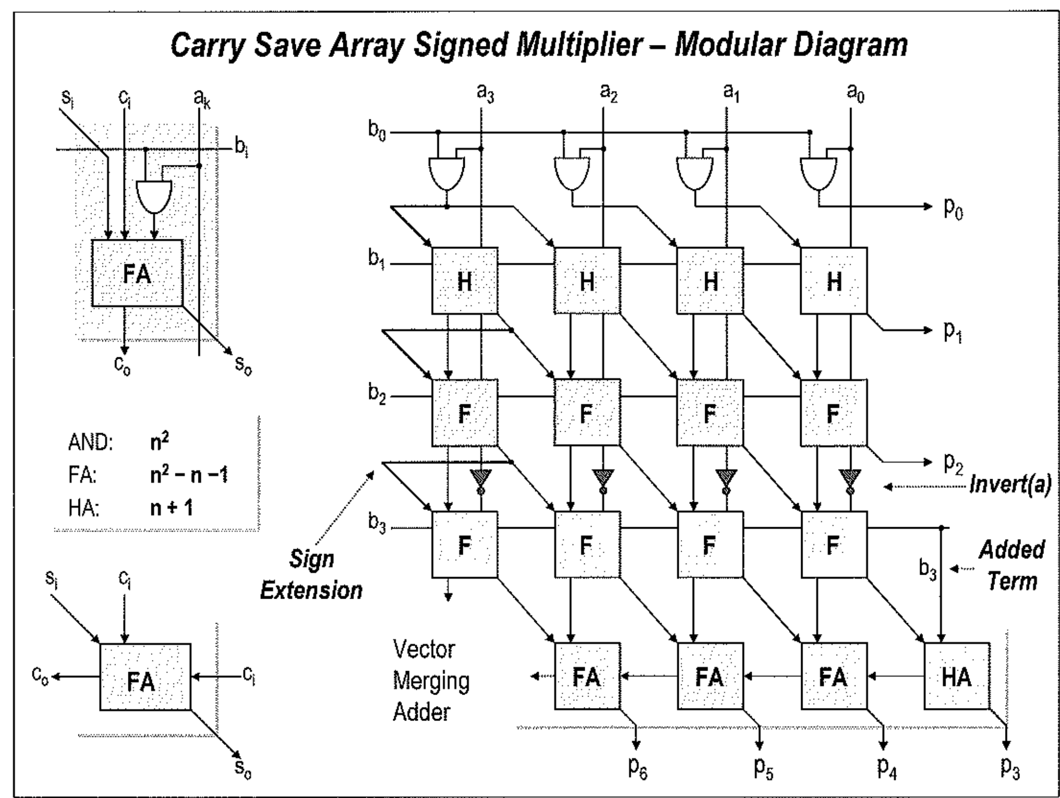
can be realised by inverting the a_i bits in the last partial product and adding the term $b_3.2^3$.

To align the partial products before addition, sign extension is used. Because only one bit sign extension is implemented, the range of the multiplicand and the multiplier must be limited to prevent internal overflow during addition of the partial products.



The overall structure can easily be compacted into a rectangle, resulting in a very efficient layout. A floor plan for the ripple carry array signed multiplier that achieves this goal is shown in the figure. Observe the regularity of the topology. This makes the generation of the structure amenable to automation.

An n by n array signed multiplier uses n^2 AND gates, (n^2 − 2n +2) FA's and (n − 2) HA's. The n inverters can be combined with the corresponding AND gates. Two extra Full Adders are needed due to the sign extension of the first partial product and the added term in the last partial product.

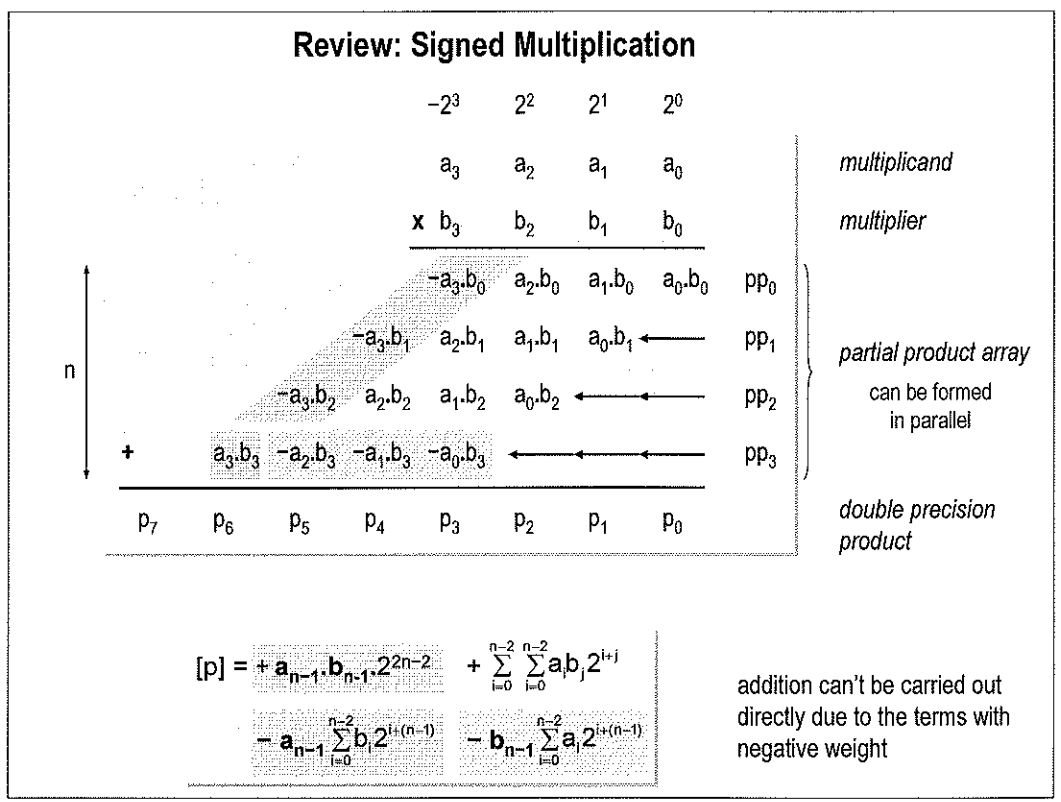


Carry Save Array Multiplier

The multiplication result does not change when the output carry bits are passed vertically downwards to the next partial product instead of only to the left in the own partial product, as shown in the figure. An extra adder is included to generate the final result: the Vector Merging Adder (VMA). The resulting multiplier is called a *carry save multiplier*, because the carry bits are not immediately added, but rather are "saved" for the next adder stage. In the final stage, carries and sums are merged in a fast carry-propagate (e.g., carry-look-ahead) adder stage.

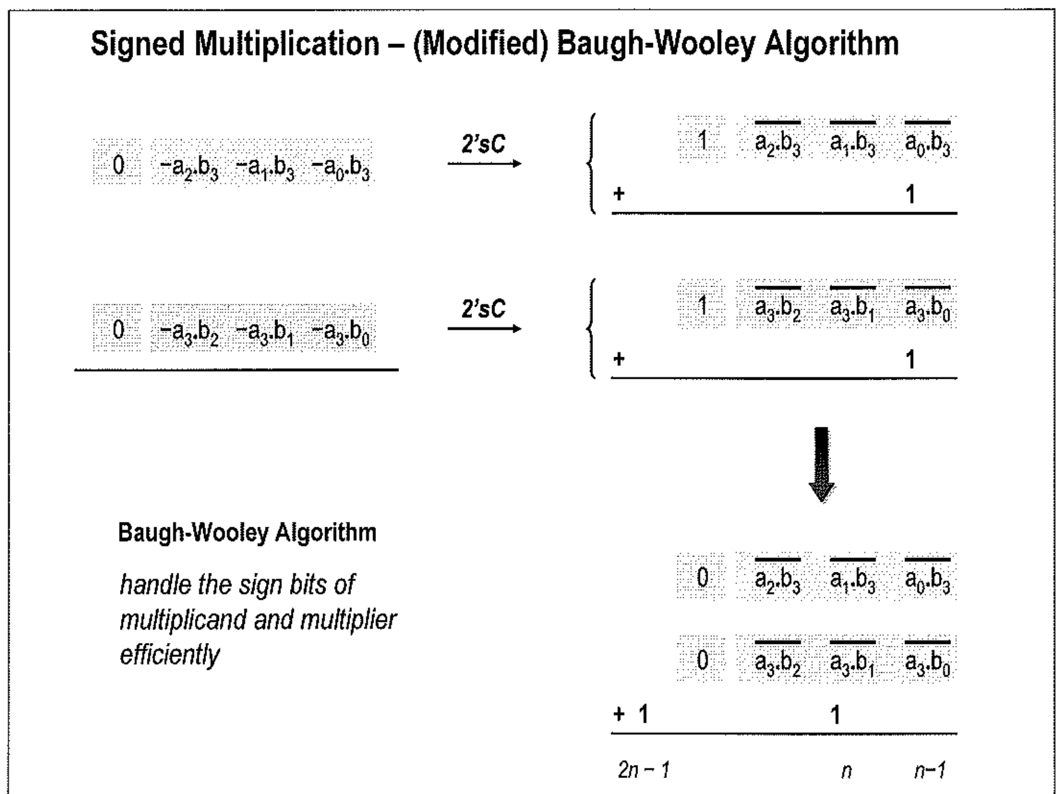
The AND gates generate the partial products. Full adders and half adders add the generated partial products. Sum outputs are connected diagonally and carry outputs are connected vertically. The last row of adders (Vector Merging Adder) which are connected from left to right, generates the n most significant product bits.

An n by n signed carry save array multiplier uses n^2 AND gates, $(n^2 - 2n - 1)$ FA's and $(n + 1)$ HA's.



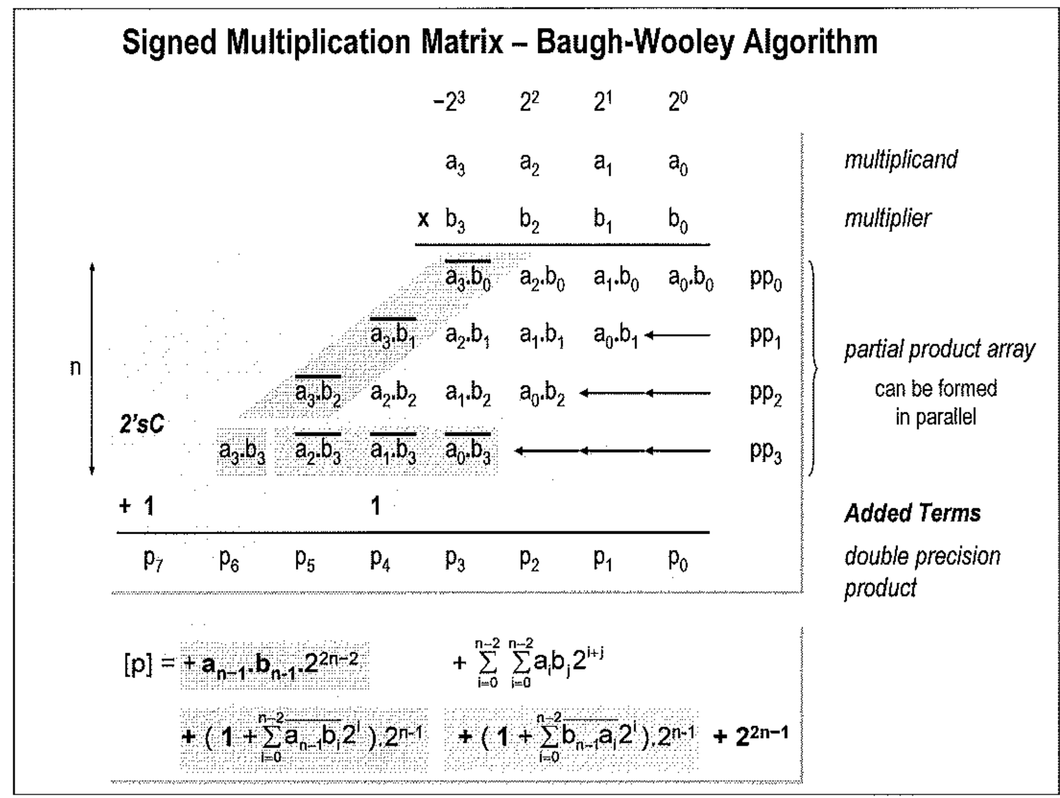
$$[p] = +a_{n-1}.b_{n-1}.2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} a_i b_j 2^{i+j} - a_{n-1}\sum_{i=0}^{n-2} b_i 2^{i+(n-1)} - b_{n-1}\sum_{i=0}^{n-2} a_i 2^{i+(n-1)}$$

Since $a_{n-1}.b_i$ and $b_{n-1}.a_i$ have negative weights they should be subtracted rather than added. This makes the design difficult to implement because it requires adder and subtractor cells.



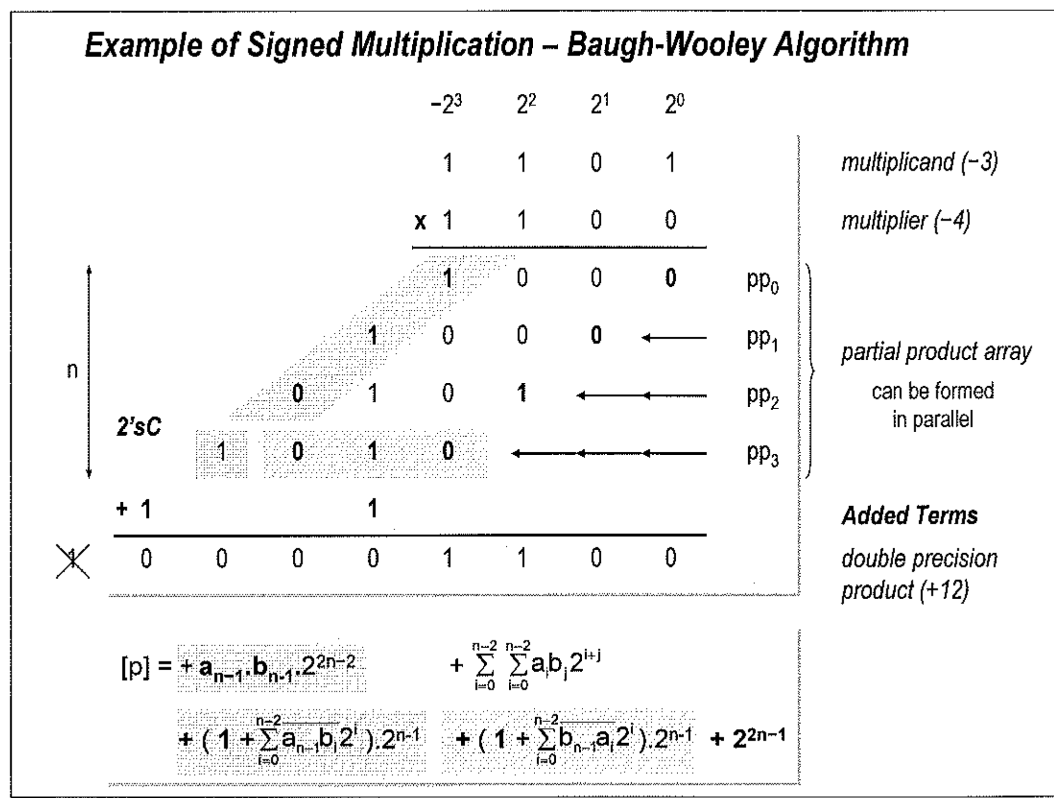
The BaughWooley algorithm provides a method for modifying the partial product matrix so that all the partial product bits have positive weights.

Two's complement multiplication is often realized using a variation of the Baugh-Wooley algorithm called the Complemented Partial Product Word Correction Algorithm. With this implementation partial product bits containing an a_{n-1} or b_{n-1}, but not both, are complemented and ones are added to columns n and (2n − 1). This is equivalent to taking the two's complement of the two negative terms in the equation of the signed multiplication.

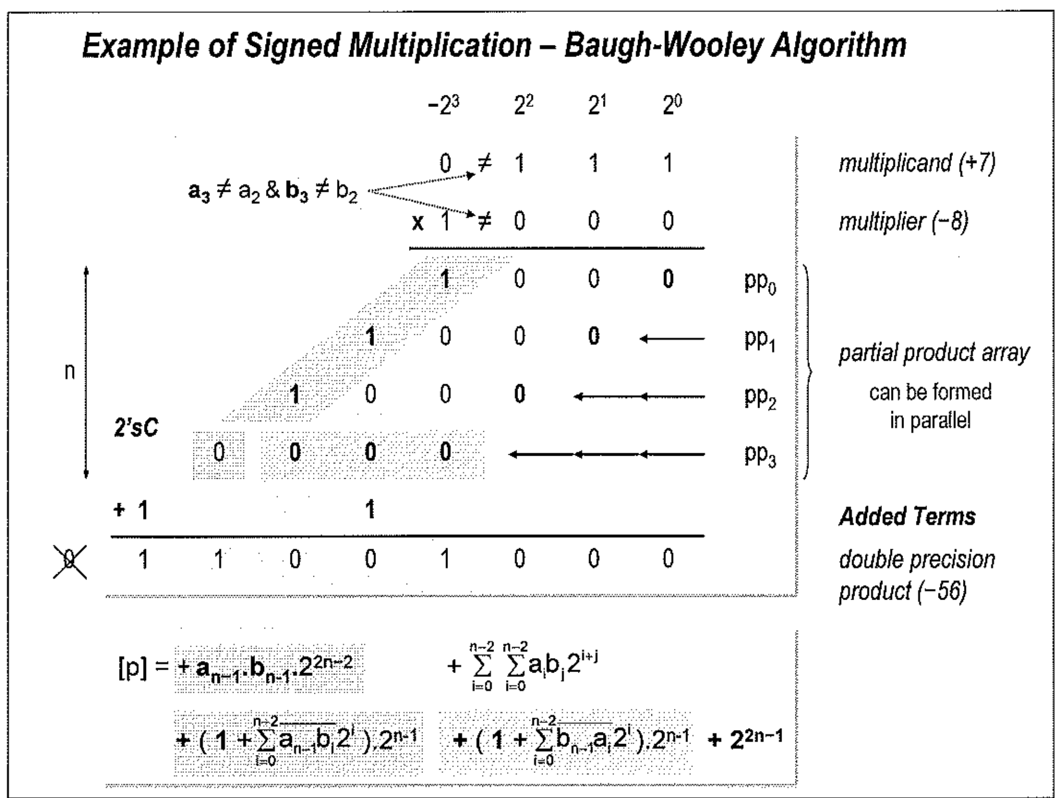


The multiplication matrix for two's complement multiplication based on the Complemented Partial Product Word Correction Algorithm is shown in the figure.

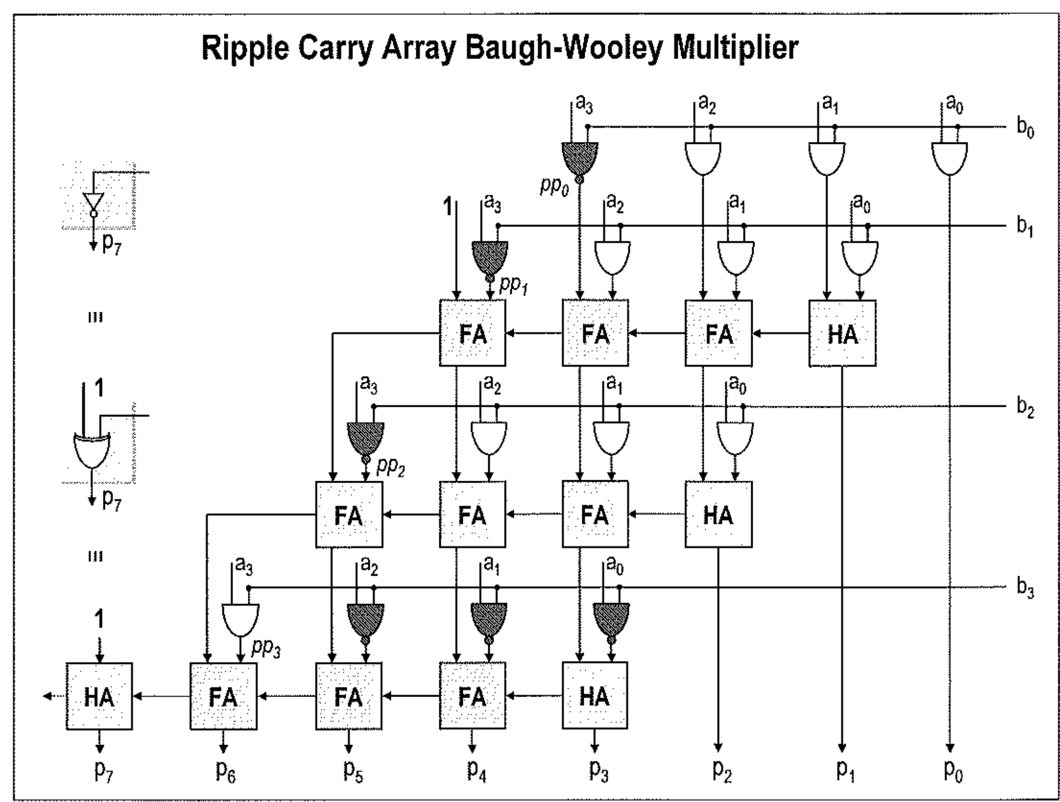
By taking the two's complement of the two negative terms in the equation of the signed multiplication results in a new equation with only positive terms.



An example illustrates the functionality of the multiplication matrix for two's complement multiplication based on the Complemented Partial Product Word Correction Algorithm.



There is no possibility for internal overflow, so the normal two's complement range of the multiplicand and multiplier can be used.

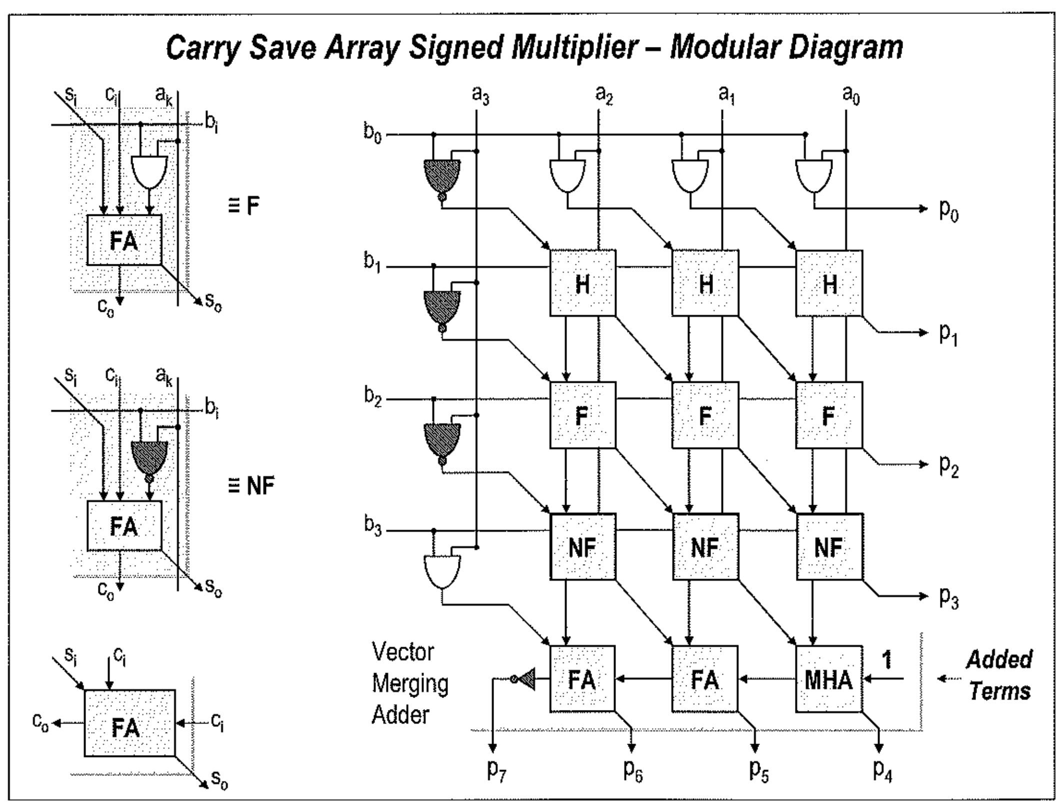


The design of an array multiplier that uses the Complemented Partial Product Word Correction Algorithm is shown. It is similar to the unsigned array multiplier design.

The AND gates in the leftmost column are replaced by NAND gates (except for the last row). Also in the last row the AND gates are replaced by NAND gates (except for the last column).

A '1' is added in column n and (2n − 1).

The last product bit p_{2n-1} is inverted to add the one in column n

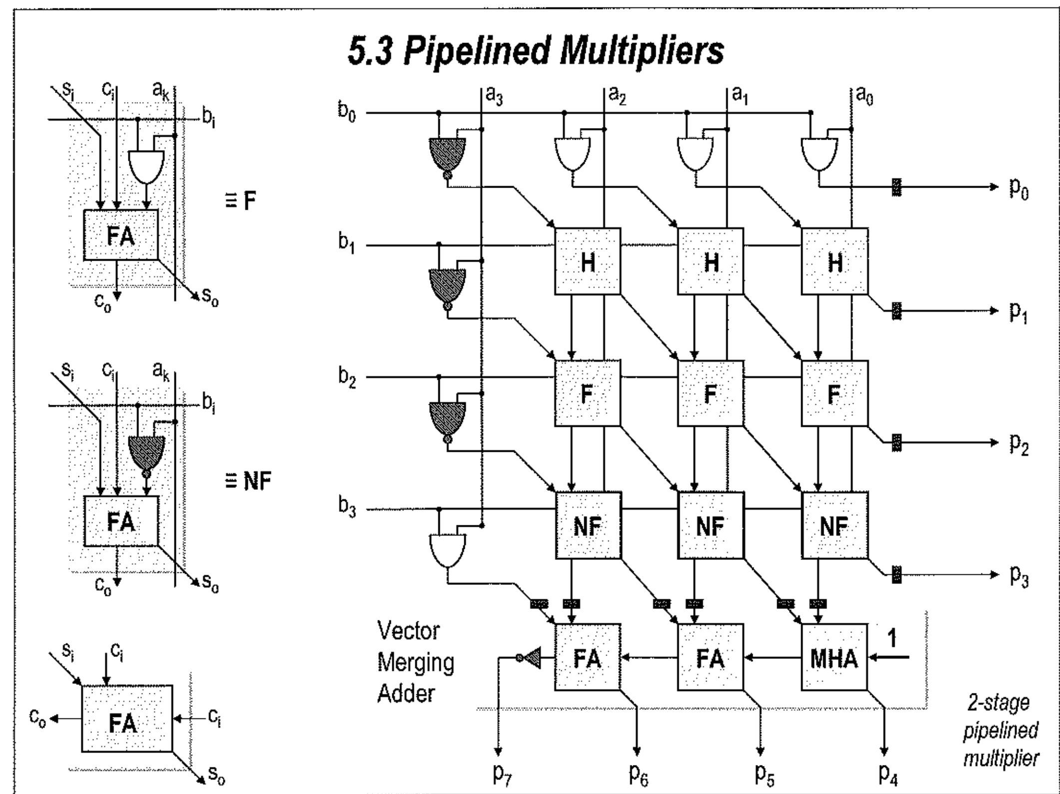


Also the design of an carry save array multiplier that uses the Complemented Partial Product Word Correction Algorithm is similar to the unsigned carry save array multiplier design.

AND gates in the leftmost column are replaced by NAND gates and the last row of modified full adder bocks F are replaced by negating modified full adder bocks NF.

The modified half adder MHA in the bottom right corner is a half adder that takes the sum and carry bits of the previous row and adds them with '1'. This cell has approximately the same area and delay as a regular half adder.

The last product bit p_{2n-1} is inverted to add the '1' in column (2n − 1) . Inverting p_{2n-1}has the same effect as adding '1' in column (2n − 1), because the carry out from this column is ignored.



The delay in the multiplier poses a major limitation on the maximum clock rate that can be attained. Array multipliers can be configured to allow a pipelined mode of operation, where the execution of separate multiplications overlaps. If this mode of operation is applied, the long delay associated with the carry propagating addition in the array multiplier can be minimized, since it determines the throughput of the pipeline. This approach yields extremely high speed implementations.

2-Stage Pipelined Multiplier

The structure of the pipelined multiplier is shown. The basic cells shown here are identical to that in the unpipelined multiplier.

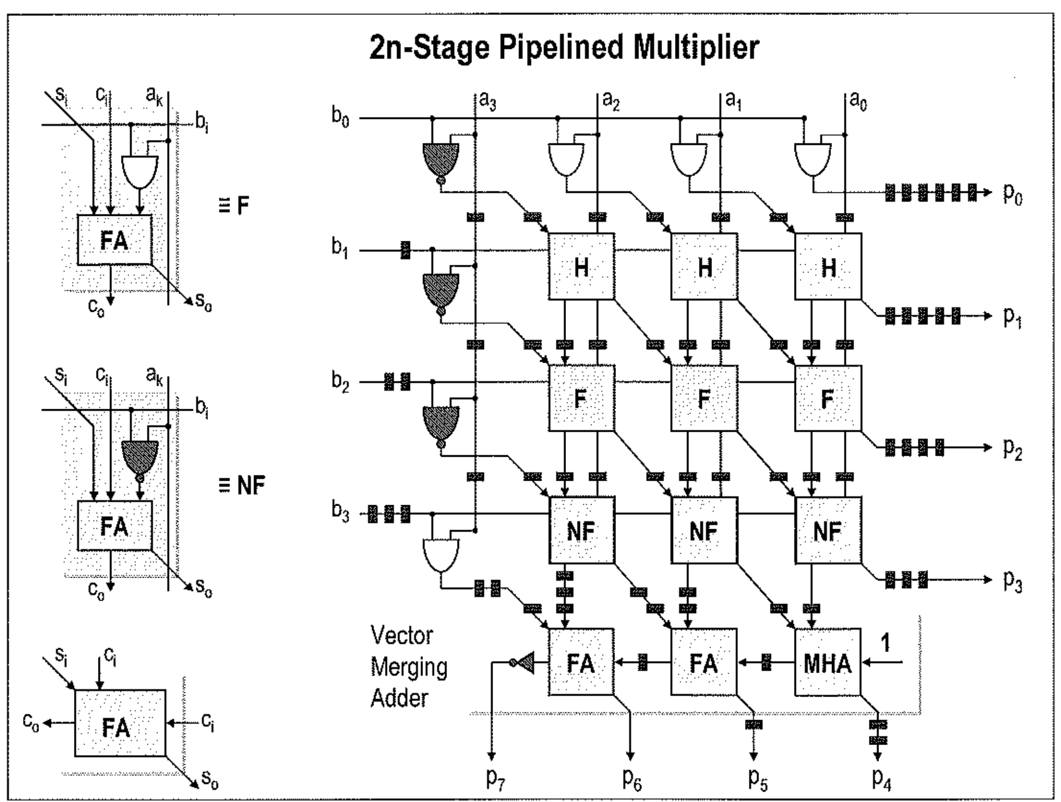
The delay of a multiplier is approximately $2.n.t_{FA}$, so 2 times t_{nadd}, the delay of an n-bit ripple carry adder.

The combinatorial logic of the carry save adder can be divided in two, by inserting registers a the inputs of the Vector Merging Adder. To align the product bits, the n LSB bits of the product must be delayed by 1 clock cycle.

The clock speed can be doubled (if both parts of the logic have approximately the same delay = balanced pipeline stages).

pipelined clock period = $T_{clk} > t_{nadd}$

latency = $2.T_{clk} \approx 2.t_{nadd}$



2n-Stage Pipelined Multiplier

The structure of the pipelined multiplier is shown. The basic cells shown here are identical to that in the unpipelined multiplier.

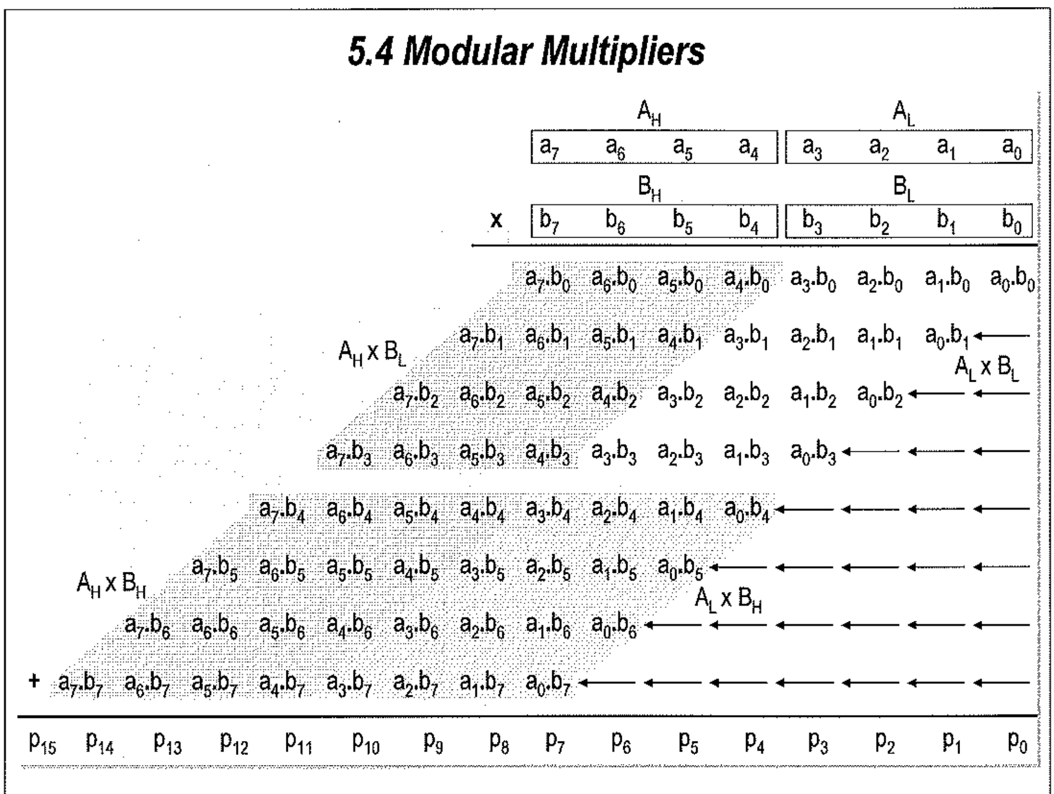
Registers are needed to propagate the multiplier and multiplicand bits to their destination and also to propagate the product bits that have been completed, which is done in parallel with the generation of new product bits.

Pipeline registers are inserted between the rows. This allows a carry propagation of only one position between any two consecutive rows. Registers are inserted for the bits of the multiplier b, to align them with the timing of the partial products. MSB bits are scheduled to arrive later.

The clock speed depends only on the delay in the cells of the multiplier.

pipelined clock period = $T_{clk} > t_{AND} + t_{FA}$

latency = $(n-1).T_{clk} \approx (2n-2).(t_{AND} + t_{FA})$



Based on the equation:

$$(A_H + A_L)x(B_H + B_L) = A_HxB_H + A_LxB_H + A_LxB_H + A_LxB_L$$

larger multipliers can be realised based on the use of multiple smaller multipliers. The subproducts (A_HxB_H , A_LxB_H , A_LxB_H , A_LxB_L) must be aligned to ensure the correctness of the result.

A n x n multiplier can be synthesised from:
. four (n/2) x (n/2) multipliers
. and a three operand addition operation

	n/2	n/2
	A_H	A_L
x	B_H	B_L

$A_L \times B_L$

$A_H \times B_L$

$A_L \times B_H$

\+ $A_H \times B_H$

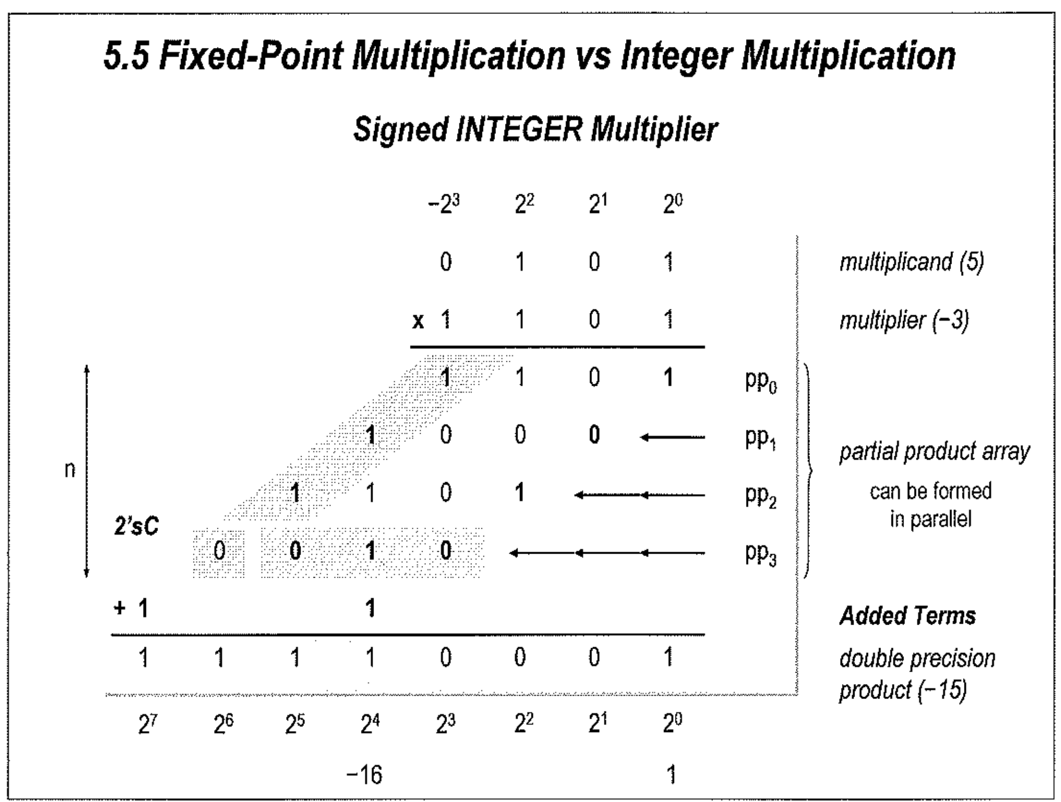
P

2n

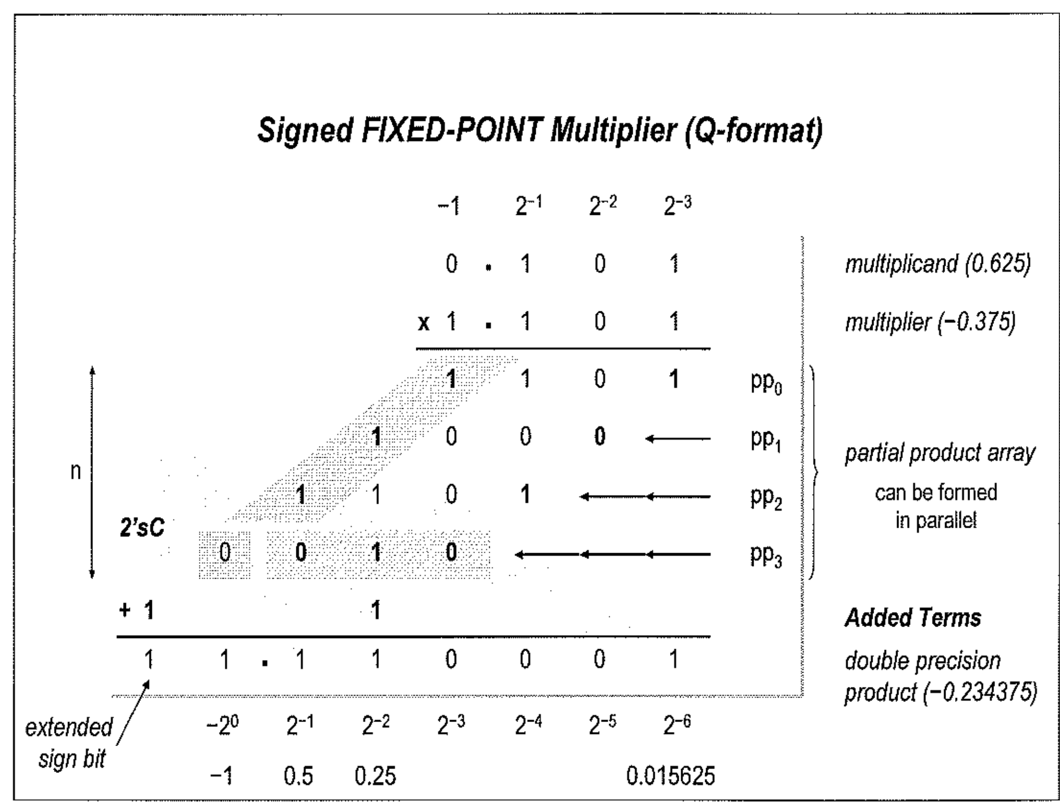
A n x n multiplier can be synthesised from:

. four (n/2) x (n/2) multipliers

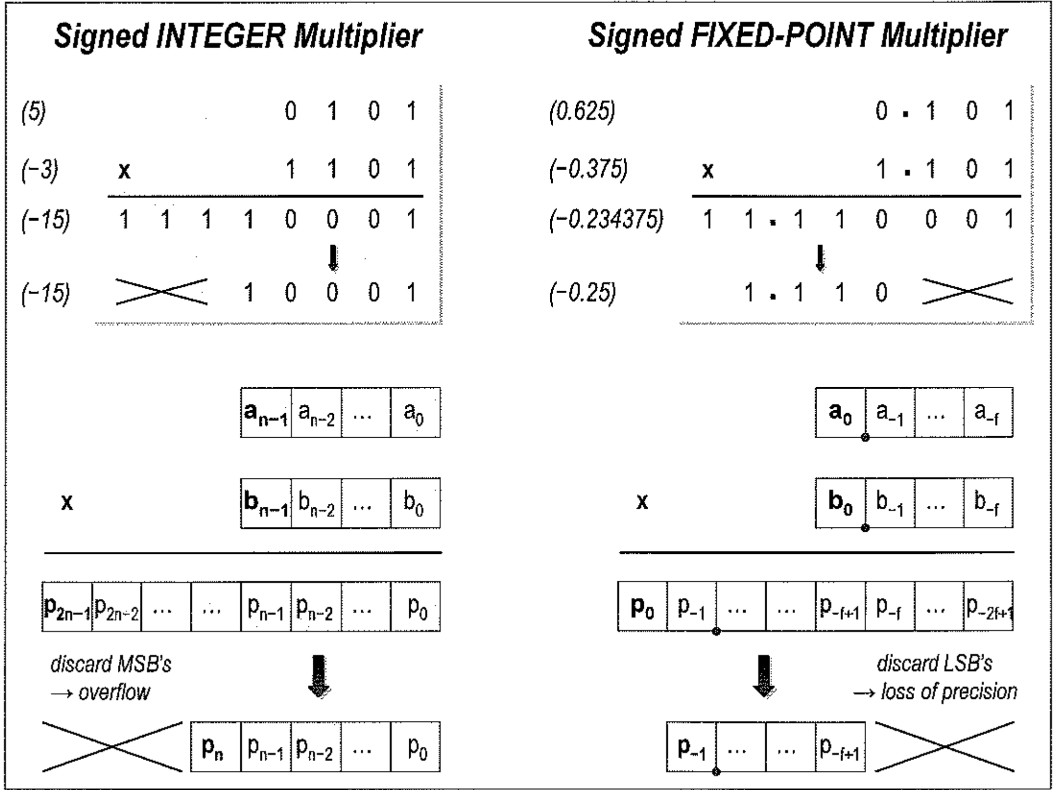
. and a three operand addition operation



The example illustrates the functionality of the multiplication matrix for the multiplication of two's complement INTEGER numbers, based on the Complemented Partial Product Word Correction Algorithm.



The example illustrates the functionality of the multiplication matrix for the multiplication of two's complement FIXED-POINT numbers, based on the Complemented Partial Product Word Correction Algorithm.

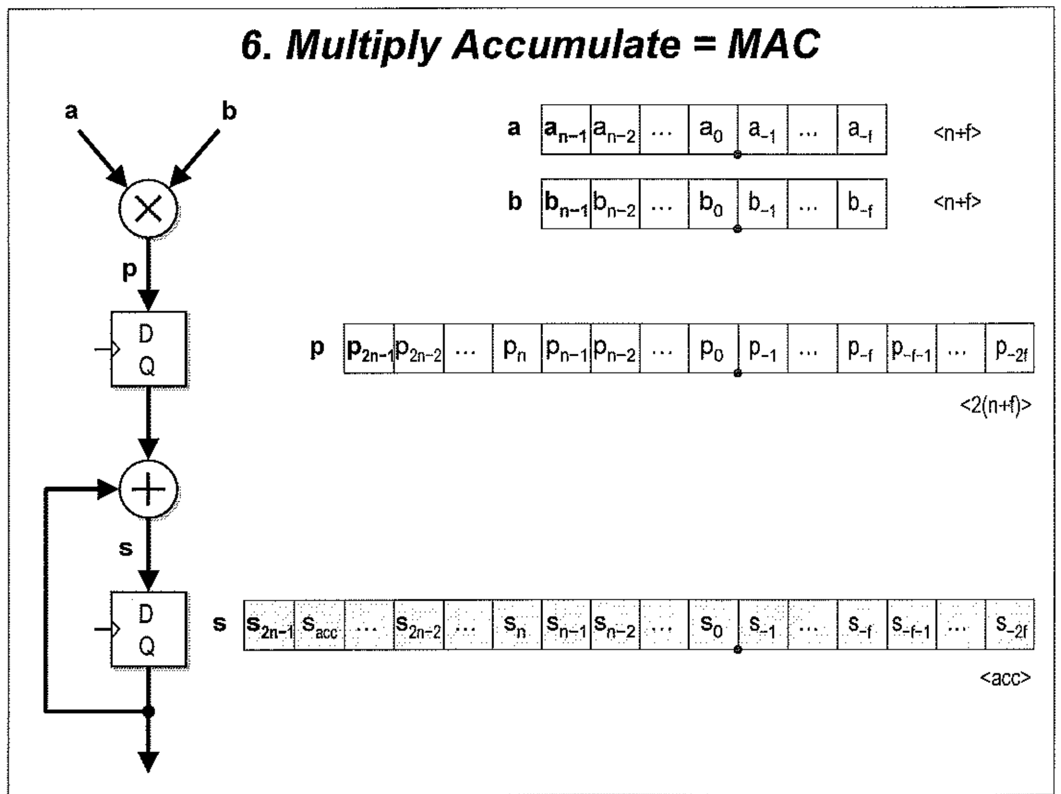


Signed Integer Multiplier

The MSB's are discarded. This can result in overflow errors.

Signed Fixed-Point Multiplier

The LSB's are discarded. This can result in loss of precision.



The multiply accumulate (MAC) operation is the kernel of various Digital Signal Processing algorithms.

The structure of a MAC unit is illustrated in the figure. The MAC unit presented consists of an (n+f)-bit multiplier and a acc-bit accumulator.

References

Baugh, Wooley, "A two's complement parallel array multiplication algorithm", IEEE Trans. Computers, 22: 1045-1047, Dec. 1973.

Parhami, "Computer Arithmetic", Oxford Univ. Press, 1999

HOGESCHOOL VOOR WETENSCHAP & KUNST **DE NAYER INSTITUUT**
SINT-KATELIJNE-WAVER

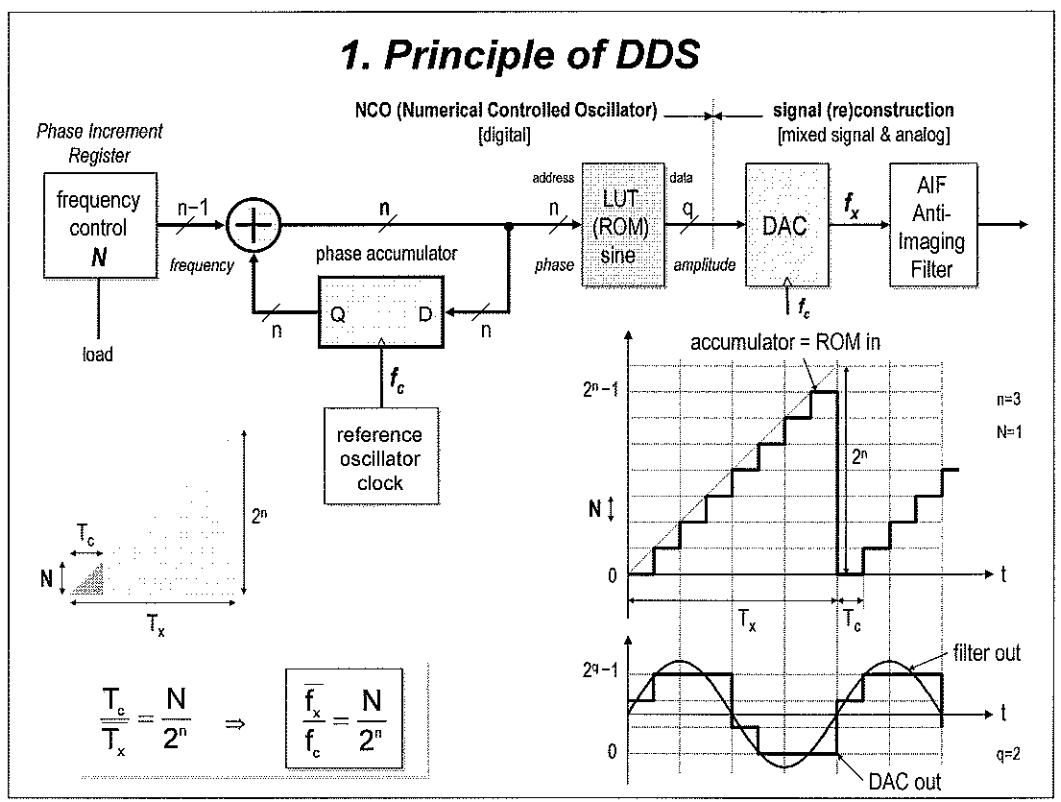
Digitale Synthese
Direct Digital Synthesis

ir. J. Meel
feb 2007

Direct digital synthesis (DDS) is a technique for using digital data processing blocks as a means to generate a frequency- and phase-tuneable output signal referenced to a fixed-frequency precision clock source. In essence, the reference clock frequency is "divided down" in a DDS architecture by the scaling factor set forth in a programmable binary tuning word.

Direct digital frequency synthesis (DDFS or simply DDS) generates real-life waveforms of repetitive nature by using digital data and mixed/analog signal processing blocks. The open-loop DDS is used especially for precise, fast frequency and phase tuneable output. Solutions can be implemented in a single chip and they play an ever-increasing role in digital waveform and agile clock generation and modulation.

As a discrete-time and discrete-amplitude system, DDS roots from sampling and quantizing theorems.



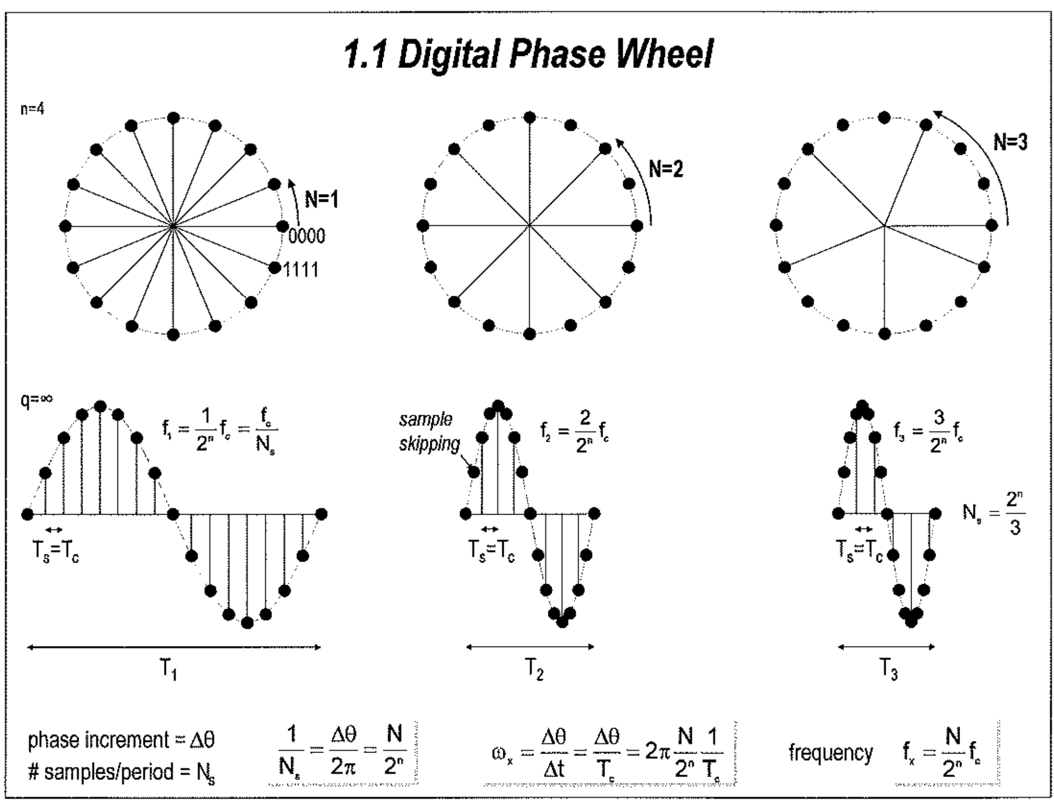
The high-level architecture of a generic DDS system can be viewed as a simple assembly containing three parts:

1. The time *reference* part: a fixed rate system clock.

2. The *digital* part, the Numerically Controlled Oscillator (NCO), generates numerical samples of (sine) waveform at clock rate. It consists of:
- An overflowing phase accumulator (register length n-bit), called also digital Frequency-to-Phase Converter or digital phase wheel (phase pointer: digital representation of the instantaneous phase). For linear phase samples, the overflowing accumulator is formed by an adder and phase accumulator register which integrates at a clock rate the frequency tuning data N in the PIR (phase increment register). The frequency setting value added to the accumulator controls the average frequency of the "overflowing (wrapping) event".
- A Look-Up Table (LUT, memory address length n-bit, data width q-bit), called also digital Phase-to-Amplitude Converter, or polar-to-rectangular transformation (projection of the real or imaginary component in time), or (sine) waveform mapping device - a memory. All techniques of Amplitude-to Phase Mapping (waveform mapping) are first modelled as a simple look-up table operation. The LUT contains one cycle of the waveform to be generated. The LUT translates phase information, being in digital form into numerical waveform sample.

3. The *mixed/analog* part reconstructs the analog wave with a digital to analog converter (DAC) and anti-imaging filter (AIF).
- Digital-to-Analog Converter (data length: q-bit), called also number-to-voltage (current) converter, realises a hybrid multiplication (the DAC function). The amplitude number sequence (the output of NCO) is fed to the DAC. Usually in NRZ (non-return-to-zero) mode.
- Anti-Image Filter, called also images-to-information (base) band converter, or analog smoothing (reconstruction, lowpass) filter. Images (linearly transposed replicas of signal in the super-Nyquist band) are natural consequences of DDS sampling process.

NCO-based DDS is a point (memory location)-skipping technique and runs at a constant update (clock)-rate.

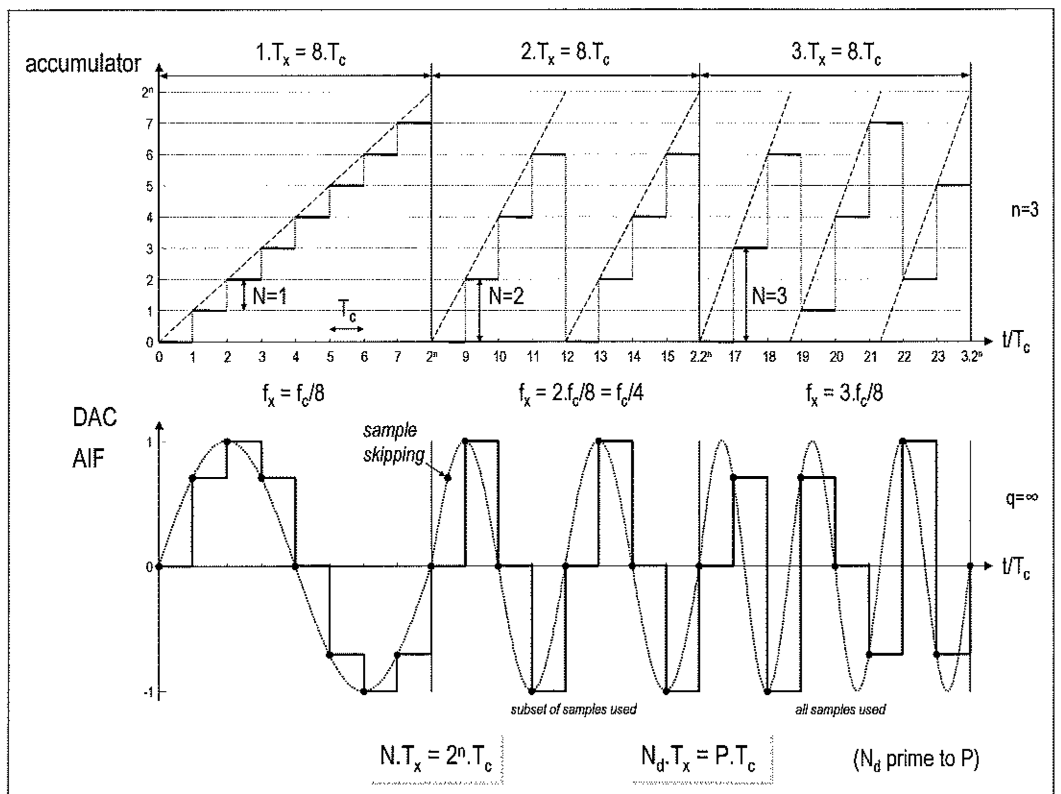


The accumulator acts as a Digital Phase Wheel. Each designated point on the phase wheel corresponds to the equivalent point on a cycle of a sine waveform. As the vector rotates around the wheel, a corresponding output sine wave is being generated. One revolution of the vector around the phase wheel, at a constant (but programmable) speed, results in one complete cycle of the output sine wave. The phase accumulator is utilized to provide the equivalent of the vector's linear rotation around the phase wheel. The contents of the phase accumulator correspond to the points on the cycle of the output sine wave. The number of discrete phase points contained in the wheel is determined by the phase increment value N, of the phase accumulator. The output of the phase accumulator is linear and cannot directly be used to generate a sine wave or any other waveform except a ramp. Therefore, a phase-to-amplitude LUT is used to convert the phase accumulator's instantaneous output value into the sine wave amplitude information that is presented to the D/A converter.

The phase accumulator is actually a modulo counter that increments its stored number each time it receives a clock pulse. The magnitude of the increment is determined by a digital word N contained in a Phase Increment Register that is summed with the overflow of the accumulator. The word in the PIR forms the phase step size between reference clock f_c updates. It effectively sets how many points to skip around the phase wheel. The larger the jump size, the faster the phase accumulator overflows and completes its equivalent of a sine wave cycle. For a n=32-bit phase accumulator, an N value of 0000...0001(one) would result in the phase accumulator overflowing after 2^{32} reference clock cycles (increments). If the N value is changed to 1000...0000 (2^{31}), the phase accumulator will overflow after only 2 clock cycles, or two reference clock cycles. There are only two samples per period. This is the maximum frequency the DDS can generate ($f_x = f_c/2$) due to the Nyquist criterion. This control of the jump size constitutes the linear frequency tuning resolution of the DDS architecture.

NCO-based DDS is a point (memory location)-skipping technique (and a constant interpolation of the stored signal) and runs at constant update (clock)-rate. As the DDS output frequency is increased, the number of samples per waveform cycles decreases.

This method is not to be confused with the common sample buffer playback, which is the traditional PPC (point-per-clock) synthesis at variable clock-rate.



Consider a continuous-time, normalized amplitude sinusoid with f_x the analog frequency and a fixed zero initial phase:

$$x(t) = 1.\sin(2\pi f_x t),\ |x(t)| \le 1$$

DDS builds up (reconstructs) the waveform from its numerical samples. Sampled at $f_c = 1/T_c$r ate, the discrete-time version of the sinusoid (with time index i):

$$x_i = x(i.T_c) = 1.\sin(2\pi(f_x/f_c).i)$$

The sample sequence x_i is P-periodic only, if the numerical frequency is rational:

$$f_x/f_c = N/2^n = N_d/P \text{ (some } N_d \text{ relative prime to P).}$$

This is the case of so-called *coherent sampling*. The finite memory requirement of the DDS imposes the periodicity $x_i = x_{i+P}$.

The rate at which the n-bit phase accumulator overflows in the NCO is controlled by N, the frequency tuning word. Accumulator overflow (mod 2^n) corresponds to a mod 2π operation: $N/2^n = \Delta\theta/2\pi$, so the output frequency (the rate of phase change):

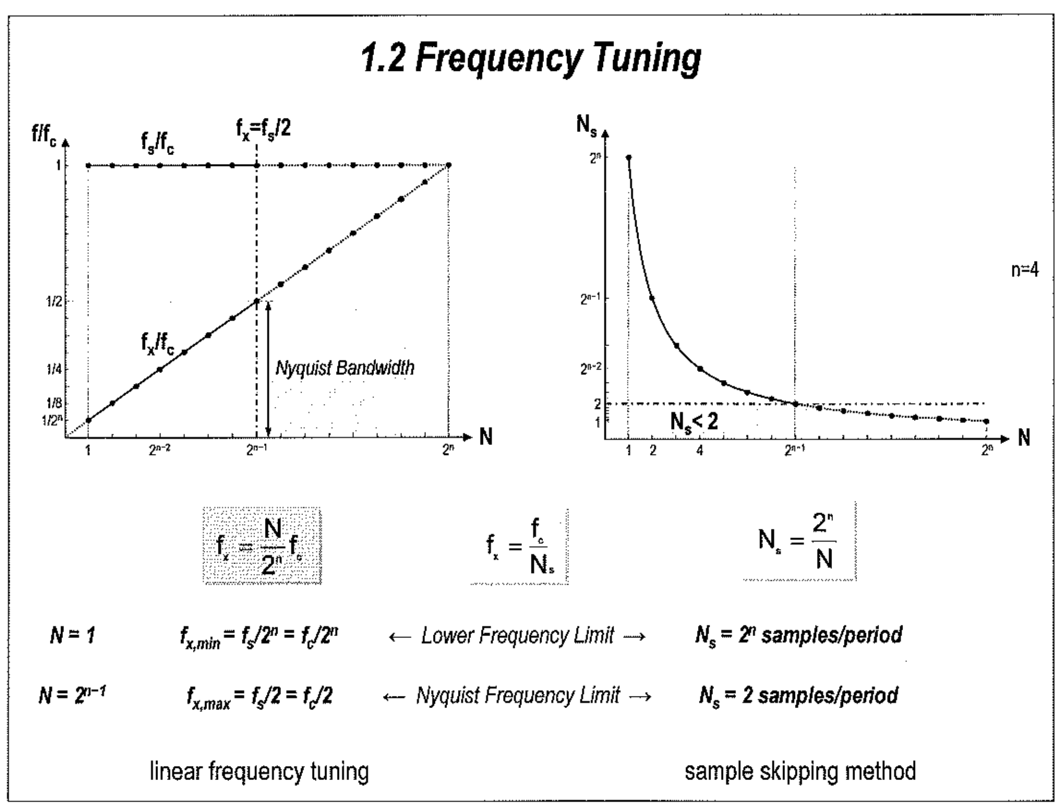
$$f_x = (1/2\pi).(\Delta\theta/T_c) = (N/2^n).f_c$$

Hence the *numerical frequency* of the DDS:

$$f_x/f_c = N/2^n = N_d/P < 1/2, \text{ some } N_d \text{ (prime to P)}$$

and P is the *numerical period* of the sample sequence. After P samples the waveform restarts with the same P samples. The waveform of P samples contains N_d periods (of length T_x) of the sinusoid:

$$N_d.T_x = P.T_c$$



The phase accumulator output will step sequentially through each LUT address for N=1 and produces the lowest frequency f_X. All the 2^n samples in the LUT are used to generate the waveform. The minimum output frequency, with the condition that every waveform point in RAM is output every waveform cycle, is defined by:

$$f_{x,min} = f_c/2^n$$

When the PIR is loaded with a larger value of N, the phase accumulator output will skip some RAM addresses, automatically "sampling" the data stored in RAM. Therefore, as the output frequency is increased, the number of output samples per waveform cycle will decrease. In fact, different groups of points may be output on successive waveform cycles. The minimum number of samples $N_{s,min}$ required to accurately reproduce a waveform will determine the maximum useful output frequency using the equation:

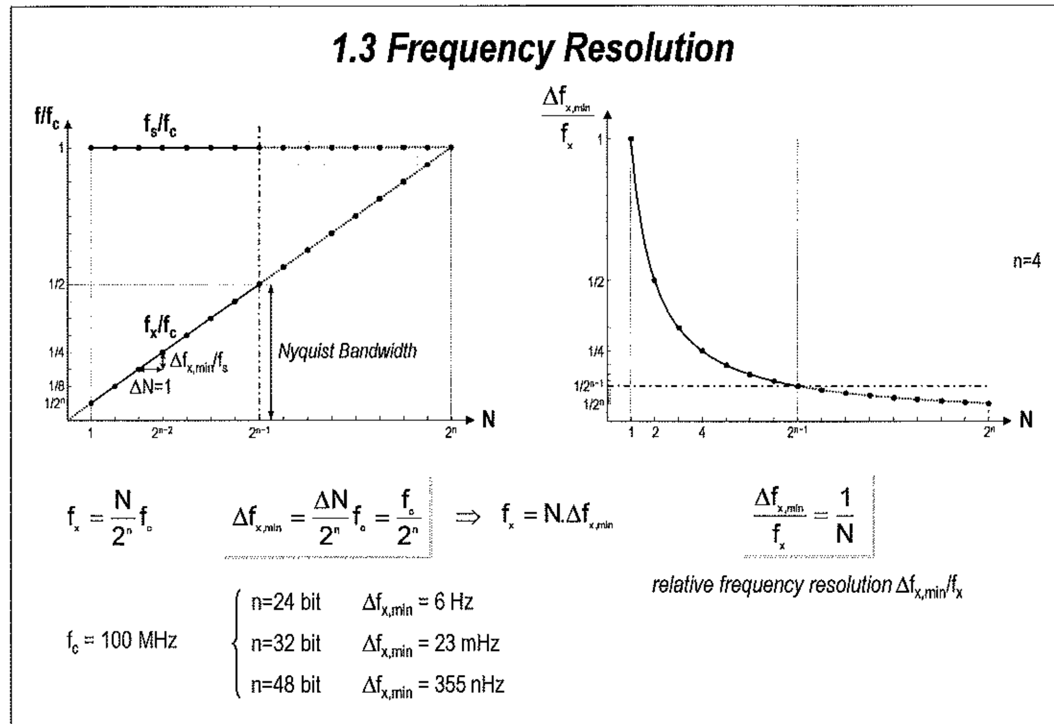
$$f_{x,max} = f_c/N_{s,min}$$

The rule governing waveforms is referred to as the Nyquist Sampling Theorem, which dictates that there is a minimum of two samples per cycle required from the highest frequency component to reconstruct the desired output waveform.

A common modulo-M clock divider produces an hyperbolic tuning of the output frequency.
DDS has the benefit to deliver a linear tuning of the output frequency.

NCO-based DDS is a point (memory location)-skipping technique (and a constant interpolation of the stored signal) and runs at constant update (clock)-rate. As the DDS output frequency is increased, the number of samples per waveform cycles decreases.

This method is not to be confused with the common sample buffer playback, which is the traditional PPC (point-per-clock) synthesis at variable clock-rate.

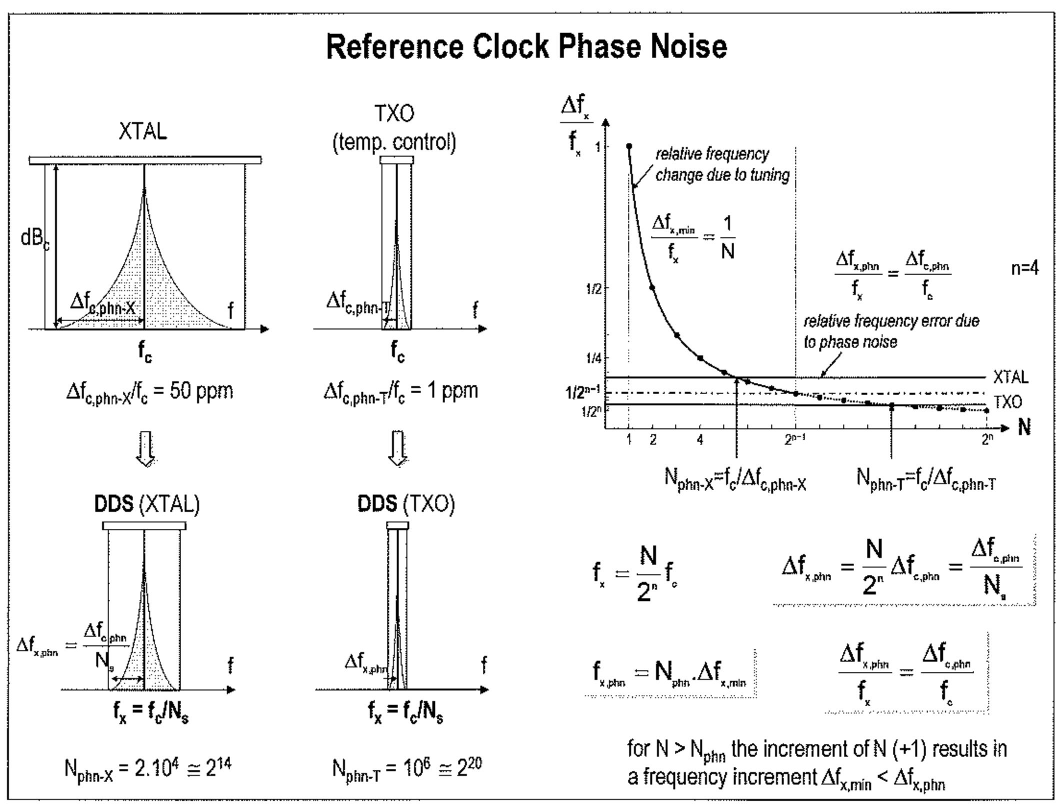


The *absolute frequency resolution* $\Delta f_{x,min}$ is the smallest frequency step and lowest DDS frequency. It is determined by f_c and n, the number of bits of the phase accumulator.

Any frequency can be generated by programming the phase change within the bit resolution of the phase accumulator. The absolute frequency resolution and the smallest frequency step can be determined by the following formula:

$$\Delta f_{x,min} = f_c / 2^n$$

Increasing the number of bits n in the phase accumulator with 1 bit reduces the smallest frequency step with a factor 2.



The reference clock of a DDS system is the major contributor to the phase noise of the system, even though its effect is reduced by the frequency division process of the DDS. The absolute phase noise of the DDS output will show an improvement over phase noise of the clock source itself of $f_c/f_x = N_s$, where f_c is the system clock frequency and f_x is the generated frequency and N_s the number of samples per period.

The frequency accuracy of the clock due to phase noise of the system clock is propagated through the DDS. Therefore, if the system clock frequency f_c is Δf_{phn} = 50 ppm higher than desired, the output frequency f_x will also be higher by an amount Δf_{phn} = 50 ppm. The system clock frequency f_c and the output frequency f_x have the same relative accuracy.

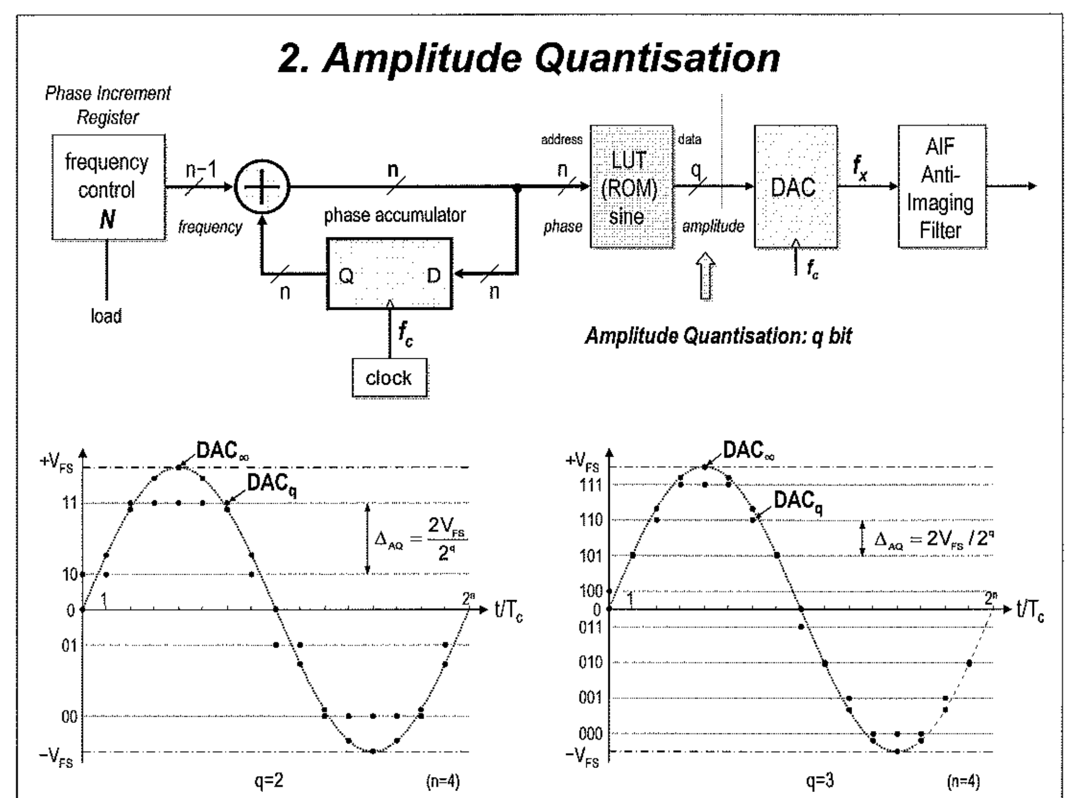
For N > N_{phn} a minimal increment Δf_x of the output frequency f_x is masked by the system clock frequency phase noise Δf_{phn}.

XTAL f_c = 104.8 MHz $\Delta f_{c,phn}/f_c$ = 100 ppm = 1/10^4^ — $\Delta f_{c,phn}$ = 10.48 kHz

acc. n = 20 → 2^n = 1048576 ≅ 10^6 — $\Delta f_{x,min} = f_c / 2^n$ = 100Hz

N	$f_x = N. \Delta f_{x,min}$	$\Delta f_{x,min}$	$\Delta f_{x,min}/f_x$ = 1/N	$\Delta f_{x,phn}/f_x$	$\Delta f_{x,phnx} = f_x.(\Delta f_{x,phn}/f_x)$
1	100 Hz	100 Hz	1	$1/10^4$	10 mHz
2	200 Hz	100 Hz	1/2	$1/10^4$	20 mHz
3	300 Hz	100 Hz	1/3	$1/10^4$	30 mHz
1000	100,0 kHz	100 Hz	1/1000	$1/10^4$	10 Hz
1001	100,1 kHz	100 Hz	1/1001	$1/10^4$	10,01 Hz
10.000	1 MHz	**100 Hz**	**1/10.000** ≡	**$1/10^4$**	**100 Hz**
100.000	10 MHz	100 Hz	1/100.000	$1/10^4$	1 kHz
		absolute error fixed		*relative error fixed*	

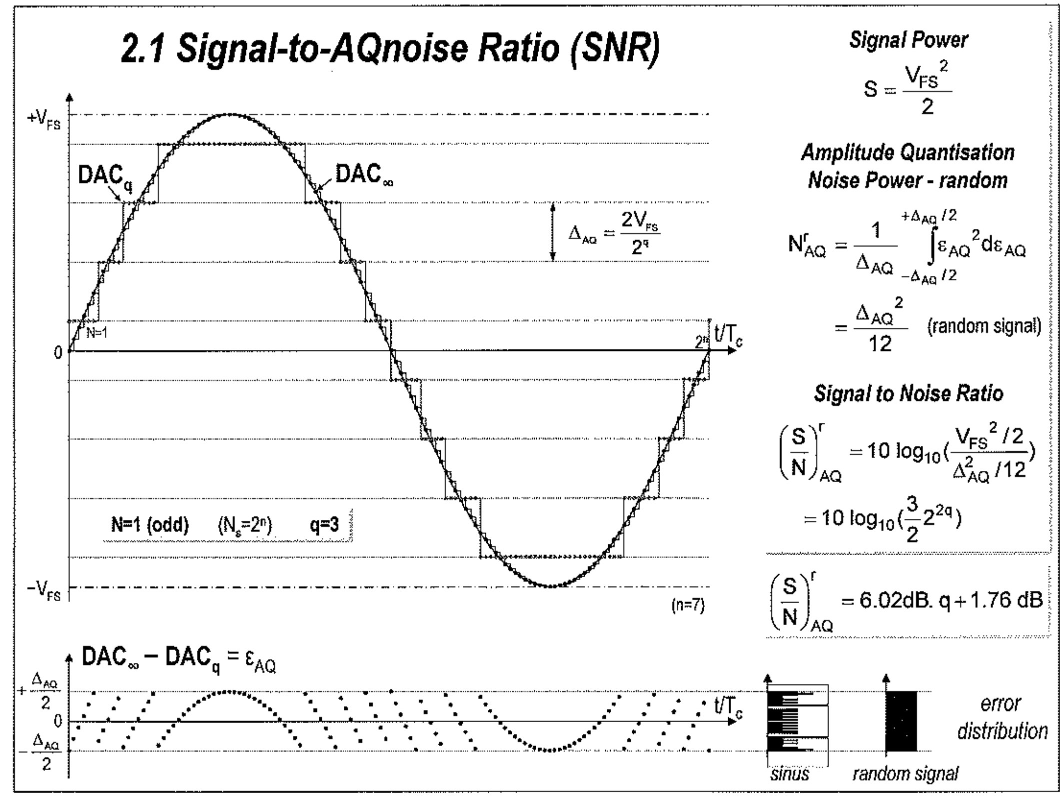
100 Hz f_x



Amplitude quantization occurs in the sine lookup process. The lookup table (LUT) takes in a fixed number of bits of phase information and converts it to the equivalent sine amplitude. Since an ideal sine representation would require an infinite number of bits for most values, the value must be truncated.

Consider first a 2-bit DAC (red points) is used to reconstruct a perfect sine wave (smooth green trace). The points identify the instants in time at which the DAC output is updated to a new value. Thus, the horizontal distance between these points is the sample period. Note the difference between the DAC output signal and the perfect sine wave. The vertical distance between the two traces at the sampling instants (black and red points) is the error introduced by the DAC as a result of its finite resolution. This error is known as the *Amplitude Quantization error* and gives rise to an effect known as *quantization distortion*.

An increase in DAC resolution (more bits) results in a decrease in quantization error. This, in turn, results in less distortion error in the reconstructed sine wave. The 3-bit DAC gives a better approximation of the perfect sine wave.



The relationship between DAC resolution and the amount of distortion is quantifiable. If the DAC is operated at its full-scale output level, then the ratio of signal power S to quantization noise power (N_{AQ}) is given by:

$$SNR_{AQ} = 6.02.q + 1.76[dB]$$

Where q is the number of bits of DAC resolution. It is supposed that the amplitude of the quantisation error is uniformly distributed.

For example, an 8-bit DAC exhibits an SNR_{AQ} of 49.92 dB. A 16-bit DAC (resolution of an audio signal on CD) has a SNR_{AQ} of 98.08 dB.

It should be noted that the SNR equation only specifies the total noise power due to amplitude quantization errors. It does not provide any information as to the frequency distribution of the spurs or the maximum spur level, only the combined power of all the spurs relative to the fundamental.

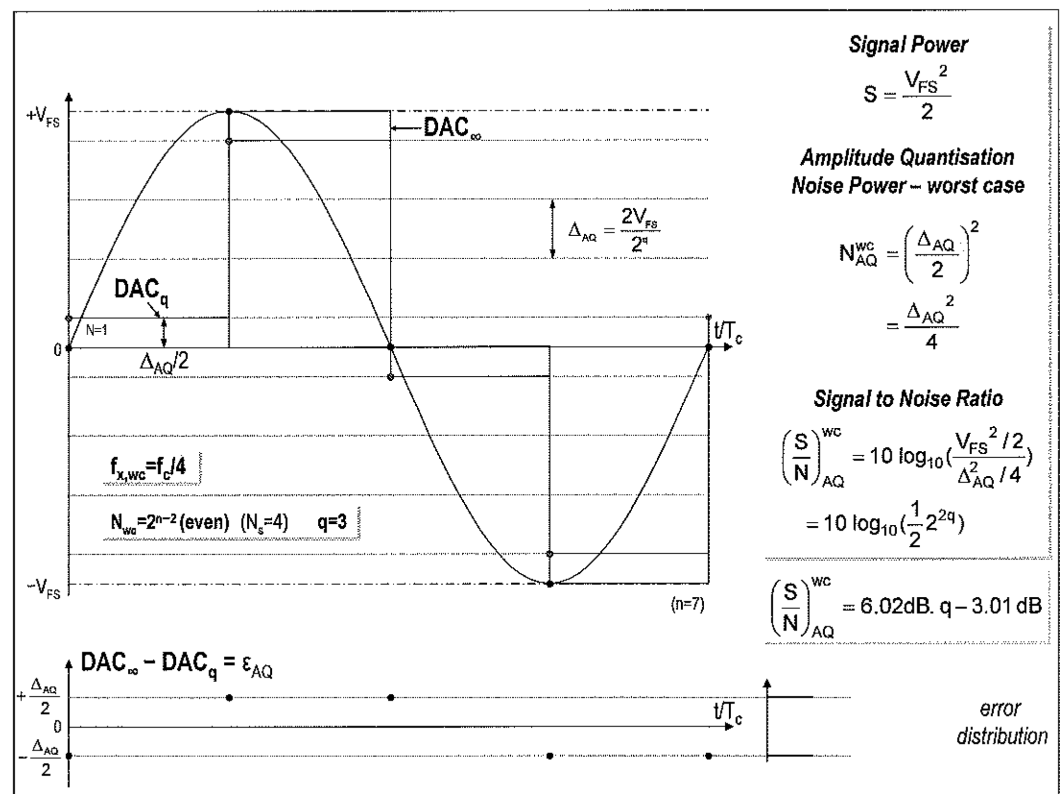
A second point to consider is that the SNR equation applies only if the DAC operates at full-scale. At output levels below full scale the power in the fundamental is reduced, but the amplitude quantization error remains constant. The net effect is a reduction in SNR; that is, the quantization noise becomes more significant relative to the fundamental. The effect of operating the DAC at less than full-scale is quantifiable and is given as:

$$F = 20\log(FFS)\ [dB]$$

where FFS is the Fraction of Full-Scale (< 1 so < 0 dB) at which the DAC operates. Thus, the SNR equation becomes:

$$SNR = 6.02B + 1.76 + F\ [dB]$$

For an 8-bit DAC operating at 70% of full-scale (F=0.7) the resulting SNR is 46.82dB (a 3.1dB reduction from the original SNR performance).

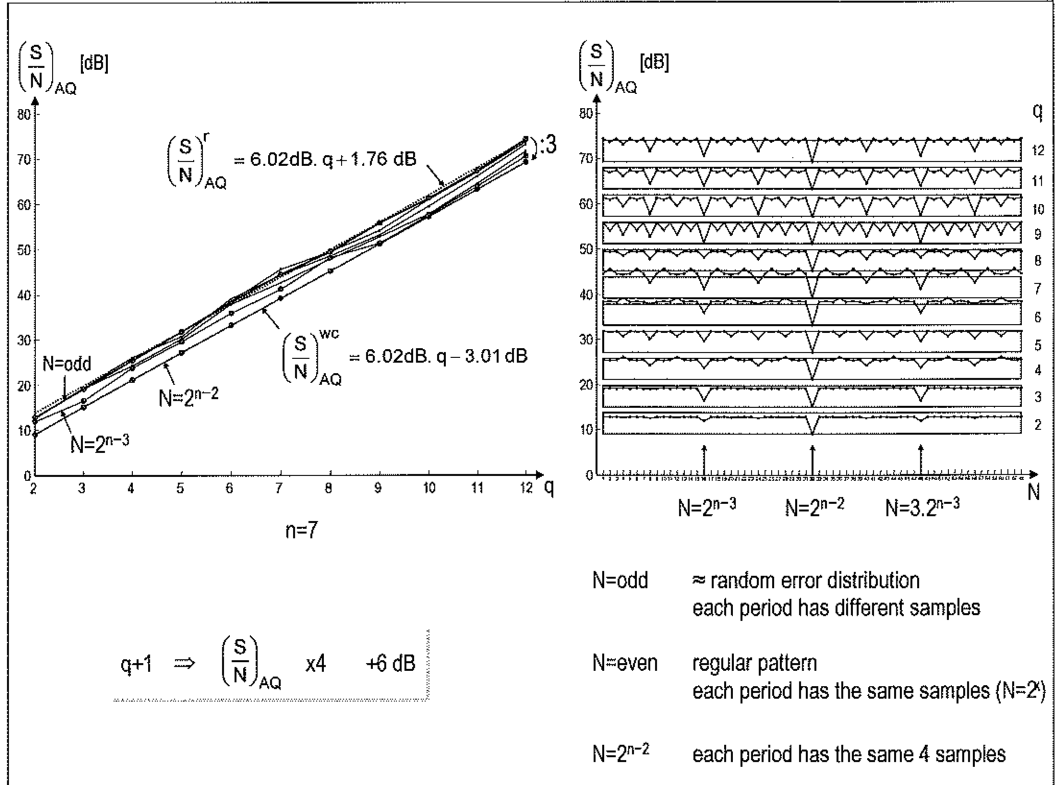


The worst case amplitude quantization occurs when f_x/f_c = ¼. Each sample has a maximum deviation of ½ LSB from the ideal sine wave. There is no uniform distribution of the amplitude of the quantization error. The frequency content of the error signal is small band (see spectrum). The period of the error signal is $1/f_x$.

A simple estimate of bound on the maximum AQ-spur to signal (carrier) ratio is:

$$SNR^{wc}_{AQ} = 6.02\,q - 3.01\ [\text{dB}]$$

This is 4.77 dB (= x 3) worse compared to a quantization error with a uniform amplitude distribution.



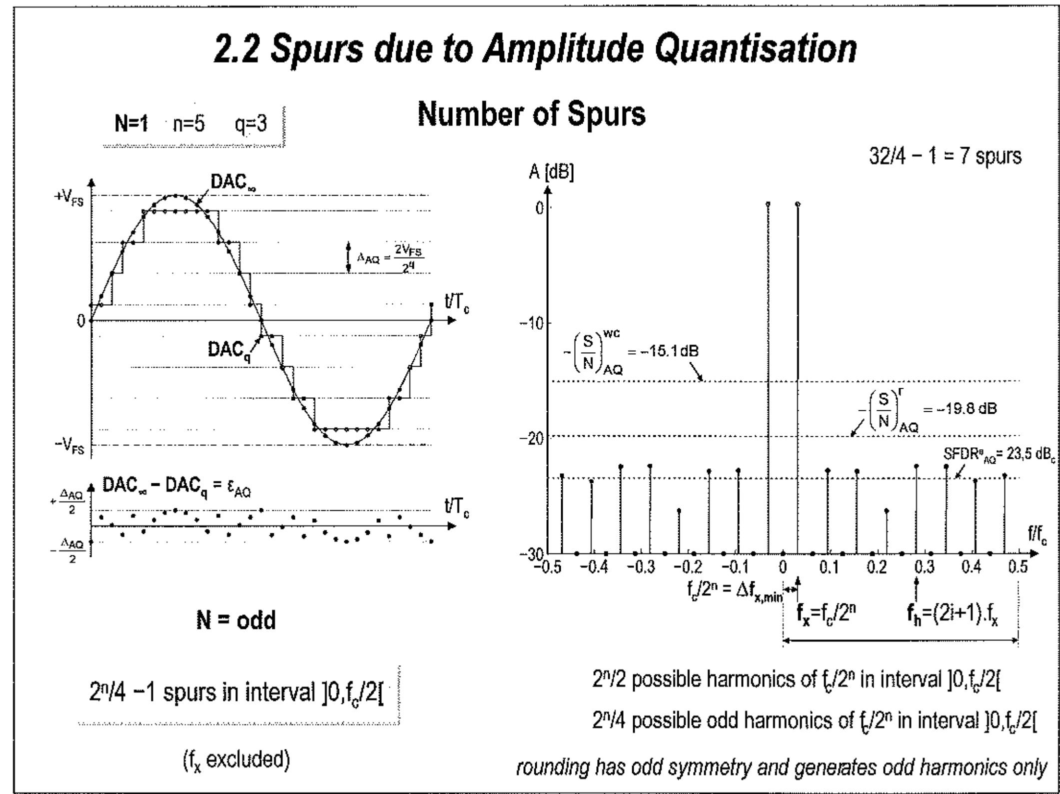
An increase in DAC resolution (more bits, q increases) results in a decrease in quantization error. This, in turn, results in less distortion error in the reconstructed sine wave. An increase of 1 bit results in an increase of 6 dB (= power x 4, voltage x 2) in SNR.

The quantization noise power also depends on the frequency tuning word N.

When the frequency tuning word N is *odd* the sampling of the sine wave is different each period (random) The SNR approximates the case of a quantization error with a uniform amplitude distribution. Also the distribution of the quantisation noise in the frequency spectrum will be quite uniform.

When the frequency tuning word N is *even* the sampling of the sine wave is the same each period (deterministic). The SNR will be less than for the case of a quantization error with a uniform distribution.

When the frequency tuning word N is 2^{n-2}($f_x/f_c = ¼$), each period contains the same 4 samples. The SNR is worst case.



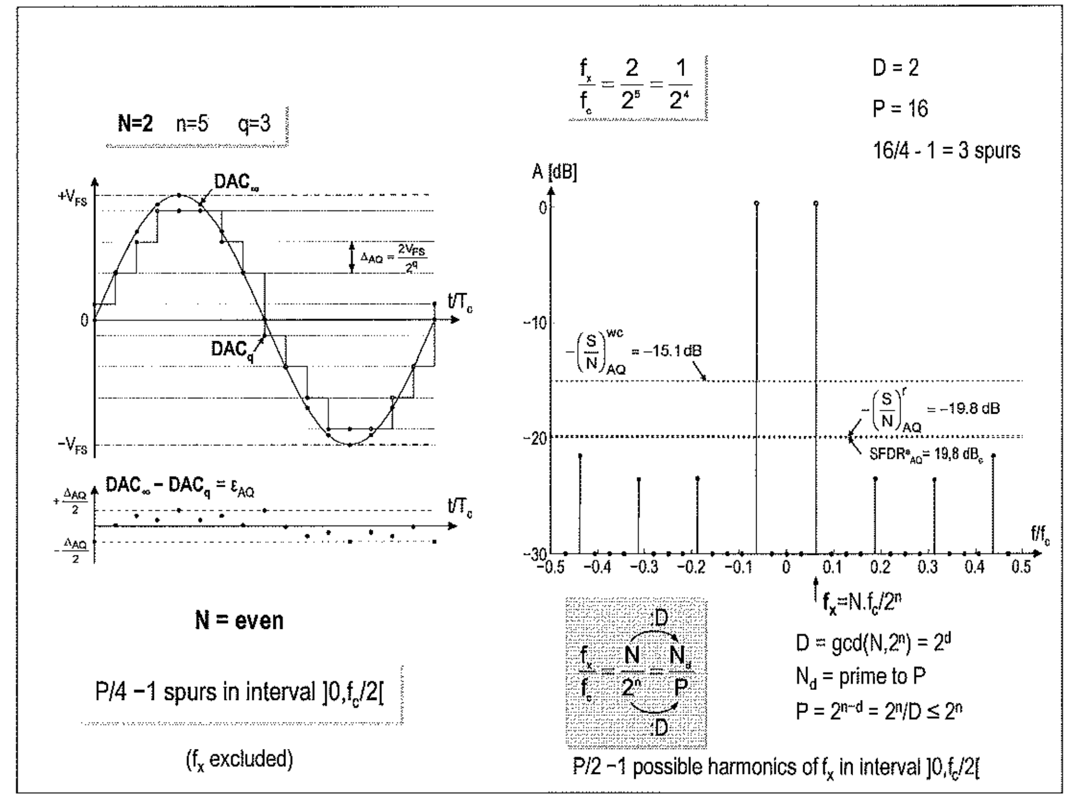
In the time domain amplitude quantization error ε_{AQ} can be seen as the difference between DAC_{∞}, the samples of perfect sine wave ($q=\infty$) and DAC_q, the output samples of the LUT represented with q bits.

The frequency content of the sampled waveforms (impulse train, before the zero-order-hold function of the DAC which results in a sin(f)/f filtering function) can be calculated with a Discrete Fourier Transformation (DFT) implemented in an FFT (Fast Fourier Transform) algorithm.

In the frequency domain, the amplitude quantization error results in extra discrete frequencies. These frequencies are aliased within the Nyquist band $[0, f_s/2]$ and appear as discrete spurs in the spectrum.

When a uniform quantizer is driven by a continuous-time sinusoid the spectrum contains one spectral line. Rounding operation, i.e. minimum distance (nearest neighbour) mapping has odd symmetry and generates *odd harmonics only*, the level of which oscillates violently and its dependence on the signal amplitude and resolution is very complicated. Moreover, because of sampling, some harmonics can fold back into the desired band and possibly overlap depending on the value of the numerical frequency. Overlapped spectral lines may contribute constructively or destructively to the level of the resulting component depending on their actual phase (and amplitude) values. Therefore the practical way to a deterministic analysis of the irregular shape of AQ-spur levels is directly given by an FFT transform of the LUT output impulse train.

As the LUT resolution increases the quantization distortion decreases; i.e., the spurious content of the LUT output spectrum decreases.



The rate at which the n-bit phase accumulator overflows is controlled by N, the frequency tuning word. Accumulator overflow (mod 2^n) corresponds to a mod 2π operation:

$$N/2^n = \Delta\theta/2\pi$$

So the output frequency (the rate of phase change) is:

$$f_x = (1/2\pi) \cdot (\Delta\theta/\Delta t) = (N/2^n).f_c$$

The numerical frequency of DDS is:

$$f_x/f_c = N/2^n = N_d/P < ½$$

with N_d prime to P, and $P = 2^n/\gcd(N, 2^n) \leq 2^n$ is the numerical period of the sample sequence.

[Note: gcd = greatest common divisor, and P is a power of 2.]

Because of odd harmonics only, the number of AQ-spurs is:

$$(P/4)-1$$

Example:

$f_x/f_c=1/16 \rightarrow P=16$

The harmonics possible in the Nyquist interval $]0,f_s/2[$:

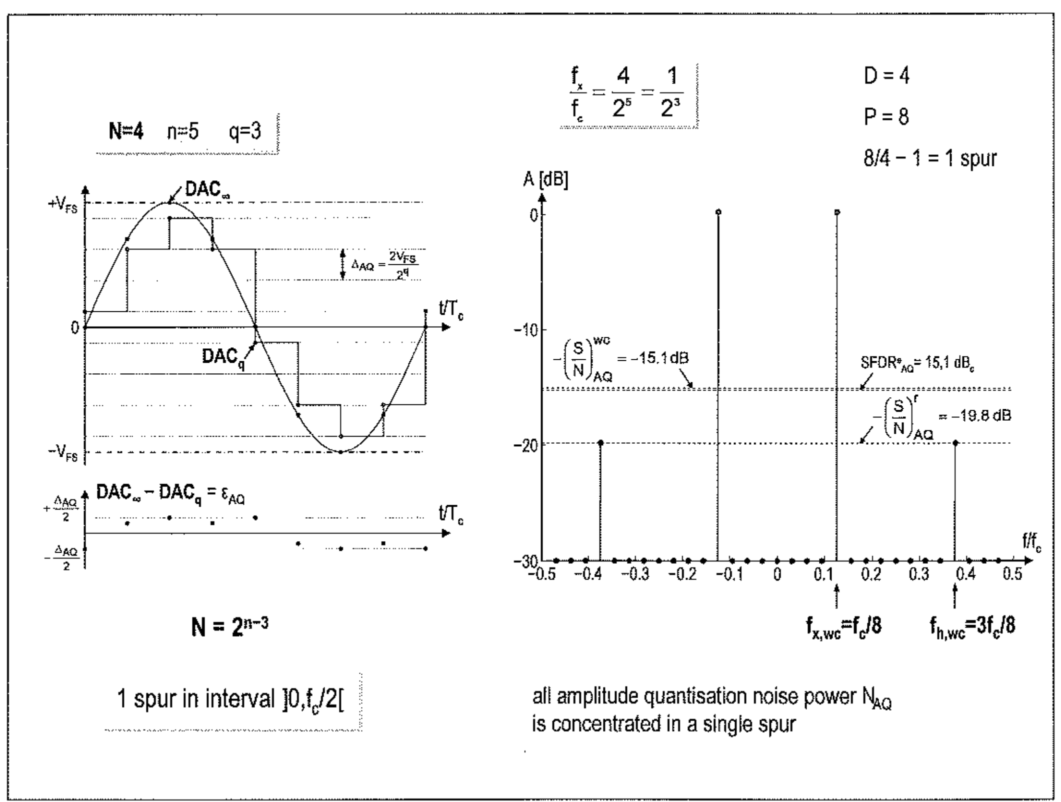
2/16 – 3/16 – 4/16 – 5/16 – 6/16 – 7/16 $\rightarrow$ 2/P – 3/P – ... – (P/2 - 1)/P

So there are P/2 – 2 harmonics possible in the Nyquist interval.

Only odd harmonics are possible:

3/16 – 5/16 – 7/16 $\rightarrow$ 3/P – 5/P – ... – (P/2 - 1)/P

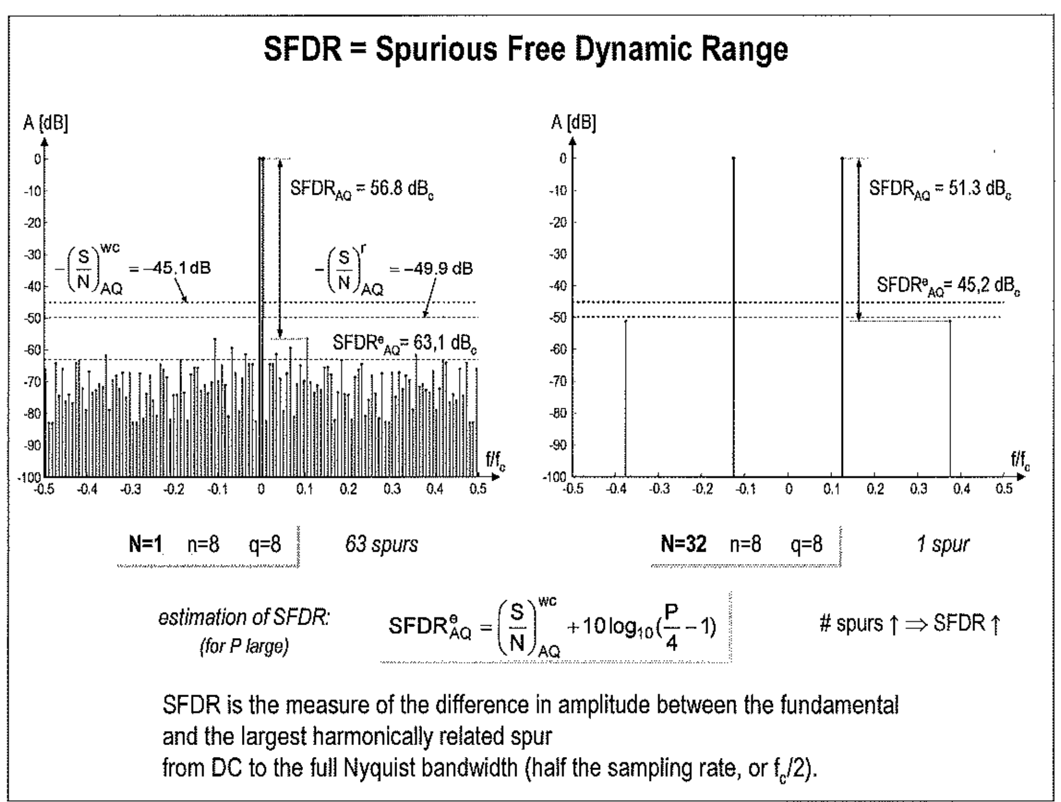
So there are (P/2 -2)/2 = P/4 -1 odd harmonics possible



The aliasing phenomenon determines the amplitudes of the AQ-spurs (overlapped harmonics).

The worst case, a highly massive overlapping occurs at P = 8 (i.e. if $f/f_c = 1/8$ or $3/8$) when the energy of the spurs apparently "concentrates" on one AQ-spur. A simple estimate of bound on the maximum AQ-spur to signal (carrier) ratio is:

$$SNR^e_{AQ} = (6.02\,q - 3.01)\ dB_c$$



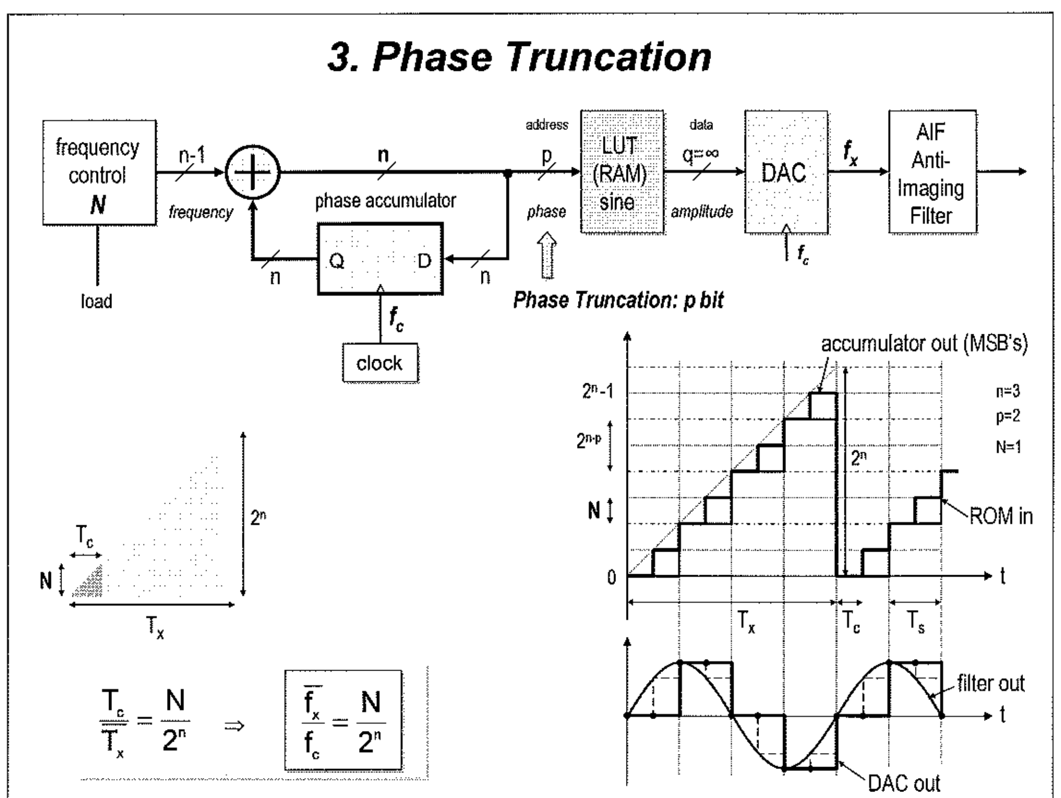
$$SFDR^{e}_{AQ} = \left(\frac{S}{N}\right)^{wc}_{AQ} + 10\log_{10}\left(\frac{P}{4} - 1\right)$$

Spurious Free Dynamic Range is the usable dynamic range of a signal before spurious noise interferes or distorts the fundamental signal. SFDR is the measure of the difference in amplitude between the fundamental and the largest harmonically or non-harmonically related spur from DC to the full Nyquist bandwidth (half the DAC sampling rate, or $f_S/2$). A spur is any frequency bin on a spectrum analyzer, or from a Fourier transform, of the analog output of the DAC. The figures are showing how SFDR is measured correctly (SFDR is usually specified in dB_c).

The worst case, a highly massive overlapping occurs at P = 8 (i.e. if f/f_c = 1/8 or 3/8) when the energy of spurs apparently "concentrates" on one AQ-spur. A simple estimate of bound on the maximum AQ-spur to signal (carrier) ratio is:

$$SFDR^{e}_{AQ} = (6.02\,q - 3.01)\ dB_c$$

The other limiting case occurs if P is long, then a "sea" of finely spaced AQ-spurs appears (or alternatively one can detect a background "noise floor").



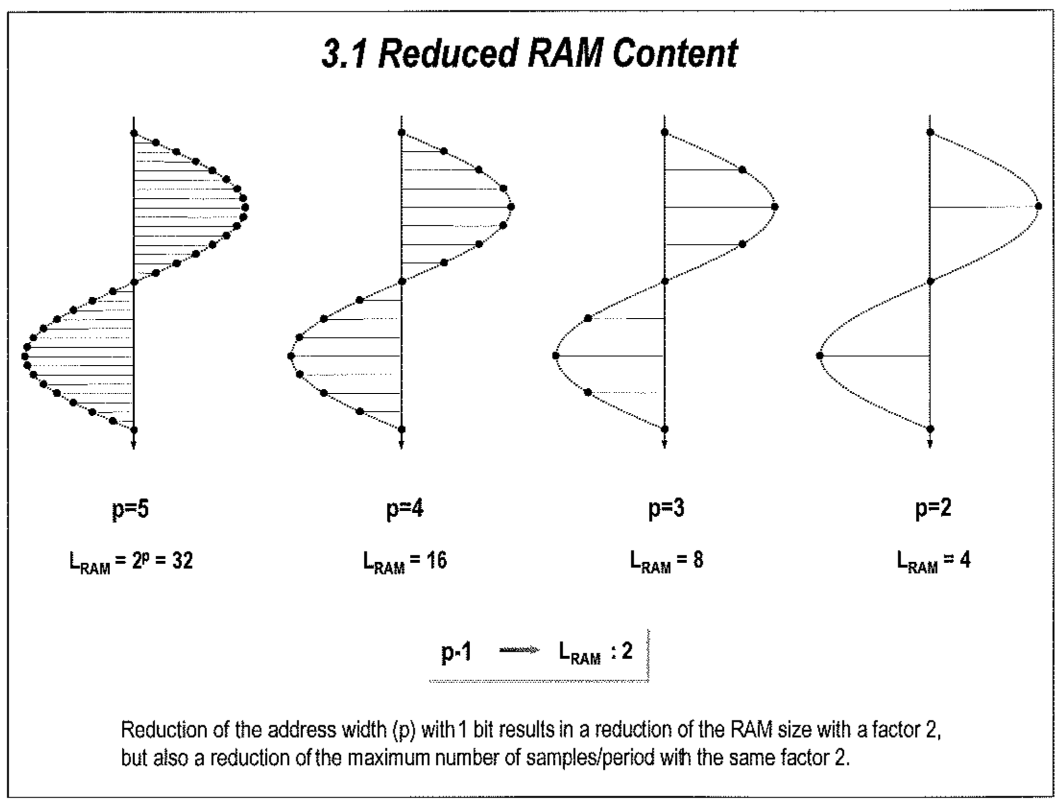
In conjunction with the system clock frequency, the phase accumulator width determines the frequency resolution of the DDS. The accumulator must have a sufficient field width to span the desired frequency resolution. For most practical applications, a large number of bits are allocated to the phase accumulator in order to satisfy the system *frequency resolution* requirements. By way of example, if the required resolution is 1 Hz, and the clock frequency is 100 MHz, the required field width of the accumulator is:

$$N = \log_2\lceil f_c/\Delta f_x \rceil = \log_2\lceil 10^8/1 \rceil = \lceil 26.5754 \rceil = 27 \text{ bits}$$

where $\lceil \; \rceil$ denotes the ceiling operator.

Due to excessive memory requirements, the full precision of the phase accumulator cannot be used to index the sine/cosine look-up table. A quantized (or truncated) version of the phase angle is used for this purpose. This is called phase angle quantization or phase truncation (PT). In FPGA, the lookup table (LUT) can be located in block or distributed memory.

Truncating an *n*-bit phase accumulator output to high *p*-bit, modifies accordingly the effect of *N* tuning word on instantaneous phase that leads to a phase truncation error (PT), but do not modify the average numerical frequency f_x. Dropping the lower (n-p) bits permits extremely fine tuning of the output frequency f_x with a reasonable LUT size (2^p addresses) at the expense of spectral purity (spurs).



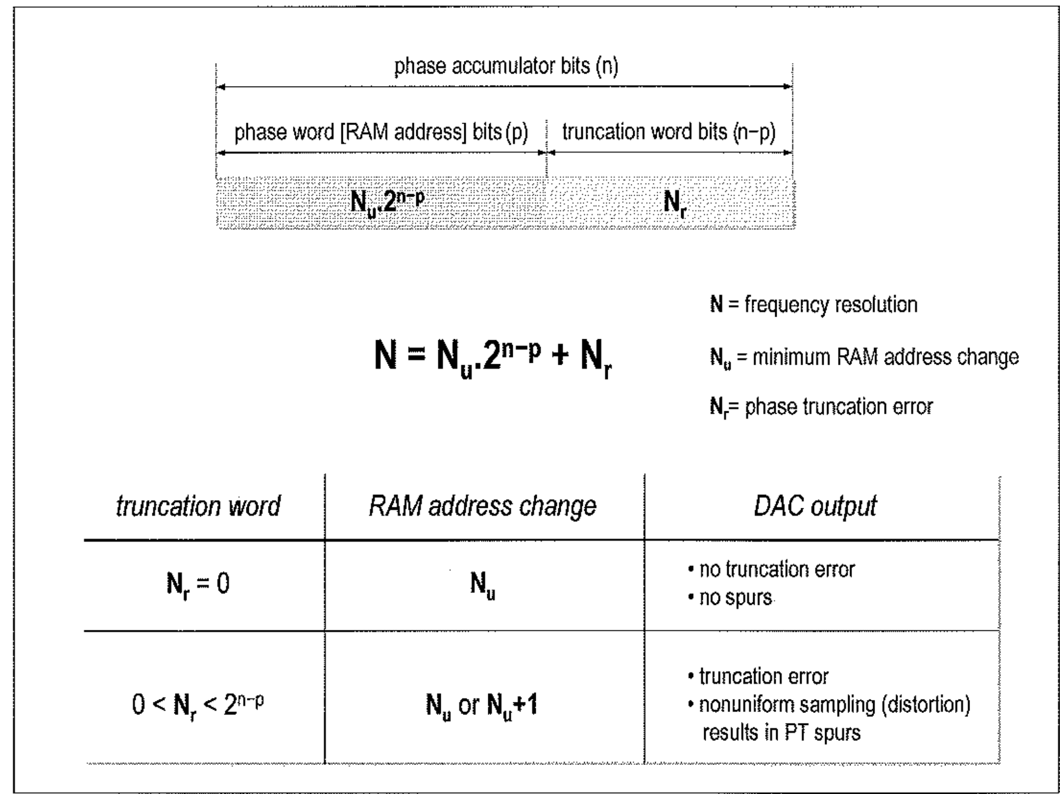
The size of the RAM (LUT) is:

$$size_{RAM} = 2^p \cdot q$$

with p = the length of the truncated phase address
q = word length of the data output

The number of words in the RAM will determine the phase quantisation error (PT) and the number of bits in each word will determine the amplitude quantization error (AQ).

For the quality of the output signal (less noise) it is desirable to increase the resolution of the RAM. But larger RAM size means higher power consumption, lower reliability, lower speed, and increased costs.



truncation word	*RAM address change*	*DAC output*
$N_r = 0$	N_u	• no truncation error • no spurs
$0 < N_r < 2^{n-p}$	N_u or N_u+1	• truncation error • nonuniform sampling (distortion) results in PT spurs

An **n**-bit word is shown, which corresponds to a *phase accumulator* with **n** bits of resolution that are to be used for frequency tuning.

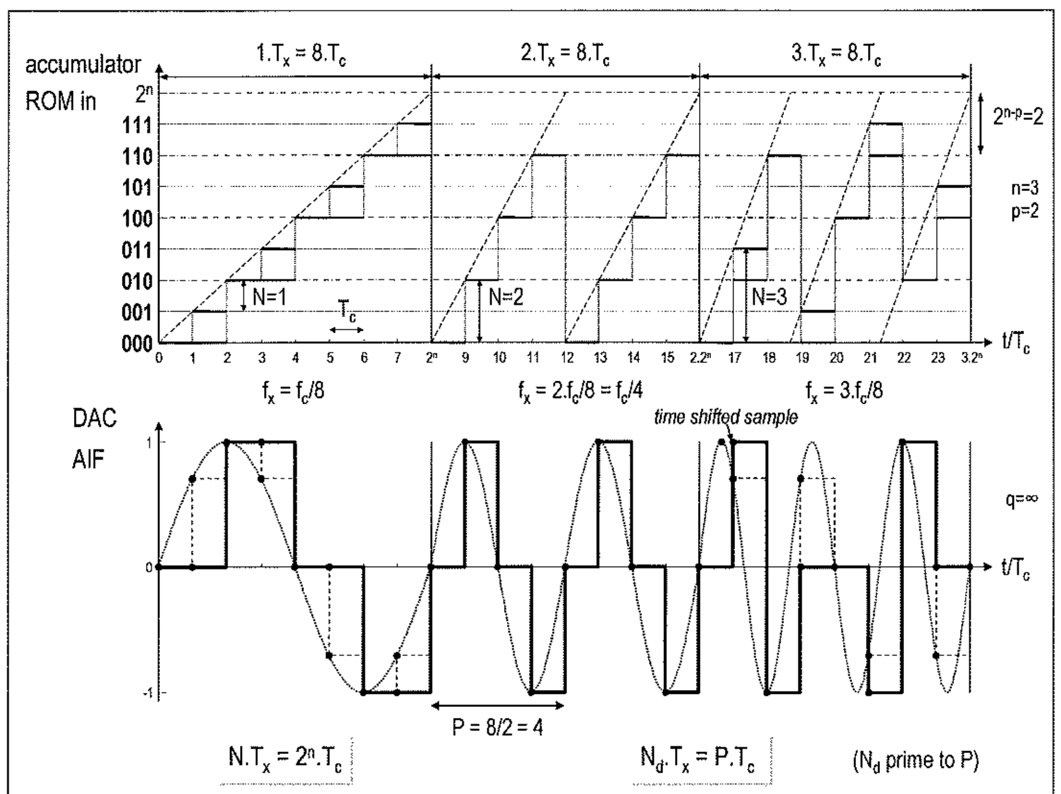
The upper **p** bits constitute the *phase word,* the bits that are to be used for conversion from phase to amplitude.

The lower **n-p** bits are the *truncation word.* These bits are truncated from the phase accumulator, that is, ignored as far as phase resolution is concerned.

Truncating an n-bit phase accumulator output to high p-bit, modifies accordingly the effect of tuning word N on instantaneous phase that leads to phase truncation (PT) - error, but do not modify the average numerical frequency. Dropping the lower bits permits extremely fine frequency tuning with reasonable RAM size at the expense of spectral purity.

When N_r is nonzero, the actual RAM address change is N_u or N_u+1, i.e. a "phase hiccup" occurs. Nonuniform sampling occurs and for this reason distortion is presented. There are Phase Truncation (PT) spurs present in the output spectrum.

If N_r is zero, then there are no PT spurs.

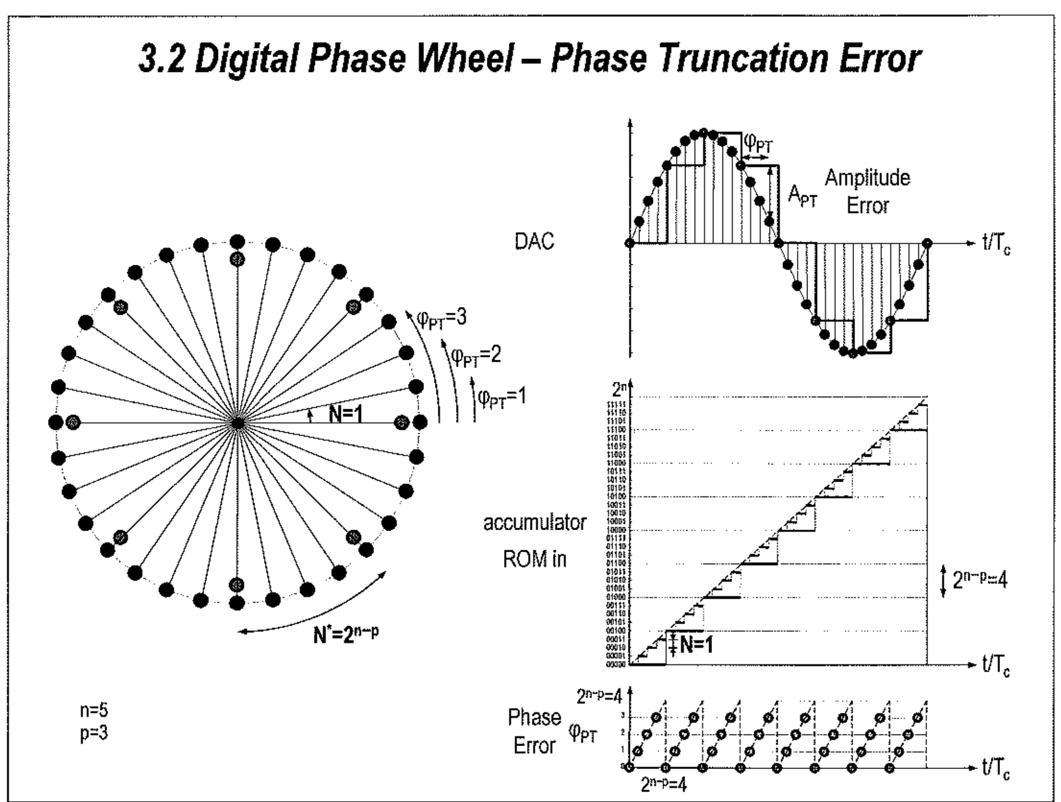


Initially, the phase accumulator contains the value of the frequency tuning word N. On each successive cycle of the DDS system clock f_c, the tuning word N is added to the previous contents of the accumulator. The accumulator is modulo 2^n, so bits that would carry beyond the MSB are simply dropped. As the accumulator sequence proceeds the value of the accumulator will eventually return to the original tuning word value and the sequence will repeat. The number of steps (or clock cycles) required to accomplish this is known as the Grand Repetition Rate (GRR):

$$GRR = P = 2^n / \gcd(N, 2^n)$$

The p-bits of the phase word are passed along to the phase-to-amplitude conversion portion of the DDS, which is used to produce the output waveform. However, the (n-p)-bits of the truncation word are not passed along to the phase-to-amplitude converter. Therefore, if the full n bits of the accumulator represent the true phase, but only p-bits of the phase word are used for determining amplitude, then the output signal is essentially in error by the value of the truncation word of (n-p)-bits. Thus, the output signal can be thought of as a composite of a full resolution signal (that which would be obtained with no phase truncation) and an error signal due to the (n-p)-bits of the truncation word.

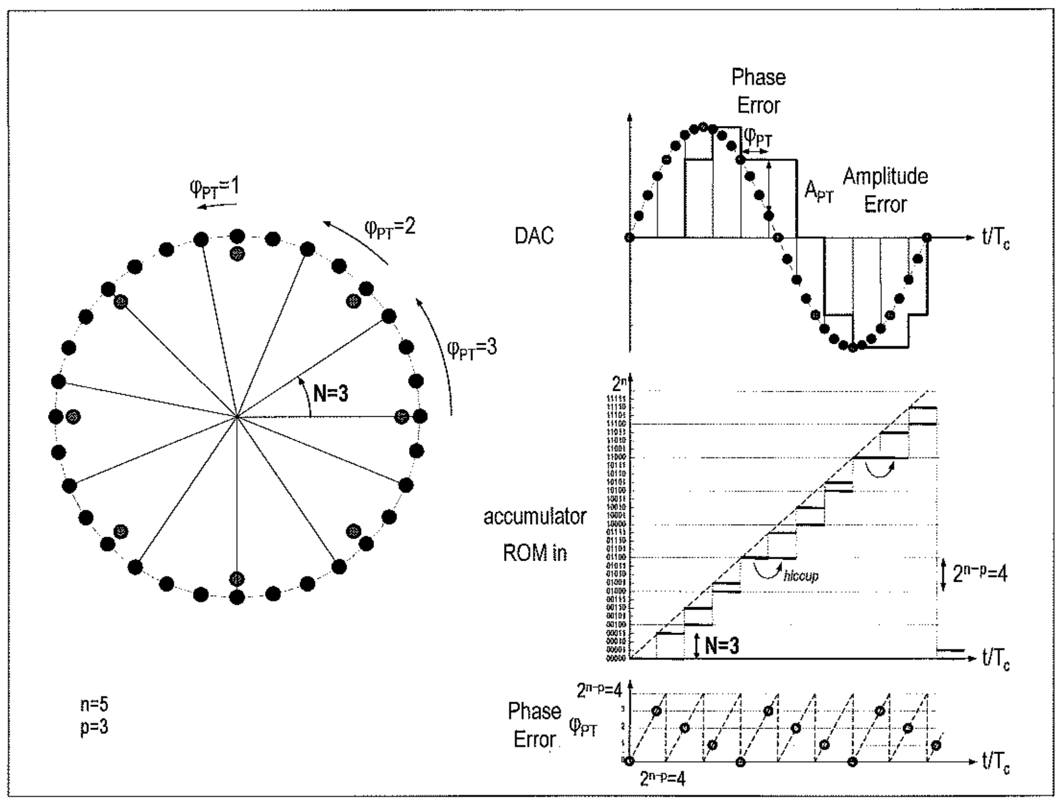
The error signal, then, is a source of spurious noise. Since the error signal is defined by the truncation word, then analysis of the behaviour of the truncation word should allow some insight into the nature of the phase truncation error signal.



The truncation word is the difference between the exact phase (n-bit accumulator output) and the truncated phase (p MSBits of the accumulator output). This value of this difference is the phase error φ_{PT}. The truncation word accumulates up to a maximum value of 2^{n-p}. It has the shape of a sawtooth waveform with a period:

$$P_{PT} = 2^{n-p} / \gcd(N_r, 2^{n-p})$$

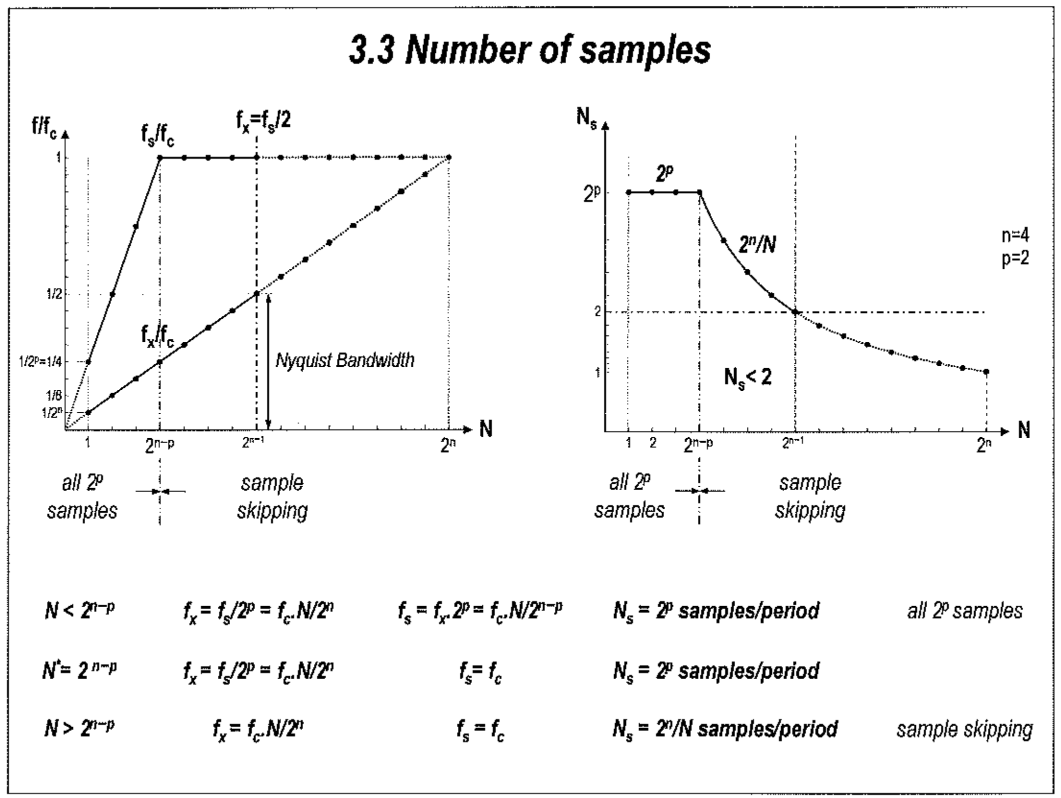
It should be apparent that the sawtooth shape results from the overflow characteristic of the accumulator. Also note that the complete sequence of truncation word values repeats after a period of P clock cycles. Since the behaviour of the truncation word is periodic in the time domain, then its Fourier Transform is periodic in the frequency domain.



Quantizing the phase accumulator introduces time base jitter in the output waveform. This jitter results in undesired phase modulation that is proportional to the quantization error.

When the truncation word N_r is nonzero, the actual RAM address change is N_u or N_u+1, i.e. a "phase hiccup" occurs. Non-uniform sampling occurs and for this reason distortion is presented. There are Phase Truncation spurs present in the output spectrum.

The phase error φ_{PT} results in an amplitude error A_{PT} in the output signal.



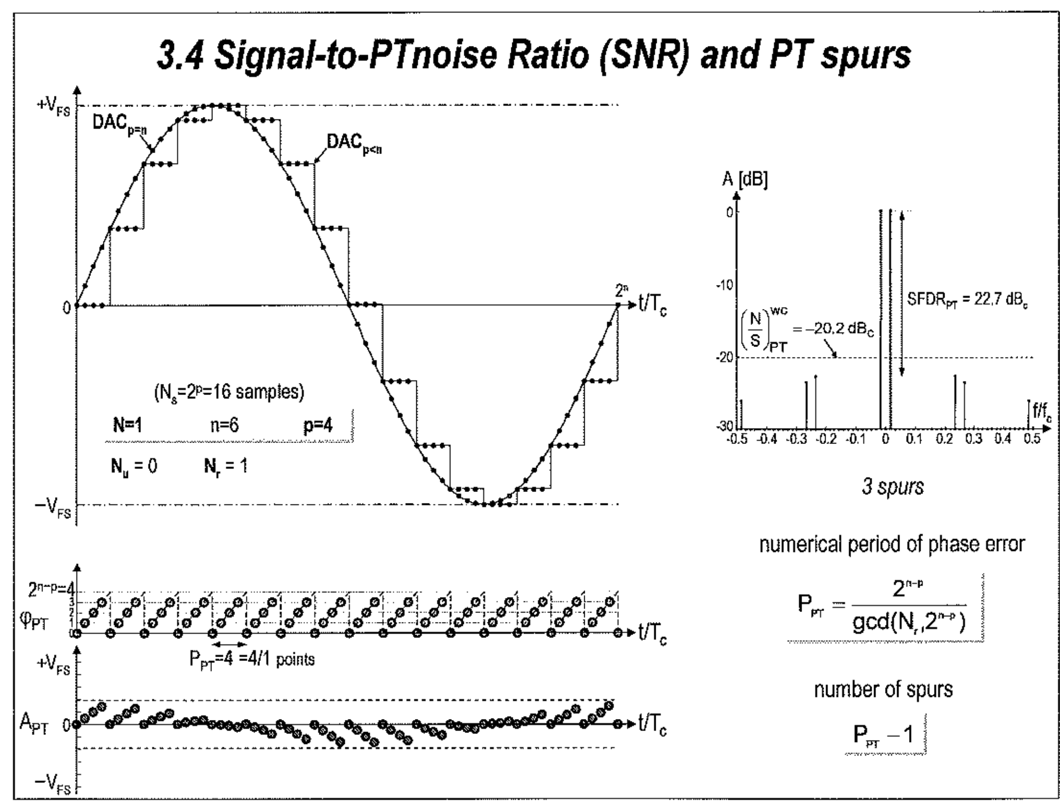
$N \leq 2^{n-p}$

The frequency tuning word N is smaller or equal to $2^{n-p} = N_{r,max} + 1$ ($N_{r,max}$ = the maximum value of the truncation word). The p most significant bits of the accumulator output (= address bits of the LUT) can not increase with more than 1 LSB. All 2^p samples stored in the LUT will be used sequentially. The number of samples used does not dependent on the value of N in this case.

$N > 2^{n-p}$

The frequency tuning word N is larger than 2^{n-p}. Now the p most significant bits of the accumulator output (= address bits of the LUT) will increase with a minimum of 1 LSB. During one signal period at least one time the accumulator output will increase with at least 2 LSB's. Some of the samples stored in the LUT will now be skipped as was the case in the original version of the DDS without phase truncation. As the output frequency is increased, the number of output samples per waveform cycle will decrease. In fact, different groups of points may be output on successive waveform cycles.

DDS has the benefit to deliver a linear tuning of the output frequency.

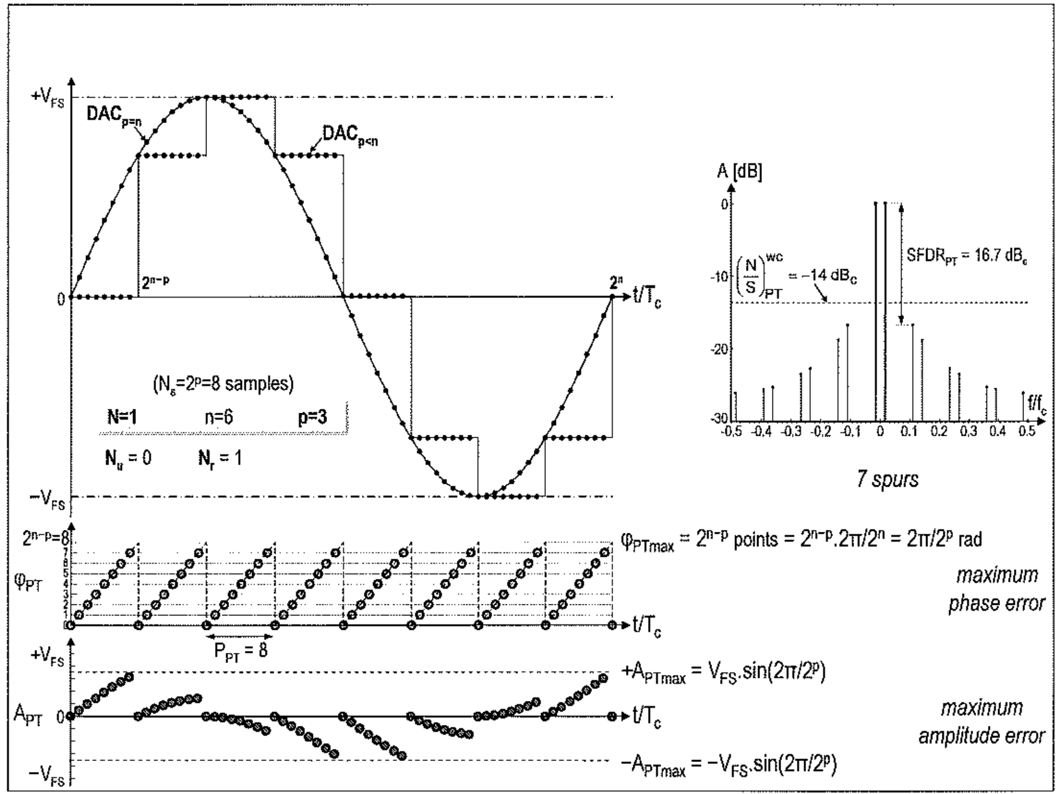


The value of the truncation word is the phase error φ_{PT}. It accumulates up to a maximum value of 2^{n-p}. It has the shape of a sawtooth waveform with a period:

$$P_{PT} = 2^{n-p} / \gcd(N_r, 2^{n-p})$$

The phase error φ_{PT} results in an amplitude error A_{PT} in the output signal. Consequently, the LUT output will be composed of the desired clean sinewave corrupted by the phase truncation. This results in the frequency domain in cosine modulated harmonics of the sawtooth waveform, the PT-spurs.

From P_{PT}, the periodicity of the phase error sequence, it follows that the number of PT-spurs is P_{PT} -1.



The maximum value of the phase error φ_{PT} approximates 2^{n-p}. This corresponds with a phase angle:

$$\varphi_{PT,max} = 2\pi/2^p$$

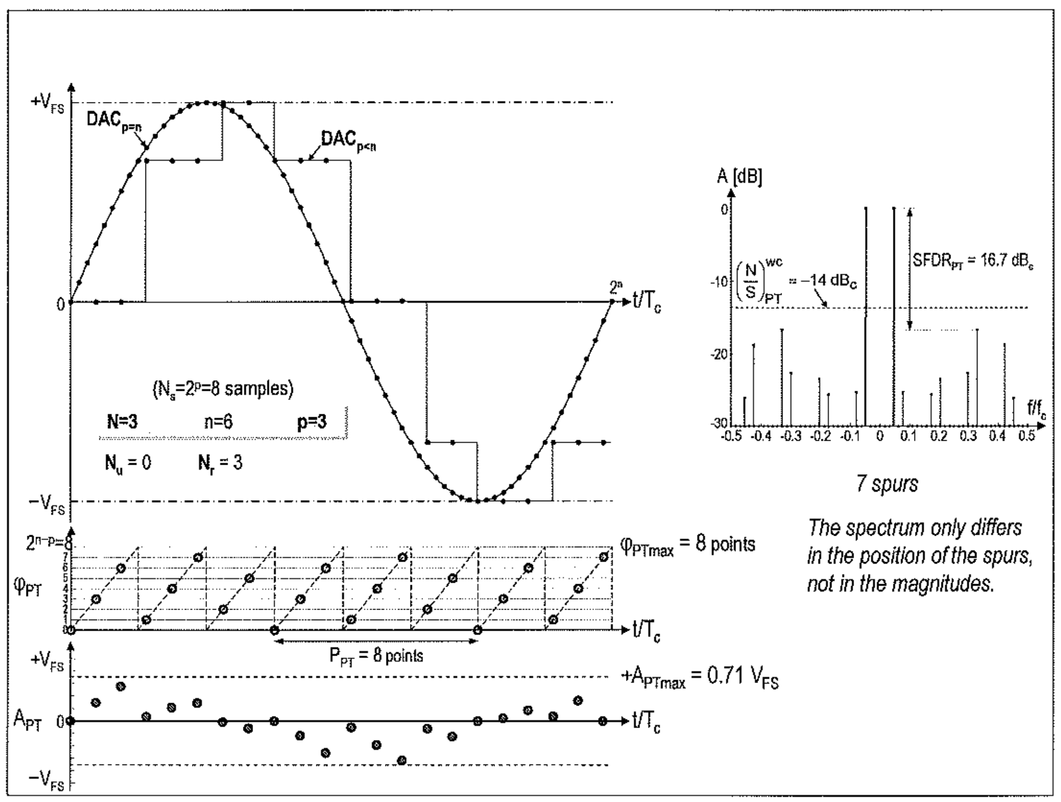
This results in a maximum amplitude error due to phase truncation at the DAC output of:

$$A_{PT,max} = V_{FS}.\sin(\varphi_{PT,max}) = V_{FS}.\sin(2\pi/2^p)$$

For large values of p this can be approximated by:

$$A_{PT,max} \approx V_{FS}.(2\pi/2^p) = 2\pi.V_{FS}./2^p$$

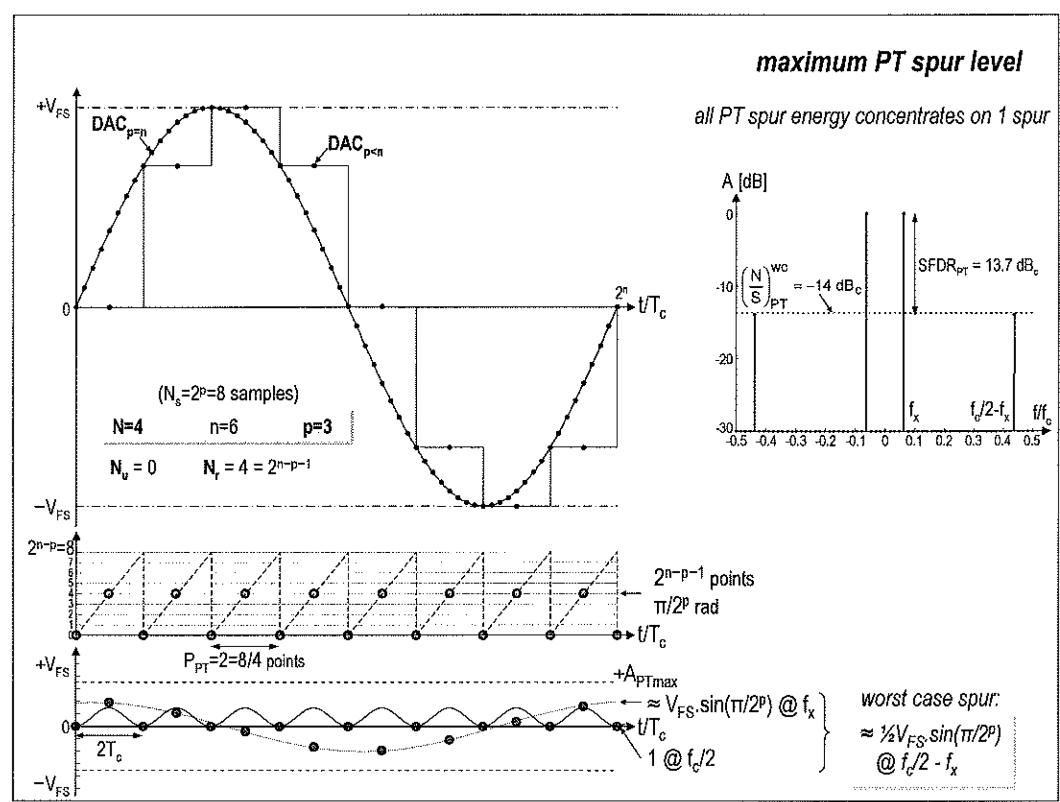
Like the amplitude quantisation error, the phase truncation error can be reduced by a factor of 2 by increasing the number op p-bits by 1.



Varying N will change the signal frequency f_x.

For a given P_{PT}, the shape of the sawtooth in the phase error sequence thus not change. Also the harmonics of the sawtooth in a continuous system would not change. In a sampled system, the harmonics above the Nyquist frequency $f_s/2$, will be remapped into the Nyquist band [0, $f_s/2$]. Note that aliasing causes spurs in frequency bands that start at odd integer multiples of $f_s/2$ are map directly into the Nyquist band . Spurs that occur in frequency bands that start at even multiples of $f_s/2$ map as mirror images into the Nyquist band. This is the nature of the aliasing phenomenon.

For a given P_{PT}, varying N will just permute the same spectral lines (only the relative location of the signal and the PT-spurs are varying, but not the levels).



The worst case SFDR occurs when $\gcd(N, 2^{n-p}) = 2^{n-p-1}$. Spur energy apparently concentrates on one PT-spur.

The phase truncation error φ_{PT} (truncation word value) alternates between 0 and 2^{n-p-1}. This corresponds with a phase error expressed in radians of:

$$\varphi_{PT} = 2\pi / 2^{n-p-1} = \pi/2^p \text{ rad}$$

This results in a maximum amplitude error A_{PT} at the sampling points of the LUT output of:

$$A_{PT,sampling,max} = V_{FS}.\sin(\pi/2^p)$$

This is approximately 2 times smaller than the maximum amplitude error due to phase truncation at the DAC output:

$$A_{PT,max} = V_{FS}.\sin(\varphi_{PT,max}) = V_{FS}.\sin(2\pi/2^p)$$

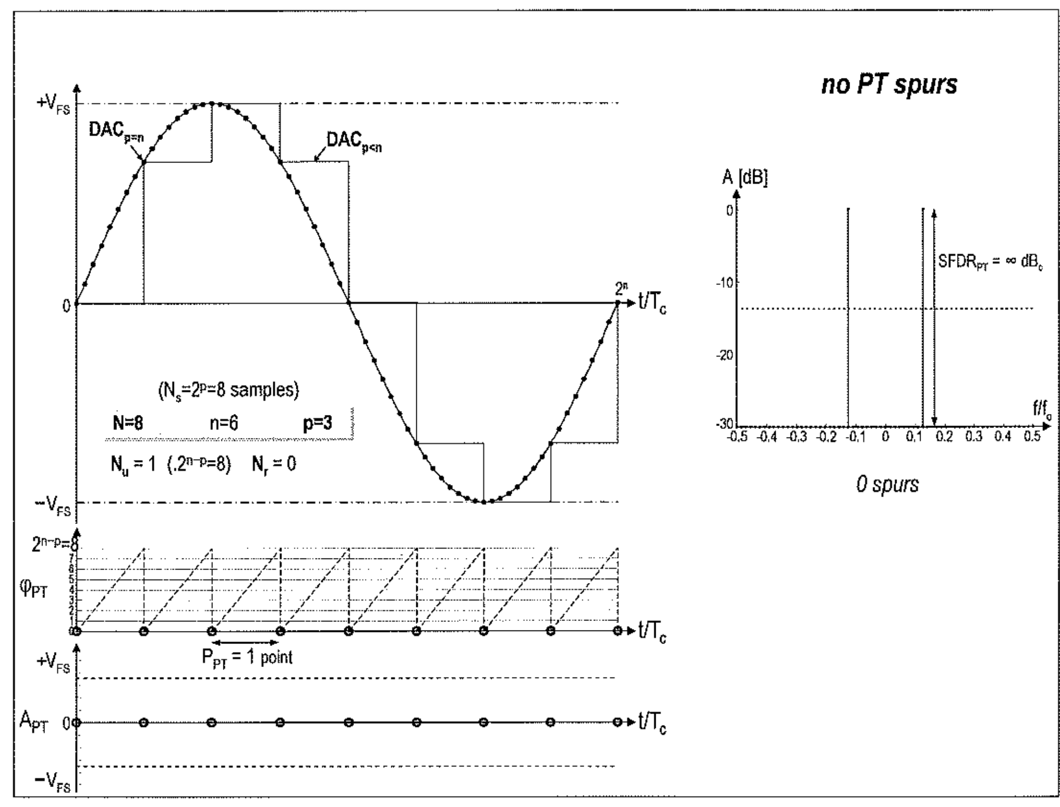
The waveform of A_{PT} can be approximated by the following equation:

$$A_{PT}(t) \approx V_{FS}.\sin(\pi/2^p).\sin(2\pi f_x t).\sin(2\pi f_c t/2) = ½.V_{FS}.\sin(\pi/2^p).[\cos(2\pi(f_c/2-f_x)t) - \cos(2\pi(f_c/2+f_x)t)]$$

There exists a single spur in the Nyquist band:

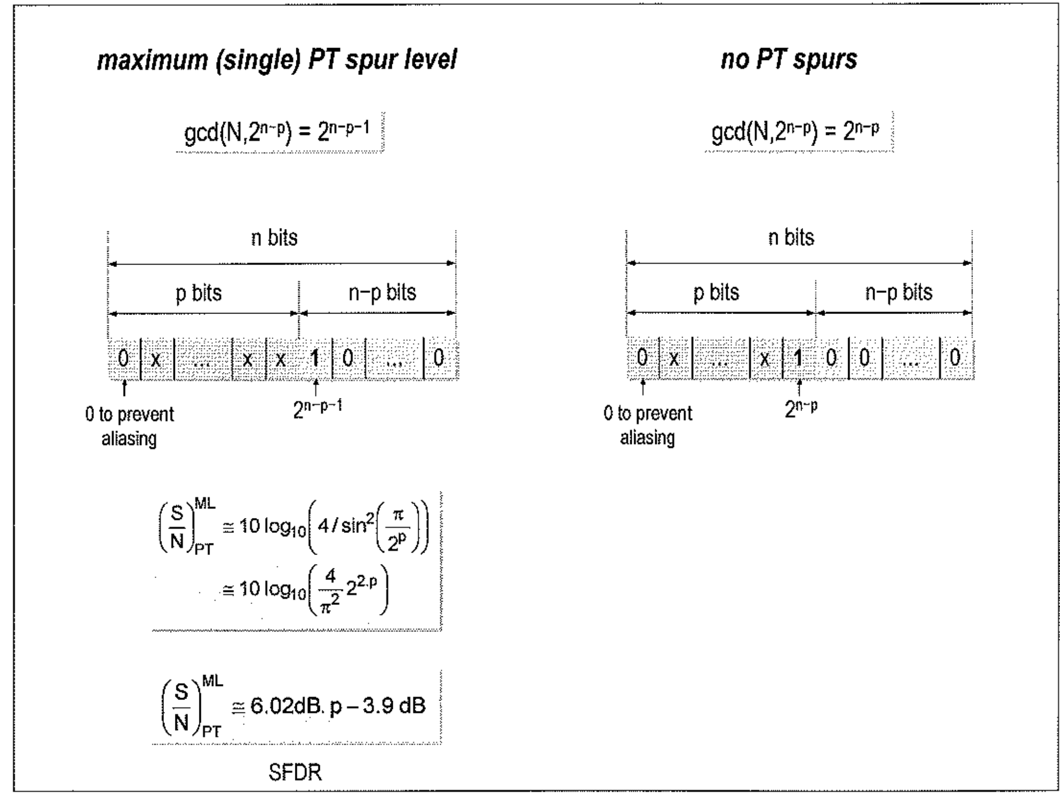
$$½.V_{FS}.\sin(\pi/2^p) \text{ @ } f_c/2-f_x \quad \rightarrow \quad SFDR^{WC}_{PT} = [6.02\,p - 3.9] \text{ dB}$$

Remark: Since $A_{PT,max} \approx 2.A_{PT,sampling,max}$, the SNR at the DAC output is even lower.



There are no PT spurs if $gcd(N, 2^{n-p}) = 2^{n-p}$. With each clock cycle, the LUT address increases with the same integer. The sine wave is sampled exactly. The phase error φ_{PT} at the sample instants is zero. Also the corresponding amplitude error A_{PT} at the sampling instants of the LUT output are zero. At the DAC output there still exists a reconstruction error due to the zero-order-hold (ZOH) interpolation function of the DAC.

For $N = 2^{n-p}$, all 2^p samples stored in the LUT are used sequentially and at the exact time (no time jitter). With each clock cycle, the LUT address increases with 1.



Tuning words that yield the maximum spur level are those that satisfy the following equation:

$$gcd(N, 2^{n-p}) = 2^{n-p-1}$$

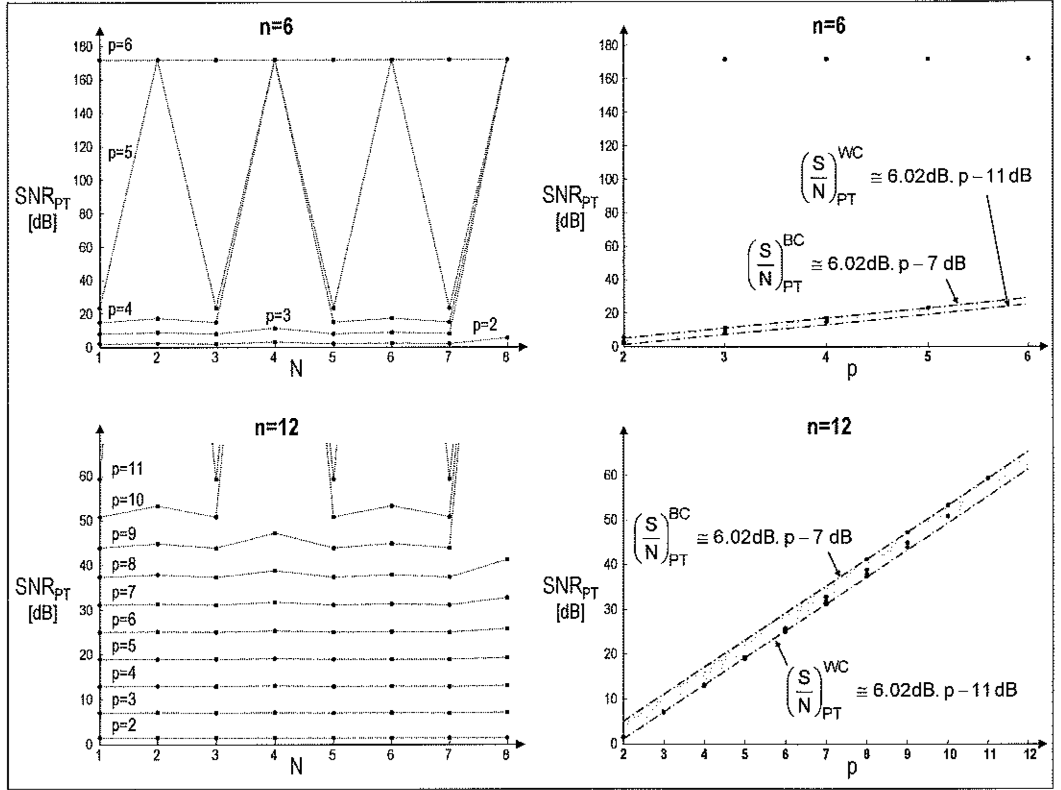
where gcd(X, Y) is the greatest common divisor of both X and Y. In order for this equation to be true, a tuning word bit pattern for the tuning word must be as shown in the figure at the left. An n-bit word is shown, which corresponds to a phase accumulator with n bits of resolution. The upper p bits constitute the phase word (the bits that are to be used for conversion from phase to amplitude). The lower n-p bits are truncated, that is, ignored as far as phase resolution is concerned. The tuning word, N, is made up of the n-1 least significant bits (the most significant bit of the tuning word must be a 0 to avoid the problem of aliasing). As shown in the figure, any tuning word with a 1 in bit position 2^{n-p-1} and 0's in all less significant bit positions will yield the worst case phase truncation spur level.

At the other extreme are tuning words that yield no phase truncation spurs. Such tuning words must satisfy the equation:

$$gcd(N, 2^{n-p}) = 2^{n-p}$$

In order for this equation to be true, the tuning word bit pattern must be as shown in the figure at the right. Thus, tuning words that yield no phase truncation spurs are characterized by a 1 in bit position 2^{n-p} and 0's in all less significant bit positions.

All other tuning word patterns that do not fit the two categories above will yield phase truncation spur levels between the two extremes.



The spurious signals caused by the phase truncation can be very high. To lower them several parameters and techniques can be used.

p = address width

A simple estimate of bound on the SNR, based on simulations:

best case: $SNR^{BC}_{PT} = [6.02\,p - 7]$ dB

worst case: $SNR^{WC}_{PT} = [6.02\,p - 11]$ dB

The worst case situation occurs for large truncation errors (n and n-p large).

An increase in address resolution (more bits, p increases) results in a decrease in phase truncation error. This, in turn, results in less distortion error in the reconstructed sine wave. An increase of 1 bit results in an increase of 6 dB (= x2) in SNR.

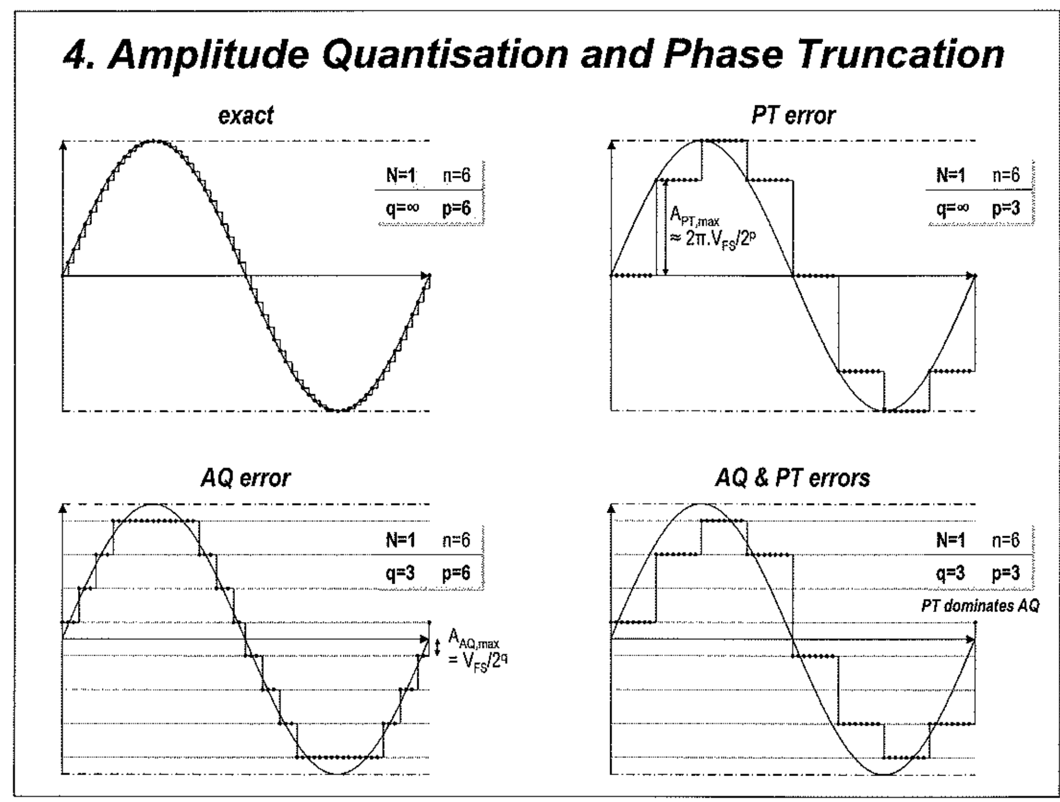
N = frequency tuning word

By keeping N even, the SNR can be improved (for p large).

For N odd, the SFDR can be improved (noise is more random).

noise reduction technique

The spur level can be further reduced by phase dithering.



The fidelity of a signal formed by recalling samples of a sinusoid from a look-up table is affected by both the phase and amplitude quantization of the process. The length and width of the look-up table affect the signal's phase angle resolution and the signal's amplitude resolution respectively.

The maximum phase truncation error can be approximated for large values of p by:

$$A_{PT,max} \approx V_{FS}.(2\pi/2^p) = 2\pi.V_{FS}/2^p$$

The maximum amplitude quantisation error is:

$$A_{AQ,max} = V_{FS}/2^q$$

The ratio between the maximum phase truncation and amplitude quantisation error is:

$$A_{PT,max} / A_{AQ,max} = 2\pi.2^q/2^p$$

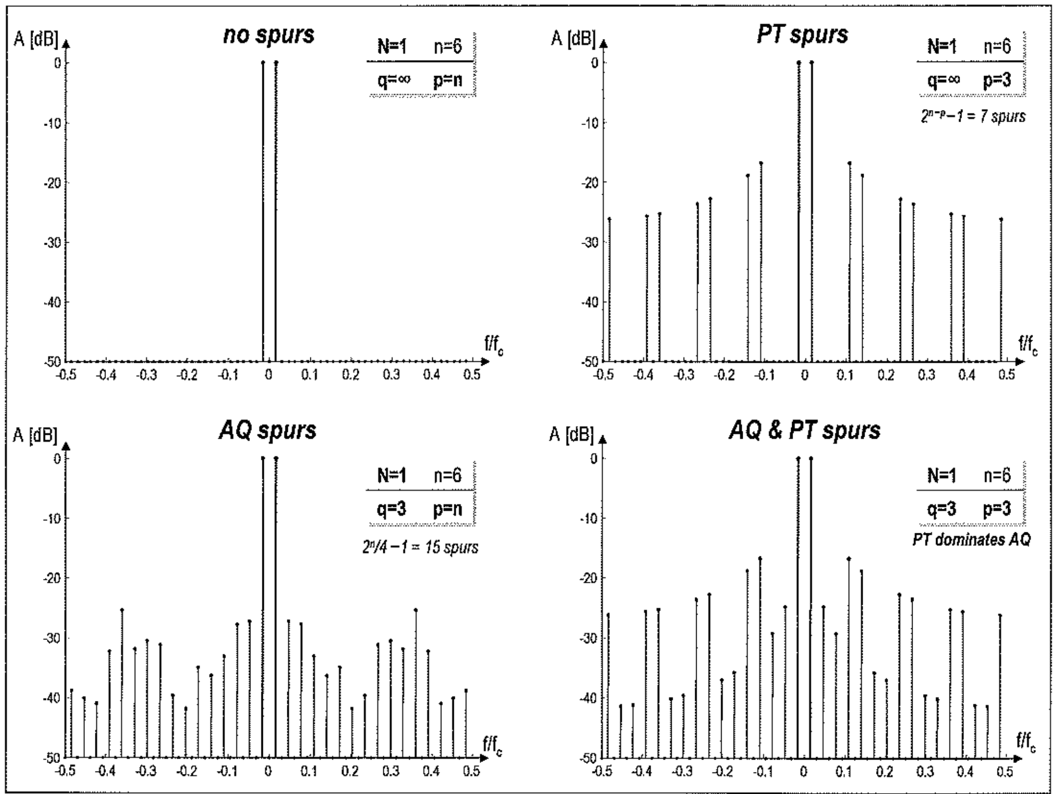
For the same number of bits p=q, the ratio becomes:

$$A_{PT,max} / A_{AQ,max} = 2\pi = 16 \text{ dB}$$

To reduce the influence of the phase truncation error below the level of the amplitude quantisation error, the following equation must hold:

$$p \geq q + 3$$

As a general rule, the LUT p-bit address length is 2 or 3 bits higher (12 dB or 18 dB) than the q-bit data precision to make the PT-error level lower than the AQ-error level.



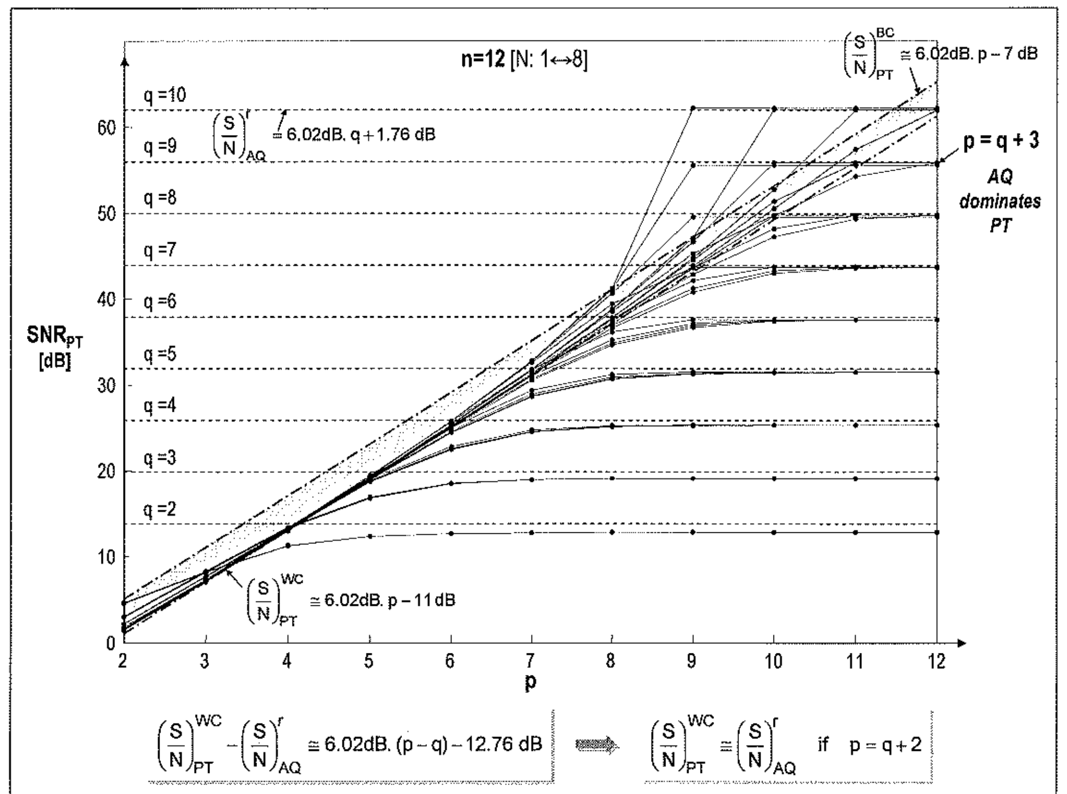
The signal's phase angle resolution and the signal's amplitude resolution limits are equivalent to time base jitter and to amplitude quantization of the signal and add spectral modulation lines and a white broad-band noise floor to the signal's spectrum.

A q-bit *amplitude quantization (AQ)* is a strong deterministic nonlinearity. The rounding operation, i.e. minimum distance (nearest neighbour) mapping has odd symmetry and generates odd harmonics only, the level of which oscillates violently. Aliasing phenomenon exposures the locations of AQ-spur (overlapped harmonics). Lines in the odd Nyquist zones map directly into the baseband (1st Nyquist zone, single sided), while components in the even zones map in a mirrored fashion with phase inversion.

Truncating an n-bit phase accumulator output to high p-bit, modifies accordingly the effect of the tuning word N on instantaneous phase that leads to *phase truncation (PT)* errors. The PT-spurs are cosine modulated harmonics of the sawtooth waveform. Varying N will just *permute* the same spectral lines (only relative location of the signal and PT-spurs varies but not the levels).

For p=q, the PT-spurs dominate the frequency spectrum and determine the SFDR.

As a general rule, the LUT p-bit address length is 2 or 3 bits higher (12 dB or 18 dB) than the q-bit data precision to make the highest PT-spur component smaller than the background AQ "noise floor".



The Signal-to-Noise ratios (SNR) due to quantisation noise and phase truncation noise are:

AQ:

Random signal:	$SNR^{r}_{AQ} = (6.02\,q + 1.76)$ dB
Worst Case:	$SNR^{WC}_{AQ} = (6.02\,q - 3.01)$ dB

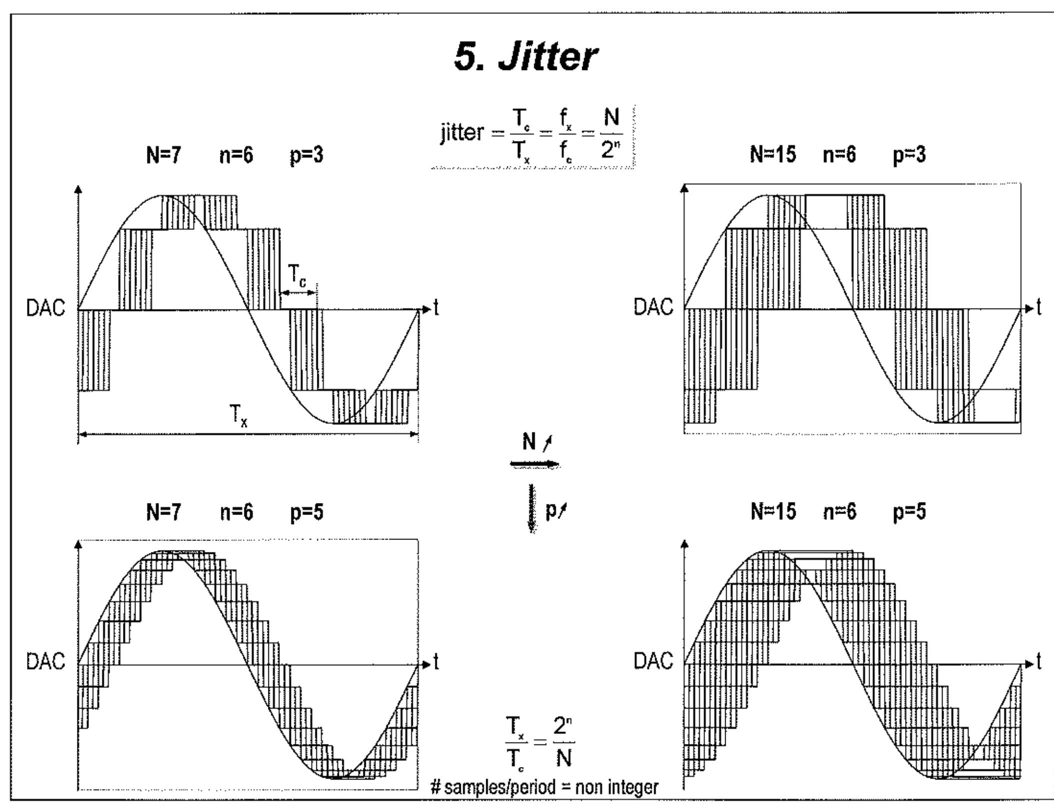
PT:

Best Case:	$SNR^{BC}_{PT} = (6.02\,p - 7)$ dB
Worst Case:	$SNR^{WC}_{PT} = (6.02\,p - 11)$ dB

The worst case difference between the SNR due to PT-errors and AQ-errors is:

$$SNR^{WC}_{PT} - SNR^{r}_{AQ} = [\,6.02\,(p - q) - 12.76]\ \text{dB}$$

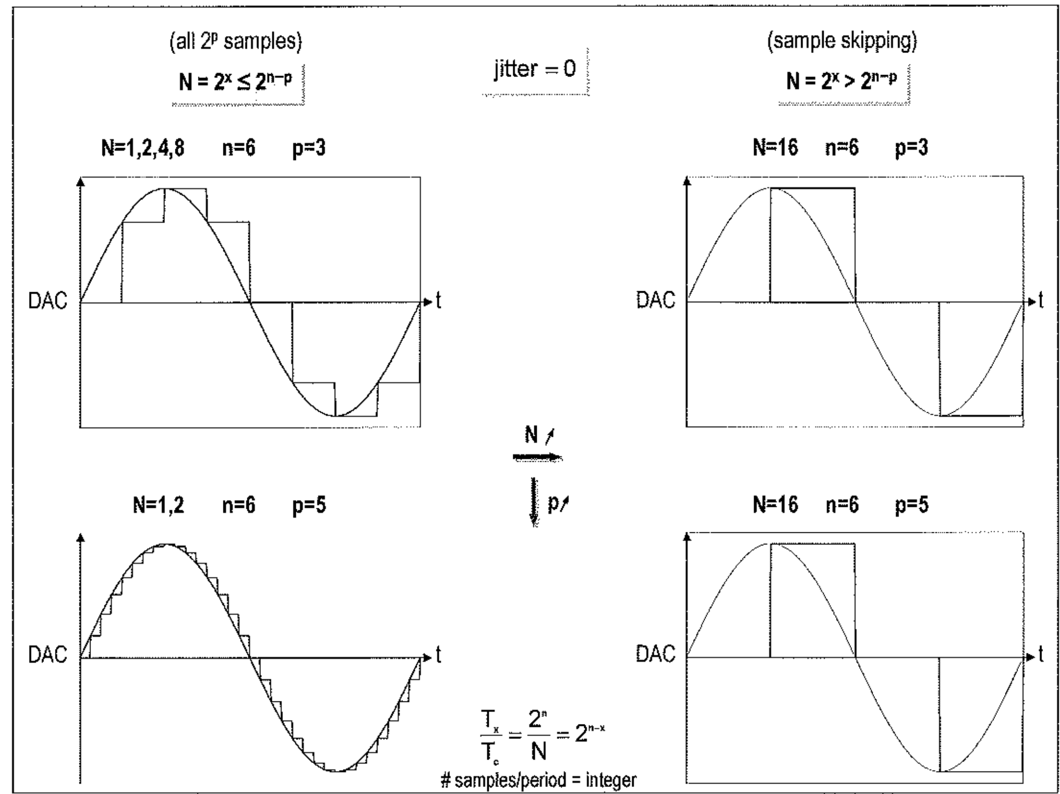
This means that p must be 2 bits larger than q ($p = q + 2$) to make the contribution of both error sources equal.



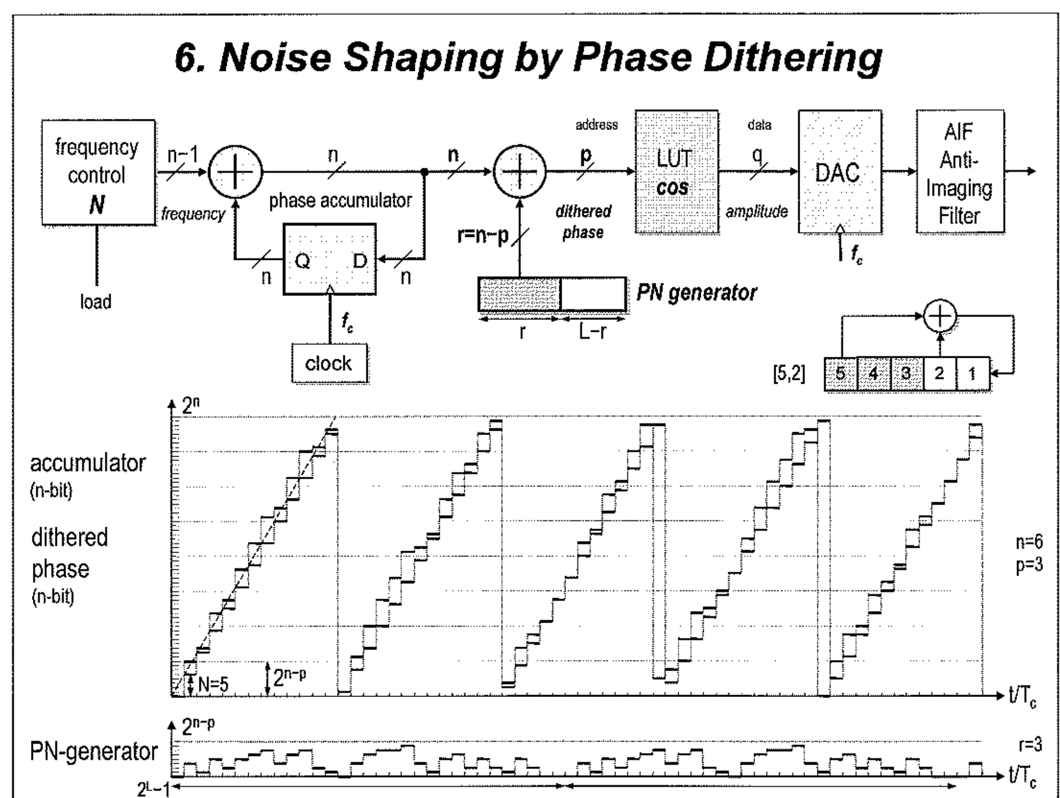
Quantizing the phase accumulator introduces time base jitter in the output waveform. This jitter results in undesired phase modulation that is proportional to the quantization error.

The DAC output of the DDS system is a sampled sine wave containing many extraneous frequency components that will create jitter if used "as is". The amount of jitter resulting from an unfiltered sampled sinewave is equal to 1 input clock cycle T_c. One clock cycle jitter will be observed from an accumulation of adjacent cycles of the DAC output signal. The cycle-to-cycle samples are different as is evident by the change in voltage levels of the samples as they progress from left to right. When this signal is routed to a comparator with a fixed zero-crossing threshold, the 1 clock period jitter becomes visible with a scope in the infinite persistence mode. Incidentally, the jitter magnitude is the same if only the MSB of the p-bit input code to the DAC was examined.

If the number of samples/period is non integer, the sample used in each reconstructed period will be different. Transitions to the next level are different each period. When the output signal of the DAC is displayed on a scope with infinite persistence, jitter of 1 input clock cycle T_c can be observed. Reducing the phase truncation error (by increasing p) does not change the amount of jitter.



If the number of samples/period is integer, the sample used in each reconstructed period will be the same. Transitions to the next level are the same each period. When the output signal of the DAC is displayed on a scope with infinite persistence, the same waveform is displayed each period. The display is stable, no jitter is observed.



In the phase truncation DDS architecture the quantisation of the accumulator output introduces a phase error in the phase slope by discarding the least significant part, actually fractional component, of the high-precision phase accumulator.

The phase error due to the discarded fractional part of the address count is a periodic series (saw tooth) which results in an undesired spectral line structure. The line spectrum associate with this correlated error sequence is impressed on the final output waveform and results in spectral lines in the synthesizer output spectrum.

The spurious signals caused by the phase truncation can be very high. By keeping N odd the spectral performance can be improve just about 3dB.

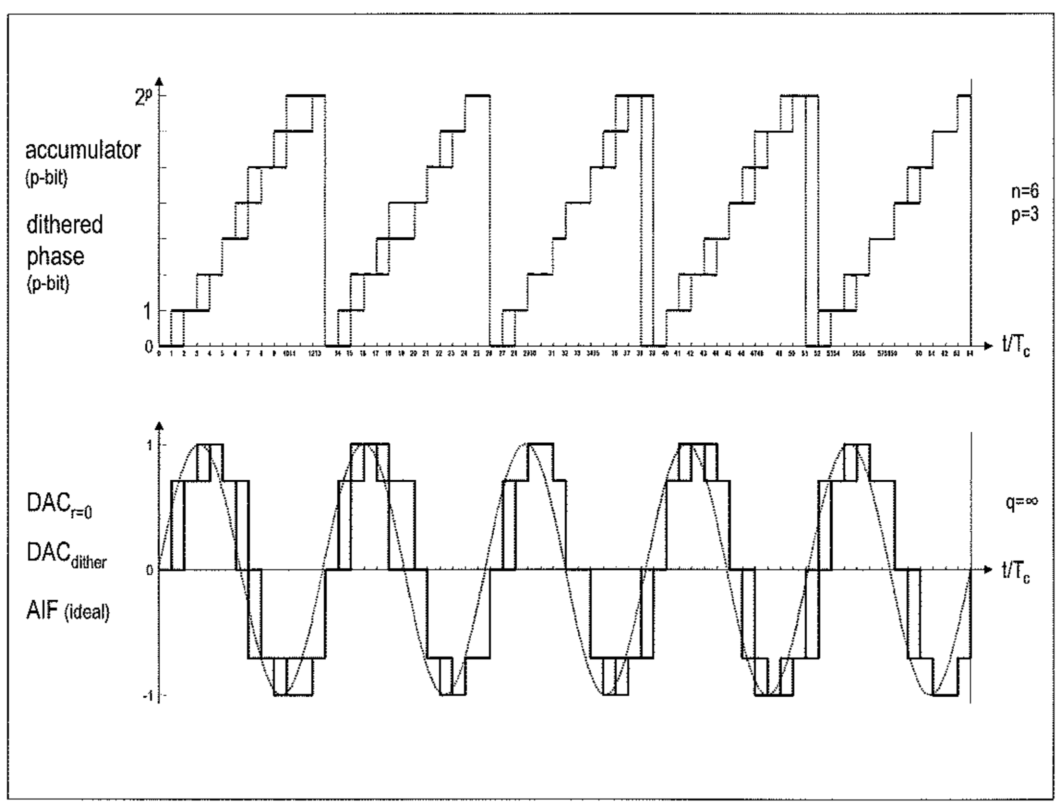
The spur level can be further reduced by breaking up the regularity of the address error with an additive randomizing signal. This randomizing sequence, called *dither*, is a noise sequence, with variance approximately equal to the least significant integer bit of the phase accumulator. The dither sequence is added to the high-precision accumulator output prior to quantization. The resulting *dithered DDS* architecture is shown.

The spur signals are located at a finite number of discrete frequencies. The idea behind the phase dithering is the same as behind a spread spectrum systems. By adding pseudorandom noise to the deterministic periodic signal the power of discrete components is lowered and their spectrum is spread over a wide range of frequencies. The price paid for doing that, is the rise of the noise floor.

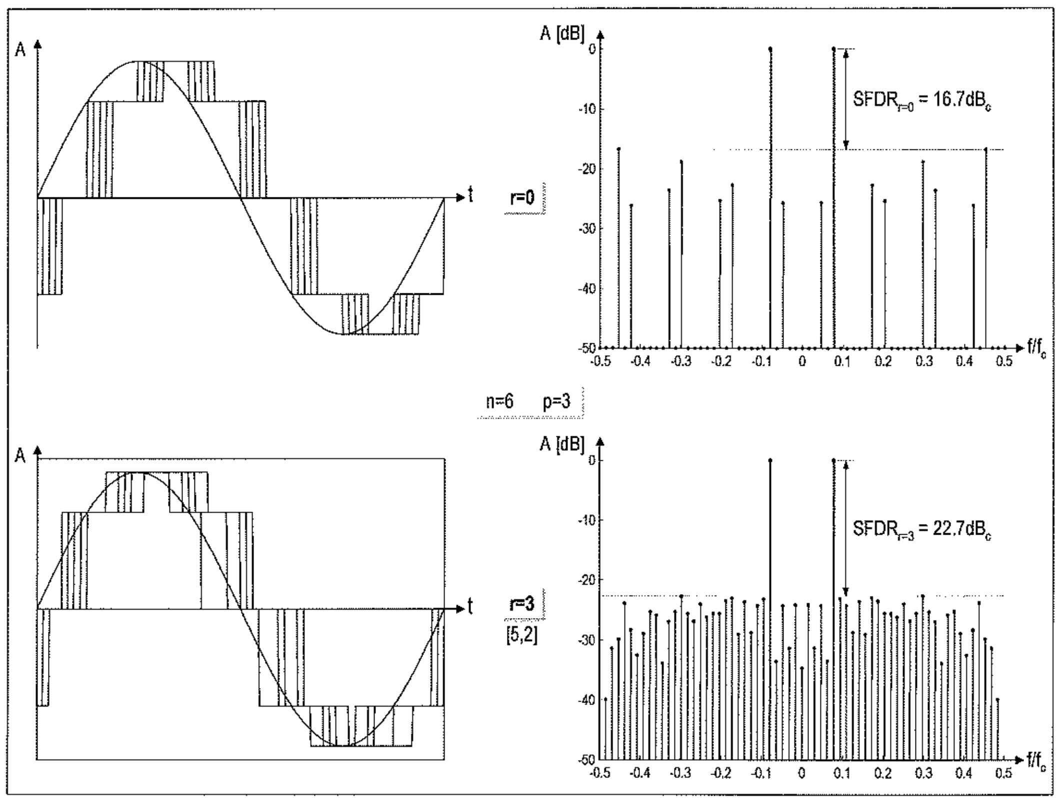
The first-order phase dithering architecture uses a pseudorandom M-sequence binary generator with a generating primitive polynomial of degree L. Any part of the generator of width r can be used for phase dithering. The period of the M-sequence is 2^L-1. This is also the period of the number sequence formed by taking any r output bits of the shift register forming the generator. The resulting noise floor level is lower for longer periods of the dithering sequence.

What remains to be determined is the value of the dither width r. If the dither width $r < p-n$ then the dithering will not work too well because in many cases its influence would remain buried in the truncated part of the phase accumulator. If $r > p-n$ the desired part of the phase value is dithered. The noise floor would be increased without really contributing to the power spreading of the spurious signals. Therefore, the best choice for the dither width is:

$$r = p-n$$



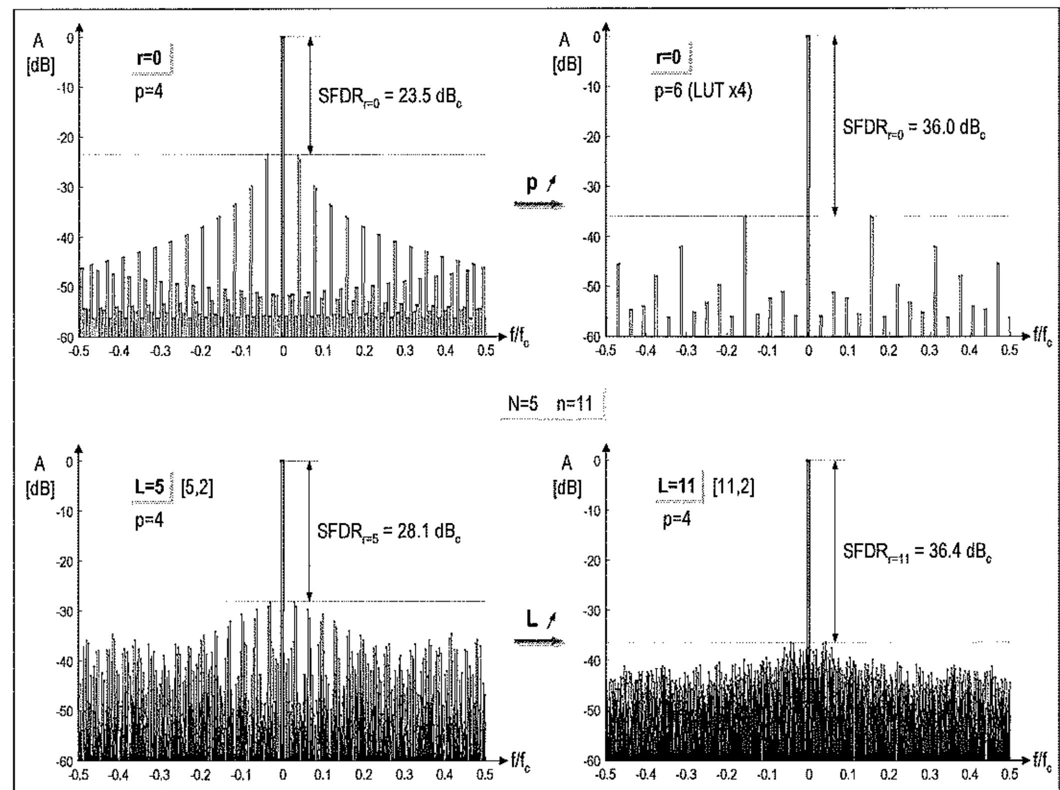
The added phase dither introduces a random change (between 0 and 1) on the address of the LUT. This results in a extra random jitter on the output waveform of the LUT and the DAC.



The phase dithering results in an extra random jitter on the output waveform of the DAC, as shown in an eye-diagram.

By adding pseudorandom noise to the deterministic periodic signal, the power of discrete spurs is lowered and their spectrum is spread over a wide range of frequencies (like in spread spectrum systems). This noise-shaping technique de-correlates the truncated phase-values and therefore reduces the discrete spurs, acting like a spreader.

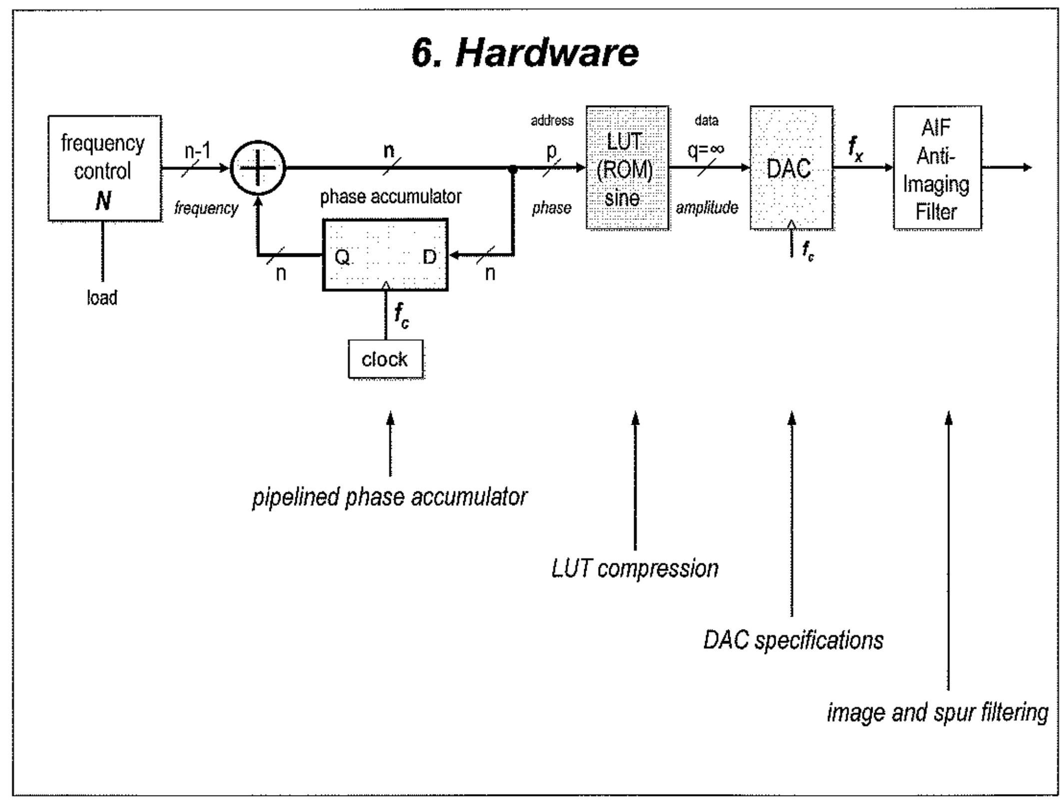
For low values of n the gain in the Spurious Free Dynamic Range (SFDR) can be 6 dB (as shown).



For large values of n, the dithered DDS supplies, approximately, an additional 12 dB of Spurious Free Dynamic Range (SFDR) in comparison to a phase truncation design. The additional logic resources required to implement the dither sequence generator are not significant.

To provide S dB of spur suppression using a phase truncation DDS, as referenced to the 0 dB primary tone, the internal lookup table must support at least ceil(S/6) address bits. To achieve this same performance using the dithered architecture requires two fewer address bits, minimizing the number of block RAMs (or logic slices for a distributed memory implementation) used in the FPGA implementation with a factor of 4.

In summary, for a dithered DDS implementation, the number of address bits needed to support S dB spur suppression is equal to ceil(S/6)



The high-level architecture of a generic DDS system is a simple assembly containing three parts:

1. The time *reference* part: a fixed rate system clock.

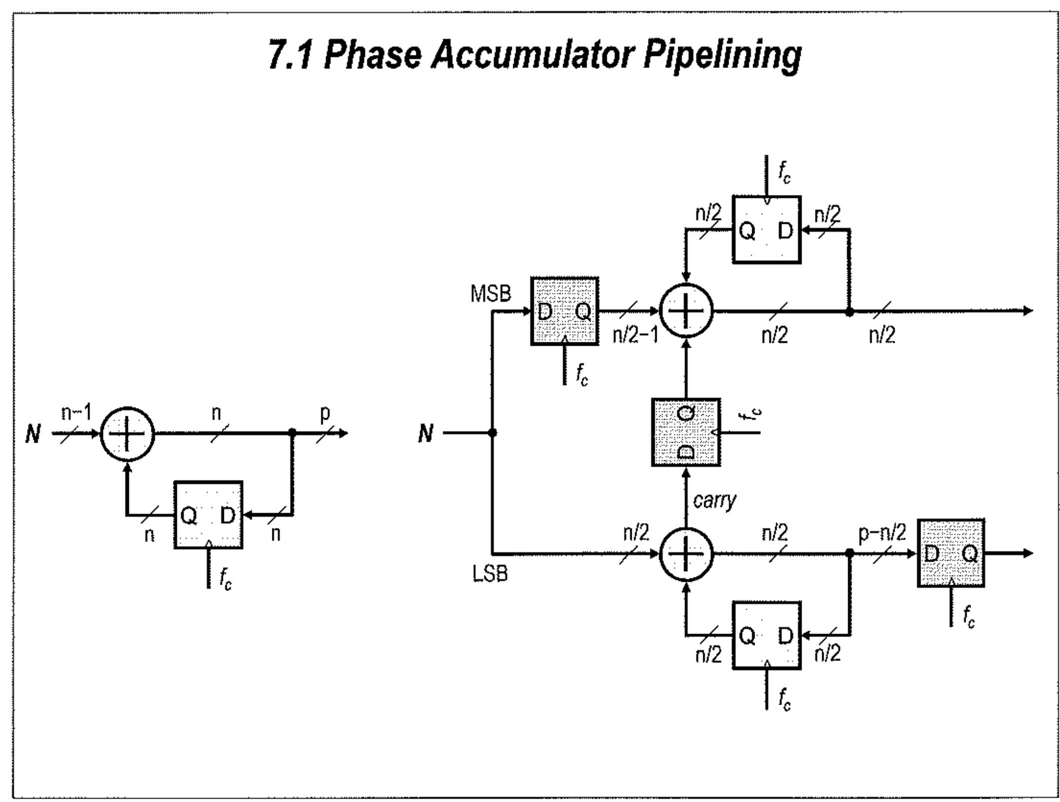
2. The *digital* part, the Numerically Controlled Oscillator (NCO), generates numerical samples of (sine) waveform at clock rate. It consists of:
- An overflowing phase accumulator (register length *n*-bit),
- A Look-Up Table (LUT, memory address length *p*-bit, data width *q*-bit)

3. The *mixed/analog* part reconstructs the analog wave with a digital to analog converter (DAC) and anti-imaging filter (AIF).
- Digital-to-Analog Converter (data length: *q*-bit)
- Anti-Image Filter.

To increase the performance (speed, spectral purity, cost) of the DDS, the different parts can be optimised:

- Accumulator: increase clock speed by pipelining
- LUT: reduce the size and increase the speed by LUT compression
- DAC: non-linearity's in the DAC reduce the spectral purity (SFDR) of a DSS (specification)
- AIF: the images of the sampling process and some spurs can be filtered out

7.1 Phase Accumulator Pipelining



The critical path in the accumulator is the carry chain of the n-bit adder. This can be seen as a cascade of two n/2-bit adders each with a speed that is 2 times higher than the original n-bit adder.

The clock speed of the total adder can be doubled by the use of pipelining. At the junction of the carry chains of the two n/2-bit adders a flip-flop is introduced. This reduces the total critical path by 2.

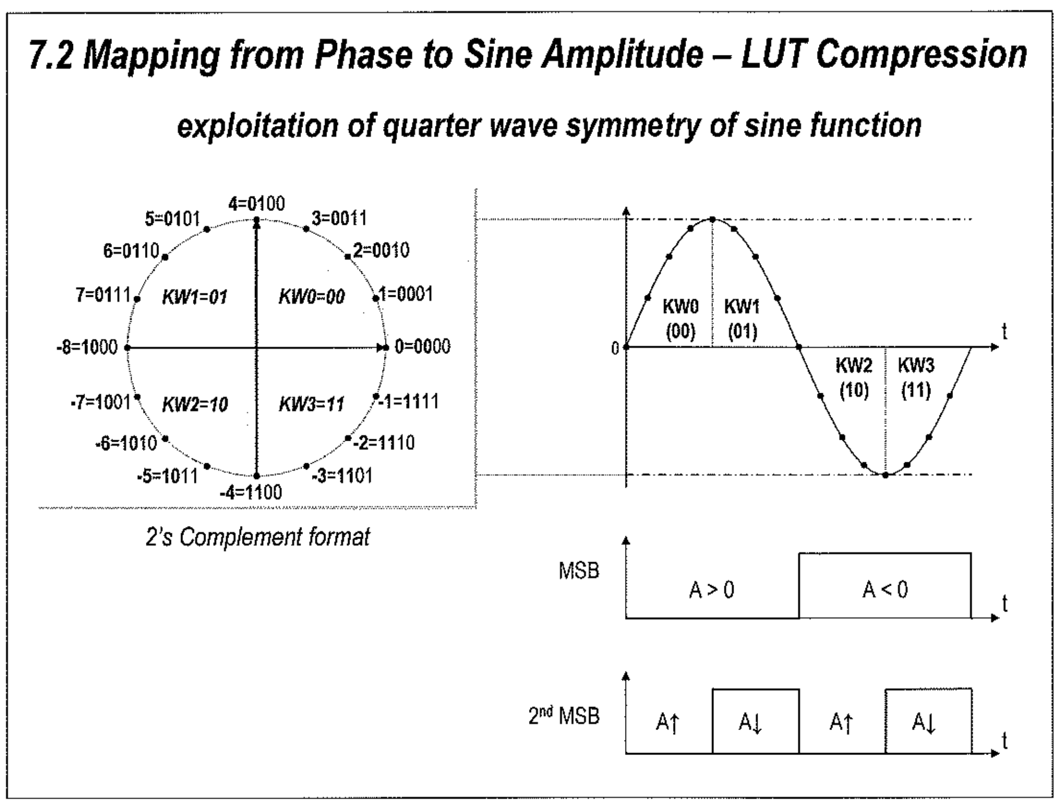
To align the n/2 - 1 MSBs of the frequency word N with the incoming carry, a register is introduced.

To align the n/2 LSBs of the accumulator output with n/2 MSBs, a register is introduced.

A total of n flip-flops are needed to increase the clock speed of the accumulator by 2.

7.2 Mapping from Phase to Sine Amplitude – LUT Compression

exploitation of quarter wave symmetry of sine function



The LUT is a bottleneck for high frequency performance since its functions cannot be pipelined to increase the sampling rate.

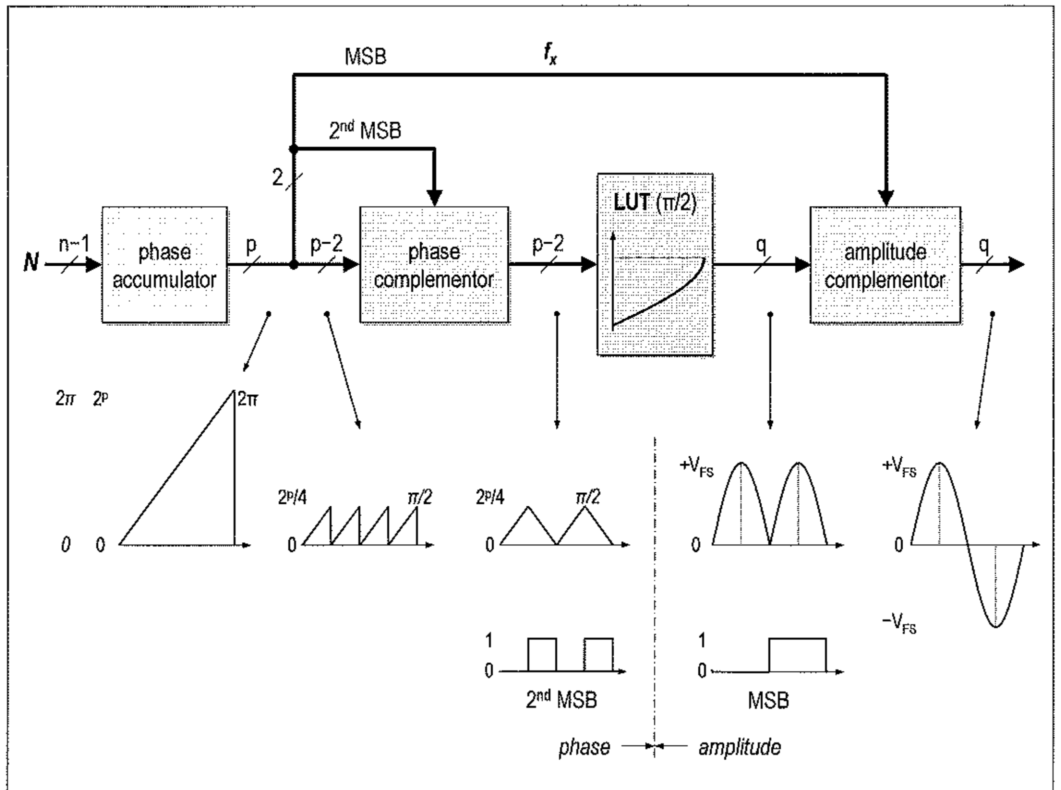
For the quality of the output signal (less noise) it is desirable to increase the resolution of the LUT. But larger LUT size means higher power consumption, lower reliability, lower speed, and increased costs. Therefore LUT compression techniques are used.

A well-known technique of LUT compression is to store only π/2 rad of sine information, and to generate the sine look-up table (LUT) samples for the full range of 2π by exploiting the quarter wave symmetry of the sine function. The decrease in look-up table capacity is paid by the additional logic necessary to generate the complements of the phase accumulator and look-up table output.

In most practical DDS digital implementations, numbers are presented in 2's Complement format. The two most significant phase bits determine the quadrant.

The most significant bit (MSB) determines the required sign of the LUT output.

The second most significant bit (2nd MSB) determines whether the amplitude of the LUT output is increasing or decreasing.



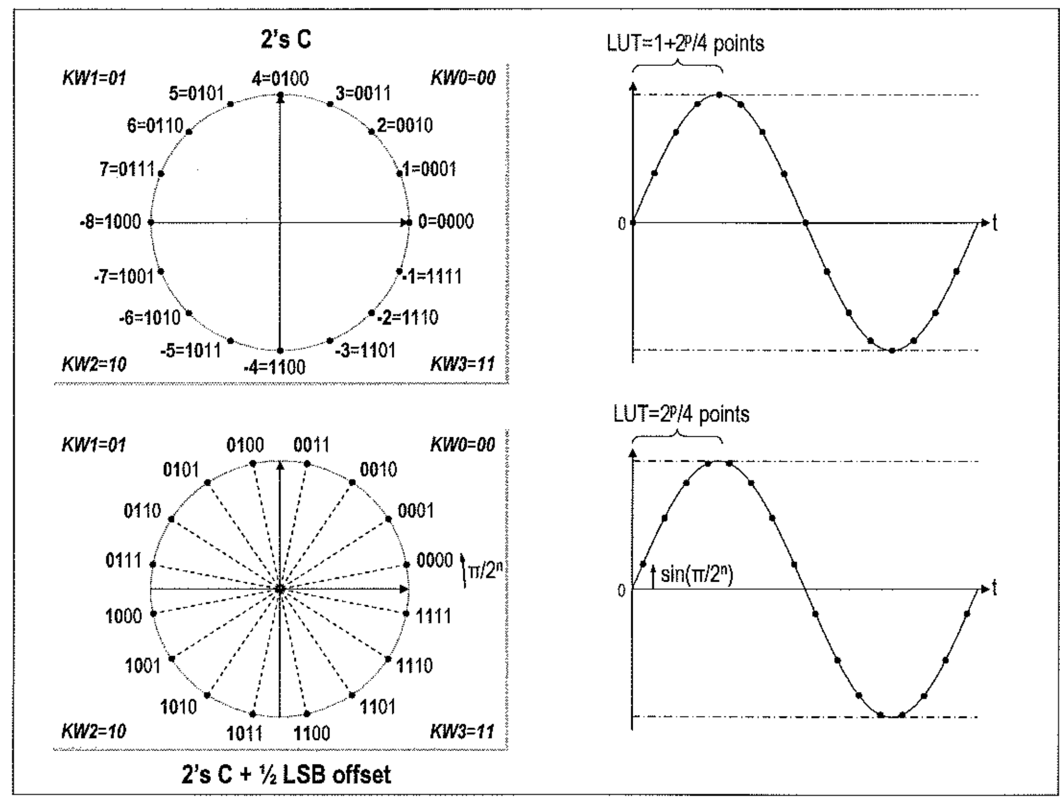
The phase accumulator output (the p-2 LSBs) is used 'as is' for the first and third quadrants.

For the second and forth quadrant, the phase bits must be complemented so that the slope of the saw tooth is inverted.

The sampled waveform of the LUT output is a full rectified version of the desired sine wave.

The final sine wave is then generated by multiplying the full wave rectified version by −1 when the phase is between π and 2π.

The MSB of the digitized sine wave outputs from a DDS is a square wave pulse whose repetition rate is that of the programmed fundamental output frequency. For many applications, the MSB can be used as a programmable clock source directly. The worst case jitter that the MSB will have for any given output frequency is $1/(2f_c)$, where f_c is the system clock frequency of the DDS. As long as the associated jitter component of the MSB is acceptable for the clock stability requirements involved, it is quite suitable for use as a clock pulse source. As seen by the clock-to-jitter relation, by maximizing the system clock frequency, the jitter can be minimized. Also, the frequency rate of the jitter component on the MSB varies for different programmed output frequencies. If the programmed output frequency is an exact integersubmultiple of f_c , (e.g., $f_x = f_c/4$) then the jitter component on the MSB output will only be the jitter inherent in the system clock itself. For noninteger submultiples of f_c, the rate of the jitter on the MSB will generally be at a low frequency that will be at different rates depending on the relationship of the output frequency to the system clock frequency. In this case, the user will have to empirically quantify the jitter rate on the MSB for a given clock-to-output frequency plan if it is critical to the system requirements involved.



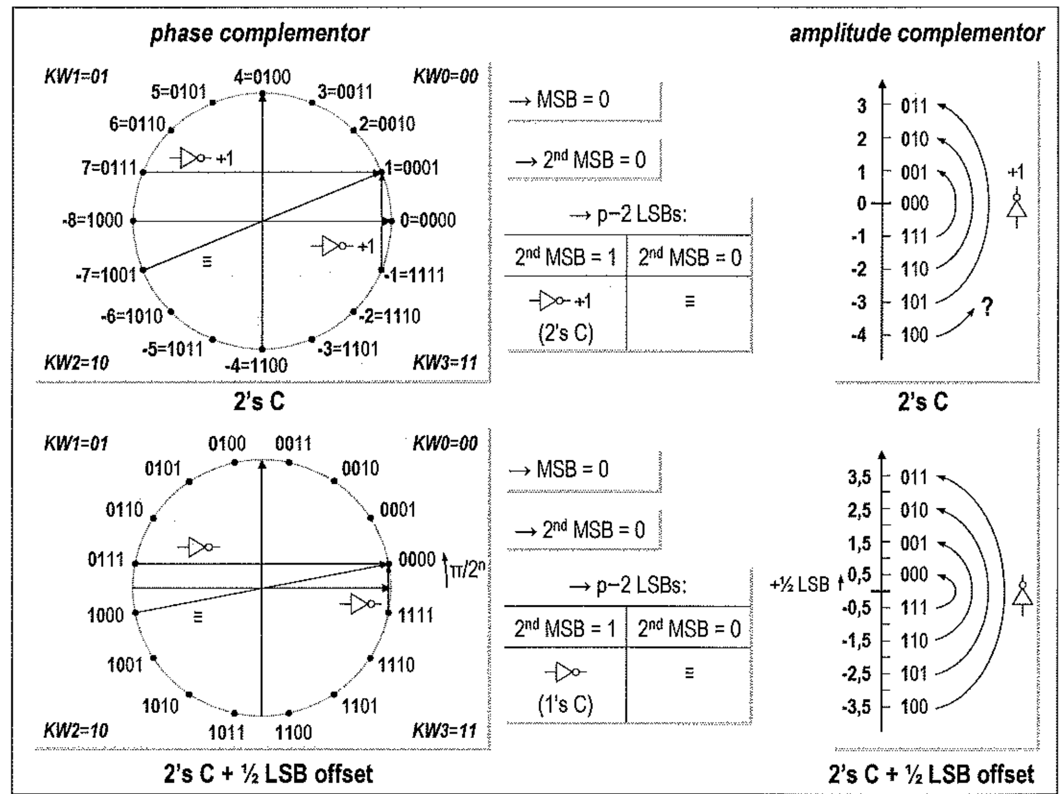
In most practical DDS digital implementations, numbers are presented in 2's Complement format.

Sign inversion (multiplication by -1) requires a 2's complementor: the needed hardware is an inversion and an incrementation (+1).

However, if a ½ LSB offset is introduced into a number that is to be complemented, than a 1's complementor may be used in place of a 2's complementor without introducing an error. This provides savings in hardware since a 1's complementor may be implemented as a set of simple exclusive-or gates.

This ½ LSB offset is provided by choosing the look-up table (LUT) samples such that there is a ½ LSB offset in both the phase and the amplitude of the samples.

If there is no phase offset, then 0 and π/2 have the same phase address to the quarter wave memory, and one more address bit is needed to distinguish these two values.

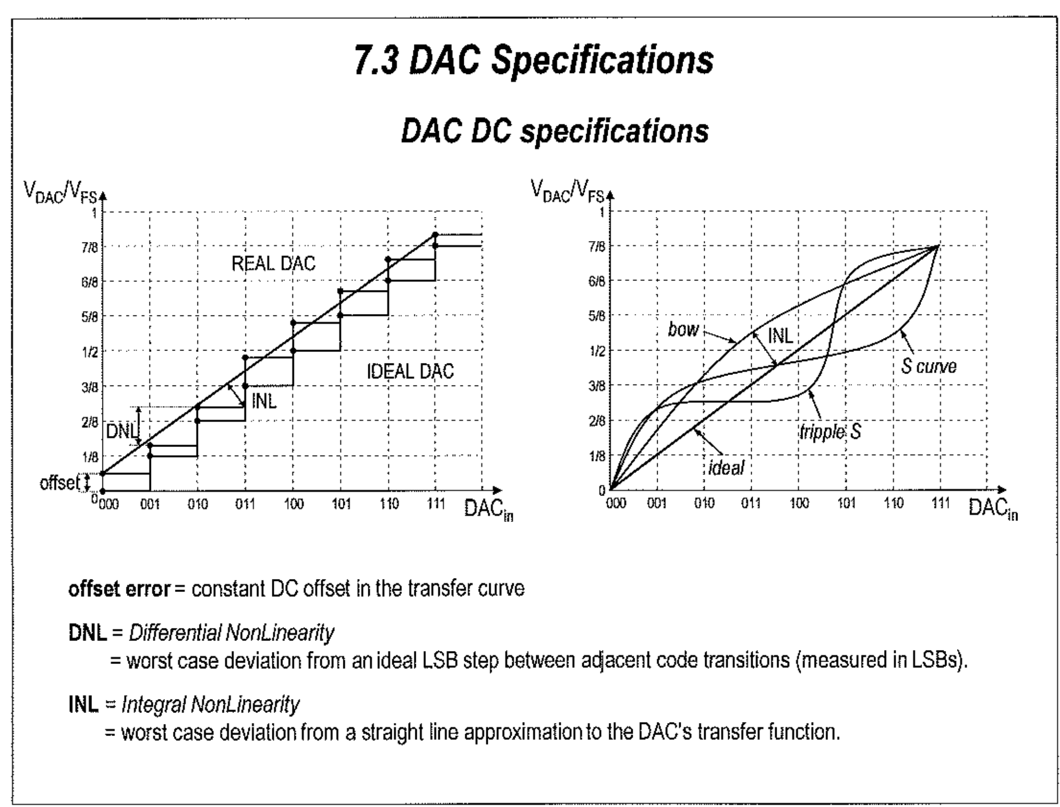


A ½ LSB offset is introduced in all phase addresses. In the case shown a ½ LSB corresponds with a phase of π/32. The ½ LSB phase offset is added to all the sine look-up table (LUT) samples. When the 2nd MSB = 1 than the 1's complementor maps the phase values to the first quadrant.

A ½ LSB offset is introduced into the amplitude (LUT output) that is to be complemented. The negation can now be carried out with a 1's complementor.

Remark:

The look-up table may be further reduced in size by dividing the phase table into coarse and fine segments, and storing values of the sine-wave after subtraction of the linear component, the dynamic range of the stored amplitude may be reduced. These techniques significantly reduce the data stored in ROM and largely replace them by arithmetic operations, which can be easily pipelined.



High speed DACs are traditionally specified in the time domain. The key specifications of a DDS system are in the frequency domain. This makes the selection of a suitable DAC difficult.

DAC DC SPECIFICATIONS

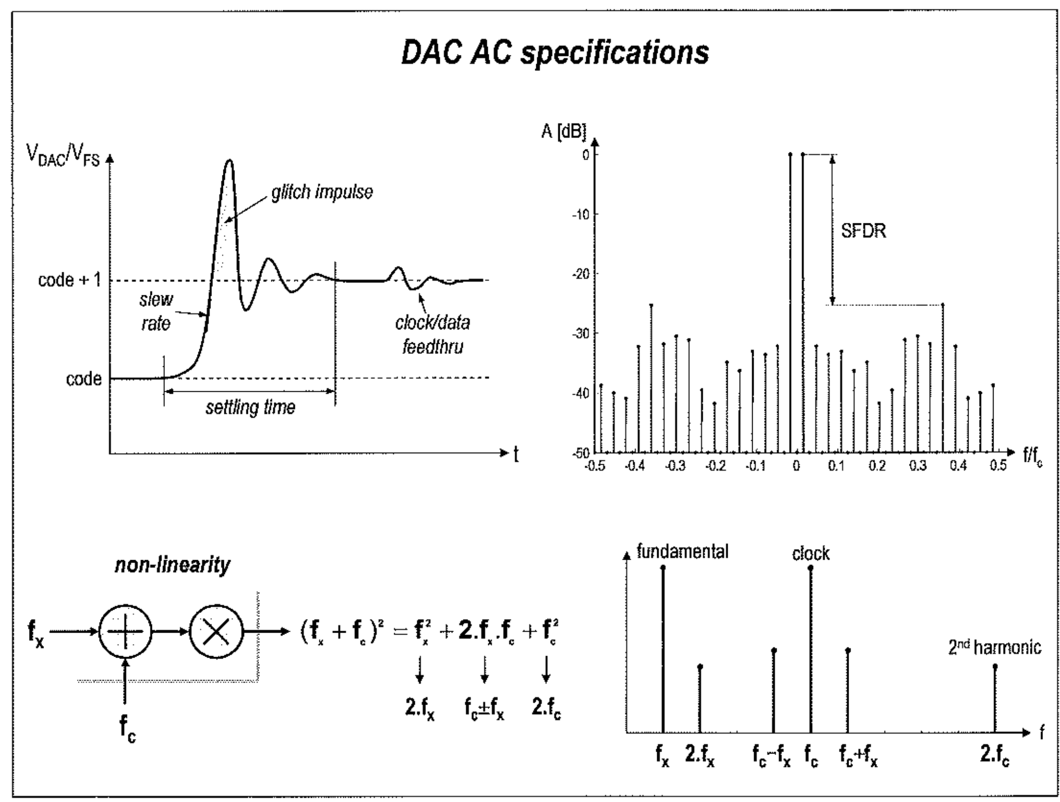
offset = constant DC offset in the transfer curve (no influence on frequency domain)

gain = full-scale output in relation to its reference circuit

DNL = Differential NonLinearity = worst case deviation from an ideal LSB step between adjacent code transitions (measured in LSBs), and can be negative (less than 1 LSB step) or positive (greater than 1 LSB step). DACs with a DNL < -1 LSB are not guaranteed to be monotonic. A DAC with a DNL error of 1 LSB (every other quantisation level missing) would have a reduction in SNR of 6 dB. This reduction in SNR, however does not predict how the distortion is spread over the Nyquist bandwidth. If the DNL error concentrate portions of the noise into a single frequency (usually a harmonic) this could limit the SFDR. The DNL specification can be misleading for DDS applications if not interpreted properly.

INL = Integral NonLinearity = worst case deviation from a straight line approximation to the DAC's transfer function. The straight line eliminates the DC errors (offset and gain) that have no influence on the frequency domain. Like the DNL specification, the INL measurement is a worst case deviation. It does not indicate how many DAC codes reach this deviation, nor which direction away from the best straight line the deviation occurred. Three transfer curves are shown, with the same INL, but with distinct effects in the frequency domain. The transfer function of the bow curve will introduce a prominent 2nd harmonic distortion, while the symmetrical S-curve will tend to introduce 3rd harmonic distortion.

Because the linearity patterns of any two specific DAC designs will be different, it is impossible to compare their frequency domain performance by comparing their linearity specifications.



DAC DC SPECIFICATIONS

The AC time domain specifications of a DAC relate to its code-to-code transitions as illustrated in the figure. It is difficult to predict the frequency domain effects of any of these specifications or their combinations.

slew rate = rate at which the output changes at the start of a transition. A DAC with a high slew rate will produce an output transition that is closer to ideal than one which is slower. A difference in rising and falling slew rate will tend to concentrate energy in the harmonics of the fundamental output frequency.

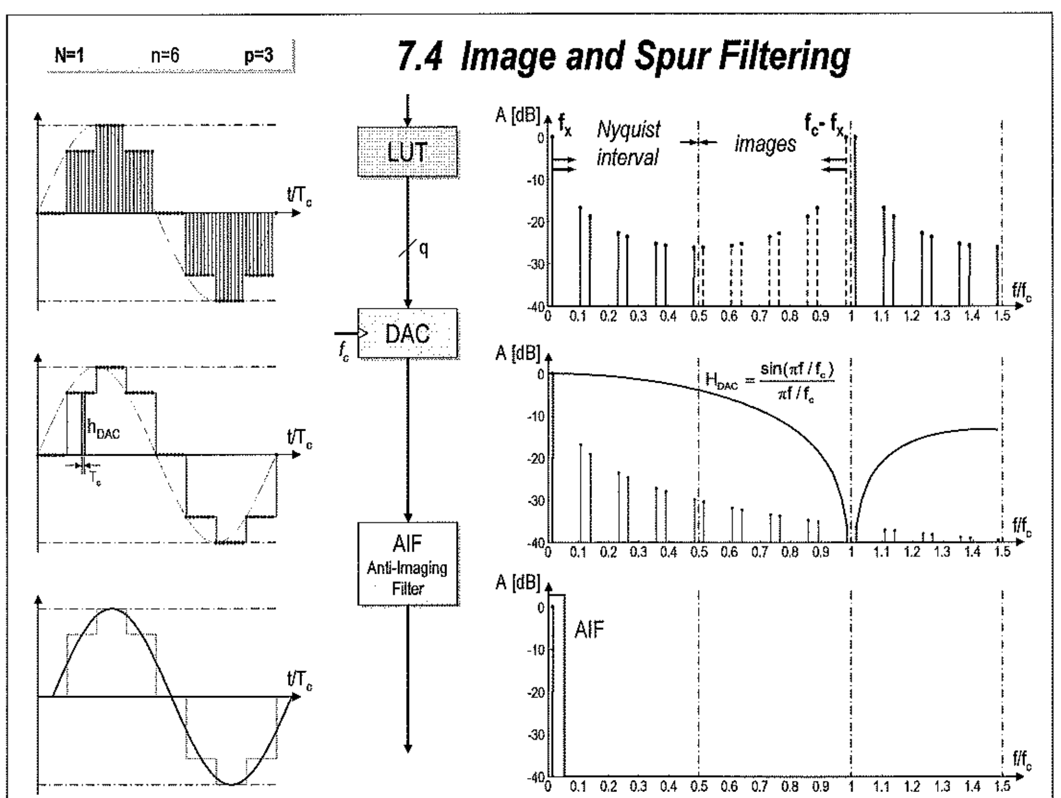
glitch impulse = a measure of the initial transient response (overshoot) of the DAC between two output levels, usually measured as the area of the transient. This transient is commonly associated with the time skew between the data bit transitions or by unequal propagations delays through the internal logic (DAC architecture). This causes the DAC's output to approach an intermediate state, and adds unwanted energy to the output spectrum. The glitch impulse has a broad spectrum, even that has spurs from DC up through the sampling frequency.

settling time = the interval from the time the DAC's output leaves the error band (usually 1/2 or 1 LSB) around its initial value, and when it settles within the error band around its final value. A small setting time will produce an output transition that is closer to ideal.

clock/data feedthru = feedthrough of the clock or data transitions to the DAC's output. This adds to the frequency content of the DDS's output spectrum. These effects are often related to the circuit layout. A series resistance in the input data connections works with the input capacitance of the DAC to form al low pass filter. This technique is not recommended on the clock connection as it will tend to add jitter (phase noise).

The AC nonlinearities become a more significant portion of each clock cycle as the clock or analog output frequency increases. The clock frequency of the DAC is reduced when possible.

Spurious distortion will tend to appear in the output spectrum at alias frequencies, $Af_x \pm Bf_c$ where A and B are integers. The effects of these aliased components are concentrated near the fundamental frequency when the fundamental output frequency is nearly an integer fraction of the clock frequency. Designs that cover wide analog output bandwidths may find it hard to avoid this effect even at low clock rates.



To understand the nature of the (amplitude and phase) quantization distortion, note the sharp edges in the DAC output signal. These sharp edges imply the presence of high frequency components superimposed on the fundamental. These frequency components constitute the quantization distortion.

In the time domain, the LUT output is the multiplication of the sampling strobes at f_c by the sinusoid wave form producing a train of impulses at the sinusoid amplitude. In the frequency domain, the sampling strobes at f_c produce a train of impulses at frequencies of m times f_c (m = ... -2, -1, 0, 1, 2 ...). This multiplication in the time domain, results in a convolution in the frequency domain of the sinusoid and the sampling clock to produce the frequency domain representation of the LUT output. This operation introduces aliasing. An alias of f_x is found at f_c-f_x.

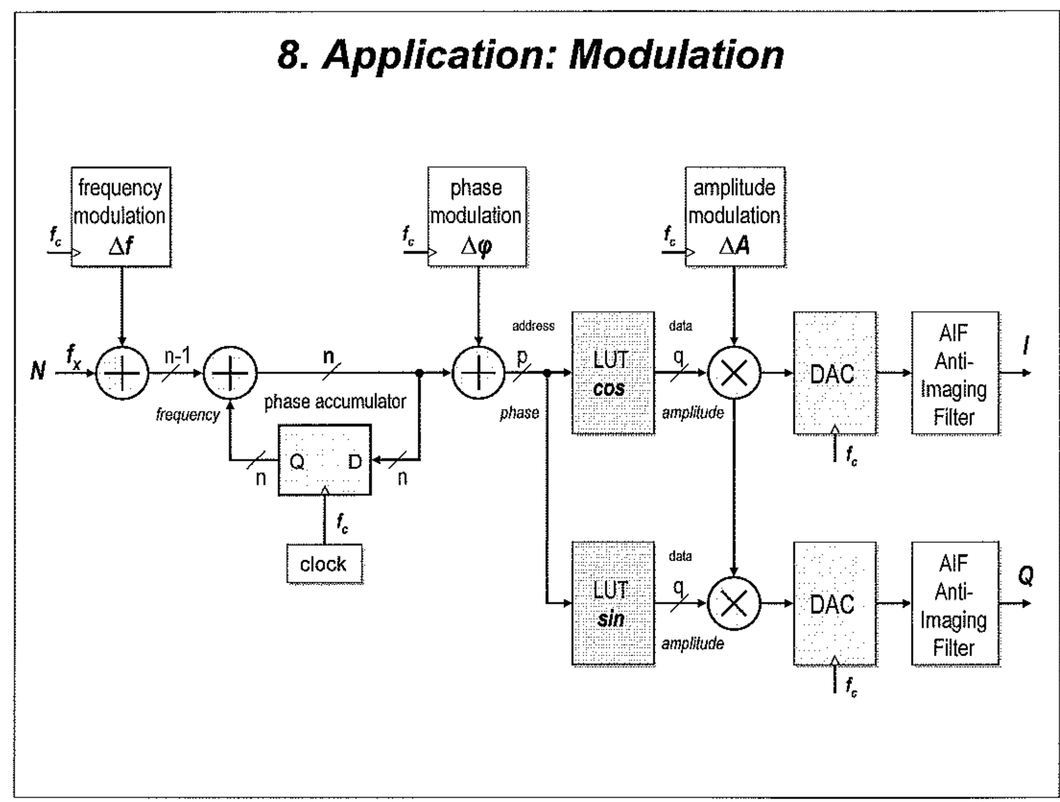
The DAC takes the LUT output values and translates these values into analog voltages. The DAC output is a sample and hold circuit that takes the LUT digital amplitude words and converts the value into an analog voltage and holds the value for one sample clock period. The time domain plot of the DAC processing is the convolution of the LUT sampled output values with a pulse of one sample clock period T_c. The frequency domain plot of the sampling pulse is a $\sin(\pi f/f_c)/(\pi f/f_c)$ function with the first null at the sample clock frequency f_c. Since the time domain was convolved, the frequency domain is multiplied. This multiplication dampens the LUT output with the $\sin(\pi f/f_c)/(\pi f/f_c)$ envelope.

The Anti-Imaging filter (AIF) must pass the highest frequency which will be generated ($f_{x,max}$), but must begin its stop-band at $f_c - f_{x,max}$. Steep roll-off filters with high stop-band attenuation are hard to build. A reasonable compromise in this trade-off occurs when $f_{x,max} = f_c/3$. This allows the filter of one octave transition band $[f_c/3 \rightarrow 2f_c/3]$. The needed stop-band attenuation depends on the spurious component specification of the output.

The inherent jitter is interpolated out.

Chebyshev filters have a very nasty overshoot. A good choice for arbitrary waveforms (ramps, triangles, ...) is the Bessel filter. It has a slower rolloff when compared to the Chebyshev filter, but it is nearly phase-linear. The lack of dispersion in a phase-linear filter will preserve the pulse shape and prevent any ringing in the time domain. A seventh degree Bessel filter, with a −3 dB cutoff at $f_{-3} = f_c/4$, is a good choice for filtering arbitrary waveforms. This filter will exhibit an output risetime of $0.35/f_{-3}$.

8. Application: Modulation



The DDS generator consists of the phase accumulator, which continuously updates the phase; arbitrary waveform memory LUT, a signal DAC and anti-aliasing low pass filter.

The direct digital synthesis technique can support various modulation types at the same time. One possible implementation of a more technically advanced DDS is depicted.

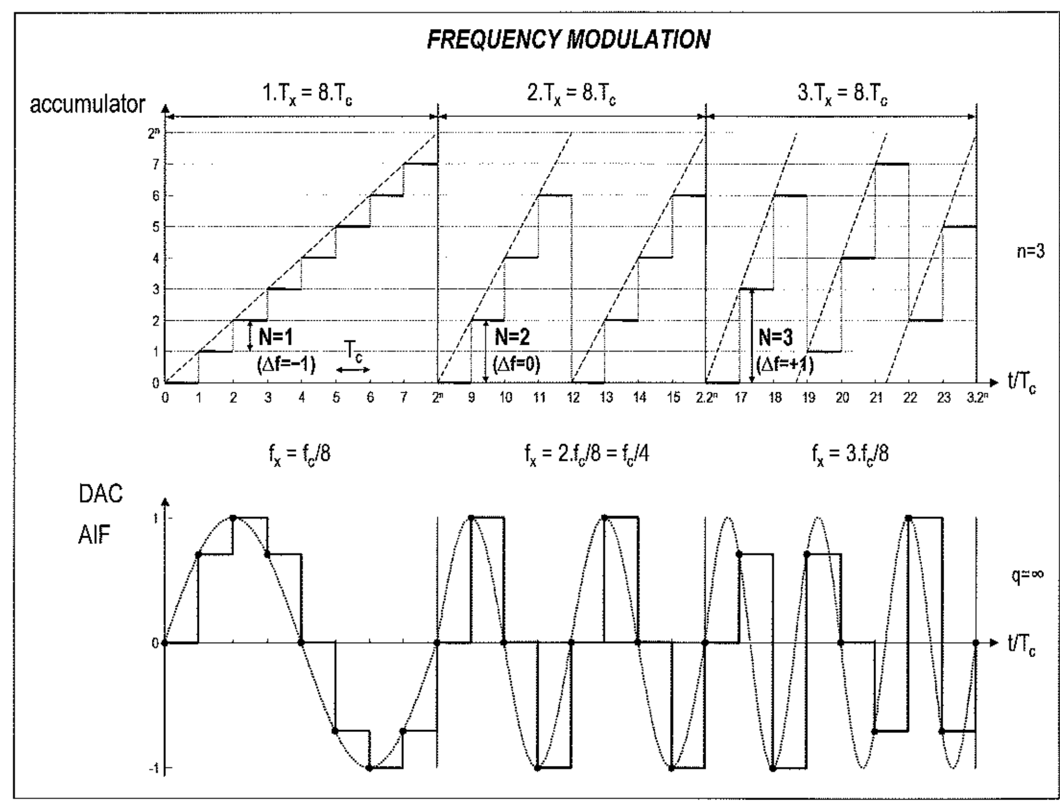
This DDS device offers I-Q modulation with frequency, phase and amplitude control of the output. Both phase and quadrature outputs are available.

The two additional adders and multipliers are intended to provide frequency, phase and amplitude modulation possibilities at the same time.

Wide frequency or phase deviations are no problem. Any phase or frequency hop may be programmed and executed in a single clock cycle. The phase relationship is accurate and independent of frequency. Since the control registers may be modified very quickly, high modulation frequencies are possible.

There are two approaches for amplitude modulation of the output waveform. Either the digital outputs from the RAM or the analog output from the DAC may be multiplied by the desired amplitude.

This DDS implementation is widely used in measurement devices (signal generators, spectrum analyzers) and communication equipment (cell phone-base stations, transceivers) to create radio-frequency signals with high-frequency resolution.



DDS presents an attractive alternative to analog PLLs for agile frequency synthesis applications. DDS is a superior technology for generating highly accurate, and frequency-agile (rapidly changeable frequency over a wide range), low-distortion output waveforms.

A DDS architecture holds major advantages over an equivalent PLL-based agile analog synthesizer:

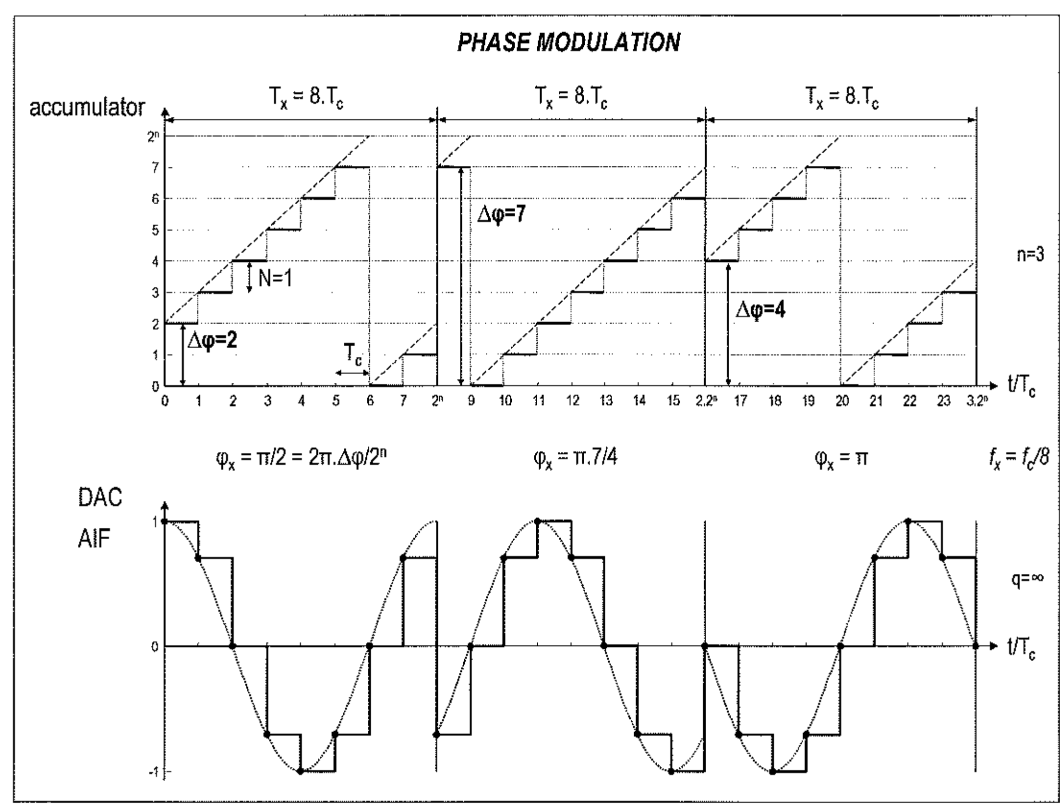
Output frequency resolution: DDS has the ability to tune with extremely fine frequency- and phase resolution (frequency control in the millihertz (mHz) range and phase control < 0.09°), and to rapidly "hop" in frequency (up to millions of output frequency changes per second). The output of a DDS is phase-continuous during the transition to the new frequency. In contrast, the basic PLL-based analog synthesizer typically has an output tuning resolution of 1 kilohertz; it lacks the inherent resolution afforded by the digital signal processing.

Output-frequency switching time: the analog PLL frequency switching time is a function of its feedback loop settling time and VCO response time, typically > 1 ms. A DDS-based synthesizer switching time is limited only by DDS digital processing delay, typically 50 ns.

Tuning range: A critical feedback loop bandwidth and input reference frequency relationship determines the stable (usable) frequency range of the typical analog PLL circuit. DDS-based synthesizers are immune to such loop filter stability issues and are tuneable over the full Nyquist range.

Phase noise: Because of the frequency division, the output phase noise of a DDS synthesizer is actually better than that of its reference clock source. While analog PLL-based synthesizers have the disadvantage of actually multiplying the phase noise present in their frequency reference.

The frequency of the DDS signal is changed by changing the frequency tuning word N.



Adding an offset $\Delta\varphi$ to the accumulator output directly results in a phase jump. It changes the address of the LUT, jumping to an other phase in the stored waveform.

The maximum accumulator output 2^n corresponds with a phase change of $2.\pi$. An offset $\Delta\varphi$ results in a phase jump:

$$\varphi_x = 2\pi.\Delta\varphi/2^n$$

References

"A Technical Tutorial on Digital Signal Synthesis" – Analog Devices

"Numerical Distortion in Single-Tone DDS" – Zs. Pápay

"Output Spectrum of a Direct Digital Synthesiser" – Martin Pechanec

"DDS" – Xilinx CORE generator System – Product Specification

"Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis" – Jouka Vankka
IEEE Transactions on Ultrasonic, Ferroelectrics and Frequency Control, vol. 44, no. 2, 1997

"Single-Chip Direct Digital Synthesis vs. the Analog PLL" – Jim Surber and Leo McHugh

"Choosing DACs for Direct Digital Synthesis" – David Buchanan

HOGESCHOOL VOOR WETENSCHAP & KUNST | **DE NAYER INSTITUUT**
SINT-KATELIJNE-WAVER

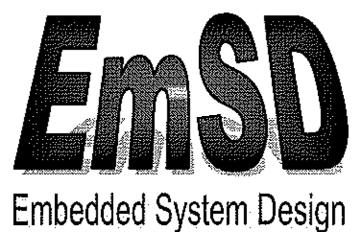
Digitale Synthese

Testen van Digitale Schakelingen

Bachelor in de Industriële Wetenschappen: elektronica-ICT
3e jaar, 6e semester

3Ba / SPE

Ba3.EL.06 / SPBa3.EL.06



ir. J. Meel

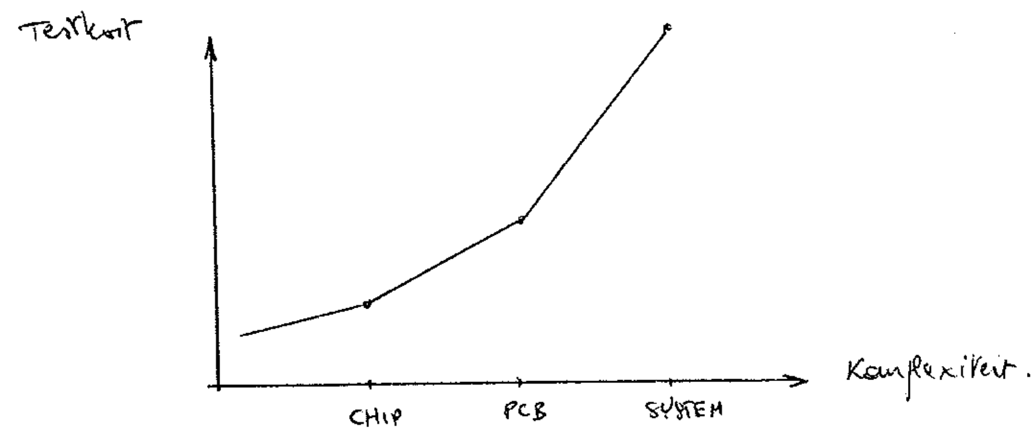
feb 2009

INLEIDING

Gedurende de ganse levenscyclus van een elektronisch systeem, moeten testen worden uitgevoerd. Fouten kunnen in de schakeling optreden tijdens:

- produktie
- assemblering
- stockage
- gebruik – onderhoud ("in the field").

De kosten verbonden met testen kunnen van minder dan 10% tot meer dan 60% van de produktiekosten in beslag nemen. Ze nemen toe met de komplexiteit van de schakeling:



Door de hoge integratiedichtheid wordt het mogelijk zeer komplexe systemen te ontwerpen, die niet meer testbaar zijn. Daar waar vroeger het logisch ontwerp en het testen (produktie) als onafhankelijke disciplines beschouwd werden, wordt het nu een noodzaak om reeds in de ontwerpfase rekening te houden met de testbaarheid van de schakeling (design for testability).

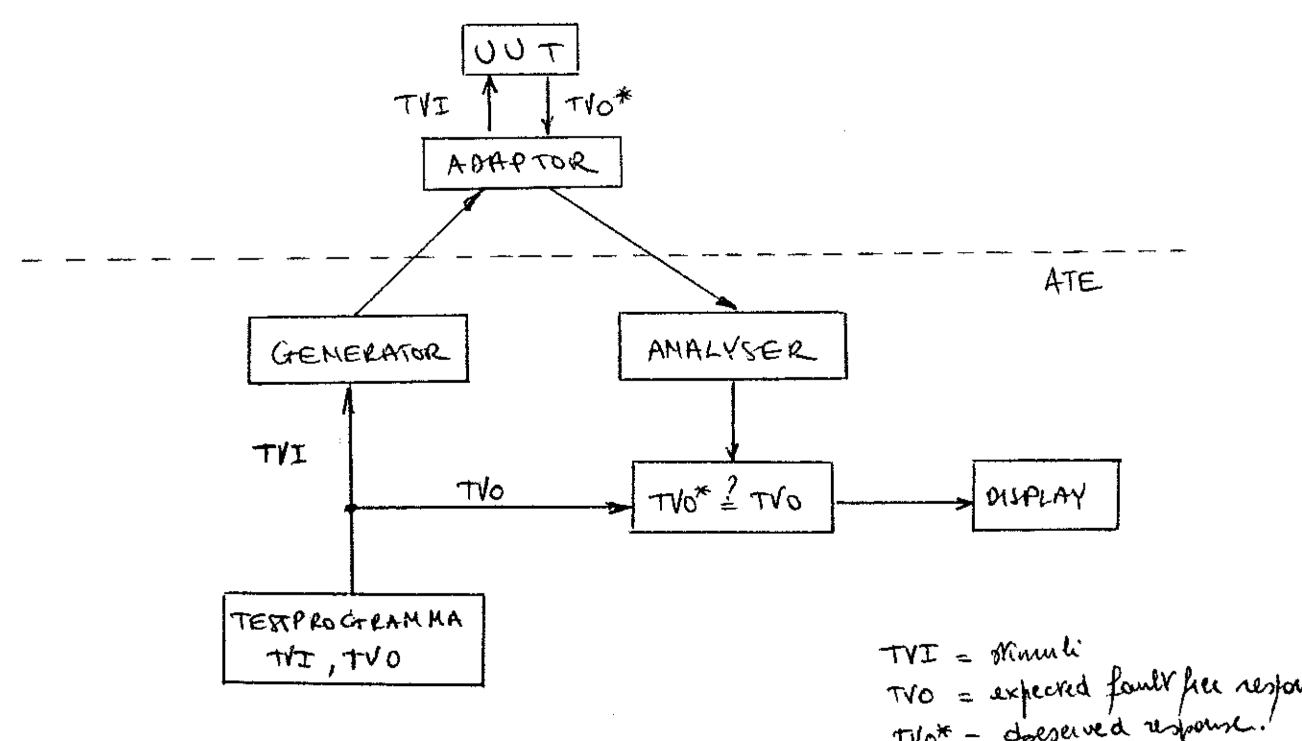
In de meest algemene zin bestaat testen uit het aanleggen van een sequentie ingangen (TVI = input testvectoren), observeren van de uitgangssequentie (TVO* = output testvectoren) en vergelijking met de verwachte uitgangssequentie van een fout-vrij systeem (TVO). Een afwijking tussen de geobserveerde en verwachte response (TVO* ≠ TVO) wijst op een faling van de hardware en wordt veroorzaakt door een fysische fout in de schakeling. Fouten worden onderverdeeld in

- logische fouten : de logische funktie wijzigt (Sa1, Sa0) → DC (static) test
- parametrische fouten : de amplitude van een elektrische parameter wijzigt ($t_p, t_r, t_d, I_{cc}, V_{TH}, \ldots$) → AC (dynamic) test

Er wordt onderscheid gemaakt tussen twee soorten testen:

- GO / NO GO (PASS / FAIL) : "detektie" hardware faling.
- DIAGNOSTIC : fout"detektie + localisatie" (foutisolatie), → onderhoud, bijsturing produktieproces.

De testen worden aangelegd aan de schakeling (Unit Under Test = UUT) met een Automatic Test Equipment.



De ontwikkeling van een testprogramma wordt opgesplitst in 3 aktiviteiten:

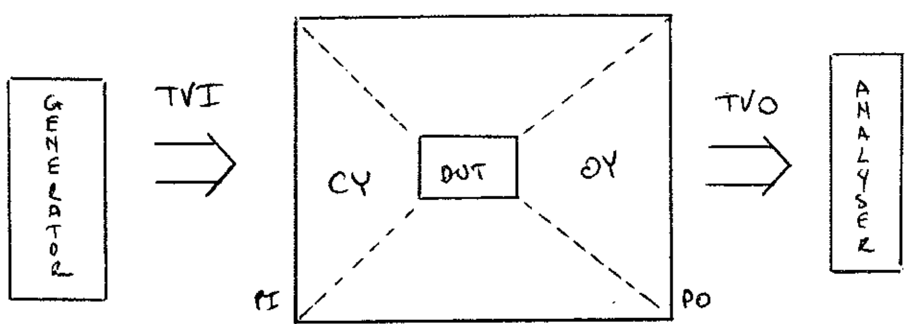
a) <u>Testgeneratie</u>

De nodige input testvektoren TVI en de bijhorende foutvrije responsie TVO moet worden opgesteld om de fouten uit een vooropgestelde foutlijst te kunnen "detekteren". (Voor fout lokalisatie moet ook de responsie van iedere fout worden bepaald).

De 'eenvoud' waarmee testpatronen kunnen worden opgesteld, wordt bepaald door de komplexiteit van de schakeling en de mate waarin de tester toegang heeft tot de schakeling (Testability = TY)

Het aantal fysische toegangspunten tot een elektronische schakeling is beperkt (verpakkingspinnen IC, edge konnektor, test socket, ...). Dit zijn de primary inputs (PI) en primary outputs (PO).

Indien een bepaalde deelschakeling moet uitgetest worden, dan moeten de patronen die door de generator worden afgeleverd, via de PI's tot aan de fysische ingangen van de deelschakeling worden gebracht. De mate waarin dit mogelijk is noemt men de <u>kontroleerbaarheid</u> (controllability = CY). De responsie van de deelschakeling moet tot aan de PO's gebracht worden opdat deze door de analyser zou kunnen worden geobserveerd. De mate waarin dit mogelijk is, noemt men de <u>observeerbaarheid</u> (observability = OY).



De <u>testbaarheid</u> van een schakeling is de kombinatie van kontroleerbaarheid en observeerbaarheid (TY = CY . OY).

Hoe beter de testbaarheid van een schakeling, hoe eenvoudiger het is de nodige testpatronen te genereren

b) Testevaluatie (foutsimulator)

Er moet worden nagegaan of het testprogramma de fouten van de vooropgestelde foutenlijst detecteert (fault coverage). Dit gebeurt met een foutsimulator: er worden fouten toegevoegd aan het software model van de schakeling en de invloed op de respons van de schakeling t.o.v. TVI wordt geëvalueerd. Op deze wijze kan worden bepaald hoeveel en welke fouten door het testprogramma kunnen worden gedetecteerd.

c). Uitvoering testprogramma (test uitvoering)

Het testprogramma wordt uitgevoerd op een gepast testtoestel (ATE).
Problemen hangen samen met de limieten van de ATE:
- hardware (snelheid, bidirectioneel switchen, ...)
- mogelijkheden (fout-woordenboek/guided probe, pulse-catching)

De testbaarheid van een schakeling bepaalt:
- ontwikkelingstijd testprogramma
- uitvoeringstijd testprogramma (produktietijd / onderhoudstijd)
- de haalbare efficiëntie van het testprogramma (defect level)

Daarom is het nodig dat reeds tijdens het ontwerp rekening wordt gehouden met de testbaarheid van de schakeling. (Testbaarheid moet worden gezien als een na te komen 'funktionele specificatie') nog voor dat het testprogramma wordt aangemaakt.

Drie benaderingen die zullen gebruikt worden bij het ontwerp van een testbare schakeling:

- Meting van testbaarheid

  Nog voordat het testprogramma wordt opgesteld, kan met de nodige software de testbaarheid van de verschillende knopen in de schakeling worden bepaald. Dit laat toe reeds in de ontwerpfase de testbaarheid te verbeteren (gericht).

- Gestructureerde ontwerptechnieken

  Specifieke ontwerptechnieken voor testbaarheid laten efficiënt ontwerp toe (+ self-test!)

- Checklist met praktische richtlijnen.

Een mogelijke definitie voor testbaarheid (Bennetts):

"Een schakeling is testbaar als een set testpatronen kan worden gegenereerd, geëvalueerd en toegepast zodanig dat de vooropgestelde performantie-niveau's (gedefinieerd in termen van fout detectie, fout-lokalisatie, test-uitvoering) binnen een vooropgesteld budget en tijdsplanning worden gehaald."

Kosten / baten analyse:

Kosten: ontwerptijd / extra hardware

Baten: Testkosten

1. ONTWIKKELING TESTPROGRAMMA.

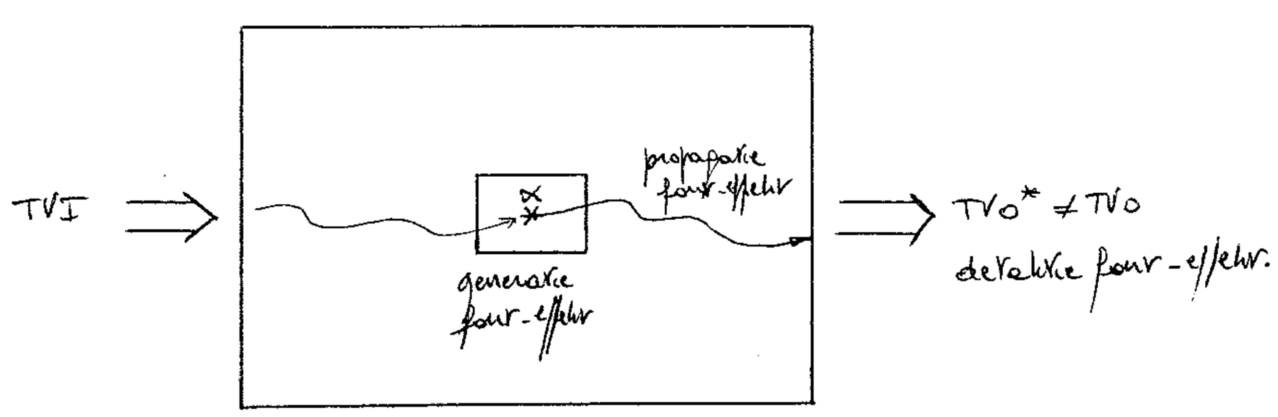
De drie voornaamste aktiviteiten bij de ontwikkeling van een testprogramma zijn :

- Test generatie
- Test evaluatie
- Test uitvoering.

Het is op dit niveau dat de problematiek van testbaarheid zich stelt. Het is vanuit deze optiek ook dat de specifieke ontwerptechnieken voor verbeterde testbaarheid moeten gezien worden.

a TEST GENERATIE

Een test die toelaat een fout α in een digitale schakeling te detekteren, bestaat uit een ingangspatroon (combinatorisch) of een ingangssekwentie (sekwentiële schakeling), die een verschillende response geeft voor een goed en een defekt systeem.
Principieel zorgt de ingangskombinatie ervoor dat ter hoogte van de fout een fout-effekt gegenereerd wordt (verschil goed / defekt systeem) dat via een geschikt pad naar een PO gepropageerd wordt :



① FOUT MODEL

Om de technieken van testgeneratie formeel te beschrijven, wordt gebruik gemaakt van een gepast foutmodel.
Het model dat meestal gebruikt wordt is het STUCK-AT foutmodel (1959, Eldred).
Principieel neemt het model aan dat een knoop van de schakeling (in of uitgang van een logisch element) een vaste waarde aanneemt, onafhankelijk van de waarde die wordt aangelegd:

1 : STUCK-AT-1 (S-a-1)

0 : STUCK-AT-0 (S-a-0).

De aanwezigheid van een stuck-at fout wijzigt de logische functie van de schakeling. Voorbeeld:

Sa1
a
b
c

$f = a.b.c$ foutvrije schakeling

$f_\alpha = b.c$ defekte schakeling.

Het stuck-at model beschrijft de fysische fout (meestal kortsluiting, open keten) als een 'logische' fout, waardoor de foutanalyse een logisch probleem wordt dat technologie onafhankelijk is.
Voor een schakeling met k knooppunten zijn er $2k$ mogelijke 'enkelvoudige' fouten (= foutlijst) Indien meerdere stuck-at fouten gelijktijdig kunnen optreden, dan moeten er $3^k - 1$ meervoudige fouten worden beschouwd. Om het testgeneratieprobleem zo eenvoudig mogelijk te houden wordt verondersteld dat enkel single stuck-at fouten optreden

In de praktijk blijkt dat testen voor enkelvoudige fouten ook vrij goed zijn voor de detektie van meervoudige fouten, maar niet voor de lokalisatie van meervoudige fouten.
Het stuck-at model dat gebruikt wordt heeft volgende eigenschappen:

logische fout	(↔ parametrisch)	: technologie onafh.
enkelvoudig	(↔ meervoudig)	: 2k fouten
permanent	(↔ tijdelijk, patroongevoelig).	: konstant.

(Testen ontwikkeld voor logische fouten zijn ook bruikbaar voor fysische fouten waarvan het effekt op het gedrag van de schakeling onvoldoende begrepen is of te komplex is)

② TESTGENERATIE VOOR KOMBINATORISCHE SCHAKELINGEN

Voor een kombinatorische schakeling die de funktie $f(x_1, x_2, \dots x_n)$ realiseert, zal de logische fout α de funktie wijzigen tot $f_\alpha(x_1, x_2 \dots x_n)$

FOUT-VRIJ — $f(x_1, x_2, \dots x_n)$

FOUT α — $f_\alpha(x_1, x_2, \dots x_n)$

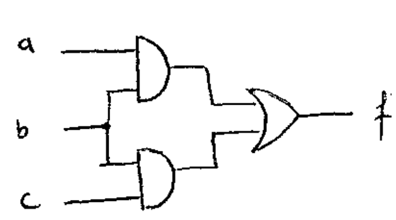
De set testvektoren T_α die de fout α DETEKTEERT voldoet aan de vgl:

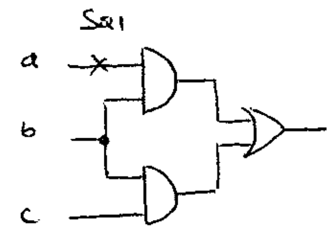
$$f \neq f_\alpha \quad \Rightarrow \quad T_\alpha = f(TVI) \oplus f_\alpha(TVI) = 1$$

De set testvektoren $T_{\alpha\beta}$ LOKALISEERT de fout α als

$$f_\alpha \neq f_\beta \ \forall \beta \quad \Rightarrow \quad T_{\alpha\beta} = f_\alpha(TVI) \oplus f_\beta(TVI) \quad \forall \beta$$

We passen deze definities toe op een eenvoudig voorbeeld:





$f = a.b + bc$ $\qquad\qquad$ $f_\alpha = b + bc = b$

De set testvectoren die de fout α detecteert:

$$T_\alpha = f \oplus f_\alpha = 1$$

$$= (a.b + bc) \oplus b = (a+c).b \oplus b$$

$$= \overline{(a+c).b} . b + (a+c).b.\bar{b}$$

$$= (\bar{a}.\bar{c} + \not{\bar{b}}).b$$

$$= \bar{a} b \bar{c} = 1 \quad \Rightarrow \quad (a,b,c) = (0,1,0)$$

Er is dus slechts 1 testvector die deze fout detecteert. In onderstaande figuur zijn de waarden van de knopen voor de foutvrije en defecte schakeling weergegeven als deze testvector wordt aangelegd:

GENERATIE fout-effect $\quad$ 0/1 $\quad$ PROPAGATIE fout-effect

0/1 $\quad$ 0/1

1

0 $\quad$ 0

foutvrij / defect

De testvector zorgt ervoor dat ter hoogte van de fysische fout een fout-effect wordt gegenereerd en via een gevoelig pad naar de uitgang wordt gepropageerd.

2.1. BOOLEAANS VERSCHIL

Uitgaande van de algemene definitie voor een set van testvectoren T_α die een fout α detecteert, wordt op FORMELE wijze deze set T_α bepaald.
De methode van het booleaans verschil wordt eerst afgeleid voor het bijzonder geval waarvoor een fout optreedt op een ingangsknoop van de logische schakeling. Nadien wordt de schakeling uitgebreid voor interne knopen.

2.1.a. INGANGSKNOOP

Beschouw een kombinatorische schakeling die de funktie $f(x_1, \dots x_n)$ realiseert. Een stuck-at fout op de ingang x_i wijzigt de gerealiseerde funktie:

$$x_i \text{ Sa1} \quad : \quad f_i(1) = f(x_1, x_2, \dots, 1, \dots x_n)$$
$$x_i \text{ Sa0} \quad : \quad f_i(0) = f(x_1, x_2, \dots, 0, \dots x_n)$$

De set testvectoren die de fout α detecteert $(x_i \text{ Sa1})$:

$$\begin{aligned}
T_\alpha = f \oplus f_\alpha &= f \oplus f_i(1) \\
&= [x_i f_i(1) + \bar{x}_i f_i(0)] \oplus f_i(1) \qquad \text{SHANNON} \\
&= [\bar{x}_i + \cancel{\bar{f}_i(1)}] . [\cancel{x_i} + \bar{f}_i(0)] . f_i(1) \\
&\quad + [\cancel{x_i f_i(1)} + \bar{x}_i f_i(0)] . \bar{f}_i(1) \\
&= \bar{x}_i \bar{f}_i(0) . f_i(1) + \bar{x}_i f_i(0) . \bar{f}_i(1) \\
&= \bar{x}_i [f_i(0) \oplus f_i(1)]
\end{aligned}$$

Het booleaans verschil wordt gedefinieerd als:

$$\boxed{\frac{df}{dx_i} = f_i(0) \oplus f_i(1)}$$

en geeft alle ingangskombinaties aan (zonder x_i te specifiëren), waarvoor de waarde van f rechtstreeks gevoelig is voor de waarde van x_i!
(f varieert als x_i varieert voor de aangegeven ingangskombinaties)
De fysische betekenis van de set testvektoren:

x_i Sa1 $$T_\alpha = \overline{x_i} \cdot \frac{df}{dx_i}$$

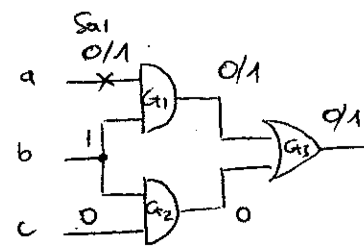
GENEREER fout-effekt
Zet ingang x_i op de TEGENGESTELDE waarde van de stuck-at fout

PROPAGEER fout-effekt.
Realiseer een gevoelig pad van x_i tot aan de uitgang om het fout-effekt aan de uitgang zichtbaar te maken.

De set testvektoren die een stuck-at fout op de ingang van een kombinatorische funktie f detekteert, kan formeel worden bepaald met het booleaans verschil:

x_i	Sa1	:	$T_\alpha = \overline{x_i} \frac{df}{dx_i}$
x_i	Sa0	:	$T_\alpha = x_i \frac{df}{dx_i}$

Voorbeeld:



$f = ab + bc$

a : Sa1 : $T_\alpha = \bar{a}\,\frac{df}{da}$

$= \bar{a} \cdot b\bar{c}$

1 (GENERATIE) ↑ (PROPAGATIE)

$\frac{df}{da} = f_a(0) \oplus f_a(1)$

$= bc \oplus b$

$= (\bar{b} + \bar{c})\,b + \text{~~}bc\bar{b}\text{~~}$

$= b\bar{c}$

De propagatievoorwaarde (gevoelig pad) wordt gegeven door het booleaans verschil en wordt bepaald voor enkele eenvoudige logische poorten.

(N)AND	(N)OR	X(N)OR
a ×, b 1, c 1		
$f = abc$	$f = a+b+c$	$f = a \oplus b \oplus c$
$\frac{df}{da} = 0 \oplus bc = bc$	$\frac{df}{da} = b+c \oplus 1 = \bar{b}.\bar{c}$	$\frac{df}{da} = b\oplus c \oplus \overline{b \oplus c} = 1$!
Alle overige ingangen 1	Alle overige ingangen 0	Steeds propagatie!!

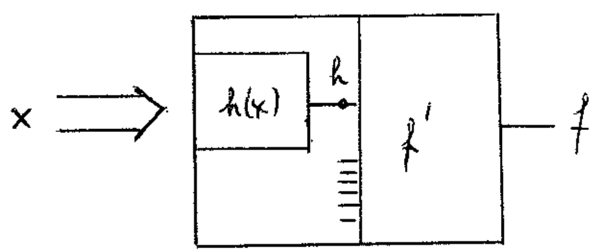
De poorten G_1 en G_3 langs het gevoelig pad staan dus allen in propagatie-mode.

2.1.b. INTERNE KNOOP

Beschouw een kombinatorische schakeling met ingangen x en interne knoop h die de funktie $f(x_1, x_2, \dots x_n)$ realiseert.
De knoop h kan uitgedrukt worden in funktie van de ingangen : $h(x)$.
De funktie f kan worden geschreven als een funktie f' van x en h :

$$f \rightarrow f'(h, x)$$

h = input van f'



De set testvektoren die een fout op knoop h detekteert wordt gegeven door de booleaanse vergelijking:

h	Sa 1	:	$T_{h1} = \bar{h} \cdot \frac{df'}{dh}$
h	Sa 0	:	$T_{h0} = h \cdot \frac{df'}{dh}$

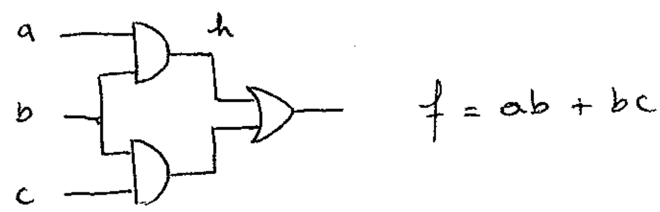
De eenvoudige ketting regel kan de berekening van het booleaans verschil vereenvoudigen voor komplexe schakelingen

$x_1, \dots x_n = x$ → h(x) → h → f' (met ingang y) → $f(x,y) = f'(h(x), y)$

$$\frac{df}{dx_i} = \frac{df'}{dh} \cdot \frac{dh}{dx_i}$$

x en y hebben geen variabelen gemeenschappelijk.

Voorbeeld



$$f = ab + bc$$

$$h = a.b \qquad f' = h + bc$$

De propagatie voorwaarde om een fout-effect aan de knoop h door te geven naar de uitgang h, wordt gegeven door het booleaans verschil:

$$\frac{df'}{dh} = bc \oplus 1 = \bar{b} + \bar{c} \longrightarrow$$

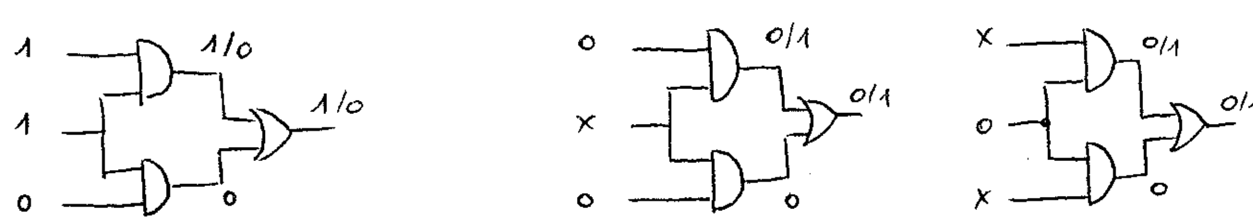
De set testvektoren voor detektie van een fout op knoop h:

$$h \text{ Sa}0 \quad : T_{h_0} = h.(\bar{b}+\bar{c}) = ab(\bar{b}+\bar{c}) = ab\bar{c}$$

$$h \text{ Sa}1 \quad : T_{h_1} = \bar{h}.(\bar{b}+\bar{c}) = (\bar{a}+\bar{b}).(\bar{b}+\bar{c})$$

$$= \bar{a}\bar{b} + \bar{b} + \bar{a}\bar{c} + \bar{b}\bar{c} = \bar{a}\bar{c} + \bar{b}$$

h Sa0 h Sa1



De methode van het booleaans verschil genereert alle testen om een gegeven fout te detekteren. Voor grote schakelingen worden de vergelijkingen echter onaanvaardbaar komplex (cpu tijd, geheugen).

2.2 D - ALGORITME

De methode van het booleaans verschil is een formele (funktionele) methode. Ze gaat uit van de booleaanse vergelijkingen van de schakeling om testen te genereren.
Het D-algoritme gaat uit van de netlijst op poortniveau (topologische beschrijving) om ALGORITMISCH testen te genereren (→ programma)
De testgeneratie methode voor de detektie van een fout α in een kombinatorische schakeling, topologisch beschreven, bestaat principieel uit drie elementaire akties:

(1) FOUT-EFFEKT GENERATIE

Ter hoogte van de fout moet een logische waarde worden gegenereerd, tegengesteld aan de waarde van de fout (0 voor Sa1, 1 voor Sa0 → $\bar{j}$ voor Sa j)

(2) FOUT-EFFEKT PROPAGATIE

Er wordt een pad geselekteerd dat de foutlokatie met de uitgang verbindt. De bijhorende signaalniveau's worden gespecifieerd opdat het fout-effekt tot aan de uitgang propageert (gevoelig pad = path sensitization)

(3) IMPLIKATIE - LIJN JUSTIFIKATIE

De specifikatie van signaalniveau's voor fout-effektgeneratie en propagatie impliceert dat ook voor andere knopen de signaalniveau's eenduidig vastliggen (implikatie).
De test voor foutdetektie moet gespecifeerd worden op de ingangen van de schakelingen (lijn justifikatie), deze produceren de signaalniveau's onder (1) en (2) gespecifieerd.

SYMBOOL D : FOUT-EFFECT

Het fout-effekt wordt verder voorgesteld door het symbool D, dat een koppel van 2 waarden is :

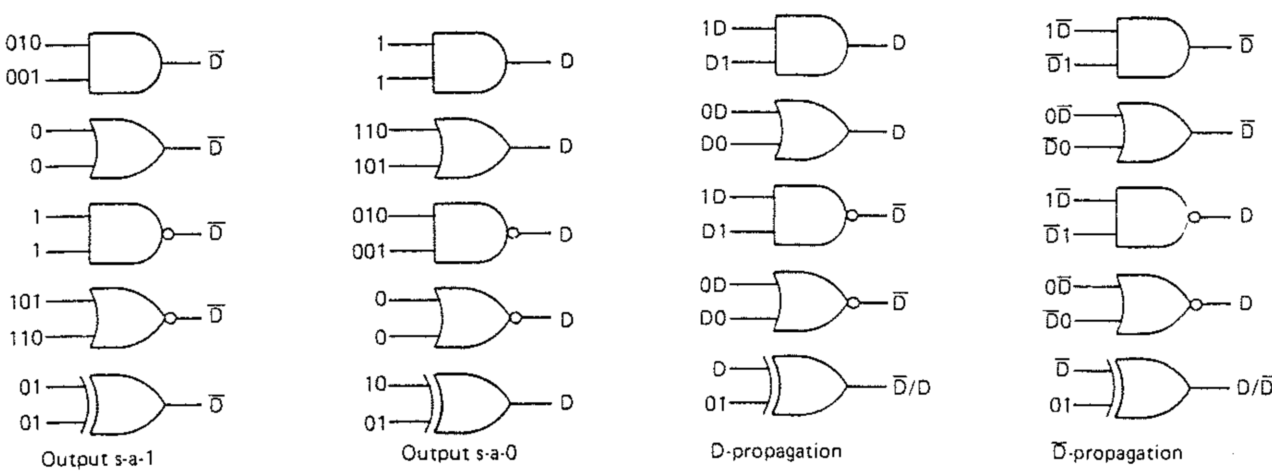
	fout-vrije schakeling	defekte schakeling	ter hoogte van foutknoop
D	1	0	Sa 0
$\overline{D}$	0	1	Sa 1

Eigenschap : D is een booleaanse variabele

$$D + \overline{D} = D + 1 = \overline{D} + 1 = 1 \qquad D.D = D+D = D.1 = D+0 = D$$

$$D.\overline{D} = D.0 = \overline{D}.0 = 0 \qquad \overline{D}.\overline{D} = \overline{D}+\overline{D} = \overline{D}.1 = \overline{D}+0 = \overline{D}$$

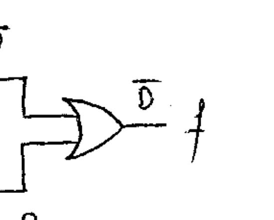
In onderstaande figuur worden de testkondities weergegeven voor generatie en propagatie van het fout-effekt (D, $\overline{D}$) bij elementaire poorten



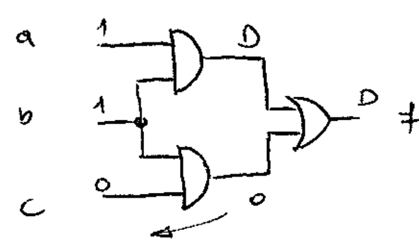
(a) test conditions for initial faults (b) conditions for D propagation

Voorbeeld.

h Sa1



h Sa0



Testgeneratie voor fout h Sa1:

(1) GENERATIE : $h = a.b = 0$ of $\bar{a} + \bar{b} = 1$ (0,x) (x,0)

(2) PROPAGATIE : $i = 0$ → fouteffekt van h naar f.

(3) IMPLIKATIE : $i = 0 \Rightarrow b.c = 0$ of $\bar{b} + \bar{c} = 1$ (0,x) (x,0)

LIJN JUSTIFIKATIE : (a, b, c)

(0, x)
(x, 0)
(0, x)
(x, 0)

(0,0,x) (x,0,x)

(0,x,0) (x,0,0)

formeel: $(\bar{a} + \bar{b}).(\bar{b} + \bar{c}) = \bar{a}\bar{c} + \bar{b}$

Testgeneratie voor fout h Sa0

(1) GENERATIE : $h = a.b = 1$ (1,1)

(2) PROPAGATIE : $i = 0$ → fout-effekt van h naar f.

(3) IMPLIKATIE : $i = 0 \Rightarrow b.c = 0$ of $\bar{b} + \bar{c} = 1$ (0,x) (x,0)

LIJN JUSTIFIKATIE : (a, b, c)

(1, 1)
(0, x) → (1, ?, 0) : inkonsistent.
(x, 0) → (1, 1, 0)

formeel : $(a.b).(\bar{b} + \bar{c}) = ab\bar{c}$

BACKTRACK

Vaak zijn er meerdere keuzemogelijkheden voor fout-effekt generatie, propagatie en lijn-justifikatie. Sommige keuzes lopen echter vast (inkonsistentie, verdwijnen fouteffekt), er moet dan worden teruggekeerd tot de laatst genomen keuze om een ander alternatief te kiezen = backtracking.

- Inkonsistentie

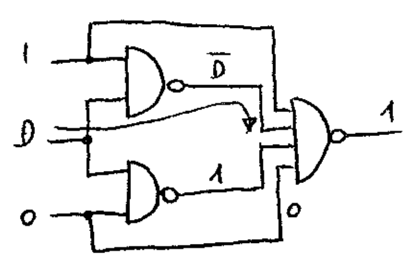
  Niet alle keuzemogelijkheden zijn konsistent met reeds gespecifieerde signaalwaarden. (Wanneer aan eenzelfde knoop tegengestelde waarden worden opgelegd) Vb (a,b) = (1,1) en (b,c) = (0,x)

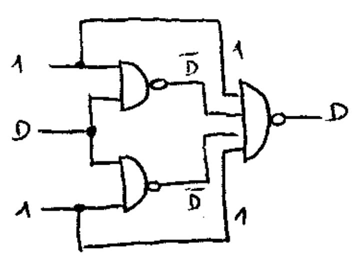
- Verdwijnen fout-effekt

  - Fout-effekten die zich langs meervoudige paden propageren, kunnen zich op sommige plaatsen opheffen (Vb reconvergente fanout)

(Deze schakeling is niet geminimaliseerd, wat vanuit het standpunt van testbaarheid niet aan te raden is.

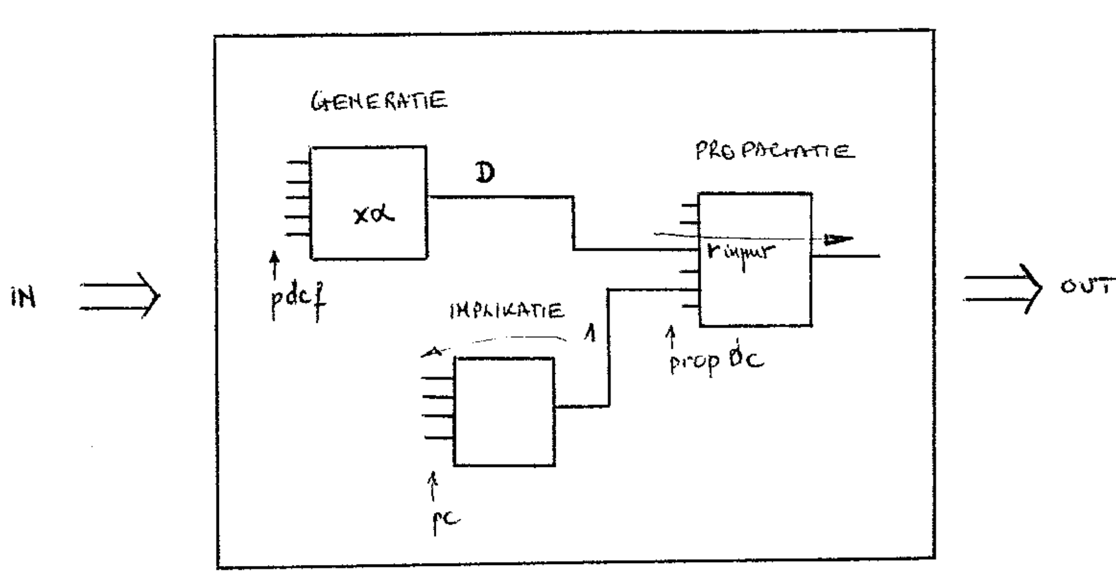
  - Soms moeten meerdere paden geaktiveerd worden om het fout-effekt tot aan de uitgang te propageren.





Door gebruik te maken van <u>kubische algebra</u>, kan de methode worden geformaliseerd voor automatische testgeneratie (ATG). Dit laat ook toe de methode te gebruiken voor meer komplexe logische elementen en meer komplexe foutmodellen.

De te testen schakeling is algemeen opgebouwd uit een aantal min of meer komplexe logische elementen.



De procedure voor testgeneratie bestaat essentieel uit 3 aktiviteiten: D-generatie, D-propagatie en implikatie. Voor elk element E wordt het gedrag voor deze aktiviteiten met 3 kubussen beschreven:

IMPLIKATIE → pc : primitive cube

Funktioneel verband tussen ingang en uitgang voor foutvrij element E (min no!)

GENERATIE → pdcf : primitive D-cube of a logic fault

Minimum input voorwaarde om $D(\bar{D})$ op de uitgang van een defekt element E te genereren. Voor elke fout een andere pdcf.

PROPAGATIE → prop dc : propagation D-cube

Minimum input voorwaarde om $D(\bar{D})$ op ingang(en) tot aan de uitgang van het foutvrij element E te propageren. Voor elke ingang andere prop DC

Elke primitieve kubus stelt de 'minimum' voorwaarde voor om een bepaalde output (0, 1, D, $\bar{D}$) te realiseren. Elke voorwaarde wordt voorgesteld als een lijn met alle knoopwaarden (positionele notatie)

	x_1	x_2	...	x_n	f	← knoopnaam
kubus:	1	x		0	1	← knoopwaarde

PRIMITIVE CUBE (PC)

De primitieve kubus stelt de priem implikanten (min. ingangsvw) voor van f én $\bar{f}$ (f = kombinatorische funktie)

	x_1	x_2	...	x_n	f
β_1	0	x	--	1	1
	1	0		x	1
β_0	1	1		x	0
	0	x		1	0

foutloos element E

	x_1	x_2	...	x_n	f_α
α_1	0	1		x	1
	0	0		x	1
α_0	x	x		1	0
	1	1		x	0

defekt element E (fout α)

De primitieve kubus voor f is nodig voor de implikatie-aktie, deze voor f_α is enkel nodig voor het opstellen van pdcf.

PRIMITIVE D-CUBE OF A LOGIC FAULT α (pdcf)

De pdcf geeft de minimum inputvoorwaarde die aan het element E moet worden aangelegd om het fout-effekt aan de uitgang te produceren. Dit is eigenlijk de elementaire testgeneratie voor het element E

De inputvoorwaarden voldoen dus aan de vergelijking:

$$T_\alpha = f \oplus f_\alpha = f \cdot \overline{f_\alpha} + \overline{f} \cdot f_\alpha$$

Een ingangskonditie genereert dus een $D(\overline{D})$ output als deze bevat is in een priemimplikant van $f(\overline{f})$ én $\overline{f_\alpha}(f_\alpha)$:

$$pdcf(D) = \beta_1 \cap \alpha_0$$
$$pdcf(\overline{D}) = \beta_0 \cap \alpha_1$$

($\alpha_i \cap \beta_i$: doorsnede van 2 kubussen per knoopwaarde : $1 \cap 0 \neq \phi$!)

PROPAGATION D-CUBE (propdc)

De propdc geeft de minimum ingangsvoorwaarden om een $D(\overline{D})$ aan ingang r van het (fout-vrij) element E voort te propageren tot de uitgang. De propagatie voorwaarde wordt gegeven door het booleaans verschil:

$$\frac{df}{dr} = f_r(1) \oplus f_r(0) = f_r(1)\,\overline{f_r}(0) + f_r(0) \cdot \overline{f_r}(1)$$

Dit kan geschreven worden als:

$$\beta_1(r=1) \cap \beta_0(r=0) \Rightarrow \begin{cases} r = D & \& \quad E = D \\ r = \overline{D} & \& \quad E = \overline{D} \end{cases}$$

$$\beta_1(r=0) \cap \beta_0(r=1) \Rightarrow \begin{cases} r = D & \& \quad E = \overline{D} \\ r = \overline{D} & \& \quad E = D \end{cases}$$

De propagatie D-kubussen komen steeds in paren voor die enkel verschillen in het feit dat alle D-komponenten komplementaire waarden hebben. (Het is ook mogelijk propdc te bepalen voor $D, \overline{D}$ op meerdere ingangen gelijktijdig).

Voorbeeld

Naast de eenvoudige logische poorten wordt vaak gebruik gemaakt van komplexere logische schakelingen als bouwsteen, bv een 2-1 multiplexer.

I_0, I_1 → [0, 1] → f, with select input S

$$f = I_0\bar{S} + I_1 S$$

$$\bar{f} = (\bar{I_0}+S).(\bar{I_1}+\bar{S}) = \bar{I_0}\bar{S} + \bar{I_1}S + \bar{I_0}\bar{I_1}$$

pc

	I_0	I_1	S	f
β_1	1	x	0	1
	x	1	1	1
β_0	0	x	0	0
	x	0	1	0
	0	0	x	0

foutvrij

	I_0	I_1	S	f
α_1	1	x	0	1
α_0	0	x	x	0
	x	x	1	0

fout: I_1 Sa0

↓

I_0	I_1	S	f
x	1	1	D

	I_0	I_1	S	f
α_1	x	1	x	1
α_0	x	0	x	0

fout: S Sa1

↓

I_0	I_1	S	f
1	0	0	D
0	1	0	$\bar{D}$

pdcf

x → 0, 1 → 1 (Sa0), S = 1 → D

0 → 0, 1 → 1, S = 0 (Sa1) → $\bar{D}$

prop Dc

I_0	I_1	S	f
1	0	$\bar{D}$	D
0	1	D	D
1	0	D	$\bar{D}$
0	1	$\bar{D}$	$\bar{D}$

ingang S

I_0	I_1	S	f
D	x	0	D

ingang I_0

I_0	I_1	S	f
D	0	$\bar{D}$	D
D	1	D	D

Ingang I_0 en S < D $\bar{D}$ / D D

IMPLIKATIE

Tijdens de D-generatie en D-propagatie aktie worden bepaalde knopen gespecifieerd (door keuze pdcf, propdc). Deze specifikatie kan ook EENDUIDIG de signaalwaarde of andere knopen impliceren. De IMPLIKATIE aktie zoekt deze signaalbepaling voorwaarts en achterwaarts in de schakeling.
Wanneer een nieuwe knoop gespecifieerd wordt, dan worden alle elementen verbonden met deze knoop (die dus in aanmerking komen voor verdere implikatie) op een lijst B geplaatst.

VOORWAARTS

ACHTERWAARTS

In de TESTKUBUS tc worden de reeds gespecifieerde knopen van de schakeling bijgehouden (topologische notatie):

$$\begin{array}{lcccccccl} & k_1 & k_2 & \cdots & k_i & \cdots & k_n & & \leftarrow \text{knoopnaam} \\ tc = (& 1 & x & & 0 & \cdots & 1 &) & 0,1 : \text{bepaald.} \end{array}$$

De implikatie wordt gevoerd door een doorsnede van de testkubus tc met één primitieve kubus c_p van het betrokken element E :

$$\text{implikatie} = tc \cap c_p$$

Hierbij moeten echter de volgende regels in acht worden genomen:

VOORWAARTSE IMPLIKATIE (INPUT GESPECIFIEERD)

Als voor een bepaalde primitieve kubus C_p' alle "bepaalde" ingangen (1,0) van het element E, ook in de testkubus t_c "bepaald" zijn, dan mag de doorsnede $t_c \cap C_p'$ genomen worden.

ACHTERWAARTSE IMPLIKATIE (OUTPUT GESPECIFIEERD)

Als er één en slechts 1 primitieve kubus C_p bestaat die de uitgang realiseert, die in de testkubus gespecifieerd is, dan mag de doorsnede $t_c \cap C_p$ genomen worden
(Er mag slechts 1 priem implikant aanleiding geven tot de gespecifieerde output).

Voorbeeld

a, b, c → [NAND] → E

	a	b	c	E
β_1	0	x	x	1
	x	0	x	1
	x	x	0	1
β_0	1	1	1	0

VOORWAARTS

$t_c = (x\ 0\ 1\ x) \cap (x\ 0\ x\ 1) = (x\ 0\ 1\ 1)$

$t_c = (1\ x\ 1\ x)\ \not\cap\ (1\ 1\ 1\ 0)$

niet alle ingang die in C_p bepaald zijn zijn ook in t_c bepaald
⇒ DUBBELZINNIG

ACHTERWAARTS

$t_c = (x\ x\ x\ 0) \cap (1\ 1\ 1\ 0) = (1\ 1\ 1\ 0)$

$t_c = (x\ x\ x\ 1)$

er zijn 3 priem implikanten die deze output genereren.
⇒ DUBBELZINNIG

PROCEDURE (D-ALGORITME)

De testgeneratie voor een fout α kan worden gespecifieerd met de primitive cubes, primitive D-cubes of the fault α, propagation D-cubes en test cube.

(1) D-GENERATIE

Selekteer een pdcf voor de beschouwde fout. Deze genereert het fout-effekt $D(\bar{D})$ op de uitgang van het defekte element (Meestal kan er gekozen worden tussen verschillende pdcf's voor dezelfde fout. Als het algoritme nadien vastloopt, dan moet een nieuwe keuze gemaakt worden = backtrack)

(2) IMPLIKATIE (D-GENERATIE)

Door de keuze van een bepaalde pdcf onder (1), worden sommige knopen gespecifieerd. Deze kunnen eenduidig (ondubbelzinnig) de waarde van andere knopen impliceren. Bij inconsistency moet teruggekeerd worden tot de laatste keuze (backtrack)

(3) D-PROPAGATIE (D-DRIVE)

De D-grens bevat de set elementen die een ongespecifieerde uitgang hebben, maar waarvan een ingang het signaal $D(\bar{D})$ bevat (deze set wordt in een lijst A bewaard).

D-propagatie tracht het $D(\bar{D})$ signaal op de ingang van het element tot aan de uitgang te brengen. Dit wordt uitgevoerd door de intersektie van een propagatie D-kubus propdc met de testkubus tc : propdc ∩ tc

(4) IMPLIKATIE (D-PROPAGATIE)

De keuze van een bepaalde propdc inkan de waarde van andere knopen eenduidig impliceren.

(5) D-OUTPUT

Herhaal (3) en (4) tot het fout-effekt signaal aan de uitgang verschijnt.

(6) LIJN JUSTIFIKATIE (KONSISTENCY)

In de vorige stappen werden de uitgangen van elementen bepaald, wat niet altijd de ingangen impliceerde. De uitgangen worden nu gejustifieerd door intersektie van de testkubus met deen primitieve kubus die de gespecifieerde uitgang produceert. Implikatie wordt dan uitgevoerd.
Dit proces wordt herhaald tot alle uitgangen gejustifieerd zijn.
De waarden van de primaire ingangen (PI) geven dan de test (set als nog x aanwezig) die de gespecifieerde fout α detekteert.

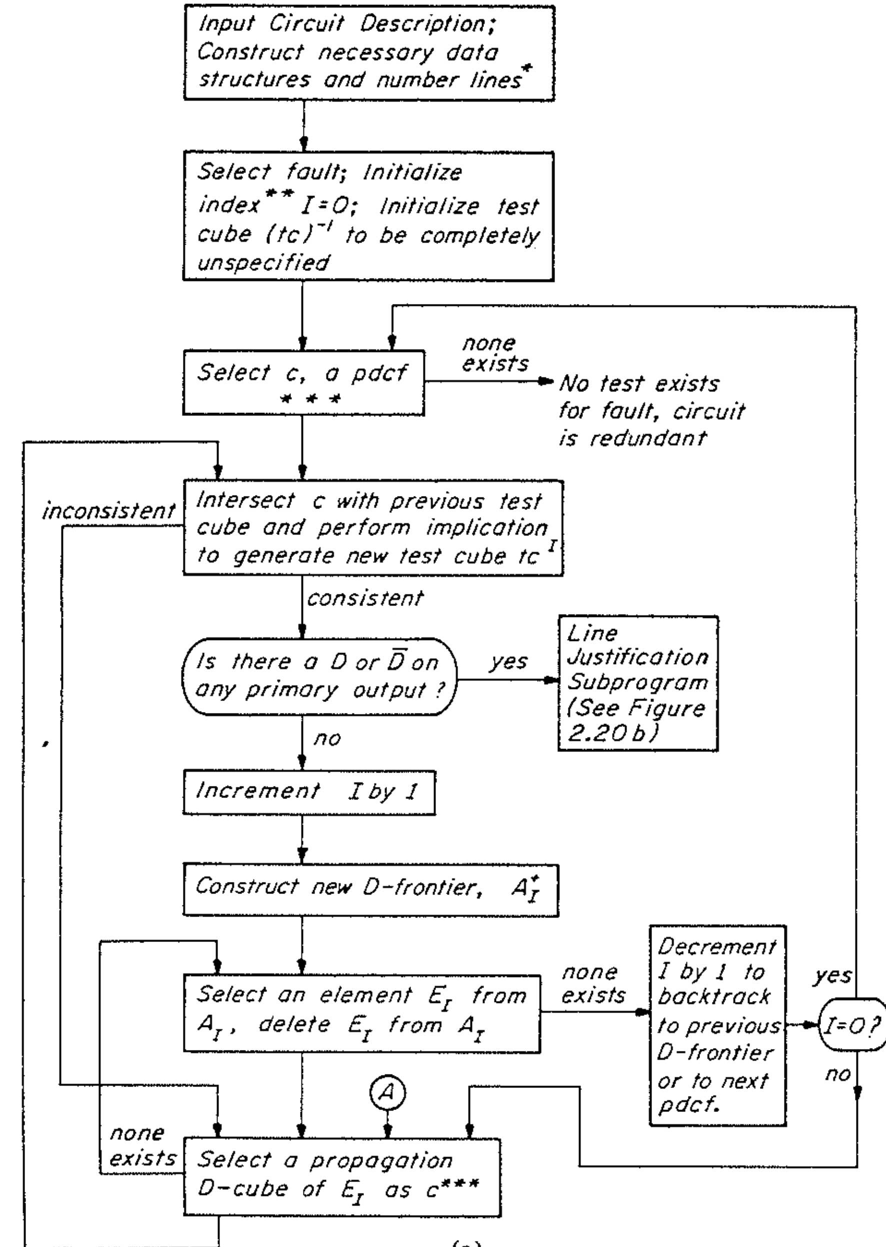


Figure 2.20 D-algorithm Flowchart

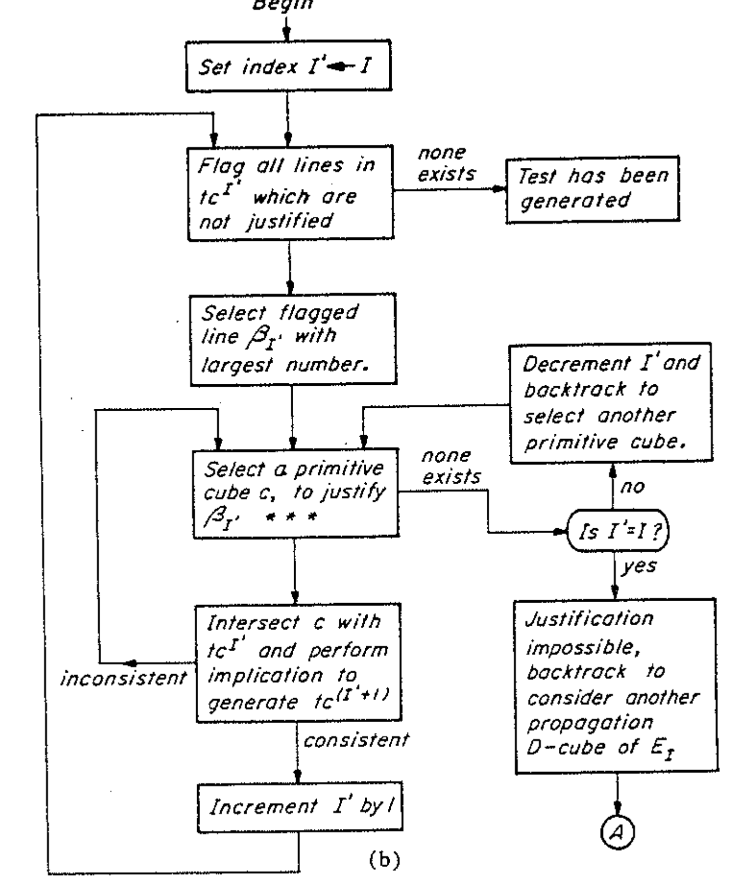


Figure 2.20 Line Justification Subprogram Flowchart

*Lines are numbered (assigned integer values) in such a way that the number associated with the output of a gate is larger than the number associated with any of the input lines of that gate. This is done to facilitate line justification.

**The index I is used to keep track of the different test cubes and D-frontiers generated during execution and to enable backtracking.

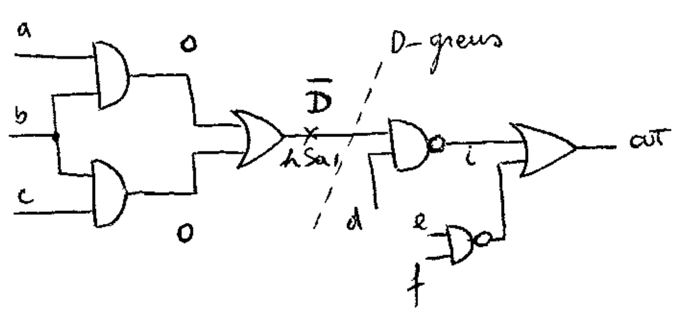
***Indicates sequential consideration of a set of possible choices which can be represented as a list or stack but typically are generated dynamically.

†The new D-frontier A_{I+1} can be constructed from the previous D-frontier A_I as follows: if we have just propagated through a_I, then

$$A_{I+1} = A_I - a_I \cup \{\text{all elements fed by } a_I \text{ which have unspecified outputs}\}.$$

This is only valid if implication of D and $\overline{D}$ values is not performed.

Voorbeeld

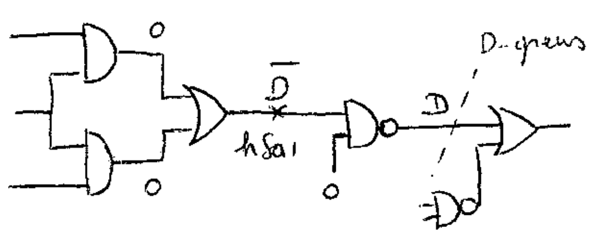


(1) D-GENERATIE

fout h Sa1 ⇒ $\overline{D}$ op h

(2) IMPLIKATIE

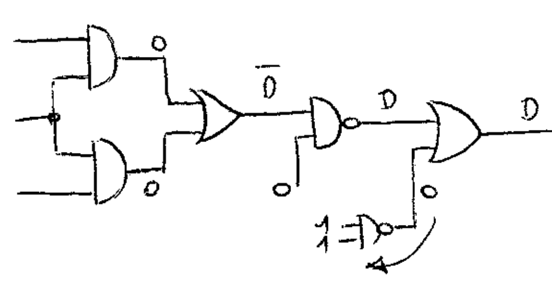
niet eenduidig.



(3) D-PROPAGATIE

(4) IMPLIKATIE

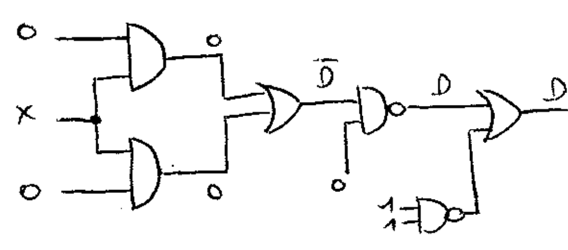
geen elementen



(3)' D-PROPAGATIE

(4)' IMPLIKATIE

éénduidig : e = f = 1



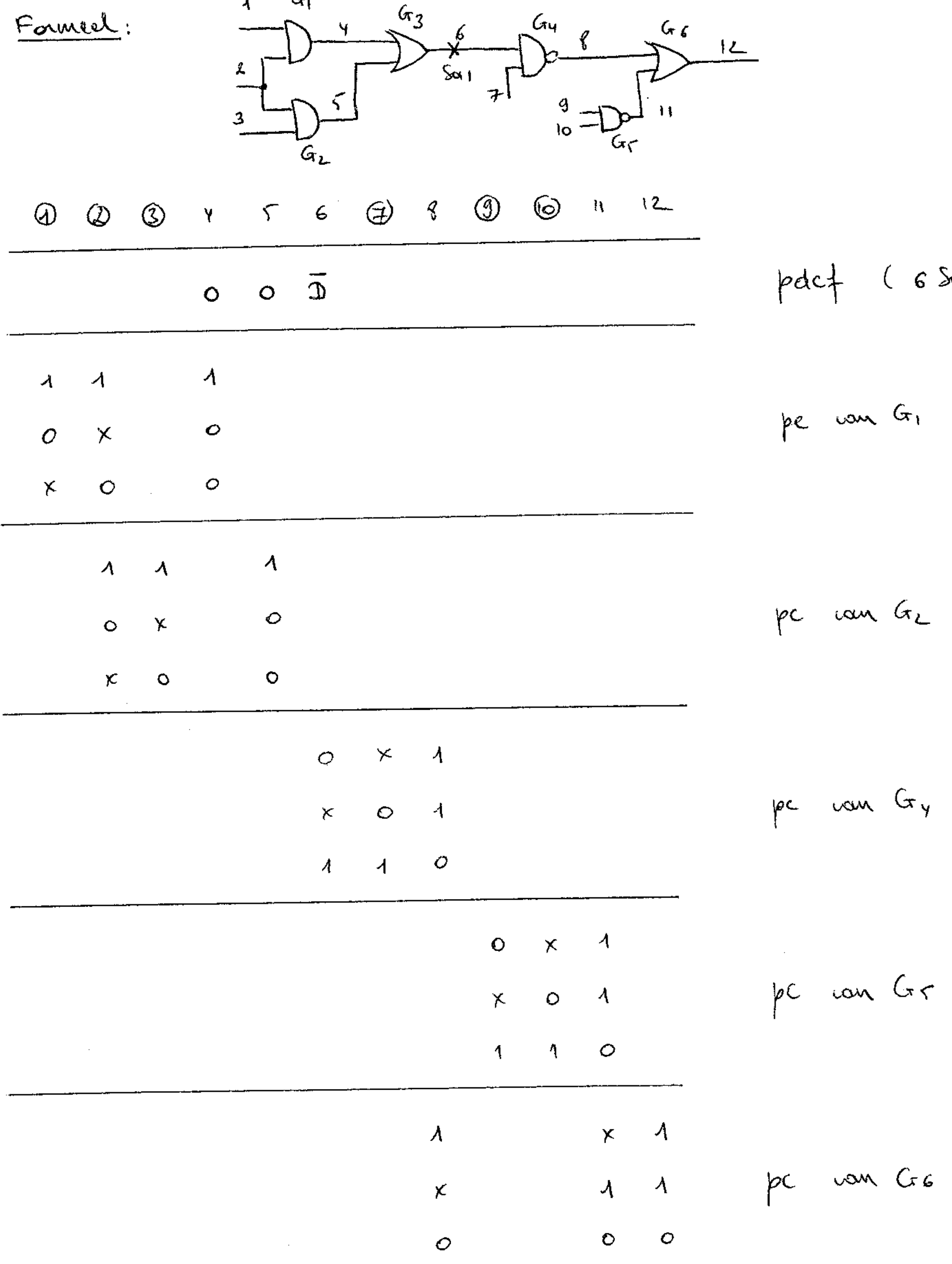
(5) JUSTIFIKATIE

Test : (a, b, c, d, e, f) = (0, x, 0, 0, 1, 1)

Fouten :
- h Sa1
- i Sa0
- out Sa0

} schrappen in foutenlijst (alle knopen gelegen op het gevoelig pad!)

Formeel:



1	2	3	4	5	6	7	8	9	10	11	12	
			0	0	$\overline{D}$							pdcf (6 Sa1) – G_3
1	1		1									pc van G_1
0	x		0									
x	0		0									
	1	1		1								pc van G_2
	0	x		0								
	x	0		0								
					0	x	1					pc van G_4
					x	0	1					
					1	1	0					
								0	x	1		pc van G_5
								x	0	1		
								1	1	0		
							1			x	1	pc van G_6
							x			1	1	
							0			0	0	

①	②	③	4	5	6	⑦	8	⑨	⑩	11	12	
			0	0	$\overline{D}$							tc^0 = pdcf (D-GENERARE)
					$\overline{D}$	0	D					prop de G4
			0	0	$\overline{D}$	0	D					tc^1 (nu D-PROPAGARE)
							D			0	D	prop de G6
			0	0	$\overline{D}$	0	D			0	D	tc^2 (nu D-PROPAGARE)
			0	0	$\overline{D}$	0	D	1	1	0	D	(nu implicare)
0	x	0	0	0	$\overline{D}$	0	D	1	1	0	D	tc^3 (nu justificare)

test (a, b, c, d, e, f) = (1, 2, 3, 7, 9, 10) = (0, x, 0, 0, 1, 1)

2.3 FOUTREDUKTIE (FAULT COLLAPSING)

De foutlijst van te detekteren fouten kan gereduceerd worden door eigenschappen van logische poorten en topologie van het netwerk.

2.3.a. EKWIVALENTE FOUTEN

De testset die onderscheid maakt tussen twee fouten α en β wordt gegeven door:

$$T_{\alpha\beta} = f_\alpha \oplus f_\beta$$

Er bestaan geen testen die een onderscheid tussen α en β maken als

$$f_\alpha = f_\beta$$

De fouten α en β worden ekwivalent genoemd.
Indien een set fouten ekwivalent zijn, dan zal een test die één van deze fouten detekteert ze allen detekteren.

n-input poort : 2(n+1) mogelijke fouten, n+1 ekwivalente fouten } => n+2 fouten testen

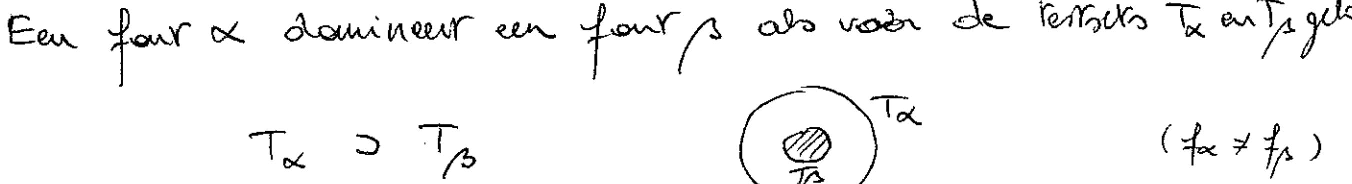
AND — NAND — OR — NOR

AND: Sa0 (inputs), Sa0 (output)
NAND: Sa0 (inputs), Sa1 (output)
OR: Sa1 (inputs), Sa1 (output)
NOR: Sa1 (inputs), Sa0 (output)

Het volstaat een test te genereren voor één fout van een set ekwivalente fouten

2.3.b. DOMINANTE FOUTEN

Een fout α domineert een fout β als voor de testsets T_α en T_β geldt:



Indien de fout β door een test gedetecteerd wordt, wordt ook de fout α gedetecteerd (voor α moet geen expliciete test gegenereerd worden).

Sa1 β — AND — α Sa1 | Sa1 — NAND — Sa0 | Sa0 — OR — Sa0 | Sa0 — NOR — Sa1

$T_\alpha = \{10; 01\}$

$T_\beta = \{01\}$

De dominante stuck-at fout aan de uitgang (Sa1 : AND, NOR, Sa0 : NAND, OR) moet dus niet gedetecteerd worden.

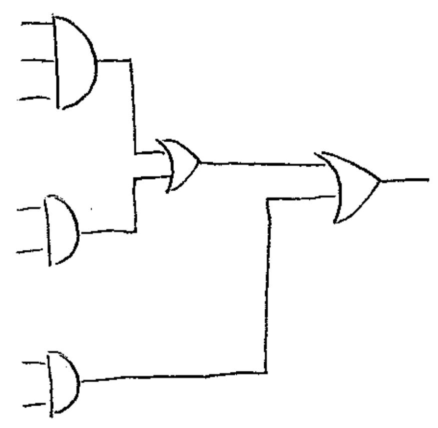
Gebruik makende van fout ekwivalentie en fout dominantie volstaat het n+1 fouten van een n-input poort te detecteren:

AND – NAND	1 – n	: iedere ingang Sa1
	n+1	: één ingang Sa0
OR – NOR	1 – n	: iedere ingang Sa0
	n+1	: één ingang Sa1

De stuck-at fouten aan de uitgang hoeven niet getest te worden.

2.3.c. FANOUT VRIJE KOMBINATORISCHE SCHAKELING

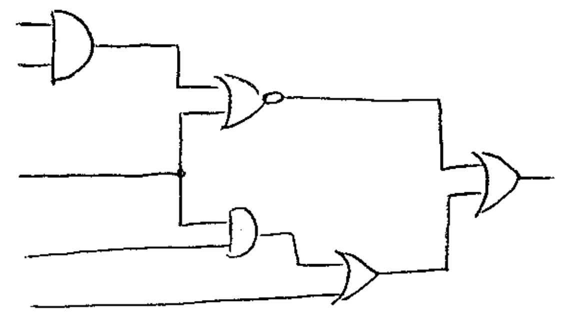
Voor elk fanout vrij netwerk (kombinatorisch) volstaat het alle stuck-at fouten op de primaire ingangen (PI) te detekteren. Dan zullen alle enkelvoudige stuck-at fouten gedetekteerd worden.



Voor elke poort volstaat het de sa fouten op de ingangen te detekteren, dan worden ook de sa fouten op de uitgangen gedetekteerd (deze zijn zelf input van slechts 1 andere poort)

2.3.d. KOMBINATORISCHE SCHAKELING

Het volstaat alle enkelvoudige stuck-at fouten op primaire ingangen en fanout takken (= checkpoints) te detekteren om alle enkelvoudige stuck-at fouten te detekteren.



5 primaire ingangen

2 fanout takken

=> 14 fouten

(Verder te reduceren door ekwivalentie / dominantie)

2.4. REDUNDANTE NETWERKEN – ONTESTBARE FOUTEN.

Een fout α is ontestbaar als geen enkele test de fout α detecteert:

$$T_\alpha = f \oplus f_\alpha = 0 \quad \Rightarrow \quad f = f_\alpha$$

De schakeling is redundant voor de fout α en kan worden vereenvoudigd:

ontestbare fout	vereenvoudiging
(N)AND ingang Sa1	verwijder ingang
(N)AND ingang Sa0	verwijder poort, vervang door 0(1)
(N)OR ingang Sa0	verwijder ingang
(N)OR ingang Sa1	verwijder poort, vervang door 1(0)

De bepaling van redundancy is moeilijk. Als er geen test bestaat voor de fout, dan is de schakeling redundant. Dit vergt echter veel rekentijd. Redundantie wordt toevallig of expliciet (maskering hazard) geïntroduceerd.

2.4.a. NIET-MINIMAAL NETWERK

Een niet-minimaal kombinatorisch netwerk is niet volledig testbaar.

a, b, c, 1, 1, 1, 1, $\bar{D}$, $\bar{D}$

$f = ab + abc = ab$

a Sa0 : 1 1 0

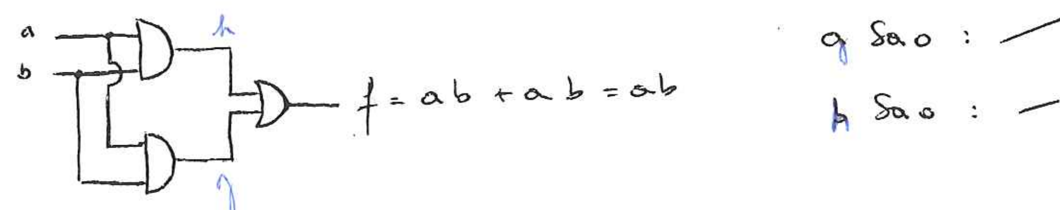
b Sa0 : 1 1 0

c Sa1 : —

De fout c Sa1 is ontestbaar. De c ingang kan weggelaten worden.

(reconvergente fan-out)

De schakeling wordt dus vereenvoudigd:

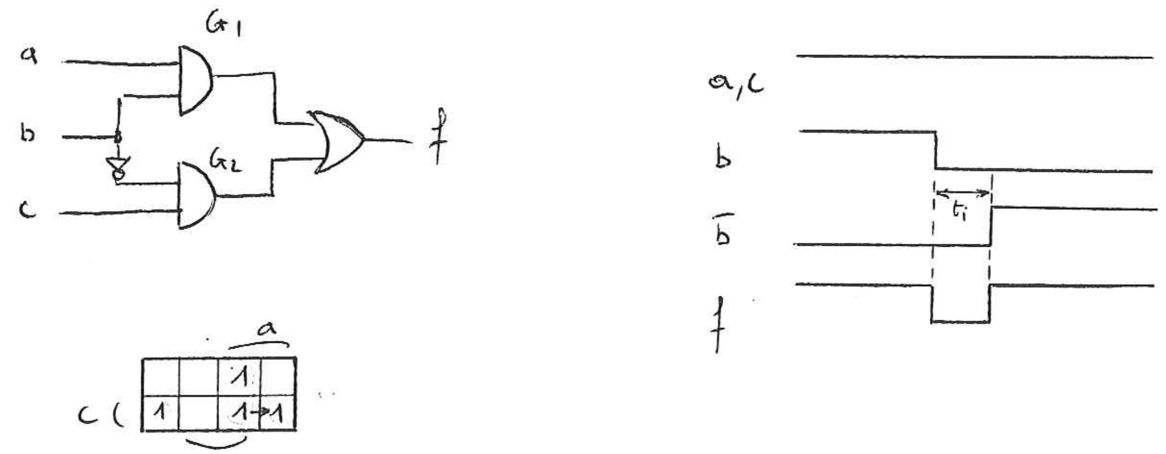


Hierdoor ontstaat 'second generation' redundantie : een nieuwe set ontestbare fouten ontstaat (g Sa0 , h Sa0). Slechts als de schakeling volledig geminimaliseerd is, wordt ze volledig testbaar.

2.4. b STATIC HAZARDS

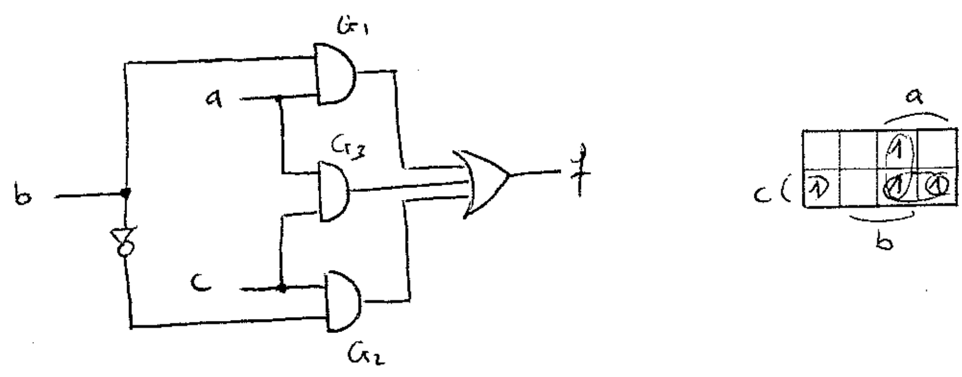
Een nadeel van minimalisatie is de mogelijke introductie van statische hazards (= 'mogelijke' glitch aan uitgang, de nodige tijdsovertragingen moeten aanwezig zijn) :

Volgende schakeling vertoont een static - 1 hazard :

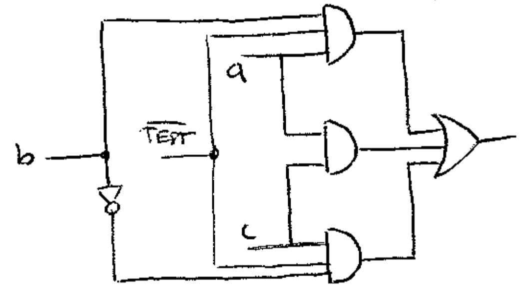


Als a en c hoog zijn en b van waarde wijzigt, dan moet de uitgang f hoog blijven. Het logisch hoog niveau wordt echter door de wijziging door een andere poort geleverd : $b : 1 \rightarrow 0$ $\quad G_1 = 1 \rightarrow G_2 = 1$

De tijdsvertraging door de invertor zorgt ervoor dat tijdelijk de uitgang van beide poorten laag is en dus f even laag wordt (hazard wordt door kritische tijdsvertraging van de invertor bepaald).
Op de Karnaugh kaart is dit zichtbaar door het niet overlappen van de mintermen G_1 en G_2. ('gap' tussen mintermen → vb EPROM)
Deze hazard kan geëlimineerd worden door een 'bridging term', een productterm die beide bestaande producttermen overlapt (bedekt 'gap').

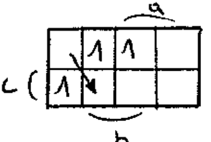


De schakeling is nu glitch-free maar wordt ontestbaar door de redundante productterm (vb G_3 : Sa0). De schakeling kan testbaar gemaakt worden door een extra observatiepunt (uit G_3) of door toevoegen van een testingang (CT)



De testingang laat in testmode toe de bridging term afzonderlijk uit te testen. De schakeling is niet redundant meer.

Hazards tgv de wijziging van één ingang kunnen steeds op deze wijze geëlimineerd worden. Voor meervoudige wijzigingen van ingangen is dit niet steeds mogelijk:



transitie (a,b,c) : (0,0,0) → (0,1,1)

→ functionele hazard

(logische hazard: wel te elimineren)

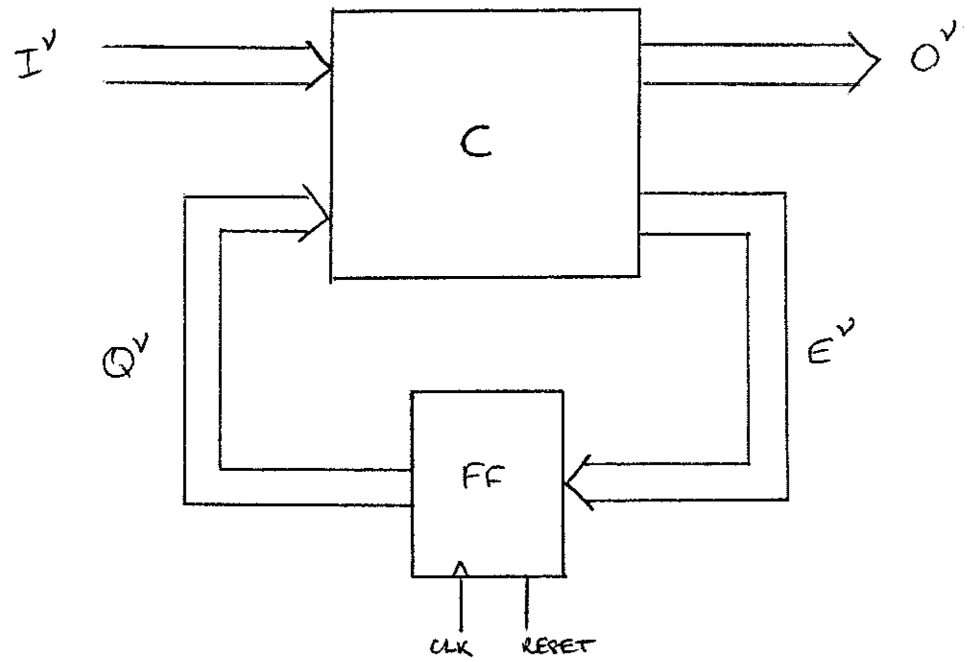
③ TESTGENERATIE VOOR SEKWENTIËLE SCHAKELINGEN.

De generatie van testen voor sekwentiële schakelingen is veel komplexer dan voor kombinatorische schakelingen:

- INPUT : testsekwentie ipv één testvektor
- OUPUT : sekwentie, afhankelijk van de initiële toestand.

De algemene voorstelling van een sekwentiële machine is een kombinatorische schakeling voorzien van een terugkoppellus met ingebouwde tijdsvertraging: synchroon : geklokte flip-flops
asynchroon : poort, bedradingsvertraging.

Daar asynchrone schakelingen moeilijk testbaar zijn, worden verder enkel synchrone schakelingen besproken.



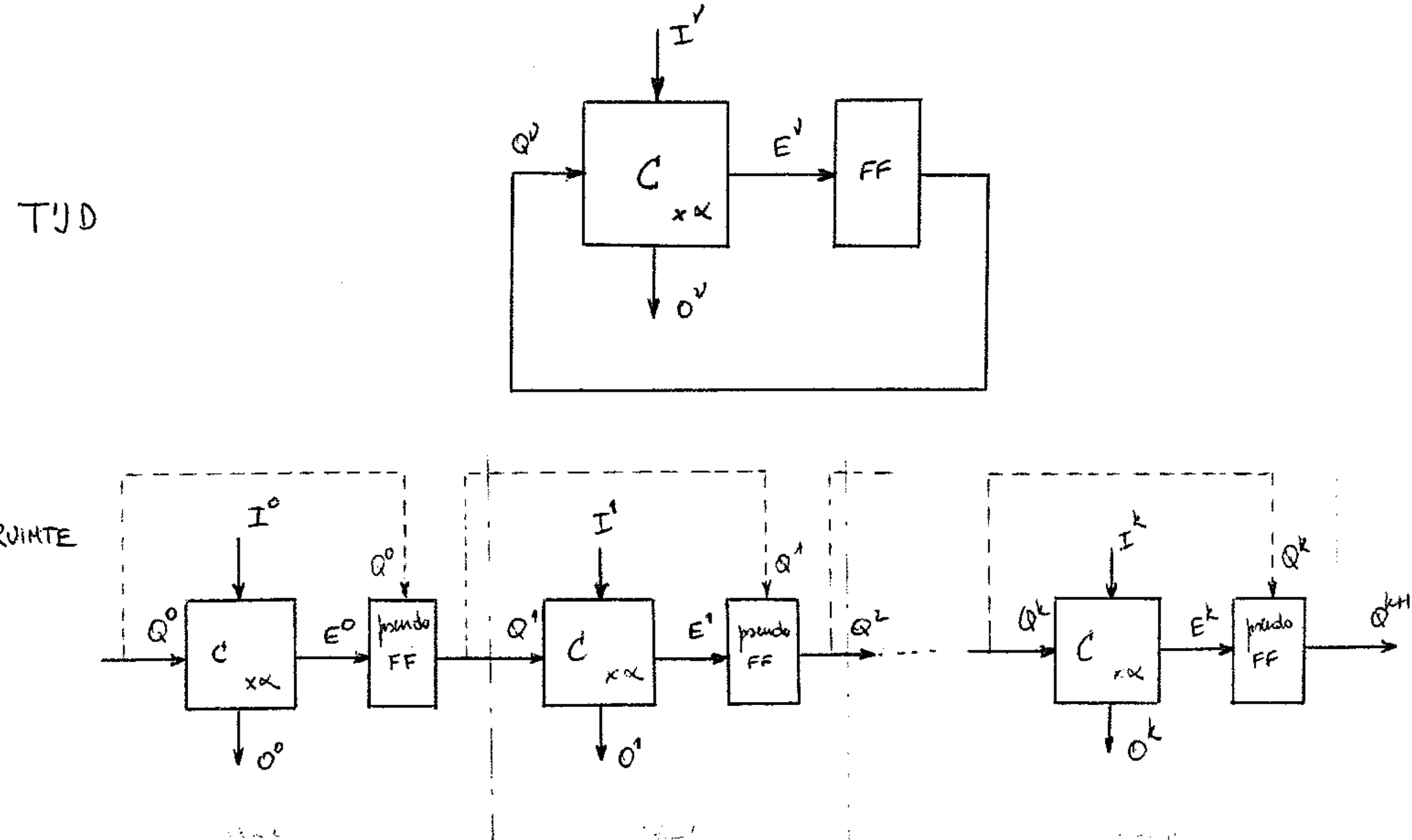
I^v : primaire ingang

O^v : primaire uitgang

Q^v : huidige toestand

E^v : excitatie volgende toestand.

Door een tijd-ruimte transformatie kan een sequentiële schakeling gemodelleerd worden door een pseudo-combinatorische schakeling.

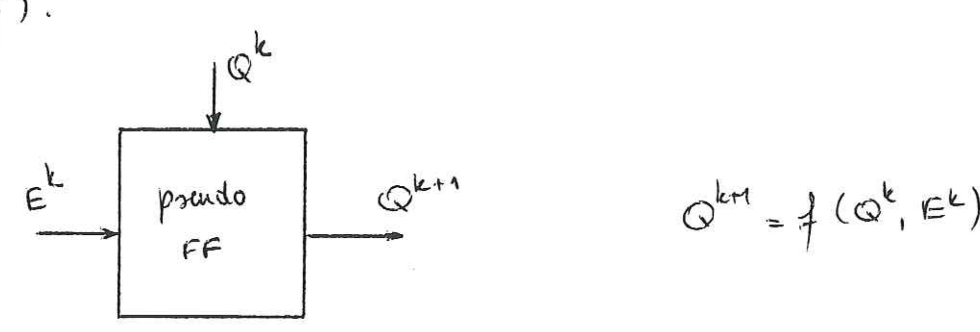


In het tijdsdomein wordt een ingangssequentie $I^0, I^1, \ldots I^k$ aangelegd aan de schakeling met initiële toestand Q^0. Deze genereert een uitgangssequentie $O^0, O^1, \ldots O^k$.

In het ruimtedomein wordt de schakeling voorgesteld door een iteratieve matrix. De ingang I^i wordt aangelegd aan cel i van de matrix. De uitgang O^i wordt door cel i gegenereerd. In de transformatie worden de geklokte flip-flops gemodelleerd door combinatorische elementen: pseudo flip-flops. De iteratieve matrix is dus combinatorisch en de testgeneratietechnieken voor combinatorische schakelingen kunnen toegepast worden. De tijdsdomein response van de sequentiële schakeling worden getransformeerd naar de ruimtedomein response van de combinatorische iteratieve matrix. Een enkelvoudige fout in het sequentiële circuit komt overeen met een meervoudige fout (zelfde fout in elke cel van de matrix)

PSEUDO FLIP-FLOP

De geklokte flip-flops worden gemodeleerd door kombinatorische elementen: pseudo flip-flops. In het algemene model van de pseudo flip-flops zijn de ingangen Q^k (huidige toestand) en E^k (excitatie). De uitgang is de volgende toestand Q^{k+1} (eventueel onderscheid met de komplementuitgangen, $\bar{y}$).



Het kombinatorisch model wordt bepaald door de next-state vergelijking van de flip flop.

D-FF: $D^k \rightarrow Q^{k+1}$ $\qquad Q^{k+1} = D^k$

T-FF: T^k, Q^k → XOR → Q^{k+1} $\qquad Q^{k+1} = Q^k \oplus T^k$

JK-FF: J^k, K^k, Q^k → AND, AND → OR → Q^{k+1} $\qquad Q^{k+1} = J^k \bar{Q}^k + \bar{K}^k Q^k$

Voor elk van deze elementen kunnen pdfs en popds opgesteld worden.

OPMERKINGEN

1. STATISCHE TEST

Er worden enkel statische testen of sequentiële machines toegepast:

- ingang aanleggen als schakeling stabiel is
- uitgang observeren als schakeling stabiel is

2. ASYNCHROON

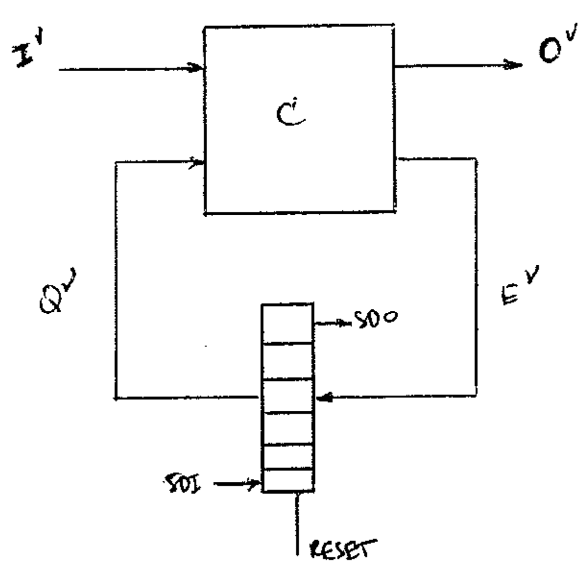
De generatie van testpatronen voor asynchrone schakelingen is zeer moeilijk:

- de werking van de schakeling hangt af van interne vertragingen, deze worden door de meeste testalgoritmes genegeerd.
- asynchrone schakelingen bevatten races (≠ inputs) en kunnen foutief werken door hazards.

Deze schakelingen worden best vermeden.

3. RESET

Voor een eenduidige testresponsie is het nodig dat de begintoestand van de schakeling rechtstreeks kontroleerbaar is. Een algemene reset voor 'alle' flip-flops is een noodzaak. De testgeneratie wordt sterk vereenvoudigd indien de resettoestand ook observeerbaar is (zie SCAN ontwerp).



PROBLEMEN

1. LENGTE TEST SEKWENTIE

De maximum sekwentie lengte voor de detektie van 1 fout is :

$$4^{FF}$$

en neemt dus exponentieel toe met het aantal geheugenelementen.
Lange testsekwenties treden vooral op door de aanwezigheid van
BEGRAVEN flip-flops :

- meeste ingangen gekontroleerd door uitgangen van andere flip-flops, niet door primaire ingangen (PI)
- meeste uitgangen gaan naar andere flip-flops en zijn niet rechtstreeks observeerbaar door primaire uitgangen (PO)

Voorbeelden : n bit schuifregister : n

n bit binaire teller : 2^n

2. LOCK-OUT STATES

Het gedrag van de lock-out preventie kan slechts geverifieerd worden als de machine eerst in deze toestand gebracht wordt. Hiervoor is een aparte kontrole van de begintoestand nodig (zie SCAN design).

b. TEST EVALUATIE (FOUTSIMULATIE)

Het gegenereerde testpatroon (testprogramma) zal gebruikt worden om fouten in geproduceerde schakelingen te detekteren (eventueel te lokaliseren). De bedoeling is zo weinig mogelijk defekte schakelingen af te leveren (klant, (sub)systeem).
De mate waarmee het testprogramma hier aan voldoet, moet worden geëvalueerd. Dit gebeurt met een foutsimulator. Deze simuleert de logische schakeling door fouten te injekteren en vergelijkt de respons met een fout-vrije simulatie.

FOUT - DETEKTIE

Een ingangssekwentie $X = x(0)x(1)\ldots x(k)$ detekteert een fout α als op een bepaald ogenblik i één van de uitgangen van het foutieve systeem verschilt van de overeenkomstige uitgang van het foutvrije systeem:

$$z_p(i) \neq z_p^\alpha(i) \qquad \text{met } z_p(i), z_p^\alpha \in \{0,1\}$$

De fout α wordt potentieel gedetekteerd als:

$$z_p(i) \neq z_p^\alpha(i) \qquad \text{met } z_p(i) \in \{0,1\} \text{ en } z_p^\alpha(i) = x$$

Een x wordt in de schakeling geïntroduceerd als een geheugenelement opstaat in een ongekende begintoestand (bv Reset ingang niet aanwezig of stuck-at fout):

1 — D Q — x
Reset
* Sa0

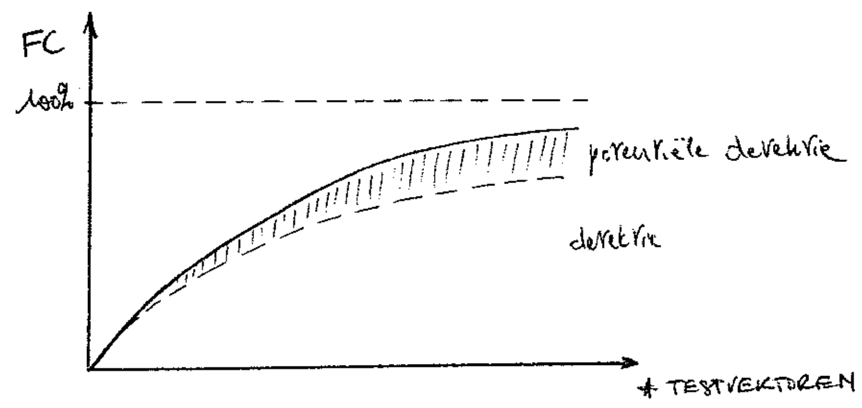
Vaak kan aan de foutsimulator aangegeven worden hoeveel maal een bepaalde fout gedetekteerd moet worden, voor ze uit de foutlijst geschrapt wordt.

FAULT COVERAGE

De fault coverage (foutdekkingsgraad) van een testprogramma wordt gedefinieerd als:

$$FC = \frac{\#\ \text{gedetekteerde fouten}}{\#\ \text{mogelijke fouten}} \times 100 \quad (\%)$$

De foutsimulator geeft een kumulatieve plot van FC in funktie van de aangelegde testvektoren. Per testvektor worden enkele extra fouten gedetekteerd:



Initieel worden veel fouten gedetekteerd. Naarmate een FC van 100% benaderd wordt, zijn er meer testvektoren nodig (knopen met lage testbaarheid: TY = CY.OY).

DEFECT LEVEL

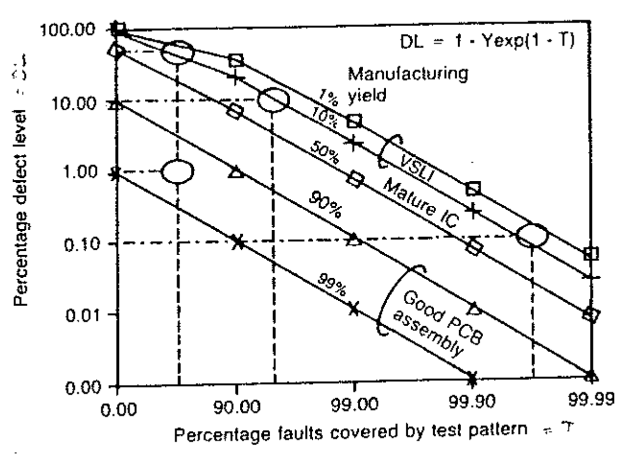
Naarmate meer testvektoren voorzien worden, wordt ook de uitvoeringstijd van het testprogramma langer en verhogen dus de produktie-testkosten (Vb: Testprogramma duur = 1 min én 1000 stuks ⇒ testduur = 1000 min > 16 uur!)

Door het feit dat niet alle mogelijke fouten gedetekteerd worden (FC < 100%) bestaat de reële kans dat defekte schakelingen door het testprogramma toch goed bevonden worden.

Voor stuck-at fouten bestaat een verband tussen yield, fault coverage en defect level, dit is een kans dat een defekte schakeling wordt goedgekeurd (% aantal defekte afgeleverde schakelingen):

$$DL = 1 - Y^{1-FC}$$

(DL = aantal defekte schakelingen, niet door de test gedetekteerd). Dit verband is in onderstaande figuren weergegeven:



Defects escaping detection at test.

DL = 1 - Y^(1-T)
Y: Yield
Y=.01
Y=.10
Y=.25
Y=.50
Y=.75
Y=.90
Y=.99
Defect Level
Fault Coverage (%)

Defect Level as a Function of Yield and Fault Coverage.

Indien geen testen uitgevoerd worden (FC=0), dan herleidt de betrekking zich tot:

$$DL = 1 - Y$$

wat ook rechtstreeks uit de definitie van yield volgt. (Met een yield van 25% worden 75% slechte produkten afgeleverd). Door extra testen uit te voeren kan het defect level gereduceerd worden. Meestal wordt een bepaald defectlevel vooropgesteld. De nodige foutdekkingsgraad FC is sterk afhankelijk van de produktie yield Y.

Voorbeeld

PCB : Y = 95% → FC = 80% : DL = 1 %

↓

VLSI : Y = 10% → FC = 80 % : DL = 37 %

FC = 99,6% : DL = 1%

VLSI komponenten gemaakt met een kleine produktie yield hebben een hoge foutdekkingsgraad nodig (veel hoger dan deze die nodig was om de PCB die vervangen werd te testen!).

Welk defect level moet gehaald worden :

- consumer products → geen grote konsekwentie DL > 1%
- VLSI memory parts → producent garandeert DL < 0.05%
- ruimtevaart, beveiliging → belangrijke konsekwentie DL < 0.05%

Telkens een defekte komponent naar een hoger testniveau wordt overgelaten (PCB, subassembly, assembly) stijgt ook de kost om het defekt te vinden. Ten slotte wordt de fout bij de klant gevonden, de kosten die dan met het defekt verbonden zijn : verloren informatie, verloren tijd, verloren vertrouwen, hoge herstellingskosten.

FOUT MODEL

In de foutsimulator werd enkel getest voor stuck-at fouten. Er kunnen in CMOS ook andere fouten optreden.

kombinatorisch netwerk wordt sekwentieel

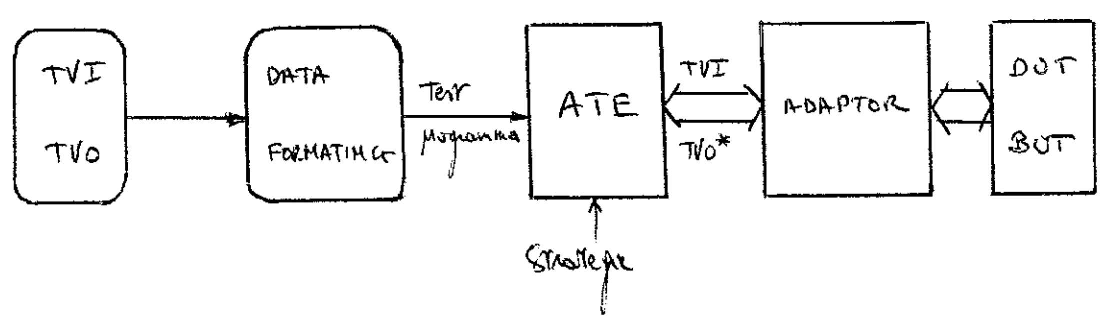
statische Dff wordt dynamische Dff

(praktisch onvertaan – $t_{delay} > 1 msec$)

De vraag blijft open of meer komplexe foutmodellen in rekening moeten gebracht worden

C TEST UITVOERING. (ATE)

De input testvektoren worden gegenereerd en met foutsimulatie geëvalueerd op foutdekking FC. De inputtestvektoren (TVI) worden gesimuleerd met een timing simulator om de overeenkomstige outputvektoren (TVO) van de foutvrije schakeling te bepalen.
Het ontwikkelde testprogramma (TVI, TVO) moet dan in het gepaste formaat omgezet worden opdat het door de ATE korrekt geïnterpreteerd zou worden.
Via een gepaste adapter kan de te testen schakeling met de ATE verbonden worden. De ATE beschikt vaak over een aantal teststrategieën.



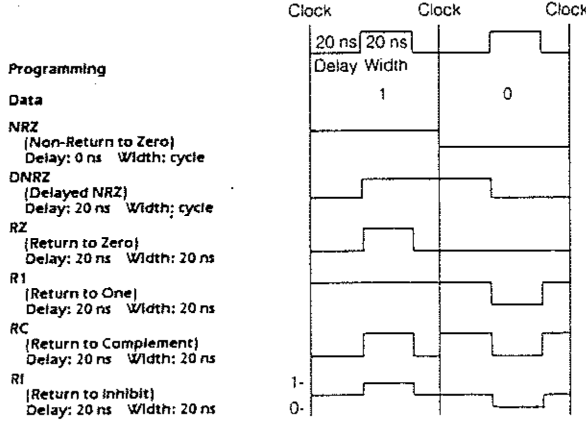
Het aantal testvektoren dat in de ATE kan worden geladen en in 1 keer uitgevoerd, wordt de patroondiepte genoemd (HP81810 : 1000 vektoren).
Deze wordt bepaald door de hoeveelheid beschikbaar geheugen. Bij Scan-ontwerp is een grote patroondiepte vereist, deze kan eventueel verhoogd worden door herkonfiguratie van het geheugen

DATAFORMAAT - TIMING

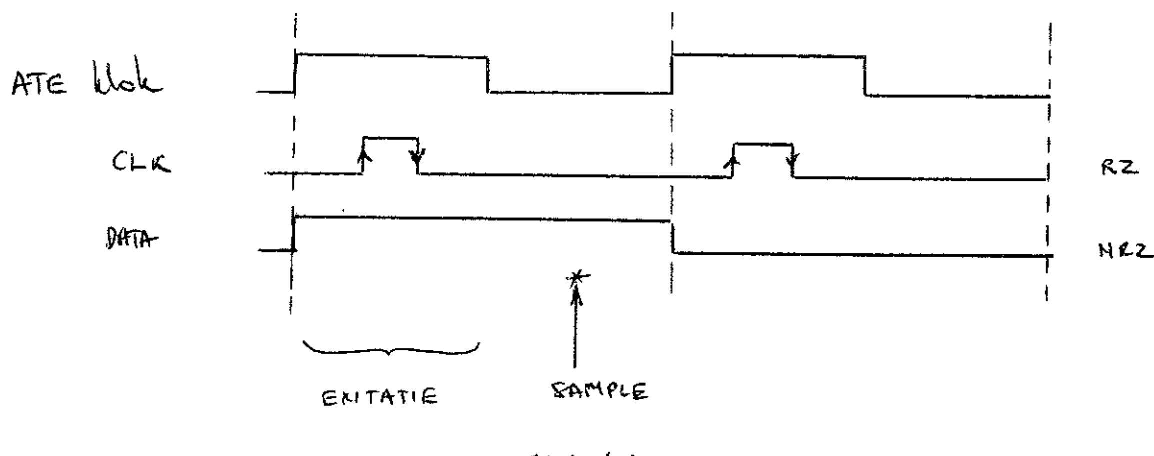
De data formatering laat een beschrijving toe van de exacte golfvorm die aan de te testen schakeling moet aangeboden worden (input)

- FORMAAT : golfvorm
- TIMING : plaatsing van flanken

Meerdere formaten worden ondersteund. Als tijdsinformatie kan een tijdsvertraging en een pulsbreedte nauwkeurig worden ingegeven.



Het sampling ogenblik van de analyser kan worden geprogrammeerd. Gedurende een testcyclus wordt eerst een exitatie door de generator uitgevoerd, daarna wordt de respons door de analyser geobserveerd.



Programmeerbare formattering wordt ook gebruikt bij de uitvoering van AC metingen.

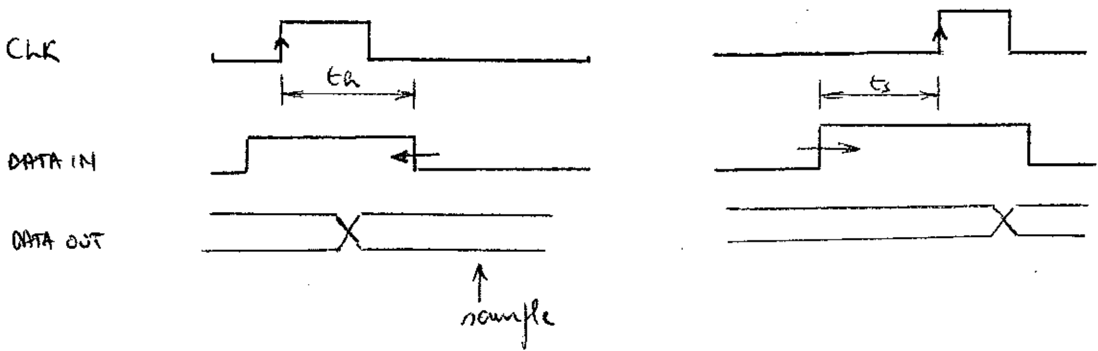
AC METING

Voor de dynamische karakterisering van de schakeling worden AC parameters opgemeten:

- t_s = set-up time
- t_h = hold time
- t_{pd} = propagation delay.

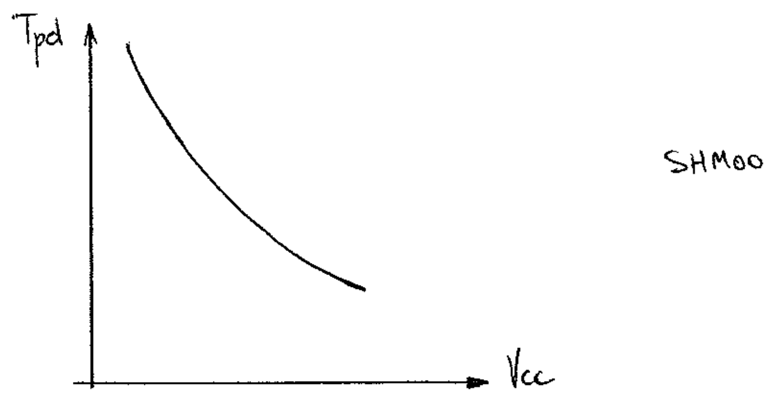
Deze metingen worden vaak uitgevoerd onder variërende kondities (spanning, temp.)

Voor de meting van t_s en t_h wordt gebruik gemaakt van de formaten RZ (clk) en RC (data input).



Voor het bepalen van AC parameters moeten meerdere tests gelopen worden (iteratief proces). De dataformaten moeten gewijzigd worden (vb Th: data in 2de flank naar aktieve klokflank van CLK bewegen tot data-out wijzigt).

Voor karakterisatie van een CMOS proces wordt vaak de tijdsvertraging uitgezet in funktie van de voedingsspanning:



SHMOO PLOT

ADAPTER

De toegang tot de te testen schakeling gebeurt via een gepaste adaptor.

PCB
- naaldadaptor (bed-of-nails)
- guided probe (diagnostic probe)
- testsocket (vb ICE = in circuit emulator)
- edge connector

IC
- personality card

Bij elke accessmogelijkheid hoort vaak ook een bepaalde testfilosofie en een specifiek testprogramma.

a) NAALDADAPTOR (BED OF NAILS)

Met een naaldenbed wordt kontakt gemaakt met de PCB langs de soldeerzijde. Daardoor komen een groot aantal testpunten beschikbaar voor kontroleerbaarheid en observeerbaarheid.
De testmogelijkheden met deze adaptor zijn:

- Loaded board opens / shorts testing.

Testen van de interkonnektie na bestukking en soldering

Short (90%) — 1 naald / verbinding

Open (10%) — 2 naalden / verbinding

- In circuit testing

Testen van elke chip op de PCB onafhankelijk van de chips op de PCB.
De primaire ingangen worden door naalden gestuurd om te vermijden dat ze door andere chips gestuurd worden.

De verschillende knopen worden op een gewenste waarde geforceerd.
Dit gaat gepaard met intensief vermogenverbruik over een korte periode (<1s)
De totale testduur moet beperkt blijven om de PCB niet te beschadigen.

Evaluatie

+ goede foutlokalisatie, eenvoudig testprogramma
- problemen met hoge stroom uitgangen
- dure testapparatuur (vele testpunten ondersteunen!)
- per type te testen kaart moet een speciale naaldadapter ontwikkeld en gebruikt worden.
- nodige mechanische voorzorgen voor kontaktering met lage weerstand.
- layout beperkingen
  . extra testvlakjes
  . komponenten op een vast raster
  . enkel bereikbaar langs soldeerzijde.
- niet toepasbaar op SMD, PLCC, SO komponenten aan de komponentzijde.
- extra kapacitieve belasting (dynamisch testen)

b) GUIDED PROBE (SIGNATURE ANALYSIS)

Een guided probe wordt niet alleen gebruikt. Meestal worden via een edge konnektor of testsocket reeds stimuli aan de te testen schakeling aangeboden. Voor diagnose doeleinden (foutlokalisatie) kan met de guided probe de responsie op testpunten worden gemeten (aangeduid door het testsysteem) → seriële doorvoerbaarheid.
In signature analysis wordt datakompaktie toegepast door de opgenomen responsie aan een lineair teruggekoppeld schuifregister (LFSR) aan te bieden. De resulterende waarde in dit schuifregister na de responsie is de 'handtekening' voor die knoop (te vgl met de handtekening het foutvrije systeem).

c) IN-CIRCUIT EMULATION

In-circuit emulation is een testtechniek die gebruikt wordt bij de ontwikkeling en soms ook bij de produktie van microprocessor systemen. De verbinding van de tester met de kaart gebeurt via de socket voor de microprocessor. Op deze wijze krijgt de ICE toegang tot de voornaamste signalen van de kaart (bus).
De emulator voorziet een 'captieve' microprocessor (identiek in funktie aan diegene die in het systeem gebruikt zal worden) en werkt op volle snelheid. Programma's kunnen vanuit interne RAM of kaartrom lopen. De interne registers van de processor kunnen geobserveerd en gekontroleerd worden.

d) EDGE CONNECTOR

In een systeem kommuniceren verschillende kaarten met elkaar via een gemeenschappelijk bussysteem. De fysische interkonnektie gebeurt via een edge connector. Dit is tevens de toegang bij uitstek voor funktionele testen.

e) PERSONALITY CARD

De interkonnektie van een ASIC (PLCC, PGA, SO, DIP ..) met de ATE gebeurt via een aangepaste personality card. In tegenstelling met PCB bestaat hier een bijkomende beperking daar de verpakkingspinnen de enige fysische toegang tot de schakeling geven.

TEST STRATEGIE

- **STATIC FUNCTIONAL TESTING**

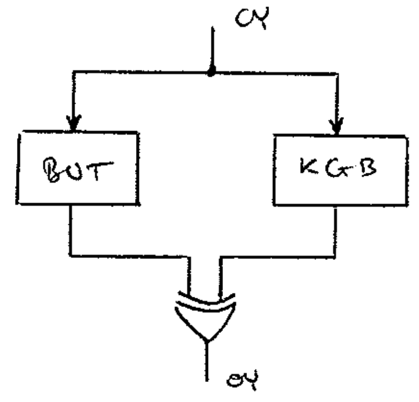
  De testvectoren worden aangeboden aan een snelheid lager dan de werkingssnelheid van de schakeling. De uitvoeringssnelheid wordt beperkt door de I/O tijd van het toestel.

- **DYNAMIC FUNCTIONAL TESTING**

  De testvectoren worden aangeboden op de snelheid van de schakeling. Er is echter per pin een grotere hoeveelheid geheugen voorzien voor snelle generatie en opslag van testpatronen.
  (geen bed-of-nails: te hoge capacitieve belasting).

- **DYNAMIC REFERENCE TESTING**

  De input testvectoren worden aangeboden aan de te testen kaart en aan een referentie kaart (known good board). De uitgangen van beide kaarten wordt met een exor vergeleken en gemonitored.



2. ONTWERP VOOR TESTBAARHEID ("DESIGN FOR TESTABILITY")

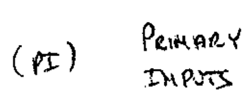
Om de kosten verbonden met testen (ontwikkeling/uitvoering testprogramma, defect level) aanvaardbaar te maken, moet reeds tijdens het ontwerp rekening worden gehouden met de testbaarheid van de schakeling.
Drie benaderingswijzen worden besproken:

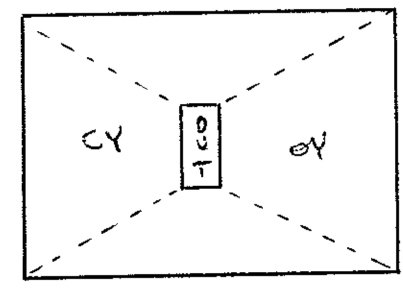
- Meting van testbaarheid
- Ontwerptechnieken (ad hoc / gestruktureerd)
- Checklist met praktische richtlijnen.

Belangrijk is vooral dat de ontwerper zelf bij de problematiek van testen wordt betrokken (waar dit vroeger strikt gescheiden was, is dit nu - gezien de komplexiteit van de schakelingen - een noodzakelijke samenwerking).

a METING VAN TESTBAARHEID

De algemene doelstelling van testpatroon generatie, is het aanleggen van een funktionele test aan iedere herkenbare deelfunktie van de schakeling.





PRIMARY OUTPUTS (PO)

De ingangsknopen van de DUT moeten op de gewenste waarde worden geplaatst door een gepast patroon op de PI's aan te leggen. De mate waarin dit mogelijk is, noemt men de kontroleerbaarheid (CY).
De respons op de uitgangsknopen v/d DUT moet op de PO's zichtbaar gemaakt worden. Dit legt extra kondities op aan de PI's. De mate waarin dit mogelijk is, noemt men de observeerbaarheid (OY).

De testbaarheid van elke interne knoop, wordt bepaald door de kontroleerbaarheid en de observeerbaarheid.

Voorspelling van het profiel van kontroleerbaarheid, observeerbaarheid en testbaarheid van een schakeling, nog vóór een testprogramma wordt opgesteld, geeft de ontwerper de mogelijkheid om gebieden van lage testbaarheid (CY, OY, TY) te lokaliseren in het ontwerp, en de nodige verbeteringen door te voeren. Eventueel kan dit testprofiel ook dienen als basis voor de bepaling van een teststrategie.

De theorie voor een specifiek analyse systeem voor testbaarheid (CAMELOT = Computer - Aided - MEasure for Logic Testability, Cirrus Computers), wordt hier uitgewerkt voor kombinatorische schakelingen.

① KONTROLEERBAARHEID (CONTROLLABILITY = CY)

De kontroleerbaarheid CY geeft aan in welke mate een knoop op de waarde 1 of de waarde 0 kan gezet worden. De extreme waarden zijn:

CY = 1 : de knoop kan even eenvoudig op 1 dan wel op 0 gezet worden (vb Primary Input)

CY = 0 : de knoop kan niet op 1 (0) gezet worden.

De waarde van CY van elke schakelknoop ligt tussen beide waarden.

De kontroleerbaarheid CY wordt gedefinieerd als (voor zuiver kombinatorische schakelingen):

$$CY(\text{uitgangsknoop}) = CTF \cdot \left\{ \frac{1}{j} \sum_j CY(\text{ingangsknopen}) \right\}$$

De kontroleerbaarheid van de uitgangsknoop van een komponent wordt bepaald door de gemiddelde kontroleerbaarheid van de ingangsknopen en de CTF = Controllability Transfer Factor van de komponent.

De CTF is enkel afhankelijk van de logische funktie van de komponent (niet van zijn positie in de schakeling) en wordt gedefinieerd als:

$$CTF = 1 - \left| \frac{N(0) - N(1)}{N(0) + N(1)} \right|$$

met $N(i)$ het aantal mogelijke manieren om i (0,1) op de uitgang van de komponent te produceren. $CTF = 0$ als $N(0)$ of $N(1) = 0$, wat aanduidt dat de uitgangsknoop niet kan gecontroleerd worden. De extreme waarden zijn:

$CTF = 1$: $N(0) = N(1)$

$CTF = 0$: $N(0) = 0$ of $N(1) = 0$

De CTF kan bepaald worden uit de waarheidstabel van de komponent en is aangegeven voor enkele eenvoudige komponenten (kombinatorisch)

N(0) = 1
N(1) = 1
CTF = 1

N(0) = 1
N(1) = 3
CTF = 0.5

N(0) = 1
N(1) = 7
CTF = 0.25

N(0) = 2
N(1) = 2
CTF = 1

Een X(N)OR poort vermindert de kontroleerbaarheid niet.

Voorbeeld

CTF = 1/2
d 1, c 1, 1/2 e
CTF = 1/4
b 1
a 1, 1, f, 0.2, g
CTF = 1

$$CY(g) = \frac{1}{4} * \frac{1}{3}(1+1+0.5) = \frac{1}{4} \times \frac{5}{6} = 0.2$$

Hoe meer het aantal poortingangen stijgt, daalt de kontroleerbaarheid.

② OBSERVEERBAARHEID OY.

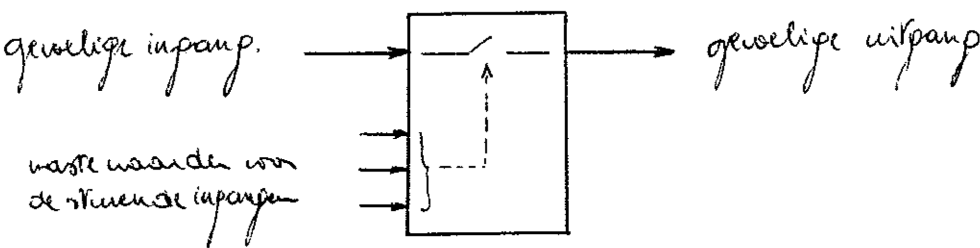
De observeerbaarheid OY van een knoop, gevraagd in welke mate de waarde van de knoop kan geobserveerd worden op één of meerdere uitgangen. De extreme waarden zijn:

OY = 1 : primaire uitgang (PO)

OY = 0 : niet observeerbaar.

De observeerbaarheid daalt van de PO's naar de PI's toe.

Bij de generatie van een test, moet het fout-effect (D | D̄) via een gevoelig pad (per device van een gevoelige ingang naar een gevoelige uitgang) tot aan een primaire uitgang (PO) gebracht worden. De transfer van OY kan worden begrepen door een relais te veronderstellen tussen de gevoelige ingang en uitgang van een komponent op het gevoelig pad.



Indien het kontakt gesloten is, propageert het pad; als het open is wordt het pad onderbroken. De kans dat het kontakt van een bepaalde komponent gesloten is (kans op propagatie), wordt bepaald door de OTF = Observability Transfer Factor van de komponent en de gemiddelde kontroleerbaarheid van de ingangspinnen van de komponent.

De observeerbaarheids – 'transfer' wordt gegeven door (voor één komponent)

$$OY(I-O) = OTF(I-O) \times \left\{ \frac{1}{i} \sum_i CY(\text{ingangsknopen}_{\text{sturende}}) \right\}$$

De kans dat het volledig propagatiepad gerealiseerd wordt, is het produkt van de OY transfers van alle komponenten op het pad:

$$OY(\text{knoop}) = OY(PO) \times OY(PO-k_1)\ OY(k_1-k_2) \ldots OY(k_i-\text{knoop})$$

De OTF = Observability Transfer Factor geeft aan in welke mate de propagatie door een komponent (van I naar O) kan gerealiseerd worden.
De extreme waarden zijn:

$OTF = 1$: steeds propagatie

$OTF = 0$: geen propagatie

De propagatie door een komponent kan worden beschreven met de begrippen van het D-algoritme:

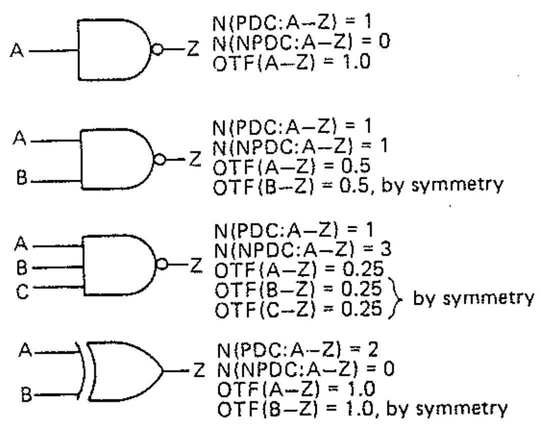
PDC = Propagating D-cube
Deze geeft de gevoelige ingang, de gevoelige uitgang en de ingangskombinatie die het pad sluit.

NPDC = Non-Propagating D-cube
Deze geeft de gevoelige ingang, de niet gevoelige uitgang en de ingangskombinatie die het pad blokkeert.

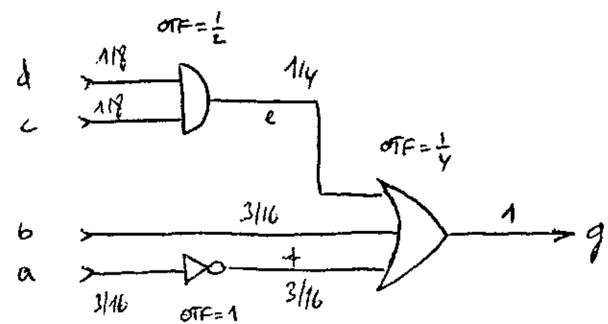
De OTF wordt gedefinieerd als (voor bep. komponent, ingang-uitgang paar):

$$OTF(I-O) = \frac{N(PDC:I-O)}{N(PDC:I-O) + N(NPDC:I-O)}$$

met $N(PDC:I-O)$ het aantal PDC voor het ingangs-uitgangs (I-O), zijnde het aantal mogelijke manieren om het fout-effekt te propageren.
Voor een aantal eenvoudige komponenten (kombinatorisch):



Voorbeeld



$$OY(e) = OY(g) \cdot OY(e-g) = 1 \cdot \left(\frac{1}{4} \times 1\right) = \frac{1}{4}$$

$$OY(f) = 1 \cdot \frac{1}{4} \times \frac{3}{4} = \frac{3}{16}$$

③ TESTBAARHEID (TESTABILITY = TY)

De testbaarheid van een knoop wordt gedefinieerd als :

$$TY(\text{knoop}) = CY(\text{knoop}) \times OY(\text{knoop})$$

met als extreme waarden :

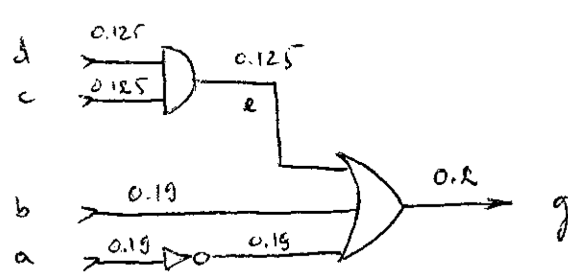
$TY = 0$: CY of $OY = 0$

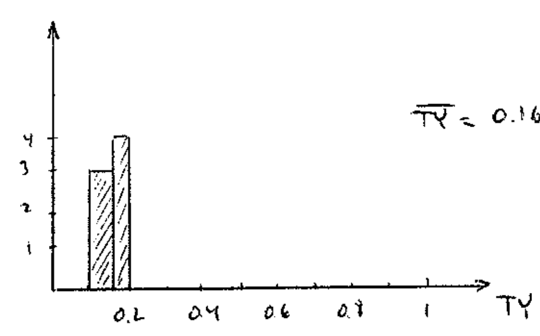
$TY = 1$: $CY = OY = 1$

De testbaarheid van de totale schakeling wordt gedefinieerd als:

$$TY(\text{schakeling}) = \frac{\Sigma(TY : \text{knoop})}{\#\text{knopen}}$$

Voorbeeld





Uit het testbaarheids-profiel kunnen de zones met lage testbaarheid bepaald worden. De invloed van acties ter verbetering van de testbaarheid (bv toevoegen van testpunten – OY/CY) op het testprofiel kunnen geëvalueerd worden.

b. ONTWERPTECHNIEKEN VOOR TESTBAARHEID

De ontwerptechnieken voor testbaarheid worden opgesplitst in 2 kategorieën:

- AD HOC

  Deze technieken lossen een testprobleem op voor een specifiek ontwerp maar zijn niet algemeen toepasbaar

- GESTRUKTUREERD ONTWERP

  Deze technieken zijn algemeen toepasbaar en stellen enkele ontwerpregels voorop bij het ontwerp van de schakeling. Ze hebben de bedoeling de sekwentiële komplexiteit van een netwerk te reduceren om testgeneratie / evaluatie te vereenvoudigen.

Door de toenemende komplexiteit van de ontworpen schakelingen, neemt ook het volume testdata belangrijk toe. Dit volume kan gereduceerd worden door SELF-TEST. Hierbij zorgt de schakeling zelf voor generatie en analyse (data-compressie) van de eigen testpatronen.

① AD HOC TECHNIEKEN

De ad hoc technieken worden meestal opgelost op PCB-niveau en vereisen niet noodzakelijk een ingrijping in het 'logisch' ontwerp van de implementatie ervan.

1.1 PARTITIONERING

Naarmate de komplexiteit van de schakeling toeneemt, stijgt ook de CPU tijd nodig voor testgeneratie en fout simulatie:

$$\text{CPU TIJD} \sim (\#\text{ gates})^3$$

Om deze wetmatigheid te doorbreken worden 'verdeel en heers' technieken toegepast

Partitionering kan op een aantal wijzen uitgevoerd worden:

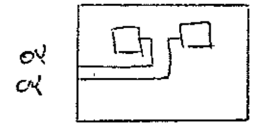
- Mechanische partitionering.

  De schakeling wordt in 2 functionele PCB's verdeeld (CPU tijd × 1/4).

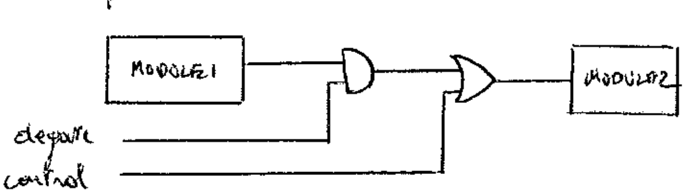
  Nadeel: extra kosten, tegen verhoogde integratie.

- Elektrische partitionering.

  - Jumper wires



  - Degating

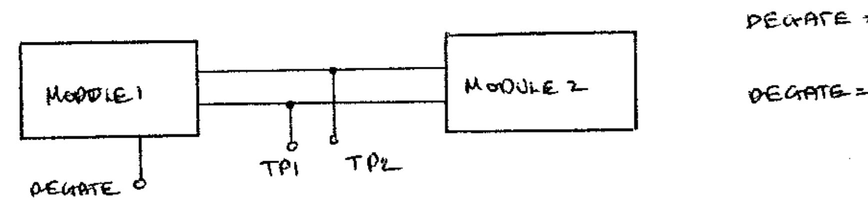


Vb oscillator.

Deze technieken vergen extra primaire ingangen en uitgangen (konnektor) en eventueel extra logika (degating).

1.2 TEST PUNTEN

Extra testpunten worden voorzien op de PCB om de observeerbaarheid (testpunt = uitgang) en kontroleerbaarheid (testpunt = ing) te verhogen.
Soms ingang + uitgang:



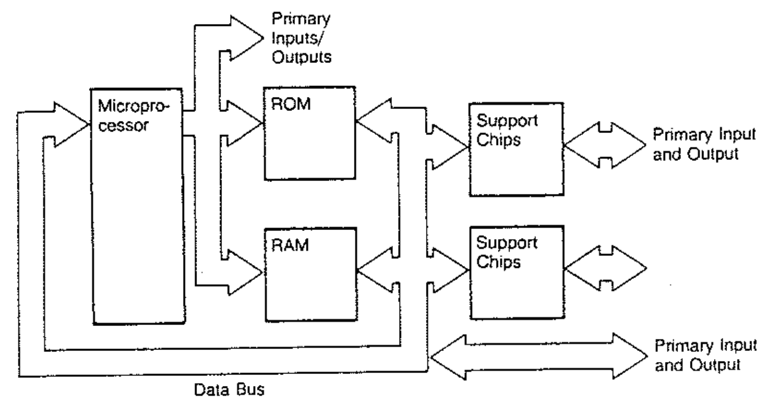
DEGATE = : TP1 = observe

DEGATE = : TP1 = control

De testpunten worden bijeengebracht op een testkonnektor / testsocket of worden gekontakteerd mbv een naaldadaptor (bed of nails) voor in-circuit-testing.

1.3. BUS ARCHITECTUUR

Bus gestruktureerde logika is opgebouwd rond een microprocessor.
(toenemend gebruik: gedistribueerde intelligentie, software gestuurde toepassingen)
De buslijnen zorgen voor een natuurlijke partitionering van de PCB,
ze geven immers toegang tot de verschillende modules op de kaart.



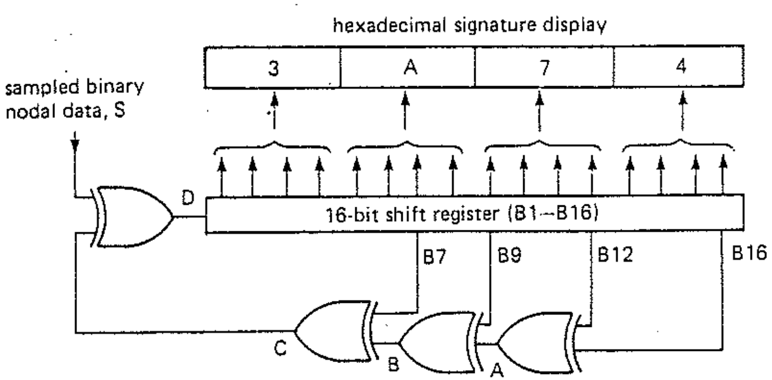
Bus Structured Microcomputer Board.

De databus is verbonden met de µP, ROM, RAM en periferie.
De busmaster (op kaart: processor, DMA controller → extern: testsysteem)
kan met de adresbus een komponent selekteren en regelt met de kontrolebus
de datatransfer (richting, snelheid)
Indien het testsysteem toegang heeft tot de bus (data, adres, control)
en busmaster kan zijn, dan kan 1 komponent afzonderlijk geselekteerd
en uitgetest worden. Bij bussystemen bestaat reeds een funktionele toegang
via een edgekonnektor, bij een single board computer moeten extra konnektoren voorzien
worden.
Gezien de grote hoeveelheid pinnen en de hoge snelheid nodig voor het
uittesten van een µP kaart, wordt de processor gebruikt om via de
bus zijn perifere komponenten (ROM, RAM, periferie) uit te testen. Dit is
'selftest' en wordt ondersteund door een extra selftest programma in (EP)ROM.

Essentieel bij bus-gestructureerde ontwerpen is de mogelijkheid tot 'isolatie' van de bussen (jumpers, tri-state buffers), zodat voorafgaandelijk eerst de fysische bus zelf eerst kan uitgetest worden.
(Normale testen voor foutisolatie maken gebruik van spanningsmetingen. Isolatie van een busfout vereist soms stroommeting, wat moeilijker is).

1.4. SIGNATURE ANALYSIS

Een standaard methode om een fout-bron te lokaliseren is het gebruik van een guided-probe (de operator brengt de probe op een aangeduide knoop in de schakeling). De schakeling wordt gestimuleerd met een gekende set patronen en de binaire response op een specifieke knoop wordt opgemeten. Er wordt data compressie toegepast op de opgemeten sequentie (reeks) om de hoeveelheid test data te reduceren. De meest gebruikte techniek is Cyclic Redundancy Check (CRC), gebaseerd op een lineair teruggekoppeld schuifregister (LFSR = Linear Feedback Shift Register), meestal 16 bit lang.



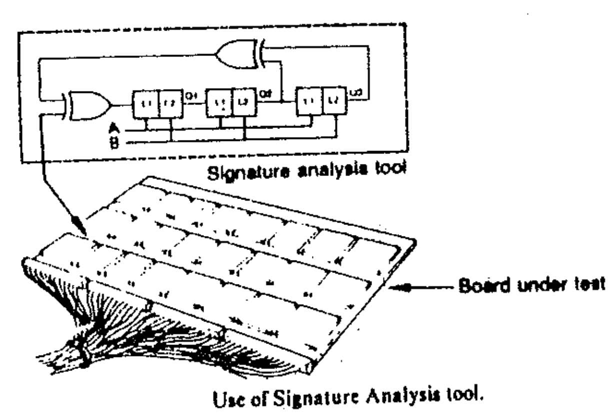
The feedback points in this register follow those advocated by Hewlett-Packard.

Linear feedback shift register.

De kaart en het LFSR worden geïnitialiseerd. Na een vast aantal klokperiodes wordt het schuifregister gestopt. De inhoud van het LFSR op dat moment is de handtekening (signature) van de specifieke knoop voor de gegeven schakeling. Een belangrijke data kompressie wordt uitgevoerd:

De 16 bit handtekening is het residu (rest) van de datastroom na deling door een niet-reduceerbare veelterm.
Foutdetectie is mogelijk indien de handtekening van het defekte systeem verschilt van het goede systeem. De kans dat een foutieve datastroom aanleiding geeft tot dezelfde handtekening als deze van het goede systeem is zeer klein (aliasing $\sim \frac{1}{2^n}$ $\Rightarrow$: 16 bit : 1/65536)

De Signature Analysis tool bestaat uit een probe verbonden met een LFSR, en maakt geen deel uit van de te testen schakeling.



Use of Signature Analysis tool.

Tijdens het ontwerp en de implementatie (PCB) moet reeds rekening gehouden worden met het feit dat de schakeling met Signature Analysis zal getest worden.

a) Controle van LFSR

- De clock van LFSR en PCB moeten synchroon zijn
- Initialisatie van LFSR en PCB
- START/STOP signaal om het tijdsvenster van de analyse vast te leggen (# klokcycli : 16 bit → meer dan 65K!)

b) Testpatronen

De testpatronen worden op de schakeling zelf gegenereerd

- µP bus systeem : testprogramma in ROM → deterministisch
- (eerst bus uittesten)
- digitale haar : LFSR, teller → pseudo-random.

De testpatronen moeten ervoor zorgen dat de testknopen minstens 1 keer van waarde wijzigen in de testschakeling (Sa 0, Sa 1)

c) Terugkoppeling.

Indien Signature Analysis gebruikt wordt voor fout-lokalisatie, is het nodig dat de terugkoppellussen onderbroken worden (In een terugkoppellus is de fout op elke plaats zichtbaar en kan de fout bron niet gevonden worden).

Identifikatie-kondities : 1. Komponent : alle I goed, O slecht

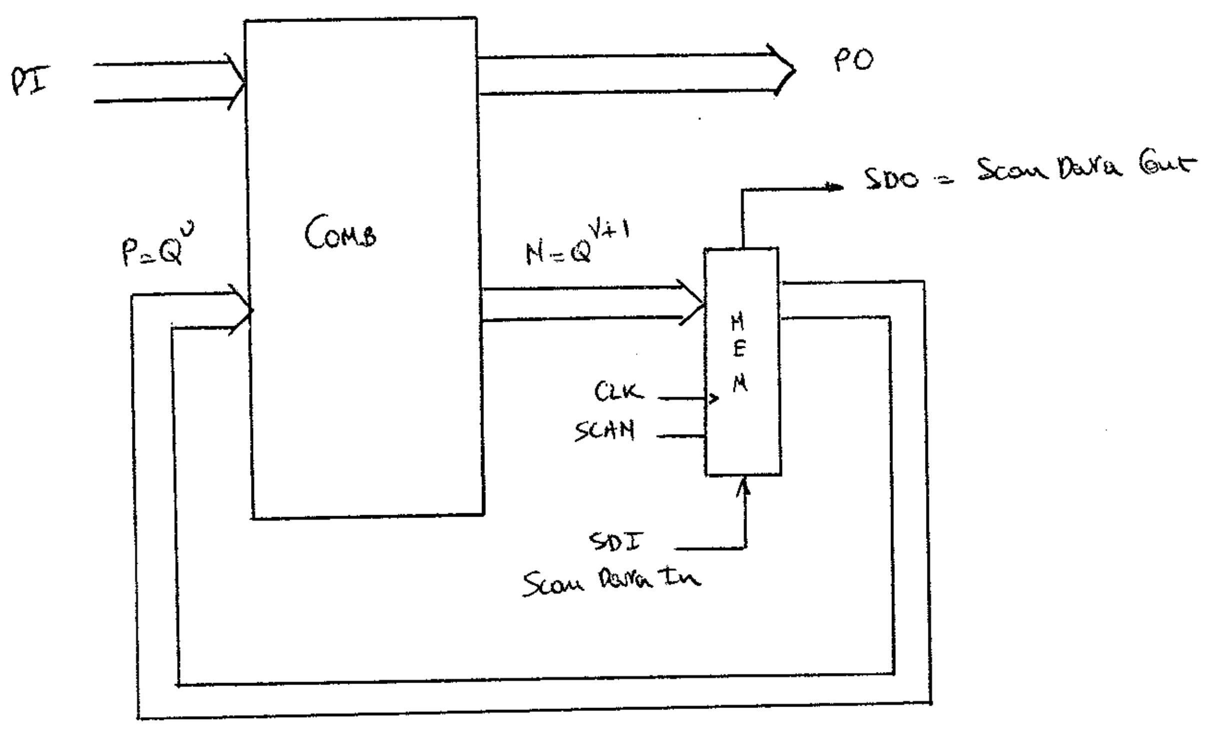
2. Interkonnektie : 1 zijde goede, andere zijde slecht.

② GESTRUKTUREERDE ONTWERPTECHNIEKEN.

Vooral sekwentiële systemen geven problemen met testgeneratie en foutsimulatie. De meeste gestruktureerde ontwerptechnieken zijn gebaseerd op een kontrole signaal dat de geheugenelementen van hun normale mode van werken omschakelt naar een mode die hen kontroleerbaar en observeerbaar maakt. De testgeneratie / fout simulatie wordt dan gereduceerd tot deze van een kombinatorische schakeling. Belangrijke producenten van main-frames (IBM) stellen vast dat ongestruktureerde ontwerpen tot onaanvaardbare testproblemen leiden. Ze hebben daarom belangrijke aandacht aan ontwerptechnieken van testbaarheid besteed.

2.1 SCAN ONTWERP (SCAN PATH)

Het scan ontwerp is gebaseerd op het koncept dat de geheugenelementen van een IC kunnen geschakeld worden als een schuifregister, en dat in die konfiguratie de waarde van elk geheugenelement kan gekontroleerd en geobserveerd worden.
Het algemene model van een synchroon sekwentieel circuit wordt dan:



$$PO = f_1(PI, Q^{\nu})$$
$$Q^{\nu+1} = f_2(PI, Q^{\nu})$$

Door rechtstreekse kontrole en observatie van de interne toestand van de geheugenelementen wordt de testgeneratie sterk vereenvoudigd:

1) De geheugenelementen kunnen afzonderlijk uitgetest worden

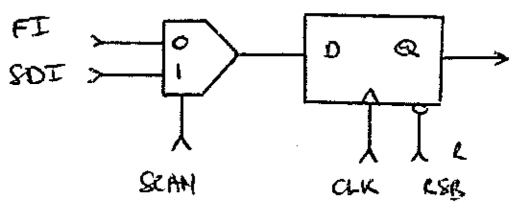
2) controle Q^{ν}, observatie $Q^{\nu+1}$ } → test

$$PO = f_1(PI, Q^{\nu})$$
$$\underbrace{Q^{\nu+1}}_{\text{observatie}} = f_2(PI, \underbrace{Q^{\nu}}_{\text{controle}})$$

De kombinatorische funkties f_1 en f_2 kunnen uitgetest worden.

SCAN FLIP_FLOP

Om de scan-path techniek te implementeren, moet gebruik gemaakt worden van geheugenelementen die met behulp van een kontrole signaal als schuifregister geschakeld worden.
Een mogelijke oplossing is de ingang met een multiplexer te verbinden:



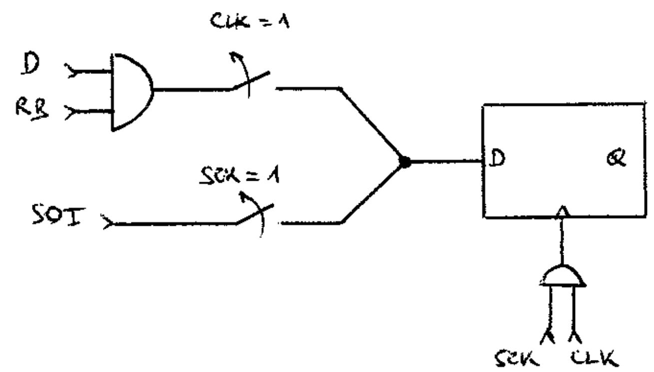
FI = Functional Input

SDI = Scan Data In

SDO = Q = Scan Data out

SCAN = Scan Select < 0 : systeem / 1 : scan

In de standaard-cel bibliotheek van Mietec gebeurt de selectie met behulp van waarden van CLK (systeemklok) en SCK (scan klok):

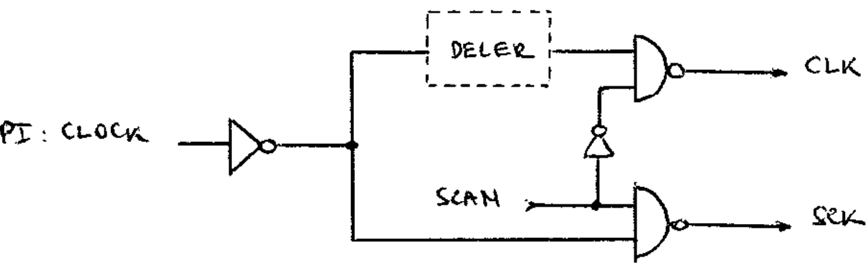


CLK = ⎍ , SCK = 1 : systeem

CLK = 1 , SCK = ⎍ : scan

RB = synchrone reset

Een gemeenschappelijke klokgeneratieschakeling moet worden voorzien:



De SCK wordt rechtstreeks door een PI gestuurd, zonder interne deling om lengte van het testprogramma te beperken. Let op voor asynchrone koppeling van geheugenelementen: CLK = 1 is met gegarandeerd in scan mode, dit geeft aanleiding tot een interne kortsluiting!

EVALUATIE SCAN-PATH

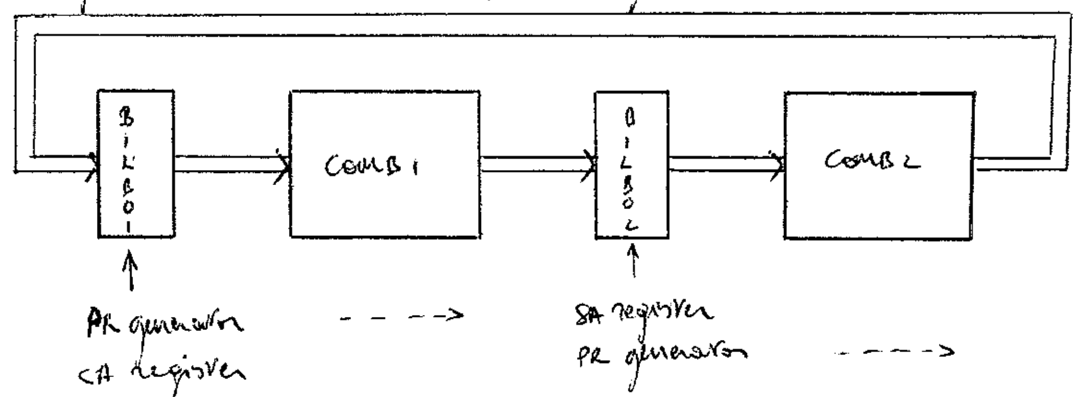
+ Vereenvoudigde testpatroon generatie / evaluatie:
  Scan path (schuifregister)
  Combinatorische logika → ATG haalbaar!

- Extra pinnen: PI (SDI, SCAN SELECT), PO (SDO)
- Extra hardware: { - komplexere geheugenelementen
  { - extra routing (SDI - SDO, SCK)

- Serialisatie van de testen maakt het testprogramma potentieel langer
- Scan in-systeem - scan out: geen real-time condities
- Vrijheid van ontwerper beperkt: specifieke FF's, geen asynchroon ontwerp.

2.2 BUILT-IN LOGIC BLOCK OBSERVATION (BILBO)

BILBO integreert de technieken van scan path en signature analysis (self-test).
De BILBO register worden in het systeem ingeschakeld als volgt:



De combinatorische logika wordt uitgetest met pseudo-random patronen en geanalyseerd met een signature analysis patroon dat wordt uitgescanned.

Evaluatie: + pseudo-random testpatronen { - eenvoudige testgeneratie
  { - reductie test data volume (minder scans!)

- probleem voor PLA → niet geschikt voor pr testpatronen (FI te hoog)
  vb: 20 inputs → 2^{20} mogelijkheden ⇒ $\frac{1}{2^{20}}$ kans voor korrekte test)
  goed voor kombinatorische logika met beperkte FanIn (≤4)

C. PRAKTISCHE RICHTLIJNEN (CHECKLIST)

Een aantal praktische richtlijnen voor de verbetering van de testbaarheid van PCB's en digitale IC's worden voorgesteld. Een aantal hebben betrekking op testgeneratie, een aantal op testuitvoering (→ ATE).

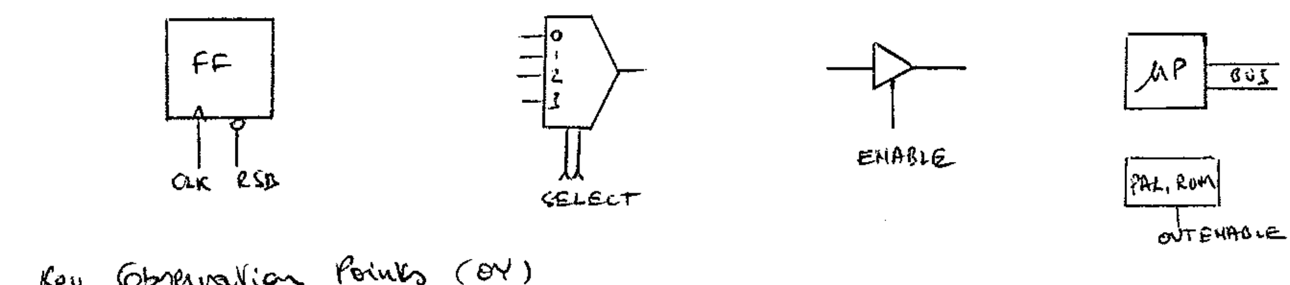
① VERBETERING TEST GENERATIE

In de ontwerpfase moeten de nodige voorzieningen getroffen worden om de testgeneratie te vereenvoudigen.

1. OPTIMALISEER KONTROLEERBAARHEID EN OBSERVEERBAARHEID

Verbeter de toegang (ACCESS) tot de belangrijke kontrole- en observatie punten.

- Key Control Points (CP)



- Key Observation Points (OP)

FF — Q

data funneling

OUT

redundante knoop

Met een programma als CAMELOT kan de testbaarheid gemeten worden en extra testpunten opgespoord worden.
Het ontwerp moet dan aangepast worden om de kontroleerbaarheid en observeerbaarheid te verbeteren.

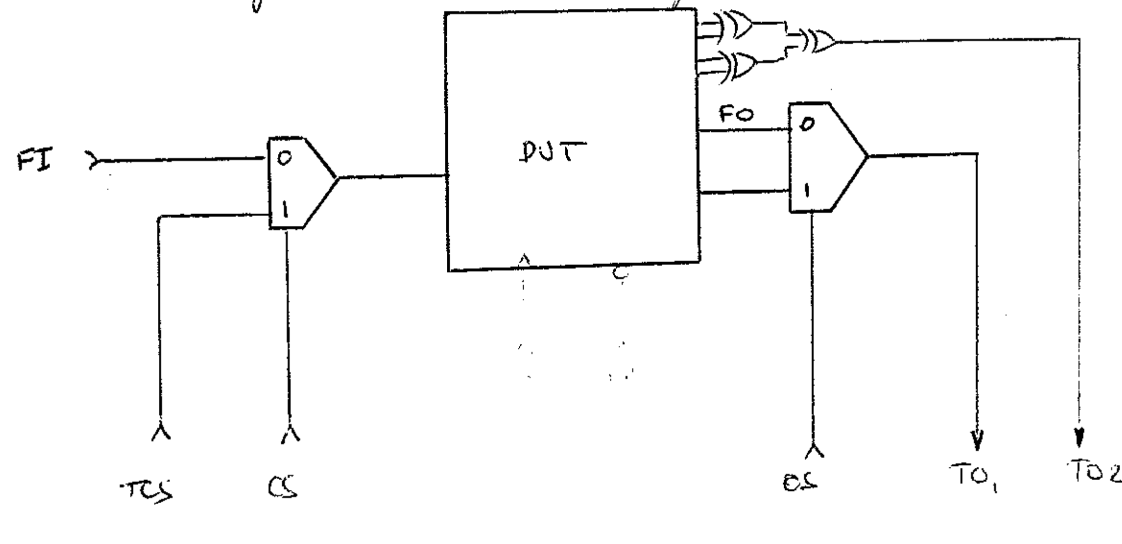
Methoden ter verbetering van de toegang (ACCESS)

- PCB
  - ongebruikte ingang / komponent
  - test socket
  - pin (wire-wrap), pin met tri-state buffer, pin-overdriving
  - chip clip
  - extra edge-konnektor

  → test of adaptor met ATE

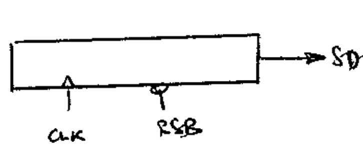
- IC

  Enkel verpakkingspinnen zijn PI en PO.
  Met behulp van demultiplexers / multiplexers kunnen andere funkties (testelementen) aan de PI's en PO's gegeven worden.
  De schakeling kan worden herkonfigureerd.



SDI

CLK RSB



INPUT (CONTROL)

OUTPUT (OBSERVE)

CS = CONTROL SELECT

TCS = TEST CONTROL SIGNAL

OS = OBSERVE SELECT

FI = FUNCTIONAL INPUT SIGNAL

TO = TEST OBSERVE SIGNAL

FO = FUNCTIONAL OUTPUT SIGNAL

Om verwarring te vermijden houdt men zich best aan volgende regel:

Systeem mode : CS = OS = 0

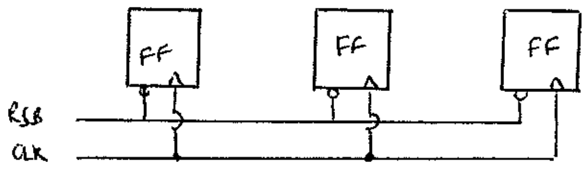
Indien een aantal te observeren signalen exclusief zijn (niet gelijktijdig actief) of niet simultaan wijzigen dan kan gebruik gemaakt worden van een pariteitsgenerator (uitgang wijzigt als 1 ingang wijzigt). Op deze wijze worden OS signalen uitgespaard.

Met behulp van schuifregisters kunnen een groot aantal interne signalen gekontroleerd en geobserveerd worden met behulp van een minimum aan PI's en PO's.

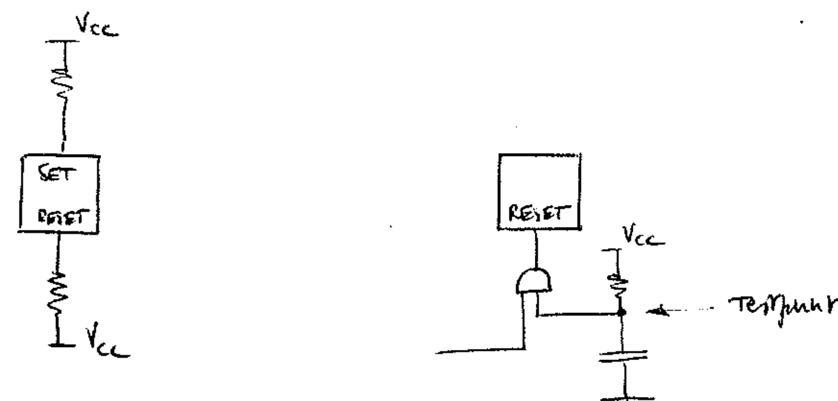
Nadeel van deze technieken is de extra hardware en eventueel extra tijdsvertraging.

2. DIREKTE KONTROLE VAN RESET EN KLOK

Het testsysteem moet rechtstreeks de resetingang (initiële toestand) en klokingang (toestandswijziging) van elk geheugenelement kunnen kontroleren. (Synchroon ontwerp geniet de voorkeur).

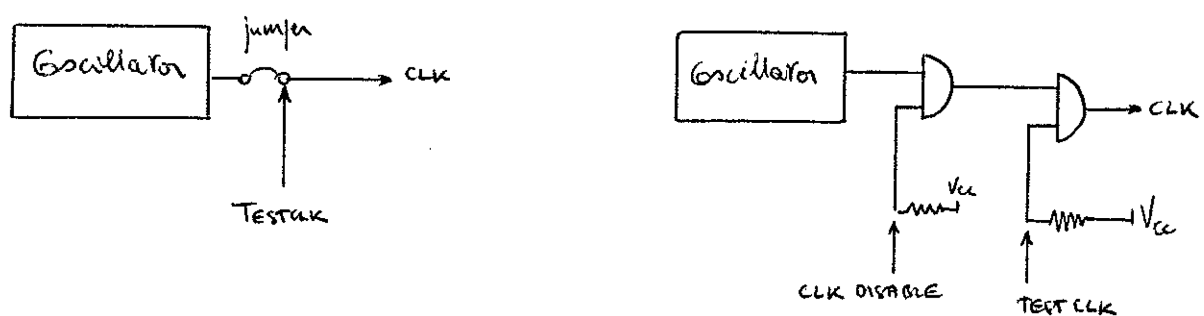


RESET



Voor testgeneratie is het een noodzaak dat de initiële toestand gekend is.

KLOK



Een on-board oscillator moet kunnen onderbroken worden en vervangen door de klok van de tester (TESTCLK):

- de tester heeft de werkingssnelheid onder kontrole (→ single step)
- de tester loopt synchroon met de kaart.

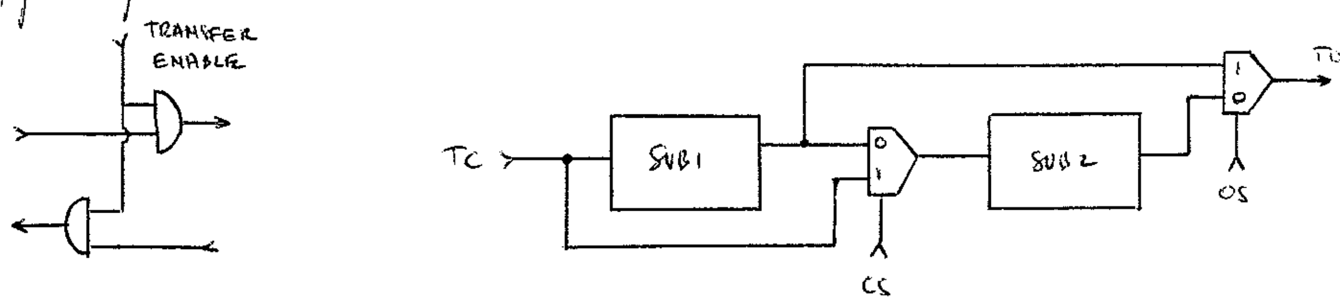
3. PARTITIONEER

- Met toenemende komplexiteit stijgen de testkosten met een 3de macht:

$$\text{TESTKOST} \sim \#(\text{gates})^3$$

Door partitionering kan deze wetmatigheid doorbroken worden $\left(x^3 \to \left(\frac{x}{2}\right)^3 + \left(\frac{x}{2}\right)^3 = \frac{x^3}{4}\right)$

De logische partitionering moet gebeuren in herkenbare deelfunkties. Dit kan fysisch gebeuren met tri-state buffers, poorten en multiplexers:



- Partitionering van binaire tellers geeft een belangrijke reduktie in het testprogramma:

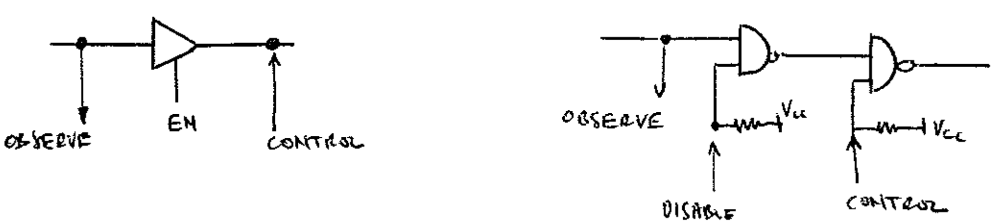
16 bit : lengte testprogramma $= 2^{16} + 1 = 65537$ klokpulsen $= 1.3$ sec (50kHz)

2x8bit : lengte testprogramma $= 2 \times (2^8 + 1) = 514$ klokpulsen $= 10.3$ msec (50kHz).

- Analoge en digitale schakelingen worden fysisch gescheiden (crosstalk, andere teststrategie).

4. TERUGKOPPELLUSSEN ONDERBREEKBAAR

Terugkoppellussen bemoeilijken testgeneratie (vb $CE = f(CE')$) en foutdiagnose (vinden van foutbron).



5. TE VERMIJDEN SCHAKELINGEN

Schakelingen die moeilijk testbaar zijn worden liefst vermeden:

- Redundante schakelingen.
  Een logisch redundante knoop is niet testbaar. Indien redundantie gewenst is (statische hazard) dan moet de redundante knoop observeerbaar gemaakt worden.
- Asynchrone schakelingen
  Asynchrone schakelingen zijn sneller (werken zonder klok) maar zijn moeilijk te ontwerpen en moeilijk testbaar: races moeten geëlimineerd worden, fouten kunnen deze terug introduceren, maar zijn moeilijk testbaar (⊐⊏)
- Monostabiele multivibrator (one-shot)
  De monostabiele multivibrator is een moeilijk testbare komponent
  - voor periodemeting is rechtstreekse observeerbaarheid van de uitgang nodig (geen kontrole van korte en lange periode)
  - korte periodes zijn moeilijk meetbaar (resolutie ATE)
- Timing komponenten
  Komponenten die voor elke kaart specifieke timing leveren, zijn liefst vermijden (vb potentiometers). Ze vragen manuele ingreep en zijn moeilijk automatisch meetbaar.

• Hoge fan-out
Lijnen met hoge fan-out (vb wired-or) geven problemen bij foutlokalisatie.

② VERBETERING TESTUITVOERING (ATE)

Vanuit de ATE worden ook een aantal specifieke eisen aan het ontwerp en de uitvoering van de schakeling gesteld.

1. EENVOUDIGE ADAPTOR

De adapter verzorgt de interface met de PI's en PO's. Om testproblemen te vereenvoudigen worden vaak specifieke interfaces voorzien:

flying leads — connectors: - edge, - onboard

Clips — + buffers.

Ze vergen echter veel mechanische ingrepen (vaak manueel) en geven extra kapacitieve belasting. Daarom moet de interface zo eenvoudig mogelijk worden gehouden ('keep it simple'). Checks voor de goede alignatie van de interface zijn welkom (vb kortsluitingsstuk op edge connector)

2. ATE EIGENSCHAPPEN & LIMIETEN

De ATE stelt eisen aan het ontwerp en het testprogramma door zijn specifieke eigenschappen en limieten:

- Eindige omschakeltijd tussen drive/sense mode van pin (uittesten van bidirectionele schakelingen): drive — SENSE, T
- Driver/sensor pinnen hebben limieten:
  - I_{max} (sink/source)
  - V_{max} (high/low)
  - Z_{in}, Z_{out} (Capaciteit) → belasting van schakeling (opletten met knopen met hoge fan-out!)

- Sensor pins
  - pulse catching voor snelle transities te meten
  - threshold waarden voor identificatie logische niveau's (TTL, CMOS, ECL)

  V_{TH} — liefst geen logische families mixen

- Driver pins
  - Skew (tijdsverschuiving tussen signalen die simultaan moeten wijzigen)
    - → skew mode ↔ broadside mode
    - δ minimaal (vb bus)
  - Slew rate van uitgangs operationele versterker. (t_r & t_f eindig).

    V_{TH} — out

    Buffers, schmitt-triggers voorzien op flankgevoelige signalen (vb klok).

- Timing
  - synchronisatie met klok of hoaur
  - drive / sense — ΔT → max tijdsverschuiving in logisch.

3. GUIDED PROBE

Fout-diagnose (fault-finding) gebeurt vaak met een guided probe. Om dit ordelijk en korrekt uit te voeren worden een aantal eisen gesteld:

- Layout
  - verzorgd
  - komponenten op regelmatige afstand.
  - nummering richtbaan
  - Testpinnen groeperen (konnektor, socket)
  - Identificatie PCB, pinnen
  - buslijnen naast elkaar op konnektor
  - voldoende plaats voor clips (geen IC's uit socket halen!)

4. IN-CIRCUIT TESTING

Met een bed-of-nails worden een groot aantal interne testpunten gekontakteerd. De interkonnekties en komponenten (geïsoleerd door back-driving) worden individueel uitgetest. Op deze wijze worden enkel produktiefouten getest. Funktionele testen worden nadien uitgevoerd via de edge-connector.

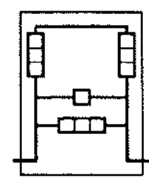
Er worden een aantal specifieke eisen gesteld aan de kaarten die met in-circuit testen worden geverifieerd:

- alle komponenten langs komponentzijde
- alignatiegaten voor korrekte kontaktering
- extra kontaktpaden (dicht tegen de komponent om gevaarlijke transities te vermijden)
- goede power/ground structuren. Door back-driving kunnen grote stromen door de grondlijn vloeien.
- high current outputs heeft in hoogimpedante toestand lage drempel door de testen (problemen met back-driving).

5. GOEDE DOCUMENTATIE VAN ONTWERP

- funktionele specifikaties
- verzorgde schema's
- timing diagramma's

BOUNDARY SCAN



1. WHY ?

- **TESTNODEN**
- **TECHNOLOGISCHE EVOLUTIE**
- **OPLOSSING**
- **HISTORIEK**

2. WHAT ?

2.1 ARCHITECTUUR

a. IC JTAG

b. PCB JTAG

2.2 TESTFUNCTIES

3. ARCHITECTUUR

3.1 BSC = Boundary Scan Cell

3.2 TAP-Controller

a. TAP - acties

b. TAP - state diagram

c. TAP - timing

3.3 DATA REGISTERS

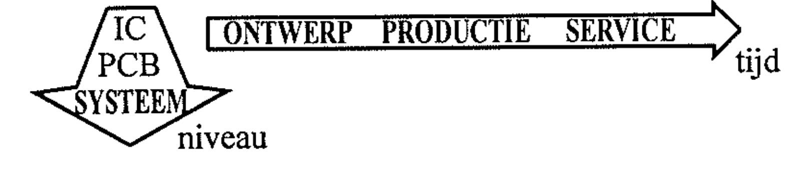
3.4 INSTRUCTIE REGISTERS

4. INSTRUCTIES

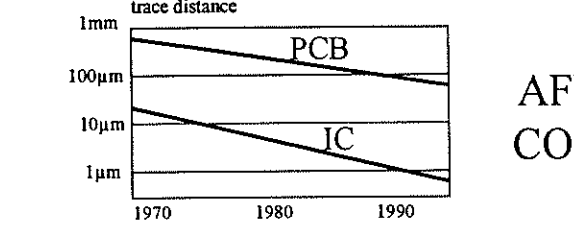
- **Sample/Preload**
- **INtest**
- **EXtest**
- **Globale timing**

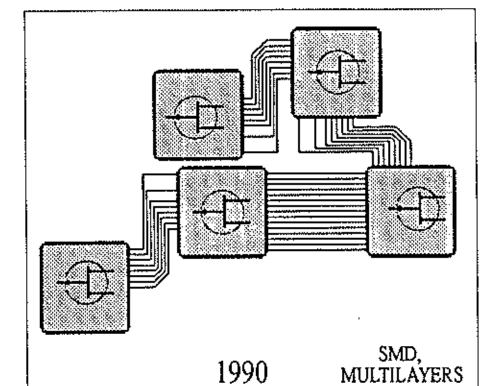
EVALUATIE

TESTNODEN



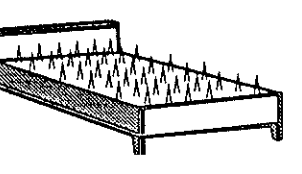
TECHNOLOGISCHE EVOLUTIE



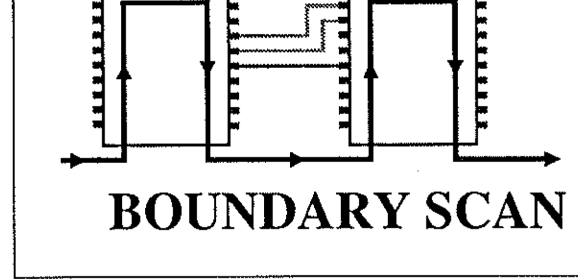


1970 INSERTION, DOUBLE LAYERS

OPLOSSING



Bed of nails



HISTORIEK

1985 — **PHILIPS**, British Telecom, Ericsson, Siemens, Bull, Nixdorf, Thomson — JETAG: Joint European Test Action Group

1986 — JETAG + (Texas Instruments, AT&T, DEC, IBM) = JTAG

1988 — LAATSTE VOORSTELLEN (JTAG versie 2.0) TER STANDARDISATIE VAN BOUNDARY_SCAN

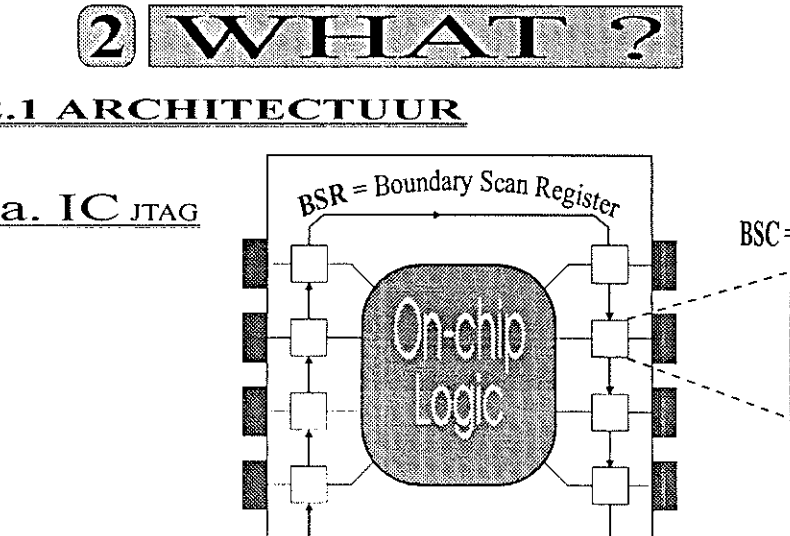
→ Institute of Electrical and Electronics Engeneers

→ IEEE Std 1149.1 1990 "Standard Test Access Port and Boundary-Scan Architecture"

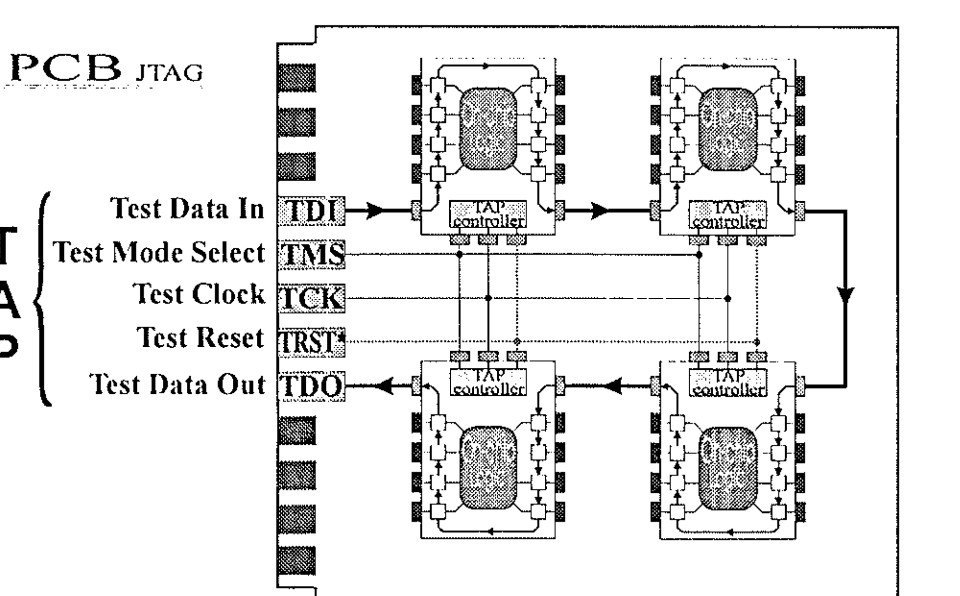
2 WHAT ?

2.1 ARCHITECTUUR

a. IC JTAG

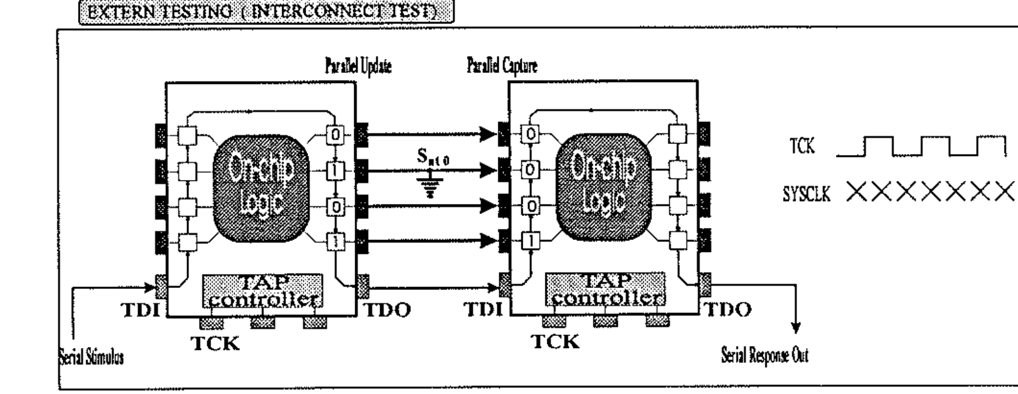


b. PCB JTAG

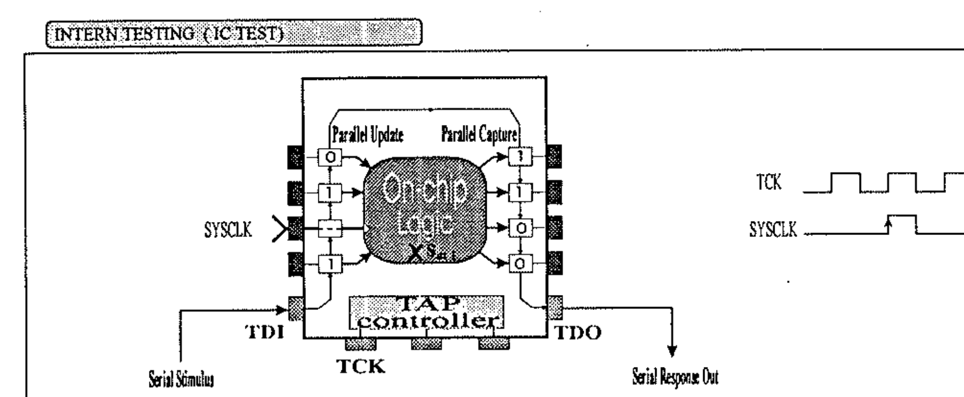


2.2 TEST FUNCTIES

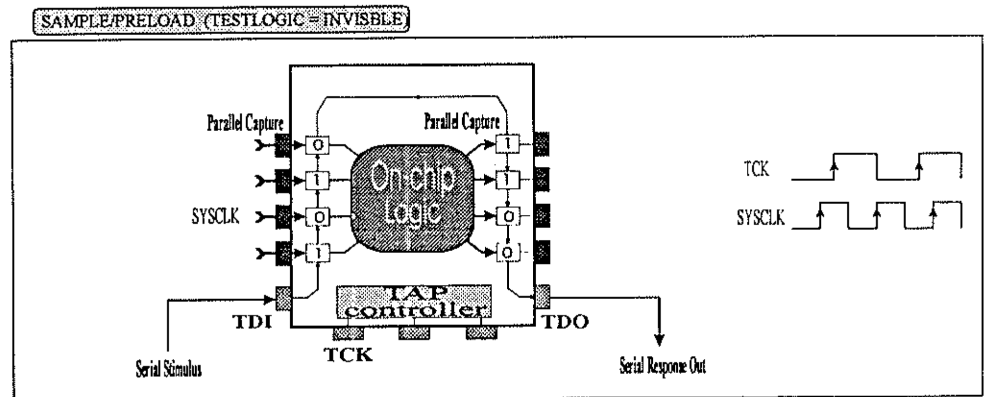
EXTERN TESTING (INTERCONNECT TEST)

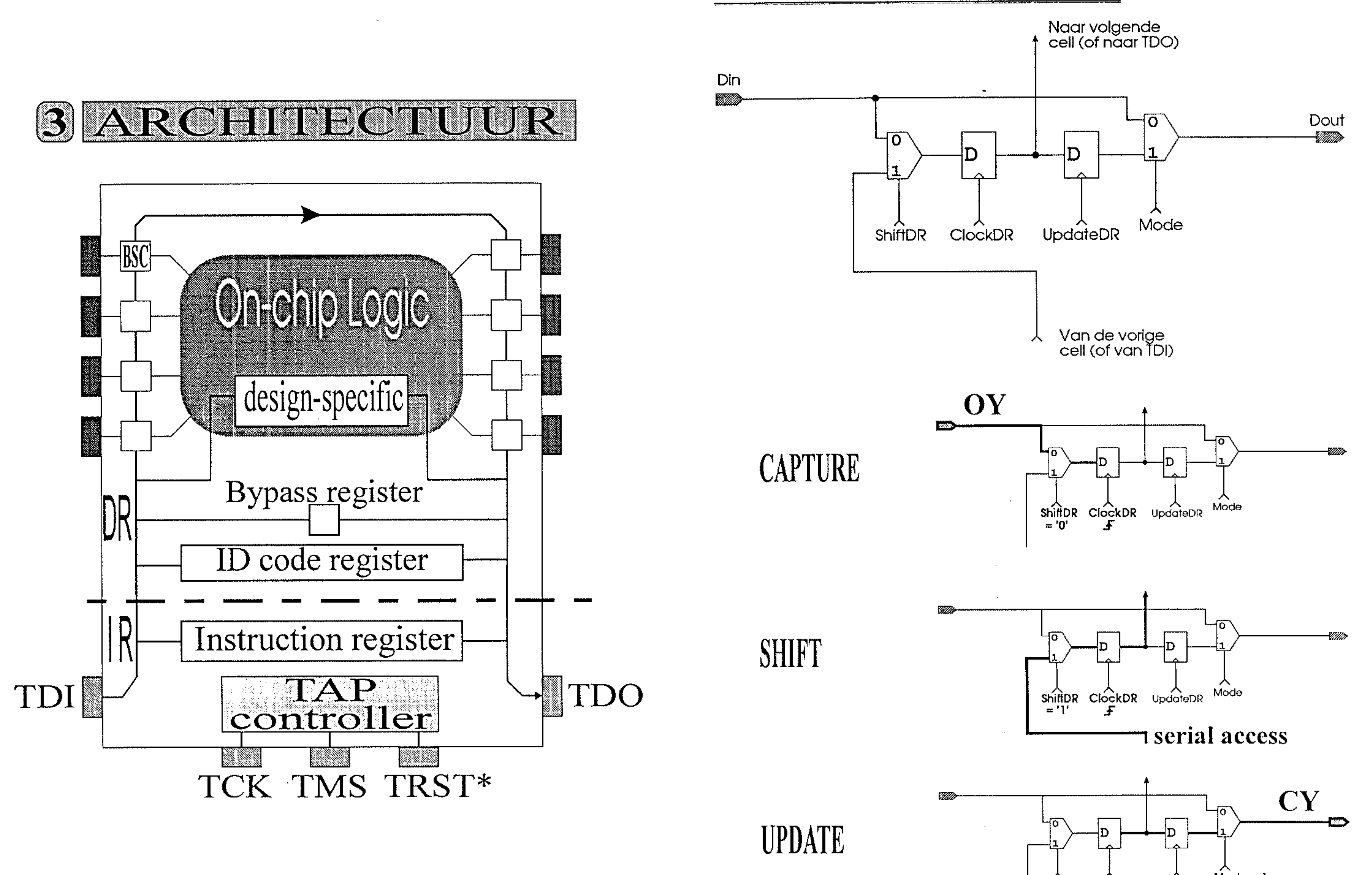


INTERN TESTING (IC TEST)



SAMPLE/PRELOAD (TESTLOGIC = INVISBLE)

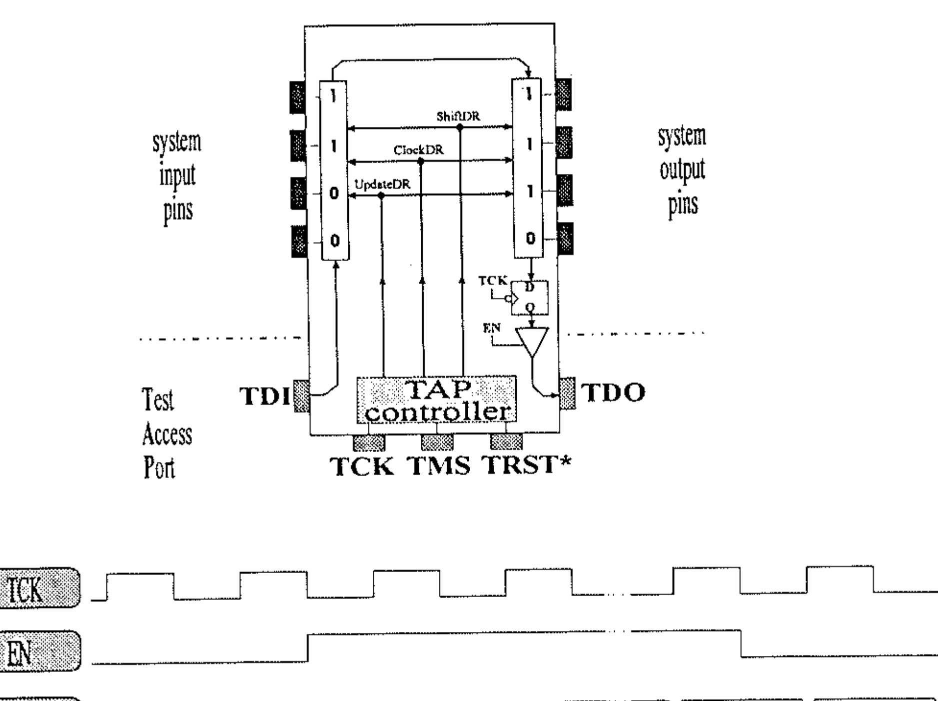


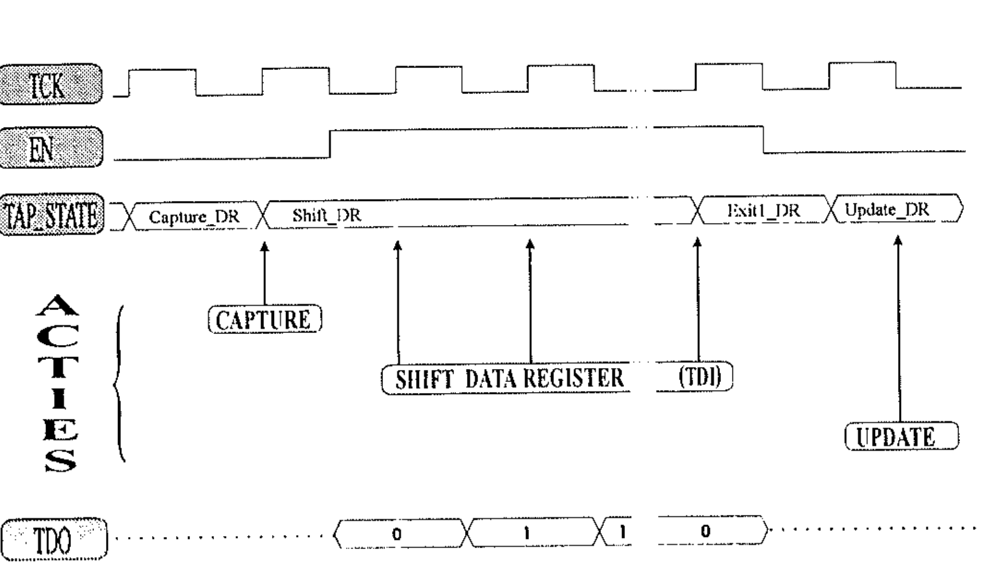


3 ARCHITECTUUR
On-chip Logic
design-specific
BSC
Bypass register
ID code register
Instruction register
TAP controller
DR
IR
TDI
TDO
TCK
TMS
TRST*
3.1 BSC = BOUNDARY SCAN CELL
Naar volgende cell (of naar TDO)
Din
Dout
ShiftDR
ClockDR
UpdateDR
Mode
Van de vorige cell (of van TDI)
CAPTURE
OY
ShiftDR = '0'
SHIFT
ShiftDR = '1'
serial access
UPDATE
CY
Mode = 1

3.2 TAP - CONTROLLER

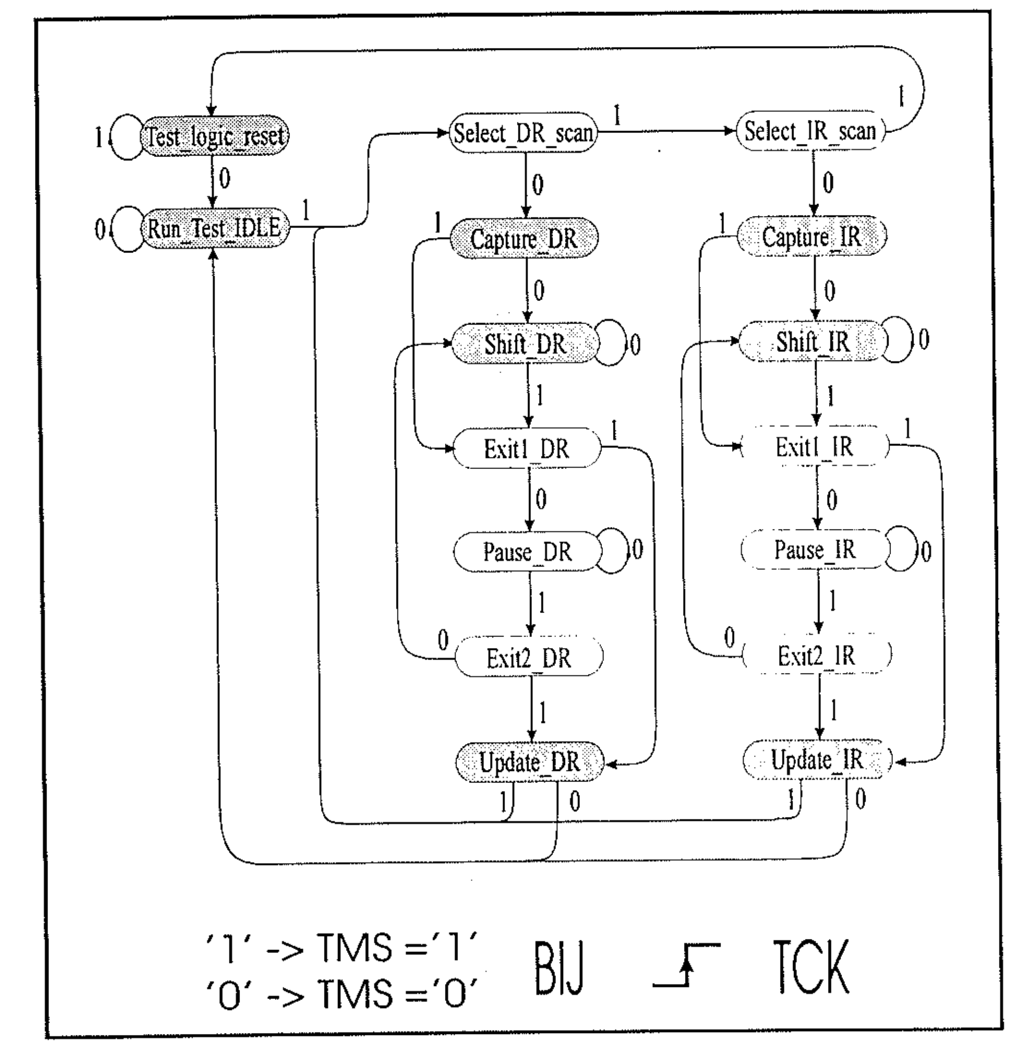
a. TAP - acties





ALGEMEEN
INLEZEN : TCK (TDI, CAPTURE)
UITLEZEN : TCK (TDO, UPDATE)

b. TAP - state diagram



3.3 DATA REGISTERS

BSC AAN BIDIRECTIONELE PIN

BIDIRECTIONAL SYSTEM PIN

EN

NAAR VOLGENDE CELL (of TDO)

CONTROLE CELL (BSC)

OUTPUT CELL (BSC)

INPUT CELL (BSC)

VAN VORIGE CELL (of TDI)

OUTPUT ENABLE

OUTPUT DATA

INPUT DATA

BYPASS REGISTER

TDO

D

TDI

ShiftDR

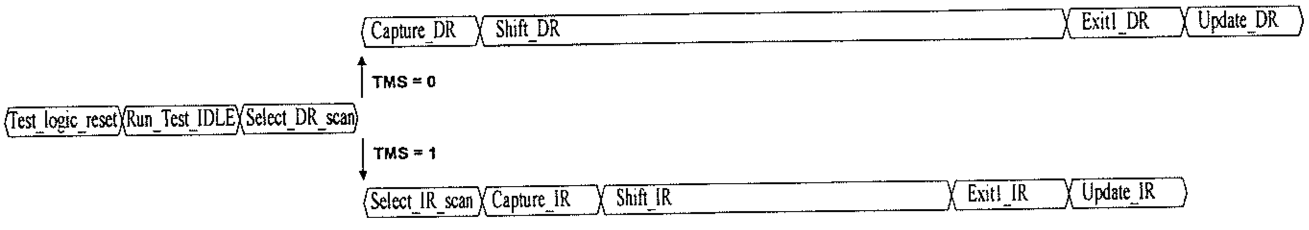
ClockDR

ID REGISTER

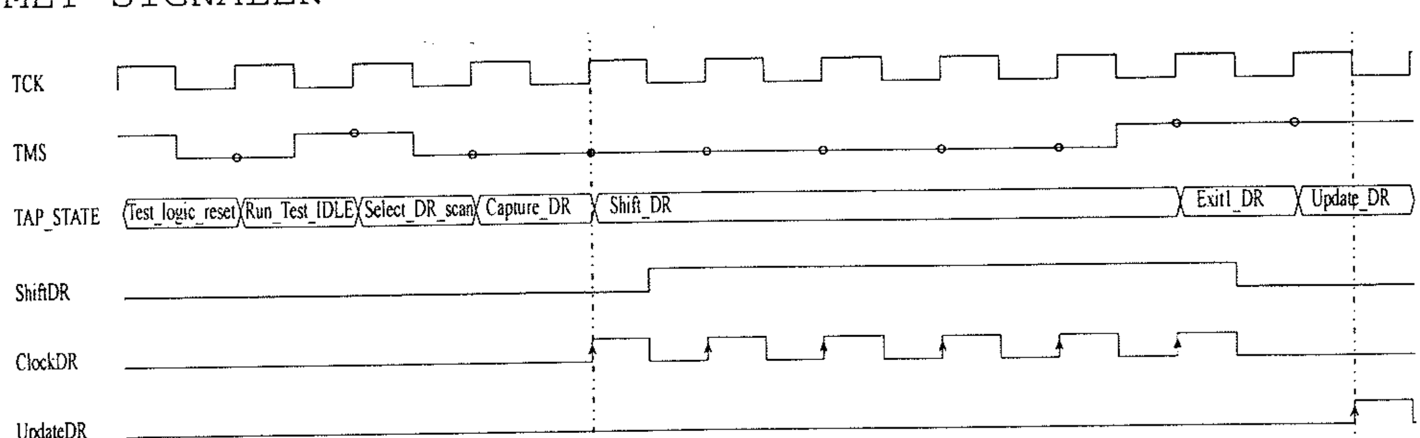
31 28	27 12	11 0
Version	Part number	Manufacturer identity 1
MSB		LSB

c. TAP - timing

PRINCIPIEEL



MET SIGNALEN

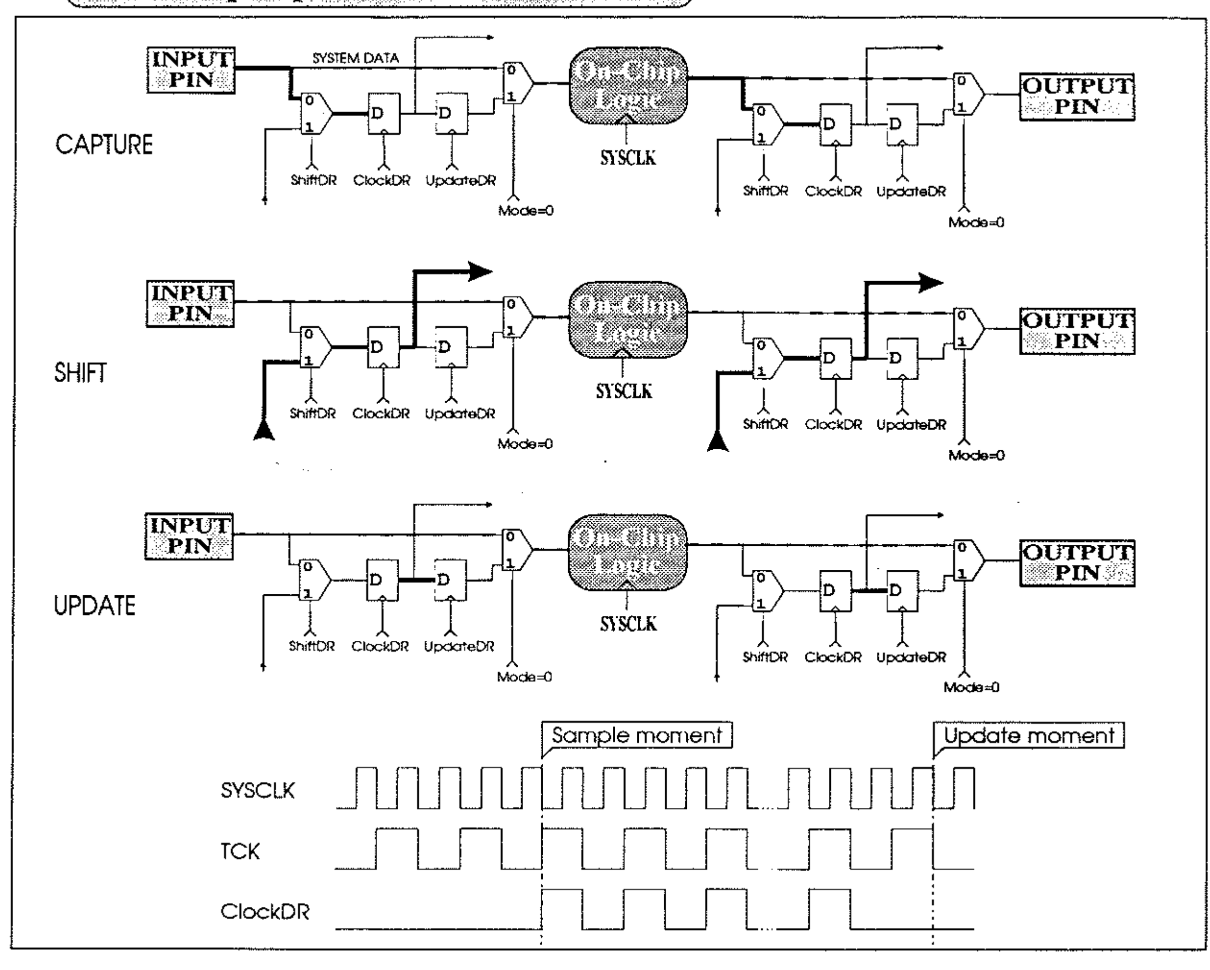


Device Inputs
Boundary-Scan register
INTERNAL LOGIC
Device Outputs
TEST DATA REGISTERS
(DR)
Device ID register
Design specific registers
BYPASS register
0
1
1D
EN
TDO
TCK
modes
reset
ClockDR
UpdateDR
ShiftDR
Instruction Decoder
instruction bits
Instruction shift register
INSTRUCTION REGISTER
(IR)
TDI
Sel_DR/Sel_IR
TMS
TCK
TRST*
TAP CONTROLLER
Enable
reset, ClockIR, UpdateIR, ShiftIR

3.4 INSTRUCTIE REGISTER

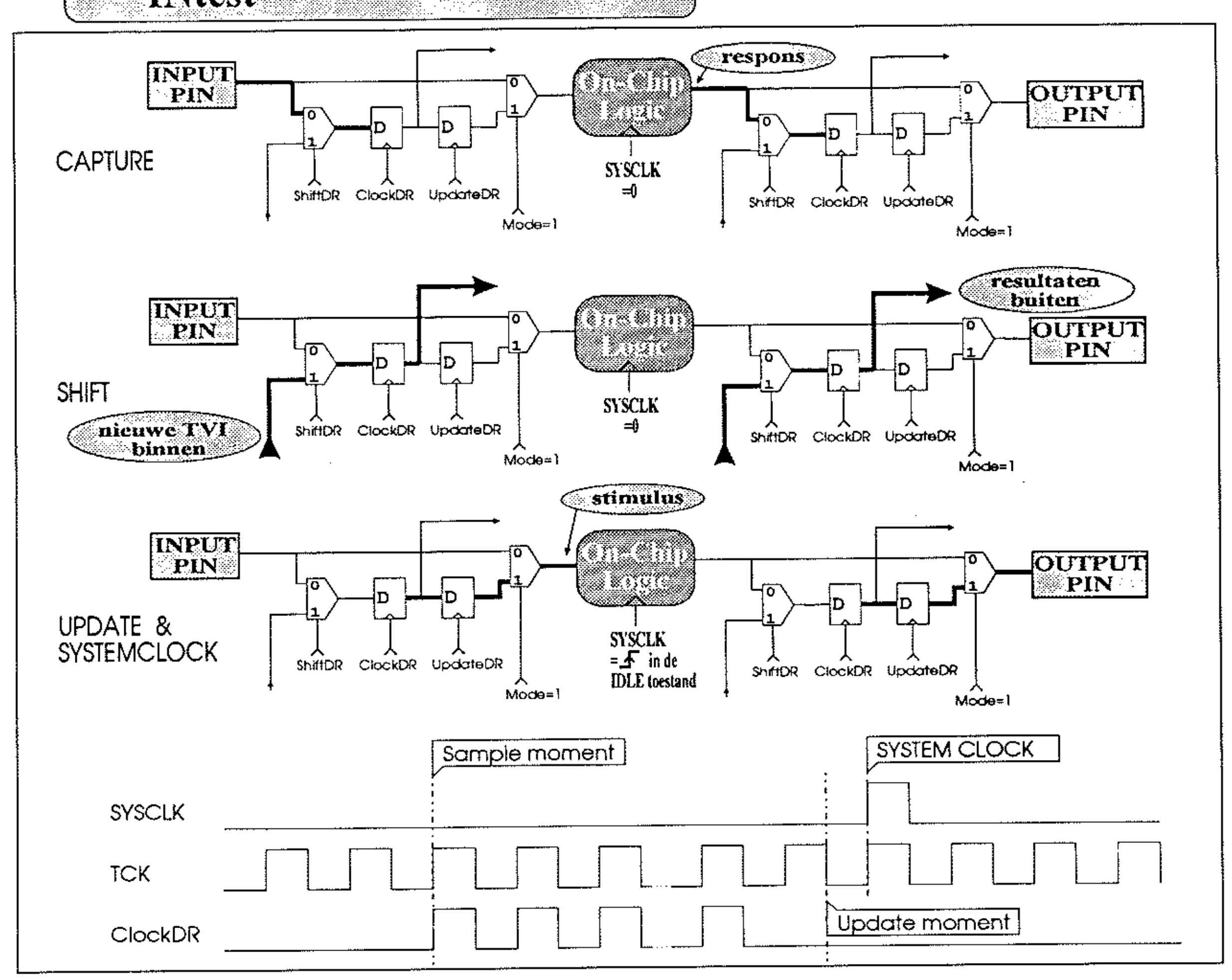
BSR
BSC_n
BSC_4
BSC_3
BSC_2
BSC_1
D_{out3}
D_{in3}
D
D
0 1
TDO
SelIR/SelDR
SelectDR
Mode
UpdateDR
ClockDR
ShiftDR
TMS
TCK
TRST*
TAP CTRL
UpdateIR
ClockIR
ShiftIR
IR
DECODER
Actuele Instructie
IR PARALLEL IN/PARALLEL UIT
IR SCHUIFREGISTER
LSB
'1'
'0'
(verplicht)
(facultatief)
Design Specific Data
TDI

Sample/preload



4 INSTRUCTIES

operatiemode	functie	instructie
NORMAL MODE OY	INSTRUCTIES INLEZEN	de vorige instructie
	SAMPLES NEMEN	Sample/Preload
	TESTVECTOREN INLEZEN	Sample/Preload
	DEVICE IDENTIFICATION CODE	IDcode
	BS REGISTER IN BYPASS	Bypass
TEST MODE CY	INTERN TESTEN	INtest
	EXTERN TESTEN	EXtest
	ZELFTEST	RunBIST



EXtest

CAPTURE

On-Chip Logic

ShiftDR ClockDR UpdateDR Mode

OUTPUT PIN

COPPER TRACK

receiver

INPUT PIN '1'

ShiftDR ClockDR UpdateDR Mode

On-Chip Logic

SHIFT

On-Chip Logic

ShiftDR ClockDR UpdateDR Mode=1

OUTPUT PIN

COPPER TRACK

INPUT PIN

ShiftDR ClockDR UpdateDR Mode=1

On-Chip Logic

UPDATE

transmitter

On-Chip Logic

ShiftDR ClockDR UpdateDR Mode=1

'1' OUTPUT PIN

COPPER TRACK

INPUT PIN

ShiftDR ClockDR UpdateDR Mode=1

On-Chip Logic

EVALUATIE

Nadelen (−)

- Meer **Silicium**
- **Delays** aan de poorten (td mux)
- Testen gebeurt **statisch** en de testvectoren worden **serieël** aangeboden (testduur ↗)
- Systeem moet onder **spanning** staan vooraleer er getest kan worden (niet nodig bij "bed of nails").

Voordelen (+)

- **Universele** testmethode voor digitale schakelingen
- **CY** en **OY** ↗↗ (IC intern)
- **Testkost** daalt drastisch ("bed of nails" = peperduur)
- Systeem en test worden **parallel** ontwikkeld (EXtest)
- Ook bruikbaar voor **kleine volumes** (ASICS)

ALGEMEEN

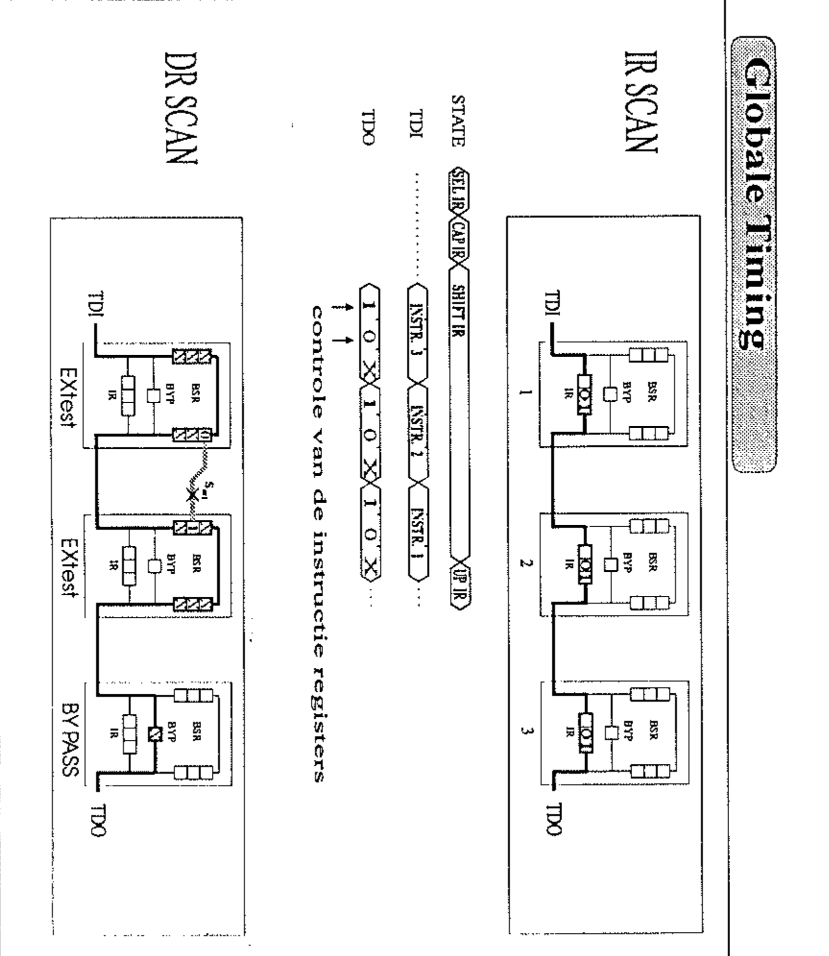
In combinatie met "interne scan technieken" en "Built in self test" is Boundary Scan een uitermate krachtig instrument om te testen op alle niveaus.

VISIE (Philips)

BS beïnvloedt alle fasen van de levenscyclus van een product.
BS is meer dan een introductie van een nieuwe design technologie.
BS is een integrale managements benadering die de globale strategie van een bedrijf omvat.

TESTKOST -50% **PCB PRODUCTIE -70%**

Globale Timing

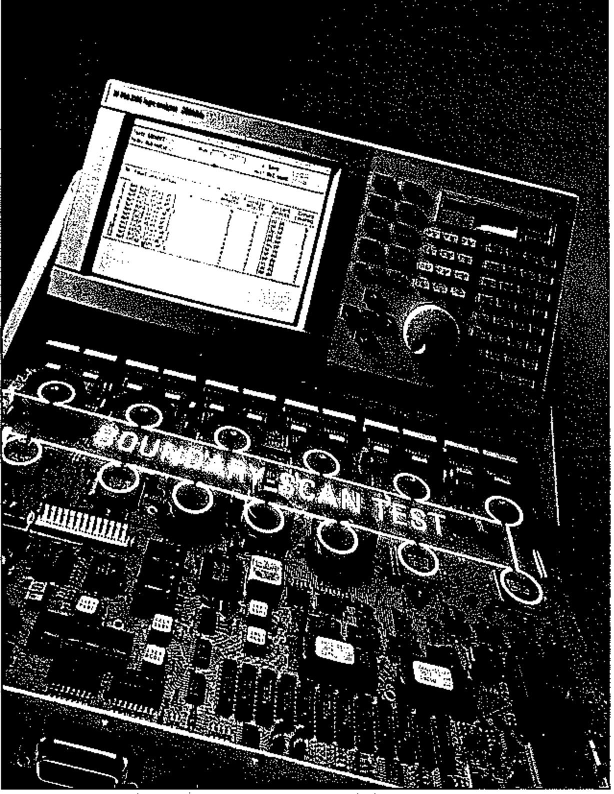


Philips Test & Measurement
Marketing Boundary-Scan Test Products
Building TQ III-4
P.O. Box 218
5600 MD Eindhoven
Tel. 31-40-782584
Fax 31-40-784559

Printed in the USA
Data subject to alterations without notice
9498 718 04313

PHILIPS

The ABCs of Boundary-Scan Test



PHILIPS

…me: ______

Title: ______

…mpany: ______

…dress: ______

______ Postcode: ______

…: ______ Fax: ______

…nciple company activity: ______

Philips Test and Measurement
Marketing Boundary-Scan
Test Products
Building TQIII-4
P.O. Box 218
5600 MD Eindhoven
The Netherlands

Philips Test and Measurement
Marketing Boundary-Scan
Test Products
Building TQIII-4
P.O. Box 218
5600 MD Eindhoven
The Netherlands

Preface

The present way of testing Printed Circuit Boards (PCBs) by means of bed-of-nails fixtures has reached technological limits. The ever-increasing miniaturization of PCBs and the growing use of complex ICs, such as ASICs has created a problem for board testing.

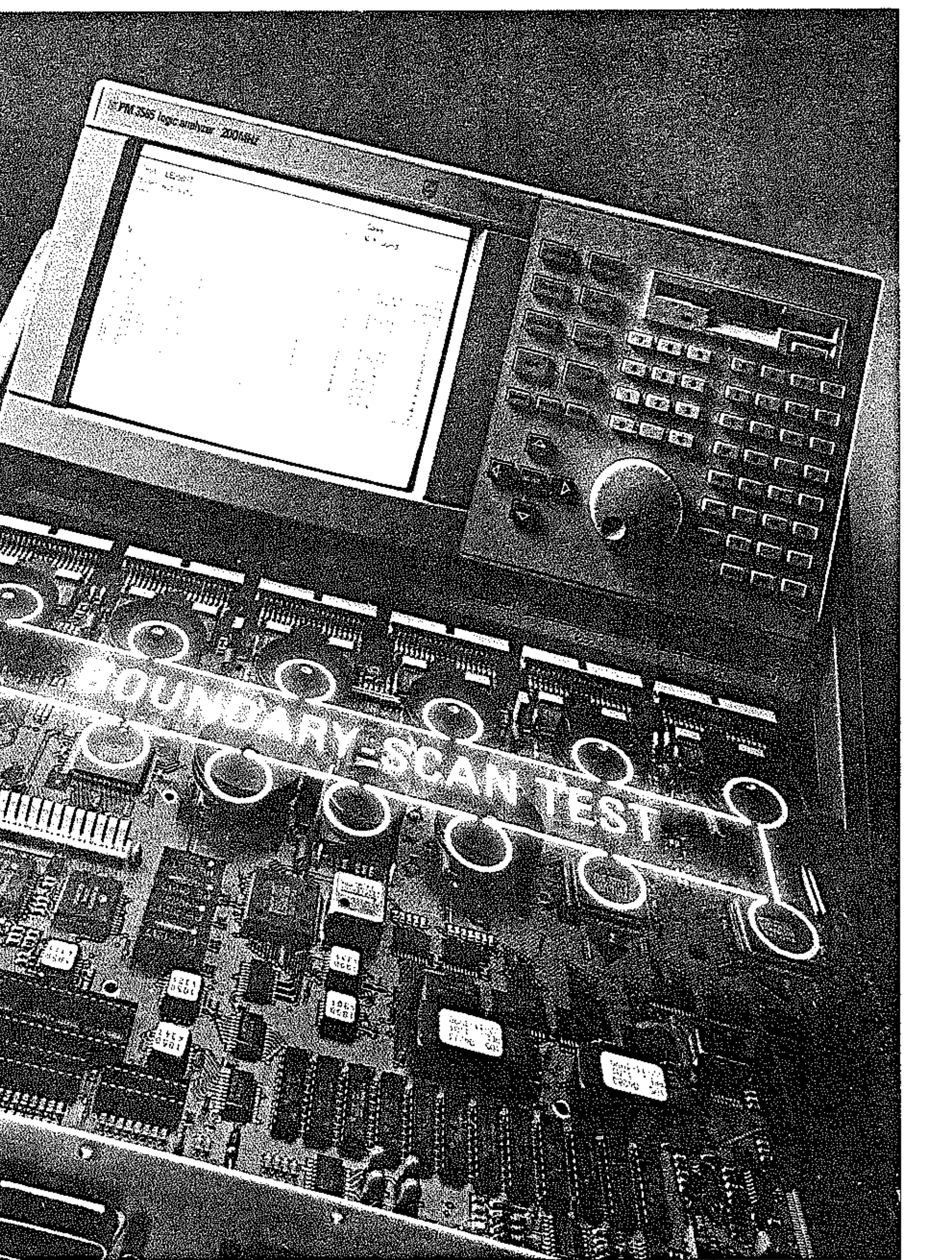
Boundary-Scan Test (BST) technology is a new approach to PCB testing. It offers access to the input and output pins of complex digital circuits on a PCB by means of a test bus which may consist of merely four wires, thereby providing an effective means for board level and system-level testing.

BST is a design-for-test (DFT) philosophy which frees the designers to incorporate the latest PCB and IC technology without the fear of making the PCB non-testable. It reduces time-to-market by saving prototype debug time and reducing production test set-up times. In the factory, lower test preparation time, time savings in fault diagnosis, and much cheaper test equipment lead to a reduction in test costs of at least 50%. As a consequence, manufacturing costs are lowered and throughput of the factory is increased. Also, in the field service phase tremendous savings are made in the prices of test equipment and test preparation times and with better coverage, spare board inventory can be reduced.

Since Boundary-Scan Test affects all phases of a Product Life Cycle, **it is more than just the introduction of a new design technology; it is an integral management approach. It concerns the global strategy of a company.** An *integral approach* towards introducing Boundary-Scan Test for PCBs certainly will help an organization to realize the full benefits of this technology.

The implementation of this new DFT technology requires some extra investment in chip design. The payback comes not only from reduced manufacturing costs, but also from reduced expense in service and design phases of the PCB. The overall result may be a cost reduction in PCB production of as much as 70%, after accounting for the extra chip development costs. This makes the Boundary-Scan technology profitable even for small volume production.

The "ABCs of Boundary-Scan Test" is meant to give a first insight into the new approach of PCB testing. It is written from the extensive experience gained in this field by the Fluke and Philips Electronics Alliance. Philips pioneered the development of the boundary-scan IEEE Std. 1149.1 and has been among the first to actually use boundary-scan technology in commercial products. Based on this experience, Fluke and Philips now offer software and hardware tools for Boundary-Scan Tests.



PM 3585 contains BST technology and offers BST facilities

Table of Contents

Why Boundary-Scan Testing? 1
- Introduction 1
- Board Density and Component Miniaturization 2
- Road Blocks for Conventional Test Methods 3
- Breakthrough in PCB Testing: Using the Principles of IC Testing 3
- Development of the BST Standard 4
- Philips Test Equipment and BST 4

What is Boundary-Scan Test 6
- Boundary-Scan IC 6
- Boundary-Scan at PCB Level 6
- Functions of a Boundary-Scan Cell 6

The Test Logic Architecture 8
- The Test Logic Elements 8
- Test Access Port 9
- TAP Controller 11
- The Instruction Register 15
- Instruction Register Operation in Each Controller State 16
- Test Data Registers (General) 17
- Operation of the Test Data Register Enabled for Shifting 18
- The Bypass Register 18
- The Boundary-Scan Register 19
- The Device ID Register 23
- Design-Specific Registers 25

Instructions 26
- General 26
- The BYPASS Instruction 27
- The SAMPLE/PRELOAD Instruction 27
- The EXTEST Instruction 29
- The INTEST Instruction 30
- The RUNBIST Instruction 31
- The IDCODE Instruction 32
- The USERCODE Instruction 33

Applications 34
- Overview 34
- Boundary-Scan Test Fault Coverage 34
- Testing of Mixed Technology PCBs 35
- PCB Life Cycle and Fault Classes 35
- A Boundary-Scan Test Feasibility Study 37
- Introduction to PF 8660/30 37

Index 43

Why Boundary-Scan Testing?

Introduction

Manufacturers of high-tech electronic products face higher testing costs as the complexity of the printed circuit boards (PCBs) increases. The main testing problem caused by the increased complexity is inaccessibility of test nodes via conventional techniques such as bed-of-nails fixtures and mechanical probes. As a result, current in-circuit testers with bed-of-nail fixtures are becoming less effective and are rapidly reaching their technological limits, forcing manufacturers to rely more heavily on expensive functional test techniques to maintain high product quality. This lack of accessibility of test nodes has two main causes:

1. *Miniaturization*
   - The trace distances and widths on the PCB are dropping to 100 µm.
   - The pitches of IC package pins are dropping from 2.5 mm (0.1 inch) to 0.3 mm.
   - The introduction of Surface Mounted Devices (SMDs) on both sides of the PCB have made it practically impossible to access them with bed-of-nails fixtures.
2. *Complex ICs*
   - ASICs are currently packaged in housings with 200 pins and more.
   - VLSI microprocessors may contain more than one million transistors.

To cope with these technology advances, automatic test equipment to test PCBs became so expensive in the late 1980s that it severely affected the profitability of producing PCBs.

As will be shown in subsequent chapters, the Boundary-Scan Test (BST) methodology is a direct response to the accessibility problem. BST originates from the design-for-test philosophy as applied in the IC technology and can dramatically reduce test costs.

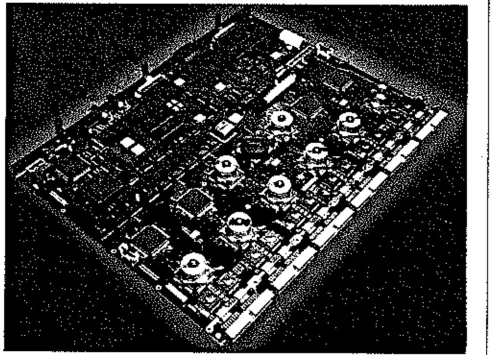
The utilization of Boundary-Scan Test requires provisions to be made in the design of the chips and of PCBs, adding some cost to the original design. The revenues are earned back many times over, however, due mainly to the simplified testing methods in the factory and in field service. The costs of both automatic test equipment and test set-up times are reduced drastically.

Reduction in production set-up time is not the only source of time-saving. BST also reduces design prototype debug time, thus improving the overall time-to-market for a product. In the highly competitive electronics market, with short product life-cycles, any improvement in time-to-market can significantly improve a company's market position.

In-circuit test site with bed-of-nails technology

Board Density and Component Miniaturization

Printed circuit boards (PCBs) add the most value to electronics hardware. Over the years, PCBs have become more and more "loaded."

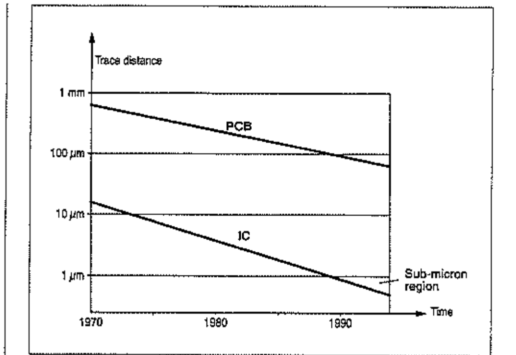


A printed circuit board containing ASICs

To start with, the number of integrated circuits (ICs) per PCB has increased, leading to narrower copper tracks and smaller distances between them.

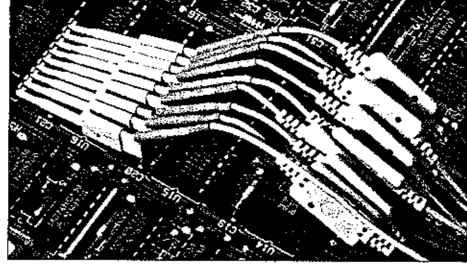
At the same time, miniaturization within the ICs itself has progressed — the sub-micron technology has arrived. This has resulted in a very large number of functions per chip, needing many more pins per IC and bigger packages with more leads (>500 pins) at smaller pitches (0.3 mm). Direct mounting of chips on PCBs is happening with Tape Automated Bonding (TAB), Chips On Board (COB) etc.

The next figure shows the reduction of trace distances on PCBs and in ICs.



Reduction of trace distances on PCBs and in ICs

This miniaturization has made it more and more difficult to test these "highly loaded" PCBs with in-circuit testing. With *in-circuit testing*, physical contact is made with the components (ICs) and the copper tracks on the PCB by means of probes or bed-of-nails technology. Input signals are applied to a component's input pins, and the response at the output pins is examined.



Conventional testing IC responses with probes (in field service)

Road Blocks for Conventional Test Methods

It is clear that, given the ongoing miniaturization (with IC pin pitches of down to even 0.3 mm or less), probing technology is no longer feasible from the mechanical point of view. The PCB technology, such as surface mounted devices (SMDs) on both sides of the PCB, also makes probing practically impossible.

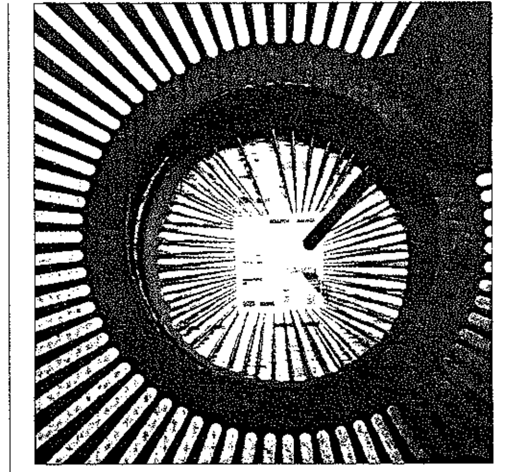
The advantages of In-Circuit Testing and the use of standard Test Pattern Sets for ICs have diminished due to the increasing application of ASICs. Each ASIC requires a custom test patterns set, which can take a long time to generate. VLSIs usually require long test sequences with extended test times; if applied with backdrive, these extended test times may impact the reliability of the PCBs.

In summary, the programming of in-circuit automatic test equipment (ATE), the quality of conventional testing and bed-of-nails probing are reaching the limit of feasibility.

Breakthrough in PCB Testing: Using the Principles of IC Testing

The solution to the problem comes from the IC test technology.

Today, ICs are tested in a functional manner, applying the design-for-test methodology. This means that *on the chip itself* arrangements have been made to allow for built-in testing; that is, a built-in self-test feature or a scan design feature.



Mechanical probes to test ICs are largely replaced by Boundary-Scan cells

The idea was that the principles of IC testing could be partially extended to PCB level, and eventually to system level.

The value of the scan test technology as an approach to loaded-board testing was first recognized by Philips in 1985. The scan test was extended to a Boundary-Scan Test as a variant of scan test for PCB level.

The Boundary-Scan technique involves the inclusion of a shift-register stage (contained in a boundary-scan cell) adjacent to each component pin so that signals at component boundaries can be controlled and observed using scan testing techniques. Thus, provisions have to be made at *component* level (IC) in order to use Boundary-Scan Testing at a higher system level, such as PCB, instrument box and integral system level.

Philips has proven that this idea can be realized.

The last chapter describes the feasibility of Boundary-Scan Test for testing PCBs and shows the results obtained with currently available products.

Development of the BST Standard

The PCB test problem affected many companies, in addition to Philips The scan techniques were already applied elsewhere, but some companies had developed their own architecture. And since proprietary solutions are usually not interchangeable, the need for a *standard* architecture was apparent.

In 1985 Philips Electronics took the lead and formed the Joint European Test Action Group (JETAG), a group of key electronics manufacturers in Europe. The aim was to reach an *industry standard* for loaded-board testing. Companies that participated with Philips were British Telecom, Bull, Ericsson, Nixdorf, Siemens, Thomson and others.

Later, North American companies like Texas Instruments, AT&T, DEC and IBM joined the group and the JETAG, no longer specifically European, became JTAG, chaired by Philips.

In 1987 the JTAG architecture version 1.0 for loaded-board testing was proposed, followed in 1988 by version 2.0. The latter version was submitted as a standard architecture to the IEEE Computer Society's Test Technology Committee. In February 1990 the document titled:*IEEE Standard Test Port and Boundary-Scan Architecture* was approved by the IEEE Standards Board, as IEEE Std 1149.1-1990. In August 1990 the American National Standards Institute (ANSI) also recognized this standard.

Philips Test Equipment and BST

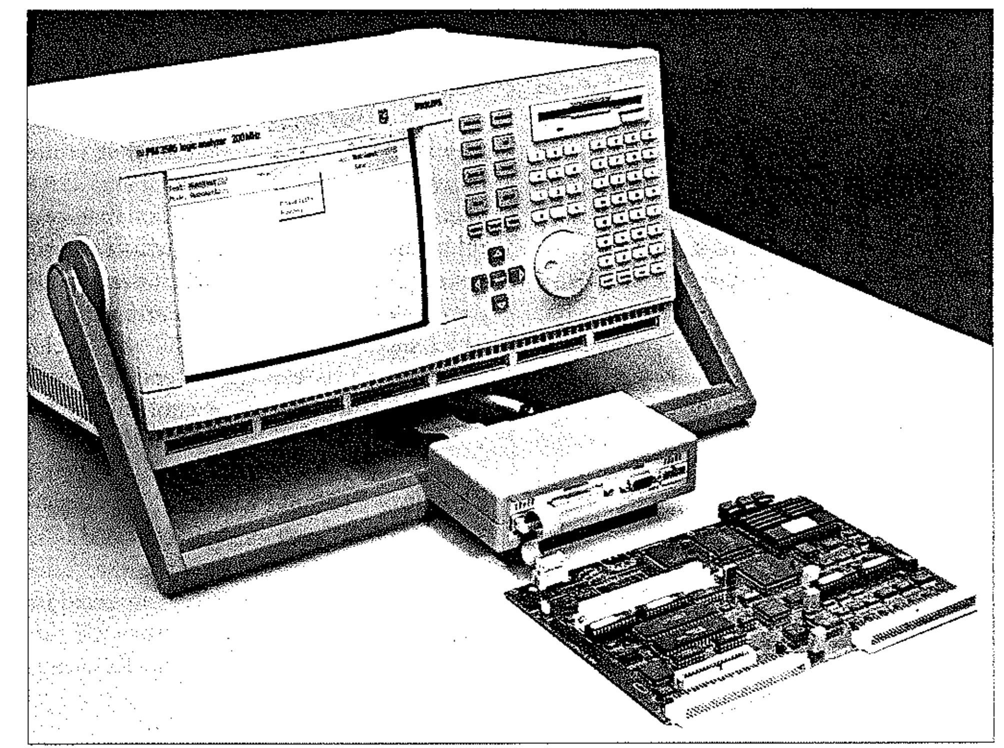
It should not be surprising that Philips is one of the first to market test equipment supporting Boundary-Scan technology.

Philips has practical implementation experience in BST with its PM 3580 family of logic analyzers in which the PCB circuitry has been adapted for testing according to the Boundary-Scan standard IEEE 1149.1-1990.

As a spin-off, the Boundary-Scan Test Option PF 8660/30 for the PM 3580 Family of logic analyzers is now available.
This Option consists of three elements:

1. a hardware adapter box
2. the test pattern generation software, which runs on a PC
3. the test execution and diagnosis software, which runs on the logic analyzer.

The operation of the PF 8660/30 Option is detailed in the last chapter.



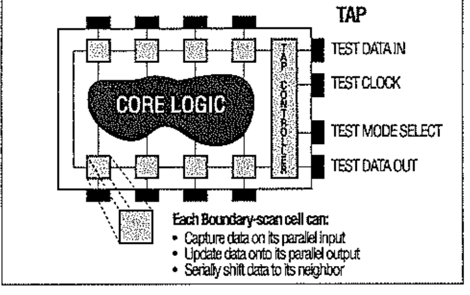
PF 8660/30 Option connected to PCB for Boundary-Scan Test

What is Boundary-Scan Test

Boundary-Scan Test (BST) is a design-for-testability technique which resolves the test node accessibility problem. In a typical boundary-scan IC—comprising of core logic and input- and output buffers—a shift-register stage is placed between the core logic and the input and output buffers adjacent to each IC pin. Each shift-register stage is contained in a boundary-scan cell (BSC). The BSCs can control and observe what happens at each input and output pin of an IC.

Boundary-Scan IC

The BSCs for the pins of a component are interconnected to form a shift-register chain, called the boundary-scan register. The boundary-scan register provides a serial path which surrounds the core logic. This path is provided with a test data input (TDI) and test data output (TDO) connections and appropriate clock and control signals. In such an arrangement, test data can be shifted through the boundary-scan path from TDI to TDO to achieve test node accessibility.

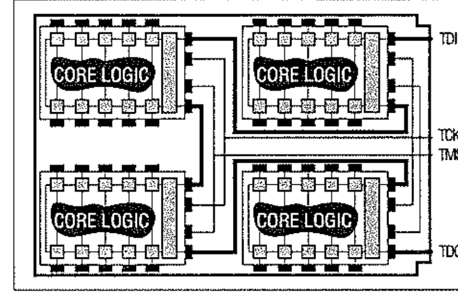


Typical Boundary-Scan IC

Other registers within a boundary-scan IC include: an Instruction Register used to select the test to be performed or the register to be accessed or both; a Bypass Register which provides a one-bit path that minimizes the distance between TDI and TDO pins; an optional IC Identification Register, called the IDCODE Register, which identifies the IC and manufacturer; other optional design-specific test data registers which support features in the IC such as self-tests, scan paths, etc. The operation of the boundary-scan circuitry is controlled by a Test Access Port (TAP) Controller.

Boundary-Scan at PCB Level

Within a PCB assembled from several ICs, the boundary-scan registers for the individual components could be connected in series to form a single path through the complete design, as shown in the following figure. Alternatively, a board design could contain several independent boundary-scan paths.



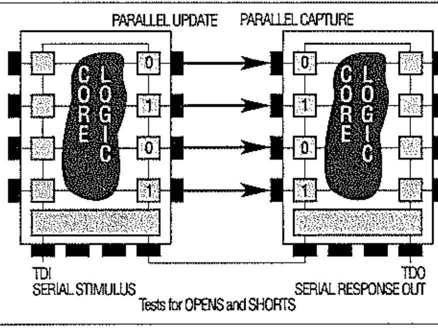
Typical Boundary-Scan PCB

Functions of a Boundary-Scan Cell

During normal IC operation, BSCs are transparent. When the IC is placed in a scan mode, the BSCs are configured to allow test stimulus to be shifted in and applied from each BSC output. Test response data is captured into each BSC input and shifted out for inspection. Depending on the control signals, each BSC either captures data on its input, updates data on its output, or serially shifts data to its neighbor.

Interconnect and Cluster Test

External testing of wiring interconnects and neighboring ICs is accomplished by applying a test stimulus from the BSCs which are associated with the output pins of the appropriate ICs and capturing the response in the BSCs which are associated with the input pins of the ICs.

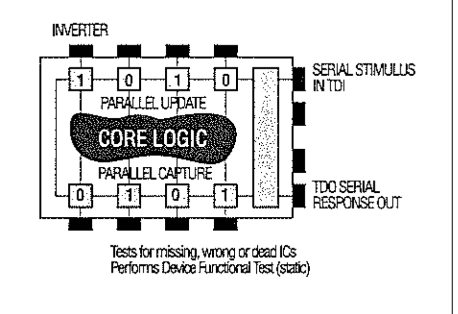


Interconnect Test

Internal IC Test

Internal testing of a IC core logic is accomplished by isolating the on-chip core logic from stimuli received from surrounding components while an internal self test is performed.

Alternatively, if the boundary-scan register is suitably designed, it can permit a limited slow-speed static testing of IC core logic. It does this by applying a test stimulus from the BSCs which are associated with the input pins of the IC and capturing a test response of the core logic from the BSCs which are associated with the output pins of the IC.



IC Test

The Test Logic Architecture

The Test Logic Elements

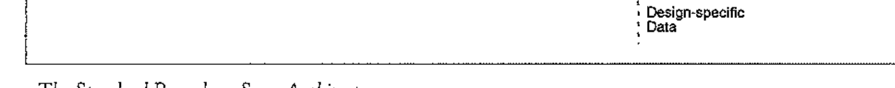
The overall test architecture, as prescribed by the IEEE 1149.1 standard, distinguishes four basic elements:

1. the Test Access Port (TAP)
2. the TAP Controller
3. the Instruction Register (IR)
4. a group of Test Data Registers (TDRs).

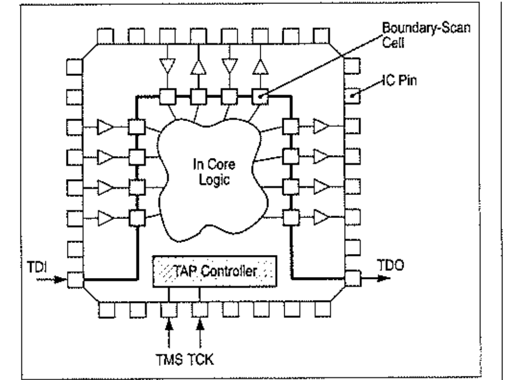
The Instruction Register and the group of Test Data Registers have separate, alternative shift-register based serial access paths, connected in parallel between the TDI and TDO pins. Selection of the alternative paths is made under control of the TAP Controller.

The next figure gives an overview of the standard boundary-scan architecture.
In the following sections, each of the elements will be described in more detail.

As mentioned in the "What is BST?" section, the boundary-scan technique involves the inclusion of a series of test access logic cells connected by shift-register stages. There is a cell for each IC pin in use. Such cells are called Boundary-Scan Cells (BSC). The whole series of BSCs is called the Boundary-Scan Register (BSR).



The Standard Boundary-Scan Architecture
Note: The optional registers and signals are drawn with dotted lines.



IC including Boundary-Scan Register

Note that both the serial TDI/TDO signal and the parallel pin-to-logic signal are passed through the boundary-scan cell.
All system (IC) input and output signals sampled by the boundary-scan cells are shifted in and out via the serial TDI/TDO path.

The operation of the boundary-scan cells is controlled by signals from the Test Access Port (TAP).
The TAP controller itself is controlled by the signals Test Mode Select (TMS) and Test Clock (TCK), obtained from a system-level bus or from automatic test equipment (ATE).
According to the IEEE 1149.1 standard, an optional test reset input signal (TRST*) may also be applied.
Details are described in a later section.

Test Access Port

The TAP provides access to many test support functions built into an IC. It consists of four input connections, one of which is optional, and one output connection. The optional input connection is used to reset asynchronously the test logic defined by the BST standard. However, this function can also be performed by logic circuitry built into the TAP Controller.

The IEEE 1149.1 standard requires that TAP connections are not used for any purpose other than testing.

The following connections form the Test Access Port.

- *The Test Clock Input (TCK)*
  TCK provides the clock for the test logic.
  A PCB may include various components (ICs) that contain a component-specific clock. To remain independent of such (different) clock frequencies, the TCK must not interfere with any system clock. This permits shifting of test data into or out of the Boundary-Scan register cells concurrently with the system operation of the component and without interfering with the on-chip system logic.

- *The Test Mode Select Input (TMS)*
  The logic signals (0s and 1s) received at the TMS input are interpreted by the TAP Controller to control the test operations. The TMS signals are

sampled at the rising edge (positive slope) of the TCK pulses. When TMS is not driven from an external source, the test logic perceives a logic 1.

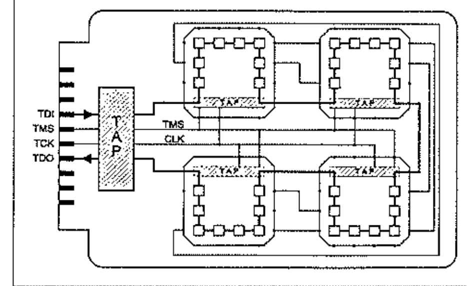
- *The Test Data Input (TDI)*
  Serial input data applied to this port is fed either into the instruction register or into a test data register, depending on the sequence previously applied to the TMS input. Both registers are described in a subsequent section. The received input data is sampled at the rising edge (positive slope) of the TCK pulses. When TDI is not driven from an external source, the test logic perceives a logic 1.

- *The Test Data Output (TDO)*
  Depending on the sequence previously applied to the TMS input, the contents of either the instruction register or a data register are serially shifted out towards the TDO. The data out of the TDO is clocked at the falling edge (negative slope) of the TCK pulses. When no data is shifted through the cells, the TDO driver is set to an inactive state, for example to high impedance.

- *The Test Reset Input (TRST*)*
  The TAP's test logic is asynchronously reset when a logic 0 is applied to this port. This is an *optional* port to the TAP, because normally the test logic is designed so that it can be reset under control of the TMS and TCK signals.
  Furthermore, the latter option saves one pin on the IC. So, only four additional pins, instead of five, have to be added to the IC package for the purpose of boundary-scan testing.

The TAP input and output connections of the ICs can be interconnected at PCB level in a way that the appropriate tests can be performed.

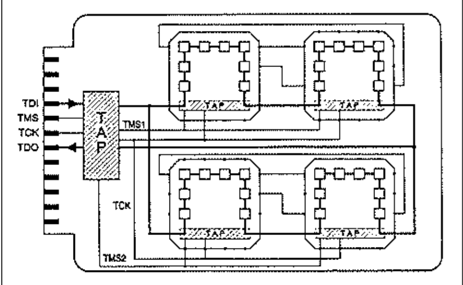
In the next figure, all IC Test Access Ports are connected in series for the TDI/TDO path.



PCB configuration with all IC BSRs connected in series.

Note that the control signals TMS and TCK are connected in parallel to each IC while all test data pins (TDI and TDO) are connected as a daisy chain to form a single serial path at the PCB level.

Other PCB test configurations are also possible. The next figure shows an example in which two serial paths for the test data pins are connected in parallel.



PCB configuration with two parallel test paths

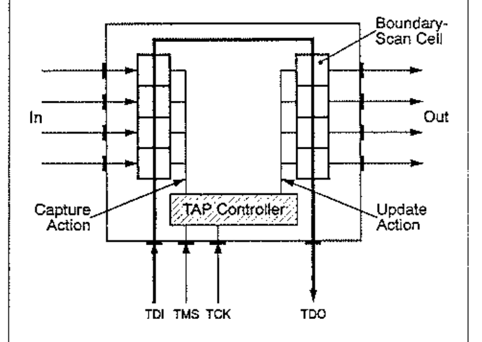
The parallel configuration uses a pair of coordinated TMS signals (TMS1 and TMS2). This set-up can be controlled such that only one serial path is tested at a time. This requires a tri-state circuitry at the TDO output pins; only the ICs delivering TDO signals should be in the active drive state.

With the TAP control signals it is possible to isolate (all) IC pins from the PCB.
Two types of test measurement are then possible.
First, the circuitry around the IC can be tested, i.e. the connecting leads (net) can be tested for example for opens or shorts, by applying test signals to the output pins of one IC and analyzing the results through the input pins of the surrounding ICs.
Secondly, internal IC tests can be initiated, for example a built-in self test. The test results are sampled into the registers of the output boundary-scan cells. These results are shifted through the boundary-scan cells towards the TAP's TDO pin.

TAP Controller

The function of the TAP Controller is to generate clock and control signals required for the correct operation of the circuitry connected to its output, that is, the Instruction Register and the Test Data Registers.

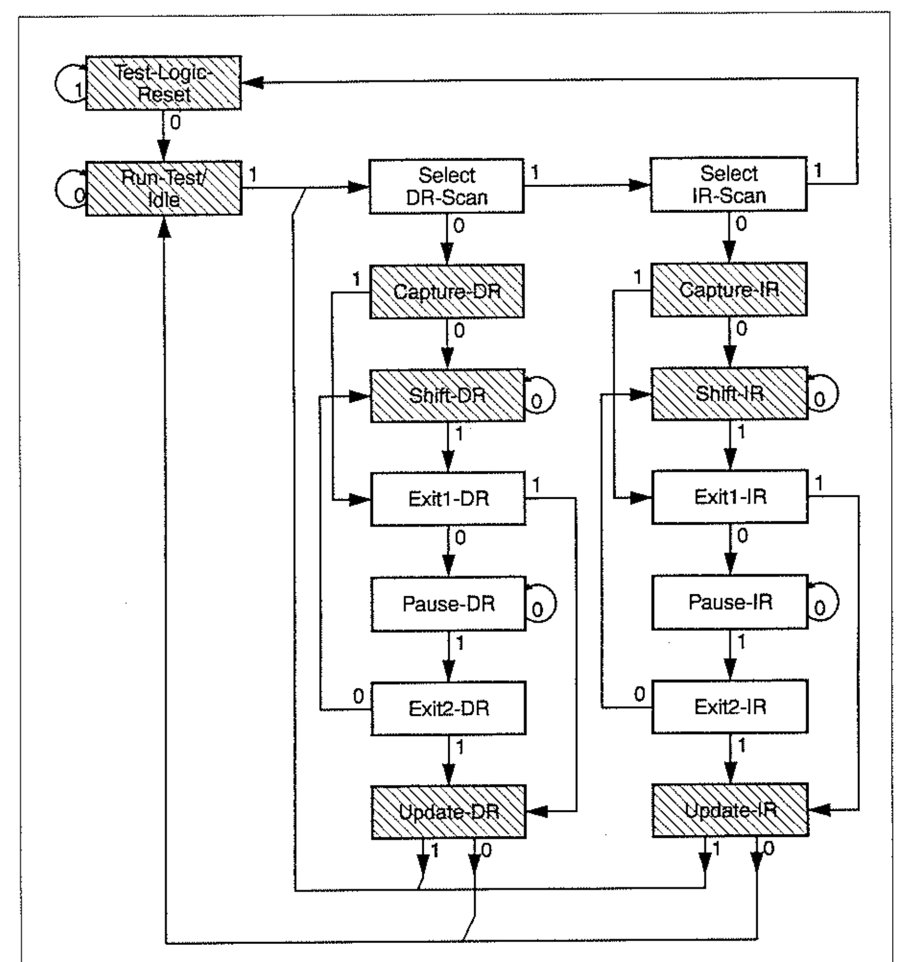
To provide boundary-scan test on an IC,a shift-register based Boundary-Scan Cell is connected to each pin. All cells are connected in series to form a Boundary-Scan Register.
The figure below symbolizes the basic principle.



Boundary-Scan Register actions are controlled by the TAP Controller

The TAP Controller is a finite state machine that operates synchronously with the TCK input. It responds to the TMS and TCK signals to generate the control signals required to shift, capture or update data through either the Instruction Register or the addressed Test Data Register. (see figure above).

The next figure shows the 16-state diagram for the TAP Controller as defined by the IEEE 1149.1 standard.



State Diagram of the TAP Controller
Note: The value shown adjacent to each state transition represents the TMS signal level at the time of a rising edge of the TCK.

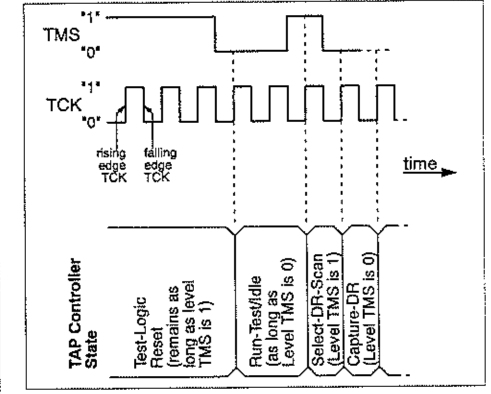
All state transitions within the TAP Controller occur at the rising edge of the TCK pulse, whereas actions in the connected test logic (registers etc.) occur at either the rising or the falling edge of the TCK.

The state diagram can be read as follows.
Suppose the TAP Controller is in its *Test-Logic-Reset* state.
As long as the TMS is held at 1, the controller remains in this state (see diagram).
In order to leave this state, the TMS is set to 0. Then, at the next rising edge (positive slope) of the TCK pulse, the TAP Controller leaves *Test-Logic-Reset* state, and enters the *Run-Test/Idle* state.

Here again (see diagram), as long as TMS is held at 0, the controller remains in the *Run-Test/Idle* state.
If TMS is now set to 1, then at the next rising edge of the TCK pulse, the *Select-DR-Scan* state is entered.

This lasts only for one TCK clock pulse because, depending on the value of the TMS signal level, either the *Select-IR-Scan* state is entered (TMS level is 1) or the *Capture-DR* state is entered (TMS value is 0).
And so on.

The next figure shows a timing diagram of the sequences described above.



Timing diagram of a few states of the TAP Controller

All controller states shown as cross-hatched in the state diagram control the connected *test* logic.

The actions performed in each of the cross-hatched states are the following.

- *Test-Logic-Reset*
  In this controller state all *test* logic is disabled, i.e. all on-chip system logic operates normally.
  Whatever may be the original state of the controller, it will enter the *Test-Logic-Reset* when TMS signal is held high (logic 1) for at least five rising edges of the TCK pulse. The exact required number of TCK pulses depends on the current state of the TAP Controller. The controller remains in this state while TMS is held high.

If the optional signal TRST* is present in the TAP, it can be used to force the controller to the *Test-Logic-Reset* state at once, at any moment during circuit operation.

- *Run-Test/Idle*
  This is a controller state between scan operations. Once the controller is in this state, it will stay there as long as the TMS signal is held low (logic 0).
  The operation of the connected test logic depends on the instructions contained in the instruction register. For example, the RUNBIST instruction causes the on-chip built-in self-test to execute. If an instruction does not cause test functions to execute, such as select a data register to scan, then all selected test data registers will stay *Idle* and retain their previous state.
  The current instruction does not change while the controller is in this state.

- *Capture-DR*
  In this controller state, data is parallel-loaded from the parallel inputs into the selected test data register. The action takes place at the rising edge (positive slope) of the TCK pulses.
  The current instruction does not change in this controller state.

- *Shift-DR*
  In this controller state, the previously captured data is shifted via TDI and out via TDO, one shift-register stage on each rising edge (positive slope) of the TCK pulse.
  The current instruction does not change in this controller state.

- *Update-DR*
  Once the controller is in this state, the shifting process has been completed.
  Test data registers may be provided with a latched parallel output. This prevents the parallel output from changing while data is shifted into the associated shift-register path.
  When these test data registers are selected by an instruction, the new data is latched into their parallel outputs in this state at the falling edge (negative slope) of the TCK pulses.
  The current instruction does not change in this controller state.

- *Capture-IR*
  In this controller state, the instruction data is parallel-loaded into the shift register stage of the instruction register. The action takes place at the rising edge (positive slope) of the TCK pulses.
  The parallel output stage of the instruction register retains its previous state. Test data registers which are selected by the current instruction retain their previous state.

- *Shift-IR*
  In this controller state, the previously captured data is shifted in the TDI/TDO path towards the output, one shift-register stage on each rising edge (positive slope) of the TCK pulse.
  The current instruction does not change in this controller state.

- *Update-IR*
  The shifted-in instruction data is loaded from the shift-register stage into the parallel output of the instruction register. The new instruction becomes valid when the TAP Controller is in this state.
  All the test data shift-register stages which are selected by the current instruction retain their previous state.
  The action takes place at the falling edge (negative slope) of the TCK pulses.

The *non-cross-hatched* states in the state diagram do not affect the connected test logic, which then keeps its state.
Of these non-cross-hatched states, *Pause-DR* and *Pause-IR* are provided to temporarily halt the shifting process. This might be needed to allow automatic test equipment to fetch data from (disk-)memory. These 'Pause' states are maintained as long as the level of the TMS signal is held low (logic 0).
The remaining non-cross-hatched states in the figure (*Select-DR-Scan*, *Select-IR-Scan*, *Exit1-DR*, *Exit1-IR*, *Exit2-DR* and *Exit2-IR*) are used to determine the route to be followed in the state diagram, as explained along with the figure.

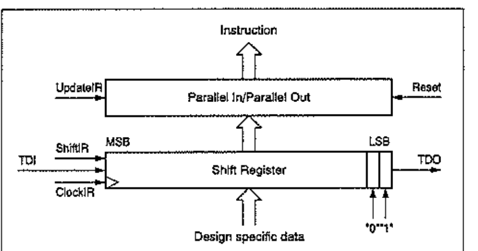
The Instruction Register

The Instruction Register allows test instructions to be shifted into each IC along the PCB-level path. At board level, all IC instruction registers are connected in series in the *Shift-IR* controller state. The test instruction defines the test to be performed, or the test data register to be addressed.

The design of the Instruction Register is a serial-in parallel-out register. Each Instruction Register cell has a shift register flip-flop and a parallel output latch. The shift register flip-flop holds the instruction bit moving through the Instruction Register. The parallel output latch holds the current instruction.
The instructions shifted into the shift register flip-flops are latched into the parallel output latches. The latched instruction can only change when the TAP Controller is in its *Update-IR* or *Test-Logic-Reset* state.

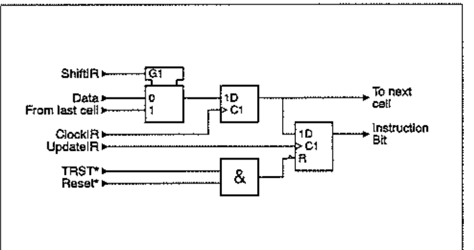
The Instruction Register must contain at least two shift-register-based cells which can hold instruction data. These two mandatory cells are located nearest to the serial outputs, i.e. they are the least significant bits. The values of the bits are 0 and 1 (the 1 is the least significant bit). These fixed values are used in locating faults in the serial path through the ICs on the PCB.

The next figure shows the set-up of an Instruction Register.



Set-up of an Instruction Register

In the next figure an example of one cell of the Instruction Register is shown. The input signals come from the TAP Controller (refer to figure of standard Boundary-Scan Architecture on page 8).



An example of an Instruction Register cell

The next table shows the operation of the Instruction Register in each of the states of the TAP Controller.

Instruction Register Operation in Each Controller State

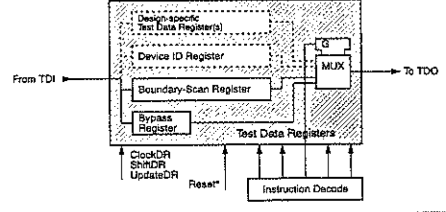
Controller State	Shift-Register Stage	Parallel Output
Test-Logic-Reset	*Undefined*	*Set to give the IDCODE (or BYPASS) instruction*
Capture-IR	*Load 01 into LSBs and design-specific data or fixed values into MSBs*	*Retain last state*
Shift-IR	*Shift towards serial output*	*Retain last state*
Exit1-IR	*Retain last state*	*Retain last state*
Exit2-IR	*Retain last state*	*Retain last state*
Pause-IR	*Retain last state*	*Retain last state*
Update-IR	*Retain last state*	*Load from shift-register stage*
All other states	*Undefined*	*Retain last state*

Referring back to the description of the TAP Controller state diagram on page 12, it can be noted from the table that indeed the *Exit1-IR*, *Exit2-IR* and the *Pause-IR* states do not influence the operation of the test logic.

Test Data Registers (General)

The Boundary-Scan Architecture must contain a minimum of two test data registers: the Bypass Register and the Boundary-Scan Register. A third register, Device Indentification Register, is optional. Each Test Data Register included in the architecture is addressable through an instruction code that can be scanned into the Instruction Register. The addition of a design-specific Test Data Register is permissible and is accomplished by adding an instruction code to the architecture's test-instruction set; these registers may (but need not) be publicly accessible.

The next figure shows a detail of the previous figure for the standard Boundary-Scan Architecture on page 8.

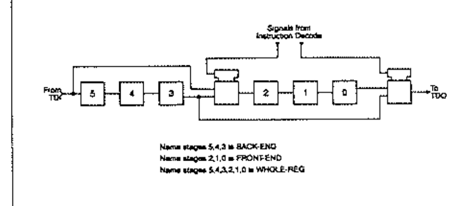


Overview of the Test Data Registers
Note: The allowed options are shown in dotted lines.

The specifics of the Test Data Registers are described in following sections. In this section, some of the *common* design requirements are described.

- Each Test Data Register must have a unique name.
- In the *Shift-DR* state of the TAP Controller the data applied to the TDI is shifted towards the TDO, which must be done without inversion of the data.
- The length of any Test Data Register must be fixed, independent of the instruction by which it is accessed. The registers can be concatenated to form further registers, provided that each distinct combination is given a new name.

The next figure shows an example.



- An instruction may select a test operation in which more than one test data register is involved. This must then be done sequentially — step by step —because at any time there can be only one test data register between the TDI and the TDO in the *Shift-DR* controller state.
- Registers which are not selected for a test operation must be configured so that they do not interfere with the operation of the on-chip logic.

- Test results are sampled in the *Capture-DR* controller state, and new test stimuli (if any) become available in the *Update-DR* state of the TAP Controller.
- Where an 'internal' test execution (e.g. a self-test) is required, this must occur while the TAP Controller is in the *Run-Test/Idle* state.

To conclude this section, an overview of the Test Data Register operations selected as the serial path between the TDI and the TDO is given in the next table.

Operation of the Test Data Register Enabled for Shifting

Controller State	Action
Capture-DR	*Load data at parallel input into shift-register stage. Parallel output register or latch retains last state.*
Shift-DR	*Shift data towards serial output. Parallel output register or latch, retains state.*
Exit1-DR	*Retain last state.*
Exit2-DR	*Retain last state.*
Pause-DR	*Retain last state.*
Update-DR	*Load parallel output register or latch from shift-register stage. Shift-register stage retains state.*
All other controller states	*Registers which have a parallel output maintain their last state of the output; otherwise undefined.*

The Bypass Register

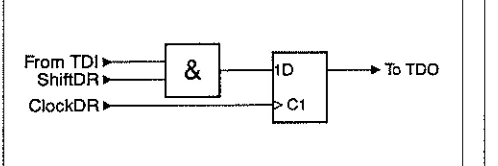
The mandatory Bypass Register is a single stage shift-register. When addressed, it provides a minimum length serial path for the test data shifting from the component's (IC) TDI to its TDO.

The benefit of the Bypass Register is a shortened scan path through the architecture when scan access of other Test Data Registers is not required.

Suppose a printed circuit board has 50 ICs. The Boundary-Scan registers of all ICs are connected into a single serial chain. If the average component register length is 100 stages, then the total serial path length contains 5000 stages. If only one IC is to be tested on that board, the remaining 49 ICs can be bypassed. For this test of one IC, only 100 stages plus 49 bypass stages are required. Now, only 149 instead of 5000 stages are involved. This reduces testing times considerably, and it does so in two ways:

1. The test cycle is shortened, in this example, by 97% (1-149/5000 X 100%).
2. Only the 100 bits for the IC under test are of importance. Hence, the rest of the bits need not be stored in the automatic test equipment, which shortens diagnosis times.

The next figure shows an example of an implementation of a bypass register or cell.



An Implementation of a Bypass Register
Note: that the register does not have a parallel data output latch.

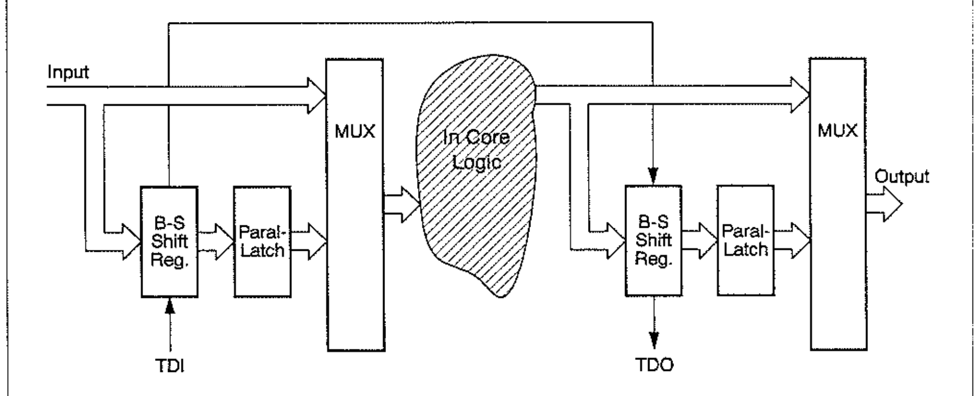
When the Bypass Register is selected, the shift-register stage is set to a logical zero on the rising edge (positive slope) of the TCK pulse when the TAP Controller is in its *Capture-DR* state.

The Boundary-Scan Register

The mandatory Boundary-Scan Register consists of a series of Boundary-Scan Cells arranged to form a scan path around the boundary of the core logic of the host IC. The Boundary-Scan Register provides the following features:

- It allows testing of circuitry external to the component (IC), which typically means the interconnect test.
- It allows self-testing of the on-chip logic (e.g. built-in self test), while also providing defined conditions at the periphery of the on-chip logic.
- It allows sampling and examination of the input and output signals without interfering with the operation of the on-chip logic.
- And last but not the least: it can stay idle, thereby showing virtually no load to the on-chip logic when signals are flowing through the system.

The global structure is shown schematically in the next figure.

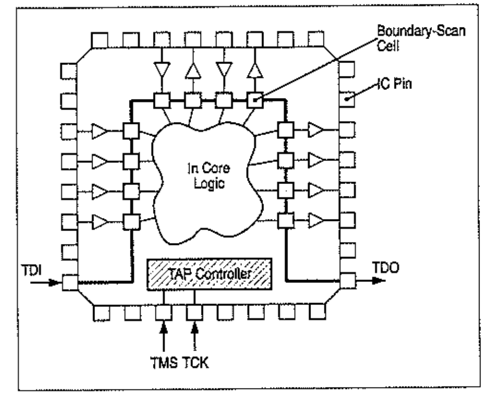


Overall structure of a Boundary-Scan Register in an IC
Note: The parallel latches prevent change of information while data is shifted in and out towards TDO.

According to the IEEE 1149.1 standard the following provisions for the Boundary-Scan Register are mandatory.

- A Boundary-Scan Register cell must be connected between each digital system pin and the on-chip logic.
- The applied Boundary-Scan cell must be able to observe the state of the signal at the component pin and, if applicable, also to control the on-chip system logic.
- A (combination of) Boundary-Scan cell(s) must be able to cope with signals at unidirectional and bidirectional system pins, at 2-state and open collector pins and at 3-state system output pins.
- For programmable components, the length of the Boundary-Scan Register must be independent of the way that the component is programmed.

The next figure shows schematically how the Boundary-Scan cells surround the on-chip logic (in-core logic). Control of the operation is provided by the TAP Controller, which has been described in a previous section.



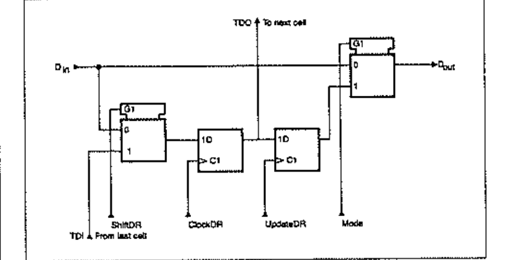
Configuration of IC with TAP Controller and Boundary-Scan Cells

A Boundary-Scan Cell

For ease of understanding, a 'universal' Boundary-Scan Cell (BSC) is considered, a configuration that can be used at both the input and the output pins of an IC. However, not all elements of the cell are always needed in a particular application.

The building blocks composing the cell can also be different.

The next figure shows an implementation of such a Boundary-Scan Cell.



An Implementation of a Boundary-Scan Cell

Dedicated Pin Cells

In addition to the mandatory overall specifications of the Boundary-Scan Register cells (see above), specific requirements are placed on cells connected to the various types of *component pins*. A number of these requirements are listed below, for each type of pin.

Note: The descriptions below refer to various instructions. These instructions are explained in the next chapter.

Cells at Input Pin

- Each cell must contain a shift-register stage.
- As a response to the *SAMPLE/ PRELOAD* or the *EXTEST* instructions, the shift-register stage loads the data presented at the pin on the rising edge (positive slope) of the TCK pulse when the TAP controller is in its *Capture-DR* state.
- A sample signal must not be inverted, i.e. a logic 0 applied to the input pin causes (later) a logic 0 to be shifted towards the TDO.
- Data presented at the input pin is supplied to the on-chip logic without modification by the cell, for example when the Boundary-Scan Register is not selected or after a *SAMPLE/ PRELOAD* instruction.
- The design of the cell must cause no interference with the operation of the on-chip self-test, when this is selected by the *RUNBIST* instruction.

Cells at 2-State Output Pin

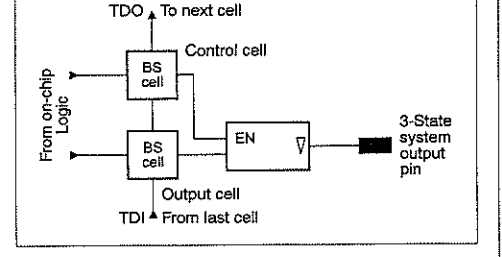
- Each cell must contain a shift-register stage with a latched parallel output.
- As a response to the *SAMPLE/ PRELOAD* or the *INTEST* instruction the shift-register stage loads the data output from the on-chip logic on the rising edge (positive slope) of the TCK pulse when the TAP Controller is in its *Capture-DR* stage.
- A sample signal must not be inverted, i.e. if the on-chip logic supplies a logic 0 signal, a logic 0 must also be shifted towards the TDO.
- Data generated by the on-chip logic is supplied to the output pin without modification by the cell, for example when the Boundary-Scan Register is not selected or after a *SAMPLE/ PRELOAD* instruction.
- When the *RUNBIST* instruction is selected, the cell must not cause any change in the signal driven to the output pin.
- As a response to the *EXTEST* or *INTEST* instruction, the signal driven to the system output pin must change at the falling edge (negative slope) of the TCK pulse when the TAP Controller is in its *Update-DR* state.
- The signals previously shifted into the cell during TDI must not change when they are driven to the output pin (a

logic 0 remains a logic 0). The applicable instructions are *EXTEST*, *INTEST* or *RUNBIST*.

Cells at 3-State Output Pin

- Two cells must control this pin: one to supply the data and one to select the active or inactive drive state.
- Both cells must meet the requirements for cells at the 2-State Output pins.
- When the *INTEST* or *RUNBIST* instructions are selected, then either one of the two following requirements must be met.
  1. The output pin is placed in the inactive drive state (e.g. high impedance).
  2. The last three requirements listed for the 2-State Output Pin must be met, if and when applicable.

The next figure shows an overview of the structure.



Configuration at 3-State Output Pin

The internal connections between the Boundary-Scan Cells should be as follows: the control lines of both cells (*Shift-DR*, *Clock-DR*, *Update-DR* and *Mode*) are connected in parallel; the TDO signal of the *Output cell* is connected to the TDI of the *Control cell*. In this way the TDI/TDO serial shift-register path is maintained.
Other internal connections between the two cells are also possible.

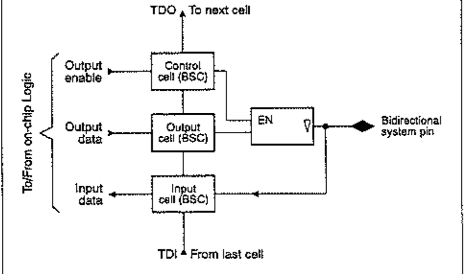
Cells at Bidirectional Pin

Bidirectional pins may be either 2-state or 3-state pins, used for input and output signals.
Requirements placed on the cells connected to these pins are the following.

- Cells are provided to select between input and output of signals, and which observe or control the data value at the pin.
- For 3-state bidirectional pins the input/output control cell must meet the requirements for the cell that drives the state of 3-state output pins.
- The input Boundary-Scan Cell must meet the same requirements as the input cell (see above).
- In output operation, the output cells must meet the requirements of the equivalent 2-State Output Pins, or where applicable the 3-State Output Pins.

In summary, the requirements are the same as those for both the 2-state or 3-state output pins and the system input pins.

The next figure shows an overview of the structure.



Cell Configuration At a Bidirectional Pin

The internal connections between the Boundary-Scan Cells for the bidirectional output pin are the same as in the 3-state output pin, except that the parallel output flip-flop of the control cell may have an extra input for the *Reset** signal.

Cells at the Clock Input Pin

- Each cell must contain a shift register.
- As a response to the *SAMPLE/PRELOAD* or the *EXTEST* instructions, the shift register stage loads the data presented at the pin on the rising edge (positive slope) of the TCK pulse when the TAP controller is in its *Capture-DR* state.
- Data presented at the input pin is supplied to the on-chip logic without modification by the cell, for example when the Boundary-Scan Register is not selected or after a *SAMPLE/PRELOAD* instruction.
- When the *RUNBIST* or *INTEST* instruction is valid, the clock signal applied to the on-chip logic must be one of the following:
  1. The signal received at a system clock input pin
  2. The TCK signal, such that the on-chip logic can only change when the TAP Controller is in its *Run-Test Idle* state
  3. For the *INTEST* instruction only: a signal that is supplied by shifting the Boundary-Scan Register.

The Device ID Register

The Device Identification Register is an optional register that may be included in an IC. However, *if* it is included, it must follow the requirements described in the IEEE 1149.1 standard.

The Device ID Register may provide (through the TAP) binary information about the manufacturer's name, part number and version number of the component (IC). This is important, for example for the following applications.

- In the factory, it allows verification that the correct IC has been mounted on the proper place on the PCB.
- When a component (IC) has been replaced, the version number of the replacement can be checked, and if required the test program can be modified.

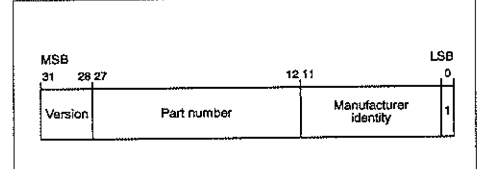
- When a PCB is added to a configuration at system level, the system test program can be adjusted to the new PCB and the ICs mounted on it.
- The correct programming of off-line programmed ICs can be checked.

The register must consist of a 32 bits shift-register, parallel-in and serial-out.
The register does not need a parallel output latch. Like the bypass register, the normal operation of the IC can continue while the ID register is in use.

Other requirements for the Device ID Register follow.

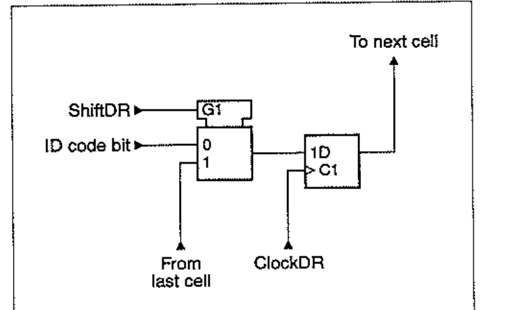
- The Device ID Register is set at the rising edge (positive slope) of the TCK pulse when the TAP Controller is in its *Capture-DR* state. The same code is shifted out towards the TDO at a subsequent shifting process.
- The vendor-defined identification must contain four fields, which can be read using the *IDCODE* instruction.
- For user-programmable components, a supplementary user-programmable identification code must be provided that can be loaded into the device identification register using the *USERCODE* instruction.

The structure of the data loaded into the Device ID Register in response to the *IDCODE* is shown in the next figure.



Structure of the Device Identification Register

The least significant bit (LSB) in the first field is always loaded as a logical 1. This is done in conjunction with the Bypass Register, which is always set to a logical 0.
If the optional Device ID Register is not present, the Bypass Register is chosen when an *IDCODE* instruction is executed. If subsequently the first bit shifted out of the component during a test data scan is a 0 (zero), then it can be deduced that the component has no Device ID register.
The other fields are applied for purposes as indicated in the figure.
The next figure shows an example configuration of a Device ID Register cell.



An Example of a Device ID Design

Design-Specific Registers

The manufacturer may add test data registers dedicated to his own design. For this purpose, design-specific instructions are developed. In the figure at the beginning of this chapter, these options are shown in dotted lines.

One point should be made here. The manufacturer may decide whether he makes the instructions for the design-specific register available in his component catalogue. In the latter case, he needs the design-specific registers only for his own in-house testing.

Instructions

General

The instructions are serially loaded into the Instruction Register from the TDI when the TAP Controller is in its *Shift-IR* state. Subsequently, the instructions are decoded to achieve two basic functions.

1. Select the set of test data registers that may operate while the instruction is current.
   Non-selected test data registers must be controlled so that they do not hamper the normal on-board operation of the related ICs.
2. Define the serial test data register path that is used to shift data between TDI and TDO during data register scanning.

The specifications of the instructions described in this chapter cover globally:

- an instruction, whether it is mandatory or optional;
- test data registers which are to be connected in the serial TDI/TDO path;
- binary code patterns (if applicable);
- the data flow through Boundary-Scan Register cells and/or on-chip logic signals.

Public and Private Instructions

Public instructions are documented and supplied by the manufacturer along with the component to the purchaser. These instructions provide the component purchaser with access to test features that help in test tasks, such as component test, board interconnect test, etc.

Supplementing the public instructions are the private instructions available solely for the use of the component manufacturer. The operation of private instructions need not be documented. If private instructions are utilized in a component, it remains the responsibility of the vendor to clearly identify any instruction codes that, if selected, will damage the component or cause hazardous operation of the component.

Overall Specifications of Public Instructions

The following specifications must be met in order to comply with the IEEE 1149.1 standard.

- All public instructions must be available to the purchasers.
- Instructions must *completely* define all test data registers that may operate and interact with the on-chip logic for as long as the instruction is current.
  Non-selected test data registers must be controlled so that they do not interfere with the operation of the on-chip logic, or with the test data registers which have been selected.
- Each instruction enables a single serial test data path along which data are shifted from TDI towards TDO while the TAP Controller is in its *Shift-DR* state.
- Unused instruction codes must be equivalent to the *BYPASS* instruction.
- If the optional Device ID Register is included in an IC, then the *IDCODE* instruction must be provided by the vendor.
  If the IC is user-programmable, the *USERCODE* must also be provided by the vendor.
- All components for which the vendor claims compliance with the IEEE 1149.1 standard must contain the following public instructions:
  BYPASS,
  SAMPLE/PRELOAD and
  EXTEST.

  The optional instructions,
  INTEST and/or
  RUNBIST,
  if available, must also comply with the IEEE 1149.1 standard.

The specifics of the general requirements of the public instructions are discussed in the following sections.

The BYPASS Instruction

- Each component (IC) must provide a *BYPASS* instruction code.
- The binary code for the *BYPASS* instruction is {111...1} (all 1s), thus a logic 1 is entered into every instruction register cell. However, additional binary codes are also permitted.
- The selected Bypass Register must be connected in the serial TDI/TDO path while the TAP Controller is in its *Shift-DR* state.
- When the *BYPASS* instruction is current, all test data registers must perform their system function.

The '*all 1s*' instructions can be entered by holding the TDI at the logical high level. Note that when the TDI is not terminated (caused by an open in the serial board-level test data path, for example), it behaves as if the input is high (logical 1). This causes the Bypass Register to be selected following an instruction-scan cycle. Thus, avoiding any unwanted interference with the normal IC operation.

If the optional Device ID Register is not present, the *BYPASS* instruction is forced into the latches at the parallel outputs of the Instruction Register when the TAP Controller is in its *Test-Logic-Reset* state. This ensures a complete serial path under all conditions, through either Bypass or Device ID Registers.

The SAMPLE/PRELOAD Instruction

Two functions can be performed by the use of *SAMPLE/PRELOAD* instruction. It allows a *SAMPLE* ('snapshot') of the normal operation of a component (IC) to be taken for examination. *Prior* to the selection of another test operation, a *PRELOAD* can place data values into the latched parallel outputs of the Boundary-Scan cells.

The following specifications must be met for the IEEE 1149.1 standard.

- The component must provide the *SAMPLE/PRELOAD* instruction.
- The *SAMPLE/PRELOAD* instruction addresses only the Boundary-Scan Registers for access via the serial TDI/TDO path in its *Shift-DR* TAP Controller state.

- During the execution of the *SAMPLE/PRELOAD* instruction the on-chip-logic operation is not hampered in any way:
  The signal states at the component input and output pins are loaded in the Boundary-Scan cells on the rising edge (positive slope) of the TCK pulse when the TAP Controller is in its *Capture-DR* state (*SAMPLE* action).
- When the *SAMPLE/PRELOAD* instruction is selected, the data present in the Boundary-Scan shift-register stage is loaded into the parallel output/latch at the falling edge (negative slope) of the TCK pulse when the TAP Controller is in its *Update-DR* state (*PRELOAD* action).

The figure on the next page shows the input and output cells when the *SAMPLE/PRELOAD* instructions has been selected.

The data flows in the Boundary-Scan cells are shown in the figure by bold lines.

Data flows (bold lines) in the PRELOAD phase of the SAMPLE/PRELOAD instruction

Data flows (bold lines) in the SAMPLE phase of the SAMPLE/PRELOAD instruction

Data Flows While the SAMPLE and PRELOAD Instructions Are Selected

The EXTEST Instruction

The *EXTEST* instruction is specifically provided to allow board-level interconnect testing of *opens*, *stuck-at* or *bridging errors* etc. This instruction also allows testing of cluster of components that do not themselves incorporate boundary-scan test but are surrounded by boundary-scan components.

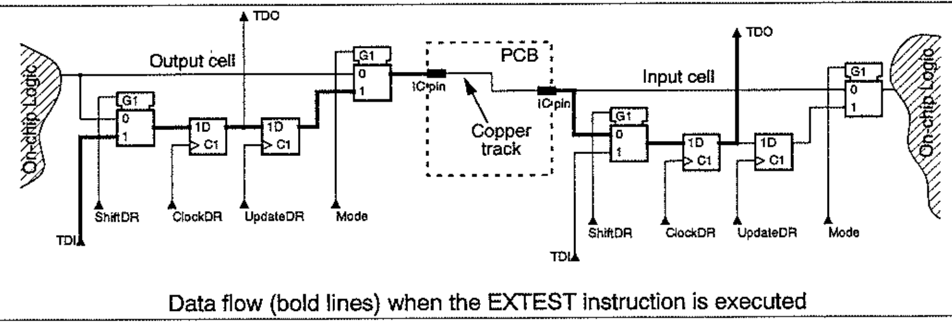
Since the board-level interconnect testing is one of the primary reasons for introducing Boundary-Scan Test, the *EXTEST* is therefore a mandatory instruction.
When the *EXTEST* instruction is selected, the on-chip logic is 'isolated' from the component's input and output pins. Data for the execution of the *EXTEST* instruction could typically be loaded beforehand into the Boundary-Scan Register stages with the *SAMPLE/PRELOAD* instruction.

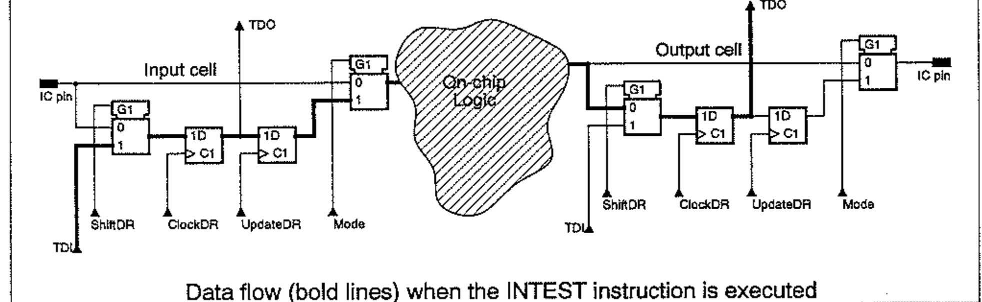
The following specifications must be met for the IEEE 1149.1 standard:

- Each component must provide the *EXTEST* instruction.
- The binary code for the *EXTEST* instruction is {000...0} (all 0s), thus a logic 0 is entered into every instruction register cell. However, additional binary codes are also permitted.
- The *EXTEST* instruction selects the *one* Boundary-Scan Register for access via the serial TDI/TDO path in the *Shift-DR* TAP Controller state.
- While the *EXTEST* instruction is executed, the on-chip logic must be protected against any damage caused by the signals received at the component's system input or clock pins.
- Signals arriving at the component's input pins are loaded in the Boundary-Scan Register at the rising edge (positive slope) of the TCK pulse while the TAP Controller is in its *Capture-DR* state.
- Test signals which are sent from the Boundary-Scan Registers through the component's output pins can only change on the falling edge (negative slope) of the TCK pulse while the TAP Controller is in its *Update-DR* state.
  These test signals must be completely defined by the test data previously shifted into the Boundary-Scan Register.

Furthermore, it is recommended that during the time of the execution of the *EXTEST* instruction, the Boundary-Scan Cells at the output pins can be backdriven (overdriven) with a pre-defined signal value without damaging the component.

The figure on the next page shows the *EXTEST* execution with the data flow in bold lines.





Data Flows When the EXTEST and INTEST Instructions Are Selected

The INTEST Instruction

This is an optional instruction which, if provided, also must comply with the IEEE 1149.1 standard. The INTEST instruction allows static (slow-speed) testing of on-chip logic after the component is mounted on the PCB.
Following this instruction, test signals are shifted in one at a time and applied to the on-chip logic. Signals for the execution of the *INTEST* instruction could typically be loaded beforehand into the Boundary-Scan Register stages with the *SAMPLE/PRELOAD* instruction.

The test results are captured into the Boundary-Scan Register and subsequently shifted out for examination.

The following specifications must be met for the IEEE 1149.1 standard.

- The *INTEST* instruction selects the *one* Boundary-Scan Register for access via the serial TDI/TDO path in the *Shift-DR* TAP Controller state.
  The associated component's on-chip logic must be able to perform single step operation in this case.
- Test signals which are sent from the Boundary-Scan Registers through the output pins can only change on the falling edge (negative slope) of the TCK pulse while the TAP Controller is in its *Update-DR* state.
  These test signals must be completely defined by the test data previously shifted into the Boundary-Scan Register.
- When the *INTEST* instruction is executed, all non-clock signals driven into the on-chip logic (IC) must previously have been shifted into the Boundary-Scan Register stages.
- Signals from the on-chip logic are loaded into the Boundary-Scan Register at the rising edge (positive slope) of the TCK pulse while the TAP Controller is in its *Capture-DR* state.

As stated above, the test signals are applied one at a time.
This requires that the on-chip logic can operate in a single-step mode, in which the circuitry moves one step forward in its operation each time the Boundary-Scan Register shifts one step forward.

Between the input of the signal and the capture of the response, the on-chip logic may receive clock signals. These clock signals may be obtained in various ways while the *INTEST* instruction is executed. Some examples are given in the description of the IEEE 1149.1 standard.

The figure for the *INTEST* instruction shows the data paths through the Boundary-Scan Cells of the input and the test result signals.

When the *INTEST* instruction is selected, the Boundary-Scan test substitutes testing by means of the bed-of-nails fixture and pin probing of automatic test equipment (ATE) for component (IC) testing. This is one of the reasons that Boundary-Scan Test was introduced as a new approach for testing, as already stated in the Introduction. Signals and responses are moved into and out of the on-chip logic by shifting the Boundary-Scan Register.

The RUNBIST Instruction

The Run Built-In Self-Test is an optional instruction which, if provided, also must comply with the IEEE 1149.1 standard. The *RUNBIST* instruction causes the execution of a self-contained self-test of the component (IC) without the need to apply complex data patterns. With the *RUNBIST* instruction the correct operation of a component (IC) can be tested dynamically, without the need to load complex data patterns and without the need for single-step operation (as required for *INTEST* instruction).

The following specifications must be met for the IEEE 1149.1 standard.

- The TAP Controller must be in its *Run-Test/Idle* state when the *RUNBIST* instruction is selected.
- No provisions need to be made for seed values to be entered in other than Boundary-Scan test data registers.
- The duration of the execution of the *RUNBIST* must be a specified item (e.g. specified as a number of system clock pulses).

- The data of test results shifted out of the component (IC) in response to the *RUNBIST* instruction must not be altered by the presence of faults in off-chip system logic, board-level interconnections etc.
- The state of the parallel output registers at the IC output pins (2-state, 3-state, bidirectional) must not change while the *RUNBIST* is selected.
- The test data register into which the results of the self-test are to be loaded must be connected for access in the serial TDI/TDO path while the TAP Controller is in its *Shift-DR* state.
- The results of the self-test must be loaded into the test data register (connected in the serial TDI/TDO path) no later than the rising edge (positive slope) of the TCK pulse in the *Capture-DR* TAP Controller state.
- Following the specified minimum duration, the test result observed by loading and shifting the test data register selected by the *RUNBIST* must be constant regardless when the *Capture-DR* TAP Controller state is entered.
- Use of the *RUNBIST* must give the same result in all versions of the component (IC).

The timing constraints have been added to ensure that all built-in self-tests are completed in one test run.
If on a PCB components are mounted with different specifications of self-test duration times, then the rules of IEEE 1149.1 standard ensure that enough time (measured in clock pulses) must have passed to allow for the completion of built-in self-tests executed simultaneously in different components on the board.

The last specification requirement has been added to ensure that the test of an assembled PCB is independent of the *versions* of the components (ICs) mounted on it. This is important when working in a maintenance or repair environment, in which the versions of the components used on the PCB are not known.

The IDCODE Instruction

If a Device Identification Register is included in a component design, the IDCODE is forced into the Instruction Register's parallel output latches during the *Test-Logic-Reset* TAP Controller state. This means of access to the Device Identification Register permits blind interrogation of components assembled onto a PCB, making it possible to determine what components exist on a board.

If a Device Identification Register is included in a component the following specifications must be met for the IEEE 1149.1 standard.

- Manufacturers must provide an *IDCODE* instruction for components with a Device Identification Register.
- When the *IDCODE* instruction is selected, only the Device Identification Register is connected between TDI and TDO during the *Shift-DR* TAP Controller state.
- When the *IDCODE* instruction is selected, the vendor identification code is loaded in the Device Identification Register.
- When the *IDCODE* instruction is selected, the test logic does not effect operation of the on-chip system logic.

The USERCODE Instruction

The *USERCODE* instruction must be provided by the manufacturer if the Device Identification Register is included in a component design and the component is user-programmable. When selected, this instruction loads the user-programmable identification code into the Device Identification Register. The IEEE 1149.1 standard requirements for this instruction are similar to those of the *IDCODE* instruction.

Overview

PCB testing generally refers to checking for the **correct connections** between the **right components.** Using a bed-of-nails technique, the interconnects are tested from nail to nail. Each nail makes physical contact with the copper track or solder joint on the PCB.

As already stated in the first chapter, from a mechanical point of view, the conventional way of testing PCBs with a bed-of-nails is no longer feasible in all cases. The Boundary-Scan Test technology was therefore proposed and introduced for PCB testing. The next table provides an overview of different types of faults covered by Boundary-Scan Test technology.

Boundary-Scan Test Fault Coverage

Instruction	Tests Performed	PCB Faults Covered
IDCODE	*Device Identity*	*Wrong IC*
EXTEST	*PCB Interconnect Cluster Test*	*Opens, shorts, Cluster failure*
INTEST	*Device Test (Static)*	*Missing, wrong or dead ICs*
RUNBIST	*Device Test (Dynamic)*	*Missing, wrong or dead ICs*

The Boundary-Scan Test technology performs these tests without requiring extensive bed-of-nail fixtures. With BST, the real bed-of-nail pins become virtual pins inside the IC, i.e. the Boundary-Scan Cell. Checking interconnects is now achieved through an imposed flow of digital signals between the BST cells of different ICs. The BST infrastructure replaces the bed-of-nails structure. To a large extent, *mechanical access and connection problems are solved by* BST, which requires a Test Access Port of a minimum of only four wires.

With Boundary-Scan Cells that comply with the IEEE 1149.1 standard, it is possible to test the interconnects (net) independently of the on-chip (in-core) logic of the ICs. The Boundary-Scan Cells can control the state of the input and output pins of a component. This *solves the problem of backdriving* (overdriving) the output circuitry of the IC connected to the net as it is found in the in-circuit tests, avoiding a hazard to reliability .

With BST, the interconnect is tested from Boundary-Scan Cell to Boundary-Scan Cell. Compared with the in-circuit test nails placed on the IC pins, BST *also includes the inner path in the IC, i.e. the output/input buffer and the IC bonding wires.* The BST, therefore, can locate faults on a track or in a IC, such as an open pin on an SMT IC or a faulty output/input buffer in an IC, which an in-circuit tester cannot locate or doesn't look for during an interconnect test.

When BST is used, the PCB under test must be powered up before the test can start. It is possible that *during BST an existing short on the PCB may damage the component (IC) before the test procedure has even started.* With the conventional bed-of-nails technology any shorts can be detected before powering up the PCB.

In *conclusion*, it can be stated that BST is very well-suited for PCB testing. It solves most of the mechanical problems faced in the testing of loaded boards when using bed-of-nails fixtures. It lowers the cost of test equipment, automates the generation of PCB interconnect test and reduces the overall lead time and cost of test program generation. This reduction in the cost of test equipment and the effort to generate test programs means that BST *tools are applicable even in the design debug phase,* thus creating a very strong link between design and test. Test patterns from the design department are re-used in manufacturing and field service, saving a lot of test preparation time. The result is considerable savings in PCB production cost.

Testing of Mixed Technology PCBs

Boundary-Scan testing does not always eliminate the need for other forms of testing, such as functional performance test of the PCB. Moreover, very few PCBs will be built exclusively from boundary-scan components. ASICs and other complex devices implemented with boundary-scan will co-exist with non-scan components. These non-boundary-scan components will usually be simpler devices.

A **dual test strategy** can be pursued for such mixed technology PCBs. First, boundary-scan test can be used wherever possible for ASICs and other complex device, this simplifies testing of the more complex parts of the PCB. The remaining parts of the PCB can be tested using cluster test via the boundary-scan parts, or using traditional test methods, such as MDA, in-circuit test, and/or functional test. The over all test development cost will still be significantly less than required for a PCB without any boundary-scan part.

PCB Life Cycle and Fault Classes

Usually the faults are categorized according to the three phases in the PCB life cycle.

Design Phase Faults

In the design phase of a PCB, the engineer wants to ensure that on the prototype PCB any encountered fault is a design error and not a manufacturing process-induced fault, such as a bad soldering fault. A boundary-scan test tool can be easily programmed to pinpoint these manufacturing process-induced faults in the design prototype.

One point must be made here. Design errors are hard to classify, because the knowledge of the design is in the head of the designer. As a consequence, it is hardly possible to model these design errors so that they could be simulated and programmed beforehand. As a result, automatic testing can be very rarely applied in the R&D environment.

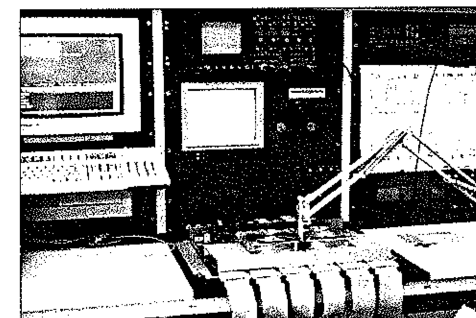
Manufacturing Phase Faults

In the factory, the emphasis in testing is on throughput. In the first place there is a

process control. Mounted PCBs are tested for the presence of manufacturing process faults like shorts, opens, wrongly assembled components (wrong value, misassembled) etc. The functionality of components is also tested as far as possible.

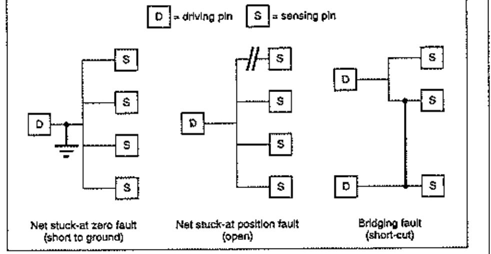
Manufacturing process faults can be modelled to a high degree of completion. The required test patterns (vectors) can be generated automatically. These could be available from the design phase of the PCBs and re-used in the factory.

The next figure summarizes the faults which are mostly detected in this environment.



Some frequent errors in manufacturing

If the PCB does not pass the process control test, a *diagnosis for repair* must be made. After process testing, the PCB is tested on functionality. *Functional tests* are performed by observing the PCB responses in its normal operating conditions. The responses can be measured at the PCB's edge connectors.



Production test site.
Boundary-Scan, In-circuit and Functional tests are performed

Since the emphasis in a factory is on volume production, throughput of the factory test equipment is very important. Short equipment set-up times and quick diagnosis are therefore of the utmost importance. For price reasons, lower costs for the (expensive) automatic test equipment are also vital.

Field Service Faults

Despite all precautions taken in the factory, a PCB may stop functioning in the field. This can be for many different reasons. But the most common reasons are improper environmental operating conditions (humidity, temperature) and/or physical mishandling resulting in interconnection or component failure, although sometimes spontaneous component faults also occur.

The test equipment for the service engineer should be able to make a quick diagnosis of the faults. Detection of interconnection failures is particularly useful, because a short or an open may be fixed at the customer's site and keep the spares inventory costs low. Otherwise, the PCB is exchanged and the faulty one is returned to a central repair shop.

A Boundary-Scan Test Feasibility Study

In the late 80's, the testing problems caused by miniaturization, increased circuit complexity, and ASICs was driving up the production test costs in Philips. In-circuit testers with bed-of-nail fixtures were becoming less effective and forcing Philips to rely more heavily on expensive functional test techniques to maintain high product quality.

As a result, in 1989, a thorough preliminary investigation was made into possible savings resulting from the introduction of Boundary-Scan Test (BST) for PCB production in a Philips Product Division. People involved were from design departments, factory engineering and Organization & Efficiency departments.

These factors were taken into account:

- programming times for various (60) types of PCBs;
- extra costs for implementing Boundary-Scan Cells;
- diagnostic times and related times to repair;
- tester costs for testing times and tester depreciation in three years,

Then, a comparison was made of the costs of two test techniques: Boundary-Scan Test and Functional Test. Adoption of BST meant a decrease and in some cases total elimination of Functional Test.

The comparative ratio of total production costs were estimated as

BST : functional test = 2 : 7

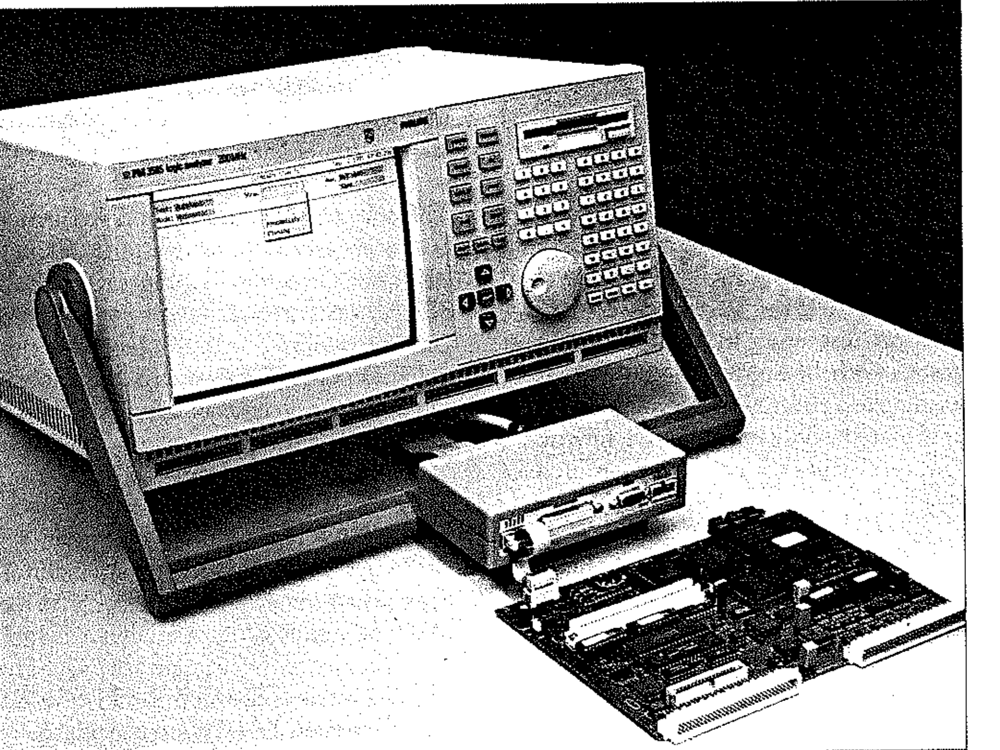
Adoption of BST indicated a reduction in total production costs of up to 70%! Management therefore had no choice but to decide to go ahead with BST.

Introduction to PF 8660/30

At Philips T&M, Boundary-Scan technology was first widely used in the design of the PM 3580 family of logic analyzers, which were introduced to the market in 1990. The first in a series of products, the PF 8660/30 Boundary-Scan Test Option is available for the PM 3580 family logic analyzers.

The PF 8660/30 Option is fully compatible with the IEEE 1149.1 standard. It consists of three elements:

1. the hardware adapter box
2. the test pattern (vector) generation software running on a PC
3. the diagnostics software running on the logic analyzer.



R&D test set-up applying the PF 8660/30 Boundary-Scan Test Option

In the next sub-sections, the characteristics and possibilities of the PF 8660/30 BST Option are explained.

Automatic Test Vector Generation for Infrastructure and Interconnect Test
The first step in the Boundary-Scan Test process is the generation of the test vectors. The starting point is the creation of a 'Net assignment file' on a PC. This file contains the netlist information for the circuit together with the description of the Boundary-Scan chain. The circuits may contain Drive, Sense, Tri-state and Bidirectional cells. The BST Pattern Generation software then automatically generates the Boundary-Scan Test vectors for Boundary-Scan infrastructure and full Boundary-Scan interconnect tests.

The Boundary-Scan infrastructure can be tested through the *Bypass* register, the *Identification register* or the complete Boundary-Scan chain. For the interconnect test the fault classes to be covered can be specified by the user.

Bridging faults using:
- binary search algorithm;
- an enhanced algorithm (better fault detection, especially for CMOS);
- the inverted patterns.
- *Stuck-at-zero*, and *stuck-at-one* faults including opens for nets and at-positions in nets.
- *Extended stuck-at* faults for bus structures.

Generation of IC or Cluster Vectors
If the internal logic of a boundary-scan IC or a cluster of non-scan circuitry, like glue logic, enclosed by boundary-scan ICs must be tested, the parallel test vectors generated elsewhere for these ICs or clusters can be easily imported. The test generation software will automatically convert the parallel test vectors into serial vectors and embed these vectors in the protocol for use by the PM 3580 Family logic analyzers.

Connecting the Target
The Boundary-Scan Test Adapter Box supports the IEEE 1149.1 standard by providing the connection to two different Test Access Ports, each containing *TDI*, *TDO*, *TCK* and *TMS*, and the optional signal *TRST**.
Three additional channels (two drive, one sense, all TTL compatible) are available to control, for example, the power supply of the Unit-Under-Test.

Test Execution and Diagnostics
The logic analyzer is connected to the circuit via the TAP port and the Boundary-Scan test is executed using all the above-mentioned test vectors. When the information from the Test Generation software has been loaded into the logic analyzer, the execution of the test can be initiated by simple menu commands. It is possible:
- to execute all test modules in a sequence automatically;
- to execute a single test module;
- to single steps through the test patterns in a test module;
  - to set up the control line options;
  - to run a test continuously.

Within seconds of a test execution, a pass or fail result is displayed on the screen of the logic analyzer. In the event of a fail, various levels of diagnostic information can be called-up which precisely describe the fault or faults to facilitate repair. The following diagnostic report levels can be selected from a menu:

- *Fault summary*: displays for each fault the pattern number, fault type, net name and detection probability.
- *Full fault specification*: additionally displays for each fault the full information including net connections, control signals and boundary scan bit positions.
- *Uninterpreted fault information*: displays bit information for all failing vectors.

The next figure gives an overview of the software structure.

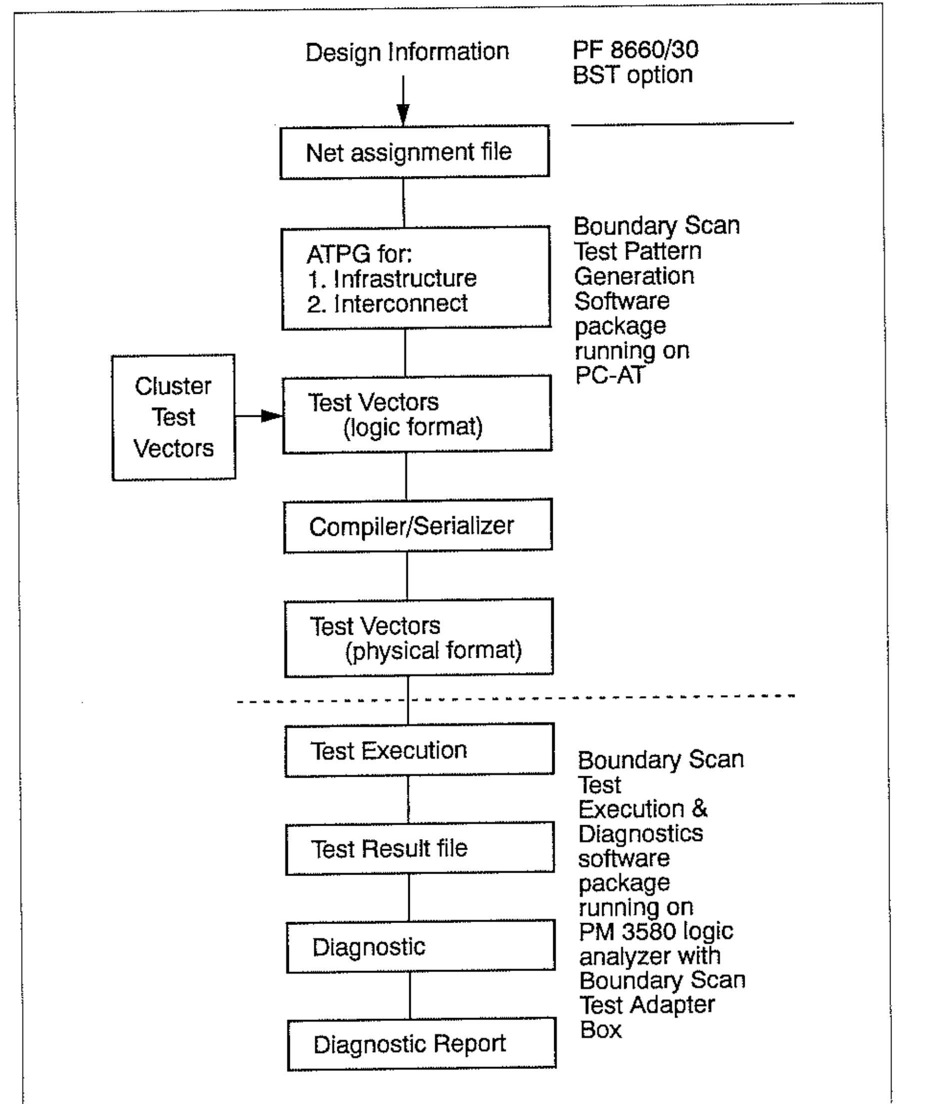


Diagram of the Test Software structure

Next are some of the screens from the logic analyzer shown with test results obtained with the PF 8660/30 Boundary-Scan Test Option.

```
                         BOUNDARY-SCAN TEST            24 May 1991 07:44
TEST: [AUTOTEST]      SHOW: ERROR VECTORS         SAVE
MODE: AUTOMATIC                                 POWER OPTIONS

                           TEST
                           AUTOTEST
                           BS_CLUST
                           [BS_IDENT]
                           BS_INTER
                           QUATTRO
```

Screen display showing the menu options for the TEST field

This screen shows the menu for the kind of tests that can be done if the TEST option is selected (by cursor movement) in the screen header.
When in the TEST menu, *Autotest* is chosen, all the following tests named in the menu are performed sequentially. *Quattro* is the name of the whole PCB.

The other options in the screen header possess the following options.

MODE: *Automatic* or *Single Step* testing.
SHOW: *Error Vectors*, *Probability* or *Pinning* errors.
SAVE: *Error Vectors* or *Diagnostics* on the disc of the logic analyzer.
POWER OPTIONS: *Manual Power On*, *Automatic + ACK* or *Automatic.*

```
                         BOUNDARY-SCAN TEST            24 May 1991 08:19
TEST: BS_IDENT        SHOW: [PROBABILITY]         SAVE
MODE: AUTOMATIC                                 POWER OPTIONS

Chain A IC 1        not ok - expected ident: 0
                                found ident: 1
Chain A IC 2        not ok - expected ident: 0
                                found ident: 1

Possible causes:
----------------
Incorrect IC 2 has been mounted
Error in test clock interconnection to IC 2 pin TCK
Error in serial test data interconnection from IC 2 pin TDO to PCB TAP

Chain B IC 3        not ok - expected ident: 0
                                found ident: 1

Possible causes:
----------------
Incorrect IC 3 has been mounted
Error in test clock interconnection to IC 3 pin TCK
Error in serial test data interconnection from IC 3 pin TDO to PCB TAP
```

Screen display showing Boundary-Scan Identification faults

```
                         BOUNDARY-SCAN TEST            24 May 1991 09:02
TEST: QUATTRO         SHOW: [PROBABILITY]         SAVE
MODE: AUTOMATIC                                 POWER OPTIONS

---------------------------------------------------------------------------
 Nr  Fault description                  Actual  Possible  Percent  Errors
                                        detects  detects  detects  covered
---------------------------------------------------------------------------
  1  Net ADA05 stuck-at 1                  60       60     100.00     60
  2  Net AAD02 stuck-at 1                  60       60     100.00     60
  3  Net QW0-5 stuck-at 0                   4        4     100.00      4
 34  Bridge between CSMEMAA2 and CSIDAA2    2        2     100.00      2
---------------------------------------------------------------------------
```

Screen display showing a summary of Stuck-at and Bridge faults

```
                         BOUNDARY-SCAN TEST            24 May 1991 08:46
TEST: QUATTRO         SHOW: [PINNING]             SAVE
MODE: AUTOMATIC                                 POWER OPTIONS

Net stuck-at faults: none

Pos stuck-at faults: none

Net bridging faults:
---------------------------------------------------------------------------
 Nr  Net       Dir IC  Pin En  Control   Actual sense        Expect sense
                               ch  pos   0-0 1-0 0-1 1-1    0-0 1-0 0-1 1-1
---------------------------------------------------------------------------
 1   CSMEMAA2  D   51  26  -   -   -     0   0   0   0      0   0   0   0
               S   12  2   -   -   -     7   0   1   4      6   1   1   4
     CSIOAA2   D   51  27  -   -   -     0   0   0   0      0   0   0   0
               S   12  3   -   -   -     7   0   1   4      6   1   1   4
---------------------------------------------------------------------------
```

Screen display showing Net Bridging faults

```
                    BOUNDARY-SCAN TEST              24 May 1991 08:53
TEST: QUATTRO          SHOW: [ERROR VECTORS]     SAVE .
MODE: AUTOMATIC                                  POWER OPTIONS
QUATTRO  : test failed, 4 errors
    PAT     CHAIN   POS
     2        A     647
     4        A     647
     9        A     647
    12        A     647
```

Screen display showing a report of the Error Vectors

Index

3-state 20
3-state bidirectional pins 22
All 0s 29
Automatic testing 35
Backdriving 29, 34
Bed-of-nails technique 34
Board connect test 19
Board Density 2
Boundary-Scan Cell 8, 20
Boundary-Scan infrastructure 38
Boundary-Scan Register 8, 17, 19
Boundary-Scan Test 47
BSC 8, 20
BSR 8
BST 47
Built-in self test 3
BYPASS instruction 27
Bypass Register 17, 18
Capture-DR 13, 14
Capture-IR 14
Cells at 2-State Output Pin 21
Cells at 3-State Output Pin 22
Cells at Bidirectional Pin 22
Cells at Input Pin 21
Cells at the Clock Input Pin 23
Complex ICs 1
Component pin 21
Component-specific clock 9
Control cell 22
Design Phase Faults 35
Design-for-test 3
Design-Specific Registers 25
Device Identification Register 23
Exit1-DR 15
Exit1-IR 15
Exit2-DR 15
Exit2-IR 15
Factory testing 35
Fault Classes 35
Field Service Faults 36
Functional test 36
Functional test 35
IDCODE 6, 16, 24
In-circuit testing 2
In-core logic 20
Inactive driver state 10
Instruction Register 8, 15
INTEST Instruction 30
IR 8
Latched parallel output 14
Manufacturing Phase Faults 35
Miniaturization 1
Net assignment file 38
On-chip logic 20
Optional reset signal 9
Output cell 22
Overdriving 34
Pause-DR 15
Pause-IR 15
PCB Life Cycle 35
PRELOAD 21, 27
Private Instructions 26
Public Instructions 26
R&D testing 35
Run-Test/Idle 13, 14
RUNBIST instruction 14, 21, 31
SAMPLE 27
SAMPLE/PRELOAD Instruction 27
Seed values 31
Select-DR-Scan 13, 15
Select-IR-Scan 13, 15
Service Testing 36
Shift-DR 14
Shift-IR 14
State Diagram 12, 13
State transitions 13
System clock 9
TAP 9
TAP Controller 8
TCK 9
TDI 10
TDO 10
TDR 8
Test Access Port 9
Test architecture 8
Test Clock Input 9
Test Data Input 10
Test Data Output 10
Test Data Registers 8
Test Mode Select Input 9
Test Reset Input 10
Test-Logic-Reset 13
The EXTEST Instruction 29
TMS 9
Tri-state circuitry 11
TRST 10
Update-DR 14
Update-IR 15
USERCODE 24

Main International Sales Offices

Australia: Philips Scientific & Industrial Pty. Ltd., 25-27 Paul Street, North Ryde, N.S.W. 2113; tel. (02)88 88 222

Austria: Philips Professionelle Elektronik GmbH, Markbereich Test- undMessgeräte, Gutheil SchoderGasse 10,A 1102 Wien; tel. (0222)60101/ 1772DW

Belgium: N.V. Philips Professional Systems S.A., Test &Measurement, Tweestationstraat 80, Rue des Deux Gares, Brussel 1070 Bruxelles; tel.(02)5256692/94

Denmark: Philips A/S, Test and Measurement, P.O. Box 1919, Prages Boulevard 80, DK-2300 Kobenhavn S. tel.(01)572222

Finland: OY Philips ab IE/T&M, P.O. Box 75/SF-02631 Espoo Tel. (0)5026371

FRANCE: S.A. Philips Industrielle et Commerciale, Science et Industrie, 105 Rue Paris, BP 62, 93002 Bobigny; tel. (1)49428028

Germany: Philips GmbH, Unternehmensbereich Elektronik für Wissenschaft und Industrie, Vertriebsbereich Test und Messtechnik, Postfach 310320, D-3500 Kassel; tel. (561)5010

Great-Britain: Philips Industrial Electronics, Test and Measurement Division, Colonial Way, Watford Herts WD2 4TT, tel. (0923)240511

Ireland: Circuit Specialists Ltd. Unit 5, Enterprise Centre, Plassey Technology Park, Castleroy, Limerick, Eire, tel. (061)330333

Italy: Philips S.pA., Divisione Industrial Electronics, Reparto T&M, Viale Elvezia 2, 20052 Monza; tel. (039)3635240/8/9

NETHERLANDS: Philips Nederland, Afd. Test- en Meetapparaten, Bld. HB4, Beatrixkade IF, 5652 BE Eindhoven, tel. 040-724444

New Zealand: Philips Scientific and Industrial Pty. Ltd., Test and Measurement Dept. Private Bag, St. Lukes, Auckland 3,2 Wagener Place; tel. (9)894160

Norway: NorskAktieselskap Philips, IE Dept, Test & Measurement, P.O. Box 1, Manglerud, Oslo 6; tel. (2)741010

Portugal: Philips Portuguesa S.A., Av. Eng. Duarte Pacheco. 6, 1009 Lisboa Codex, tel; (01)657181

South Africa: South Africa Philips (PtY) Ltd., Dept. I+E, P.O. Box 7703, Johannesburg 2000; tel. (11)889 3911

Spain: Philips Ibérica S.A.E. Dpto Aparatos de Medida, Martinez Villergas 2 Madrid 28027; tel. (1)4042200

Sweeden: Philips Industrial Electronics AB, tegeluddsvägen 1, s-11584 Stockhlm, tel. (08)782 1300

Switzerland: Philips AG, Test und Meßtechnik, Riedskrasse 12, Postfach 360, CH-8953 Wietikon 1, tel. (1)745 22 44

USA & Canada: John Fluke Mfg. Co., Inc. P.O. Box C9090, Everett, WA 98206; tel. (206)347-6100

For countries not listed above:
Philips Export B.V., Industrial Electronmics Division, Bld. TQ III-3, P.O. Box 218, 5600 MD Eindhoven, The Netherlands; tel. (3140)788667

he figures and descriptive materials contained in this "ABCs" e for illustrative purposes only.

he contents are subject to change without notice.

he reader is free to use (parts of) the contents of this ›cument for their own purposes, as long as reference is made this publication of the Fluke and Philips Test and Measure- ent Alliance.

›r more information on Boundary-Scan Test or more copies this booklet, please use the enclosed reply card.

HOGESCHOOL VOOR WETENSCHAP & KUNST DE NAYER INSTITUUT SINT-KATELIJNE-WAVER

Digitale Synthese FPGA: XC Virtex-II

XILINX®

EmSD Embedded System Design

ASSOCIATIE K.U.LEUVEN

ir. J. Meel

may 2006

1. XILINX Virtex-II Familiy of FPGAs

VIRTEX-II Features

- 0.15μ 8-layer copper CMOS Process
- Power Supply: 1.5 V Core, 3.3V I/O
- capacity to 10M systems gates
- ***Flexible Logic Resources***
  - CLB: 8 LUTs, 8 FFs
  - dedicated SOP-logic (Sum of Products)
  - wide multiplexer logic
- ***Arithmetic Functions***
  - fast look-ahead carry logic chains
  - distributed multiplier
  - dedicated 18x18 Multiplier blocks
- ***Internal Memory***
  - distributed RAM (LUT)
  - 18Kbit BlockRAMs

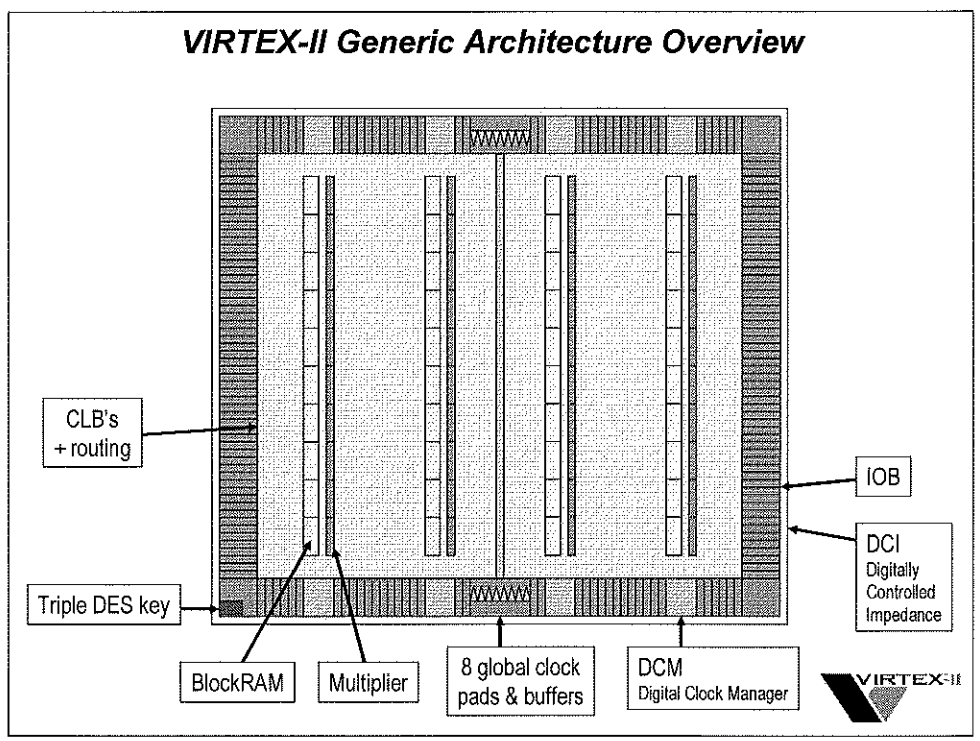
The Virtex-II family is a platform FPGA developed for high performance from low-density to high-density designs that are based on IP cores and customized modules. The family delivers complete solutions for telecommunication, wireless, networking, video, and DSP applications, including PCI, LVDS, and DDR interfaces.

The leading-edge 0.15 μm / 0.12 μm CMOS 8-layer metal process and the Virtex-II architecture are optimized for high speed with low power consumption. Combining a wide variety of flexible features and a large range of densities up to 10 million system gates, the Virtex-II family enhances programmable logic design capabilities and is a powerful alternative to mask-programmed gates arrays.

The Virtex-II family comprises 11 members, ranging from 40K to 8M system gates.

VIRTEX-II Features

- ***High-Performance Interfaces to External Memory***
  - DRAM interfaces (SDR / DDR SDRAM)
  - SRAM interfaces (SDR / DDR / QDR SRAM)
  - CAM interfaces
- ***Digital Clock Managers (DCM)***
  - 200MHz+ System Clock
  - Precise clock de-skew
  - Flexible frequency synthesis
  - High-resolution phase shifting
- ***Select I/O Technology***
  - 840 Mb/s Low-Voltage Differential Signaling I/O (LVDS)
  - Digitally Controlled Impedance (DCI)
  - PCI-X compatible (133 MHz and 66 MHz) at 3.3V
  - 19 single-ended and 6 differential standards
- ***Triple DES encrypted bitstream***
- ***Flip-Chip and Wire-Bond BGA Packages***



Virtex-II devices are user-programmable gate arrays with various configurable elements. The Virtex-II architecture is optimized for high-density and high-performance logic designs.

The programmable device is comprised of input/output blocks (IOBs) and internal configurable logic blocks (CLBs).

Programmable I/O blocks provide the interface between package pins and the internal configurable logic. Most popular and leading-edge I/O standards are supported by the programmable IOBs.

The internal configurable logic includes four major elements organized in a regular array.

• Configurable Logic Blocks (CLBs) provide functional elements for combinatorial and synchronous logic, including basic storage elements. BUFTs (3-state buffers) associated with each CLB element drive dedicated segmentable horizontal routing resources.

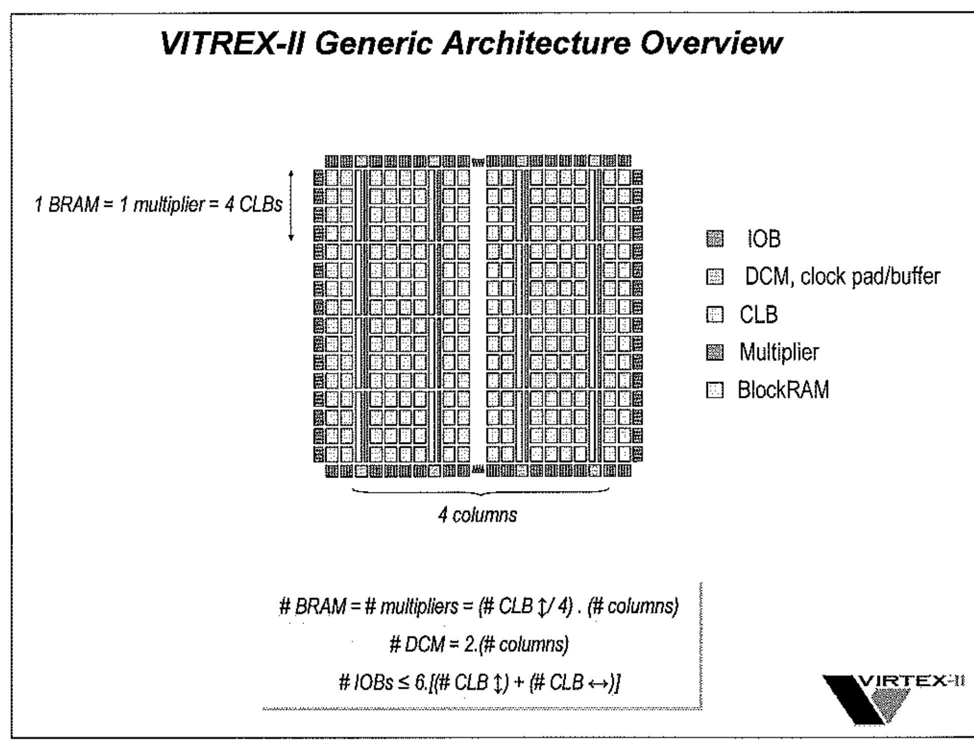
• Block SelectRAM memory modules provide large 18 Kbit storage elements of dual-port RAM.

• Multiplier blocks are 18-bit x 18-bit dedicated multipliers.

•DCM (Digital Clock Manager) blocks provide self-calibrating, fully digital solutions for clock distribution delay compensation, clock multiplication and division, coarse- and fine-grained clock phase shifting.

A new generation of programmable routing resources called Active Interconnect Technology interconnects all of these elements. The general routing matrix (GRM) is an array of routing switches. Each programmable element is tied to a switch matrix, allowing multiple connections to the general routing matrix. The overall programmable interconnection is hierarchical and designed to support high-speed designs.

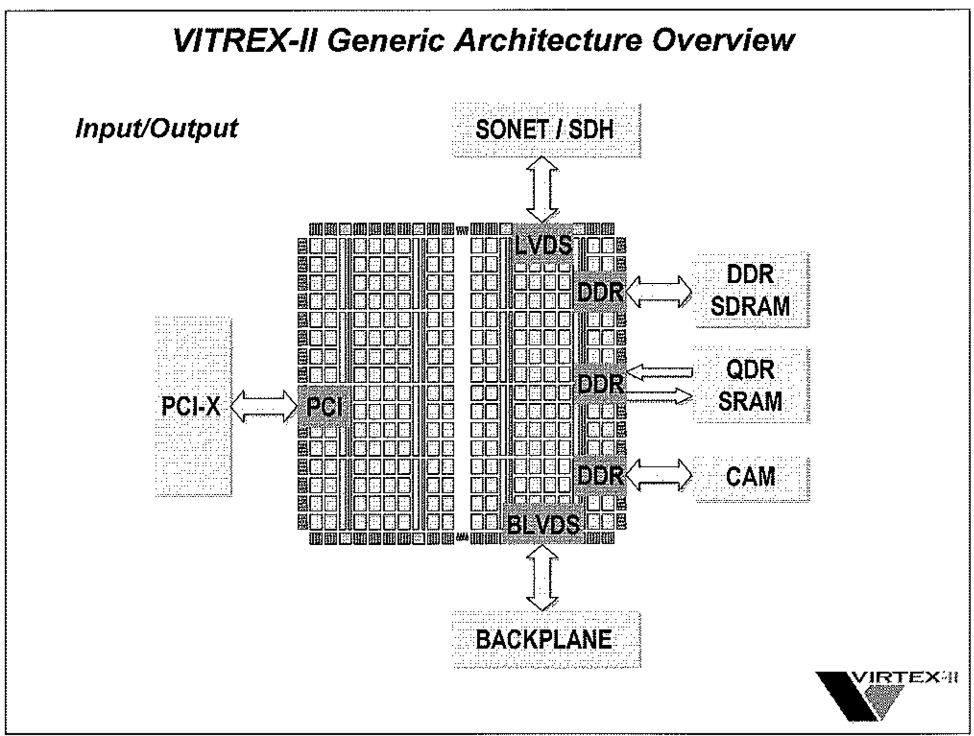
All programmable elements, including the routing resources, are controlled by values stored in static memory cells. These values are loaded in the memory cells during configuration and can be reloaded to change the functions of the programmable elements.



Virtex-II 18Kbit Block SelectRAM memory blocks are located in either two, four or six columns. The number of blocks per column depends of the device array size and is equivalent to the number of CLBs in a column divided by four.

Multiplier organization is identical to the 18 Kbit Block SelectRAM organization, because each multiplier is associated with an 18 Kbit block SelectRAM resource.

Virtex-II DCMs are placed on the top and bottom of each block RAM and multiplier column.



Input/Output Blocks (IOBs)

IOBs are programmable and can be categorized as follows:

• Input block with an optional single-data-rate or double-data-rate (DDR) register

• Output block with an optional single-data-rate or DDR register, and an optional 3-state buffer, to be driven directly or through a single or DDR register

• Bidirectional block (any combination of input and output configurations)

These registers are either edge-triggered D-type flip-flops or level-sensitive latches.

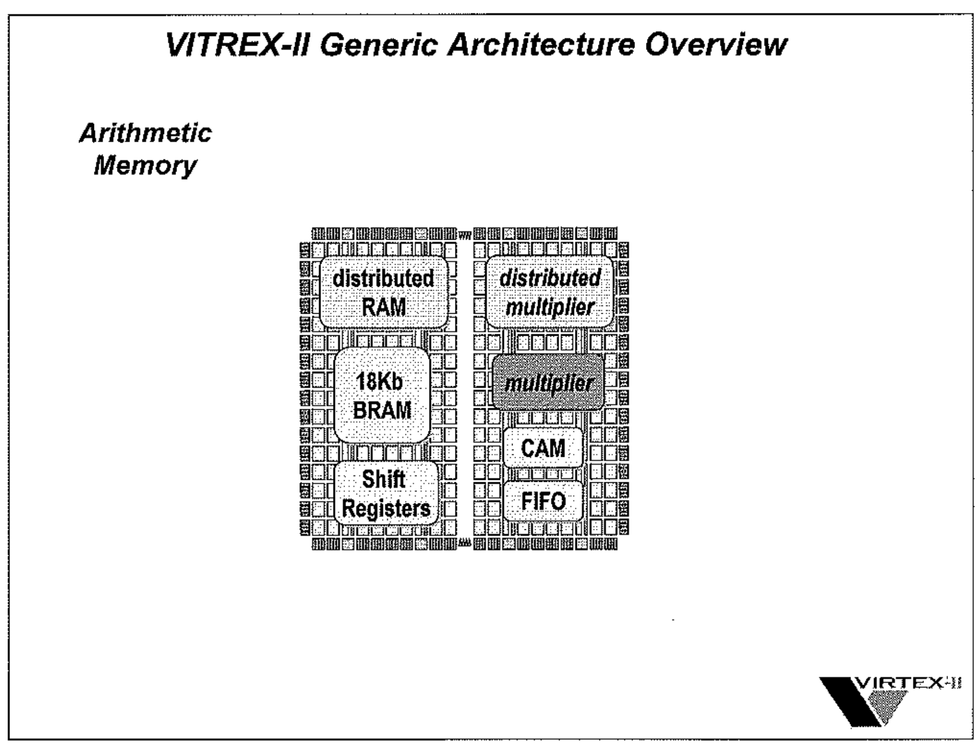
IOBs support the single-ended I/O standards (LVTTL, LVCMOS (3.3V, 2.5V, 1.8V, and 1.5V), PCI-X compatible (133 MHz and 66 MHz) at 3.3V, PCI compliant (66 MHz and 33 MHz) at 3.3V, HSTL (Class I, II, III, and IV), SSTL (3.3V and 2.5V, Class I and II) ...)

The digitally controlled impedance (DCI) I/O feature automatically provides on-chip termination for each I/O element.

The IOB elements also support differential signaling I/O standards (LVDS, BLVDS (Bus LVDS), ULVDS, LDT, LVPECL). Two adjacent pads are used for each differential pair. Two or four IOB blocks connect to one switch matrix to access the routing resources.

High-Performance Interfaces to External Memory:

- DRAM interfaces:
  · SDR / DDR SDRAM
  · Reduced Latency DRAM
- SRAM interfaces
  · SDR / DDR SRAM
  · QDR™ SRAM
- CAM interfaces

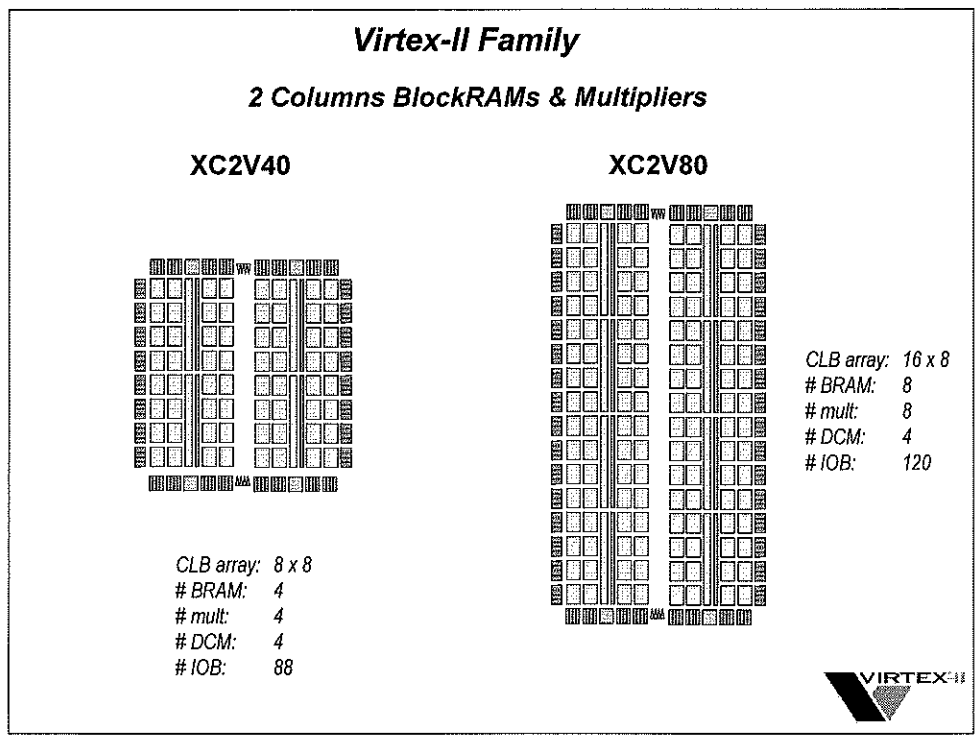


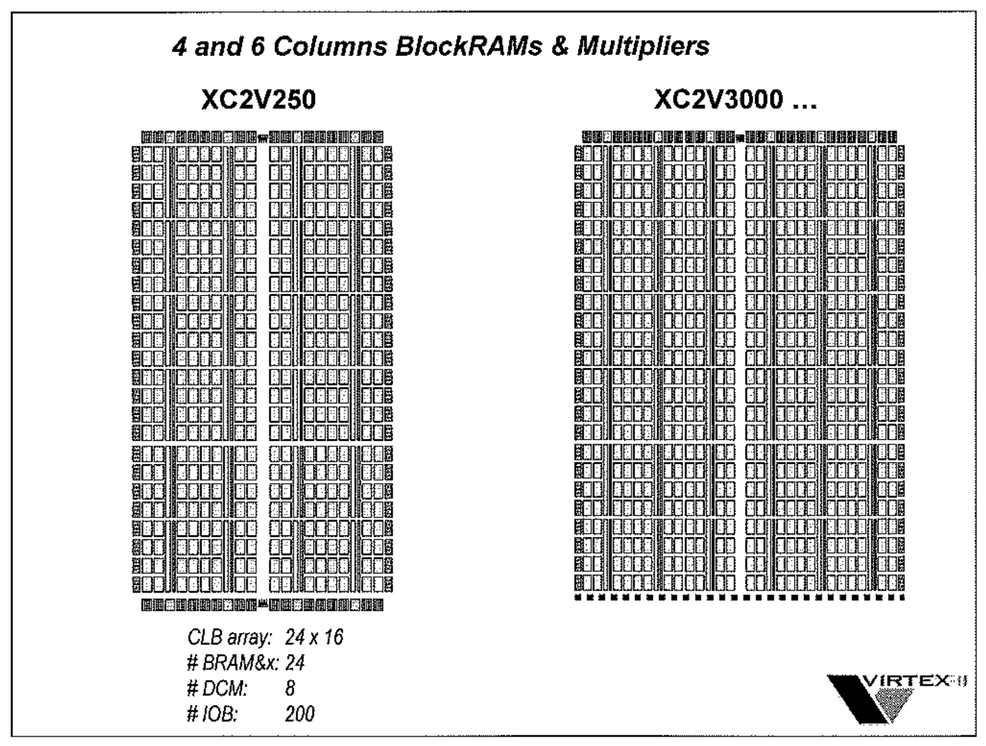
SelectRAM™ Memory Hierarchy

- Up to 1.5 Mb of *distributed* SelectRAM resources (can be organised as shiftregisters)
- Up to 93,184 cascadable 16-bit shift registers (distributed)
- 3 Mb of dual-port RAM in 18 Kbit *Block* SelectRAM resources

Arithmetic Functions

- Fast look-ahead carry logic chains
- The CLB elements have dedicated logic to implement efficient multipliers in logic
- Dedicated 18-bit x 18-bit multiplier blocks


Virtex-II Family
2 Columns BlockRAMs & Multipliers
XC2V40
XC2V80
CLB array: 16 x 8
BRAM: 8
mult: 8
DCM: 4
IOB: 120
CLB array: 8 x 8
BRAM: 4
mult: 4
DCM: 4
IOB: 88
VIRTEX-II


4 and 6 Columns BlockRAMs & Multipliers
XC2V250
XC2V3000 ...
CLB array: 24 x 16
BRAM&x: 24
DCM: 8
IOB: 200
VIRTEX

Virtex-II Family Members

Device XC2V	**40**	**80**	**250**	**500**	**1000**	**1500**	**2000**	**3000**	**4000**	**6000**	**8000**
system gates	40K	80K	250K	500K	1M	1.5M	2M	3M	4M	6M	8M
CLB array	**8 x 8**	**16 x 8**	**24 x 16**	**32 x 24**	**40 x 32**	**48 x 40**	**56 x 48**	**64 x 56**	**80 x 72**	**96 x 88**	**112 x 104**
# slices	256	512	1536	3072	5120	4680	10752	14336	23040	33792	46592
# CLB FFs	512	1024	3072	6144	10240	15360	21504	28672	46080	67584	93184
CLB RAM (kbits)	8	16	48	96	160	240	336	448	720	1056	1456
# 18Kbits BRAM	***4***	***8***	***24***	***32***	***40***	***48***	***56***	***96***	***120***	***144***	***168***
# multiplier	4	8	24	32	40	48	56	96	120	144	168
max IOB (single)	**88**	**120**	**200**	**264**	**432**	**528**	**624**	**720**	**912**	**1.104**	**1.296**
DCM	4	4	8	8	8	8	8	12	12	12	12
configuration memory (bit)	0.4M	0.6M	1.7M	2.8M	4.1M	5.7M	7.5M	10.5M	15.7M	21.9M	29.1M

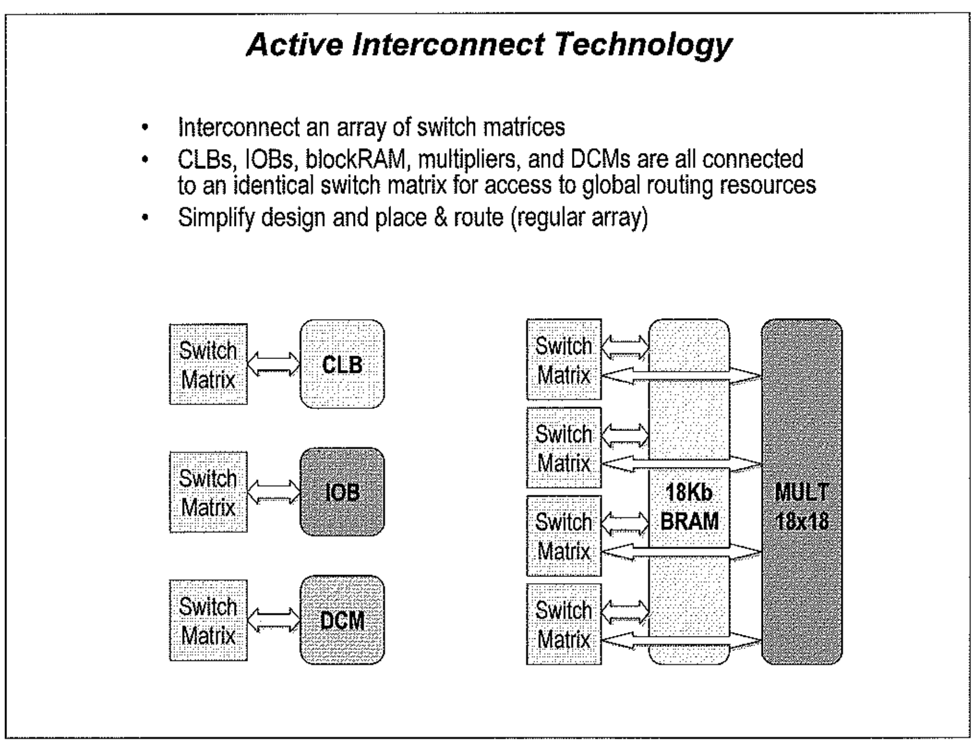
BRAM & Multipliers — *2 Columns* (40, 80) — *4 Columns* (250 – 2000) — *6 Columns* (3000 – 8000)

1 CLB = 4 slices = 8 FFs = 128 RAM bits

 # BRAM = # multipliers = (# CLB ↕/ 4) . (# columns)

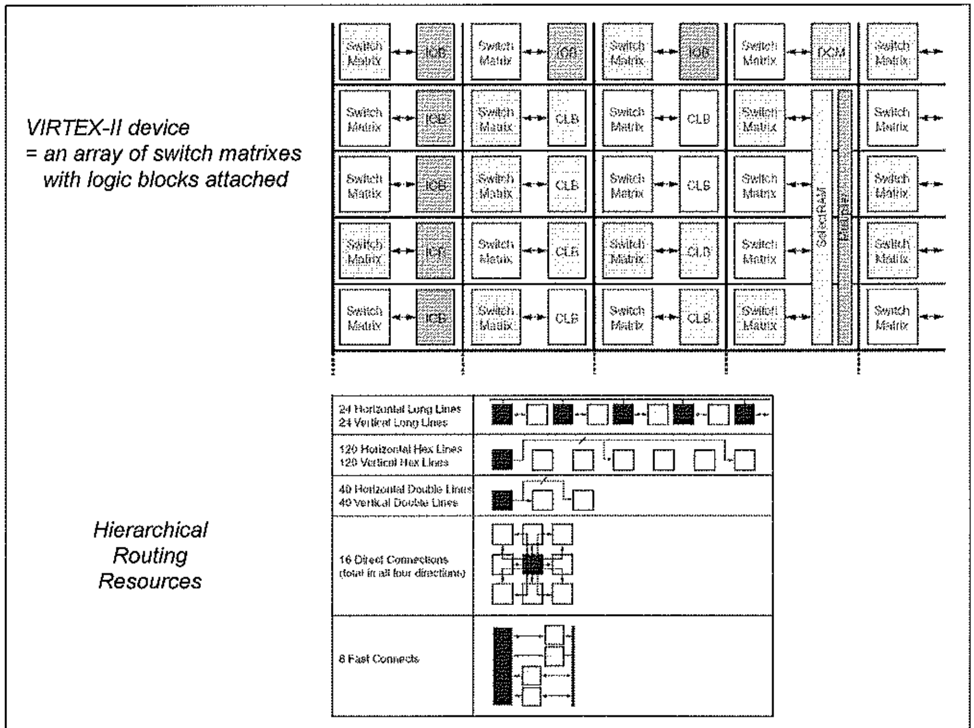
 # DCM = 2.(# columns)

 # IOBs ≤ 6.[(# CLB ↕) + (# CLB ↔)]



Local and global Virtex-II routing resources are optimized for speed and timing predictability, as well as to facilitate IP cores implementation. Virtex-II Active Interconnect Technology is a fully buffered programmable routing matrix. All routing resources are segmented to offer the advantages of a hierarchical solution.

The IOB, CLB, Block SelectRAM, multiplier, and DCM elements all use the same interconnect scheme (connection to an identical switch matrix) for access to the global routing matrix. Timing models are shared, greatly improving the predictability of the performance of high-speed designs.



Each Virtex-II device can be represented as an array of switch matrixes with logic blocks attached.

Place-and-route software takes advantage of this regular array to deliver optimum system performance and fast compile times. The segmented routing resources are essential to guarantee IP cores portability and to efficiently handle an incremental design flow that is based on modular implementations. Total design time is reduced due to fewer and shorter design iterations.

Hierarchical Routing Resources

Most Virtex-II signals are routed using the global routing resources, which are located in horizontal and vertical routing channels between each switch matrix.

Virtex-II has fully buffered programmable interconnections, with a number of resources counted between any two adjacent switch matrix rows or columns. Fanout has minimal impact on the performance of a net.

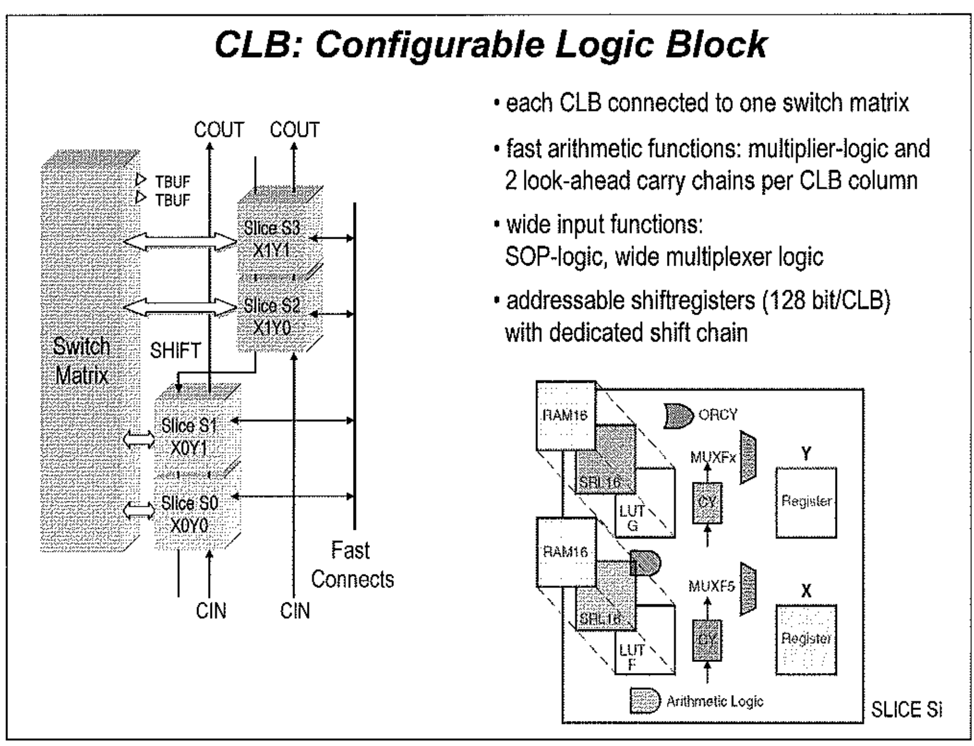
• The *long lines* are bidirectional wires that distribute signals across the device. Vertical and horizontal long lines span the full height and width of the device.

• The *hex lines* route signals to every third or sixth block away in all four directions. Organized in a staggered pattern, hex lines can only be driven from one end. Hex-line signals can be accessed either at the endpoints or at the midpoint (three blocks from the source).

• The *double lines* route signals to every first or second block away in all four directions. Organized in a staggered pattern, double lines can be driven only at their endpoints. Double-line signals can be accessed either at the endpoints or at the midpoint (one block from the source).

• The *direct connect lines* route signals to neighbouring blocks: vertically, horizontally, and diagonally.

• The *fast connect lines* are the internal CLB local interconnections from LUT outputs to LUT inputs.



The Virtex-II configurable logic blocks (CLB) are organized in an array and are used to build combinatorial and synchronous logic designs.

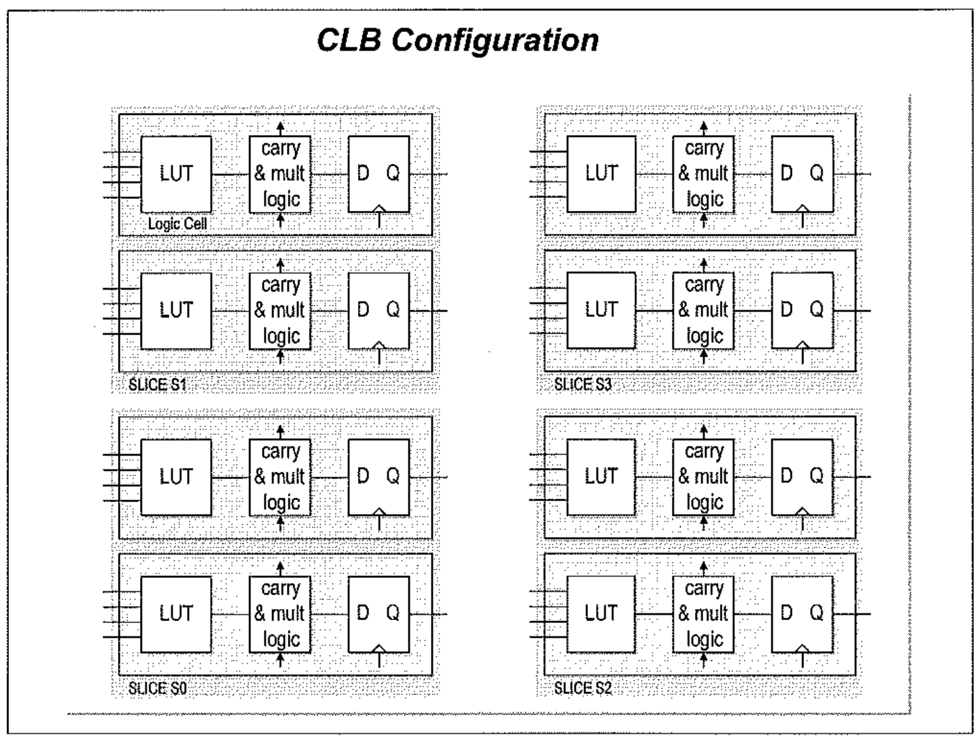
CLB

Each CLB element is tied to a switch matrix to access the general routing matrix. A CLB element comprises 4 similar slices, with fast local feedback within the CLB. The four slices are split in two columns of two slices with two independent carry logic chains and one common shift chain.

SLICE

Each slice includes two 4-input function generators, carry logic, arithmetic logic gates, wide function multiplexers and two storage elements (either edge-triggered D-type flip-flops or level-sensitive latches). Each 4-input function generator is programmable as a 4-input LUT, 16 bits of distributed SelectRAM memory, or a 16-bit variable-tap shift register element.

The output from the function generator in each slice drives both the slice output and the D input of the storage element.



Configurable Logic Blocks (CLBs)

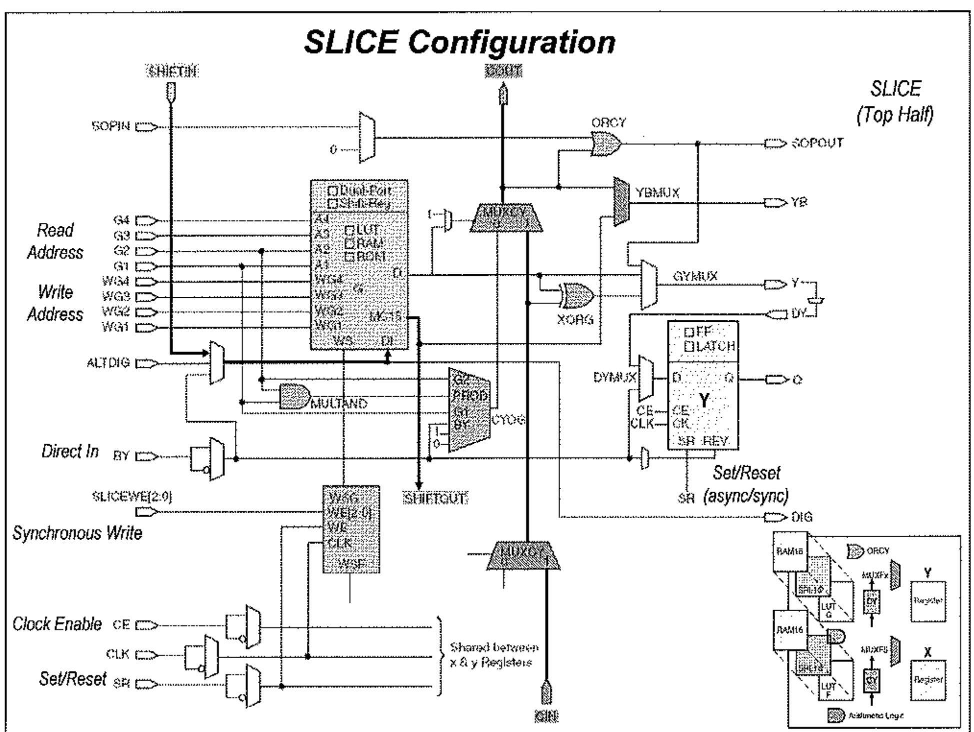
CLB resources include four slices and two 3-state buffers. Each slice is equivalent and contains:

- Two function generators (LUT F & G)
- Two storage elements
- Arithmetic logic gates
- Large multiplexers
- Wide function capability
- Fast carry look-ahead chain
- Horizontal cascade chain (OR gate)

The function generators F & G are configurable as 4-input look-up tables (LUTs), as 16-bit shift registers, or as 16-bit distributed SelectRAM memory.

The two storage elements are either edge-triggered D-type flip-flops or level-sensitive latches.

Each CLB has internal fast interconnect and connects to a switch matrix to access general routing resources.



The output from the function generator in each slice drives both the slice output and the D input of the storage element.

Look-Up Table

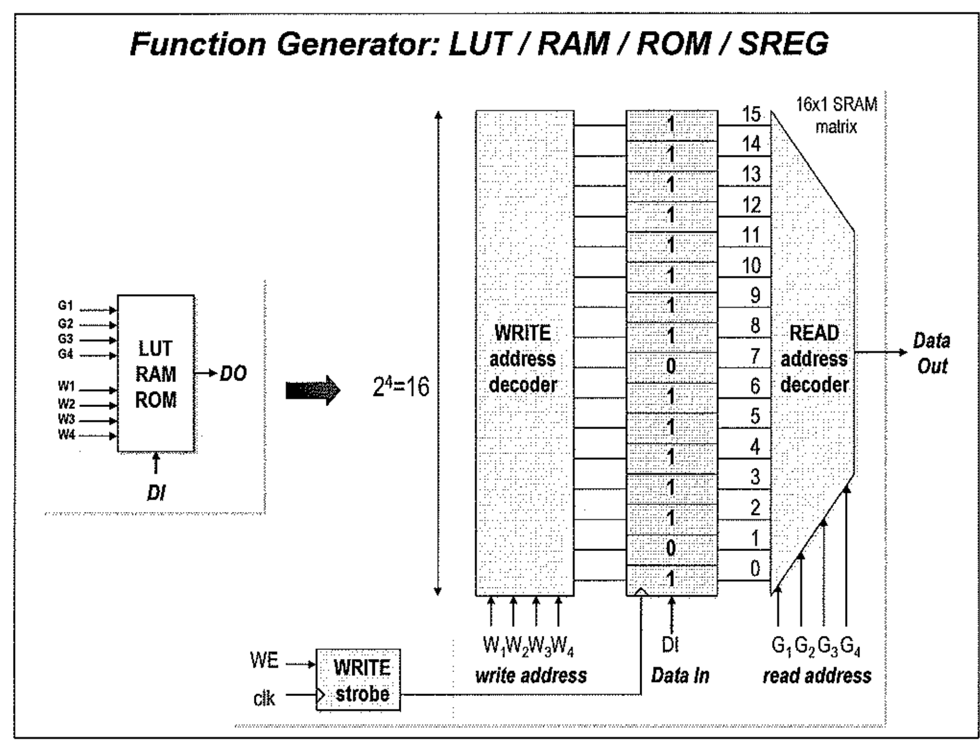
Virtex-II function generators are implemented as 4-input look-up tables (LUTs). Four independent inputs are provided to each of the two function generators in a slice (F and G). These function generators are each capable of implementing any arbitrarily defined boolean function of four inputs. The propagation delay is therefore independent of the function implemented. Signals from the function generators can exit the slice (X or Y output), can input the XOR dedicated gate (for arithmetic logic), or input the carry-logic multiplexer (see fast look-ahead carry logic), or feed the D input of the storage element, or go to the MUXF5 (not shown).

Register/Latch

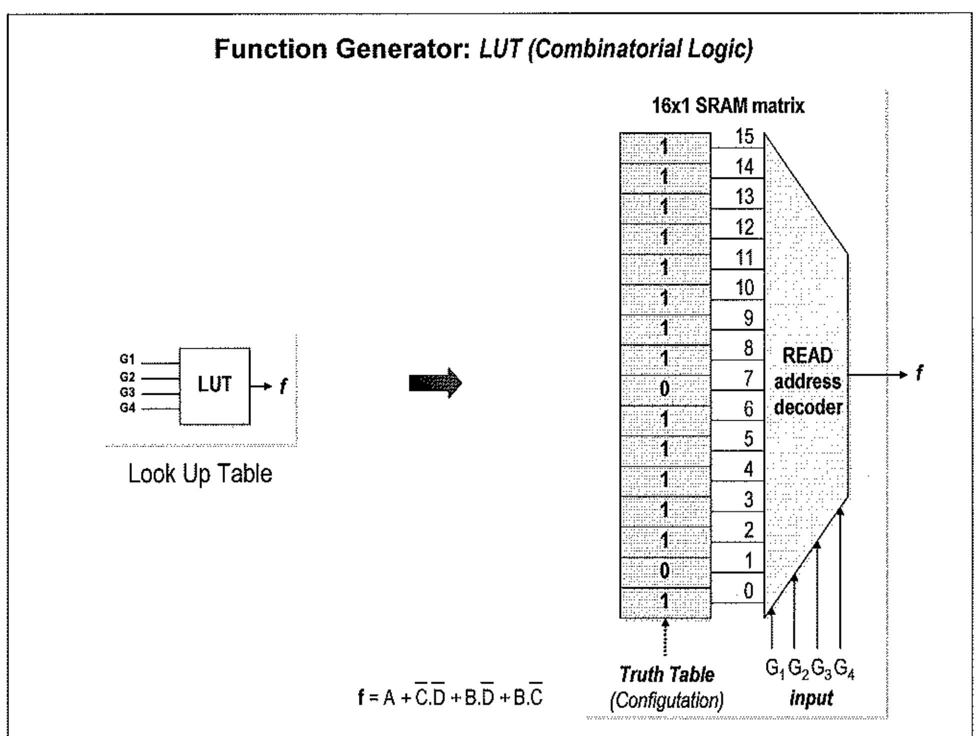
The storage elements in a Virtex-II slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D input can be directly driven by the X or Y output via the DX or DY input, or by the slice inputs bypassing the function generators via the BX or BY input (Direct In). The clock enable signal (CE) is active High by default. If left unconnected, the clock enable for that storage element defaults to the active state.

In addition to clock (CK) and clock enable (CE) signals, each slice has set and reset signals (SR and BY slice inputs). SR forces the storage element into the state specified by the attribute SRHIGH or SRLOW. SRHIGH forces a logic "1"when SR is asserted. SRLOW forces a logic "0". When SR is used, a second input (BY) forces the storage element into the opposite state. The reset condition is predominant over the set condition.

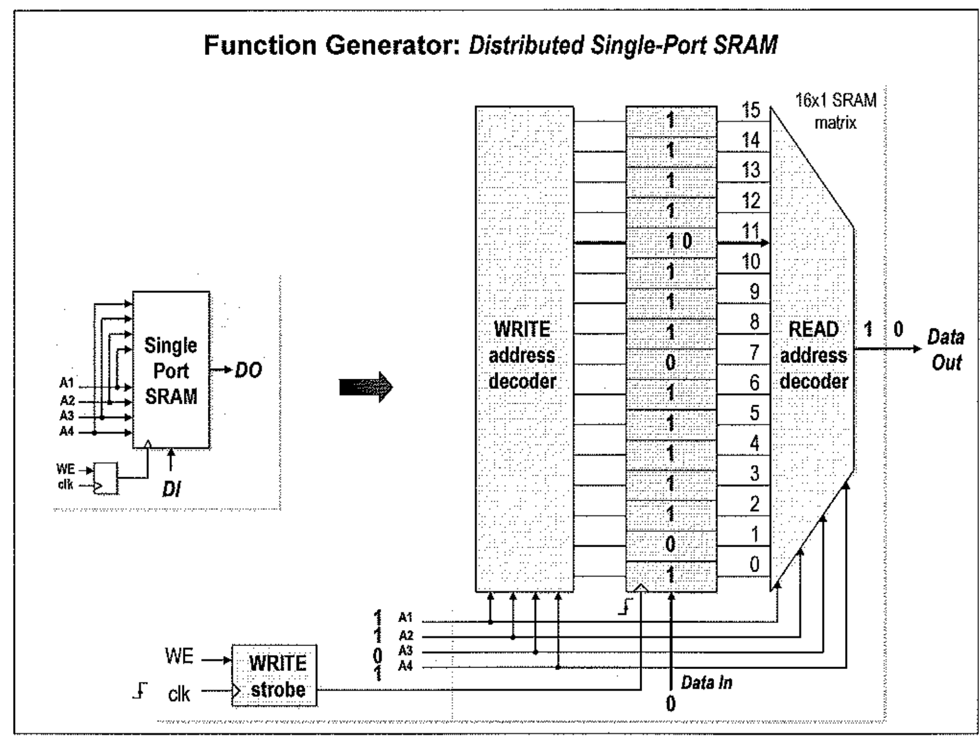
The initial state after configuration or global initial state is defined by a separate INIT0 and INIT1 attribute. For each slice, set and reset can be set to be synchronous or asynchronous. The control signals clock (CLK), clock enable (CE) and set/reset (SR) are common to both storage elements in one slice. All of the control signals have independent polarity.



Distributed SelectRAM memory modules are synchronous (write) resources. The combinatorial read access time is extremely fast, while the synchronous write simplifies high-speed designs. A synchronous read can be implemented with a storage element in the same slice. The distributed SelectRAM memory and the storage element share the same clock input. A Write Enable (WE) input is active High, and is driven by the SR input.

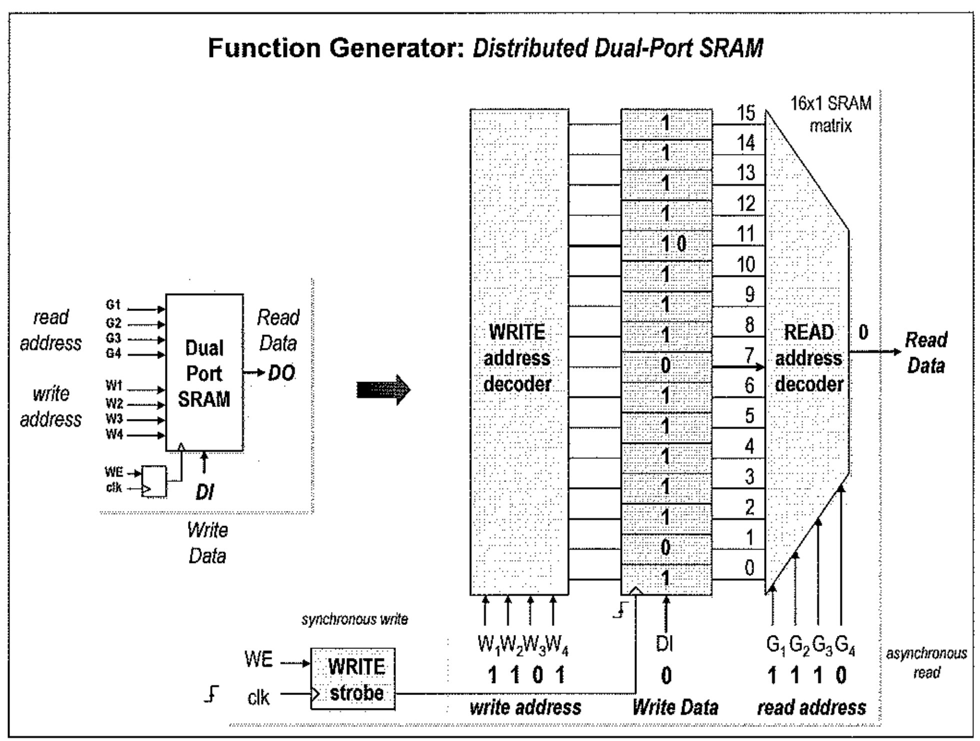


Virtex-II function generators are implemented as 4-input look-up tables (LUTs). Four independent inputs are provided to each of the two function generators in a slice (F and G). These function generators are each capable of implementing any arbitrarily defined boolean function of four inputs. The propagation delay is therefore independent of the function implemented.



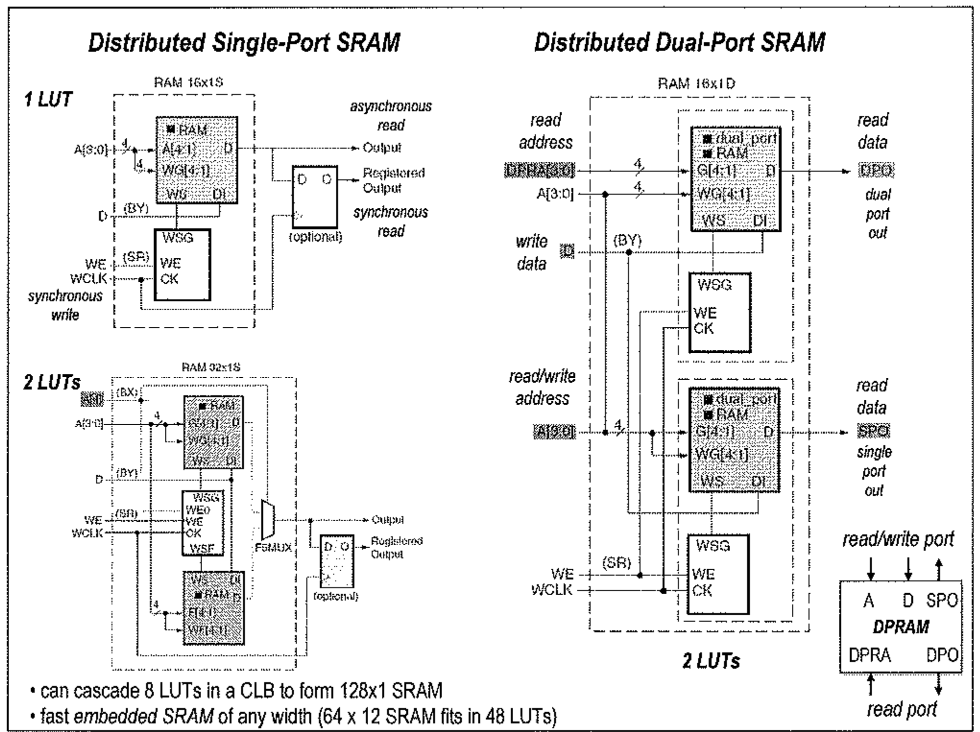
Distributed SelectRAM memory modules are synchronous (write) resources. The combinatorial read access time is extremely fast, while the synchronous write simplifies high-speed designs. A synchronous read can be implemented with a storage element in the same slice. The distributed SelectRAM memory and the storage element share the same clock input. A Write Enable (WE) input is active High, and is driven by the SR input.

For single-port configurations, distributed SelectRAM memory has one address port for synchronous writes and asynchronous reads. Read and write addresses share the same address bus.



For dual-port configurations, distributed SelectRAM memory has one port for synchronous writes and asynchronous reads and another port for asynchronous reads. The function generator (LUT) has separated read address inputs (A1, A2, A3, A4) and write address inputs (WG1/WF1, WG2/WF2, WG3/WF3, WG4/WF4).

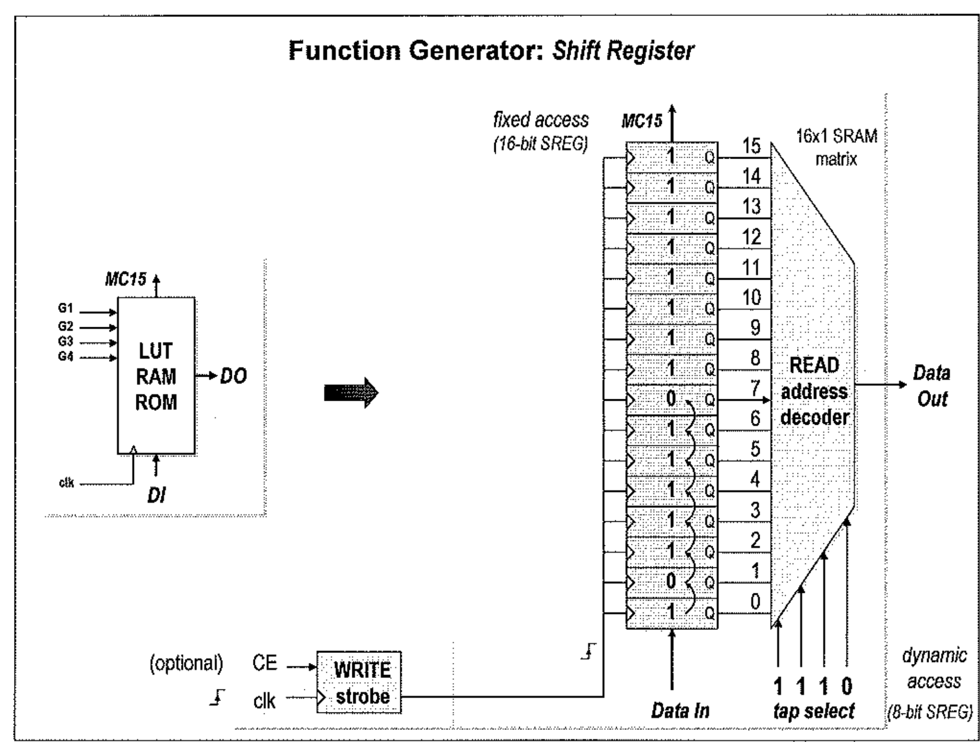
One function generator (R/W port) is connected with shared read and write addresses. The second function generator has the A inputs (read) connected to the second read-only port address and the W inputs (write) shared with the first read/write port address.



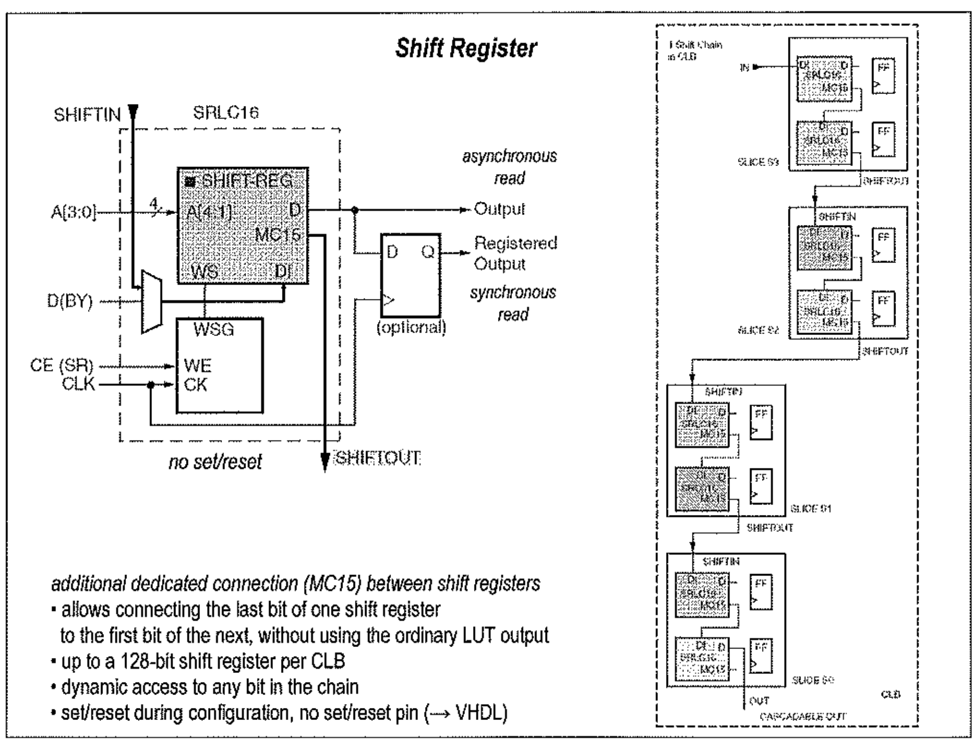
Each function generator (LUT) can implement a 16 x 1-bit synchronous RAM resource called a distributed SelectRAM element. The SelectRAM elements are configurable within a CLB to implement the following:

- Single-Port 16 x 8 bit RAM
- Single-Port 32 x 4 bit RAM
- Single-Port 64 x 2 bit RAM
- Single-Port 128 x 1 bit RAM
- Dual-Port 16 x 4 bit RAM
- Dual-Port 32 x 2 bit RAM
- Dual-Port 64 x 1 bit RAM

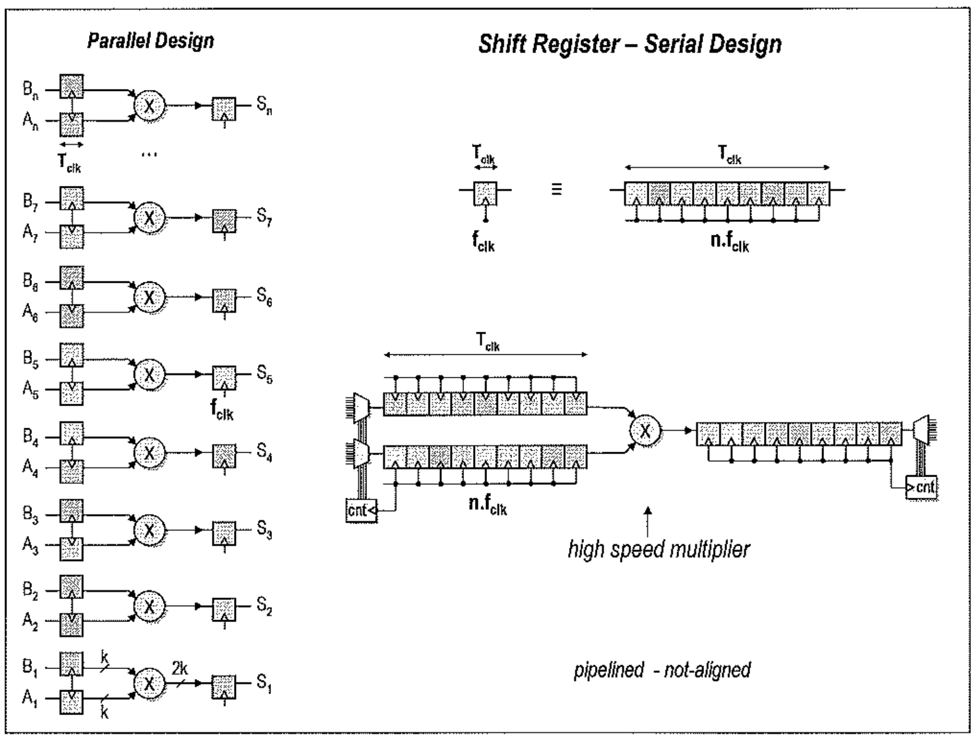
Distributed SelectRAM memory modules are synchronous (write) resources. The combinatorial read access time is extremely fast, while the synchronous write simplifies high-speed designs. A synchronous read can be implemented with a storage element in the same slice. The distributed SelectRAM memory and the storage element share the same clock input. A Write Enable (WE) input is active High, and is driven by the SR input.



Each function generator can also be configured as a 16-bit shift register. The write operation is synchronous with a clock input (CLK) and an optional clock enable CE. A dynamic read access is performed through the 4-bit address bus, A[3:0]. The configurable 16-bit shift register cannot be set or reset. The read is asynchronous, however the storage element or flip-flop is available to implement a synchronous read. The storage element should always be used with a constant address. For example, when building an 8-bit shift register and configuring the addresses to point to the 7th bit, the 8th bit can be the flip-flop. The overall system performance is improved by using the superior clock-to-out of the flip-flops.



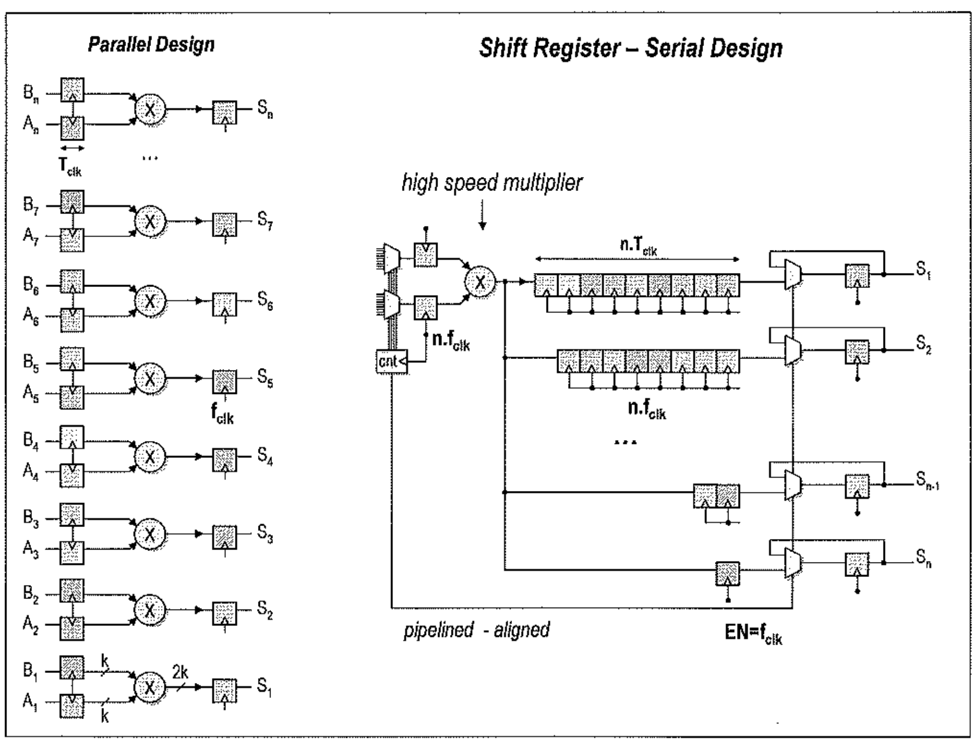
An additional dedicated connection between shift registers allows connecting the last bit of one shift register to the first bit of the next, without using the ordinary LUT output. Longer shift registers can be built with dynamic access to any bit in the chain. The shift register chaining and the MUXF5, MUXF6, and MUXF7 multiplexers allow up to a 128-bit shift register with addressable access to be implemented in one CLB.



With a time multiplexing scheme the shift register can be used to share common hardware with n data streams.

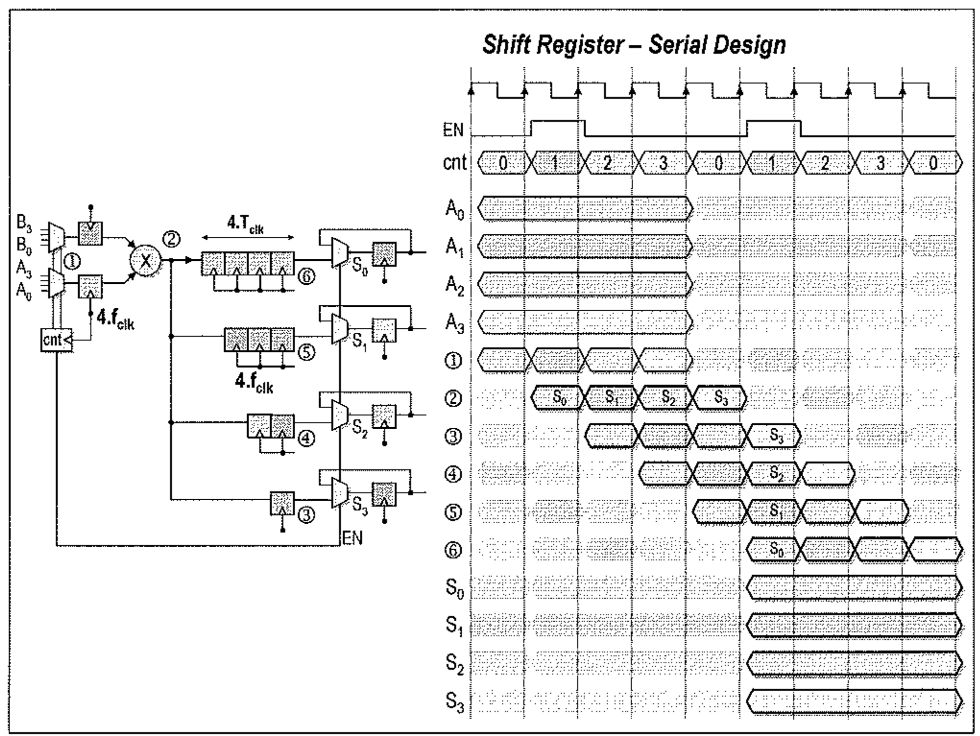
The clock speed must be increased with n. An original delay of T_{clk} must be implemented by a shift register of n flip-flop's.

The shared hardware must run at a n times higher clock speed.



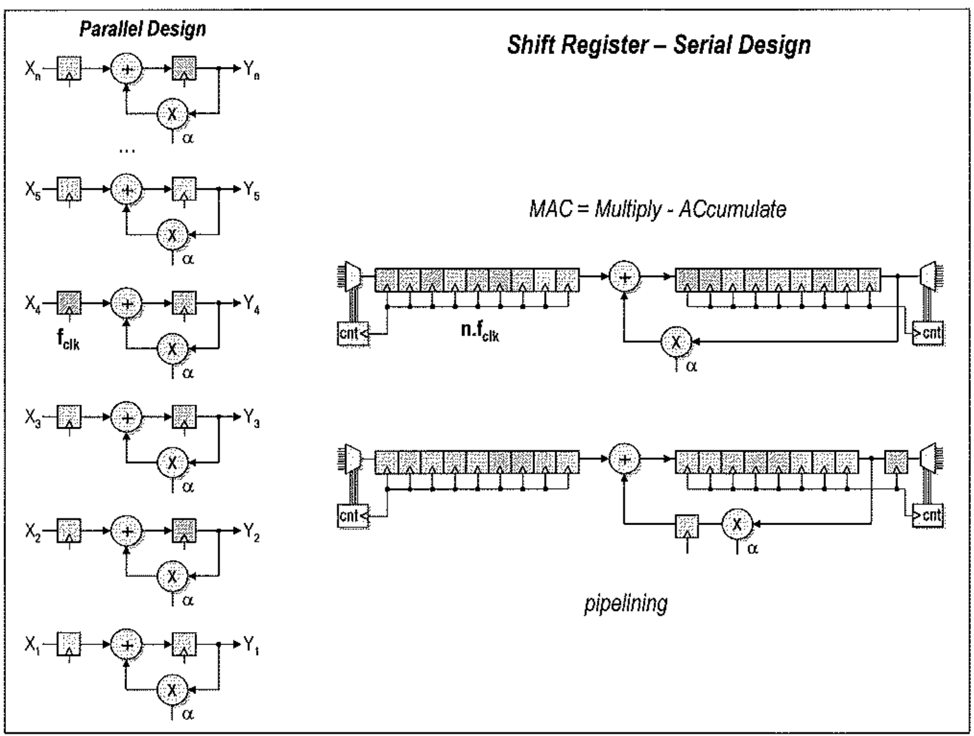
In a time multiplexing scheme the original parallel and aligned data is reduced by a factor n and placed in a sequential data stream. This is the parallel-to-serial conversion.

After processing by the shared hardware a serial-to-parallel conversion is executed. To align the resulting parallel data a delay must be introduced which depends on the position of the serial data in the sequential data stream. The data processed first will be delayed for the longest time. The different delays can be implemented by independent shift registers. A single shift register with n sequential branches could also be used, but this has to be implemented with flip-flop's because the shift registers implemented in the LUTs have only a single output.

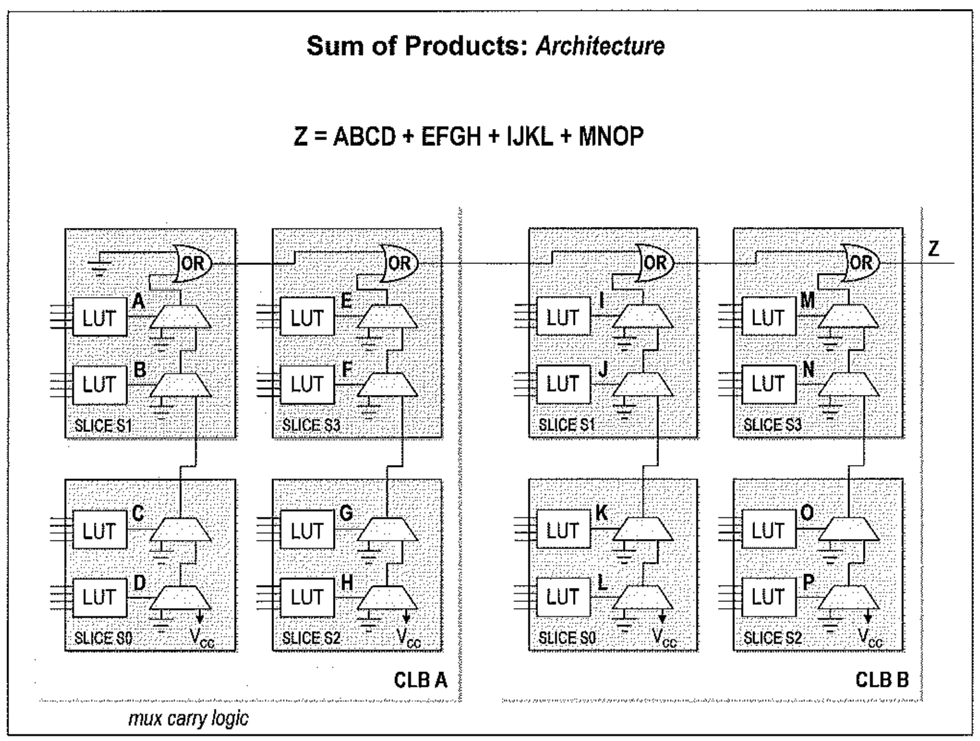


In a time multiplexing scheme the original parallel and aligned data is reduced by a factor n and placed in a sequential data stream. This is the parallel-to-serial conversion.

After processing by the shared hardware a serial-to-parallel conversion is executed. To align the resulting parallel data a delay must be introduced which depends on the position of the serial data in the sequential data stream. The data processed first will experience the longest delay.



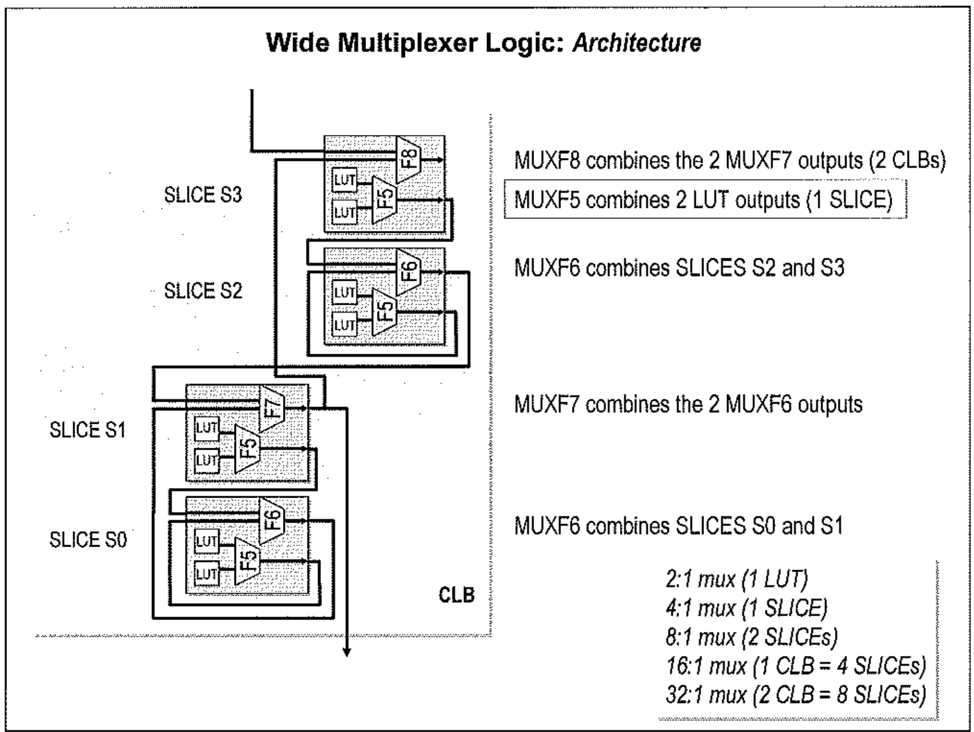
Pipelining can be used to increase the system clock speed. Different combinatorial operations are separated by registers to reduce the critical path in the system.



Sum of Products

Each Virtex-II slice has a dedicated OR gate named ORCY, ORing together outputs from the slices carryout and the ORCY from an adjacent slice. The ORCY gate with the dedicated Sum of Products (SOP) chain are designed for implementing large, flexible SOP chains. One input of each ORCY is connected through the fast SOP chain to the output of the previous ORCY in the same slice row. The second input is connected to the output of the top MUXCY in the same slice.

LUTs and MUXCYs can implement large AND gates or other combinatorial logic functions.



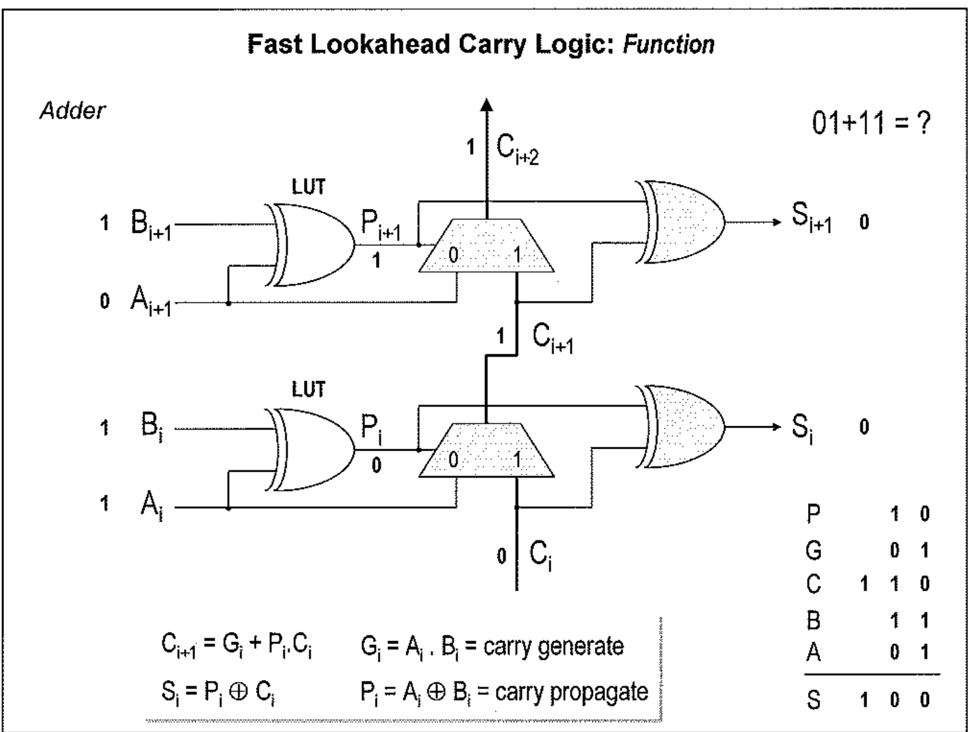
In addition to the basic LUTs, the Virtex-II slice contains logic (MUXF5 and MUXFX multiplexers) that combines function generators to provide any function of five, six, seven, or eight inputs. The MUXFX are either MUXF6, MUXF7 or MUXF8 according to the slice considered in the CLB. Selected functions up to nine inputs (MUXF5 multiplexer) can be implemented in one slice. The MUXFX can also be a MUXF6, MUXF7, or MUXF8 multiplexers to map any functions of six, seven, or eight inputs and selected wide logic functions.

Virtex-II function generators and associated multiplexers can implement the following:

- 4:1 multiplexer in one slice
- 8:1 multiplexer in two slices
- 16:1 multiplexer in one CLB element (4 slices)
- 32:1 multiplexer in two CLB elements (8 slices)

Each Virtex-II slice has one MUXF5 multiplexer and one MUXFX multiplexer. The MUXFX multiplexer implements the MUXF6, MUXF7, or MUXF8.

Each CLB element has two MUXF6 multiplexers, one MUXF7 multiplexer and one MUXF8 multiplexer. Any LUT can implement a 2:1 multiplexer.

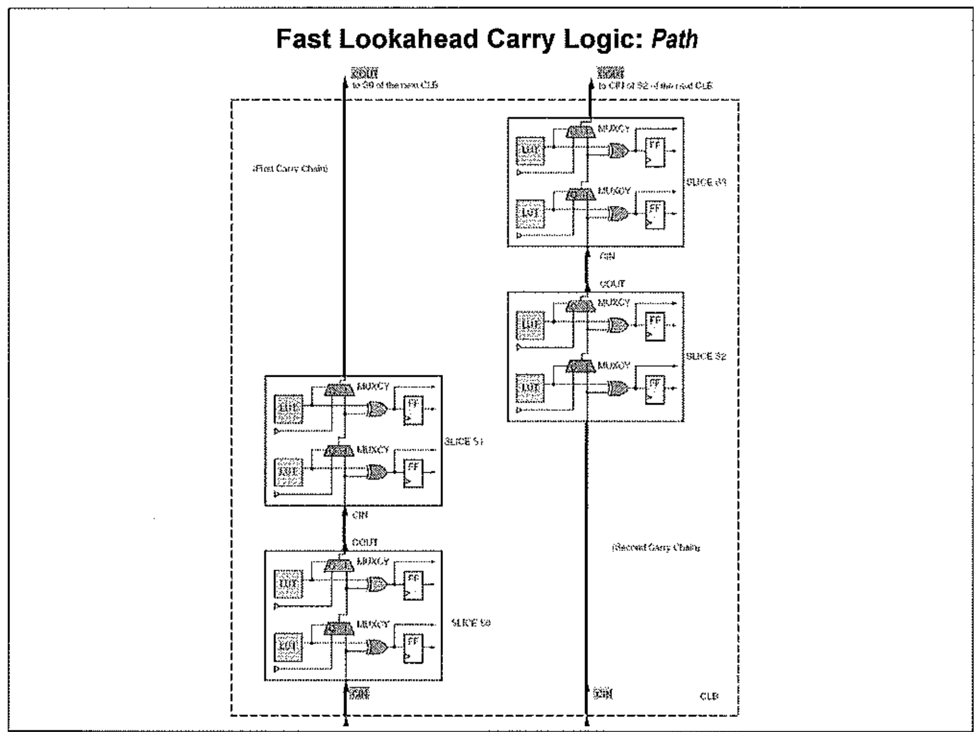


Fast Lookahead Carry Logic

Dedicated carry logic provides fast arithmetic addition and subtraction. The Virtex-II CLB has two separate carry chains. The height of the carry chains is two bits per slice. The carry chain in the Virtex-II device is running upward. The dedicated carry path and carry multiplexer (MUXCY) can also be used to cascade function generators for implementing wide logic functions.

Arithmetic Logic

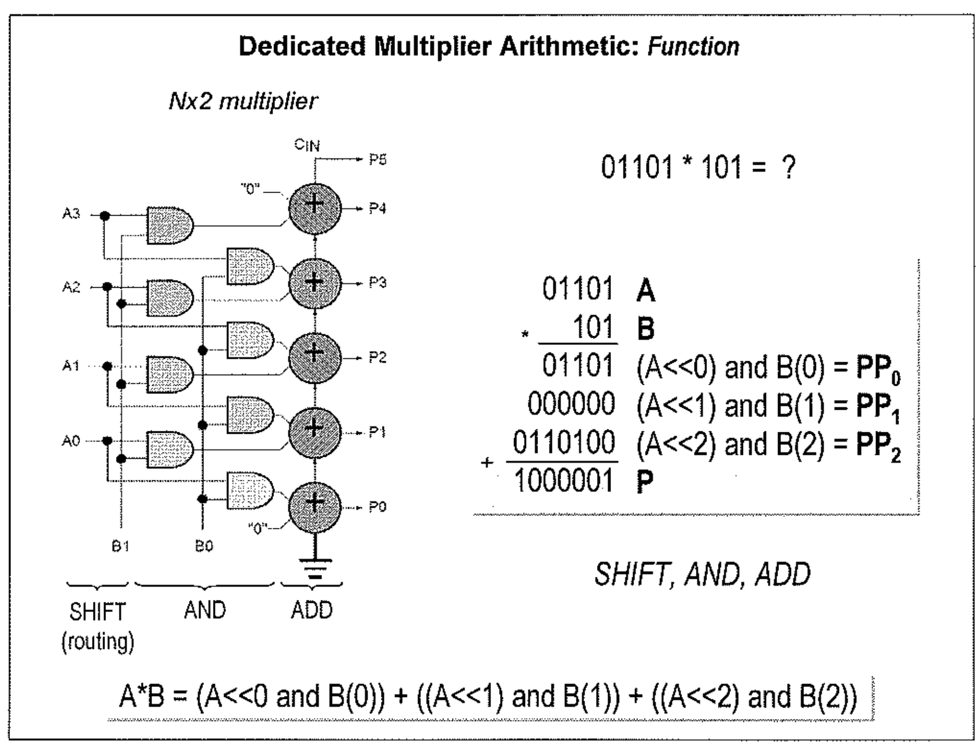
The arithmetic logic includes an XOR gate that allows a 2-bit full adder to be implemented within a slice. A dedicated AND (MULT_AND) gate improves the efficiency of multiplier implementation.



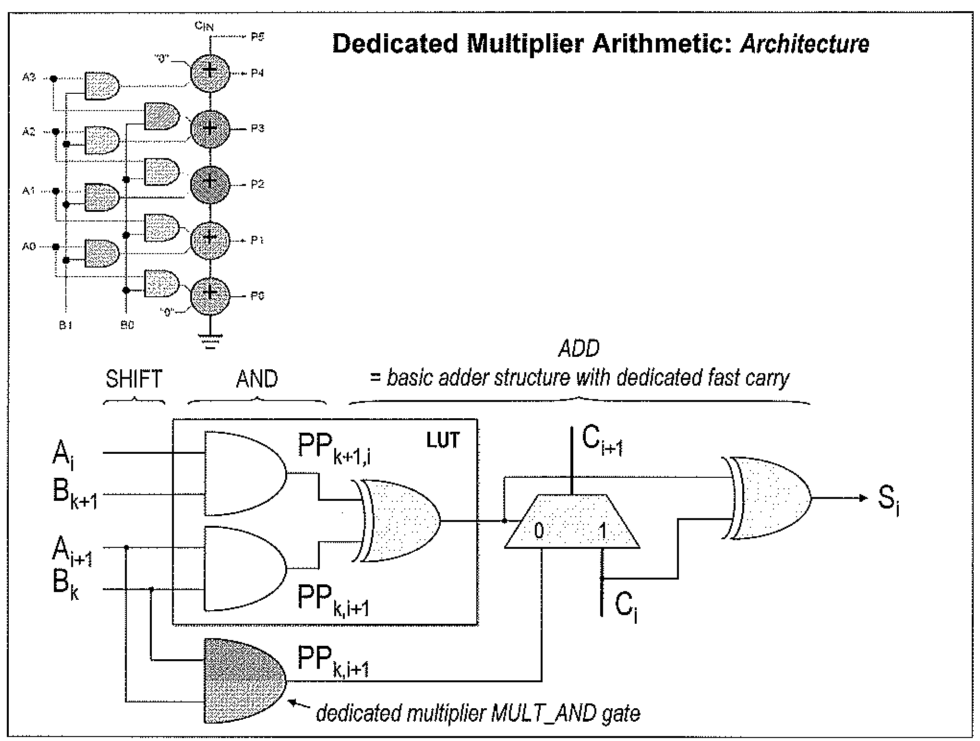
Dedicated carry logic provides fast arithmetic carry capability for high speed arithmetic functions.

There are two separate carry chains in the CLB, one per slice. The height of the carry chains is 4 bits per CLB. The logic consists of a 2-input MUX and an XOR gate. The XOR gate allows a 1-bit full adder to be implemented for 1 bit.

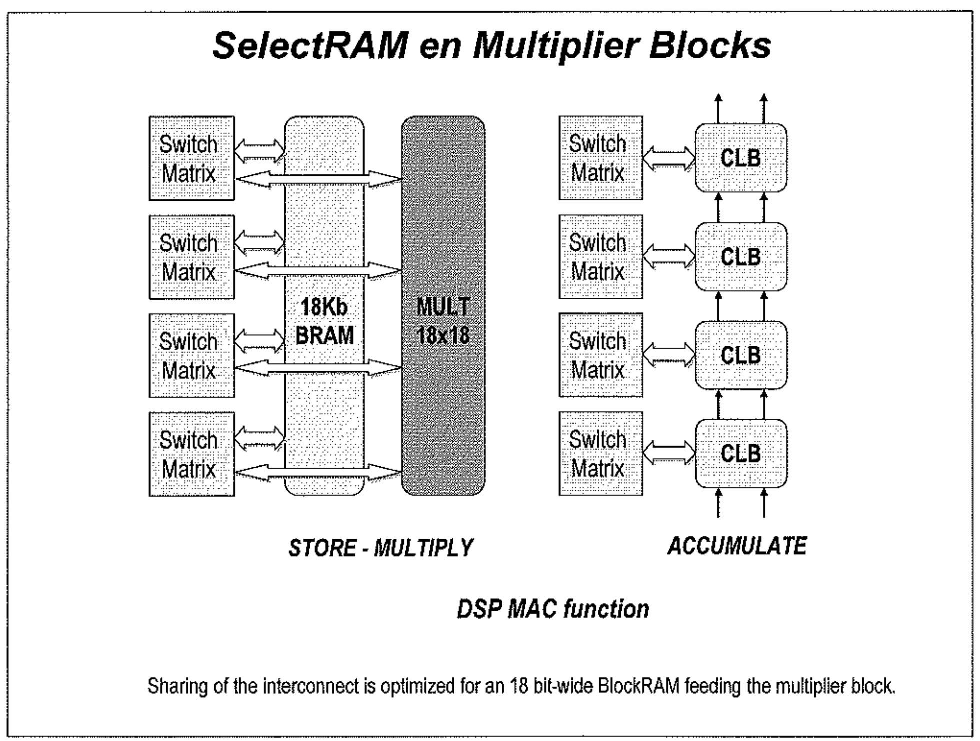
In addition, a dedicated AND gate improves the efficiency of the multiplier implementation. The dedicated carry path is used to cascade LUT functions for implementing wide logic functions. This reduces logic delays due to the decreased number of logic levels even for very high fan-in functions.



The multiplication of two (unsigned) binary multiplicands is essentially a series of shift and add operations. The figure illustrates how an N x 2 Full Multiplier is implemented using the same logic resources as a simple adder. The key to implementing multipliers efficiently in a Virtex device is leveraging the extra AND gate (next to each LUT) and the carry-chain logic.



A dedicated AND (MULT_AND) gate improves the efficiency of multiplier implementation.

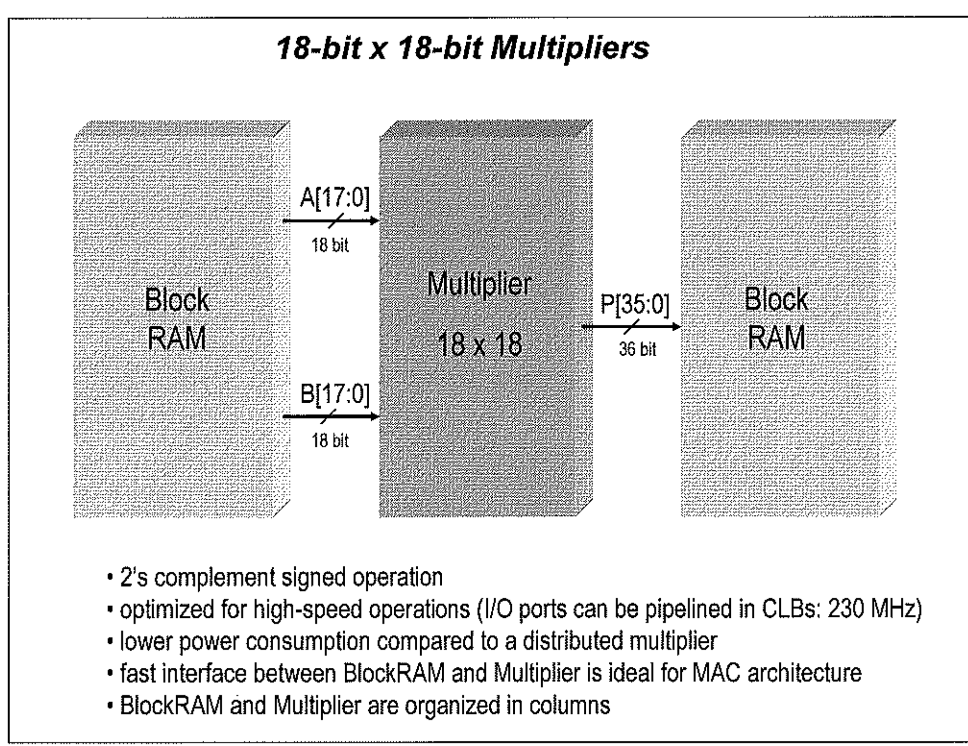


Each SelectRAM memory and multiplier block is tied to four switch matrices.

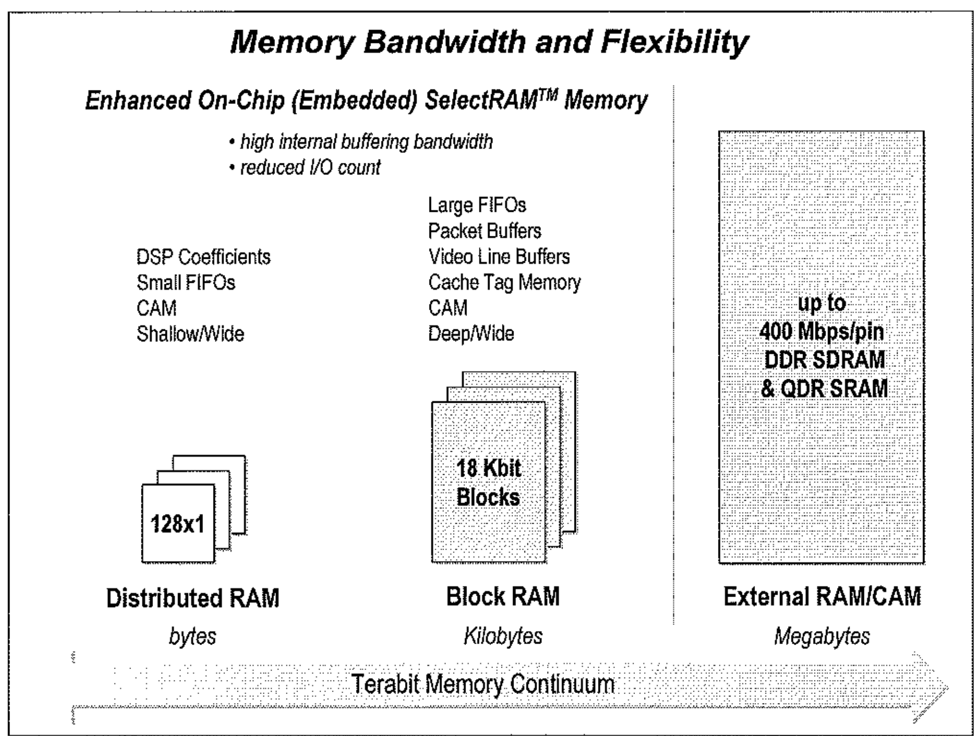
The interconnect is designed to allow SelectRAM memory and multiplier blocks to be used at the same time, but some interconnect is shared between the SelectRAM and the multiplier. Thus, SelectRAM memory can be used only up to 18 bits wide when the multiplier is used, because the multiplier shares inputs with the upper data bits of the SelectRAM memory.

This sharing of the interconnect is optimized for an 18-bit-wide block SelectRAM resource feeding the multiplier.

The use of SelectRAM memory and the multiplier with an accumulator in LUTs allows for implementation of a digital signal processor (DSP) multiplier-accumulator (MAC) function, which is commonly used in finite and infinite impulse response (FIR and IIR) digital filters.



A Virtex-II multiplier block is an 18-bit by 18-bit 2's complement signed multiplier. Virtex-II devices incorporate many embedded multiplier blocks. These multipliers can be associated with an 18 Kbit block SelectRAM resource or can be used independently. They are optimized for high-speed operations and have a lower power consumption compared to an 18-bit x 18-bit multiplier in slices.



A terabit memory continuum is available:

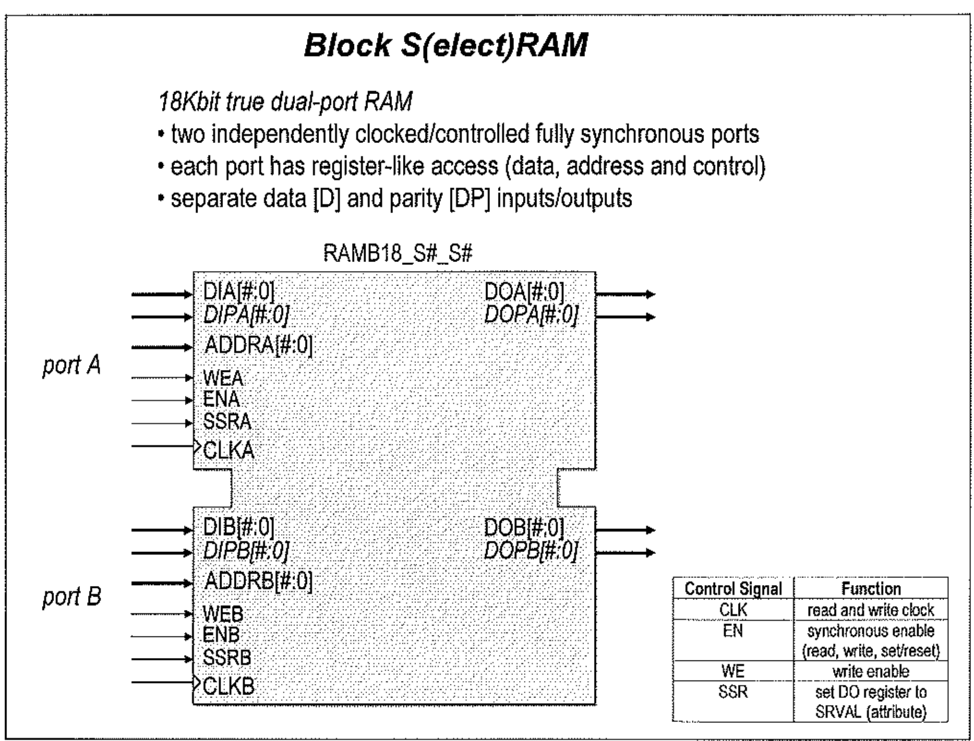
SelectRAM™ Memory Hierarchy

Virtex-II devices incorporate large amounts of 18 Kbit block SelectRAM. These complement the distributed SelectRAM resources that provide shallow RAM structures implemented in CLBs.

- 3 Mb of dual-port RAM in 18 Kbit *block* SelectRAM resources
- up to 1.5 Mb of *distributed* SelectRAM resources

High-Performance Interfaces to External Memory (IOB)

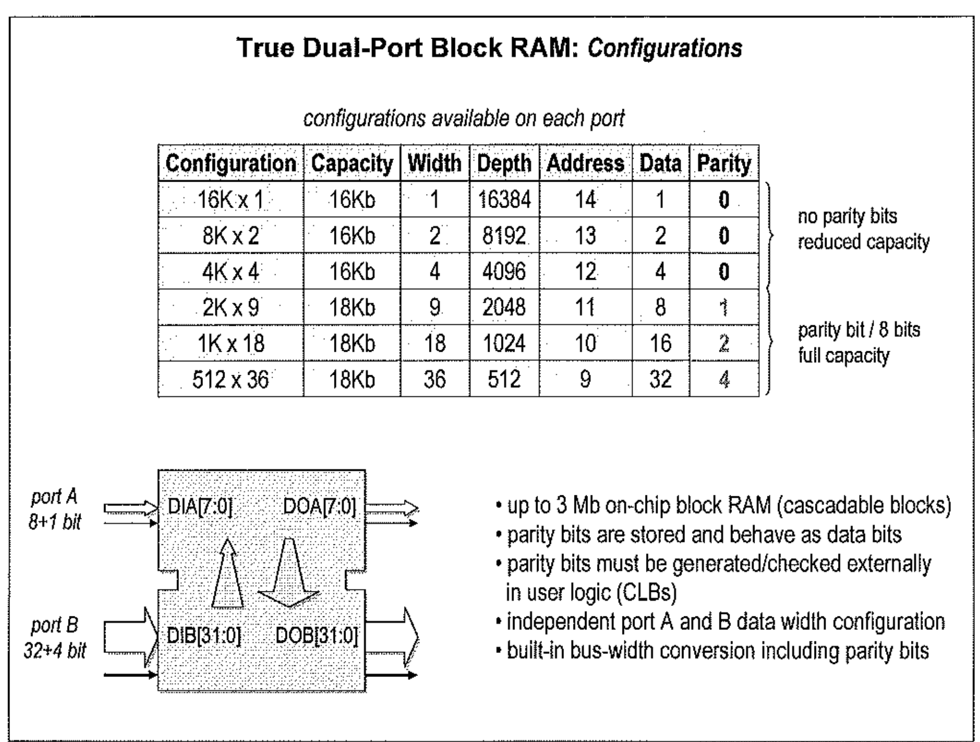
- DRAM interfaces
  - SDR / DDR SDRAM
  - Network FCRAM
  - Reduced Latency DRAM
- SRAM interfaces
  - SDR / DDR SRAM
  - QDR™ SRAM
- CAM interfaces



Virtex-II devices incorporate large amounts of 18 Kbit block SelectRAM. These complement the distributed SelectRAM resources that provide shallow RAM structures implemented in CLBs. Each Virtex-II block SelectRAM is an 18 Kbit true dual-port RAM with two independently clocked and independently controlled synchronous ports that access a common storage area. Both ports are functionally identical. CLK, EN, WE, and SSR polarities are defined through configuration.

Each port has the following types of inputs: Clock and Clock Enable, Write Enable, Set/Reset, and Address, as well as separate Data/parity data inputs (for write) and Data/parity data outputs (for read).

Operation is synchronous; the block SelectRAM behaves like a register. Control, address and data inputs must (and need only) be valid during the set-up time window prior to a rising (or falling, a configuration option) clock edge. Data outputs change as a result of the same clock edge.



True Dual-Port Block RAM: ***Configurations***

configurations available on each port

Configuration	Capacity	Width	Depth	Address	Data	Parity
16K x 1	16Kb	1	16384	14	1	0
8K x 2	16Kb	2	8192	13	2	0
4K x 4	16Kb	4	4096	12	4	0
2K x 9	18Kb	9	2048	11	8	1
1K x 18	18Kb	18	1024	10	16	2
512 x 36	18Kb	36	512	9	32	4

The Virtex-II block SelectRAM supports various configurations, including single- and dual-port RAM and various data/address aspect ratios.

Single-Port Configuration

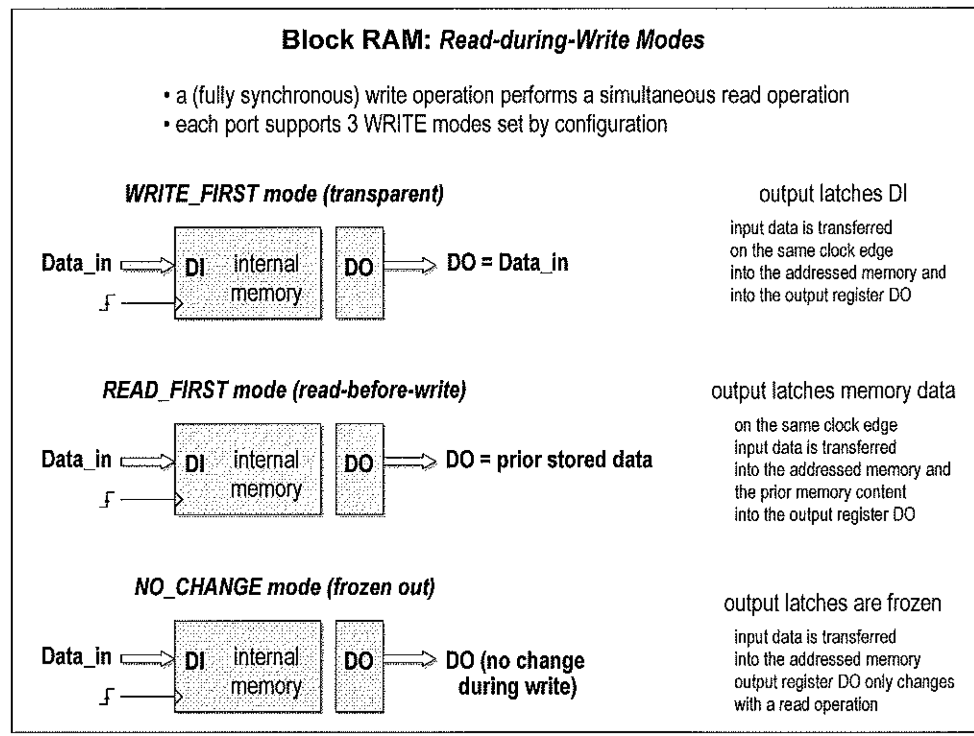
As a single-port RAM, the block SelectRAM has access to the 18 Kbit memory locations in any of the 2K x 9-bit, 1K x 18-bit, or 512 x 36-bit configurations and to 16 Kbit memory locations in any of the 16K x 1-bit, 8K x 2-bit, or 4K x 4-bit configurations. The advantage of the 9-bit, 18-bit and 36-bit widths is the ability to store a parity bit for each eight bits. Parity bits must be generated or checked externally in user logic. In such cases, the width is viewed as 8 + 1, 16 + 2, or 32 + 4. These extra parity bits are stored and behave exactly as the other bits, including the timing parameters. Video applications can use the 9-bit ratio of Virtex-II block SelectRAM memory to advantage.

Each block SelectRAM cell is a fully synchronous memory. Input data bus and output data bus widths are identical.

Dual-Port Configuration

As a dual-port RAM, each port of block SelectRAM has access to a common 18 Kbit memory resource. These are fully synchronous ports with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion.

If both ports are configured in either 2K x 9-bit, 1K x 18-bit, or 512 x 36-bit configurations, the 18 Kbit block is accessible from port A or B. If both ports are configured in either 16K x 1-bit, 8K x 2-bit. or 4K x 4-bit configurations, the 16 K-bit block is accessible from Port A or Port B. All other configurations result in one port having access to an 18 Kbit memory block and the other port having access to a 16 K-bit subset of the memory block equal to 16 Kbits.



Read/Write Operations

The Virtex-II block SelectRAM read operation is fully synchronous. An address is presented, and the read operation is enabled by control signals WEA and WEB in addition to ENA or ENB. Then, depending on clock polarity, a rising or falling clock edge causes the stored data to be loaded into output registers.

The write operation is also fully synchronous. Data and address are presented, and the write operation is enabled by control signals WEA or WEB in addition to ENA or ENB. Then, again depending on the clock input mode, a rising or falling clock edge causes the data to be loaded into the memory cell addressed.

A write operation performs a simultaneous read operation. Three different options are available, selected by configuration:

1. "WRITE_FIRST"

The "WRITE_FIRST" option is a transparent mode. The same clock edge that writes the data input (DI) into the memory also transfers DI into the output registers DO.

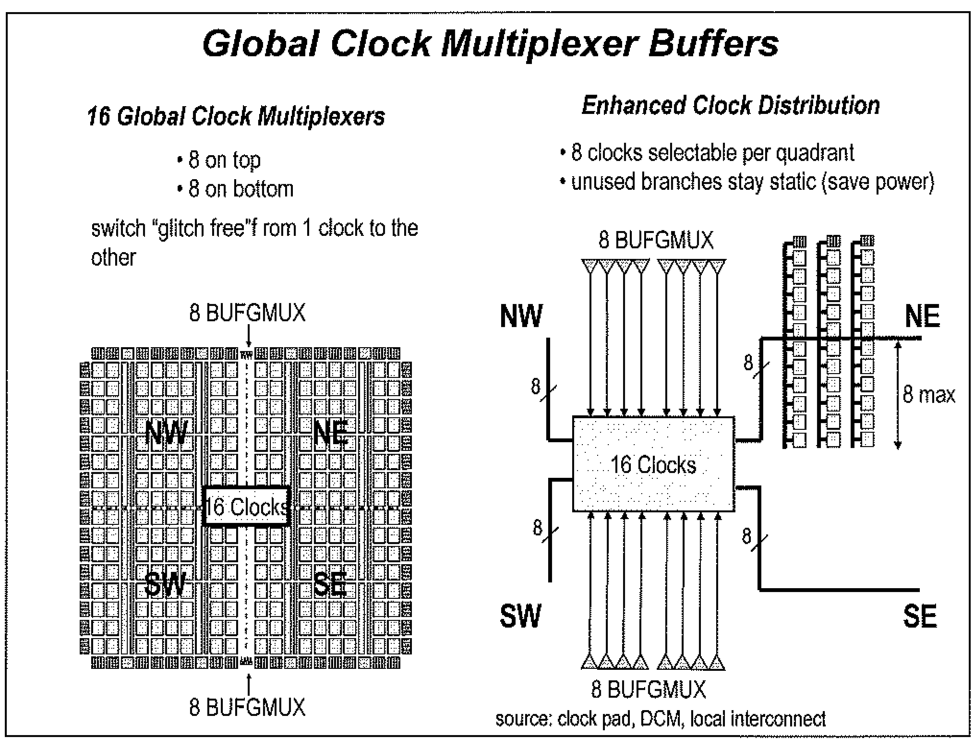
2. "READ_FIRST"

The "READ_FIRST" option is a read-before-write mode.

The same clock edge that writes data input (DI) into the memory also transfers the prior content of the memory cell addressed into the data output registers DO.

3. "NO_CHANGE"

The "NO_CHANGE" option maintains the content of the output registers, regardless of the write operation. The clock edge during the write mode has no effect on the content of the data output register DO. When the port is configured as "NO_CHANGE", only a read operation loads a new value in the output register DO.



The DCM and global clock multiplexer buffers provide a complete solution for designing high-speed clocking schemes.

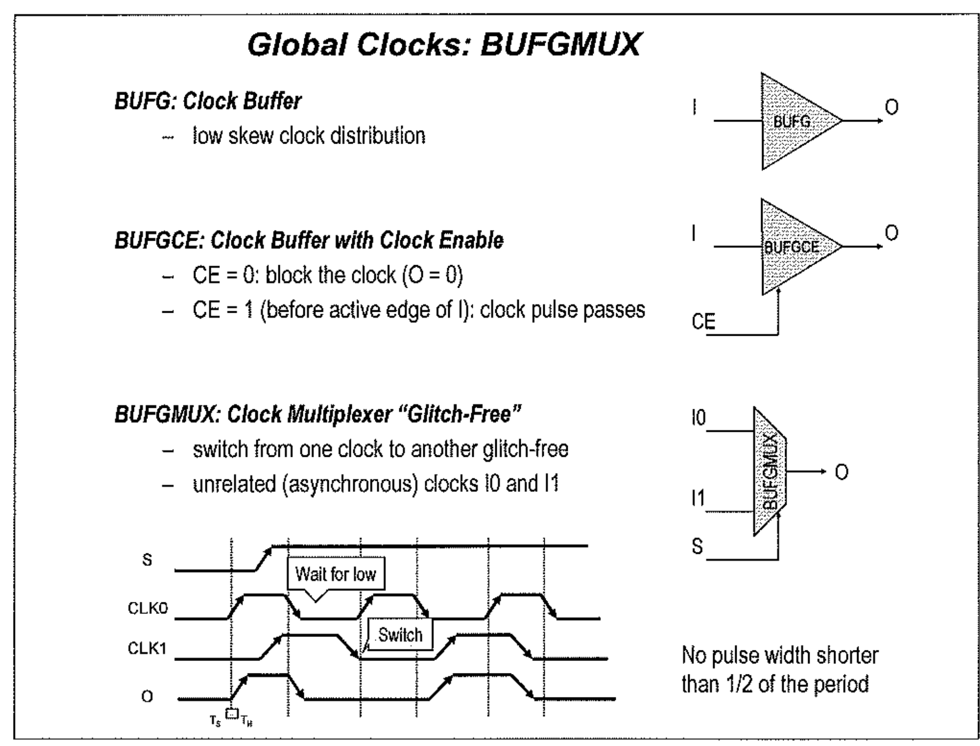
Virtex-II devices have 16 clock input pins that can also be used as regular user I/Os. Eight clock pads are on the top edge of the device, in the middle of the array, and eight are on the bottom edge. The global clock multiplexer buffer represents the input to dedicated low-skew clock tree distribution in Virtex-II devices. Like the clock pads, eight global clock multiplexer buffers are on the top edge of the device and eight are on the bottom edge.

Each global clock buffer can either be driven by the clock pad to distribute a clock directly to the device, or driven by the Digital Clock Manager (DCM).

Each global clock buffer can also be driven by local interconnects. The DCM has clock output(s) that can be connected to global clock buffer inputs. Global clock buffers are used to distribute the clock to some or all synchronous logic elements (such as registers in CLBs and IOBs, and SelectRAM blocks. Eight global clocks can be used in each quadrant of the Virtex-II device.

Clock distribution in Virtex-II devices:

In each quadrant, up to eight clocks are organized in clock rows. A clock row supports up to 16 CLB rows (eight up and eight down). For the largest devices a new clock row is added, as necessary.



Global clocks are driven by dedicated clock buffers (BUFG), which can also be used to gate the clock (BUFGCE) or to multiplex between two independent clock inputs (BUFGMUX).

BUFGCE

If the CE input is active (High) prior to the incoming rising clock edge, the following clock pulse passes through the clock buffer. Any level change of CE during the incoming clock High time has no effect.

If the CE input is inactive (Low) prior to the incoming rising clock edge, the following clock pulse does not pass through the clock buffer, and the output stays Low. Any level change of CE during the incoming clock High time has no effect.

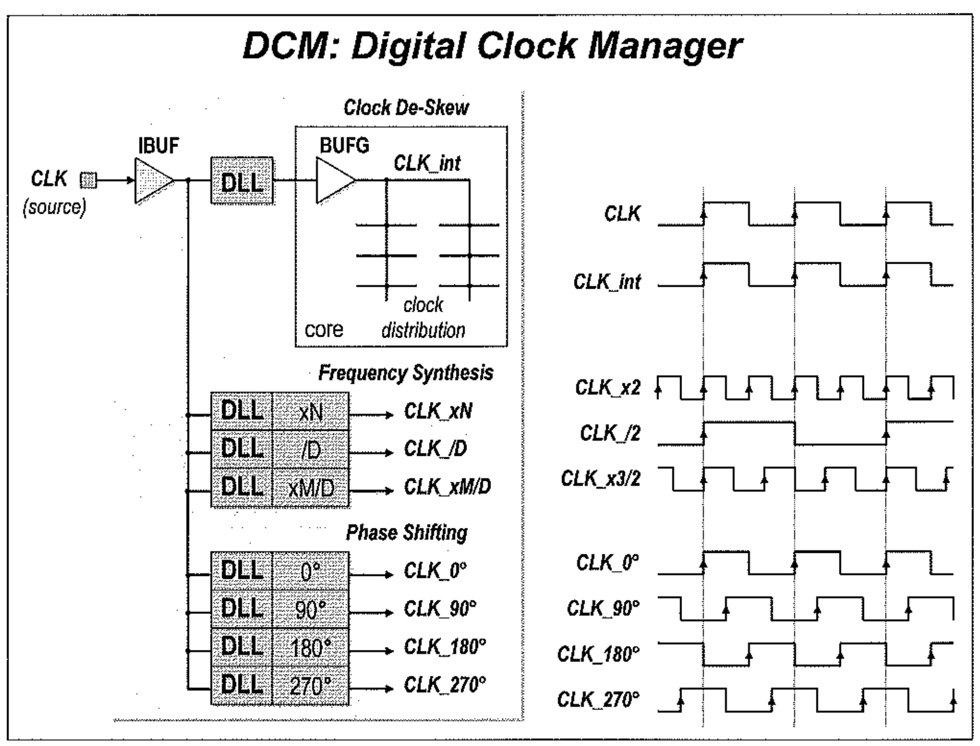
CE must not change during a short setup window just prior to the rising clock edge on the BUFGCE input I. Violating this setup time requirement can result in an undefined runt pulse output.

BUFGMUX

BUFGMUX can switch between two unrelated, even asynchronous clocks. Basically, a Low on S selects the I0 input, a High on S selects the I1 input. Switching from one clock to the other is done in such a way that the output High and Low time is never shorter than the shortest High or Low time of either input clock. As long as the presently selected clock is High, any level change of S has no effect .

If the presently selected clock is Low while S changes, or if it goes Low after S has changed, the output is kept Low until the other ("to-be-selected") clock has made a transition from High to Low. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the rising edge of the presently selected clock; that is, prior to the rising edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.



The Virtex-II DCM offers a wide range of powerful clock management features.

• **Clock De-skew**: The DCM generates new system clocks (either internally or externally to the FPGA), which are phase-aligned to the input clock, thus eliminating clock distribution delays.

• **Frequency Synthesis**: The DCM generates a wide range of output clock frequencies, performing very flexible clock multiplication and division. Very flexible frequency synthesis provides a clock output frequency equal to any M/D ratio of the input clock frequency, where M and D are two integers. To generate de-skewed internal/ external clocks, each DCM can be used to eliminate clock distribution delay.

Frequency synthesis can be used in multiple clock domain applications:

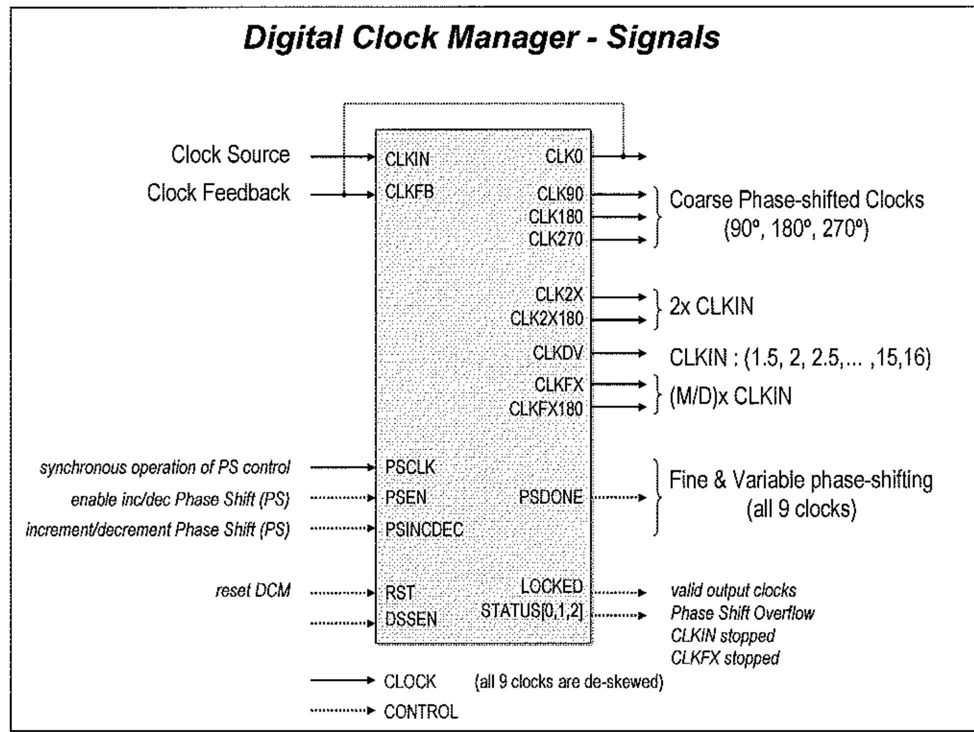
- fast internal clock, slow external clock
- time domain multiplexing (use one circuit twice/clock cycle)

• **Phase Shifting**: The DCM provides coarse phase shifting (90-, 180-, and 270-degree phase-shifted versions of the output clocks). This can be used in clock multiplexed applications:

  - 0° - 180°: DDR (Double Data Rate)
  - 0° - 90° - 180° - 270° : QDR (Quadruple Data Rate)

Fine-grained phase shifting with dynamic phase shift control offers high-resolution phase adjustments in increments of 1/256 of the clock period.

The DCM utilizes fully digital delay lines allowing robust high-precision control of clock phase and frequency. It also utilizes fully digital feedback systems, operating dynamically to compensate for temperature and voltage variations during operation.



Up to 4 of the 9 DCM clock outputs can drive inputs to global clock buffers or global clock multiplexer buffers simultaneously. All DCM clock outputs can simultaneously drive general routing resources, including routes to output buffers.

- The CLK2X and CLK2X180 outputs double the clock frequency.
- The CLKDV output creates divided output clocks with division options of 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- The CLKFX and CLKFX180 outputs can be used to produce clocks at the following frequency:

FREQCLKFX = (M/D) * FREQCLKIN where M and D are two integers.

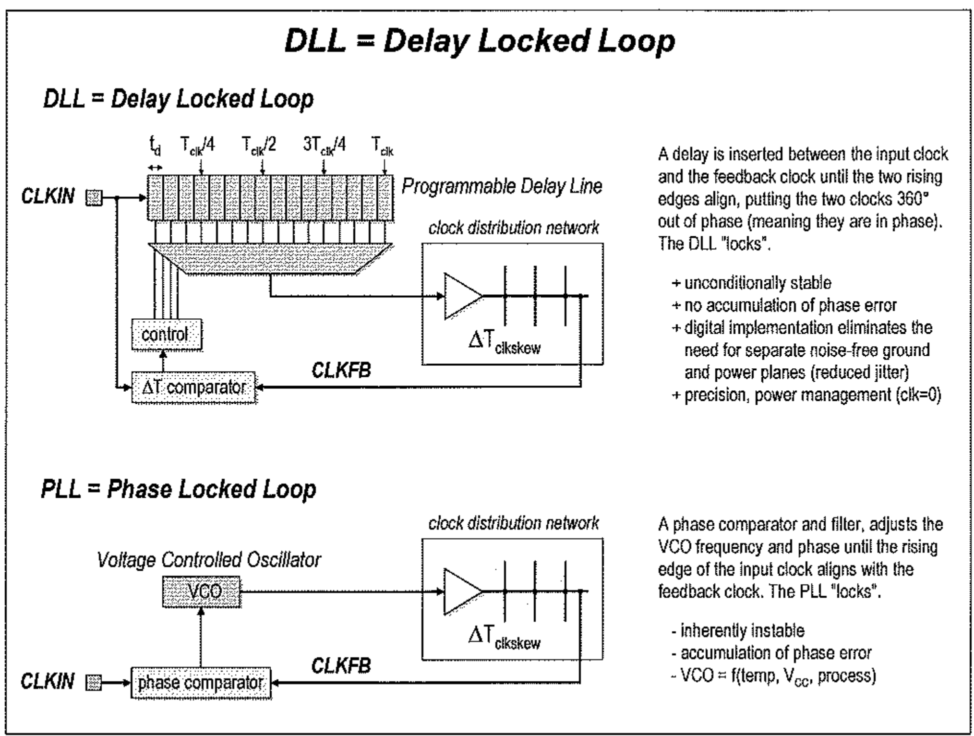
Very flexible frequency synthesis provides a clock output frequency equal to any M/D ratio of the input clock frequency, where M and D are two integers. To generate de-skewed internal/ external clocks, each DCM can be used to eliminate clock distribution delay.

- CLK2X180 is phase shifted 180 degrees relative to CLK2X.
- CLKFX180 is phase shifted 180 degrees relative to CLKFX.

The DCM has the following general control signals:

- RST input pin: resets the entire DCM
- LOCKED output pin: asserted High when all enabled DCM circuits have locked.
- STATUS output pins (active High): STATUS[0] = Phase Shift Overflow
  STATUS[1] = CLKIN Stopped
  STATUS[2] = CLKFX Stopped

The DCM can be configured to delay the completion of the Virtex-II configuration process until after the DCM has achieved lock. This guarantees that the chip does not begin operating until after the system clocks generated by the DCM have stabilized.



Clock De-Skew

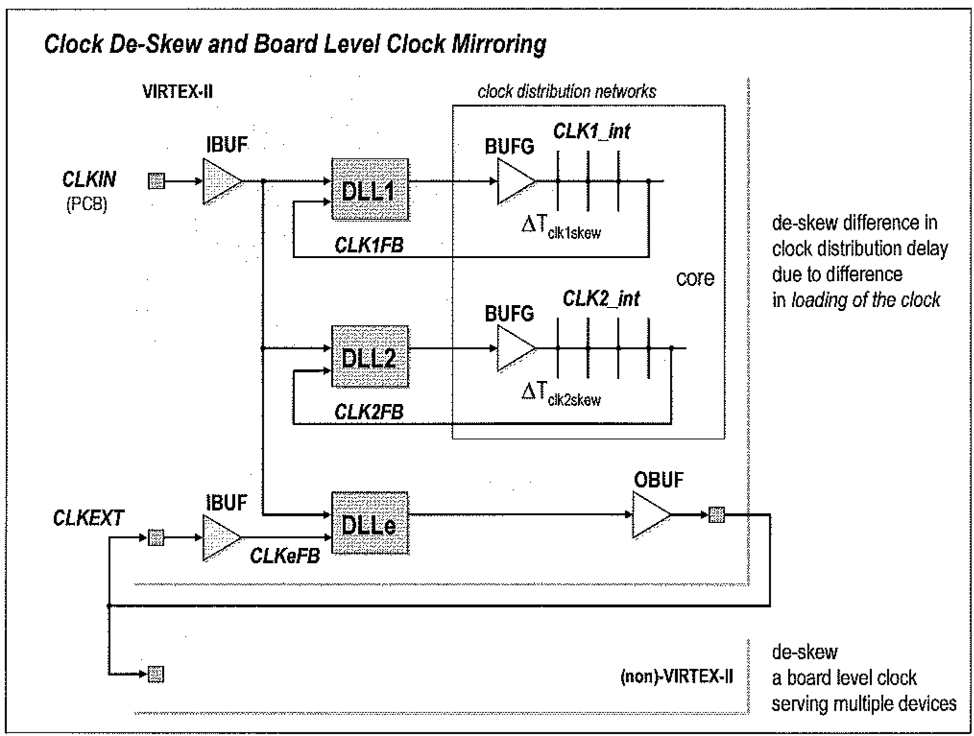
The DCM de-skews the output clocks relative to the input clock by automatically adjusting a digital delay line. Additional delay is introduced so that clock edges arrive at internal registers and block RAMs simultaneously with the clock edges arriving at the input clock pad. Alternatively, external clocks, which are also de-skewed relative to the input clock, can be generated for board-level routing. All DCM output clocks are phase-aligned to CLK0 and, therefore, are also phase-aligned to the input clock.

To achieve clock de-skew, the CLKFB input must be connected, and its source must be either CLK0 or CLK2X. Note that CLKFB must always be connected, unless only the CLKFX or CLKFX180 outputs are used and de-skew is not required.

The DCM contains a digitally-controlled feedback circuit (delay locked loop) that can completely eliminate clock distribution delays. Clock de-skew works as follows:

The incoming clock drives a long chain of delay elements (individual small buffers). A wide multiplexer selects any one of these buffers as an output. A controller drives the select inputs of this multiplexer. The phase detector in this controller compares the incoming clock signal (CLKIN) against a feedback input (CLKFB), which must be another version of the same clock signal, usually from the far end of the internal clock distribution network (but it can also be from an output pin).

The phase detector steers the controller to adjust the tap selection, and thus the through-delay in the DCM, in such a way that the two inputs to the phase comparator coincide.(This is a typical servo loop.) The tap controller adds exactly the right amount of delay to the clock distribution network to give it a total delay of one full clock period. For a repetitive clock signal, this effectively eliminates the clock distribution delay completely.

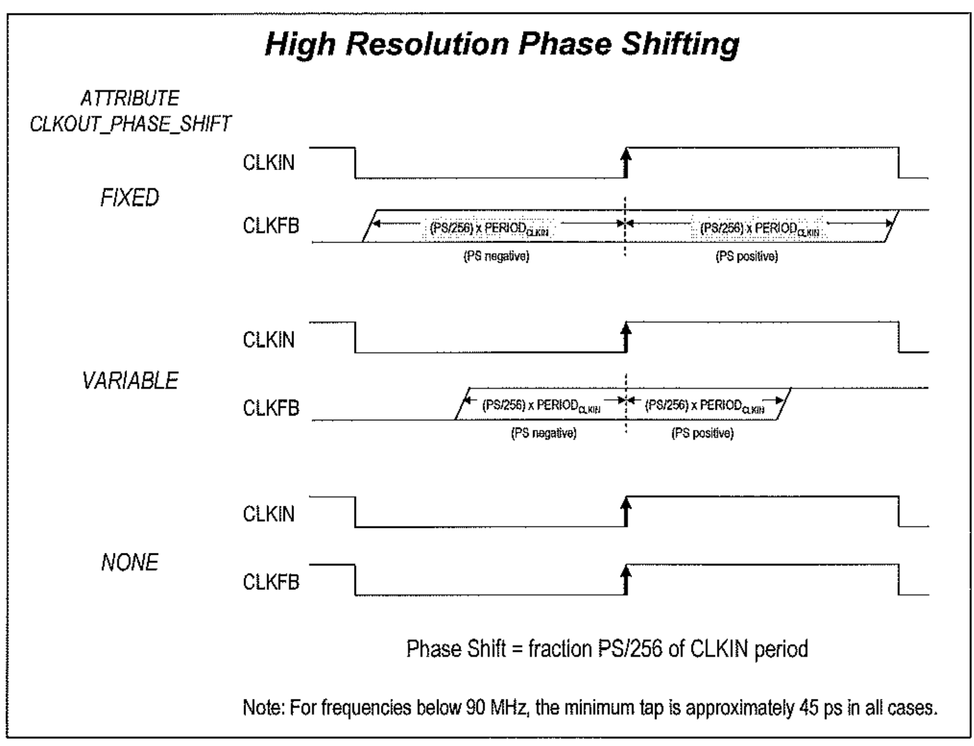


The DLL can also act as a clock mirror. By driving the DLL output off-chip and then back in again, the DLL can be used to de-skew a board level clock between multiple devices.

The clock mirroring scheme should be used for systems (such as PCI) that specify a loading/fan out limit on the system clock. This scheme can also be used when the clock source in a single board system is not capable of driving all the loads on the board.

While designing the board level route, ensure that the return net delay to the source equals the delay to the other chips involved.

Do not use the DLL output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.



Phase Shifting

The DCM provides additional control over clock skew through either coarse or fine-grained phase shifting.

The CLK0, CLK90, CLK180, and CLK270 outputs are each phase shifted by 1/4 of the input clock period relative to each other, providing coarse phase control. Note that CLK90 and CLK270 are not available in high-frequency mode.

Fine-phase adjustment affects all nine DCM output clocks. When activated, the phase shift between the rising edges of CLKIN and CLKFB is a fixed or variable fraction of the input clock period.

In variable mode, the PHASE_SHIFT value can also be dynamically incremented or decremented as determined by PSINCDEC synchronously to PSCLK, when the PSEN input is active. The PHASE_SHIFT attribute is the numerator in the following equation:

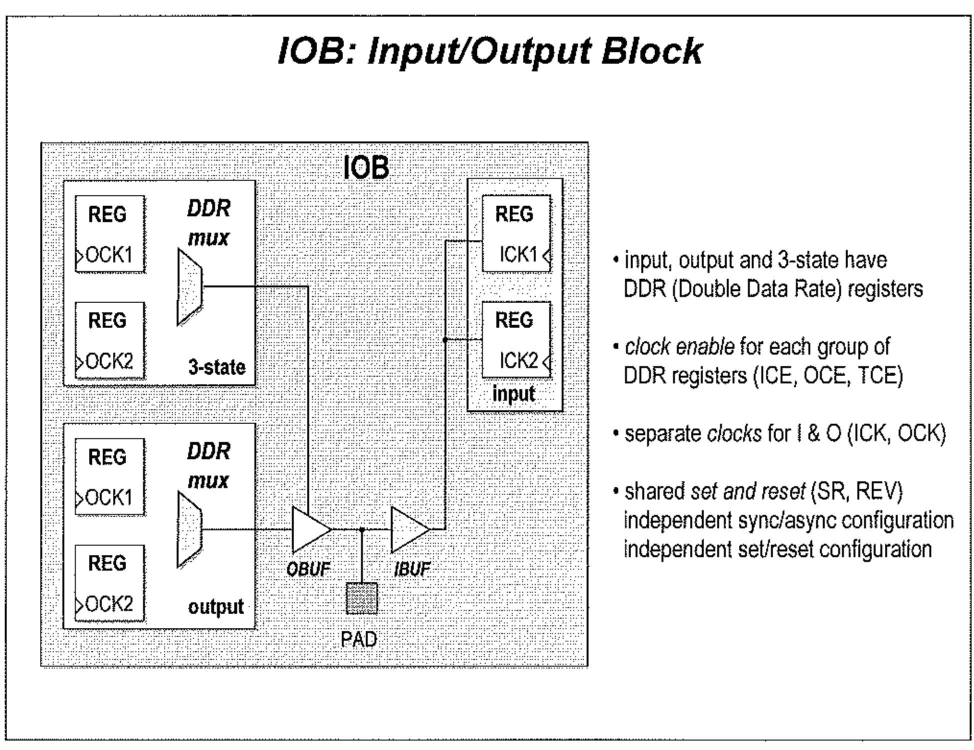
Phase Shift (ns) = (PHASE_SHIFT/256) * PERIODCLKIN

The full range of this attribute is always -255 to +255, but its practical range varies with CLKIN frequency, as constrained by the total delay achievable by the phase shift delay line (FINE_SHIFT_RANGE). Total delay is a function of the number of delay taps used in the circuit, the process, voltage, and temperature.

Absolute range (fixed mode) = ± FINE_SHIFT_RANGE

Absolute range (variable mode) = ± FINE_SHIFT_RANGE/2

The reason for the difference between fixed and variable modes is as follows. For variable mode to allow symmetric, dynamic sweeps from -255/256 to +255/256, the DCM sets the "zero phase skew" point as the middle of the delay line, thus dividing the total delay line range in half. In fixed mode, since the PHASE_SHIFT value never changes after configuration, the entire delay line is available for insertion into either the CLKIN or CLKFB path (to create either positive or negative skew).



IOBs are programmable and can be categorized as follows:

• Input block with an optional single-data-rate or double-data-rate (DDR) register

• Output block with an optional single-data-rate or DDR register, and an optional 3-state buffer, to be driven directly or through a single or DDR register

• Bidirectional block (any combination of input and output configurations)

These registers are either edge-triggered D-type flip-flops or level-sensitive latches.

IOBs support the following single-ended I/O standards: LVTTL, LVCMOS (3.3V, 2.5V, 1.8V, and 1.5V), PCI-X compatible (133 MHz and 66 MHz) at 3.3V, PCI compliant (66 MHz and 33 MHz) at 3.3V, ...

The digitally controlled impedance (DCI) I/O feature automatically provides on-chip termination for each I/O element.

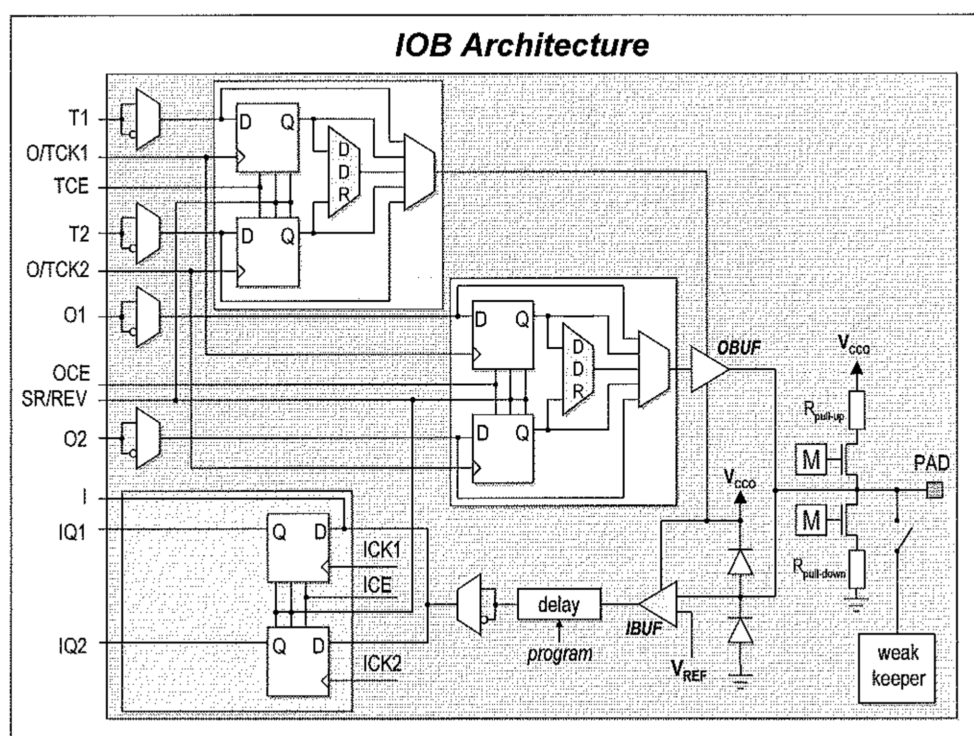
The IOB elements also support the following differential signaling I/O standards: LVDS, BLVDS (Bus LVDS), ULVDS, LDT, LVPECL.

Virtex-II I/O blocks (IOBs) are provided in groups of two or four on the perimeter of each device. Each IOB can be used as input and/or output for single-ended I/Os. Two adjacent IOBs can be used as a differential pair. A differential pair is always connected to the same switch matrix to access the routing resources

Logic Resources

IOB blocks include six storage elements. Each storage element can be configured either as an edge-triggered D-type flip-flop or as a level-sensitive latch.

On the input, output, and 3-state path, one or two DDR registers can be used.



Each group of two registers has a clock enable signal (ICE for the input registers, OCE for the output registers, and TCE for the 3-state registers). The clock enable signals are active High by default. If left unconnected, the clock enable for that storage element defaults to the active state.

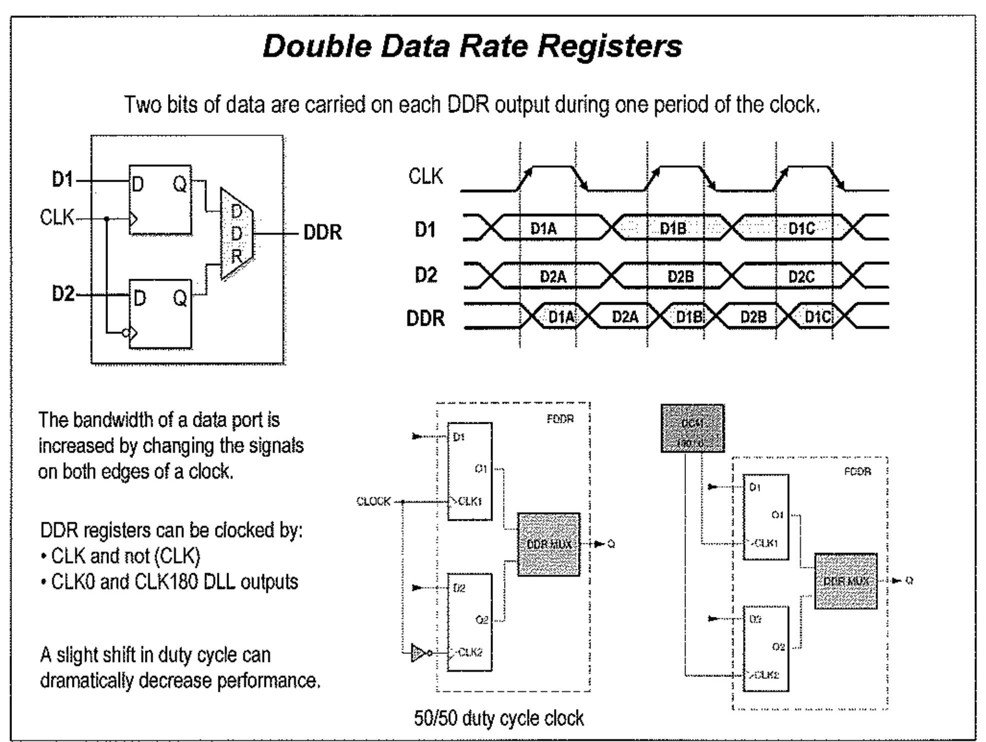
Each IOB block has common synchronous or asynchronous set and reset (SR and REV signals).

SR forces the storage element into the state specified by the SRHIGH or SRLOW attribute. SRHIGH forces a logic "1". SRLOW forces a logic "0". When SR is used, a second input (REV) forces the storage element into the opposite state. The reset condition predominates over the set condition. The initial state after configuration or global initialization state is defined by a separate INIT0 and INIT1 attribute. By default, the SRLOW attribute forces INIT0, and the SRHIGH attribute forces INIT1. For each storage element, the SRHIGH, SRLOW, INIT0, and INIT1 attributes are independent. Synchronous or asynchronous set / reset is consistent in an IOB block. All the control signals have independent polarity. Any inverter placed on a control input is automatically absorbed.

Each device pad has optional pull-up and pull-down in all SelectI/O-Ultra configurations. Values of the optional pull-up and pull-down resistors are in the range 10 - 60 KΩ,. Each device pad has an optional weak-keeper in LVTTL, LVCMOS, and PCI SelectI/O-Ultra configurations. The clamp diode is always present, even when power is not.

The optional weak-keeper circuit is connected to each user I/O pad. When selected, the circuit monitors the voltage on the pad and weakly drives the pin High or Low. If the pin is connected to a multiple-source signal, the weak-keeper holds the signal in its last state if all drivers are disabled. Maintaining a valid logic level in this way eliminates bus chatter; pull-up or pull-down override the weak-keeper circuit.

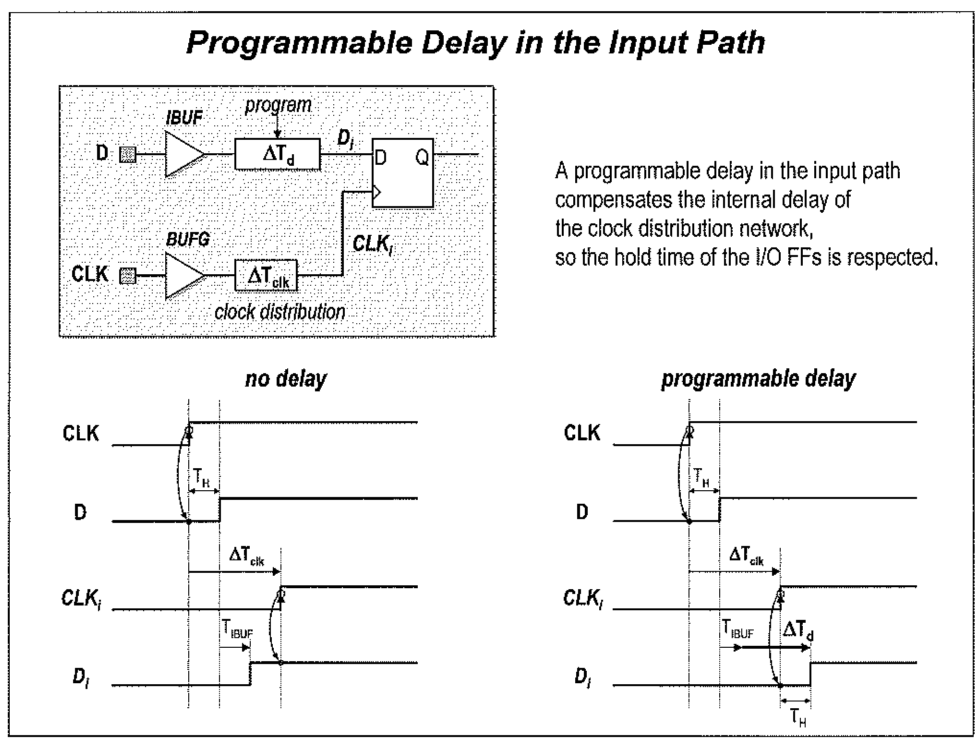
All pads are protected against damage from electrostatic discharge (ESD) and from over-voltage transients.



Double data rate is directly accomplished by the two registers on each path, clocked by the rising edges (or falling edges) from two different clock nets. The two clock signals are generated by the DCM and must be 180 degrees out of phase. A single clock triggers one register on a Low to High transition and a second register on a High to Low transition.

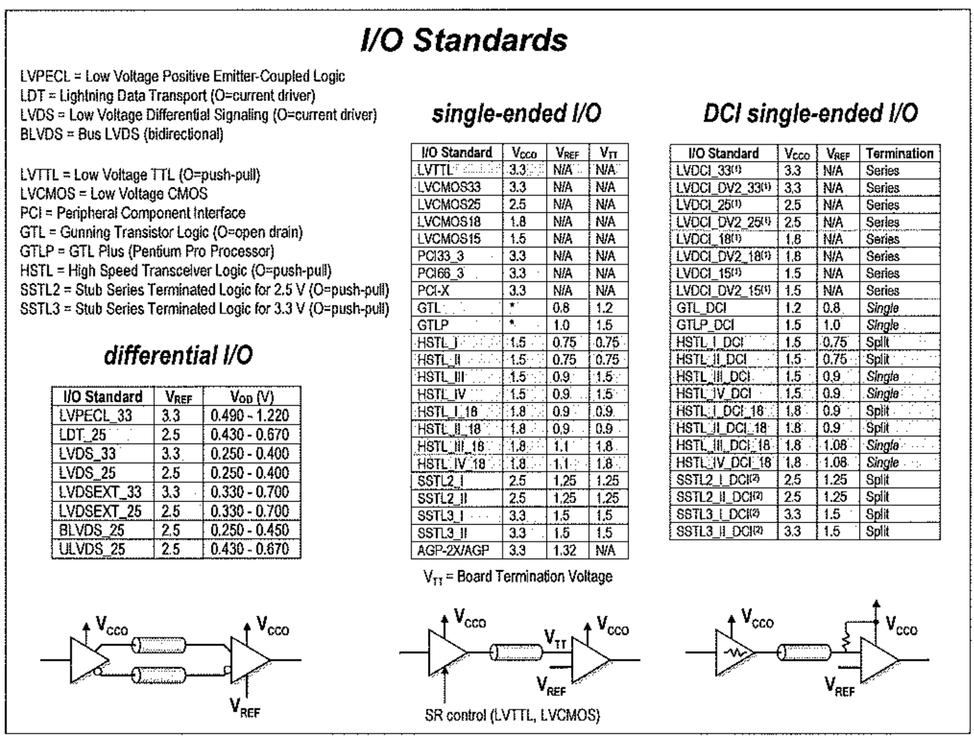
There are two input, output, and 3-state data signals, each being alternately clocked out.

The DDR mechanism can be used to mirror a copy of the clock on the output (like Manchester coding). This is useful for propagating a clock along the data that has an identical delay. It is also useful for multiple clock generation, where there is a unique clock driver for every clock load. Virtex-II devices can produce many copies of a clock with very little skew.



A programmable delay in the input path compensates the internal delay of the clock distribution network, so the hold time of the I/O FFs is respected.

This optional delay element at the D-input of the storage element eliminates pad-to-pad hold time. The delay is matched to the internal clock-distribution delay of the Virtex-II device, and when used, assures that the pad-to-pad hold time is zero.



I/O Standards

LVPECL = Low Voltage Positive Emitter-Coupled Logic
LDT = Lightning Data Transport (O=current driver)
LVDS = Low Voltage Differential Signaling (O=current driver)
BLVDS = Bus LVDS (bidirectional)

LVTTL = Low Voltage TTL (O=push-pull)
LVCMOS = Low Voltage CMOS
PCI = Peripheral Component Interface
GTL = Gunning Transistor Logic (O=open drain)
GTLP = GTL Plus (Pentium Pro Processor)
HSTL = High Speed Transceiver Logic (O=push-pull)
SSTL2 = Stub Series Terminated Logic for 2.5 V (O=push-pull)
SSTL3 = Stub Series Terminated Logic for 3.3 V (O=push-pull)

differential I/O

I/O Standard	V_{REF}	V_{OD} (V)
LVPECL_33	3.3	0.490 - 1.220
LDT_25	2.5	0.430 - 0.670
LVDS_33	3.3	0.250 - 0.400
LVDS_25	2.5	0.250 - 0.400
LVDSEXT_33	3.3	0.330 - 0.700
LVDSEXT_25	2.5	0.330 - 0.700
BLVDS_25	2.5	0.250 - 0.450
ULVDS_25	2.5	0.430 - 0.670

single-ended I/O

I/O Standard	V_{CCO}	V_{REF}	V_{TT}
LVTTL	3.3	N/A	N/A
LVCMOS33	3.3	N/A	N/A
LVCMOS25	2.5	N/A	N/A
LVCMOS18	1.8	N/A	N/A
LVCMOS15	1.5	N/A	N/A
PCI33_3	3.3	N/A	N/A
PCI66_3	3.3	N/A	N/A
PCI-X	3.3	N/A	N/A
GTL	*	0.8	1.2
GTLP	*	1.0	1.5
HSTL_I	1.5	0.75	0.75
HSTL_II	1.5	0.75	0.75
HSTL_III	1.5	0.9	1.5
HSTL_IV	1.5	0.9	1.5
HSTL_I_18	1.8	0.9	0.9
HSTL_II_18	1.8	0.9	0.9
HSTL_III_18	1.8	1.1	1.8
HSTL_IV_18	1.8	1.1	1.8
SSTL2_I	2.5	1.25	1.25
SSTL2_II	2.5	1.25	1.25
SSTL3_I	3.3	1.5	1.5
SSTL3_II	3.3	1.5	1.5
AGP-2X/AGP	3.3	1.32	N/A

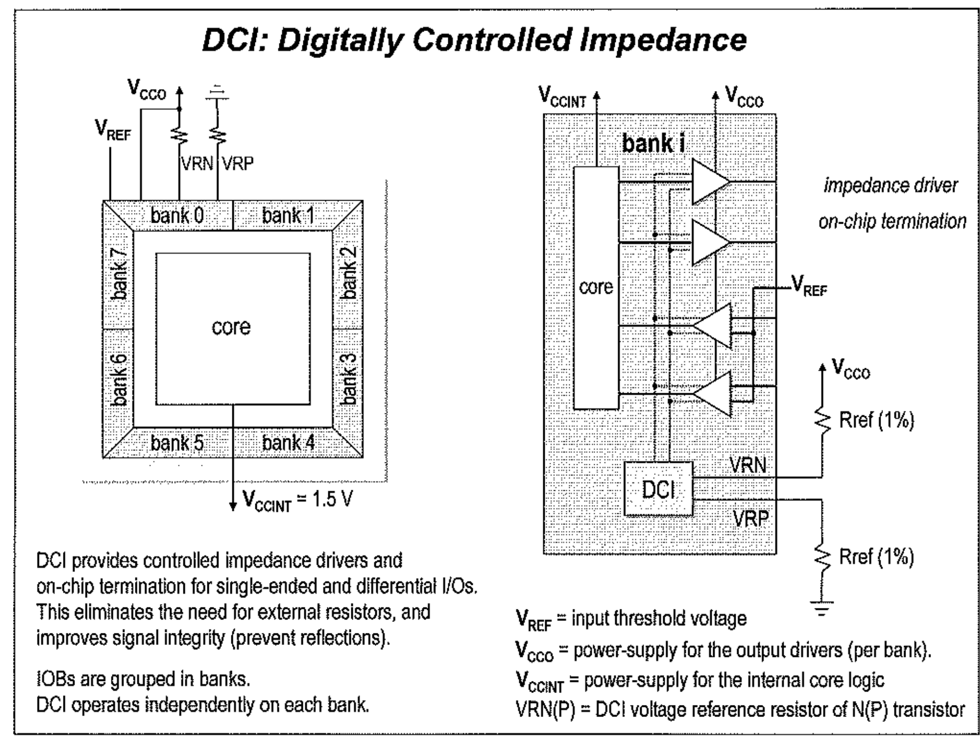
V_{TT} = Board Termination Voltage

DCI single-ended I/O

I/O Standard	V_{CCO}	V_{REF}	Termination
LVDCI_33(1)	3.3	N/A	Series
LVDCI_DV2_33(1)	3.3	N/A	Series
LVDCI_25(1)	2.5	N/A	Series
LVDCI_DV2_25(1)	2.5	N/A	Series
LVDCI_18(1)	1.8	N/A	Series
LVDCI_DV2_18(1)	1.8	N/A	Series
LVDCI_15(1)	1.5	N/A	Series
LVDCI_DV2_15(1)	1.5	N/A	Series
GTL_DCI	1.2	0.8	*Single*
GTLP_DCI	1.5	1.0	*Single*
HSTL_I_DCI	1.5	0.75	Split
HSTL_II_DCI	1.5	0.75	Split
HSTL_III_DCI	1.5	0.9	*Single*
HSTL_IV_DCI	1.5	0.9	*Single*
HSTL_I_DCI_18	1.8	0.9	Split
HSTL_II_DCI_18	1.8	0.9	Split
HSTL_III_DCI_18	1.8	1.08	*Single*
HSTL_IV_DCI_18	1.8	1.08	*Single*
SSTL2_I_DCI(2)	2.5	1.25	Split
SSTL2_II_DCI(2)	2.5	1.25	Split
SSTL3_I_DCI(2)	3.3	1.5	Split
SSTL3_II_DCI(2)	3.3	1.5	Split

IOB blocks are designed for high performances I/Os, supporting 19 single-ended standards, as well as differential signaling with LVDS, LDT, Bus LVDS, and LVPECL.

Virtex-II IOB blocks feature SelectI/O-Ultra inputs and outputs that support a wide variety of I/O signaling standards. In addition to the internal core supply voltage (V_{CCINT} = 1.5V), output driver supply voltage (V_{CCO}) is dependent on the I/O standard (see Table). An auxiliary supply voltage (V_{CCAUX} = 3.3 V) is required, regardless of the I/O standard used.



Today's chip output signals with fast edge rates require termination to prevent reflections and maintain signal integrity. High pin count packages (especially ball grid arrays) can not accommodate external termination resistors.

Virtex-II XCITE DCI provides controlled impedance drivers and on-chip termination for single-ended and differential I/Os. This eliminates the need for external resistors, and improves signal integrity. The DCI feature can be used on any IOB by selecting one of the DCI I/O standards.

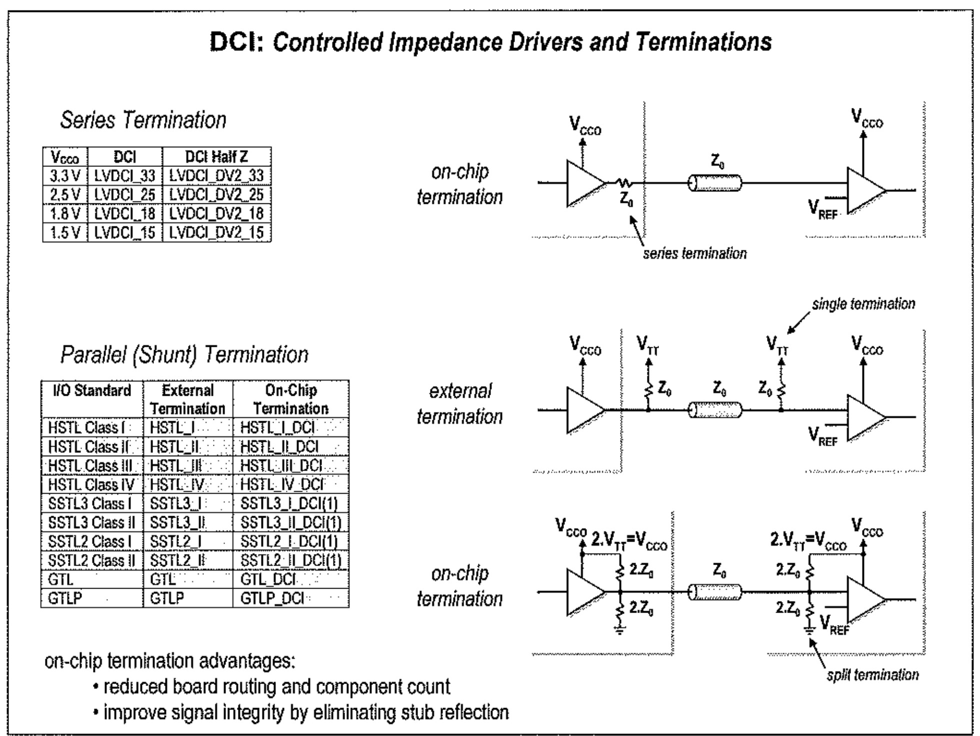
When applied to inputs, DCI provides input parallel termination. When applied to outputs, DCI provides controlled impedance drivers (series termination) or output parallel termination.

Some of the I/O standards require V_{CCO} and V_{REF} voltages. These voltages are externally supplied and connected to device pins that serve groups of IOB blocks, called banks. Consequently, restrictions exist about which I/O standards can be combined within a given bank. Eight I/O banks result from dividing each edge of the FPGA into two banks. Each bank has multiple V_{CCO} pins, all of which must be connected to the same voltage. This voltage is determined by the output standards in use.

DCI operates independently on each I/O bank. When a DCI I/O standard is used in a particular I/O bank, external reference resistors must be connected to two dual-function pins on the bank. These resistors, voltage reference of N transistor (VRN) and the voltage reference of P transistor (VRP) are shown in the figure.

When used with a terminated I/O standard, the value of resistors are specified by the standard (typically 50 Ω). When used with a controlled impedance driver, the resistors set the output impedance of the driver within the specified range (25 Ω to 100 Ω). For all series and parallel terminations, the reference resistors must have the same value for any given bank. One percent resistors are recommended.

The DCI system adjusts the I/O impedance to match the two external reference resistors, or half of the reference resistors, and compensates for impedance changes due to voltage and/or temperature fluctuations. The adjustment is done by turning parallel transistors in the IOB on or off.



DCI: *Controlled Impedance Drivers and Terminations*

Series Termination

V_{CCO}	DCI	DCI Half Z
3.3 V	LVDCI_33	LVDCI_DV2_33
2.5 V	LVDCI_25	LVDCI_DV2_25
1.8 V	LVDCI_18	LVDCI_DV2_18
1.5 V	LVDCI_15	LVDCI_DV2_15

Parallel (Shunt) Termination

I/O Standard	External Termination	On-Chip Termination
HSTL Class I	HSTL_I	HSTL_I_DCI
HSTL Class II	HSTL_II	HSTL_II_DCI
HSTL Class III	HSTL_III	HSTL_III_DCI
HSTL Class IV	HSTL_IV	HSTL_IV_DCI
SSTL3 Class I	SSTL3_I	SSTL3_I_DCI(1)
SSTL3 Class II	SSTL3_II	SSTL3_II_DCI(1)
SSTL2 Class I	SSTL2_I	SSTL2_I_DCI(1)
SSTL2 Class II	SSTL2_II	SSTL2_II_DCI(1)
GTL	GTL	GTL_DCI
GTLP	GTLP	GTLP_DCI

Output Driver

DCI can be used to provide a buffer with a controlled output impedance. It is desirable for this output impedance to match the transmission line impedance (Z).

For controlled impedance output drivers, the impedance can be adjusted either to match the reference resistors or half the resistance of the reference resistors.

DCI can configure output drivers to be the following types:

1. Controlled Impedance Driver (Source Termination)

2. Controlled Impedance Driver with Half Impedance (Source Termination)

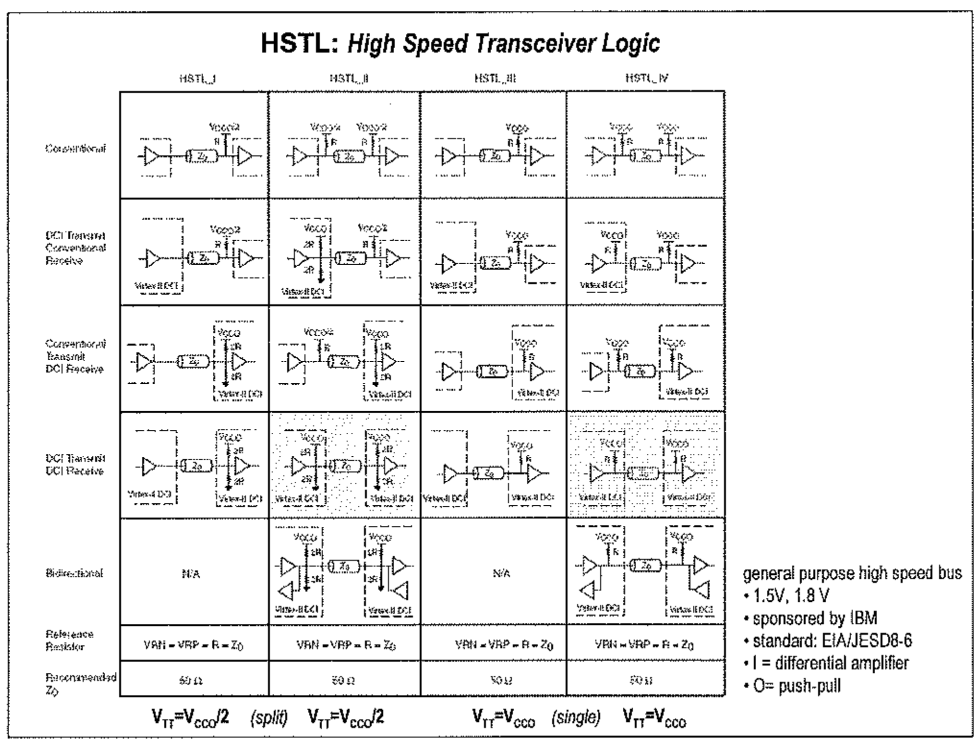
Inputs

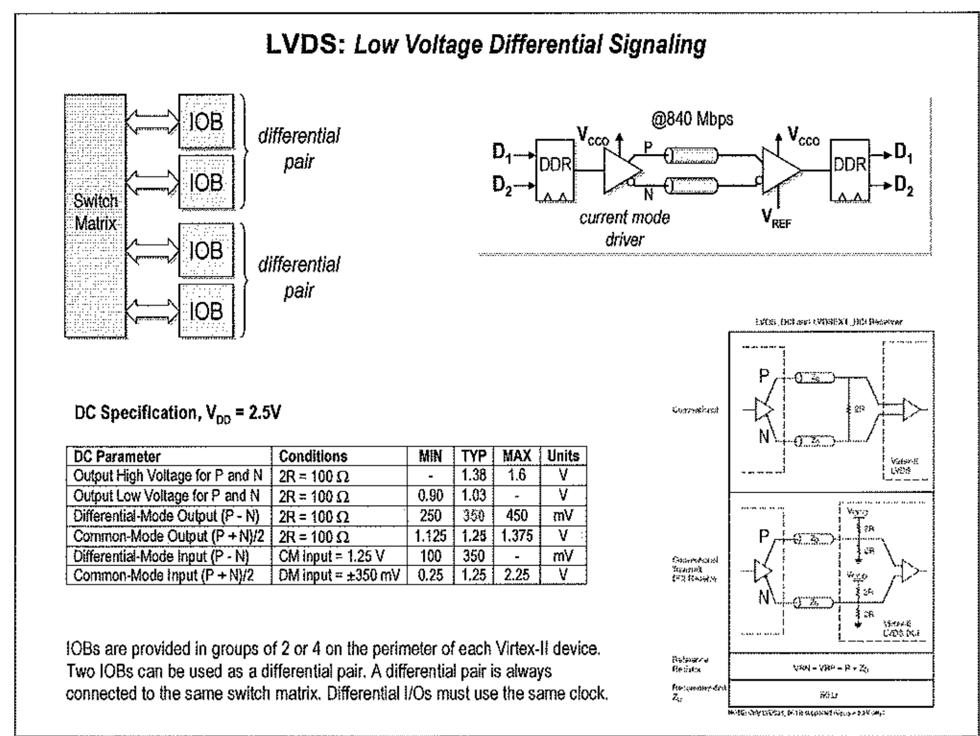
For on-chip termination, the termination is always adjusted to match the reference resistors.

It can also configure inputs to have he following types of on-chip terminations:

1. Input termination to V_{CCO} (Single Termination)

2. Input termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)


HSTL: High Speed Transceiver Logic
HSTL_I
HSTL_II
HSTL_III
HSTL_IV
Conventional
DCI Transmit Conventional Receive
Conventional Transmit DCI Receive
DCI Transmit DCI Receive
Bidirectional
N/A
Reference Resistor
VRN = VRP = R = Z0
Recommended Z0
50 Ω
general purpose high speed bus
• 1.5V, 1.8 V
• sponsored by IBM
• standard: EIA/JESD8-6
• I = differential amplifier
• O= push-pull
VTT=VCCO/2 (split) VTT=VCCO/2
VTT=VCCO (single) VTT=VCCO



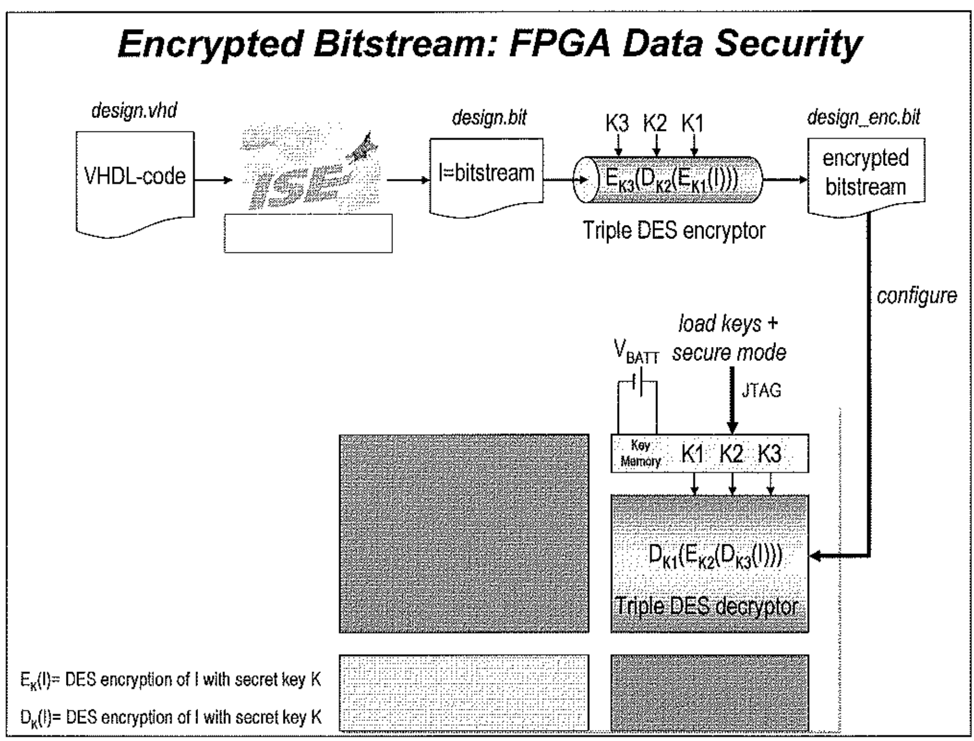
DC Parameter	Conditions	MIN	TYP	MAX	Units
Output High Voltage for P and N	2R = 100 Ω	-	1.38	1.6	V
Output Low Voltage for P and N	2R = 100 Ω	0.90	1.03	-	V
Differential-Mode Output (P - N)	2R = 100 Ω	250	350	450	mV
Common-Mode Output (P + N)/2	2R = 100 Ω	1.125	1.25	1.375	V
Differential-Mode Input (P - N)	CM input = 1.25 V	100	350	-	mV
Common-Mode Input (P + N)/2	DM input = ±350 mV	0.25	1.25	2.25	V

LVDS provides higher noise immunity than single-ended techniques, allowing for higher transmission speeds, smaller signal swings, lower power consumption, and less electro-magnetic interference than single-ended signaling. Differential data can be transmitted at these rates using inexpensive connectors and cables. LVDS provides robust signaling for high-speed data transmission between chassis, boards, and peripherals using standard ribbon cables and IDC connectors with 100 mil header pins. Point-to-point LVDS signaling is possible at speeds of up to 622 Mb/s.

An LVDS driver on the left drives the two 50 Ω transmission lines into an LVDS receiver on the right. The P and N outputs of the LVDS driver pass to the corresponding inputs of the LVDS receiver. The two 50 Ω single-ended transmission lines can be microstrip, stripline, or a 100 Ω differential twisted pair or similar balanced differential transmission line. A 100 Ω resistor R_T terminates the P and N signals.

LVDS uses a current-mode driver, behaving like two equal and opposite current sources with a high output impedance. LVDS outputs typically drive ±3.5 mA to flow through the 100 Ω resistor R_T generating a ±350 mV voltage swing (P - N). This results in ≈1.2 mW of power delivered to the load.

The terms "common-mode voltage" and "offset voltage" refer to the average of P and N, (P+N)/2. LVDS has a typical output common-mode voltage of 1.25 V, determined by the LVDS driver.



Virtex-II devices have an on-chip decryptor using one or two sets of three keys for triple-key Data Encryption Standard (DES) operation. Xilinx software tools offer an optional encryption of the configuration data (bitstream) with a triple-key DES determined by the designer.

The keys are stored in the FPGA by JTAG instruction and retained by a battery connected to the V_{BATT} pin, when the device is not powered. Virtex-II devices can be configured with the corresponding encrypted bitstream, using any of the configuration modes described previously.

Virtex-II solves design security issue for FPGAs

- Bitstream secured using triple DES (3x56 bit keys)
- Prevents SRAM FPGA design theft
- Enables protected customer-specific chip-sets

2 banks of 3 DES keys, allows 2 users with different keys

All configuration methods (Serial, JTAG) support encryption at full speed (50MHz @ 8 bits wide)

PROCEDURE

1. Encrypt your bitstream with triple DES using 56 bit keys
2. Load 1 or 2 banks of keys into SRAM registers using JTAG
3. (Optional) Read back values of keys to ensure they are correct
4. Enter secure mode in FPGA by issuing JTAG instruction
   - Allows one encrypted bitstream write, locks out all subsequent bitstream reads and writes
   - Prevents reading or writing of keys
5. Load encrypted bitstream into FPGA specifying which triple DES key bank to use for decryption
6. Operate chip as normal FPGA (without possibility of reading and writing bitstreams)

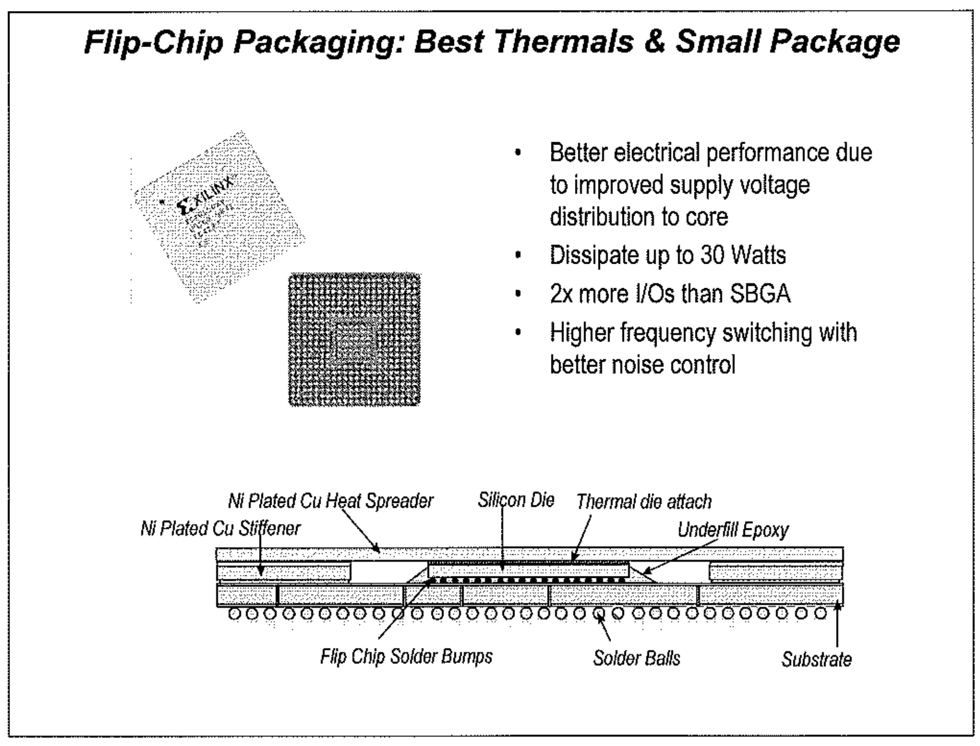
Virtex-II Packaging

- Wire-bond packages
- Flip-chip packages
  - Higher device I/O count
  - Higher thermal capacity
- Ball-grid arrays:
  - FGxxx: wire-bond fine-pitch BGA (1.00 mm pitch)
  - BGxxx: wire-bond BGA (1.27 mm pitch)
  - FFxxx: flip-chip fine-pitch BGA (1.00 mm pitch)
  - BFxxx: flip-chip BGA (1.27 mm pitch)

Offerings include ball grid array (BGA) packages with 0.80 mm, 1.00 mm, and 1.27 mm pitches. In addition to traditional wire-bond interconnects, flip-chip interconnect is used in some of the BGA offerings. The use of flip-chip interconnect offers more I/Os than is possible in wire-bond versions of the similar packages. Flip-chip construction offers the combination of high pin count with high thermal capacity.

Flip-Chip and Wire-Bond Ball Grid Array (BGA) Packages in Three Standard Fine Pitches (0.80 mm, 1.00 mm, and 1.27 mm):

- CS denotes wire-bond chip-scale ball grid array (BGA) (0.80 mm pitch).
- FG denotes wire-bond fine-pitch BGA (1.00 mm pitch).
- FF denotes flip-chip fine-pitch BGA (1.00 mm pitch).
- BG denotes standard BGA (1.27 mm pitch).
- BF denotes flip-chip BGA (1.27 mm pitch).



Flip-chip construction offers the combination of high pin count with high thermal capacity.

Virtex-II Packaging: Overview

Device XC2V	40	80	250	500	1000	1500	2000	3000	4000	6000	8000
Max user I/Os	88	120	200	264	432	528	624	720	912	1,104	(1296) 1108
CS144	88	92	92								
FG256	88	120	172	172	172						
FG456			200	264	324						
FG676						392	456	484			
FF896					432	528	624				
FF1152								720	824	824	824
FF1517									912	1,104	1,108
BG575					328	392	408				
BG728							456	516			
BF957							624	684	684	684	684

- **FF** and **BF** are flip-chip ball grid arrays packages
- Pinout compatibility inside same color rectangle

Wire-bond and flip-chip packages are available. The table shows the maximum possible number of user I/Os for all device/package (wire-bond and flip-chip) combinations.

The number of I/Os per package include all user I/Os except the 15 control pins (CCLK, DONE, M0, M1, M2, PROG_B, PWRDWN_B, TCK, TDI, TDO, TMS, HSWAP_EN, DXN, DXP, and RSVD) and VBATT.

- CS denotes wire-bond chip-scale ball grid array (BGA) (0.80 mm pitch).
- FG denotes wire-bond fine-pitch BGA (1.00 mm pitch).
- FF denotes flip-chip fine-pitch BGA (1.00 mm pitch).
- BG denotes standard BGA (1.27 mm pitch).
- BF denotes flip-chip BGA (1.27 mm pitch).