**Delay Faults**

Delays along every path from PI to PO or between internal latches must be *less than* the operational system clock interval.

We have already discussed a number of defects that can cause delay faults:
- GOS defects
- Resistive shorting defects between nodes and to the supply rails
- Parasitic transistor leakages, defective pn junctions and incorrect or shifted threshold voltages
- Certain types of opens
- Process variations can also cause devices to switch at a speed lower than the specification.

An *SA0* or *SA1* can be modeled as a delay fault in which the signal takes an *infinite* amount of time to change to 1 or 0, respectively.

Passing stuck fault tests is usually not sufficient however for systems that operate at any appreciable speed.

Running stuck-at fault tests at higher speed can uncover some delay faults.
Delay Tests

- Delay tests consist of vector-pairs.
- All input transitions occur at the same time.
- The longest delay combinational path is referred to as the critical path, which determines the shortest clock period.
  
  A delay fault means that the delay of one or more paths (not necessarily the critical path) exceeds the clock period.

Test Definition:

- At time $t_1$, the initializing vector of the two-pattern test, $V_1$, is applied through the input latches or PIs and the circuit is allowed to stabilize.
- At time $t_2$, the second test pattern, $V_2$, is applied.
- At time $t_3$, a logic value measurement (a sample) is made at the output latches or POs.

The delay test vectors $V_1$ and $V_2$ may sensitize one or more paths, $p_i$. 
**Delay Tests**

Let:

- \( T_C = (t_3 - t_2) \) represent the time interval between the application of vector \( V_2 \) at the PIs and the sampling event at the POs.
- The *nominal delay* of each of these paths be defined as \( p_{d_i} \).
- The *slack* of each path be defined as \( s_{d_i} = T_C - p_{d_i} \).

This is the difference between the propagation delay of each of the sensitized paths in the nominal circuit and the test interval.

![Diagram of delay in combinational logic](image)

- **From FFs or PIs**
- **Combinational Logic**
- **To FFs or POs**

Delay of combination logic cannot exceed the clock period.
Delay Fault Test Generation

Difficulties with delay fault test generation:

- Test generation requires a sensitized path that extends from a PI to a PO.

- Path selection heuristics must be used because the total number of paths is exponentially related to the number of inputs and gates in the circuit.

- The application of the test set must be performed at the rated speed of the device. This requires test equipment that is capable of accurately timing two-vector test sequences.

- The detection of a defect that introduces an additional delay, \( ad_i \), along a sensitized path is dependent on satisfying the condition:
  \[ ad_i > sd_i \text{ (or } pd_i + ad_i > T_C) \]
A path sensitized by a delay test consists of \textit{on-path} nodes and \textit{off-path} nodes. The nodes along the \textit{sensitized path} are referred to as \textit{on-path} nodes.

\textbf{Static sensitization} defines the case when all \textit{off-path} nodes settle to non-controlling values (0 for OR/NOR, 1 for AND/NAND) following application of \( V_2 \).

This is a necessary condition to test a path for a delay fault.

The gates along the sensitized path have exactly \textit{one} on-path input and zero or more non-controlling off-path inputs.

Delay fault tests are classified according to the voltage behavior of the \textit{off-path} nodes. Such tests can be \textit{invalidated} under certain conditions.

Hazards can invalidate tests:

\begin{itemize}
  \item \textbf{Static hazard}: describes a circuit condition where \textit{off-path} nodes change momentarily when they are supposed to remain constant.
  \item \textbf{Dynamic hazard}: describes a circuit condition where \textit{off-path} nodes make several transitions when they are supposed to make a single transition.
\end{itemize}
Hazards

Static Hazards:

\[ F = (A \cdot B) + (C \cdot \bar{B}) \]

Two vector sequence is \( ABC = (111), (101) \).

Gate \( G_1 \) introduces an additional delay of 1 unit.

Output \( E \) of gate \( G_3 \) is driven to a logic 1, one time unit behind \( D \rightarrow 0 \).

Produces a glitch on \( F \).
Hazard

Dynamic Hazards:

\[ F = (A \cdot \bar{B}) + (\bar{A} \cdot B) \]

Two vector sequence is \( AB = (01), (11) \).
Gate \( G_2 \) has a delay value of 3 time units.

Tested path

Glitch on output F

Time line starting when vector \( AB = (11) \) is applied.
Hazard and Invalidation

*Static hazards* can create *dynamic hazards* along tested paths and need to be considered during test generation.

Note, unlike the previous example, the glitch occurs *before* the intended transition, and can invalidate the test (e.g. fault is not detected).
Delay Tests and Invalidation

The critical path(s) of this circuit is 6 time units.
Let's set the clock period \( T = 7 \).

Assume only one faulty path.
No delay fault is detected if path delay along P3 is less than 7 units.
This test will *not* detect single delay faults along paths P1 or P2.

Assume there can be *multiple faulty* paths.
Assume P2 and P3 are faulty and P2 extends the *static glitch* at the output beyond 7 units, then it masks P3's delay fault.
This test is called a *non-robust* test for delay fault P3.
Delay Fault Path Classification

Most paths in a circuit can be classified as:

- Hazard-free robust testable
- Robust testable
- Non-robust testable

Hazard-free robust test

Off-path inputs do not change.

Off-path inputs are stable and hazard-free throughout the test interval, $T_C$. This is the most desirable test since invalidation is not possible.
**Robust Test**

Hazard-free robust tests are desirable but it's not possible in many cases to generate them. Transitions that occur at fan-out points often *reconverge* as off-path inputs along the tested path.

However, *robust* tests are still possible even when static hazards are present on the off-path inputs.

Static hazards are *necessary but not sufficient* to make a delay test non-robust.

A delay test is a *robust* test if the *on-path* nodes control the *first* occurrence of a transition through all gates along the tested path.

This ensures that a delay test is not *invalidated* or a path delay fault *masked* by delay characteristics of gates not on the tested path.

A *robust path-delay test* guarantees to produce an incorrect value at the output if the delay of the path exceeds the clock period, irrespective of other path delays.
Robust Test

Robust delay test:

\[ F = (A \cdot \overline{B}) + (\overline{A} \cdot B) \]

This test is robust since \( F \) will not change state until the transition on \( E \) has occurred.

In other words, any assignable delay to \( D \) can never mask a delay fault that may occur on the tested path.

This is true because the on-path node \( E \) holds the **dominant** input value on gate \( G_4 \), and therefore determines the **earliest** transition possible on \( F \).

Therefore, \( D \) is allowed to delay the transition on \( F \) but not speed it up.
**Robust Test**

It is possible that:
- $D$ can cause a *transition* to occur on $F$ after the transition on-path node $E$ has occurred.
- $D$ may *further delay* the transition of $F$ since it too can hold the dominant input value on gate $G_4$.

The former condition is sufficient to cause a glitch on $F$ (as shown).

The latter condition implies that a robust test does not require the sensitized path to *dominate the timing*, or, to be the *last* transition to occur on all gates along the sensitized path.

An *on-input* node will make the transition either:
- From the *dominant* input state of the gate to the *non-dominant* input state.
- From the *non-dominant* input state of the gate to the *dominant* input state.
**Robust Test**

For the first case, the *off-path* inputs of the gate must behave in either one of two ways.

- If the *off-path* input node changes state, then it must make a transition from the *dominant* to the *non-dominant* input state of the gate.
- If it does not change state, then it must remain in steady-state at the *non-dominant* value during the entire test interval.

"don't care" regions are shaded

All must settle to non-controlling values otherwise the path is not sensitized.

When all *off-path* inputs honor these constraints, the outputs of the gates along the test path will *not* make the transition until the *last* of all transitioning input lines have toggled.
Robust Test

For the second case, the *off-path* inputs must remain at their *non-dominant* states during the entire test interval.

No off-path transition is allowed.

In either case, hazards will *not* be visible at the output until after the desired transition has propagated along the tested path.

However, for many circuits, even this weaker set of constraints permits only a small percentage of path delay faults to be robust tested.
**Non-Robust Test**

A *non-robust* test allows the output to change *before* the *on-path* transition propagates along the tested path.

A non-robust test *cannot* guarantee the detection of a delay fault along a test path in the presence of *other faults*.

Non-robust test for path delay fault  \( \uparrow \) A-B-C

**Pulse Generator circuit**

Note that the steady-state output is always 0.

Although the delay fault introduced by the inverter is detected (as shown), a delay fault along A-C may cause the output to remain at 0 or it may push the pulse beyond \( T = 4 \) -- which invalidates it.
Non-Robust Test

A non-robust path delay test guarantees the detection of a path-delay fault only when no other path delay fault is present.

Single fault assumption (similar to the Stuck-At fault model).

Assume X takes 2 additional time units beyond A to switch.

Also assume a slow-to-fall delay fault exists for gate G3, adding an additional time unit.

The fault is called a singly-testable path-delay fault in cases where a test exists.
Path Delay Fault Test Generation
To generate test for falling transition on path P3:

(1) F0

(2) C = U0, e = F0

(3) j = R1

(4) q = U0

(5) h = S0, K = R1

(6) A = S0, 0

V1 = ABC = (01X)
V2 = ABC = (000)

(P1: B-e-g-j-K) (P2: B-e-q-h-K)

P1's falling test is robust, i.e., it cannot be invalidated by P2's delay.

(1) R1

(2) C = S0, e = R1

(3) g = U1, j = U0

(4) q = R1

(5) h = R1

(6) h = R1

(7) j = S0

Conflicting with no alternatives

V1 = ABC = (X00)
V2 = ABC = (110)

P2's rising test is non-robust, delay along P1 can invalidate.
Path Delay Fault Test Generation

Previous procedure terminates showing no robust test is possible, so we are stuck with a non-robust test for the rising transition of P2.

Single input change (SIC): a simpler method of generating non-robust tests.

Use a combinational ATPG algorithm to statically sensitize the entire path for \( V_2 \).

\( V_1 \) is obtained by changing one bit in \( V_2 \) that corresponds to the origin of the path.

Validatable non-robust tests

It is desirable to find as many robust tests as possible.

The presence of robust tests for some paths improves the reliability of non-robust tests for other paths.

For example, there are 6 robustly testable paths in the previous circuit.

With these tests, the rising transition test of P2 is as good as a robust test.
Some paths are not even non-robust testable.

P2 falling has no delay test.