Dependability and Fault Tolerance
Zuverlässigkeit und Fehlertoleranz

Chapter 6: Production Testing for ICs

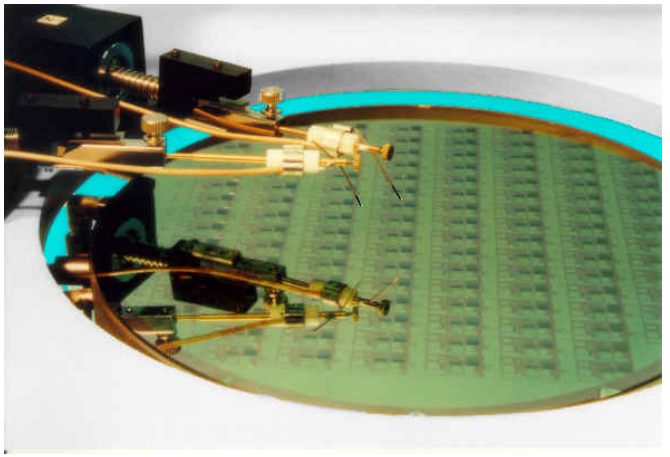
H. T. Vierhaus

WS 2018 / 2019

IC Production Test

- Any type of electronic device may be produced with a defect. Due to irregularities in materials, pressure, temperature in the manufacturing process, dust, dirt, wear-out from tools, etc.
- In most cases, faulty basic devices such as resistors, transistors, ICs have to be found and removed before circuit and / or even system assembly.
- Among electronic devices, integrated circuits are most challenging in terms of test because of their complexity (up to 50 Bio. Transistors) and the limited (virtually impossible) direct access to most of the basic devices such as single transistors in the interior of an IC.
- Internal structures of ICs such as transistors and logic gates suffer from poor „Controllability“ and poor „Observability“ from the outside. This results in a poor level of „Testability“.
- Testing ICs after production is a very complex and expensive process. Test equipment is very expensive, therefore also testing time is costly. For economic reasons, the time allowed to test a single IC is limited to seconds.
- Typically, IC production test must be short, fast and cheap. Typically, it is stopped after the first „fault detect“ without fault diagnosis, the defect device is discarded.
- IC production test is never done in a „real“ operating environment. In most cases, even the mode of operation in production test differs a lot from „real use“. This may result in „test misses“ as well as „overtesting“.

Wafer Test

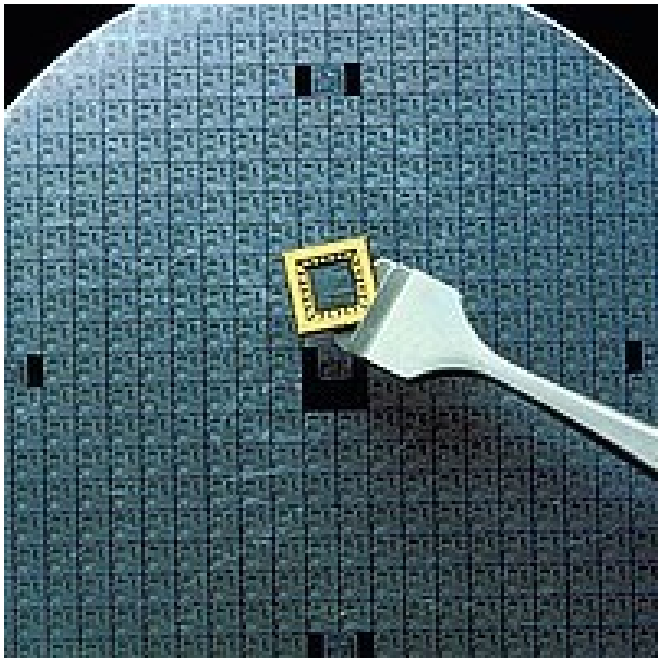


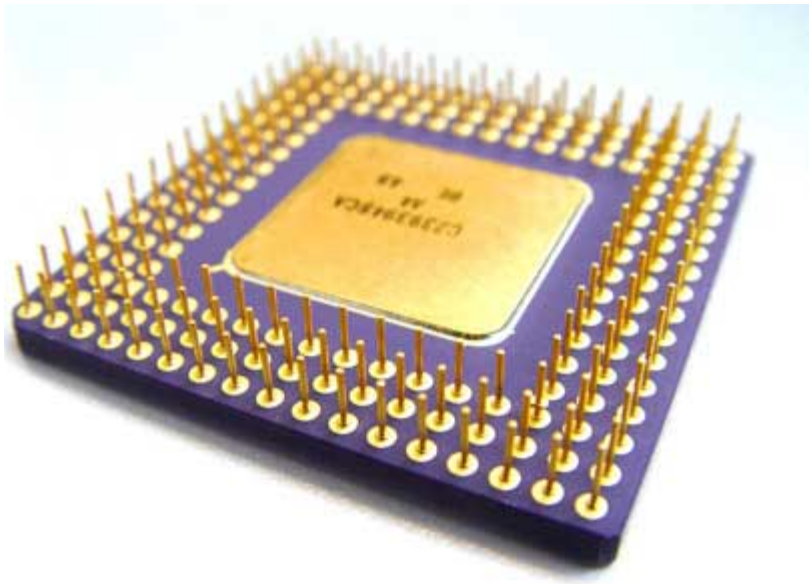
Contact points on the chip (die) , known as „pads“, are contacted by needles.

Typically, the maximum frequence to be transmitted is below 1 GHz.

Location of wafer test: At the site of the wafer manufacturer, before wafer cutting.

Device Test

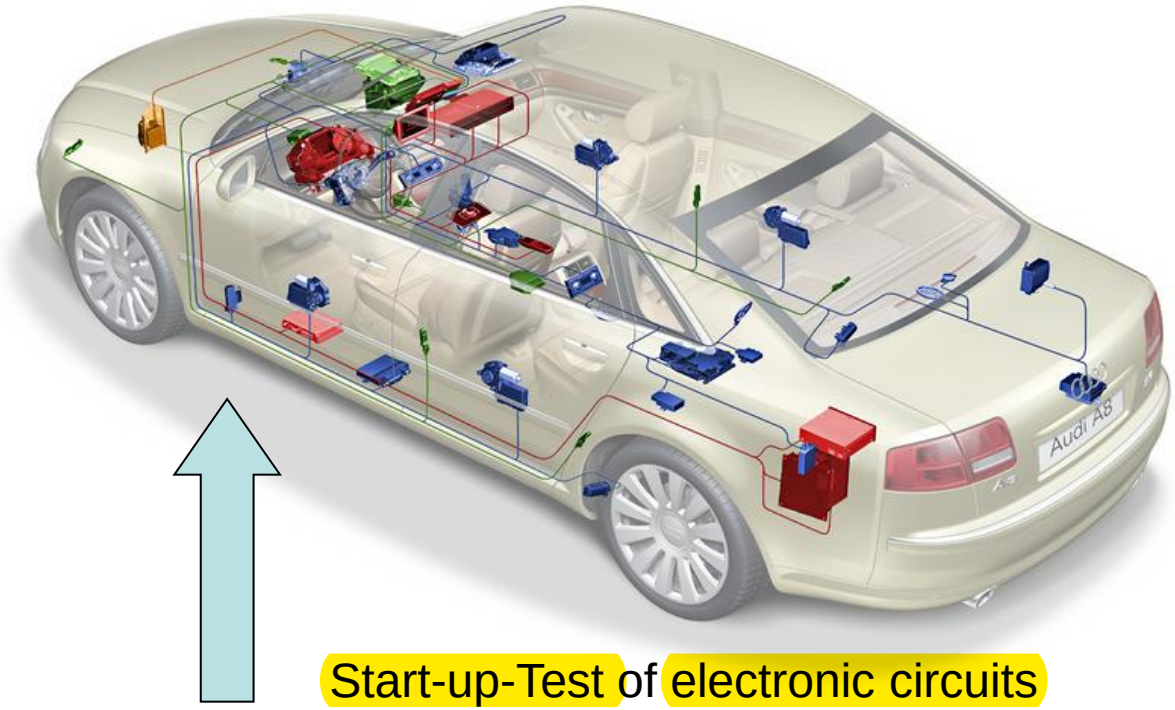




Wafers are shipped to „packaging“ houses for cutting into single dice and packaging. Typically, only „good“ devices are packaged, followed by a final „device test“.

Wafer test can be left out, if alle dies are packaged and tested, but the packages are lost with „non-good“ dies. Test at the packaging house or by special companies.

Off-Line-Test in the Field

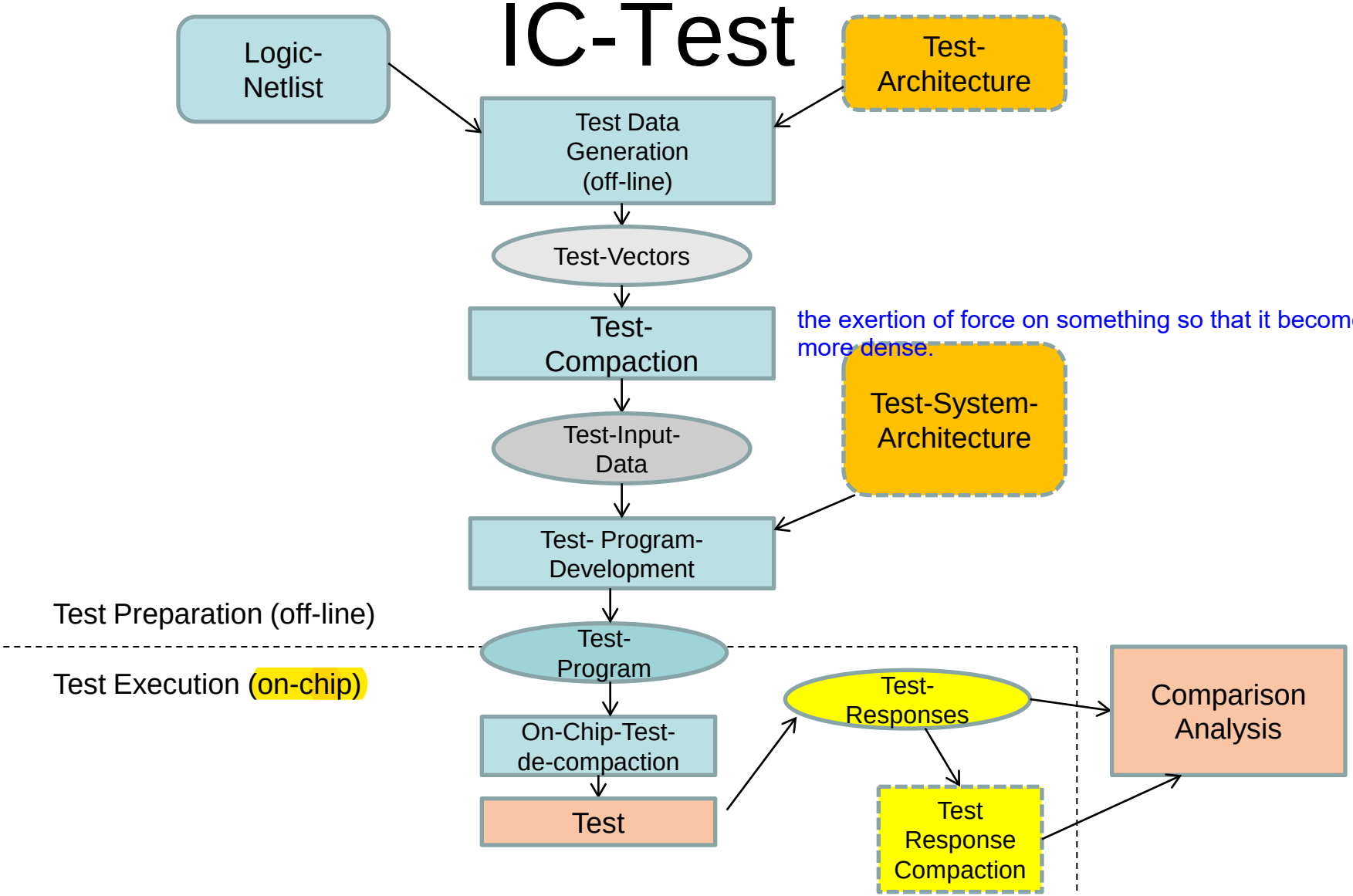


Start-up-Test of electronic circuits

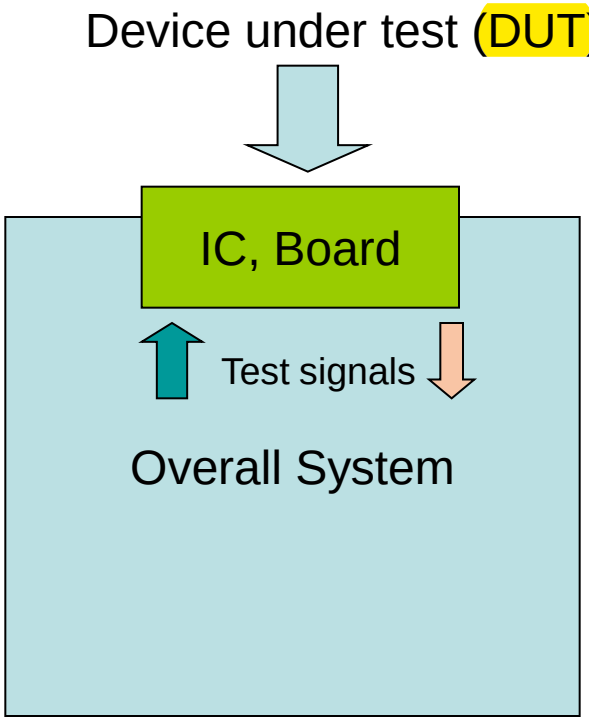
In systems such as cars, airplanes, trains electronic systems get an off-line test at „start-up“ before starting their normal service. Typically, this test procedure is much less detailed and comprehensive than production testing.

complete; including all or nearly all elements or aspects of something.

IC-Test


Logic-Netlist
Test-Architecture
Test Data Generation (off-line)
Test-Vectors
Test-Compaction
the exertion of force on something so that it becomes more dense.
Test-System-Architecture
Test-Input-Data
Test- Program-Development
Test Preparation (off-line)
Test-Program
Test Execution (on-chip)
On-Chip-Test-de-compaction
Test
Test-Responses
Test Response Compaction
Comparison Analysis

Funktional Testing
(Black Box Test)

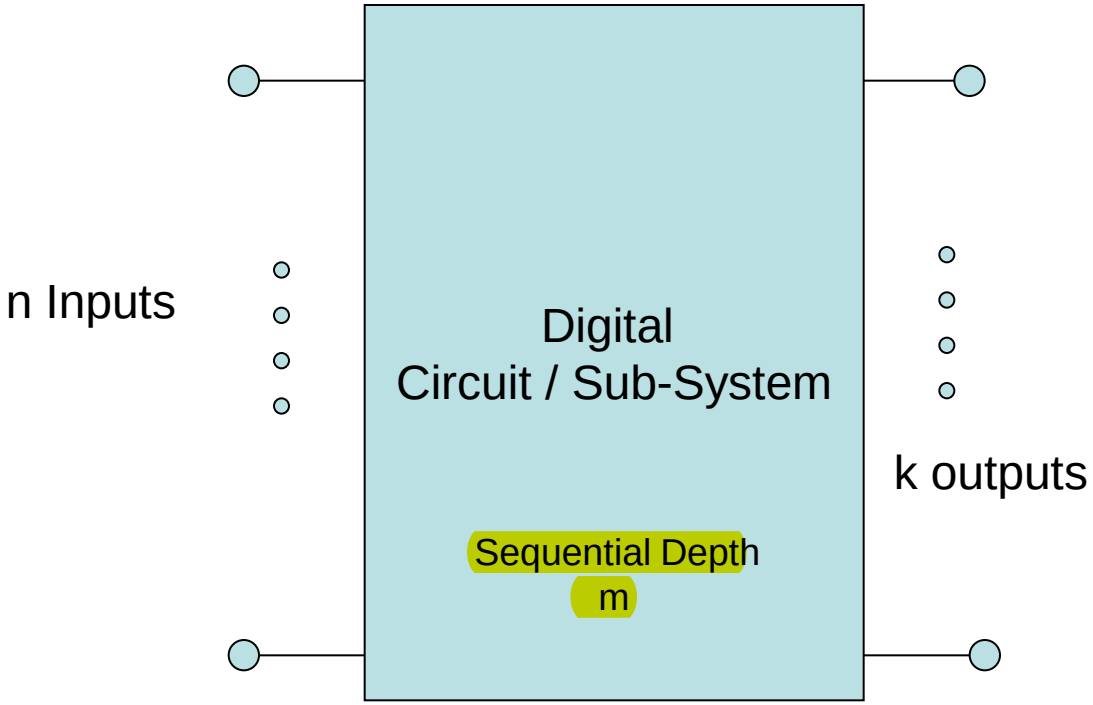


The device under test (DUT) is put into either a real target system of a testing machine (e. g. IC-tester) and then operated with the „real“ signals and observed.

An „on-line“- test of this kind in the real target system (car on highway, plane in flight) is mostly not possible due to cost and safety reasons !

.. but this scheme works for all types of circuits, digtal, analog, mixed… !

Functional Test Complexity



Real for ICs:

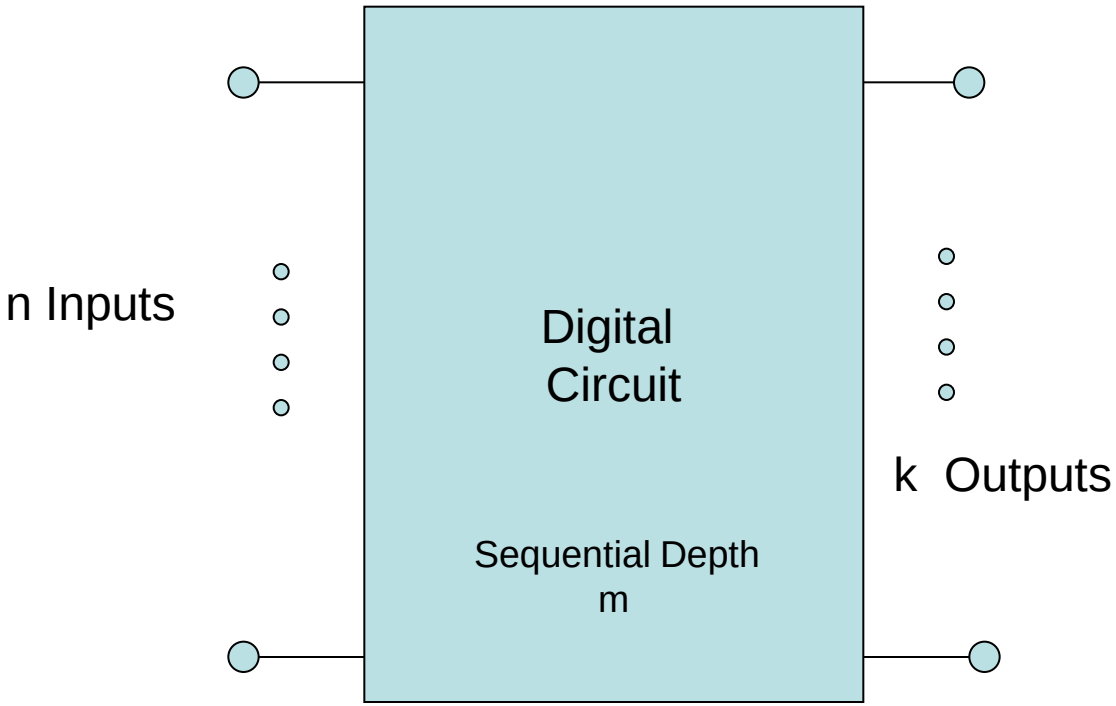
n > 500,
m > 32,
k >250

Number of possible input bit combinations at m = 0 (combinational): $Z = 2^n$

Maximum number of bit combinations for a sequential circuit: $Z = 2^{(m+n)}$

⟹ $Z = 4 * 10^9$ at m = 0, n = 32 !!

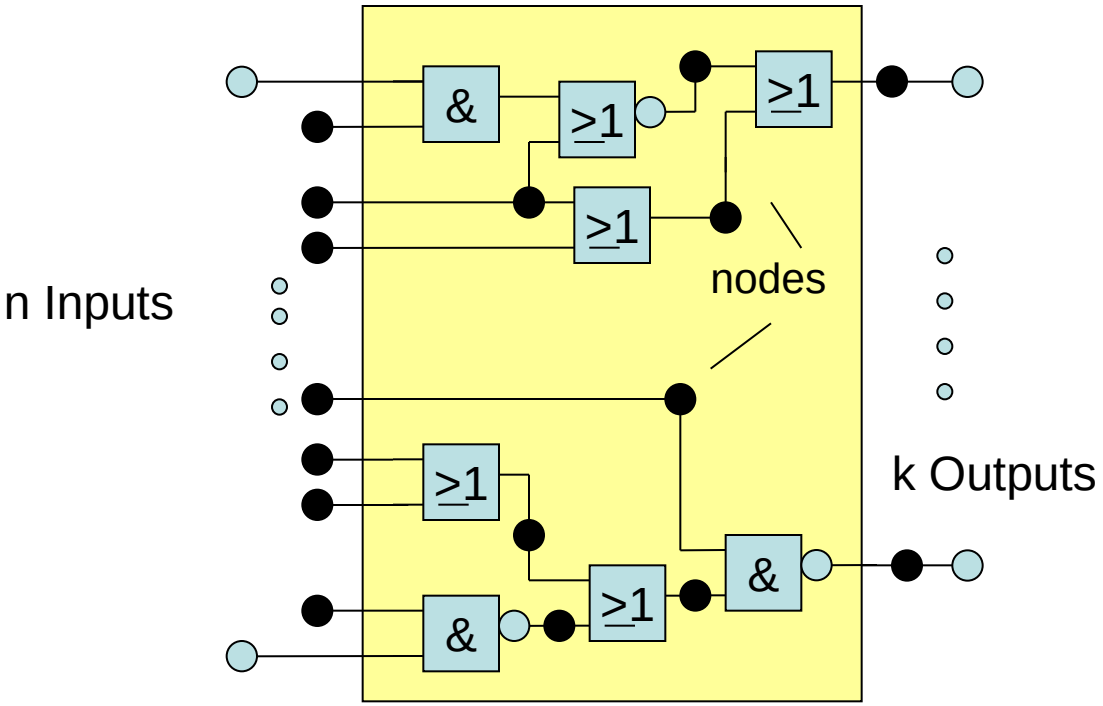
Exhaustive Test



A test is called „exhaustive“, if it has generated all logic states in a system ! This already causes an explosion in the complexity of testing. Actually, a test would be really „exhaustive“, if it would trigger also all possible transitions between logic states !

Structure-Orientiented Test

„white box test“



We know the internal structure of the „DUT“, typically at the level of logic gates. Then we also know the „nodes“ connecting the gates. This allows for a structural „node-oriented“ test approach, based on „fault models“. *Typically such models exist only for digital circuits*.

⟹ Much lower number of test input vectors, compared with exhaustive test !

Fault Models

Fault models describe an assumed, typically simplified fault behavior of structural elements in a digital circuit.

Static fault models:

(node) stuck-at model (Eldred, 1959)

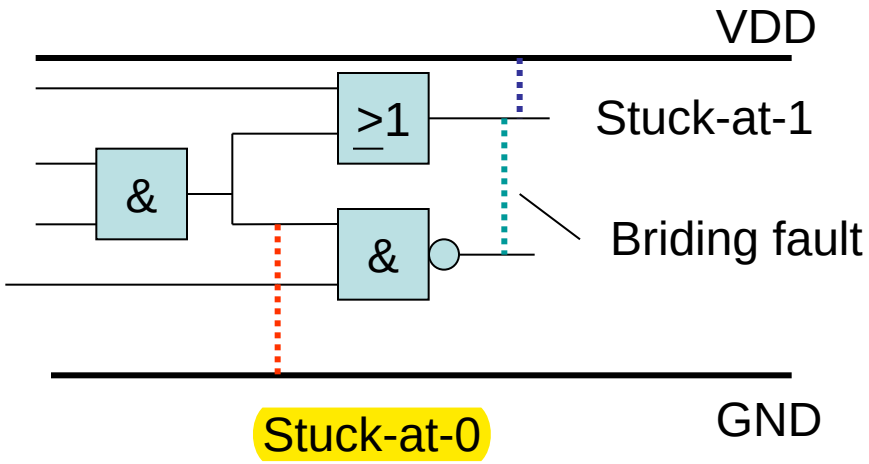
bridging fault (line bridge)

Interrupted line (open) (line-break)

Transistor stuck-on (stuck-short)

Transistor stuck-off (stuck-open)

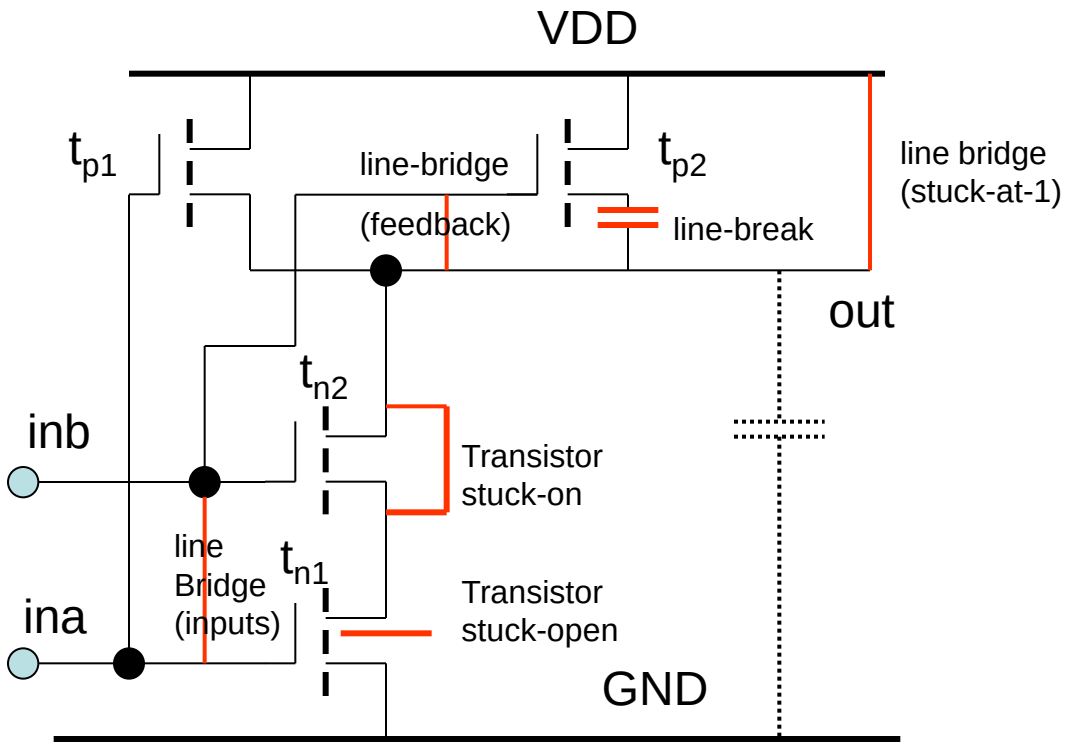
Stuck-at 0 / 1 fault model at gate level



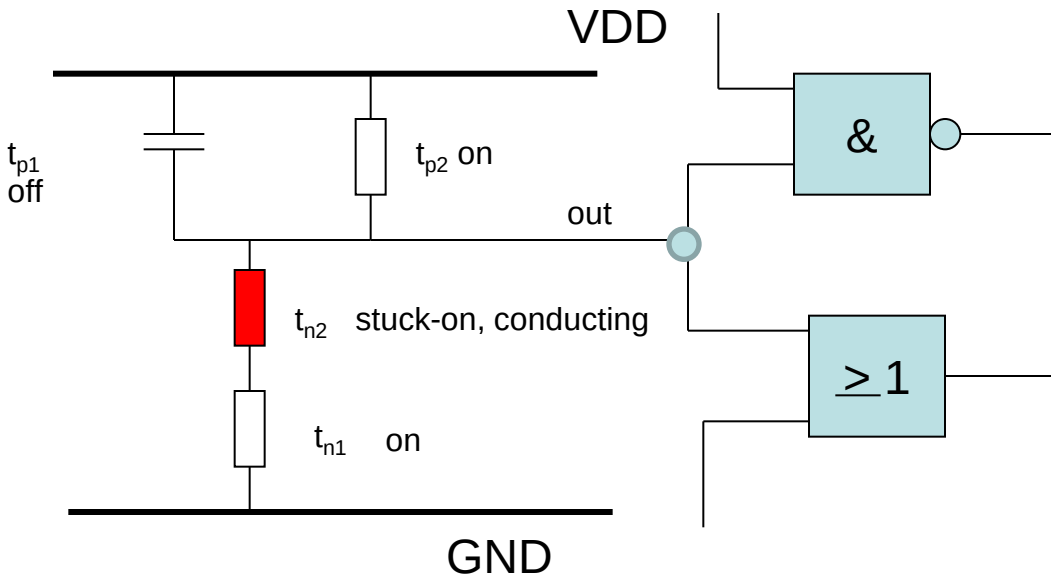
There is no general and accepted set of fault models for analog circuits. Hence analog test is more an art than a science. Favorably, mostly anaog parts are much smaller in real systems.

Typically, test tools also work with a „single fault assumption“, assuming that no combination of faults can „mask“ a specific fault.

CMOS NAND Gate Faults



Stuck-On-Fault and Voltage Divider

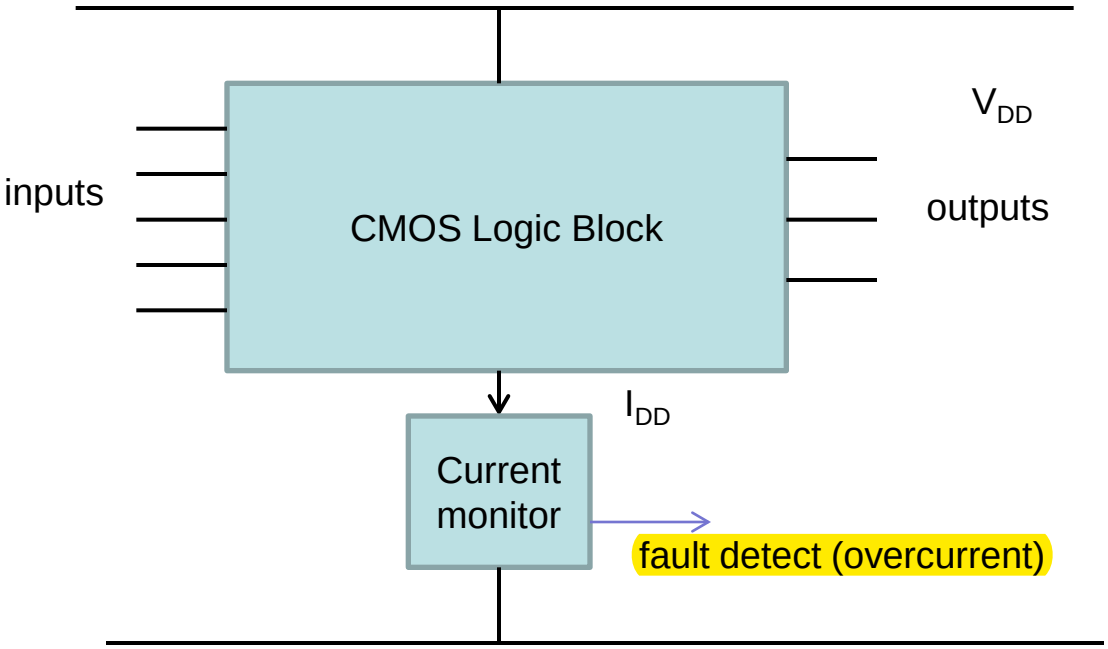


Bridge-type faults between signal lines and stuck-on / stuck-short faults can genarate internal voltage levels, which are neither clearly „0“ or „1“. Even two gated fed this way can see „0“ and „1“.

⇒ „Byzantine General´s Problem“.

Testability is a problem. Either indirectly by delay affects, or by overcurrent test (I_{dd}-test).

Overcurrent Testing

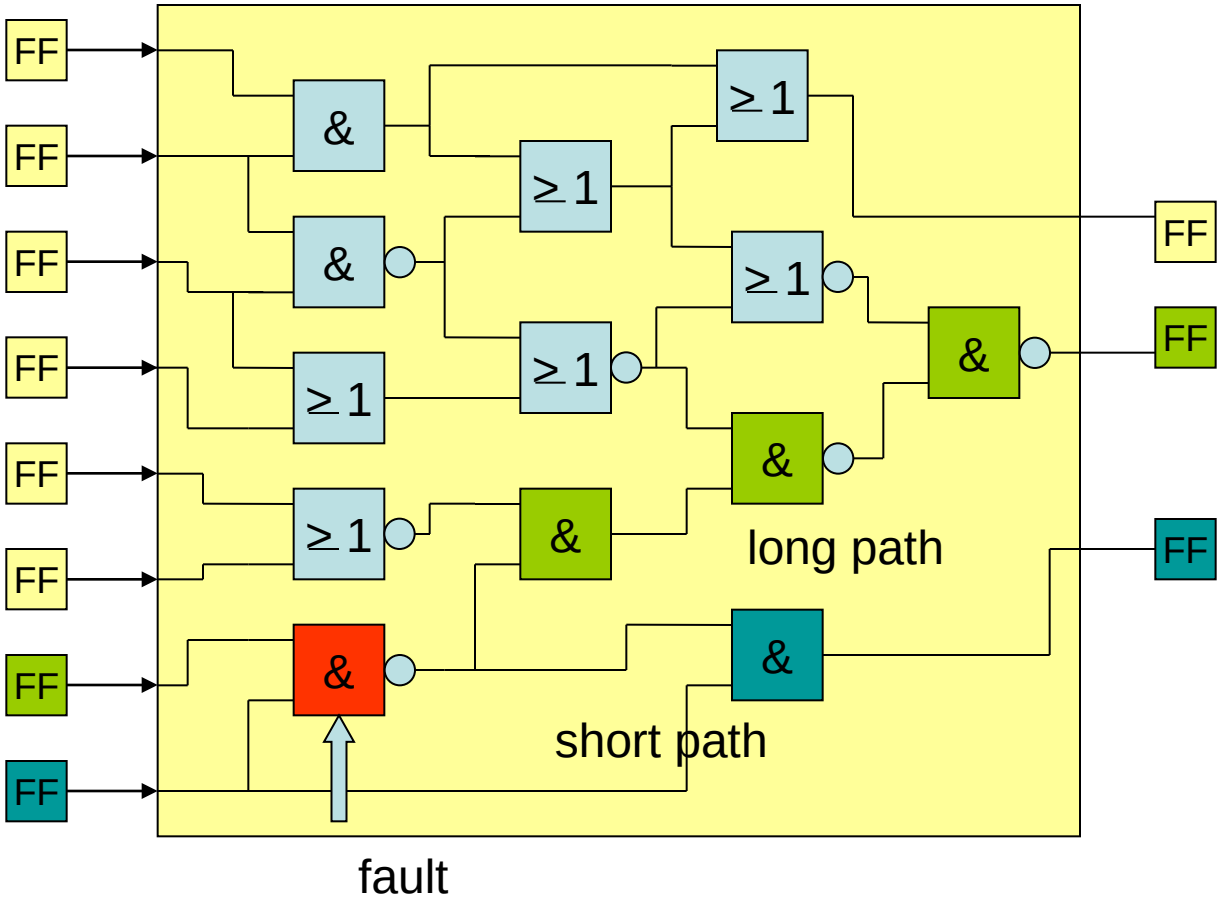


CMOS circuits show quiescent currents in the order of nano-amps, if switching events are not present . This is true for CMOS at structure size over about 100 nm. Hence faults that cause false currents to flow may become detectable (bridges, transistor stuck-on).

Problems: - in nano-technologies, leakage currents may dominate,
- the current monitoring device itself is not very small (about 20 transistors) and consumes part of the supply voltage swing.

move or cause to move back and forth or from side to side while suspended or on an axis.

Dynamic Faults



Transition fault: A logic gate cannot be switched to out 0 / 1 and 1 / 0 by one or more of its inputs.

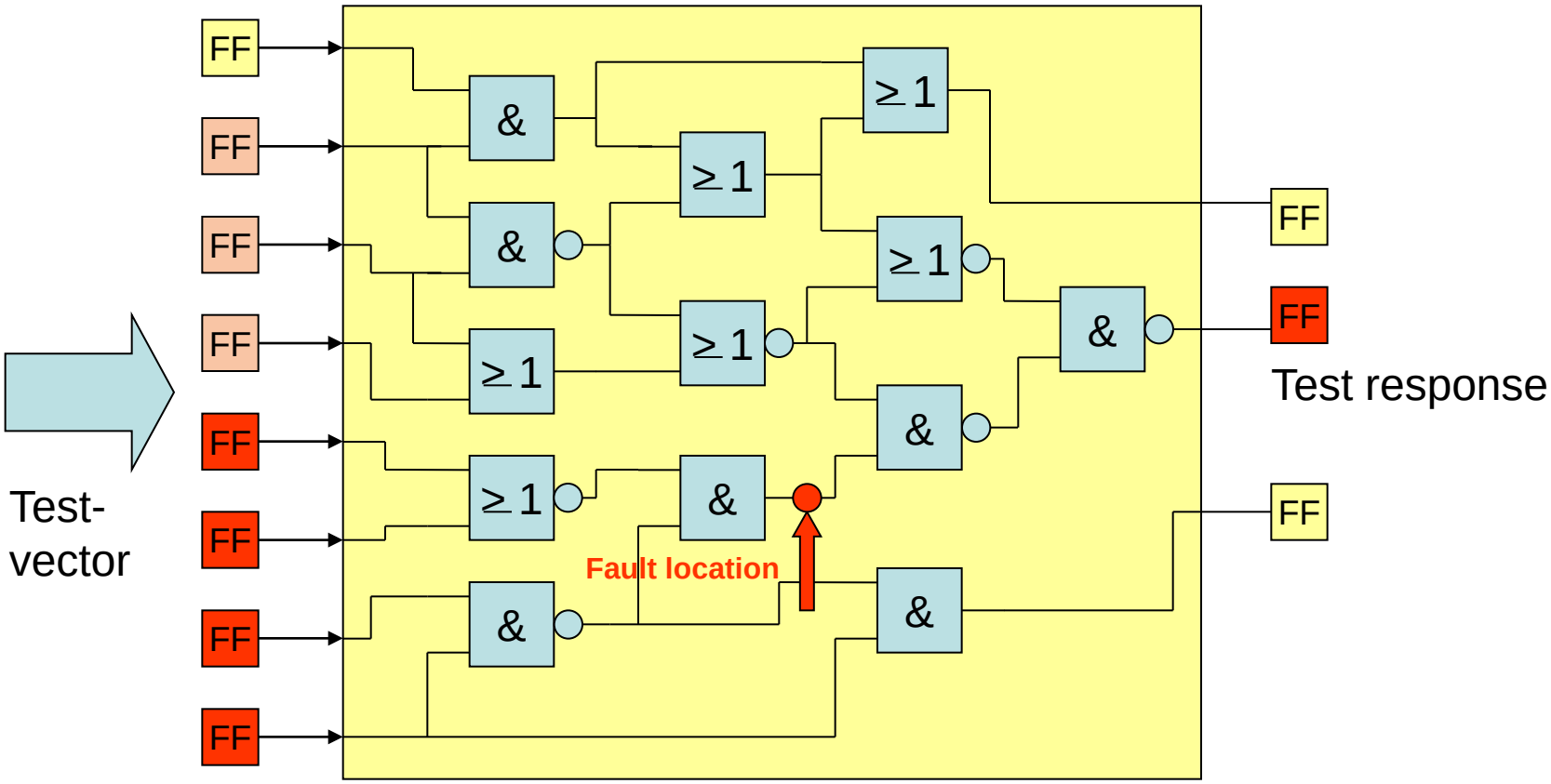
Delay fault: A single gate or a logic path performs 0/ 1 or 1/0 switching at output(s) with a significant delay..

Test Vectors

- A **test vector** is a digital input pattern to a (combinational) circuit under test, which can:
  - excite the potential fault event at the site of the fault (e. g. try to bring a node which is „stuck-at-0“ to a logic „1“),
  - make the fault event visible at outputs by setting a path for propagatimng the fault effect through the circuit to an output to make it „visible“.
- A **test pattern** is almost the same. But often the term is used even if a pair of test vectors is used.
- Static faults (such as stuck-at) need a single test vector for excitatiion and propagation.

  the application of energy to something.
- Dynamic faults (such as path-delay) need a pair of test vectors (init vector, test vector) to make the fault effect detectable.
- For sequential circuits, testing for only a single static fault typically requires a „test sequence“, which may be hundreds of test vectors long, depending on the sequential depth.
- The sum of all test vectors generated to test a logic circuit is the „test set“.

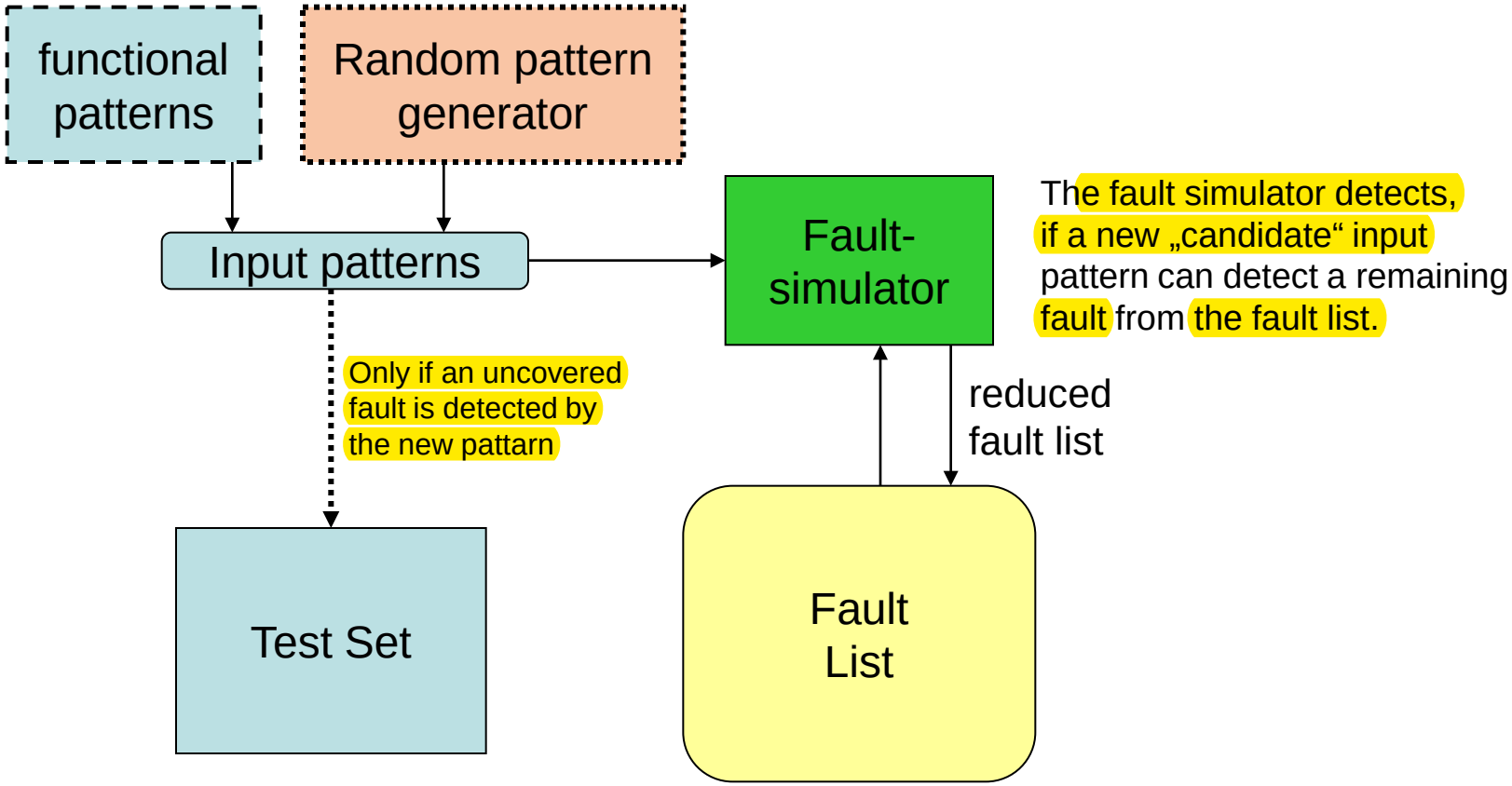
Generating a Test Vector



1. Select the fault location
2. Generate the setting at the fault location that „envokes“ the fault
3. Propagate the fault effect to an observable (primary) output

Test Patterns / Test Vectors

using functional and / or random patterns



Fault Simulator

A fault simulator is a simulator tool, which can simulate the behavior of a digital circuit in the „good“ state as well as under various fault conditions. The fault simulator thereby „assumes“ specified fault models. It is primarily used to find out, if a specific input vector can detect (cover) a specific fault or not..

Problem: Complexity

With a circuit that has n nodes, using the single stuck-at fault model, we get 2n different fault cases plus the „good“ case. If we investigate k test vectors, this means R = k* (2n+1) simulation runs. Any „normal“ simulator will fail for complexity reasons. Therefore specific fault simulators are a must.

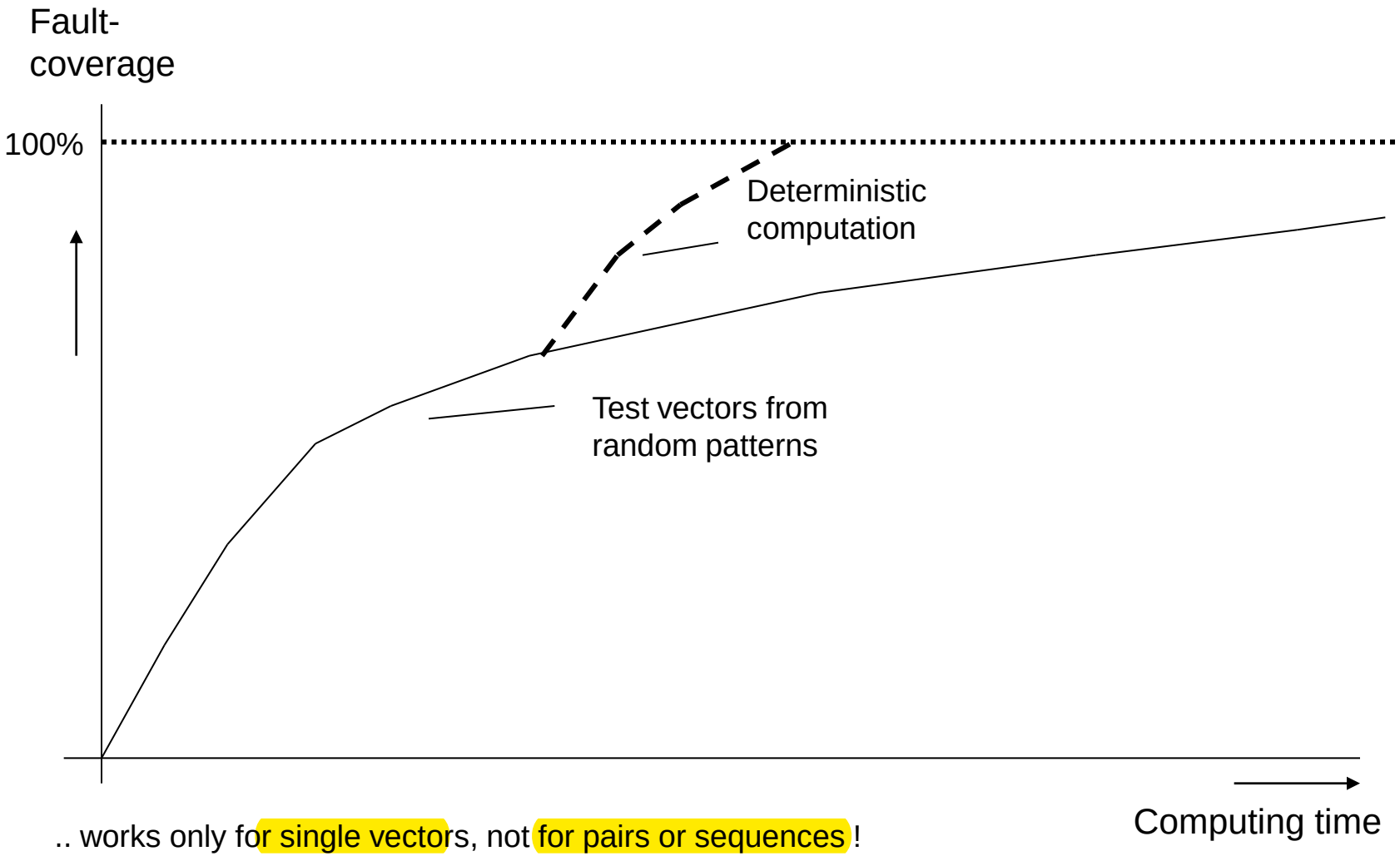
Serial fault simulation: Each fault and the good case is simulated individually. Far too slow !

Parallel fault simulation: Several fault cases are simulated in parallel. As most runs, while going through the „good“ part of the circuit, will not differ from the good case, a lot of effeort can be saved systematically.

Parallel fault: For the same input vector, several different fault cases are simulated in parallel.
Parallel pattern: For the same fault case, several different input vectors are simulated in parallel.

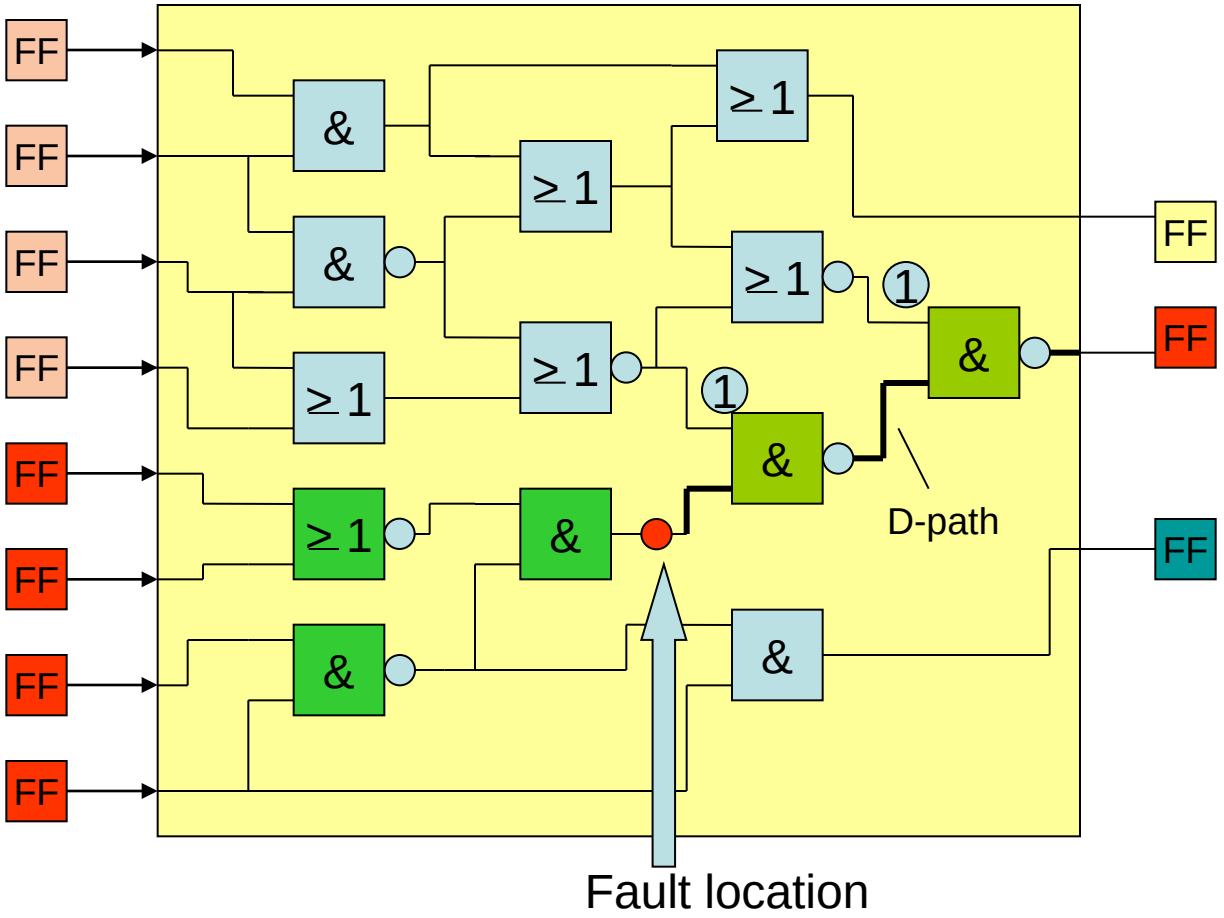
Deductive fault simulation: The „good“ part of the circuit is simulated once only. From a fault location, the simulation is split into 2 strings, describing the faulty and the good circuit. At every subsequent fault location, the specific fault simulation is again split from the good simulation. Problem: exploding data base !

Fault Coverage by Random Patterns



.. works only for single vectors, not for pairs or sequences !

Path Tracing Methods



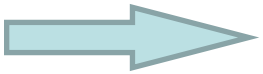
Test-Abstractions

- Test patterns and test sets are calculated almost ever from structural information at the gate level.
- Fault models in real use are:
  - single stuck-at,
  - transition fault model,
  - delay model for longest paths.
- Modeling always assumes a single fault only. This is reasonable, since dice are discarded after the first fault is found (in most cases). The probability that 2 faults may mask each other is small. However, it is not impossible ! Working with multiple fault assumptions explodes complexity.
- For cost reasons, test sets with minimum test lenght / minimum test application time are preferred. Often test sets are compacted by methods such as „reverse simulation“.
- Typically, a test set should cover 100 % of testable single stuck-at faults. This does not include untestable redundant nodes.
- Diagnostic testing, based on patterns that can deliver an optimum doagnostic resulution, is uncommon because of high cost.
- Test pattern sets for combinational logic often get along with much less than 1or 2 (dynamic faults) per node, since many patterns can cover several different faults.
- Tests generated with automatic tools deliver patterns that are quite different from „normal“ operation.

Algorithmic Test Generation SSA-Faults

1. Select the „target node“ or the „fault location“ and the type of fault (s. a. 0 / 1).
2. Try to find a path through the circuit to propagate the fault to a primary output (forward trace)
3. Set the inputs of all gates on the propagation path and the gate feeding the faulty node such that the right conditions for fault propagation are established (multiple backward trace).

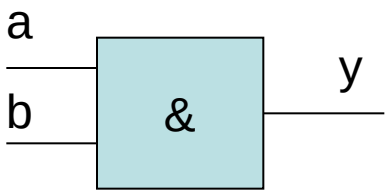
Problem: This process does not work easily, because:
- in backward path tracing, there are often „degrees of freedom“, specifically at fan-out nodes.
- the knowledge that a provisional setting of a node was false and causes contractions in the required setting of nodes may come up much later in the serach process..



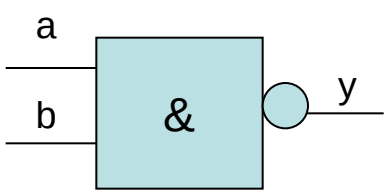
Backtrack of the search tree with all settings at the last decision point !

Algorithmc test pattern genaration is a very hard (n-p-hard !!) problem for combinational circuits already, which typically explodes with complexity. Under worst case, the effort grows exponentially with circuit size.

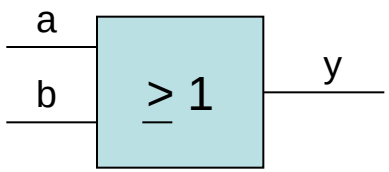
D-Notation



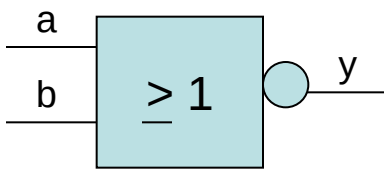
a	b	y
0	0	0
0	1	0
1	0	0
1	1	1
D	0	0
0	D	0
1	D	D
D	1	D
$\overline{D}$	0	0
0	$\overline{D}$	0
1	$\overline{D}$	$\bar{D}$
$\overline{D}$	1	$\bar{D}$
D	D	D
D	$\overline{D}$	0
$\overline{D}$	$\overline{D}$	$\overline{D}$



a	b	y
0	0	1
0	1	1
1	0	1
1	1	0
D	0	1
0	D	1
1	D	$\overline{D}$
D	1	$\overline{D}$
$\overline{D}$	0	0
0	$\overline{D}$	0
1	$\overline{D}$	D
$\overline{D}$	1	D
D	D	$\overline{D}$
D	$\overline{D}$	1
$\overline{D}$	$\overline{D}$	D



a	b	y
0	0	0
0	1	1
1	0	1
1	1	1
D	0	D
0	D	D
1	D	1
D	1	1
$\overline{D}$	0	$\overline{D}$
0	$\overline{D}$	$\bar{D}$
1	$\overline{D}$	1
$\overline{D}$	1	1
D	D	D
D	$\overline{D}$	1
$\overline{D}$	$\overline{D}$	$\overline{D}$



a	b	y
0	0	1
0	1	0
1	0	0
1	1	0
D	0	$\overline{D}$
0	D	$\bar{D}$
1	D	0
D	1	0
$\overline{D}$	0	D
0	$\overline{D}$	D
1	$\overline{D}$	0
$\overline{D}$	1	0
D	D	$\overline{D}$
D	$\overline{D}$	0
$\overline{D}$	$\overline{D}$	D

Conflict-Table for D-Algorith*

	0	1	X	D	$\overline{D}$
0	0	K	0	K	K
1	K	1	1	K	K
X	0	1	X	D	$\overline{D}$
D	K	K	D	D	A
$\overline{D}$	K	K	$\overline{D}$	A	K

K: Conlict, cannot be resolved

⇒ Backtrack !

A: Resolvable conflict, select $\overline{D}$ oder D.

ATPG technology: Since backtracks are enormously costly in terms of computing power, try to apply heuristics which minimize the orrucrence of backtracks.

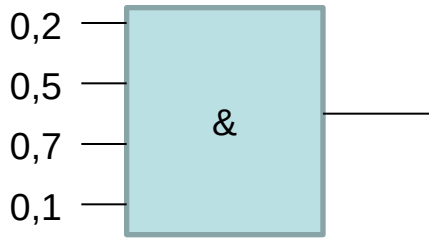
enabling a person to discover or learn something for themselves

* First industrial ATPG tool, Roth, IBM, about 1965.

ATPG-Control by formal Controllability / Observability

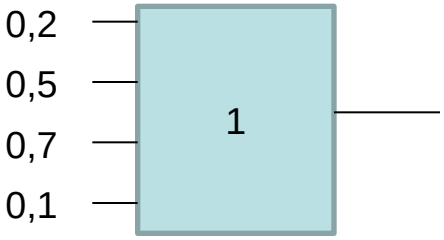
In a pre-process it is formally evaluated for every node, how easily it can be set to 0 / 1.
a measure of „1“ means, it can directly be set from inputs, „0“ means the node is not controllable at all.

Formal evaluation of controllability



„0“ at the output is **easy** to set:
only one input must be at „0“.

„1“ at the output is **difficult** to set, since all inputs must be at „1“.

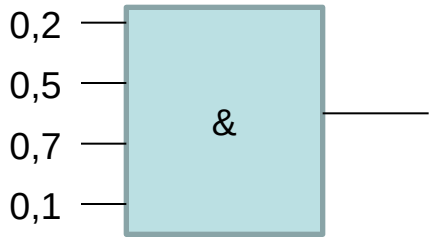


„1“ at the output is **easy** to set, since only one input must be at „1“.

„0“ at the output is **difficult** to set, since all inputs must be at „0“.

Directing Path Search

Controllability



If the „easy“ condition is to be set, then use and set the input which is easiest to control.

If the „difficult“ condition is to be set, start with the least controllable input. If you cannot set that one, forget about the other inputs.

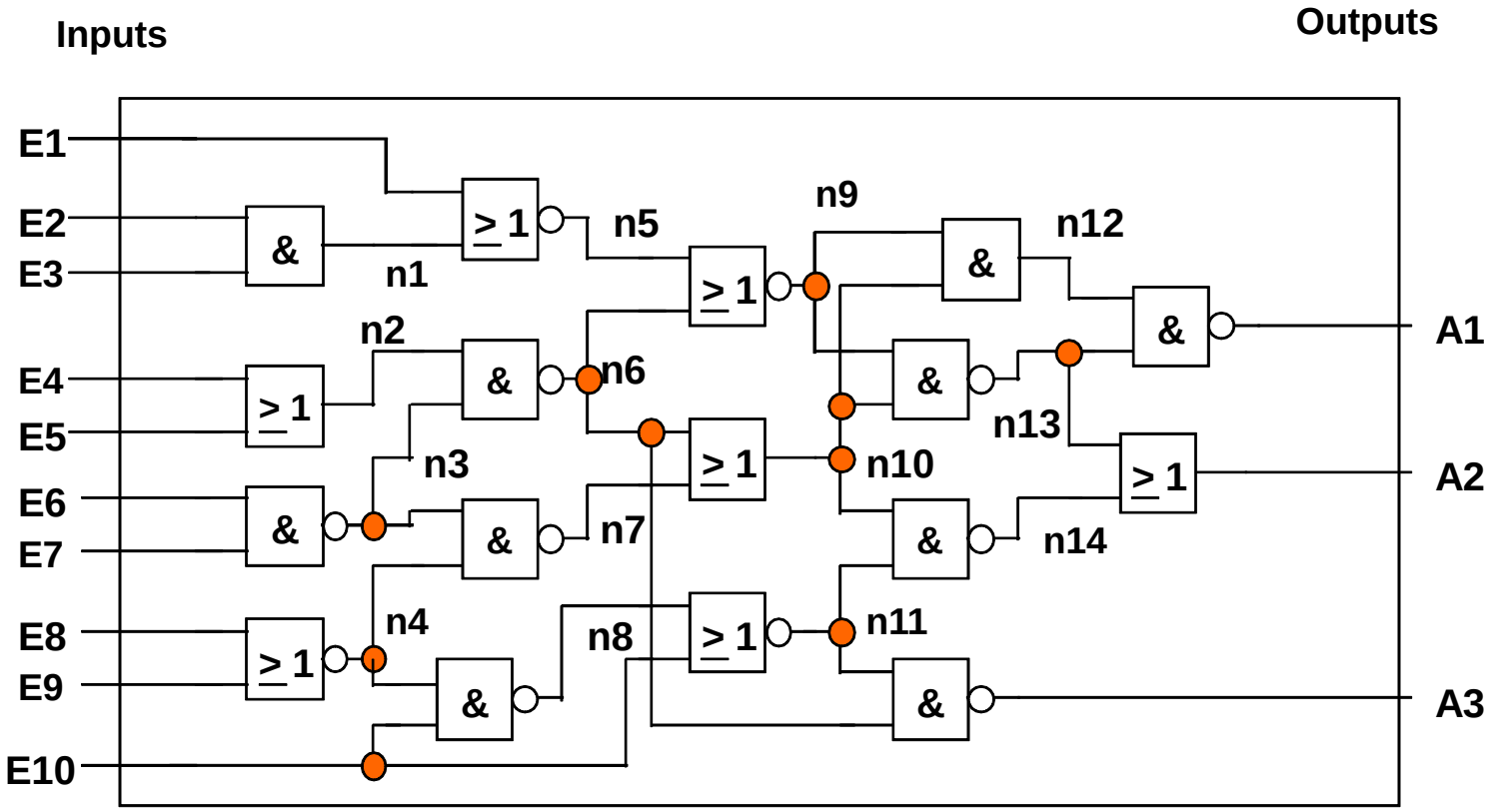
Practical ATPG tools limit the allowed number of backtracks.

If a pattern cannot be found even after several backtracks using the heuristics, try again with all heuristics reversed.

If a pattern for a specific fault cannot be found directly, stop and leave it for a „hard“ list. Often even „hard to find“ settings are created „by chance“ when searching for other tests.

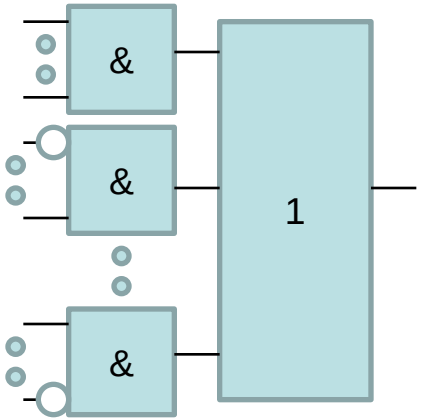
Test sets are often compacted by re-simulation. Then patterns for „hard“ faults come first, which will often cover many other „easy“ faults as well.

FAN-Algorith and Learning Function



FAN keeps a record about „favorable“ setting of fanout-nodes. Any time a multiple backtrace arrives at a FO-node, the favorable setting is recorded. Path propagation is continues elsewhere, until no progress can be made without „resolving“ a FO-node. This is done as a „majority vote“.

SAT-based Test Generation

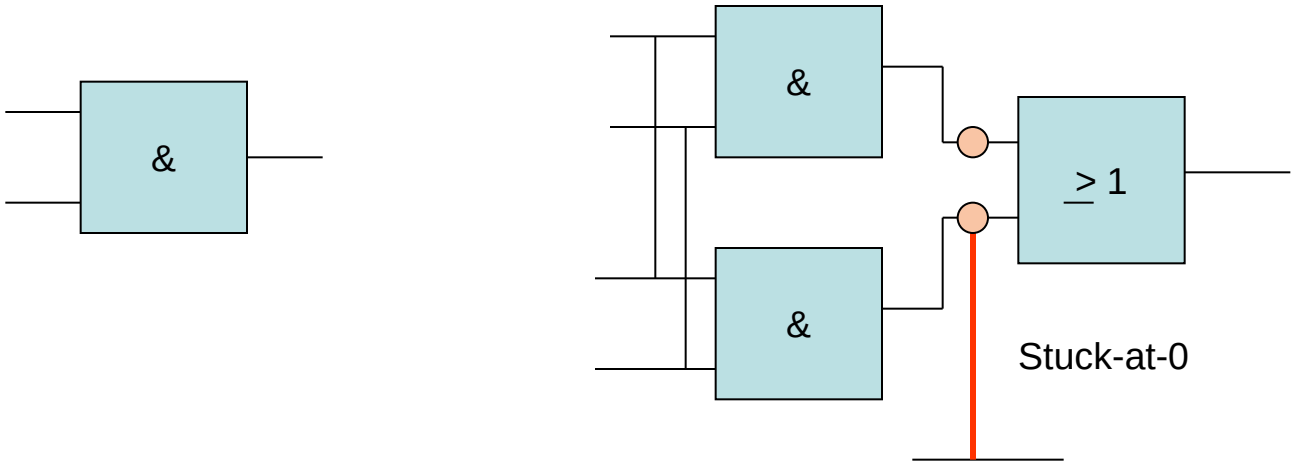


regular 2-stage logic
(according to disjunctive normal form, DNF).

If a logic circuit has a structure that can be described by a disjunctive normal form (DNF) or a conjunctive normal form (CNF), then there are special tool, so called „SAT-solvers“, which can rapidly , if there is an input pattern that can produce a specific output.
This principle can be applied only, if a normal (combinational) logic circuit can be re-structured into a DNF or a CNF. Such a conversion proved to be much easier than expected.

State-of-the-Art ATPG tools often use SAT-Solvers !

Logic Containing Redundancy



Redundancy generates untestable faults. Therefor many „real“ circuits are not fully tetsable.
We can distinguish between „combinational“ und „sequential“ redundancy

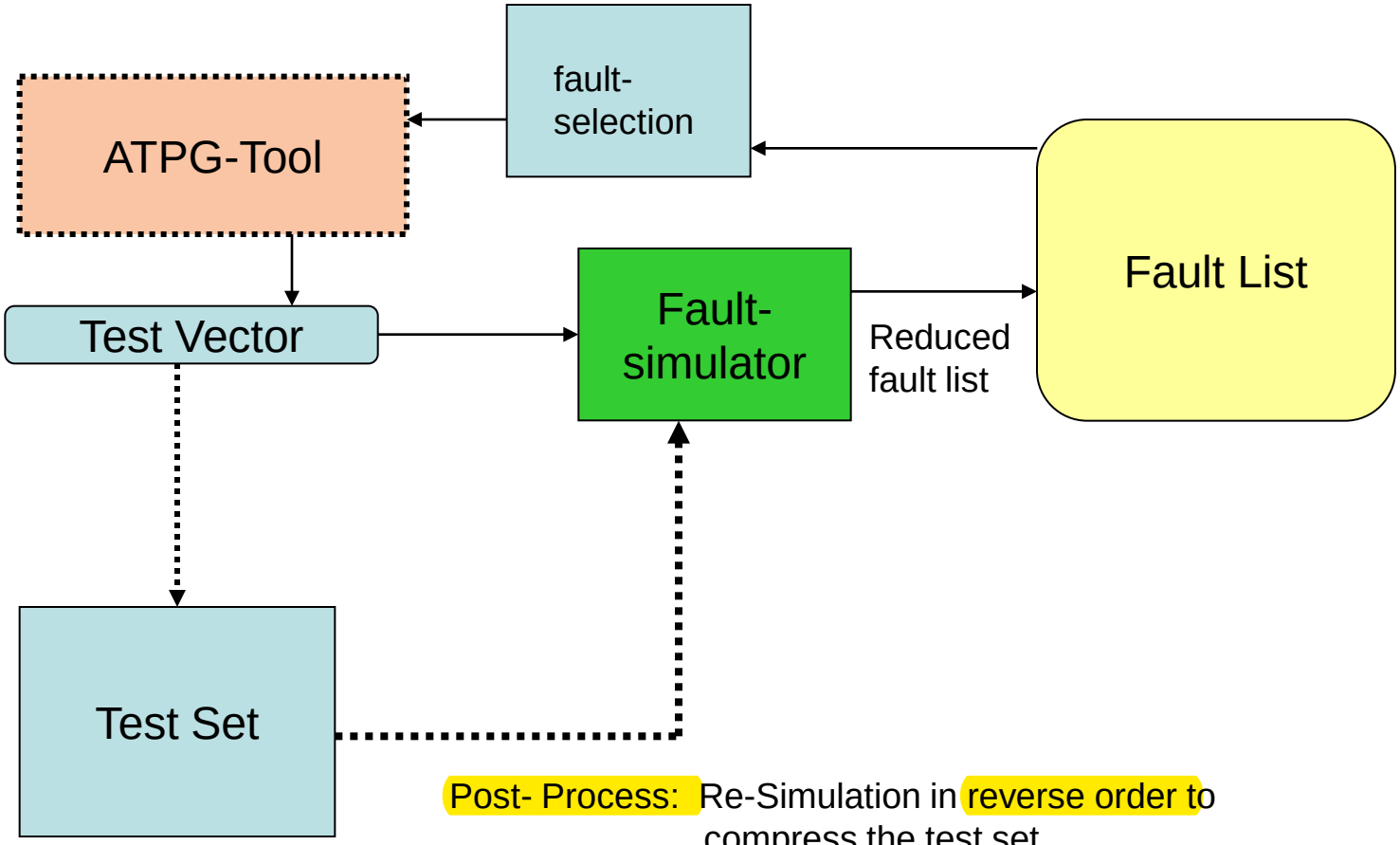
Detecting redundancy is about as complex and expensive as direct ATPG !

⇨ Many real circuits, even those generated by logic synthesis, contain 2-5 % of redundant gates !

„Fault / test coverage“ of a test set : Share of all assumed faults that can be tested.

„Test effectiveness“: Share of all testable faults that are tested by the test set.

Test-Compaction



Post- Process: Re-Simulation in reverse order to compress the test set.

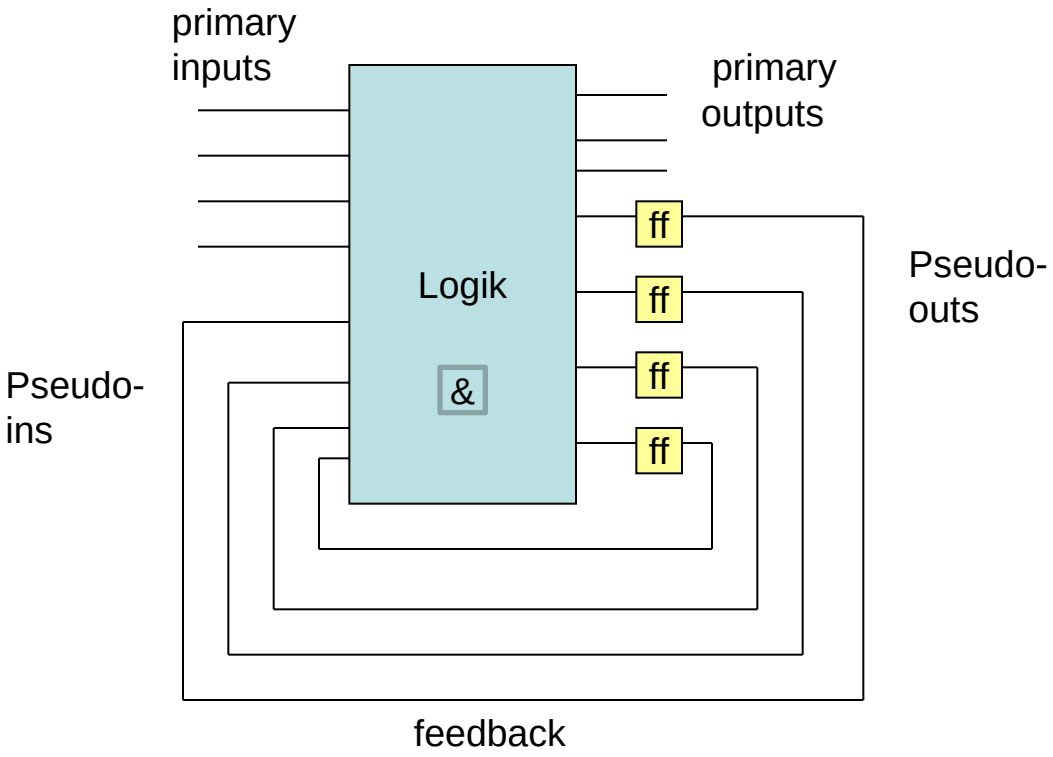
What makes Circuits „Untestable“ ?

- High sequential depth. Effectively generated by „burried“ counter devices.
- Parts of the circuits which are not controlled by diect and simple clocking schemes. Asynchronous circuits are worst. But also circuits where clock trees are controlled by signals are harmful .
- Digital circuits, which can be controlled / observed only indirectly via analog circuits.
- Digital circuits which are controllable / observable only through memory blocks.

→ Analog circuits and memory blocks must habe a direct test access from the outside.

By the way: Memory blocks need a special test approach, but because of their regularity they are candidates for hardware-controlled built-in self test.

Sequential Circuit



In sequential circuits, it may take a sequence of several input vectors to set an internal node to a specific value, (initialisation problem). Then, again, it may take several to many clock cycles, before a fault appears at a primary output. That means, a single fault may need a specific sequence of dozends or hundreds of test vectors !!!

Automatic Test Generation (ATG) for Sequential Circuits

- Automatic test generation / test pattern generation (ATG, ATPG) is a complex algotithmic path-tracing problem for combinational circuits already.

- For ATG in sequential circuits, we need a suitable model. A standard approach is to multiply and cascade the circuits according to the number of time steps needed for initialisation, fault excitation and propagation to outputs.

  pass (something) on to a succession of others.

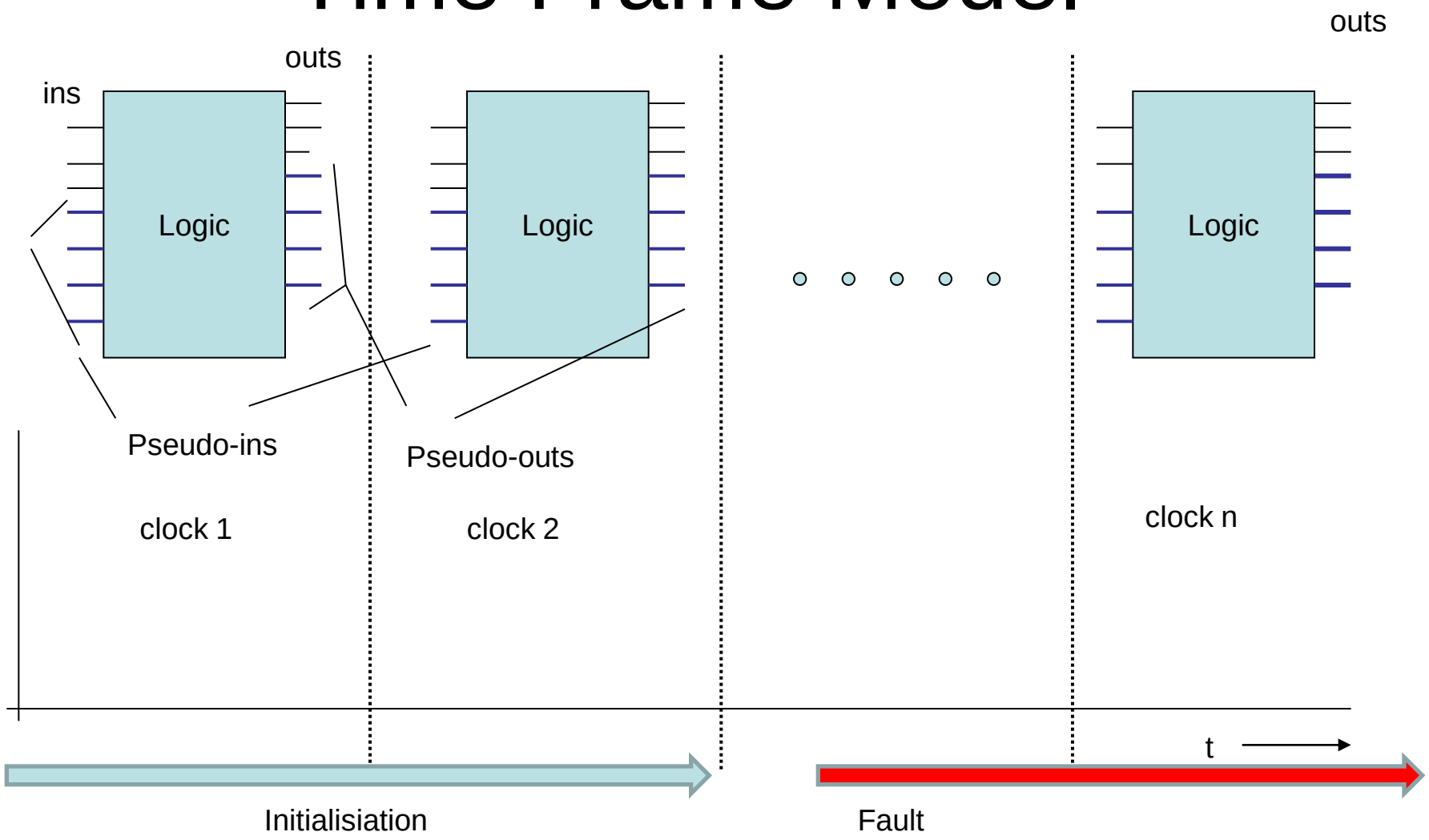
- In the resulting „time frame“ model, the circuit is actually multiplied be a factor of k+(n-1), if it takes k clock cycles to excite the fault and n clock cycles to propagate it to a primary output.

  give rise to (a feeling or reaction).

- Path tracing for sequential ATPG includes multiple backtrace over several time steps. This will often include backtracks over time steps. Typically, an ATPG tool will be able to control only a limited number of time steps at a time.

- Optimization methods such as „genetic algorithms“ have been used with some success in sequential ATPG.

Time Frame Model



Testing Sequential Circuits

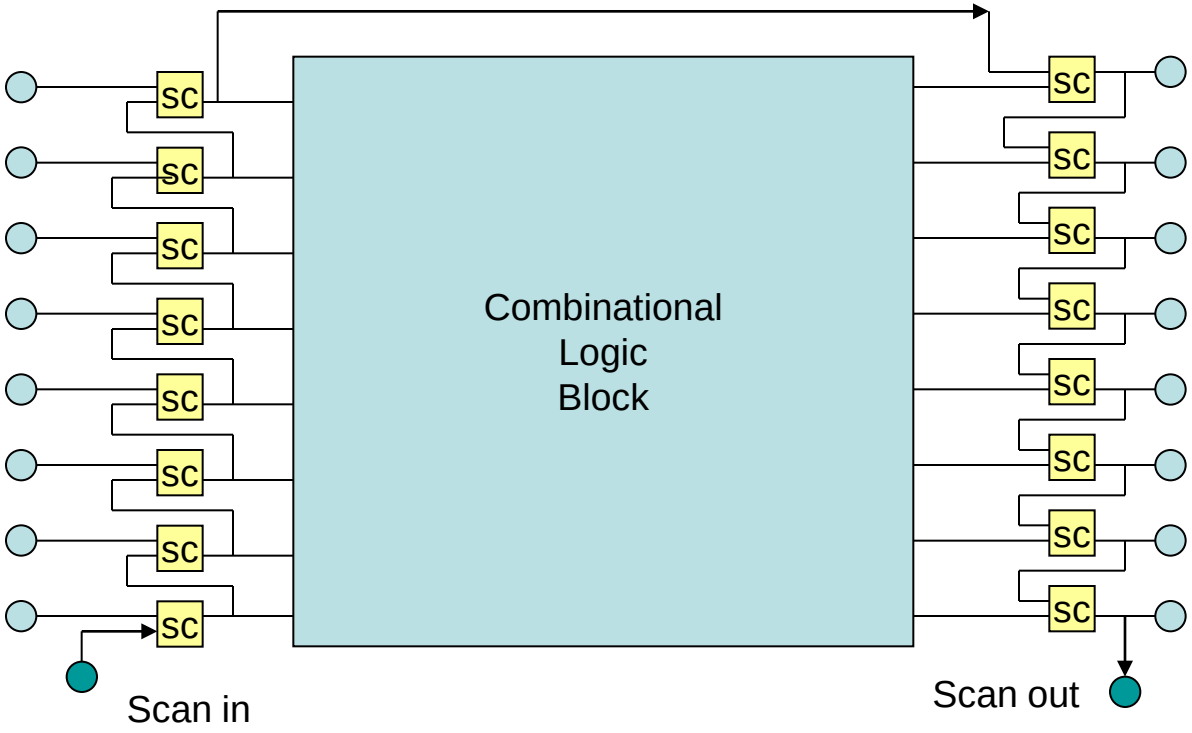
- Sequential circuits require sequences of test vectors for every single fault. Of course, a sequence may also catch other faults „by chance“.
- Automatic test pattern generation for sequential circuits is even much more complex and costly than ATPG for combinational logic.
- Using test sequences will blow up the site of the test sets.

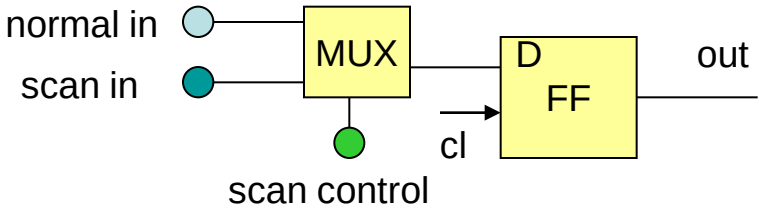
⇒ **Structured design for testability !!**

Step 1: Make storage elements such as flip-flops and registers easy to in initialize by an extra „Reset“ input.

Step 2: Allow for test access as if the circuit were combinational.

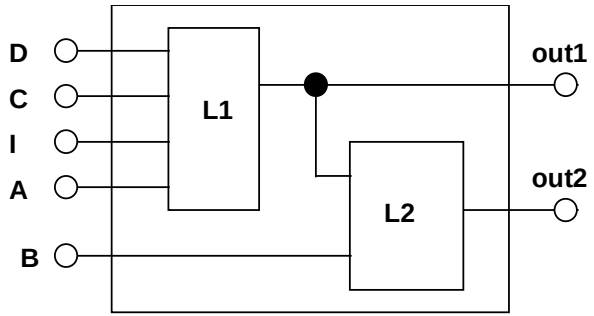
Scan-Path-Technology

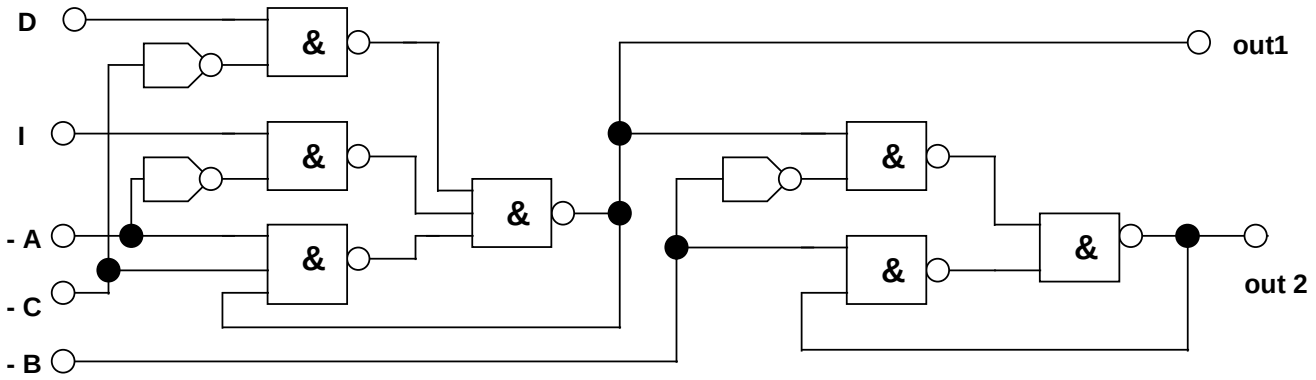




Input / output flip-flops are modified into scan- FFs, which are connected to make a „shift register“ in a specific scan-mode. We get enhanced test access with only one extra input and output !

Level-Sensitive Scan Design (LSSD)



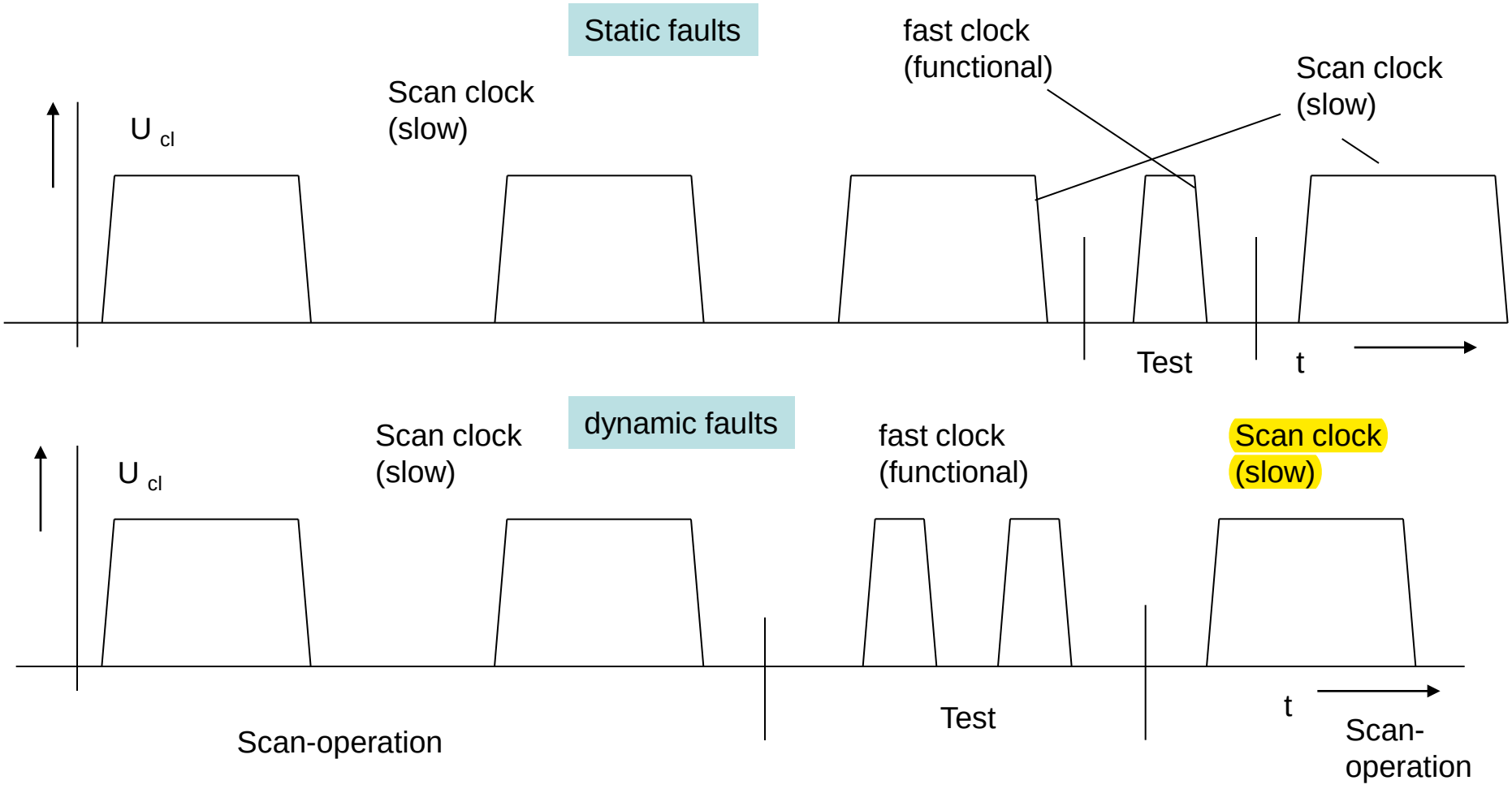


Developed by IBM. Uses double-latch, level-sensitive latches and two phase- shifted clock signals.

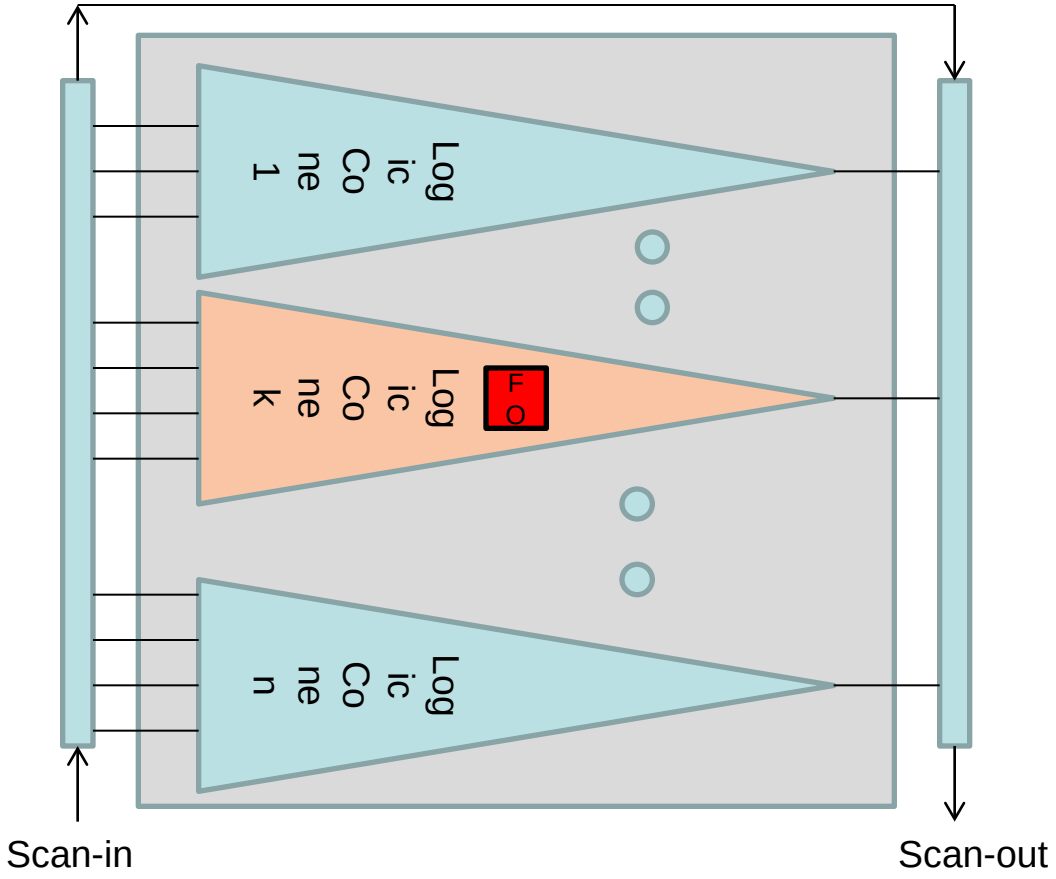
How Does Scan Test Work ?

- The circuit under test is modeled as one single large block of combinational logic. Inputs from flip-flops are modeled as „pseudo-inputs“, outputs into flip-flops are modeled as „pseudo-outputs.
- All flip-flops are replaced by scan – FFs and inserted into one or more scan chains for test access.
- Every scan chain obtains its own (extra) primary input and output.
- All scan chains are connected with an extra control signal, which allows to switch inputs between „functional in“ and „scan in“.
- Test vectors are fed in serially via the scan chain. Test outputs are captured in the scan chain and can be scanned out to a primary output.
- After „scan-in“ of a test vector, the circuit is operated functionally for a single clock cycle, so the output scan elements can capture the test output. Typically, the scan-clock is slower than the system clock, about 20-50 MHz.

Fast / Slow- Clock- Scheme



Virtual Structure for Scan Test

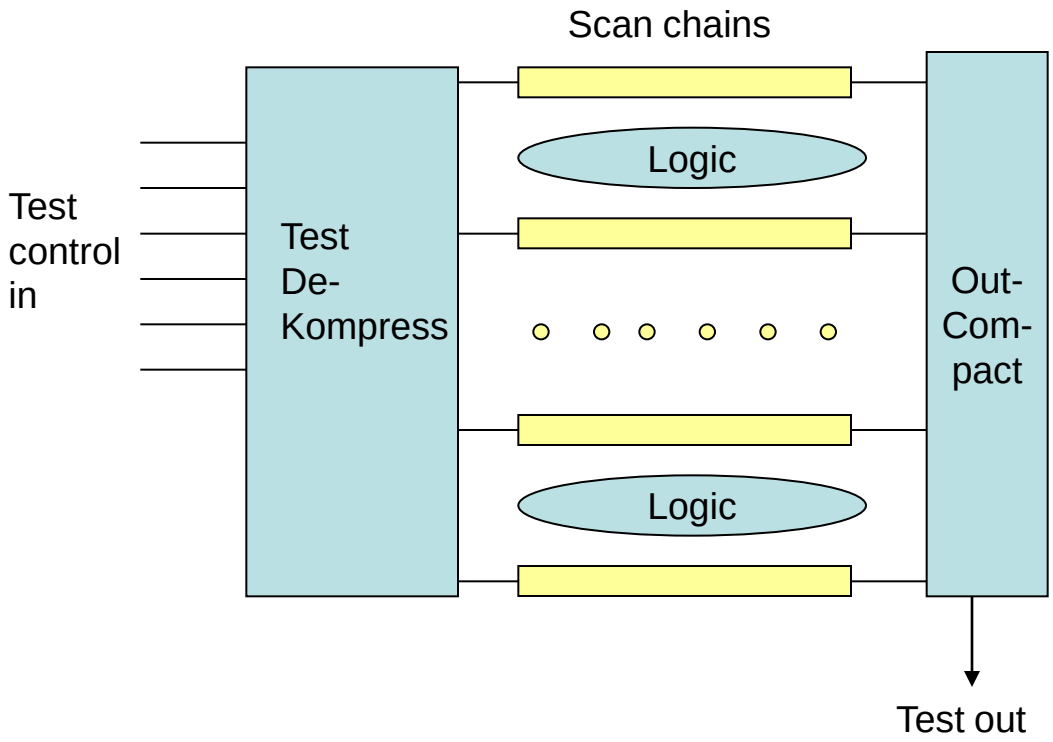


Large circuits require thousands of scan elements, resulting in long scan path and long scan-in and scan-out times. Faults in such broad but combinational shallow circuits often have an influence only on a small part of the whole circuit. May inputs are not related to a specific fault !

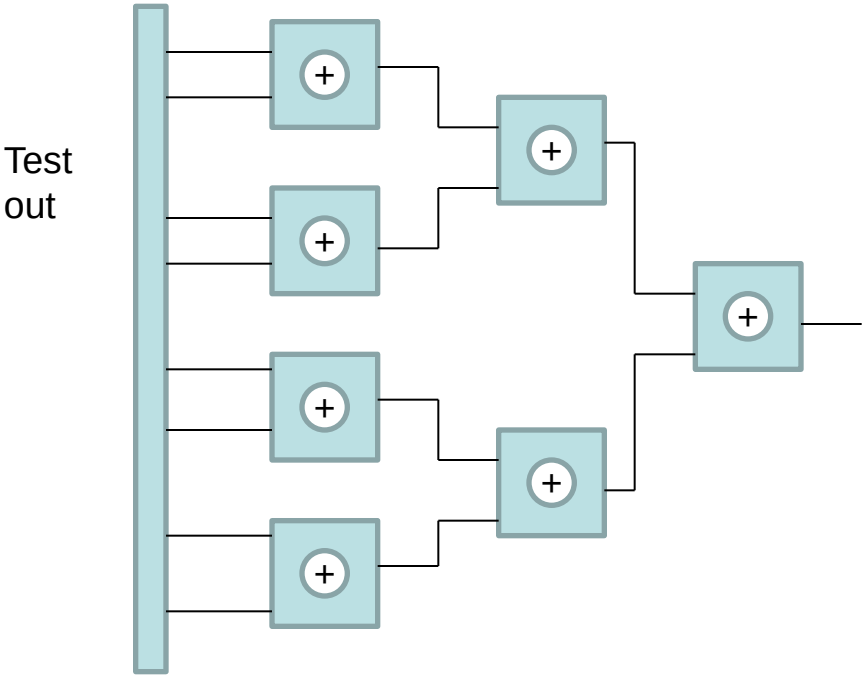
Test Compress

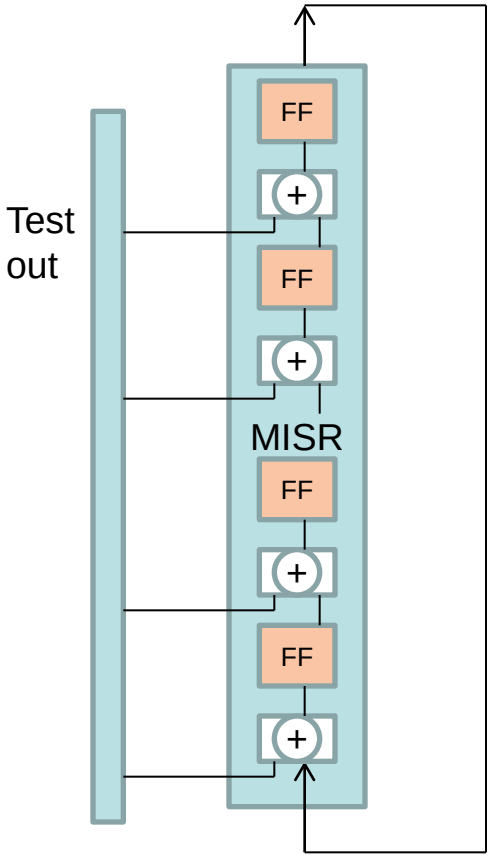
- The circuit model created by „full scan“ architecture is a large combinational logic , which is very wide at inputs / outputs, but not very „deep“ combinationally.
- Most inputs can affect only a small number of outputs. Hence, given the need to create an input vector for testing a specific node or gate, many inputs have a „don´t care (X) status.
- For the transfer of test patterns from tester outputs to „device under test“ outputs, we need to transfer only the input information that really matters. The other inputs may be set „on chip“ by e. g. a random pattern source.
- There are pattern generators such as, for example, feedback shift registers, which can generate large amounts of patterns in a pseudo-random way, but upon demand certain input bits must be „deterministic“.
- For reduced test application time, there may be several (up to 1000 !) scan chains fed and running in parallel.
- Test output compression is also desirable, but that must be done without „don´t care bits“.

Test Architecture Using Multple Parallel Scan Chains



Compression of Test Output Data

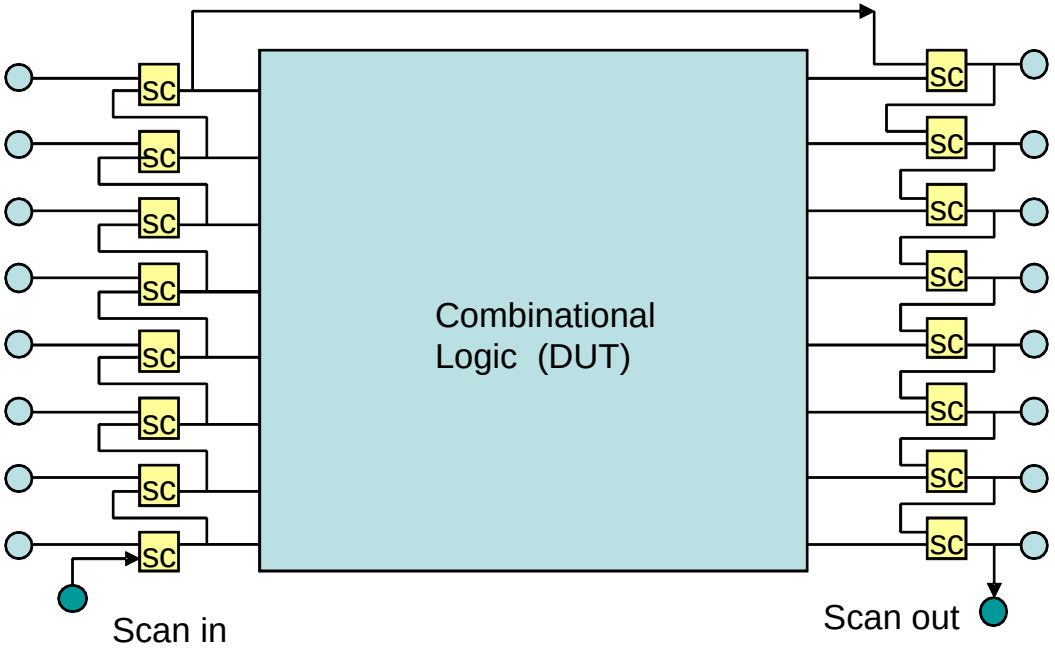




Compaction in space using XORs:
This works only if only 1, 3, 5…test outputs are faulty. ! A single fault that comes in via two outputs is not visible.

The MISR (multi-input shift register) allows for a test output compaction in time ! This scheme works only if all outputs are known for the „good“ case, only then the expected reference content of the MISR can be calculated.

Scan-Test for Dynamic Faults



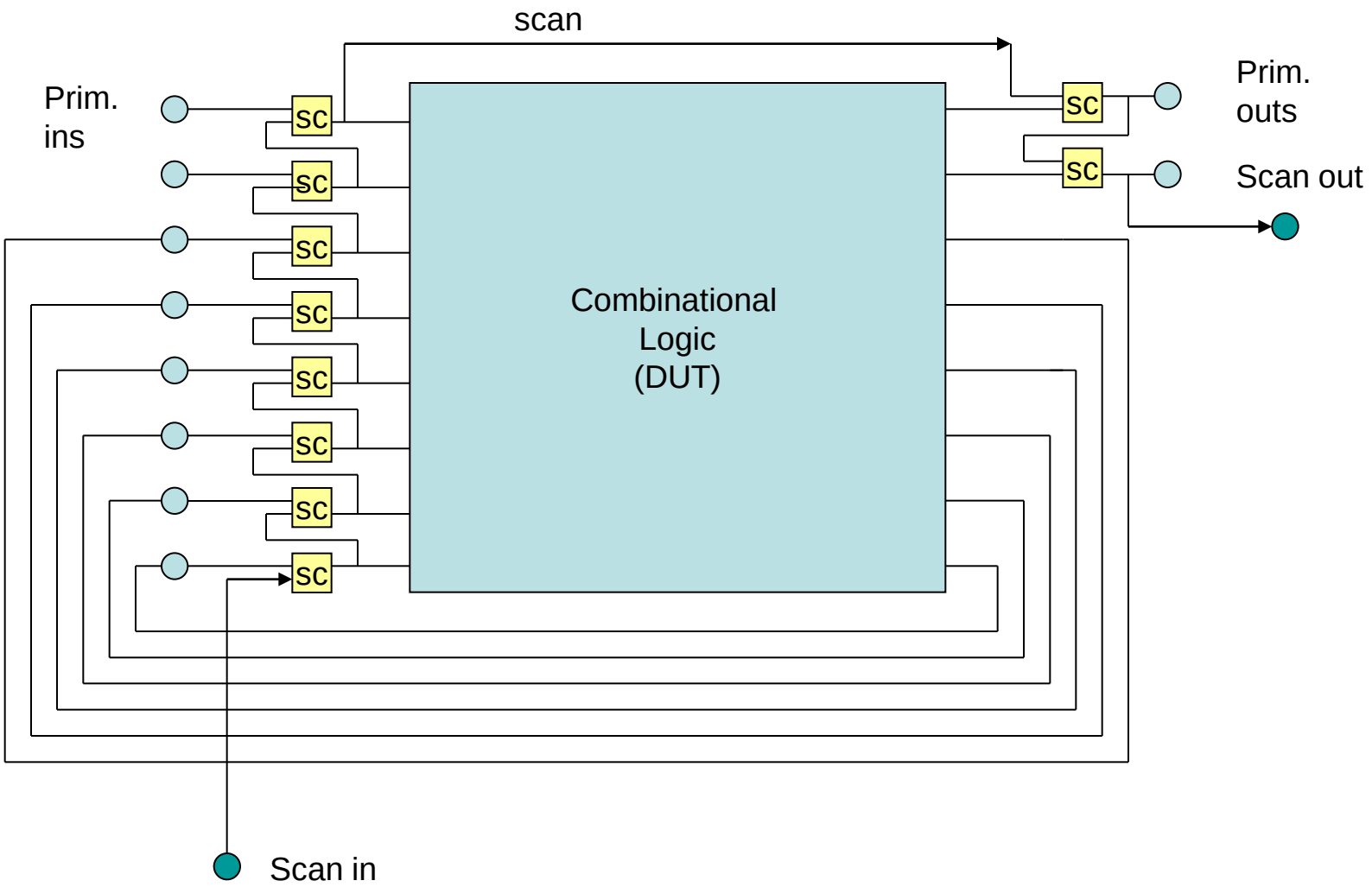
Testind for dynamic faults requires the application of a test sequence. The normal scan structure cannot deliver a sequence.

Double Latched Scan: Requires an enhanced scan architecture, where a scan cell can store two bits to be applied in two clock cycles. Sometimes used, but expensive !

Sequence by scan shift: The second input vector is derived from the first by performing a single scan-clock cycle between 2 functional clocks. **Patterns are sub-optimal** !

Sequence via feed back (Broadside-Scan): **Patterns are not ideal, but mostly useful !**

Broadside-Test



Scan Problems

- Dynamic faults: Test coverage does not reach the „normal“ 99% typicel for scan tests.
- Normal scan path elements change outputs during scan-in / scan-out. The combinational logic is therefore highly active, though outputs are not captured. This may result in excess power consumption already.
- When testing in a scan structure for dynamic faults via feedback, typicall a much higher percentage of gates is switched than in normal operation. This results in high dynamic currents, much over the functional level, and stress on VDD / GND rails. This may result in faults that would not occur in normal operation. **Overtesting !**
- Too much dynamic activity results in excess power consumption and extra heat. Circuits may even be damaged.

⇒ Some companies combine full-scan for static faults with functional test for dynamic faults.

But: Full Scan is an absolute Industrial Standard !!

Selft Test

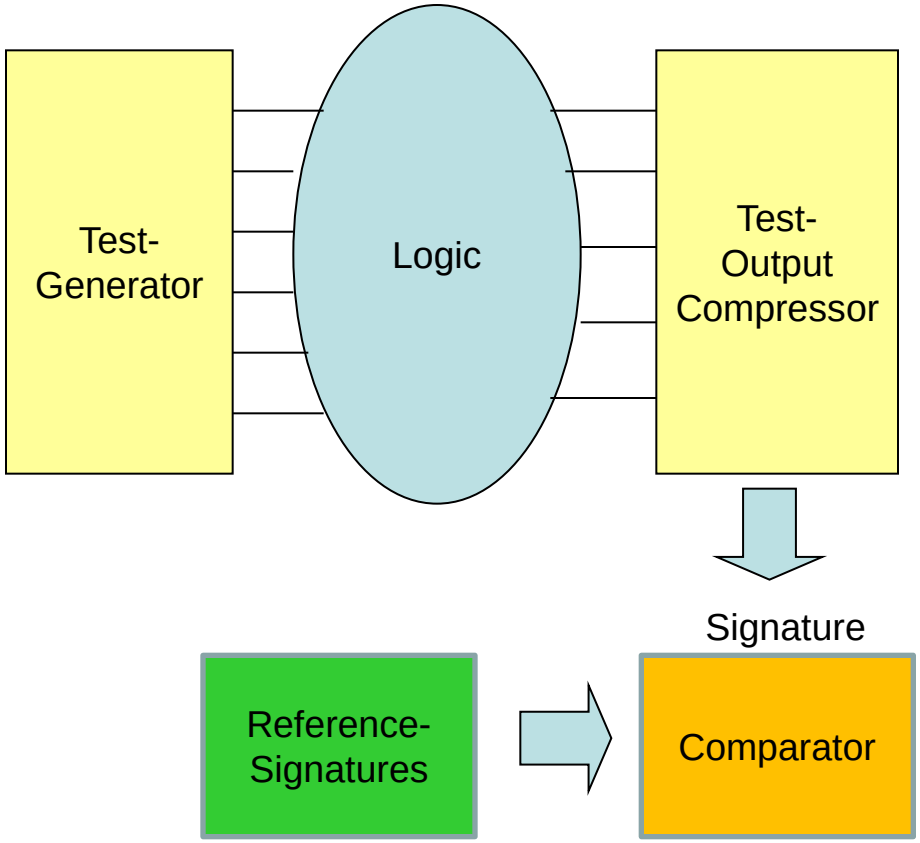
„**Self Test**“ means that test vectors (digital) or test signals (in general) are not applied to the device under test from outside, but either locally generated or taken from a memory. Comparison with reference values then requires either signature generation and comparison with references, or output compaction and external comparison.

Self Test based on stored patterns and references:
Universally applicable to digital circuits, if an „embedded“ processor is there**.** Typically, the volume of test date to be stored is very high.

Self Test based on locally executed test programs:
Software-based self test (SBST) requires an embedded processor, which performs a self test. Additionally, such processor may support the testing of other structures.

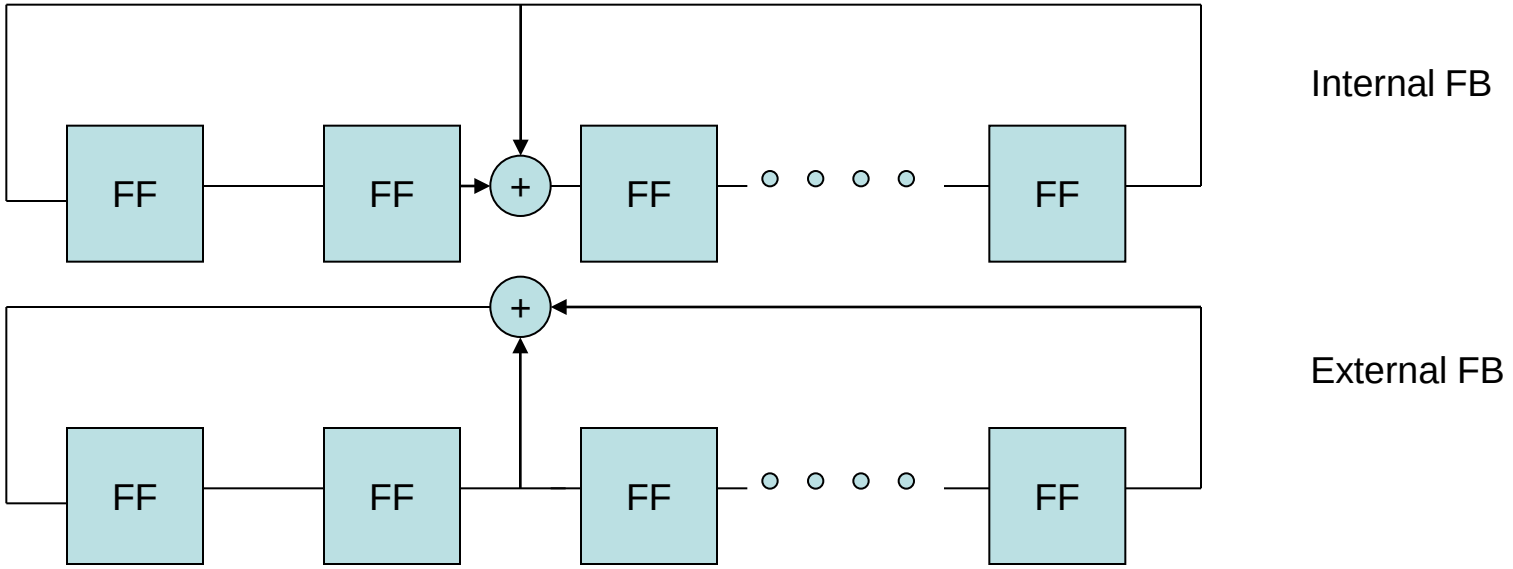
Self Test based on special logic structures for pattern generation and response compaction:
Well-developed schemes are logic built-in self test (L-BIST) and memory self test (M-BIST).

Logic-BIST-Structure



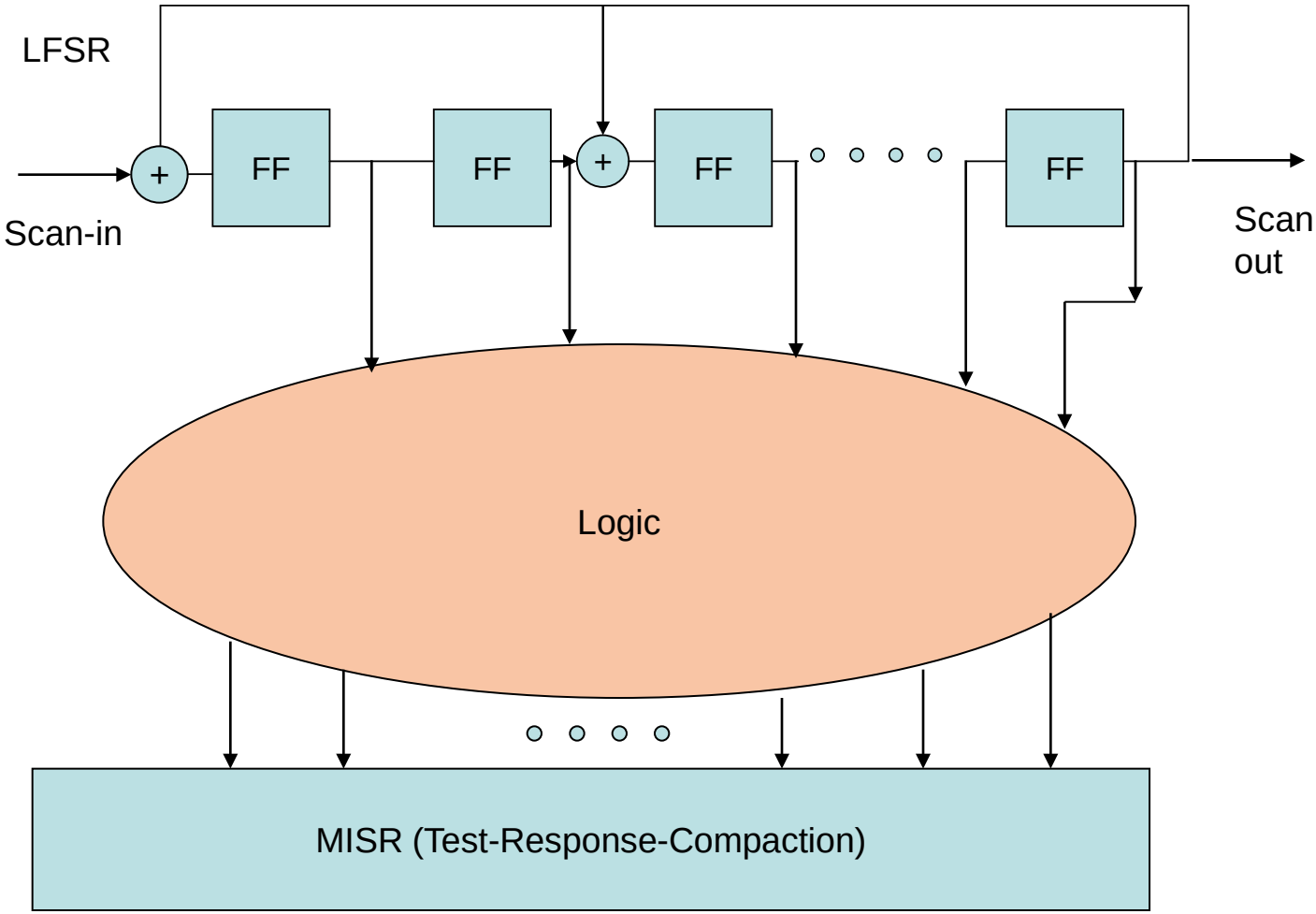
Linear feedback Shift Register (LFSR)

Feedback is done only via XORs to obtain a linear behavior. 0- and 1- values are equally likely to occur. Only the „all bits at 0“ state is not allowed, since it cannot be left.



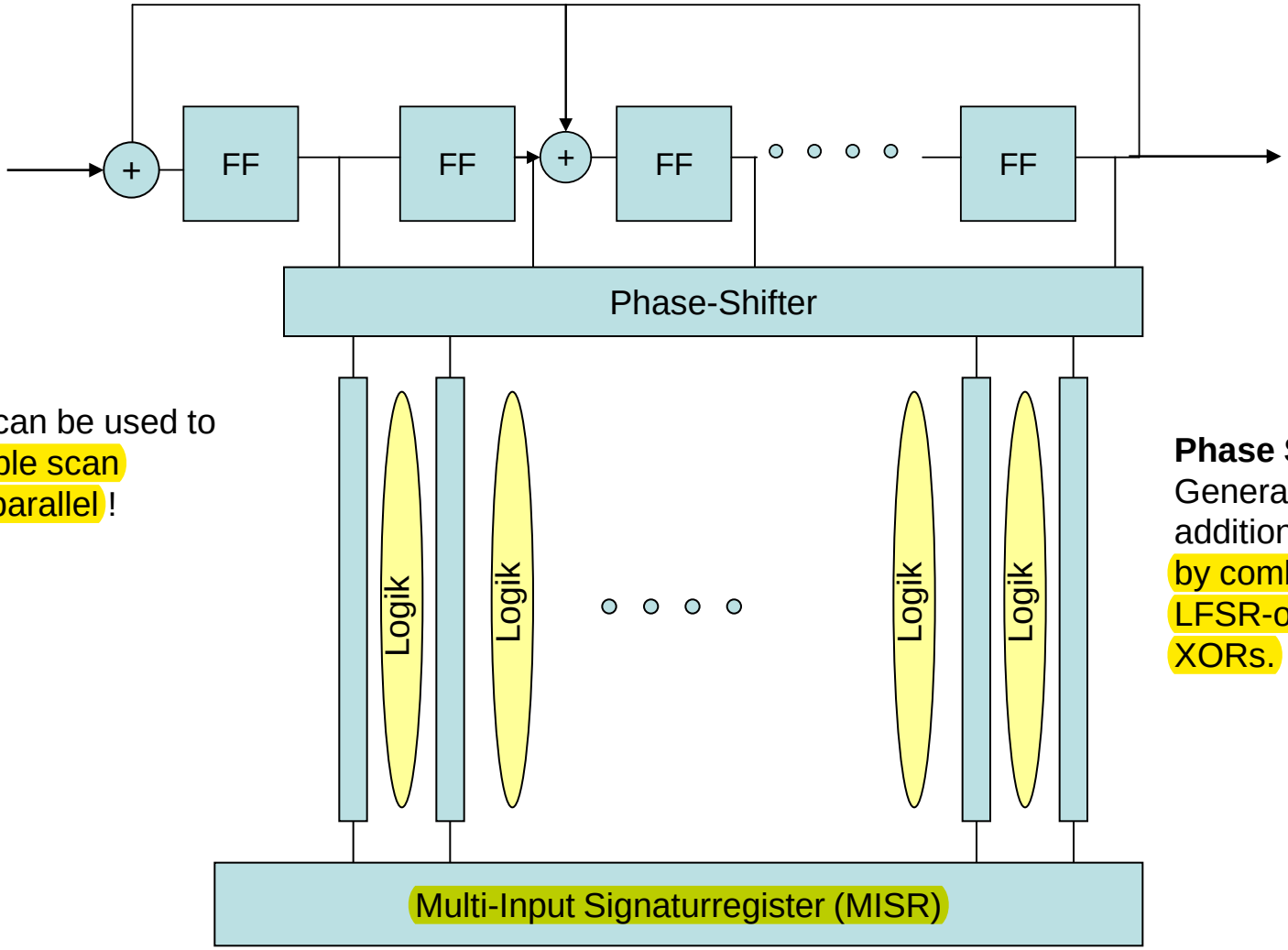
The value of the bit-positions in the register can be described by a polynome. The feedback configuration is described by another polynome, the generator polynome. The content of the shift register is obtained by deviding The „content“ polynome be the „generator“ polynome. The result is a sequence of patterns, which is typically a cyclic one. The length of a sequence for an n-bit register is 2^n –1 at maximum. Patterns generated this way „random“, but can be calculated in advance.

Test-per-Clock



Initalization of the LFSR via scan-path !

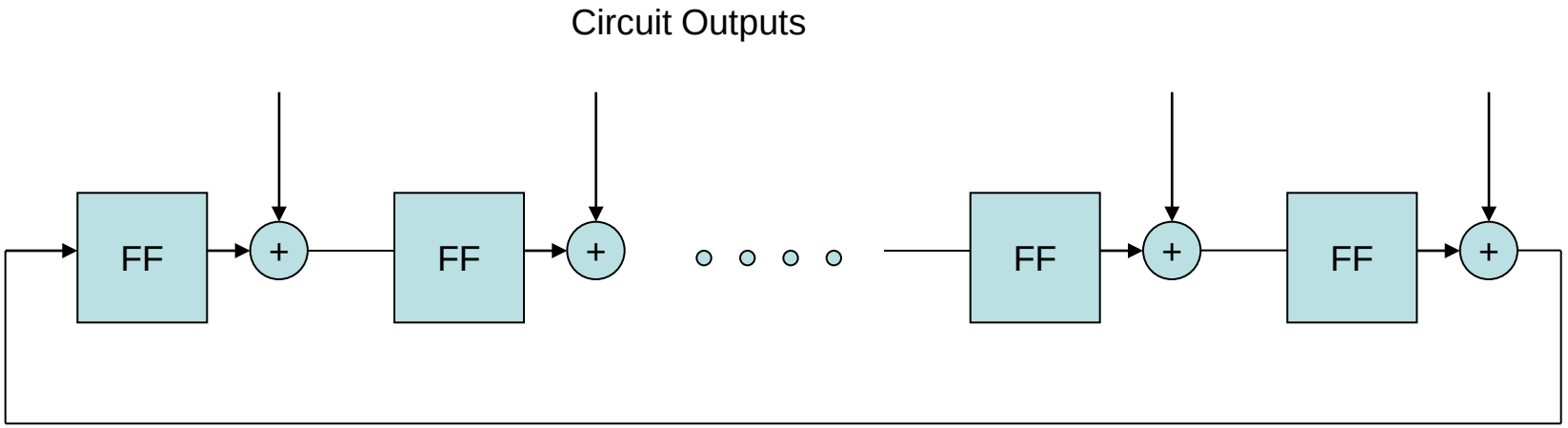
Test-per-Scan



An LFSR can be used to feed multiple scan chains in parallel!

Phase Shifter: Generates new additional outs by combining LFSR-outputs via XORs.

Multi-Input-Signature Register (MISR)

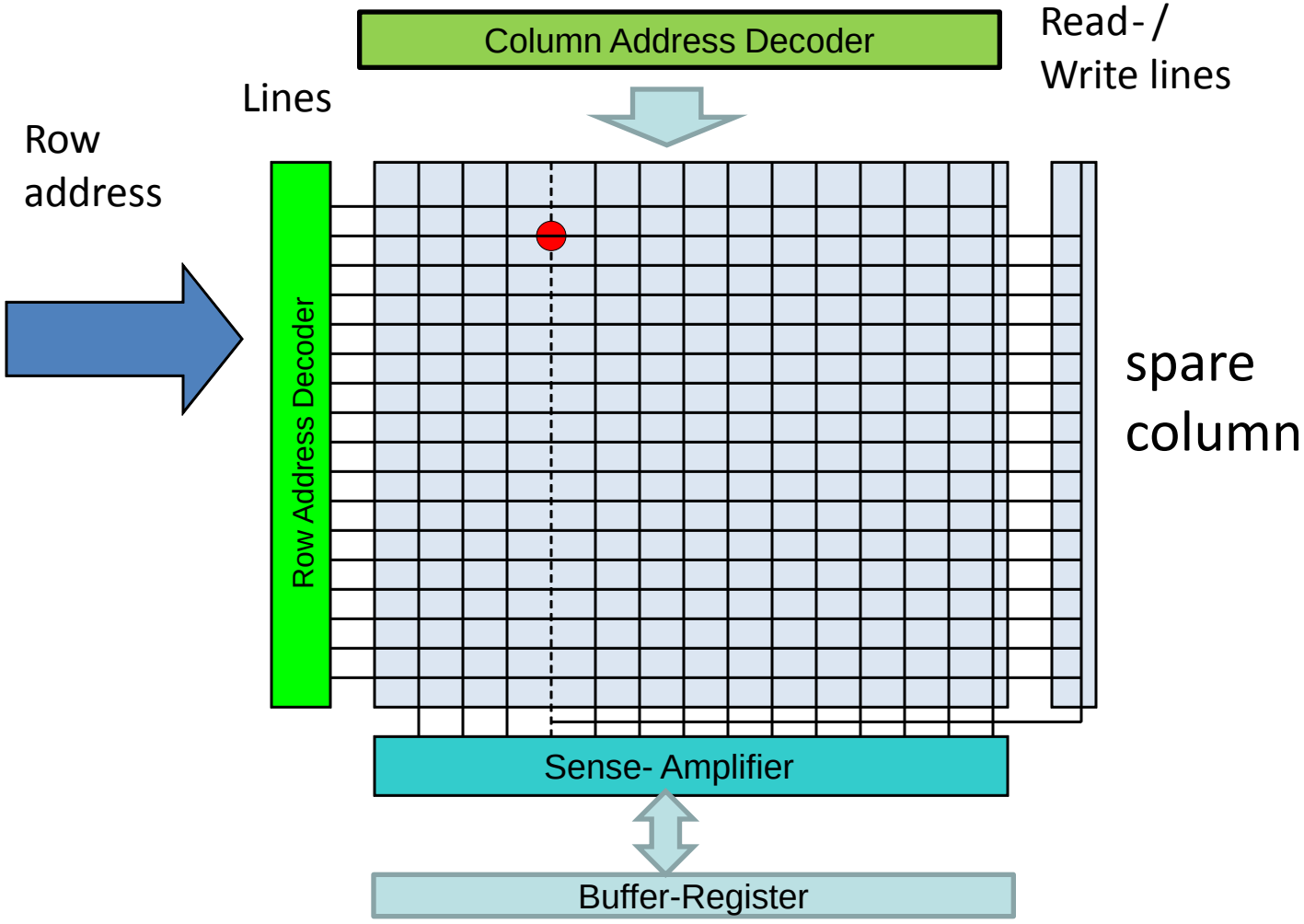


Depending on the initial content of the register and the circuit outputs (have to be known for the good case), the MISR will contain a kown pattern (again obtainable by polynome divison) after a certain number of clock cycles. Any fault bit on the circuit output will modify the content. For a single fault event it is even possible to perform a fault diagnosis (which bit in which clock cycle).

If the length of the MISR is small, there is a chance that two fault may mask each other. This probability falls exponentially with MISR length.

Problem: MISRs cannot compact circuit outputs with unknown („X“) values !

Memory Structure



Memory faults: Adress-Decoder, memory cells, Sense- Amplifier, output register

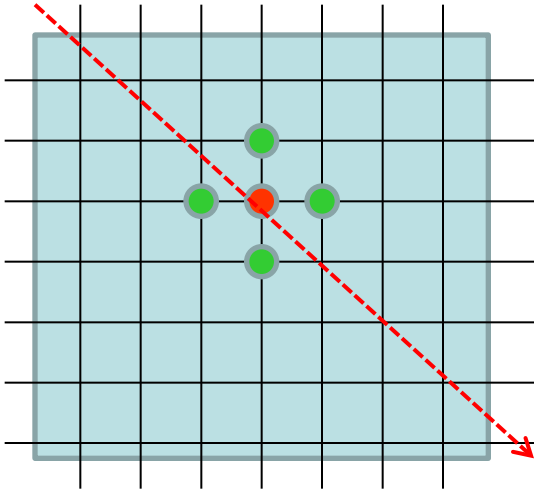
Memory Testing

- **Tests of Address-Decoding: I. e. a „stuck-at-0“ on a decoder-bit.**
  Take the address, possibly false bit set to „0“. Write a „0“. Then take the address with the possibly faulty address bit set to „1“. Write „1“ to this cell. Read both cells and compare !

- **Tests for the memory cells:**
  - Bit stuck-at 0 / 1
  - Memory cell is affected by reading / writing neighboring cells (coupling fault)
  - Memory cell does not keep information (data retention fault), mainly in dynamic RAMs

March-Tests:

There is a regular pattern „marches“ through the structure on a diagonal path, checking for stuck-at and for coupling faults.



Example:
Memory cell x/ y written to „1“,
all neighbors to „0“ and read again.
Memory cell x / y written to „0“,
all neighbors to „1“ and read again.

Memory test is very regular and can easily be done by a hardware-based BIST function !