TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Institut für Computertechnik
Institute of Computer Technology

Design for Testability

Introduction to testing integrated circuit designs

Michael Rathmair

2 lectures about testing circuits

- PART 1: **What to test**
  - Why is testing so important
  - Which faults can happen
  - Generation of test patterns

20.11.

- PART 2: **How to test**
  - How to test a design
  - How good is my test
  - How to test hardware

27.11.

Motivation

Design Phase	Error (mechanism), failure, bug, improper definition, ...	Approx. cost
Specification	Functionality, Performance, Testability, conflicting definitions, ..	1 €
Modeling, Design	Interoperability, Functionality, Performance,	1000 €
Prototype	Verification errors, Performance, ...	100 k€

Impl. Phase	Error (mechanism), failure, bug, improper implementation, ...	Approx. cost
Wafer	Yield, Silicon properties, Dust, Speed, Noise, ...	1 €
Chip	Packing, Bonding, ...	10 €
Module	Soldering, ESD impact, ...	100 €
Subsystem	Cable connections, Connectors, ...	1000 €
After deployment	Stress, Vibration, Reliability, ...	10 k€

- Example: Intel FDIV bug (1994)
  - Logic error not caught until > 1M units shipped
  - Recall cost $450M

Motivation

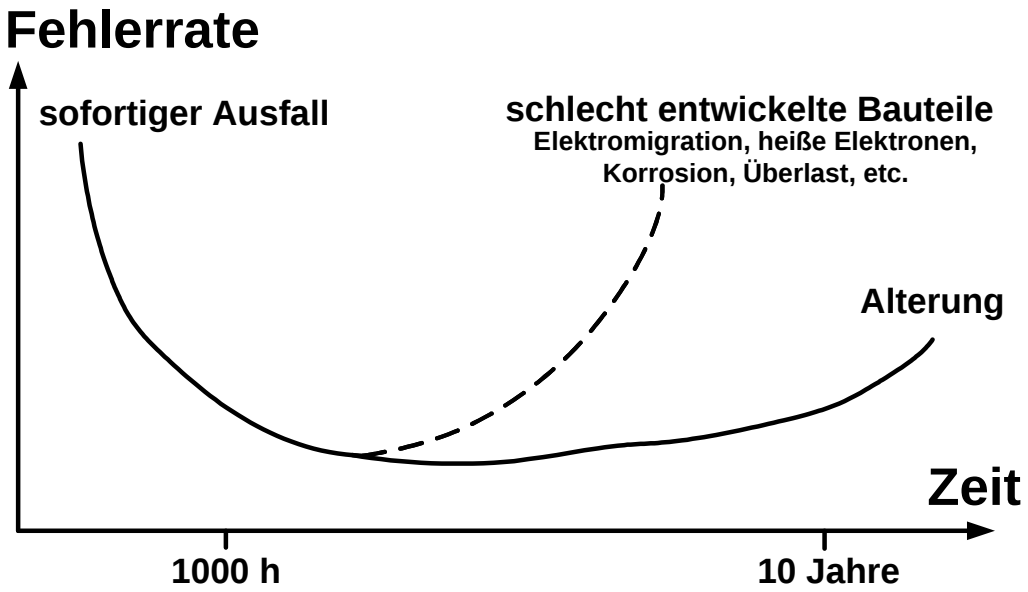
- **Failure rate** for a full system
  - Probability for a system failure

$$p(System|faulty) = 1 - p(System|ok) = 1 - \prod_{i=1}^{N} p_i(component|ok)$$

  - 70 components 99% OK → Full system is OK for 50%
  - 70 components 90% OK → Full system is OK for 0%

Motivation

- **Reliability** of integrated circuits
  - “Bathtub” characteristic



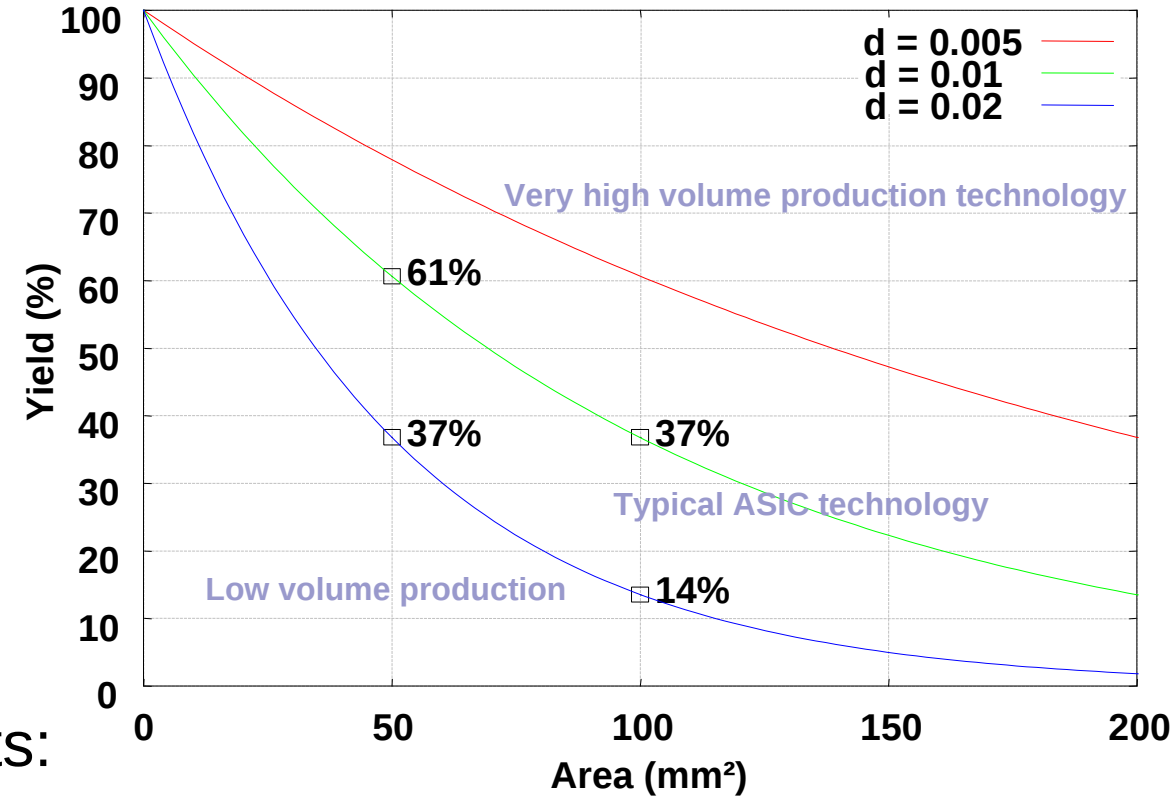
  - Number of components which fails within 1000 operation hours: 1...5 %
  - Burn-In: Operating the ICs at 125° shortens this 1000 hours interval to about 24 hours.

Motivation

- Yield:

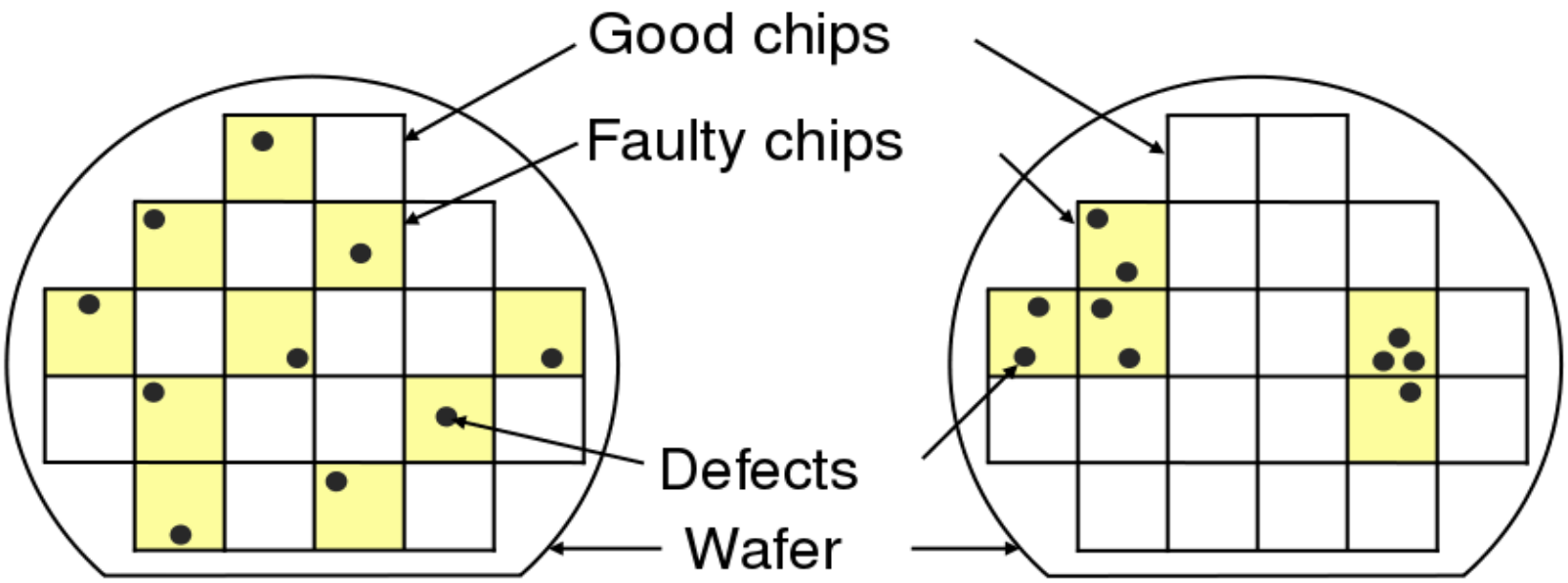
$$100 \cdot e^{-d \cdot A}$$

d=defects/mm²
typically d =
0.005 … 0.02



- Yield improvements:
  - Technology improvements
  - Redundancy (e.g. memory)
- Cost: 100mm² → 50mm² chip with yields from figure
  - 100mm²/50mm² x 0.61/0.37 = 3.4 (d=0.01)
  - 100mm²/50mm² x 0.37/0.14 = 5.4 (d=0.02)

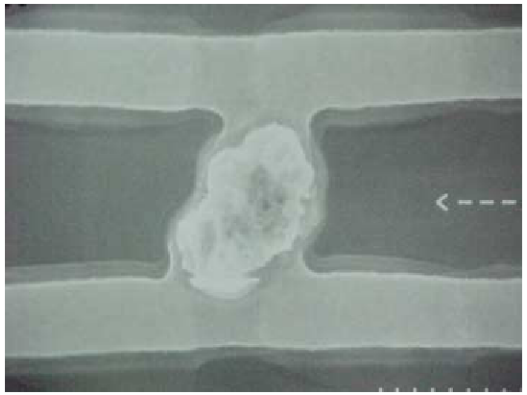
Chip Yield



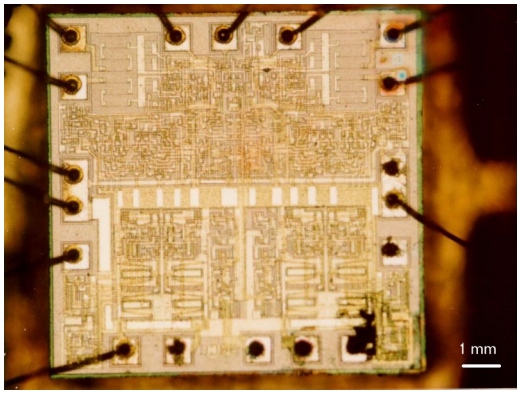
Wafer yield = 12/22 = 0.55

Wafer yield = 17/22 = 0.77

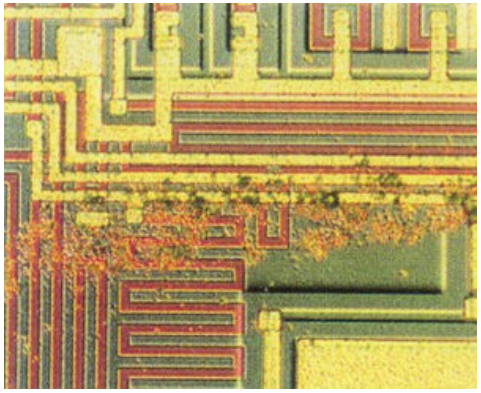
Defects



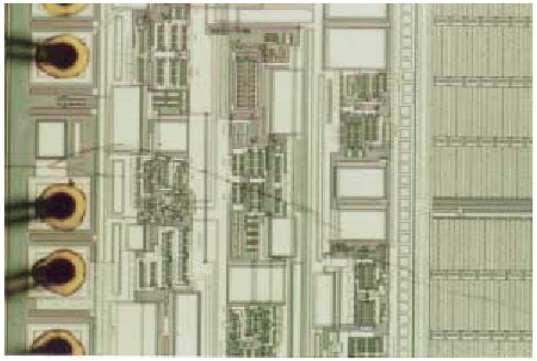
1) Kurzschluß zwischen Leiterbahnen durch Staub



2) Elektrischer Durchschlag (rechts unten)

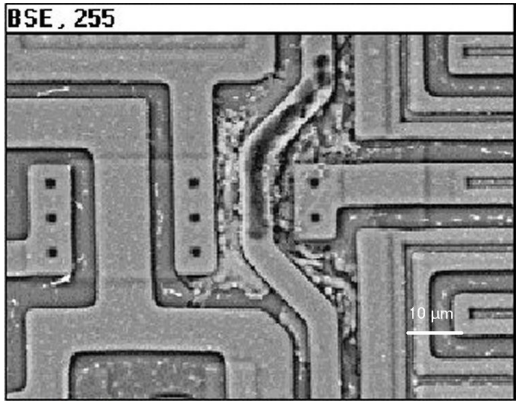


3) Verunreinigungen des Wafers (Diffundieren ins Polysilizium)



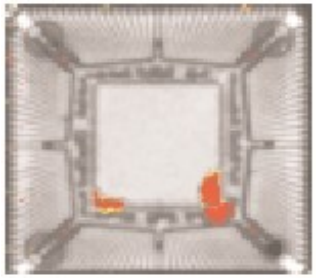
4) gebrochenes Die (mechanische Verspannungen)

www.sony.co.jp/en/Products/SC-HP/PDF



5) überlastete Leiterbahn

www.era.co.uk/product

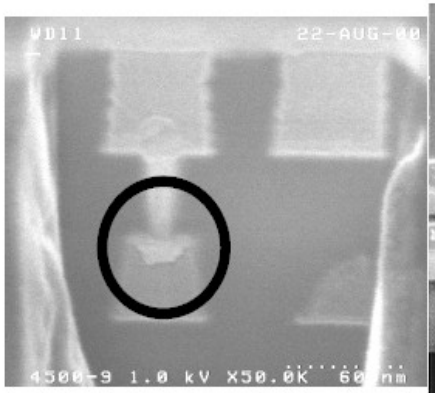


6) ins Gehäuse eingedrungener Schmutz (sonographisches Bild)

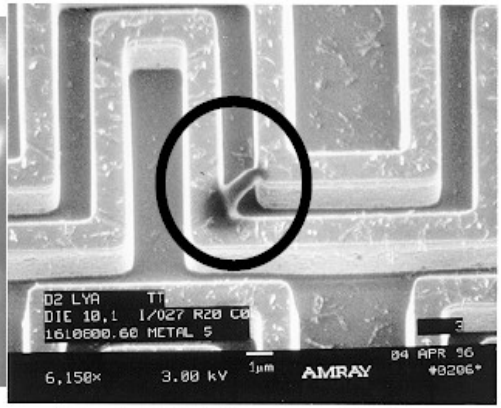
www.fabtech.org/features/tap/articles

www.empf.org/html

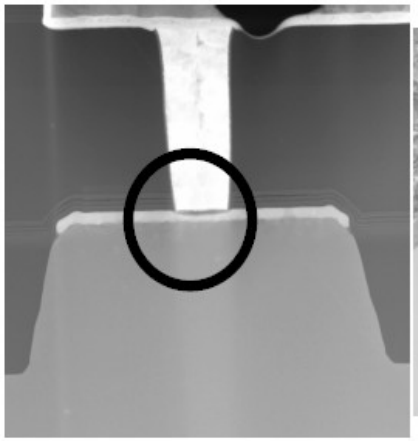
Defects



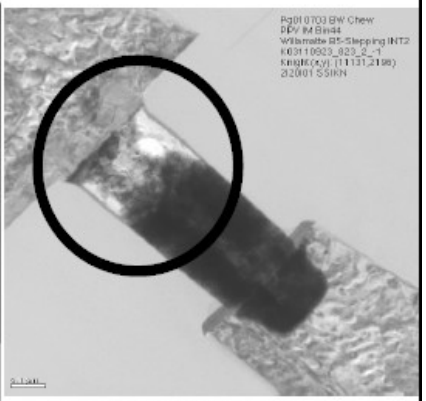
Metal 1 Shelving



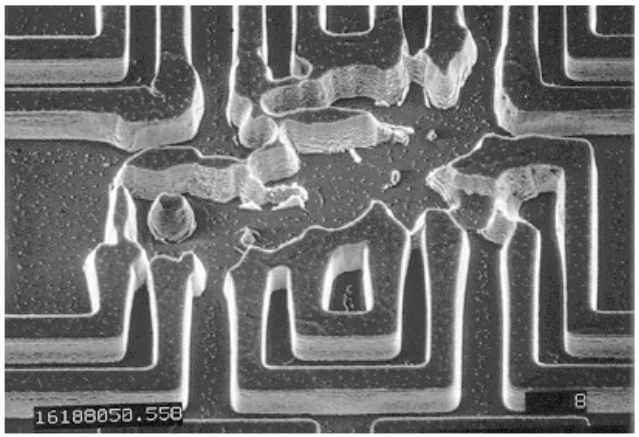
Metal 5 film particle
(bridging defect)



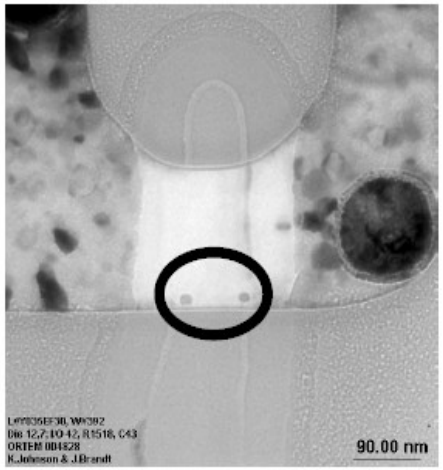
Open defect



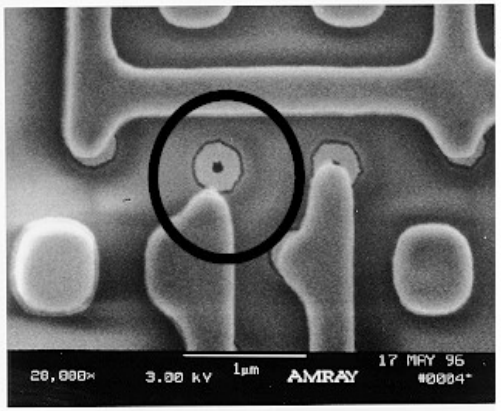
Spongy Via2
(Infant mortality)



Metal 5 blocked etch
(patterning defect)



Spot defects
“Co” Defect under Gate



Metal 1 missing pattern
(open at contact)

SEM images courtesy Intel Corporation

Terminology

- Defect:
  - Is an unintended difference between the implemented hardware and the design.
  - Defects may be caused by disturbances during manufacturing or usage of the device.
- Fault:
  - The impact of a hardware defect to an abstracted functional level
  - Fault indicates an event
- Error:
  - Wrong (unspecified) behavior of the defective system
  - An error is caused by a fault
  - Error indicates a state.

Terminology

- **Controlability**

  Effort for reaching a defined state in the system. This might be the initial state for a testsequence. Ease of forcing the output of a node to 0 or 1 by driving the inputs.

- **Observability**

  The ability to observe the state or logic values if internal nodes. Effort for observing effects caused by a test sequence

  Combinatorial logic is usually easy to observe and control. FSM can be very difficult, requiring many cycles to reach a desired initial test state.

- **Testability**
  - Ability to generate test sequences for a dedicated focus and/or to reach a required test coverage.
  - Localization of faults. A test must be designed that the cause (component, signal, etc.) of a fault can be identified.
  - Testability = Controlability + Observability

Terminology

- Fault Coverage

$$FC = \frac{N\left(observable\ \ System\ errors\right)}{N\left(\ possible\ \ System\ errors\right)} \cdot 100$$

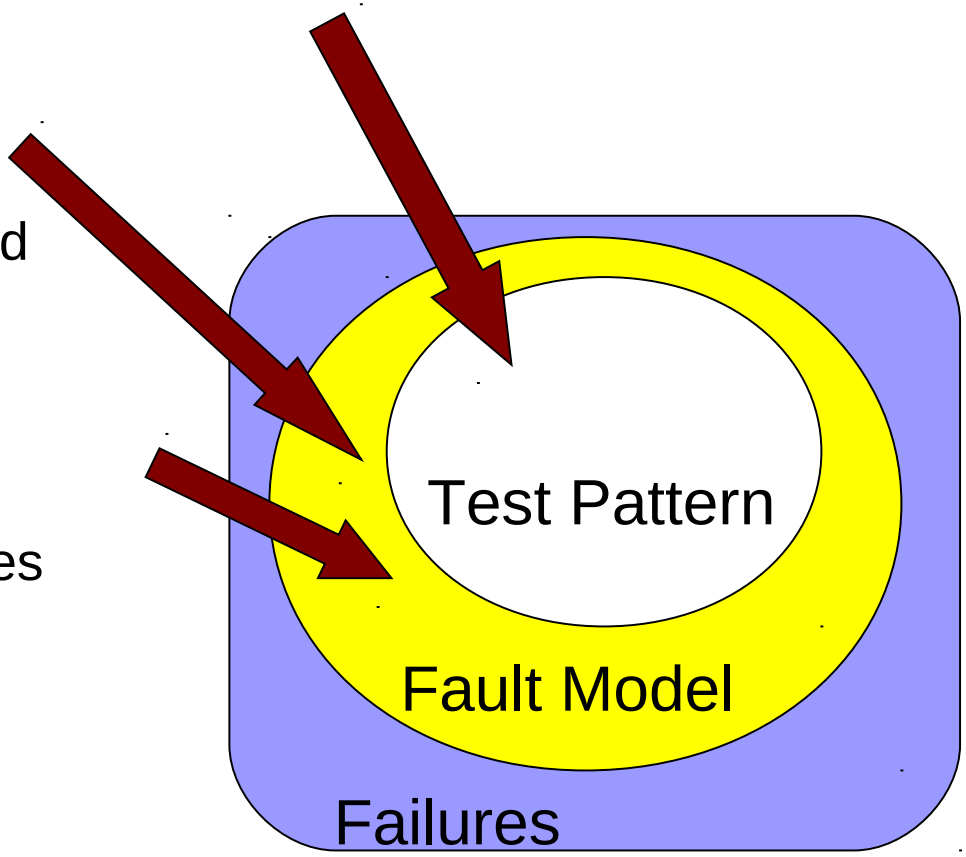
- Fault Model:

  Hypothesis on which defects can occur at a used process technology

- Fault Model +

  System structure:

  Results in all possible cases a system can reach malformed behavior.

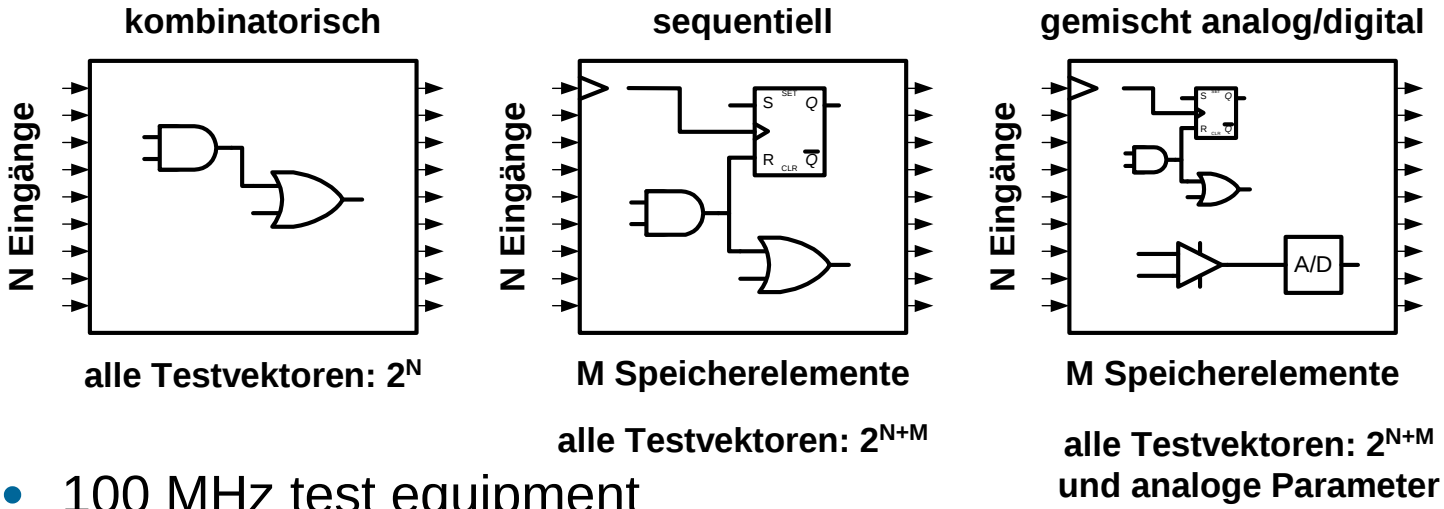


Overview

- Testing is a very important phase of the development and design of a new IC
- Testing is cost intensive and complex
  - Cost for testing is about **2/3 of total ASIC development cost**
- Testing might be a bottleneck for IC fabrication
- Test time is cost intensive and must be reduced
  - Trade-off between coverage (number of testcases) and test time.
- Test-effort
  - Sometimes complex and expensive test equipment is required (analog testing, RF testing, …)
  - External test equipment for digital functionalities
    100 k€ to 10 M€; 32k – 100M testvectors
  - On-chip test equipment → chip area

Overview

- What is being tested?



- 100 MHz test equipment
  - N=32 → testtime = 40 seconds
  - N=63 → testtime = 6000 years
- Topology of the DUT (device under test) must be used to reduce the number of testcases.
- Synchronization of analog and digital Signals

Testability Measures

- TMEAS [Stephenson & Grason, 1976]
- SCOAP [Goldstein, 1979]
- TESTSCREEN [Kovijanic 1979]
- …

- Importance of testability measures
  - Guide designers to improve testability in their designs
  - Test generation algorithms using heuristics apply testability measures to speed up the test generation process (e.g. in search decisions)

SCOAP

- Sandia Controllability Observability Analysis Program
- Using integers to reflect the difficulty of controlling and observing the internal nodes.
- Higher numbers indicate more difficult to control or observe.
- Applicable to both combinational & sequential circuits.
- SCOAP measurements on each line
  - Combinational 0-controlability, CC0(l)
  - Combinational 1-controlability, CC1(l)
  - Combinational observability, CO (l)
  - Sequential 0-controlability, SC0(l)
  - Sequential 1-controlability, SC1(l)
  - Sequential observability, SO(l)

SCOAP

- Combinational measures related to number of signals to be manipulated to control or observe 1
- Sequential measures indicating number of clock cycles needed to control or observe signals on line 1
- Controlability ranging from 1 to inf
- Observability ranging from 0 to inf
- High measures indicating difficulties with controlling or observing the given line

SCOAP – combinatorial Controlability

- Computed at propagating from inputs to outputs

- Step1: For all primary inputs set all CC0= 1 and all CC1 = 1
- Step 2: Traverse in level order (Level of logic gate: max distance of its logic inputs) through circuit towards primary outputs updating controlability measures
- Step 3: For each traversed logic gate:
  - If logic output produced by setting only one input to controlling value then

    output controllability = min(input controllability)+1
  - If logic output only obtained by setting all inputs to non controlling values then

    output controllability = sum(input controllabilities)+1
  - If possible to control output by multiple input sets (XOR: “01” or “10” cause output 1) then

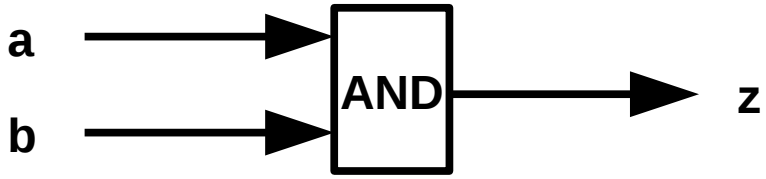
    output controllability = min(controllabilites of input sets)+1

SCOAP – combinatorial Observability

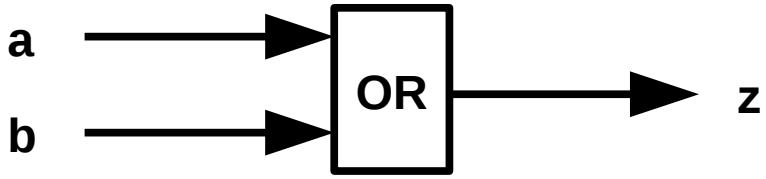
- Computed at propagating from outputs to inputs

- Step1: For all primary outputs set all CO= 0
- Step 2: Traverse through circuit towards primary inputs updating observability measures
- Step 3: For each traversed logic gate:
  - **CO = observability of output +**
    difficulty in setting all inputs to non-controlling values +
    1 to accommodate for logic depth
  - No distinction between 0 and 1 observability

SCOAP - Examples

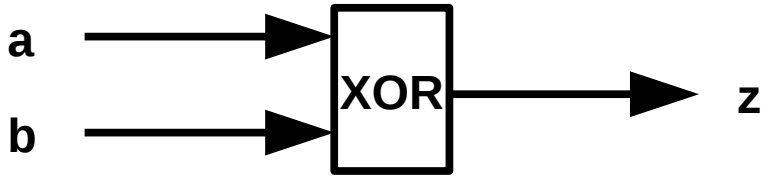


- CC0(z) = min(CC0(a), CC0(b)) + 1
- CC1(z) = CC1(a) + CC1(b) + 1
- CO(a) = CO(z) + CC1(b) + 1
- CO(b) = CO(z) + CC1(a) + 1

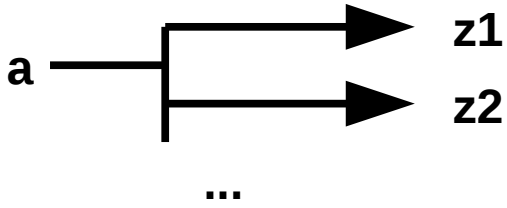


- CC0(z) = CC0(a) + CC0(b) + 1
- CC1(z) = min(CC1(a), CC1(b)) + 1
- CO(a) = CO(z) + CC0(b) + 1
- CO(b) = CO(z) + CC0(a) + 1

SCOAP - Examples

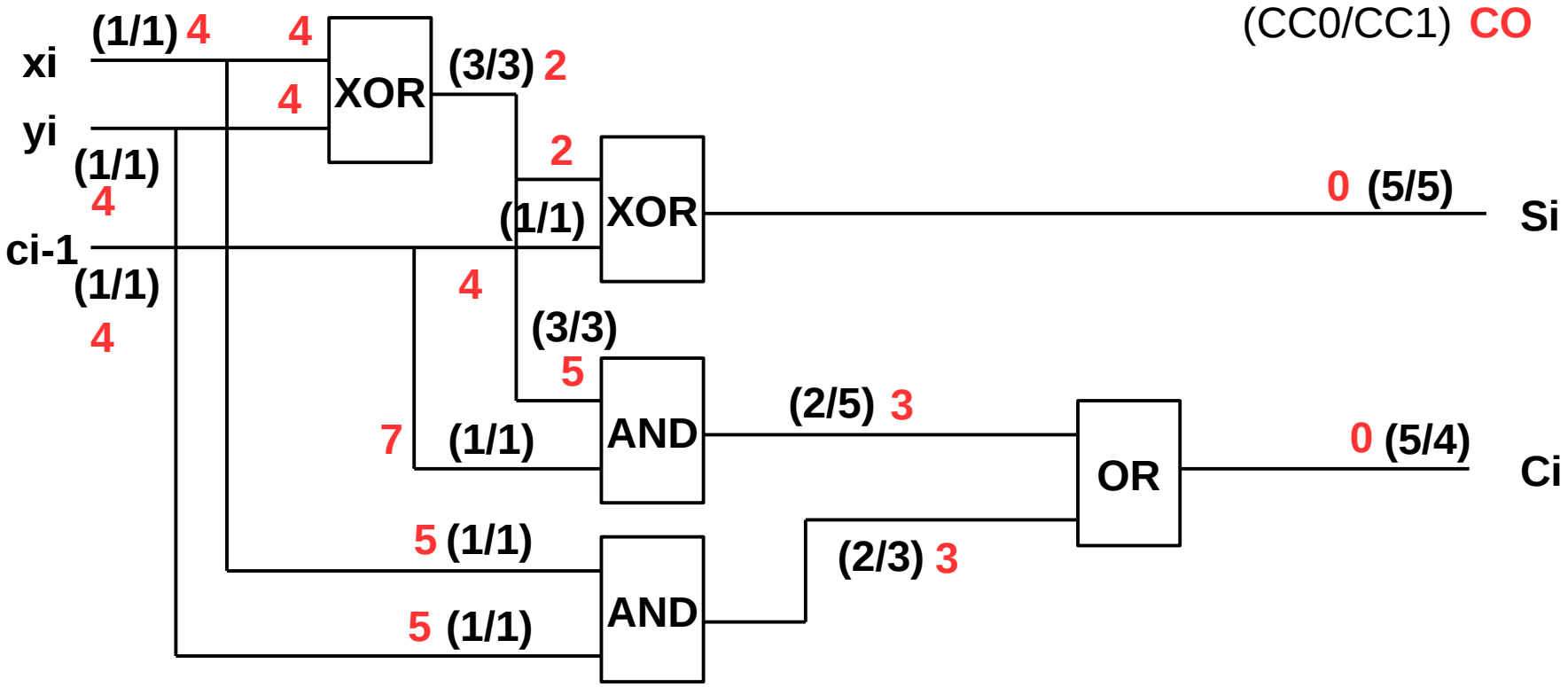


- CC0(z) = min(CC0(a)+CC0(b), CC1(a)+CC1(b)) + 1
- CC1(z) = min(CC1(a)+CC0(b), CC0(a)+CC1(b)) + 1
- CO(a) = CO(z) + min(CC0(b), CC1(b)) + 1
- CO(b) = CO(z) + min(CC0(a), CC1(a)) + 1



- CO(a) = min(CO(z1), CO(z2), ...)

SCOAP – Example Adder



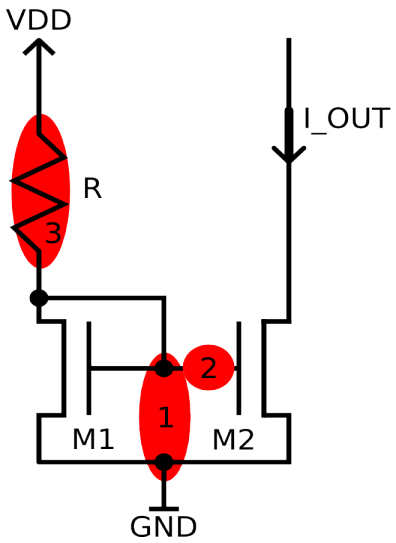
- CC0(si) = 5 CC1(si) = 5 CO(si) = 0
- CC0(ci) = 5 CC1(ci) = 4 CO(ci) = 0
- CC0(xi) = 1 CC1(xi) = 1 CO(xi) = 4
- CC0(yi) = 1 CC1(yi) = 1 CO(yi) = 4
- CC0(ci-1) = 1 CC1(ci-1) = 1 CO(ci-1) = 4

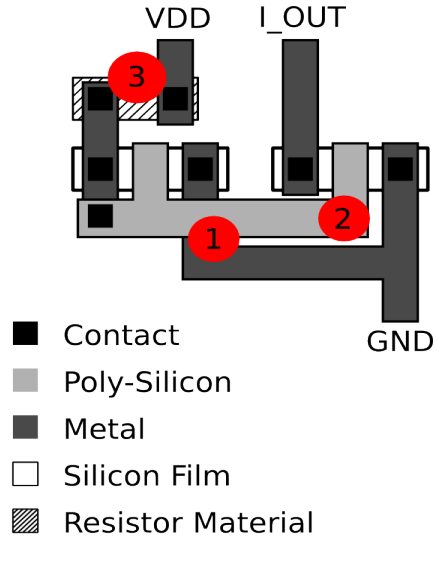
Basic Models for defects

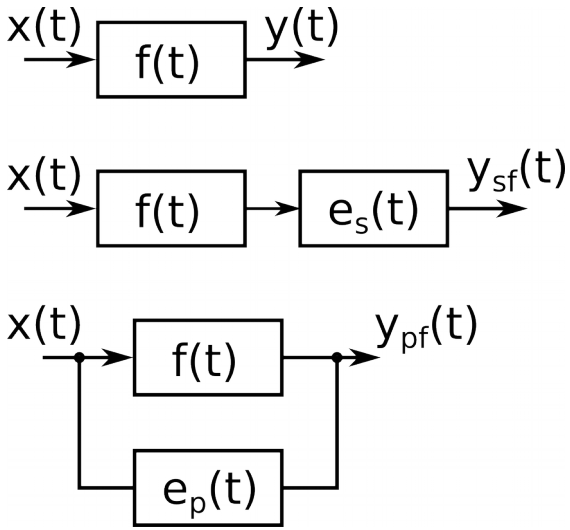
- **Hard Faults** (Catastrophic faults)
  - Large deviations of the designed system behavior. May be electrical characteristics, structural deviations, mechanical faults, etc.
  - Stuck at short: Unspecified electrical connections inside a component, at component interconnects, etc.
  - Stuck at open: Unspecified disconnection.
  - Punch through: Unspecified electrical connection caused by a high voltage impact.
- **Soft faults** (Deviations inside a specified tolerance)
  - Variability in electical, mechanical, structural characteristics. Process fluctuations
  - Silicon (material) characteristics: Carrier mobility, doping concentrations, etc.

Fault Models

- Layout level fault model
  - Schematic as a unique model – layout as a realization
- Behavioral-Functional fault model
  - Serial and parallel fault modeling
- Parametric fault model
  - Deviation modeling – range based analysis of the system

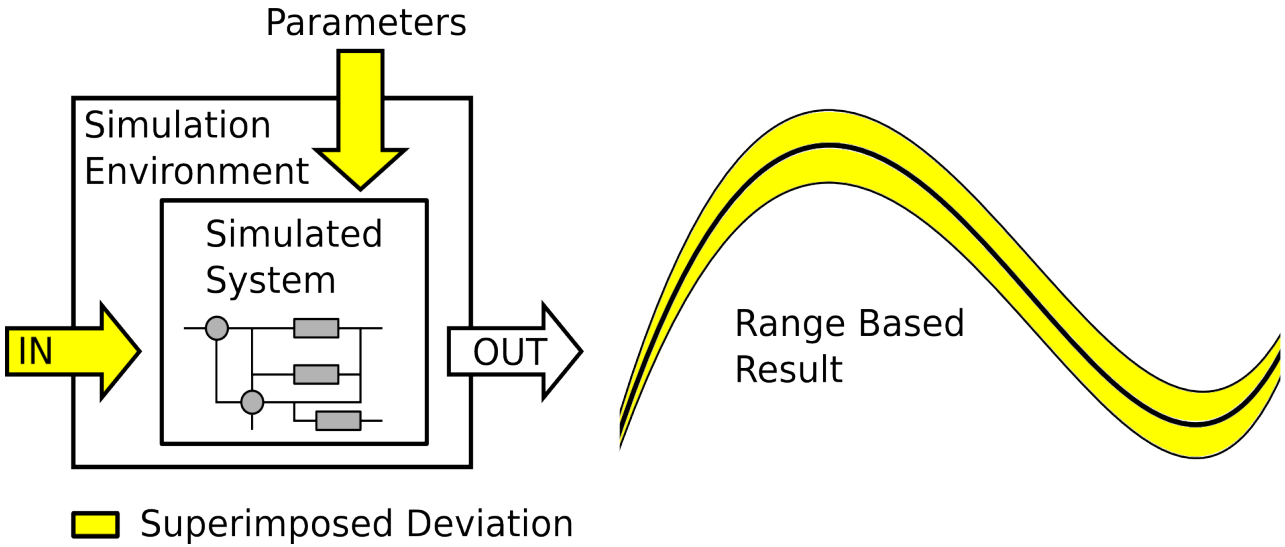






Fault Models

- Layout level fault model
  - Schematic as a unique model – layout as a realization
- Behavioral-Functional fault model
  - Serial and parallel fault modeling
- Parametric fault model
  - Deviation modeling – range based analysis of the system



Fault Types

- Permanent faults

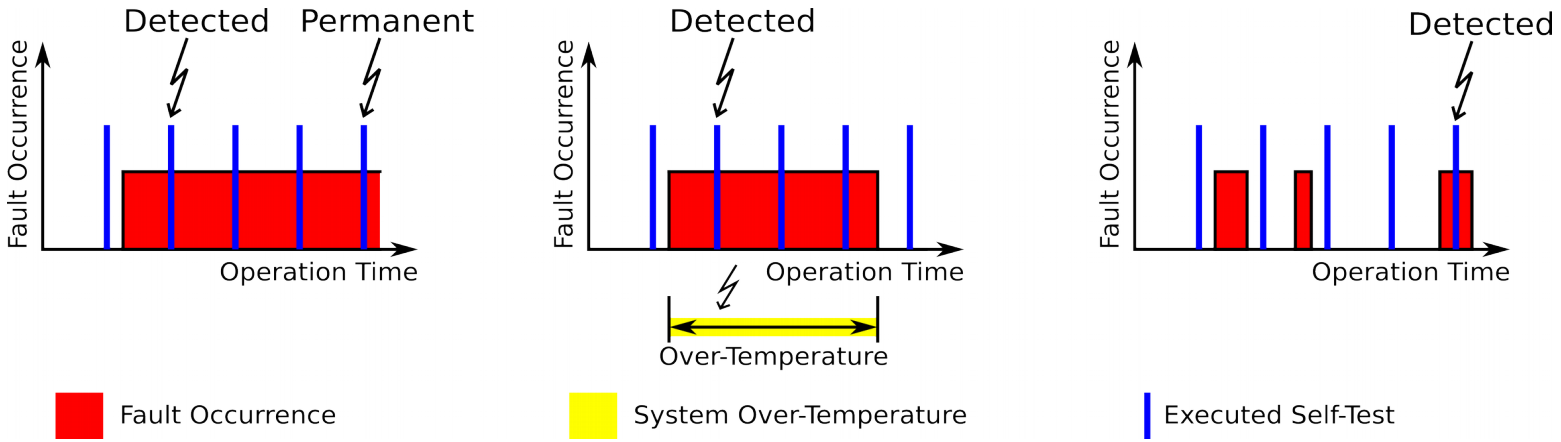
  Errors at the production or during operation. Can be detected by cyclic self tests

- Intermittent faults

  Correlated with environmental conditions. Additional measurement of system parameters to differ from a permanent fault

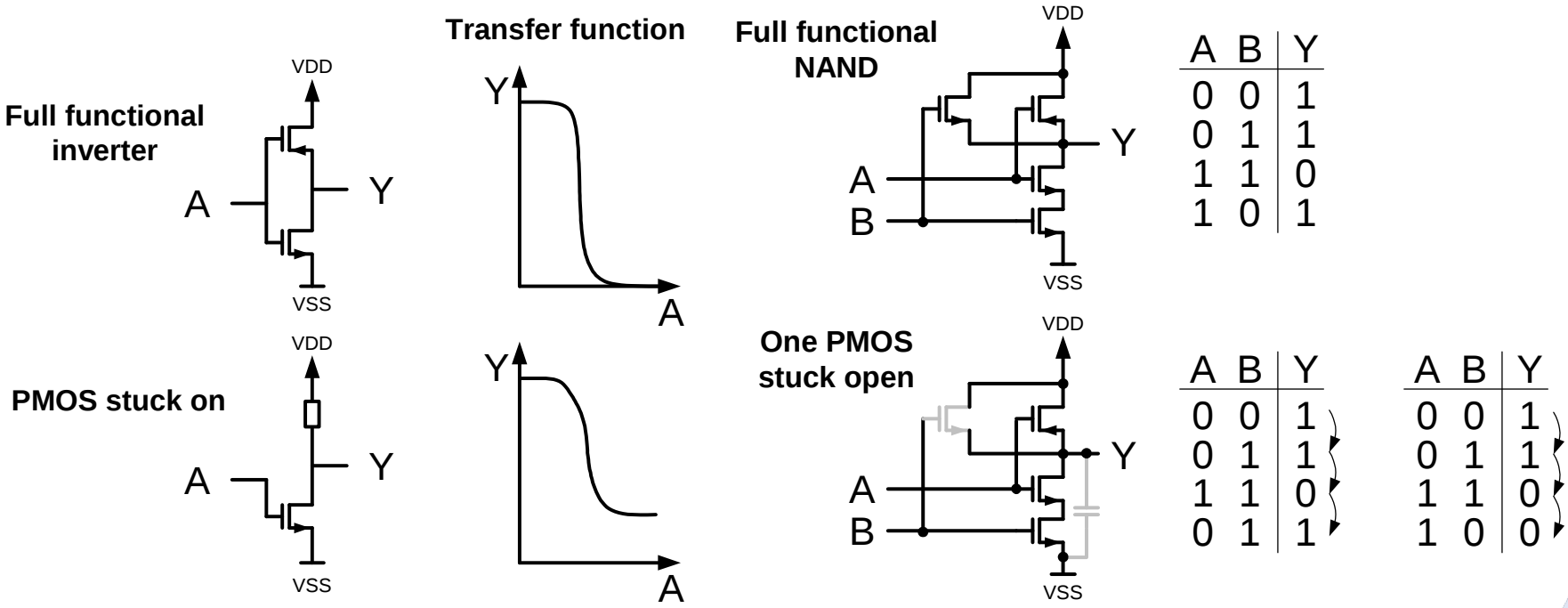
- Transient faults

  completely random manner. Hard to detect if fault time is shorter than self test period.



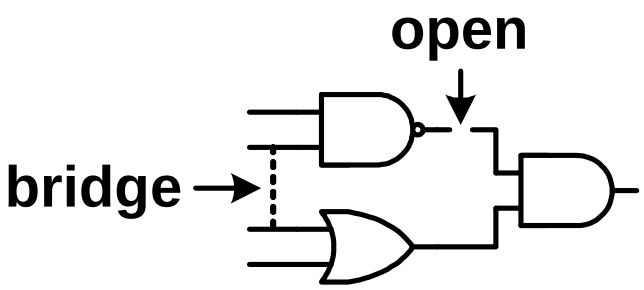
CMOS stuck at faults

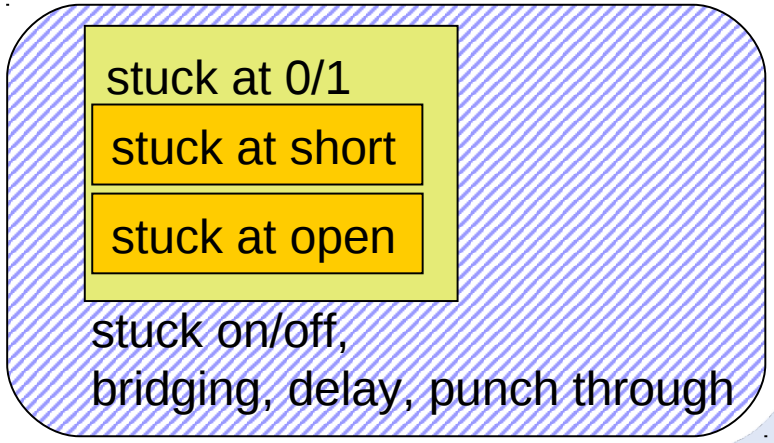
- It is assumed that a fault in a logic gate results in one of its inputs or the output is fixed to either logic 0 or 1
- A single line in the structure is connected to 0 or 1



CMOS stuck at faults

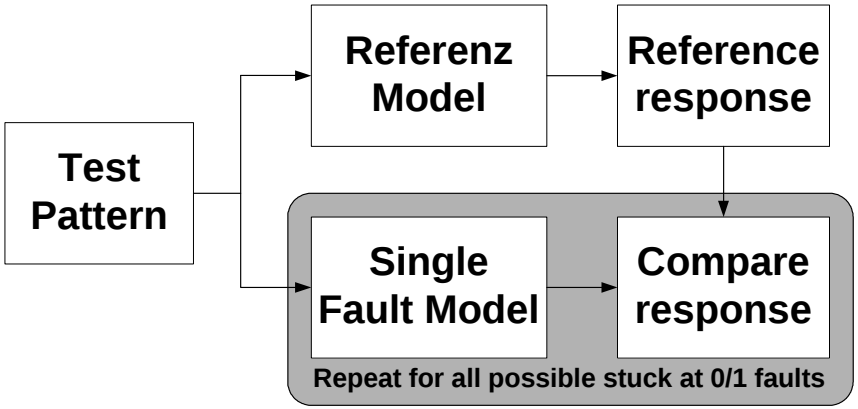
- Complexity compared to a parametric fault model is reduced.
- For each component (gate) in the netlist connections to 0 and 1 are considered.
- Reduced effort for fault simulations and achieving a requested coverage.
- For VLSI designs sometimes other parametric fault models are too computation intensive.
- Covers 80-90 % of classical defects.

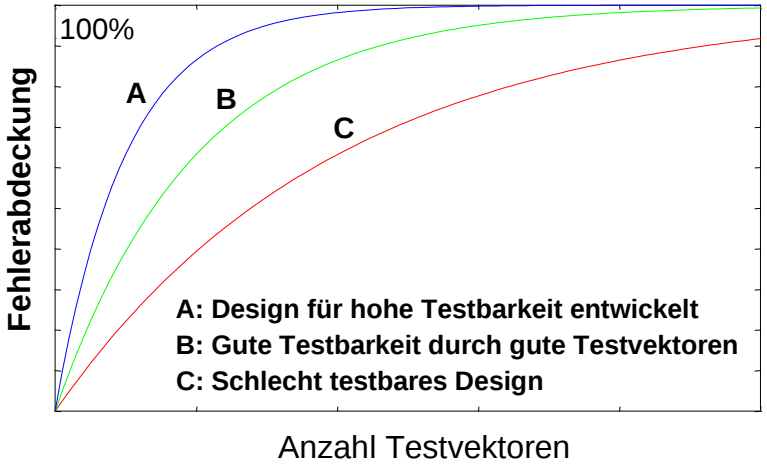




Fault simulation

- For the calculation of the required test coverage
- Returns just detectable faults
- Conclusions regarding the quality of testvectors
- For complex designs long simulation runs





Generation of testvectors

- Testcases are designed by a Test team not by the designers.
  - Require a functional description of the DUT
  - Use script languages for modeling test scenarios.
- Design of testvectors is guided by CAE systems and fault simulation software.
- Parts of the testvectors can be derived from functional simulation and system verification processes.

Test and Test Set

- A **test t** for a fault α in circuit C (realizing function f) is an input combination X for which the output of C is not equal when α is present than when it is not.

  $f(X) \text{ XOR } f_{\alpha}(X) = 1$

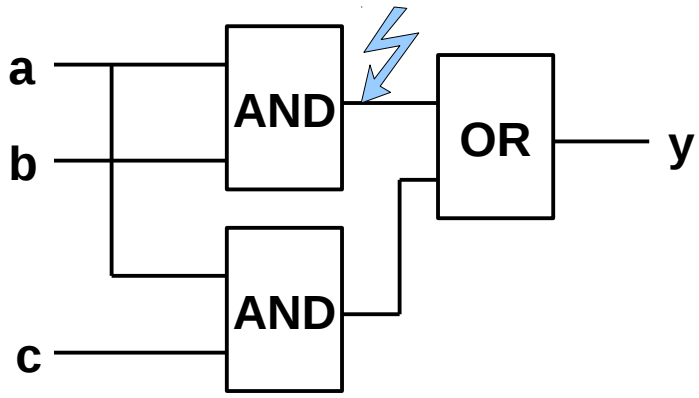
- A **test set T** is a set of t for a class of faults A,

  $\forall \alpha \in A, \exists t \in T$

  The test set for a fault α is $T_{\alpha} = f \text{ XOR } f_{\alpha}$

ATPG – from truth table

- Consider a single fault and evaluate the truth table for good and faulty circuit



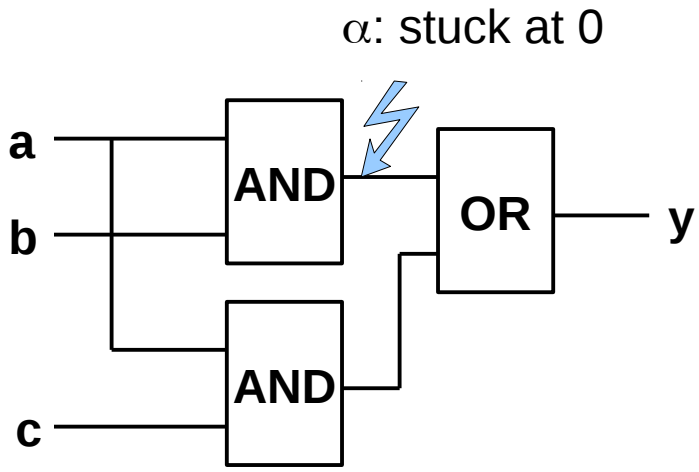
a	b	c	f	fα
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

t={1,1,0}

- T for all stuck at faults at each line results analogously where T =min{t1,t2, … }

ATPG – from boolean equations

- Consider a single fault and evaluate boolean equations for good and faulty circuit



$$f=(a\wedge b)\vee(a\wedge c), f_\alpha=a\wedge c$$

$$T_\alpha=f\ XOR\ f_\alpha$$

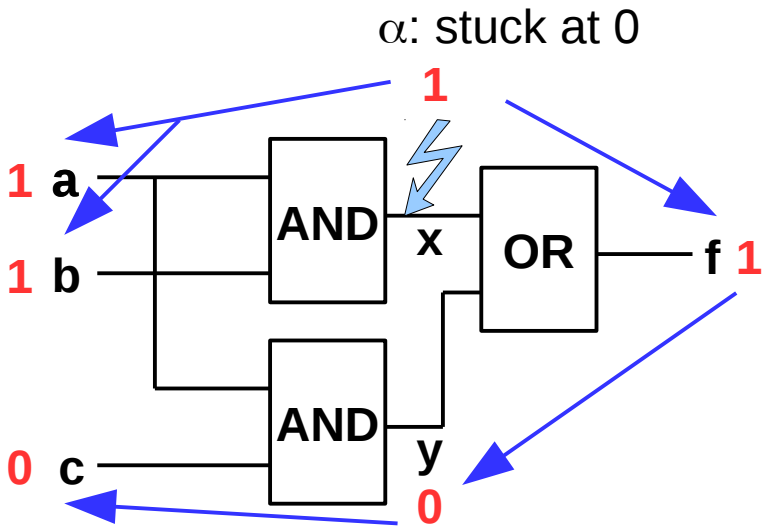
$$T_\alpha=(((a\wedge b)\vee(a\wedge c))\wedge\neg(a\wedge c))$$
$$\vee(\neg((a\wedge b)\vee(a\wedge c))\wedge(a\wedge c))=1$$

$$T_\alpha=a\wedge b\wedge\neg c=1$$

$$T_\alpha=\{1,1,0\}$$

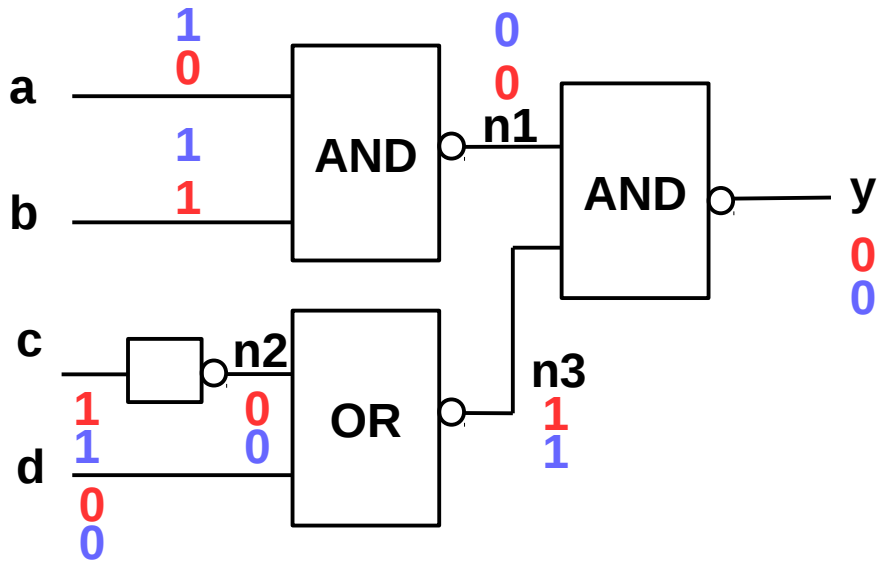
- Complexity is high since it needs to compute the faulty function for each fault

ATPG – from circuit structure



- **Fault Activation:** For observing the fault α we need to set the line x to 1.
- **Line Justification:** We set the input of previous gate(s) in a way to satisfy fault activation.
- **Fault propagation:** Propagate the fault on line x to the output f of the circuit.

ATPG – Example



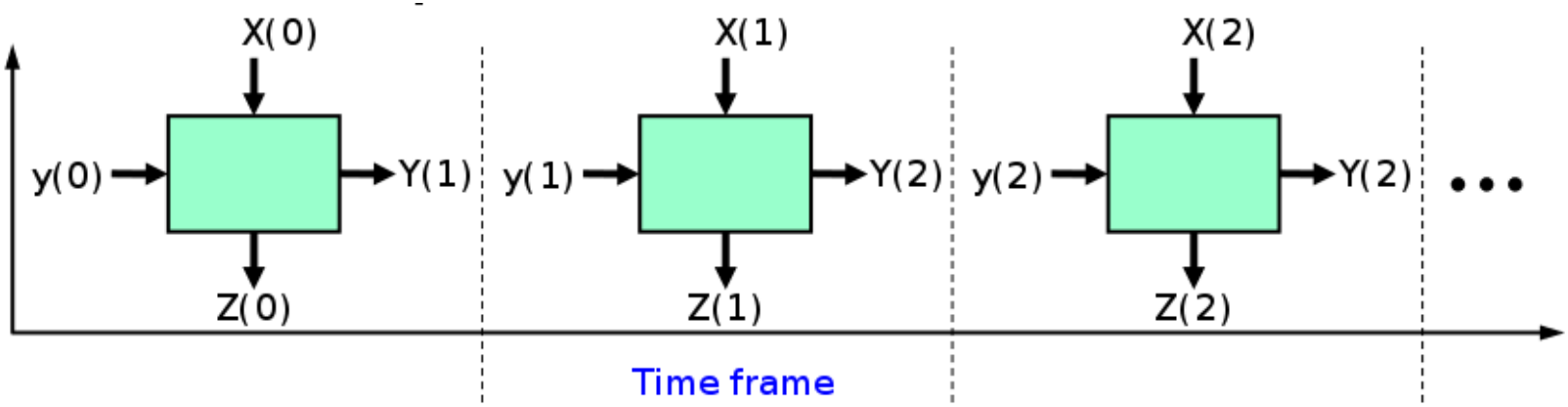
t={a,b,c,d}

	stuck at 1	stuck at 0
a	{0,1,1,0}	{1,1,1,0}
b	{1,0,1,0}	{1,1,1,0}
c	{0,1,0,0}	{0,1,1,0}
d	{0,1,1,0}	{0,1,1,1}
n1	{1,1,1,0}	{0,1,1,0}
n2	{0,1,1,0}	{0,1,0,0}
n3	{0,1,0,1}	{0,1,1,0}
y	{0,1,1,0}	{1,1,1,0}

T=min(t1,t2,t3 …) = {0,1,1,0}, {1,1,1,0}, {1,0,1,0}, {0,1,0,0}, {0,1,1,1}, {0,1,0,1}

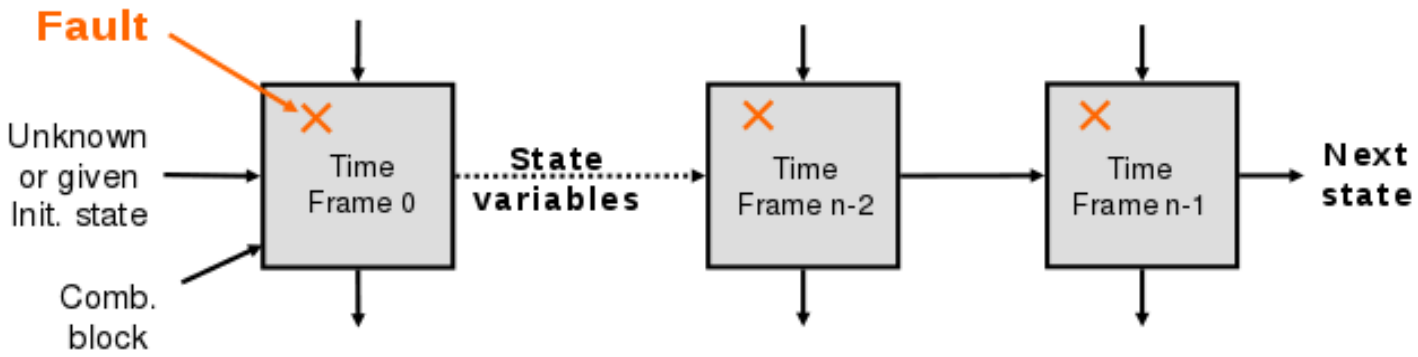
ATPG for sequential circuits

- Combinatorial test generations are all reduced to a SAT problem (see next lecture)
- Sequential circuits are computed by sequential operation of the combinatorial logic of the corresponding FSMs.
- Concept of time frame expansion



ATPG time frame expansion

- For FSM test pattern generation
- When a single stuck at fault is present a real sequential circuit, it will appear as a multiple fault in each tunfolded timeframe.
- It may happen that conflicting logical states appear.
- D-Algorithm for each timeframe.



ATPG – D-Algorithm

- Structural method which performs a boolean search using the circuit topology [Roth'66]
- Use a 5-valued logic representing states in the good and faulty circuit. Implies that only one circuit is necessary.
  - 0 → 0 in true circuit, 0 in faulty circuit
  - 1 → 1 in true circuit, 1 in faulty circuit
  - D → 1 in true circuit, 0 in faulty circuit
  - D' → 0 in true circuit, 1 in faulty circuit
  - X → unknown value in either true or faulty circuit

- Goal: Find an assignment, an input sequence that causes a D or D' at one of the output(s)

ATPG – D-Algorithm

- Termination rules of the Algorithm:
  - If an output is D or D', the stuck at error is propagated to an output → an input sequence is found for the fault
  - If k>4^n, where n is the number of FF of the circuit, then the circuit is redundant.

A and B

A\B	0	1	X	D	D'
0	0	0	0	0	0
1	0	1	X	D	D'
X	0	X	X	X	X
D	0	D	X	D	0
D'	0	D'	X	0	D'

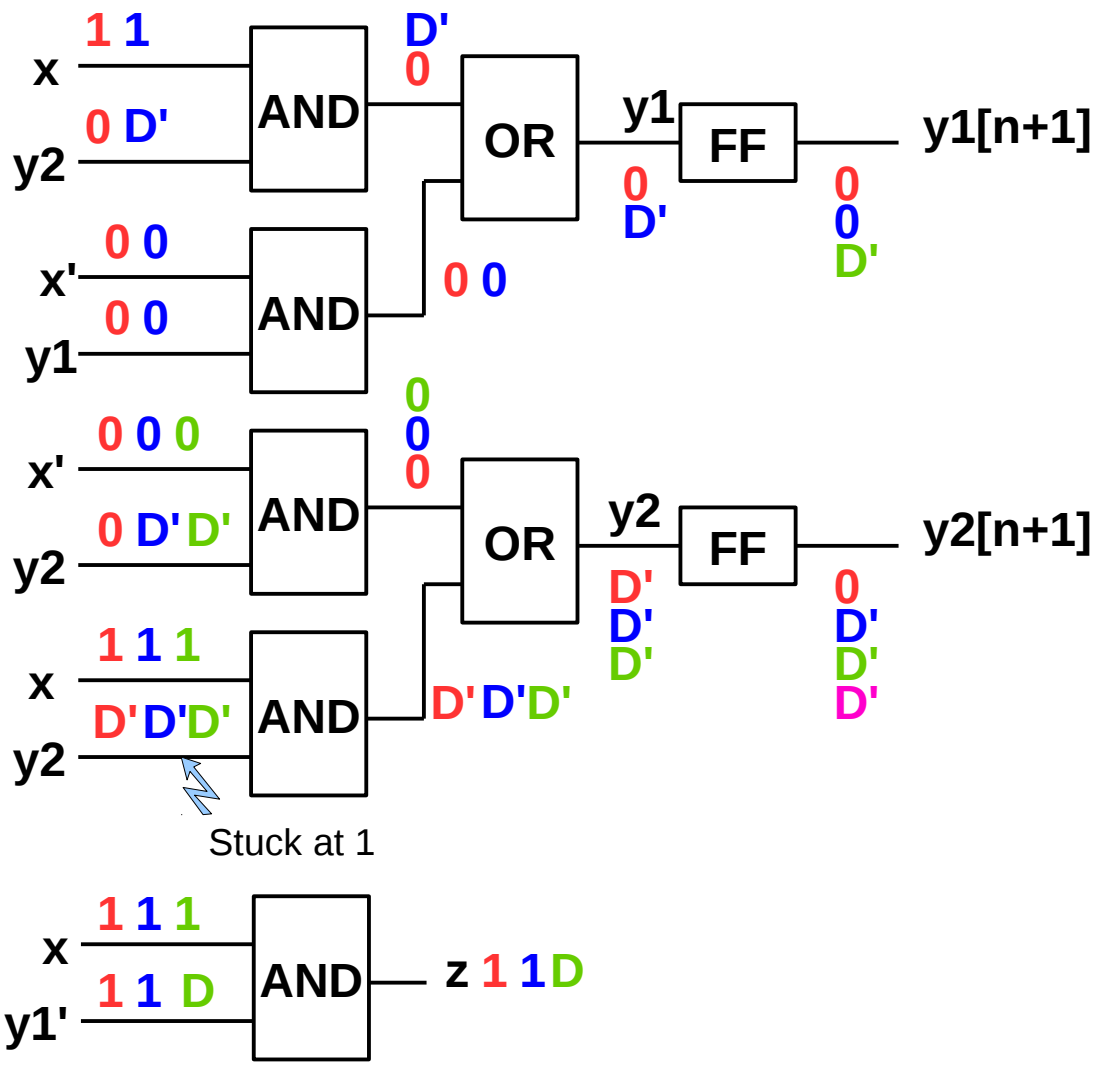
A or B

A\B	0	1	X	D	D'
0	0	1	X	D	D'
1	1	1	1	1	1
X	X	1	X	X	X
D	D	1	X	D	1
D'	D'	1	X	1	D'

not A

A	A'
0	1
1	0
X	X
D	D'
D'	D

ATPG – Example



Initial state of the FF
y1 = 0, y2 = 0

t0
t1
t2
t3

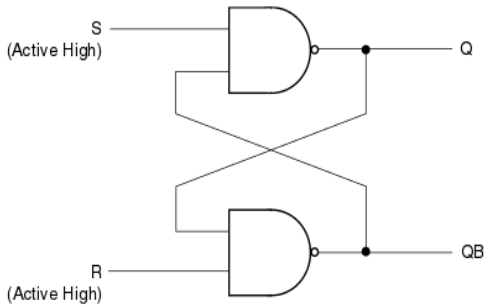
Resulting test sequence
T = [1, 1, 1]

ATPG – Example

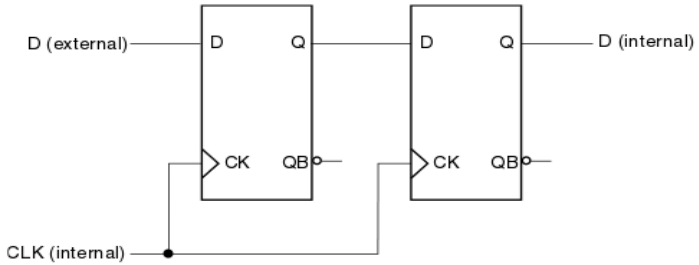
- T0:
  - Initial state y1=y2=0; y2=0 and y2=1 → D'
  - Set x(0)=1 → y2(1)=D' and z(0)=1
  - z!=D or D' → continue
- T1:
  - Set x(1)=1 → y1(2)=y2(2)=D' and z(1)=1
  - z!=D or D' → continue
- T2:
  - X(2)=1; y1(2)=D' → z(2)=D

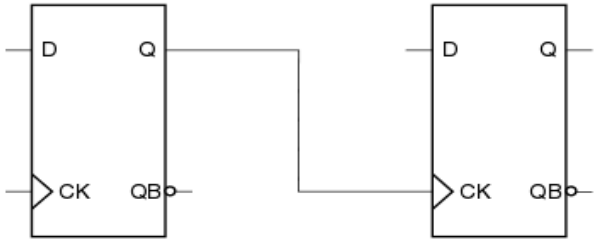
- Resulting test sequence is {1, 1, 1}. Under this test sequence the stuck at fault can be observed at the circuit output after 3 clock cycles.

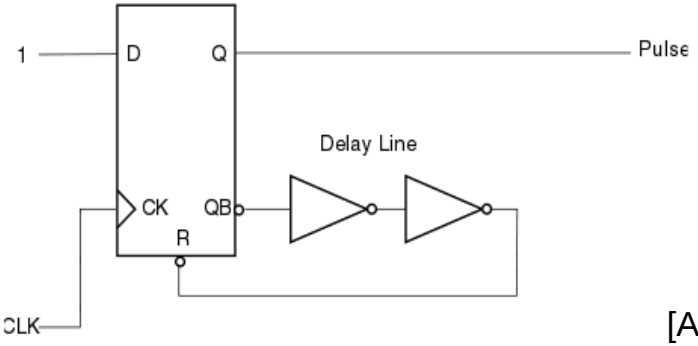
ASIC design for Testability



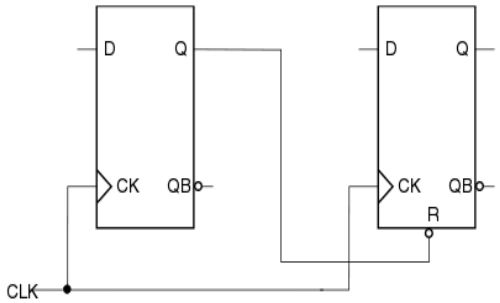
- Synchronous design
  - Use D FF, avoid latches
  - Capture (synchronize) asynchronous inputs
  - A FF must not drive reset
  - Do not generate pulses based on delay line behavior







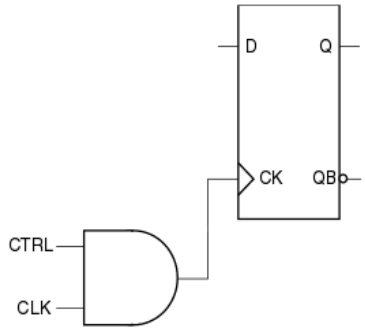
NO !

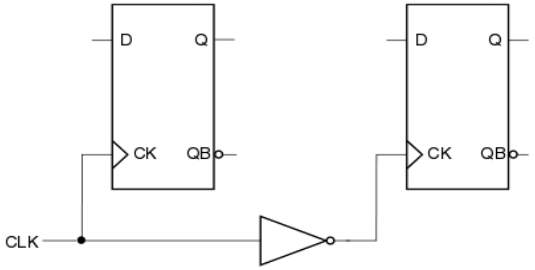


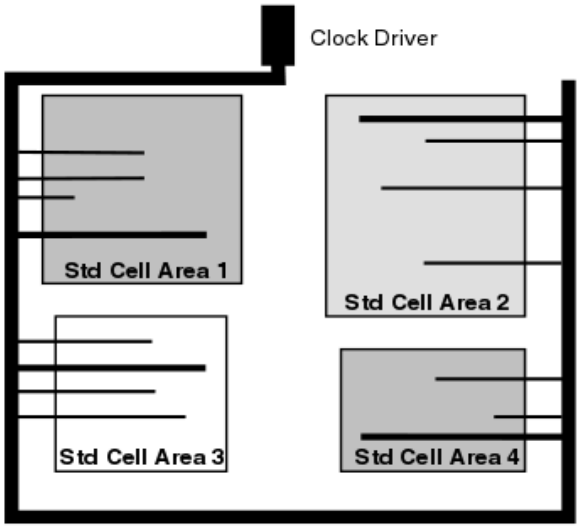
[ATMEL ASIC design guidelines]

ASIC design for Testability

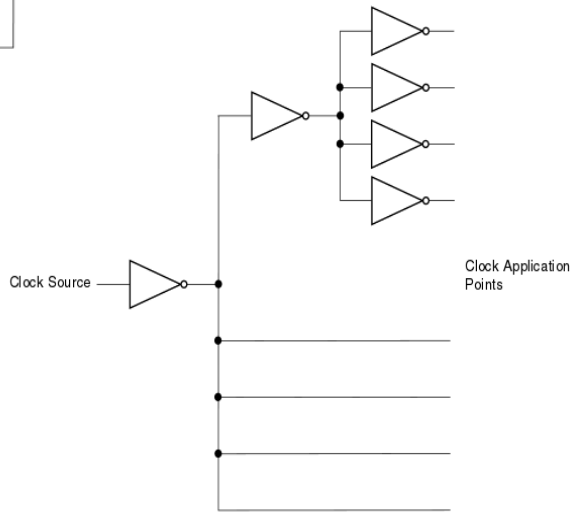
- Clock signals
  - Avoid unequal depth of clock buffering
  - Take care on a good clock tree signal distribution
  - No gated clocking
  - No double edge clocking







NO !



[ATMEL ASIC design guidelines]

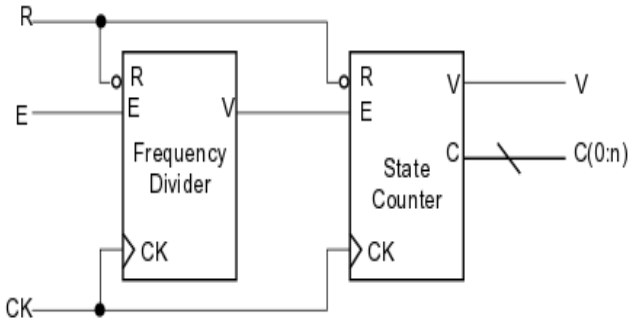
ASIC design for Testability

- Chain of counters:

  The first counter is not directly observable, and the second is not directly controllable

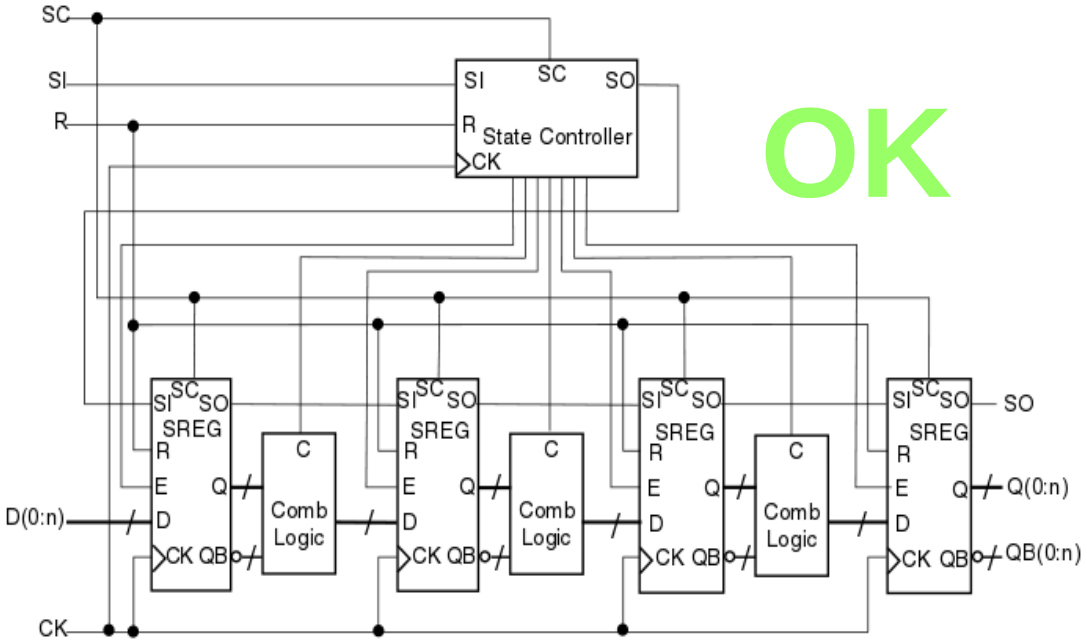
- Use BIST with compiled Megacells
- Form scan path, from scan input si to scan output so

NO !



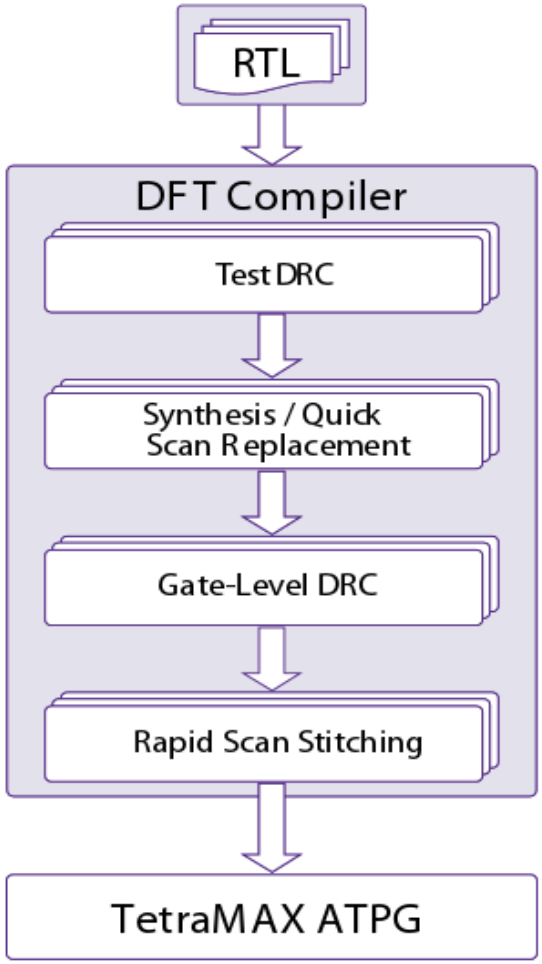
[ATMEL ASIC design guidelines]

OK



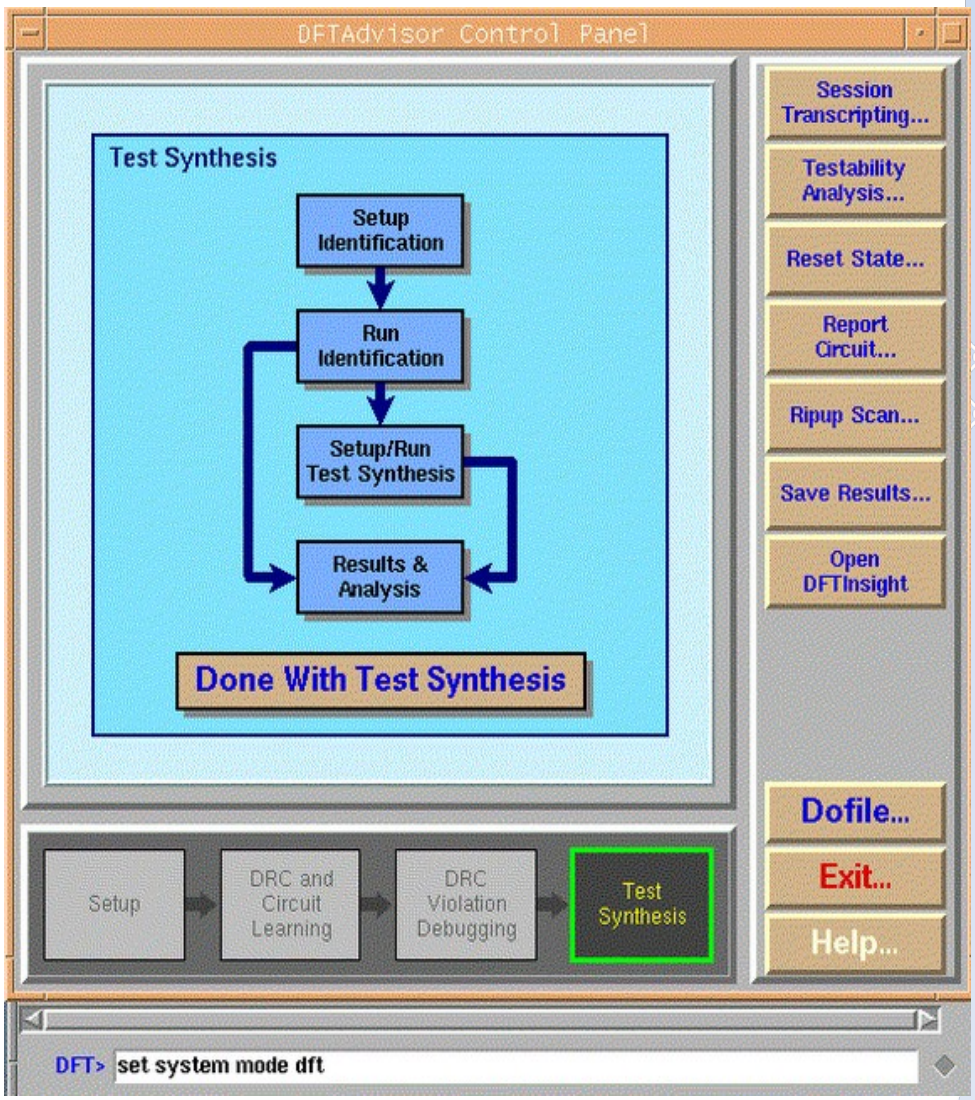
Software Support

- SYNOPSYS DFT Compiler
  - Delivers scan DFT transparently within Synopsys' synthesis flows
  - Conduct in-depth testability analysis at RTL level
  - Enables rapid implementation of the most effective test structures at the hierarchical block level.
  - TetraMAX ATPG:
    - PrimeTime interface selects critical timing path
    - Tester-ready patterns with complete timing
    - Easy-to-use flow with graphical support for analysis and debug



Software Support

- MENTOR DFT ADVISOR
  - Supports both full and partial-scan identification and insertion
  - Reads most standard gate-level netlist
  - Contains a powerful design rules checker
  - After inserting the scan circuitry, you can then generate test vectors for your design by using Fastscan



Thank you for your attention

rathmair@ict.tuwien.ac.at