1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN TECHNICAL UNIVERSITY

DESIGN FOR TESTABILITY

Raimund Ubar

raiub@pld.ttu.ee

ICT-523

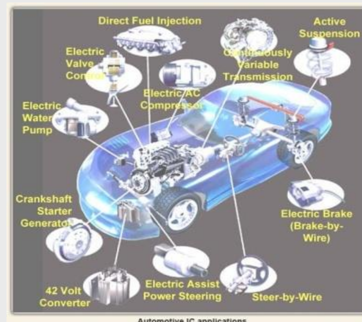
Testability of Digital Systems

- Technology advances have enabled the implementation of complex digital systems in single chips, reducing size and power consumption
- This has intensified the complexity and the cost of testing such chips to verify that they function correctly
- As the result, special design techniques have to be used to make a chip fully testable
- This course is about these techniques

The course have been given annually also in the International Summer School at TU Darmstadt in Germany since 2002 already more than 15 years

Computers and Embedded Systems

Universal computers 1%



Embedded systems 99%

Microprocessor market shares

We notice our dependency on electronics only when it suddenly gives up to work

Why testing is important?

We depend too much on computers and on the technical systems controlled by computers

Computer History: The First Computer Bug

1945 in Harward

"Hey, we actually found a bug that was a real bug!"

A bug landed between two relais contacts, and caused an electronic bridge:

A bridging fault

Why thinking about test is important?

Test is costly and still produce no value, except trust

- ✓ From the whole **cost of car 40%** goes for electronics and software

  From that

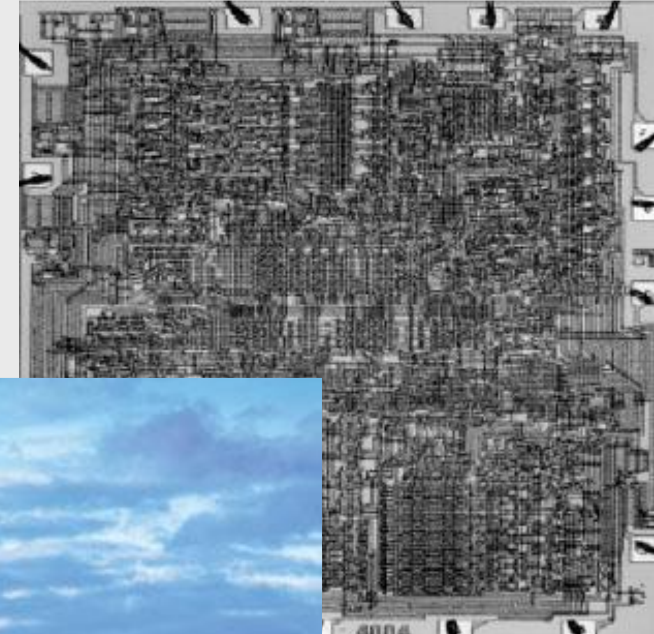
  - 60% for **software**
    (25% from the whole cost of car)
  - 40% for **hardware**
    (15% from the whole cost of car)

The cost of fault diagnosis is about 20-25% from the whole cost of the car

- ✓ From the whole cost of **software design 50%** goes for fault diagnosis and repair (**10-15%)**
- ✓ From the whole cost of **hardware design 70%** goes for fault diagnosis and repair (**10%**)

Why DFT is important?

- **The main property of today's systems is COMPLEXITY**
- To manage the complexity we have to know methods like:
- - abstraction
- - modeling
- - simulation
- - hierarchical „*divide and conquer*"
- - ...



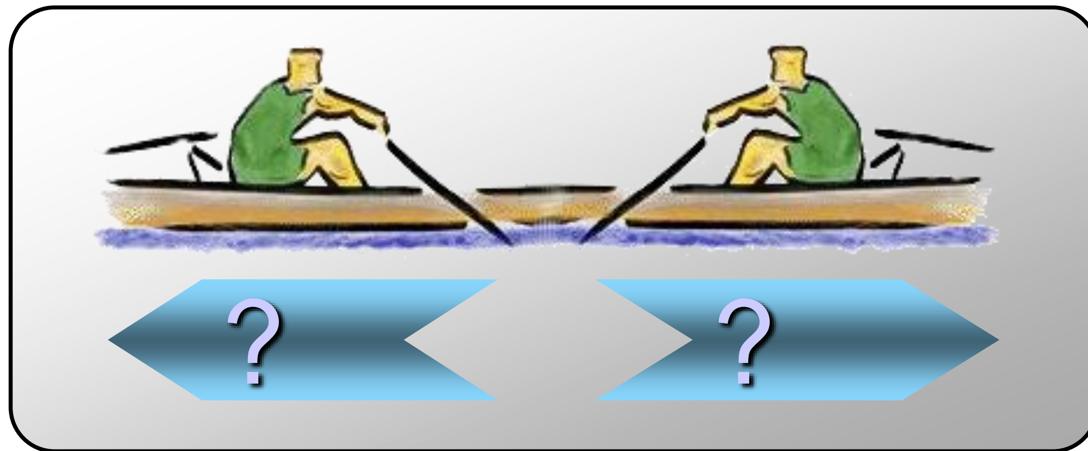
Cooperation in the Fields of Design & Test

Designer to Test engineer:

- **Check if all my implemented functions are working**

Test engineer answers:

- **Redesign the thing, so that it were testable, I cannot access to the pins I need**



H.-J. Wunderlich, U Stuttgart

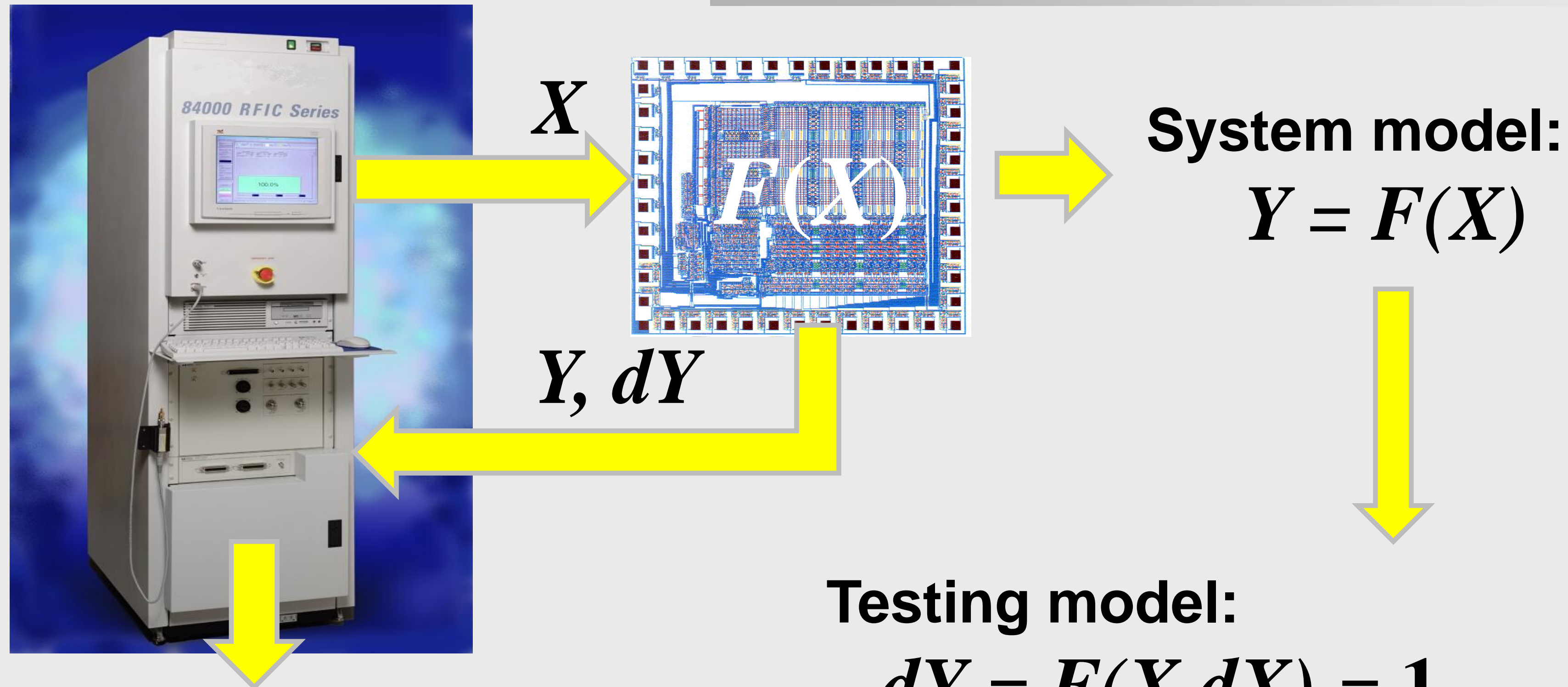
Paradigm change: ***Design for testability***

Design and Test as Reverse Problems

Difference between design and design for test

- **Design**– is the field of direct problems
- **Design for Test** - is the field of reverse problems
- The central question of the diagnosis and DFT is – **WHY?**
- **This is the general question of every creative engineer**
- **How?** – is technical question and problem

Diagnosis as Reverse Problem to Design



System model:

$$Y = F(X)$$

Testing model:

$$dY = F(X,dX) = 1$$

Diagnosis model:

(reverse problem)

$$dX = F^{-1}(X,dY)$$

Task	Given	Find
Test synthesis	dX	X
Test analysis	X	dX

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

About the Course

- This course is an introduction to **fault diagnosis and self-testing** of digital systems. More difficult than to create a system is to guarantee that the designed and implemented system is correctly functioning
- **Fault diagnosis** is a reverse task to design, and has the target to answer the question, why a system is not working properly
- **Self-testing system** is the prerequisite of dependability of the technical world around us

- ***The difference between a programmer and a test engineer is that the programmer can write a software only for a computer that is working, whereas the test engineer must be able to program any faulty computer while the number of possible faults is infinite***

- This course is about **how to program a faulty computer**

Design for Testability

Lectures

- Testability of Digital Systems
- Design for Testability Methods
- Built-in-Self-Test/Diagnosis - BIST/BISD

Practical Works

- Two laboratory Works (Begin: 19.09)
- Course work

Exam

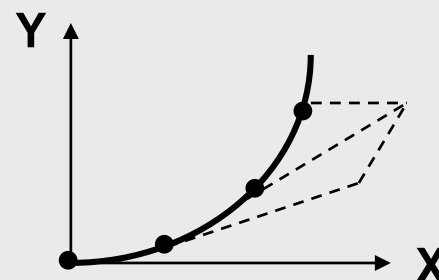
References

1. M.Bushnell,V.Agrawal. **Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits**. Springer Publishing Comp., Incorp., 2013.
2. M.Ottavi, D.Gizopoulos, S.Pontarelli (ed.). **Dependable Multicore Architectures at Nanoscale.** Springer, July 2017.
3. H.-J.Wunderlich (ed.). **Models in Hardware Testing**. Springer, 2010, 255 p.
4. L.-T.Wang a.o.. **VLSI Test Principles and Architectures**. Elsevier, 2006, 777 p.
5. D.Gizopoulos (ed.). **Advances in Electronic Testing: Challenges and Methodologies**. Springer, January 2006
6. O.Novak, E.Gramatova, **R.Ubar**. **Handbook of Testing Electronic Systems**. Czech TU Publishing House, 2005, 395 p.
7. **R.Ubar**, J.Raik, H.-T.Vierhaus. **Design and Test Technology for Dependable Systems-on-Chip.** IGI Global, 2011, 550 p.
8. **R.Ubar**, A.Jasnetski, A.Tsertov, A.Oyeniran, Software-Based Self-Test with Decision Diagrams for Microprocessors. Lambert Publishing House, 2018, 171 p.

Why the topic of DFT is important?

- To learn logic by mastering the **problems of diagnosis**
- The most difficult logic in the technical world is the **logic of digital systems**
- Even more difficult **is the logic of diagnosis**
- This is the reason why the best way to learn technical diagnostics is to use as objectives the digital systems
- The abstraction and reality are in digital systems nearly in one-to-one relationship

Analog system (amplifier):

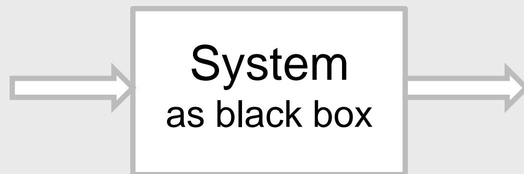


Three measurements is OK

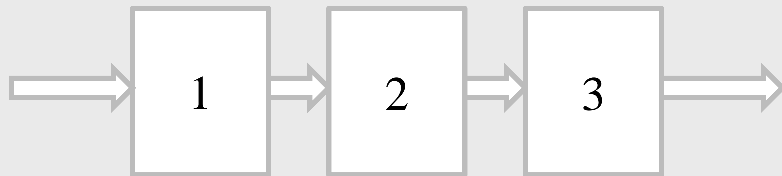
Digital system:

$X \rightarrow 2^{64} = 18\ 446\ 700\ 000\ 000\ 000\ 000 \approx 10^{19}$

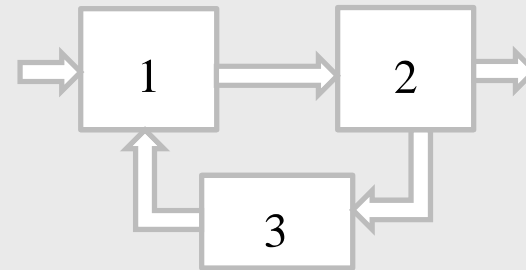
Difference between Testing and Diagnosis



Testing and diagnosis are not distinguishable



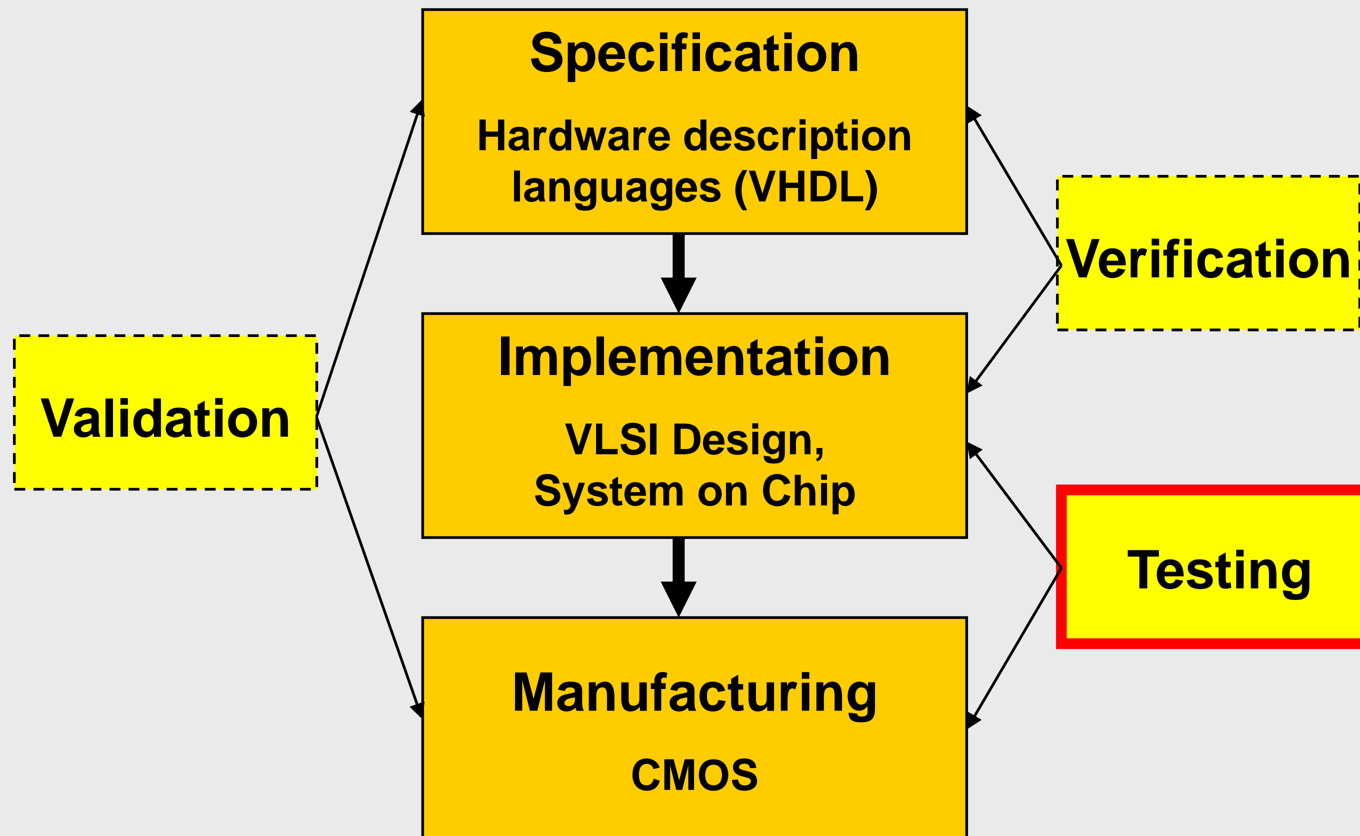
Displaying the structure of the system makes diagnosis possible



Feedback loop is the headache of testing and diagnosis

Huge number of components with a complex and deep nested feedback structure introduces the problem of complexity

Terminology: Verification, Validation, Testing

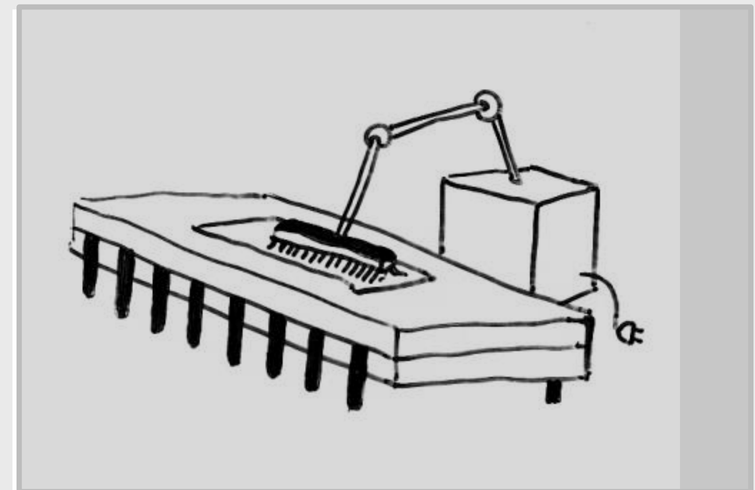


Verification is to check the consistence between the individual development phases

Validation is checking the system whether it conforms to the user requirements

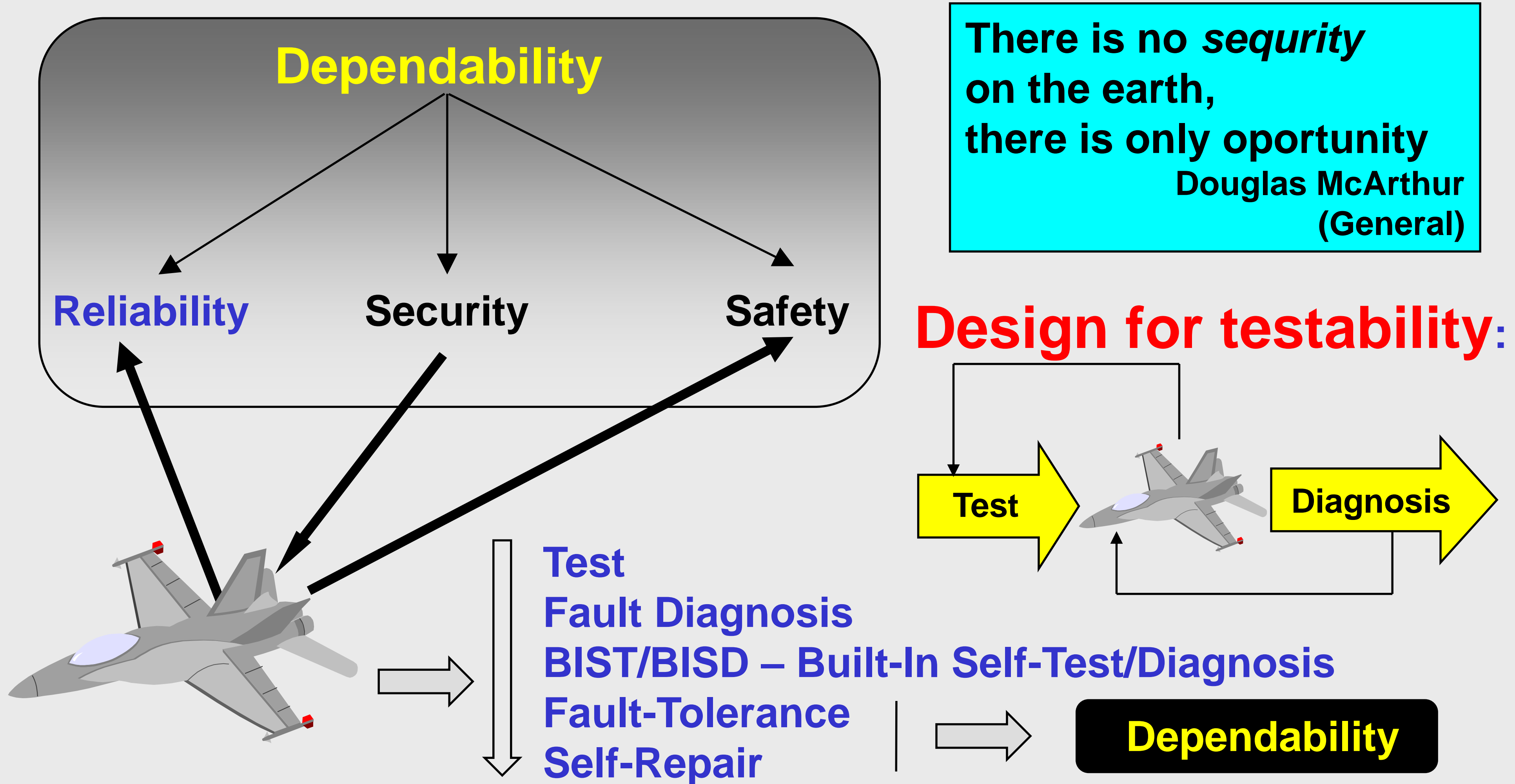
Practical Importance of Testability?

- To improve the manufacturing processes and to increase the yield
- To design reliable systems out of not reliable components which leads to the need of fault-tolerance
- Field diagnosis as the traditional task
- The Rule of Ten is the Sword of Damokles
- The increasing complexity of VLSI circuits has made test and diagnosis the most complicated problems in digital design



Automated diagnosis is needed

Position of DFT in Dependability of Systems



Test Related Tasks, Data and Tools

Test experiment

(BIST)

System

Test result

Fault simulation

Fault diagnosis

Test

System model

Fault dictionary

Test generation

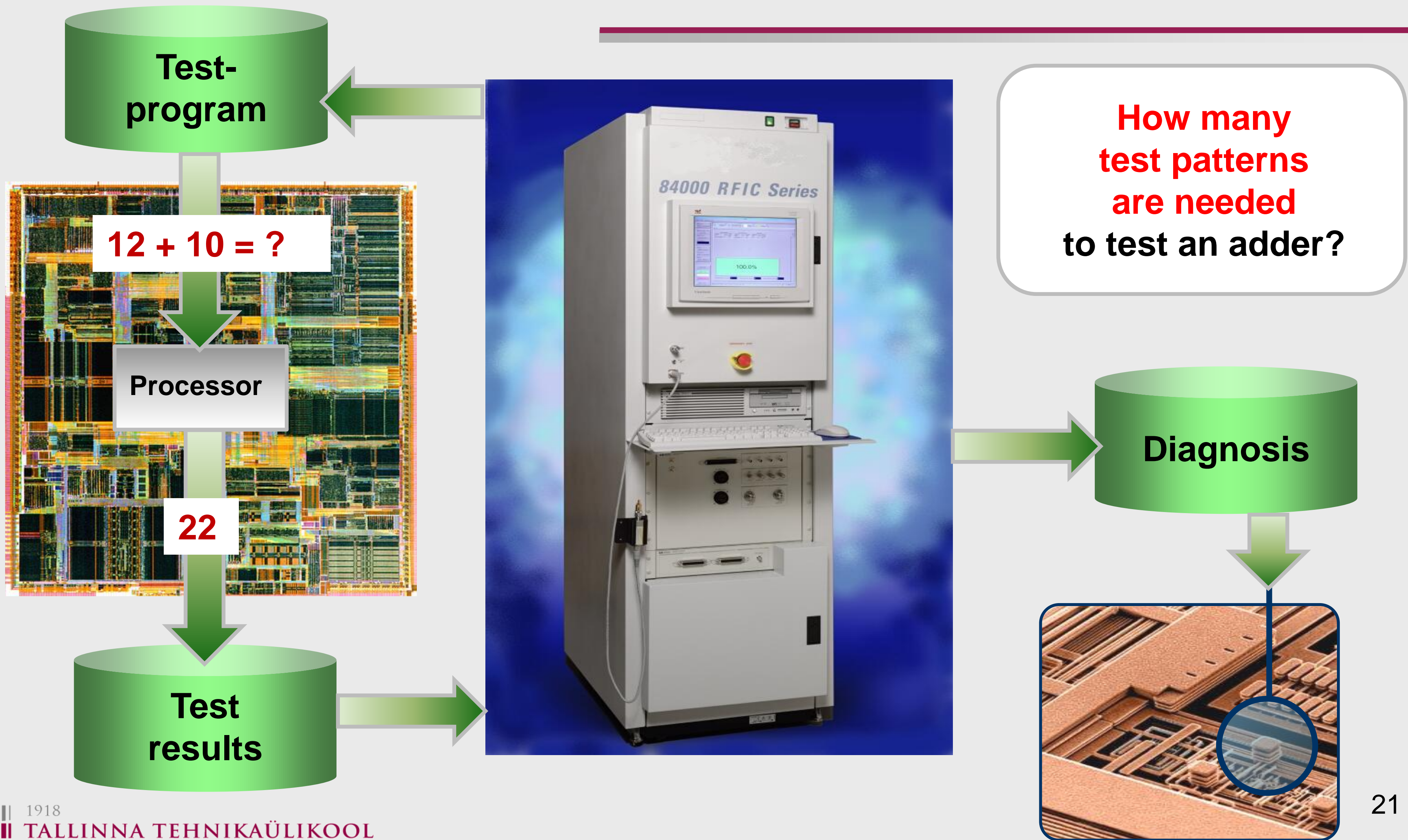
Go/No go
Located defect

Test tools

Goals of the DFT Course

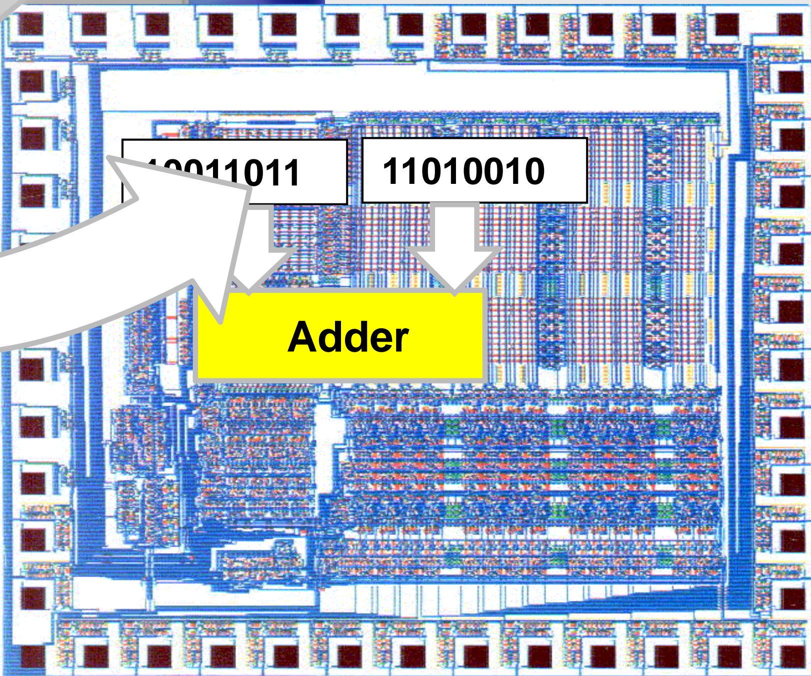
- To give the basic knowledge:
  - How to improve test quality at increasing complexities of systems?
- This knowledges includes
  - understanding of how the physical defects can influence on the behavior of systems, and what is diagnostic modelling
  - learning fault simulation, test generation and fault diagnosis
  - understanding the meaning of testability
  - learning the basic methods of making systems self-testable
- The goal is also to give some hands-on experience of solving test related problems

What is a Test?



Complexity vs. quality in Testing

32 bit adder has 64 inputs
The number of all possibile test patterns is
2^{64} = 18 446 744 073 709 551 616 ≈ 10^{19}



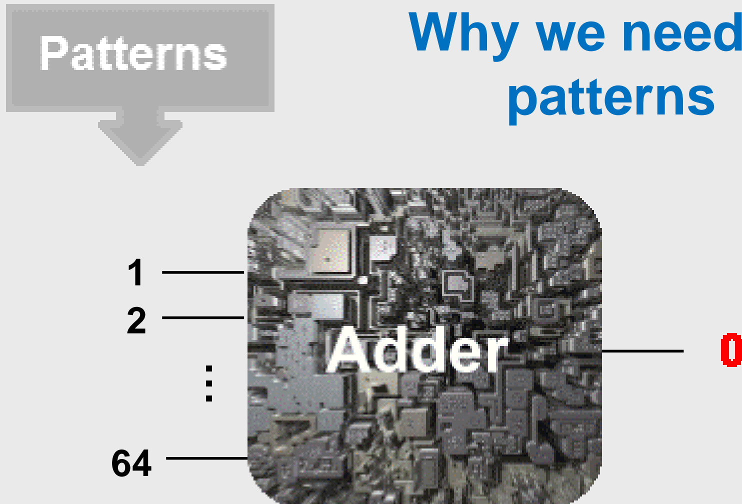
1 GHz processor will need
2^{64} = 18 446 700 000 ≈ 10^{10} sec
or **584** years

Mikroprocessor
in the factory is tested
only with **10** sec

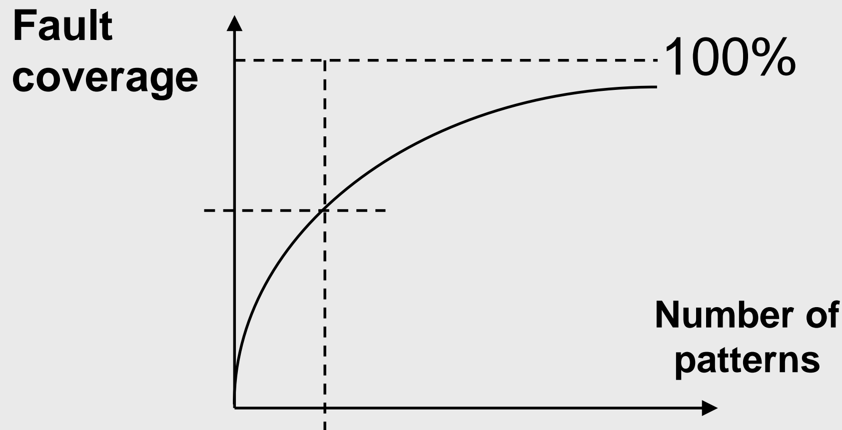
How about the quality of test in this case?

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Fault Coverage: Functional View



Why we need 2^{64} patterns



Truth table for adder:

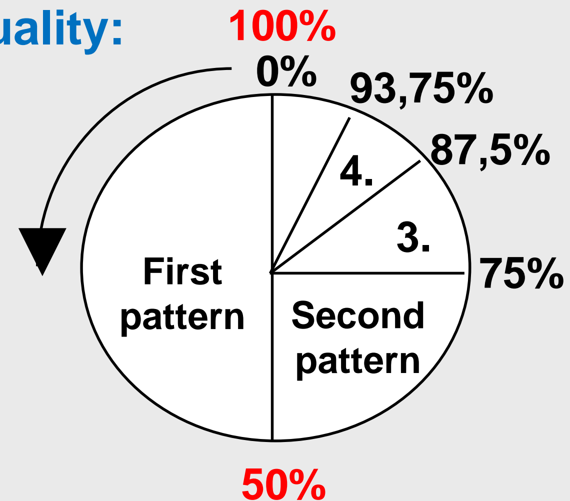
	Patterns	Possible functions
2^{64}	00...000	0 1 0 1 0 1...101
	00...001	0 0 1 1 1 0...011
	00...010	0 0 1 0 0 1...101
	...	...
	11...111	0 0 0 0 0 0...111

First pattern

Correct function

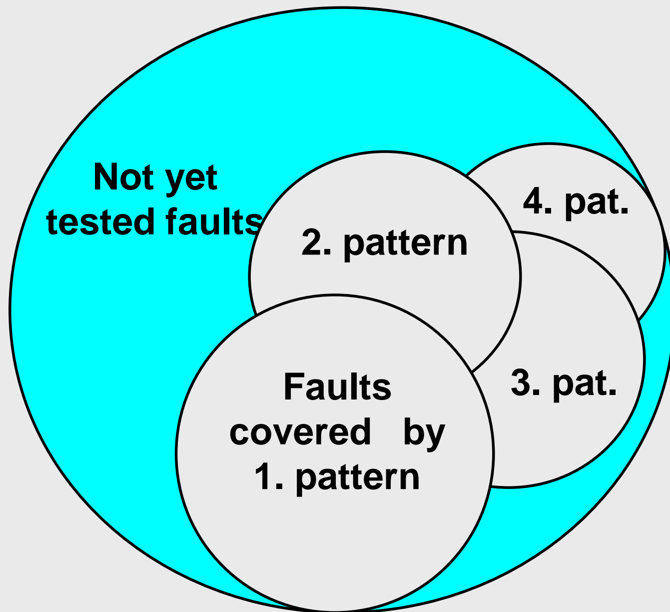
50% tested

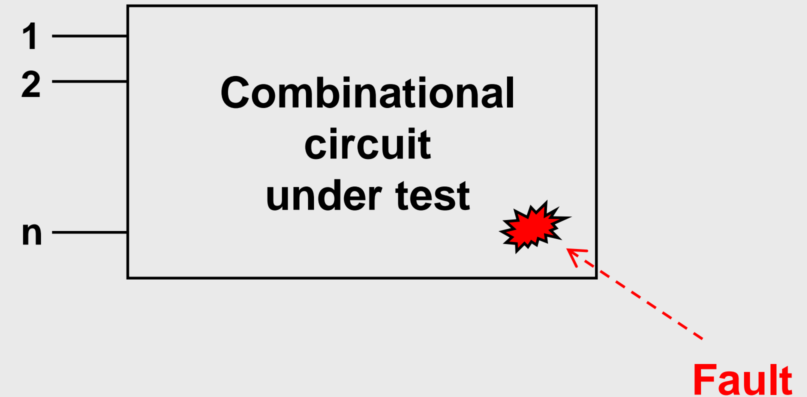
Test quality:

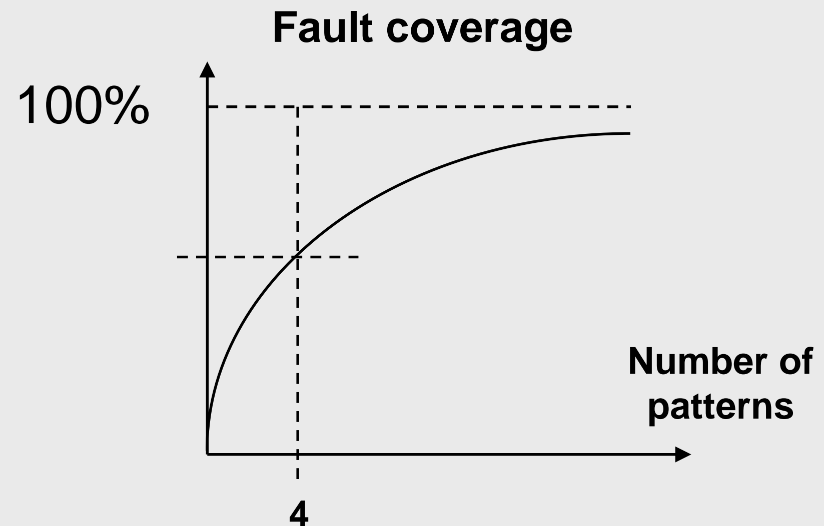


Fault Coverage: Structural View

Testing of structural faults:

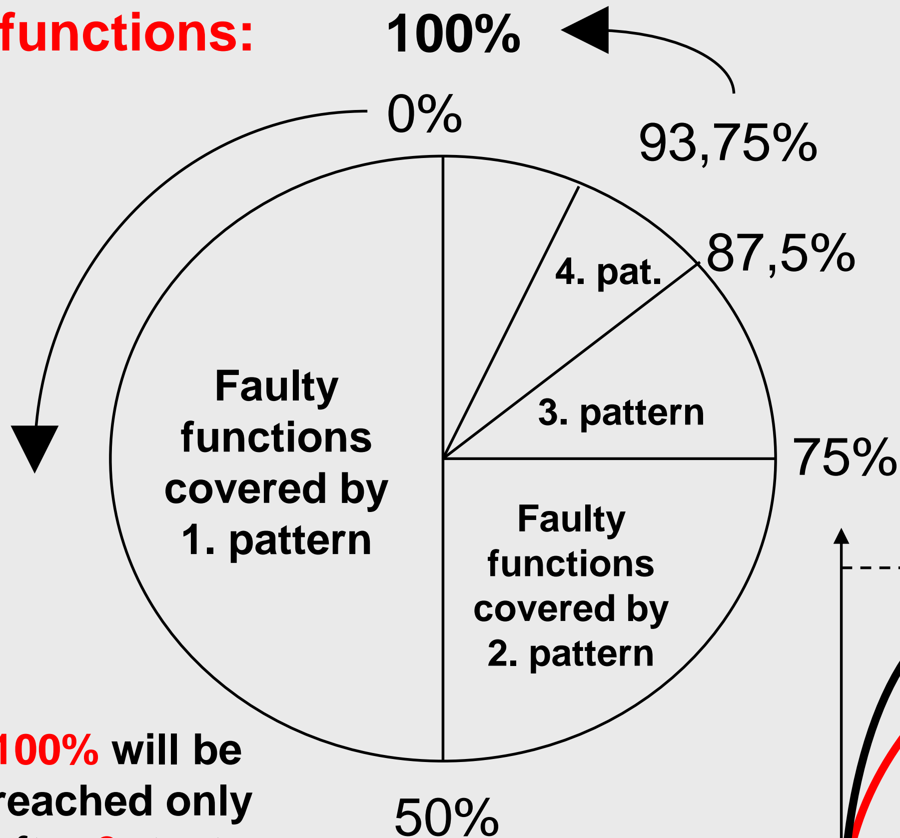






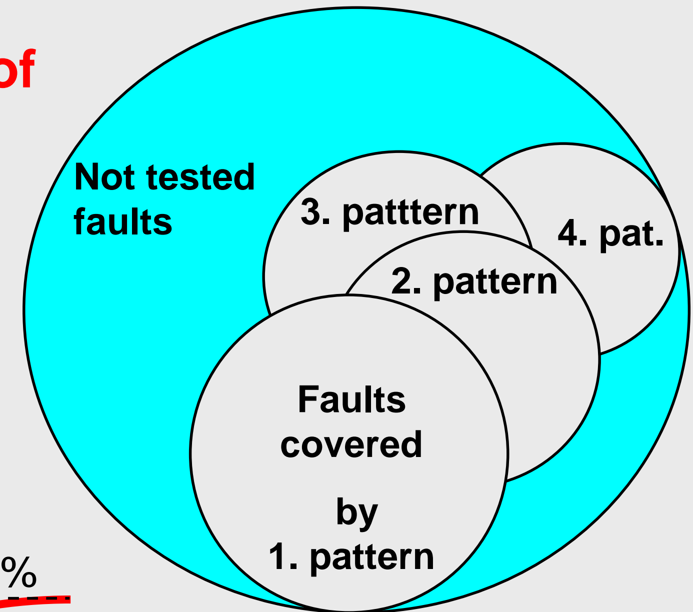
Comparison of Two Quality Measures

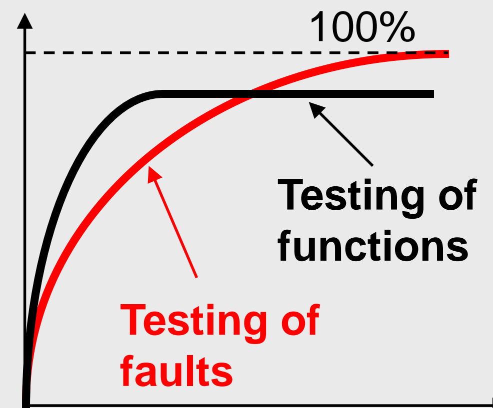
Testing of functions:



100% will be reached only after 2^n test patterns

Testing of faults:





100% will be reached when all faults from the fault list are covered

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Difficulties with Fault Model Based Testing

Diversity of fault models:

- ✓ Delays, bridges, shorts, opens...
- ✓ SAF, constrained SAF, X-faults...

Problems:

- ✓ Fault list – which faults, how many?
- ✓ Number of fault models is very large
- ✓ Standard model is SAF, however, it is not enough adequate to represent real defects
- ✓ Multiple faults can mask each other, they must be taken into account – too complex task

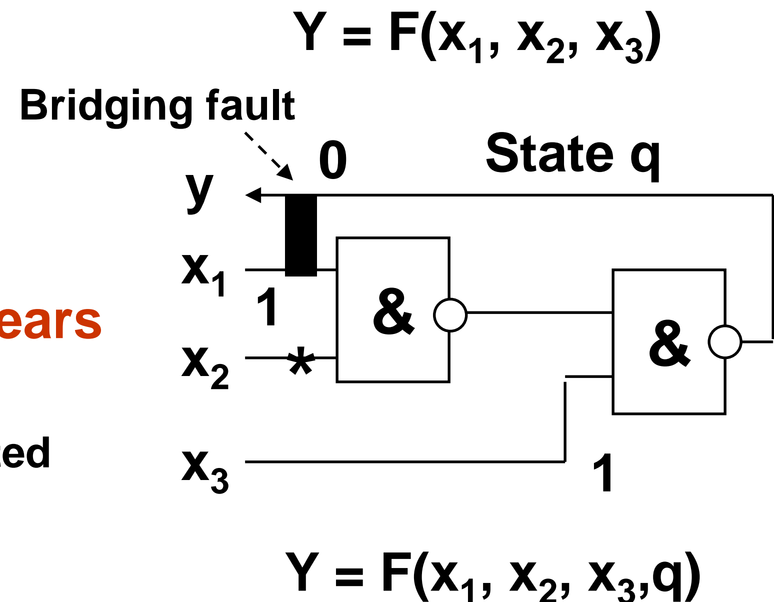
Test Complexity and Quality Paradoxes

1. Paradox of fault model:

2^{64} input patterns (!)
for 32-bit accumulator
will be not enough.

A short will change the circuit
into sequential one,
and you will need because of that
2^{65} input patterns

Time can be your best friend
or your worst enemy
(Ray Charles)



2. Paradox of test quality:

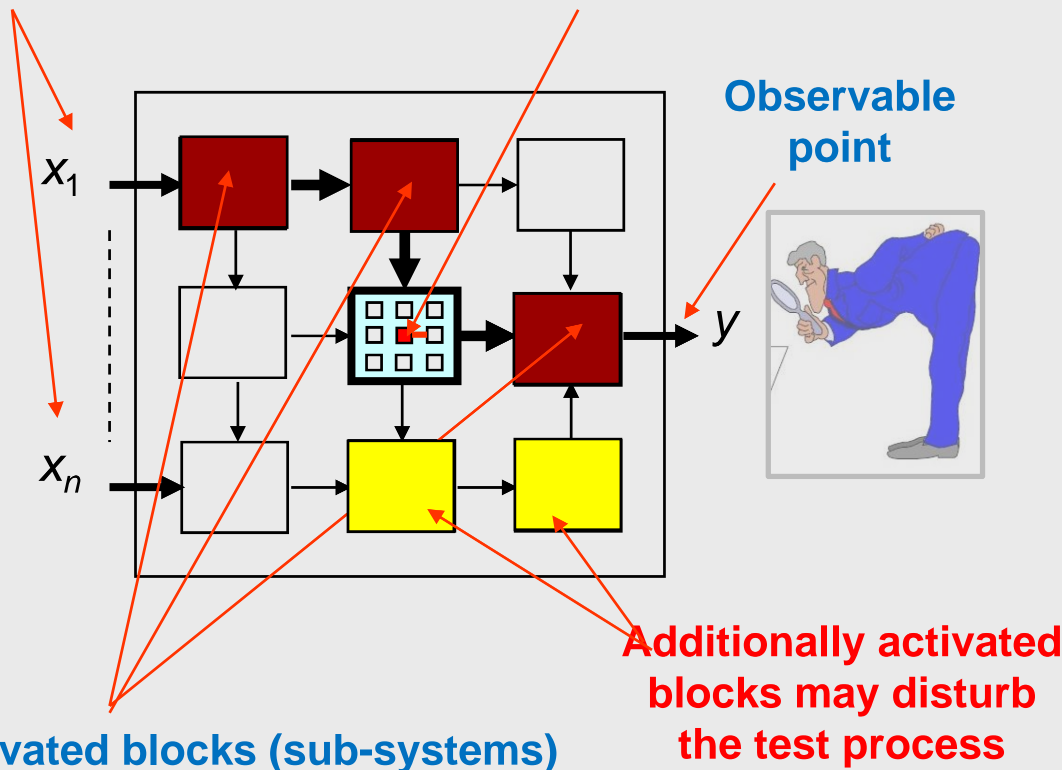
It has been counted that Intel 8080
needed for exhaustive testing **37 (!) years**

Manufacturer did it by **10 seconds**
Majority of functions will never activated
during the lifetime of the system

The Fault List tends to be infinite

The Complexity of Test Generation

Fault propagation in complex systems:



Example:

32-bit adder

Functional test:

Number of test patterns

$N = 2^{64} = 10^{19}$

Fault model based structural test:

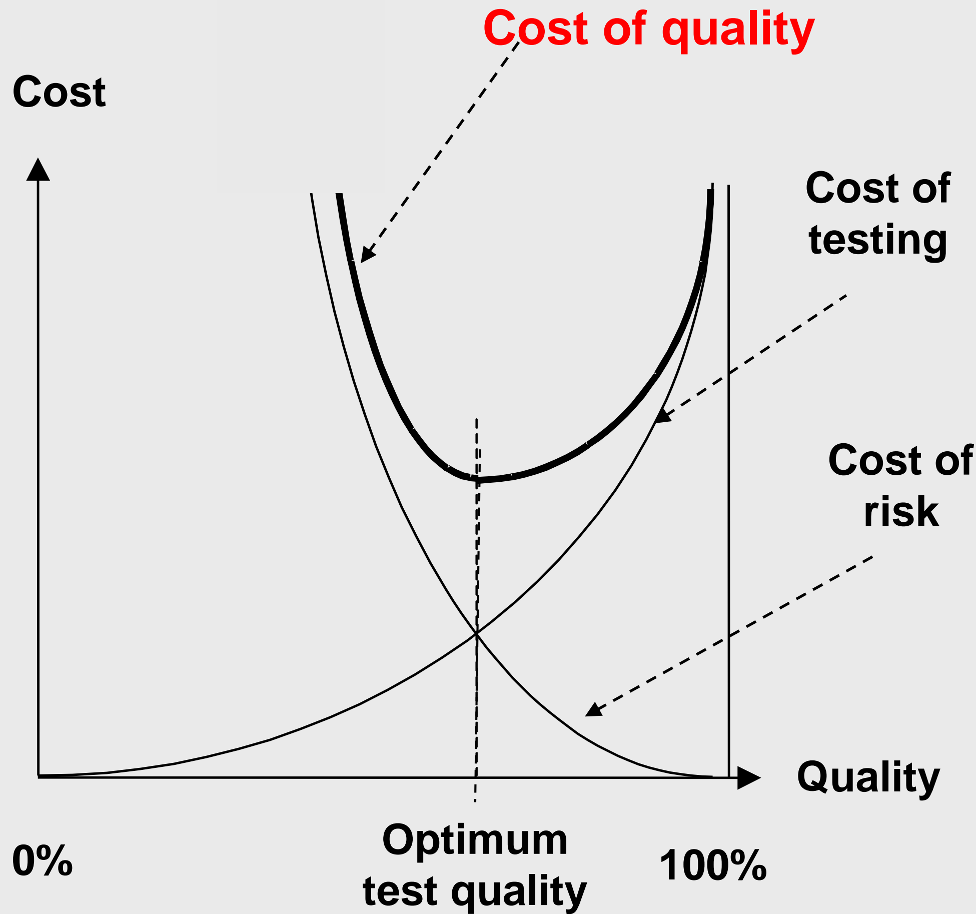
Alltogether about 2000 faults

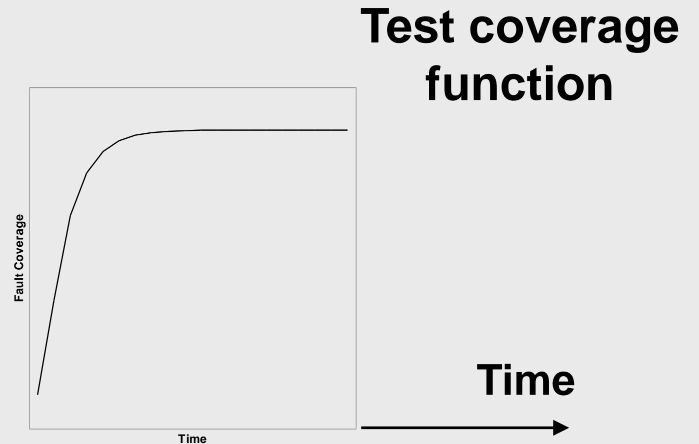
Number of test patterns:

$N << 2000 << 10^{19}$

© Raimund Ubar

The Problem of Quality is Money?





Conclusion:

"The problem of testing can only be contained not solved"

T.Williams

Motivation for DFT

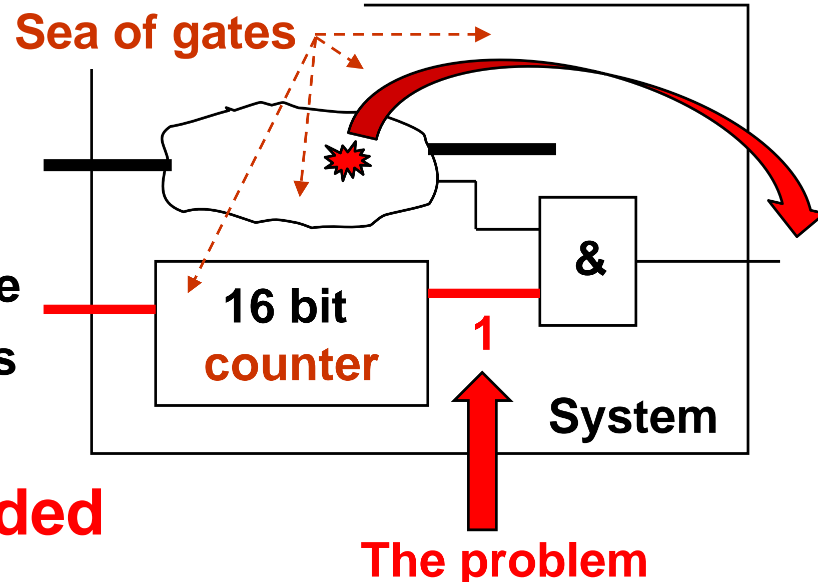
Engineer vs. computer:

To generate a test
for a component
in a system,
the computer
needed
2 days and 2 nights

An engineer
did it „by hand“
with 15 minutes

Sequence
of 2^{16} bits

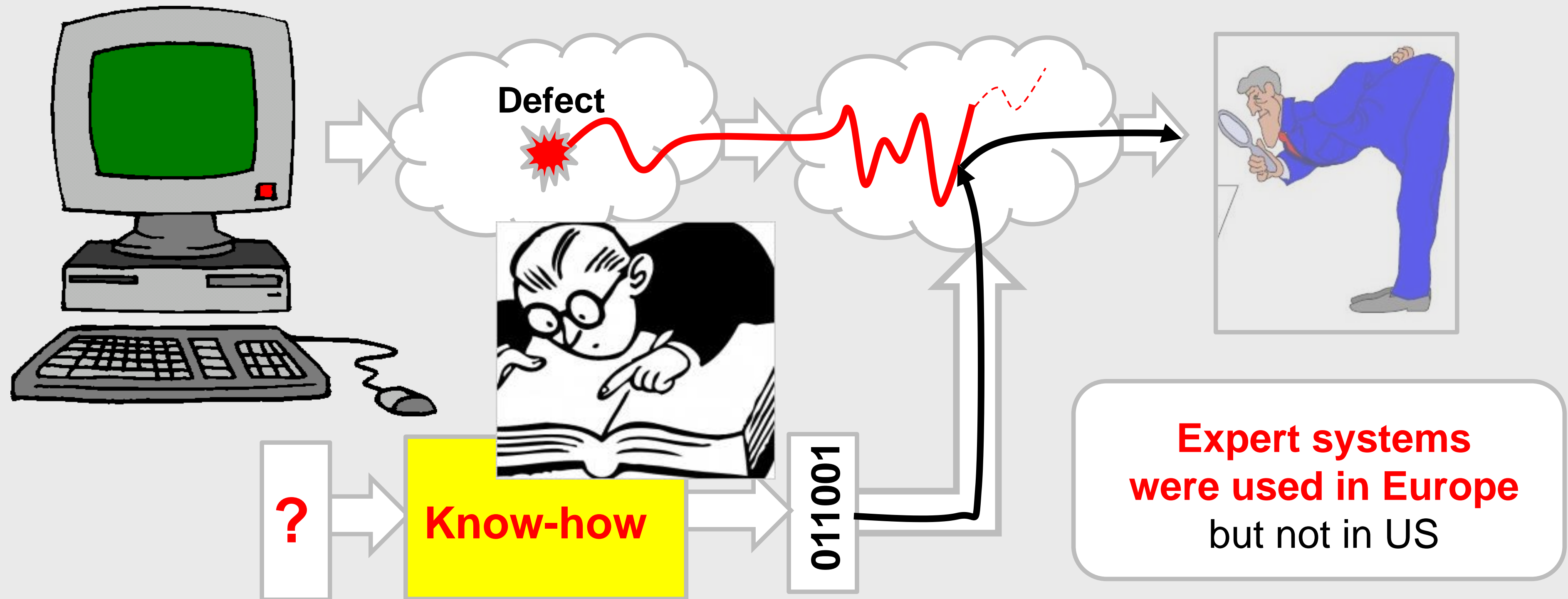
The best place to start is
with a good title.
Then build
a song around it.
(Wisdom of country music)



Design for Testability is needed

Motivation for DFT

Test generation process for detecting a fault:



Expert system is needed to help the test programmer

DFT is „Easy"

Test generation process for detecting a fault:

Defekt

?

Hard-to-test-part

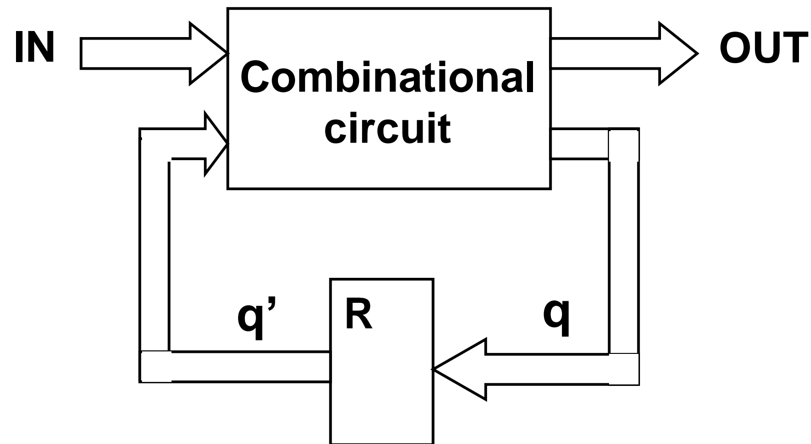
011001

011001

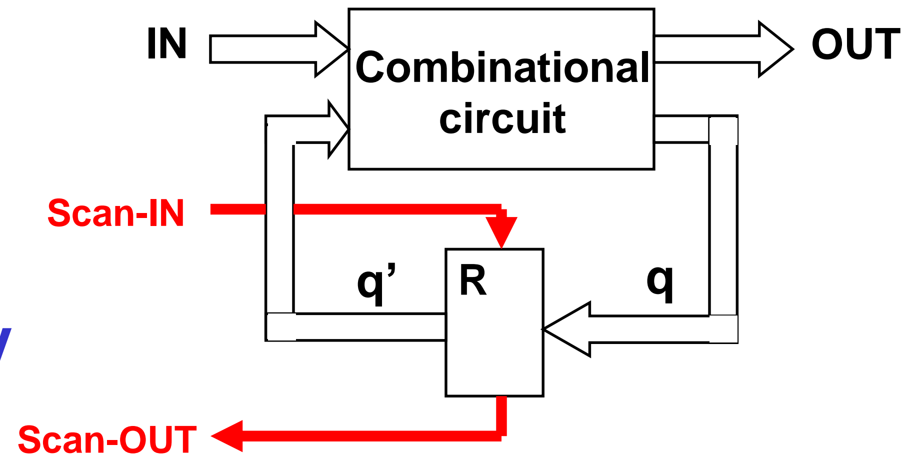
New paradigm
ScanPath Design

mindmappingsoftwareblog.com

DFT: Making Systems Transparent

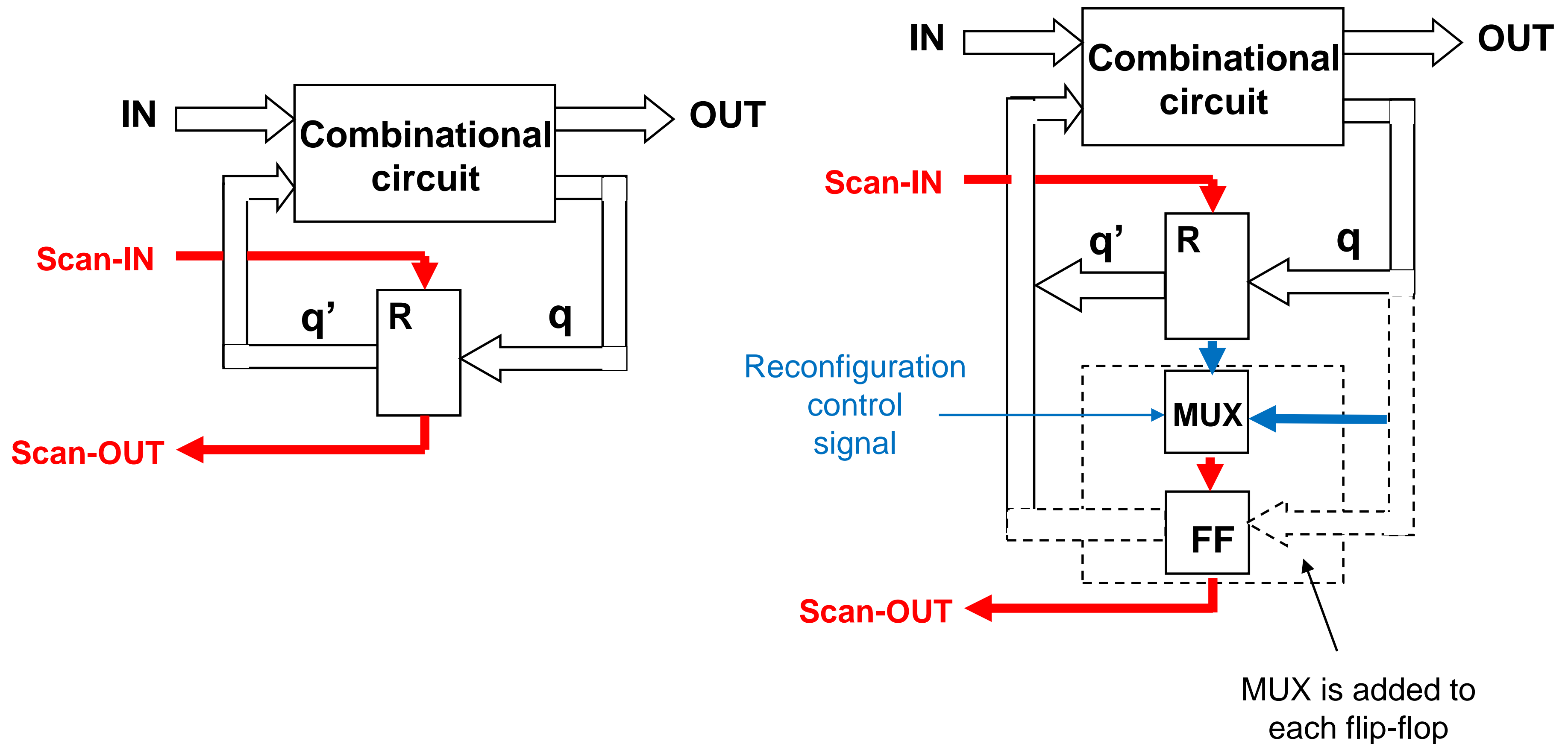


theisleofwightcomputergeek.co.uk

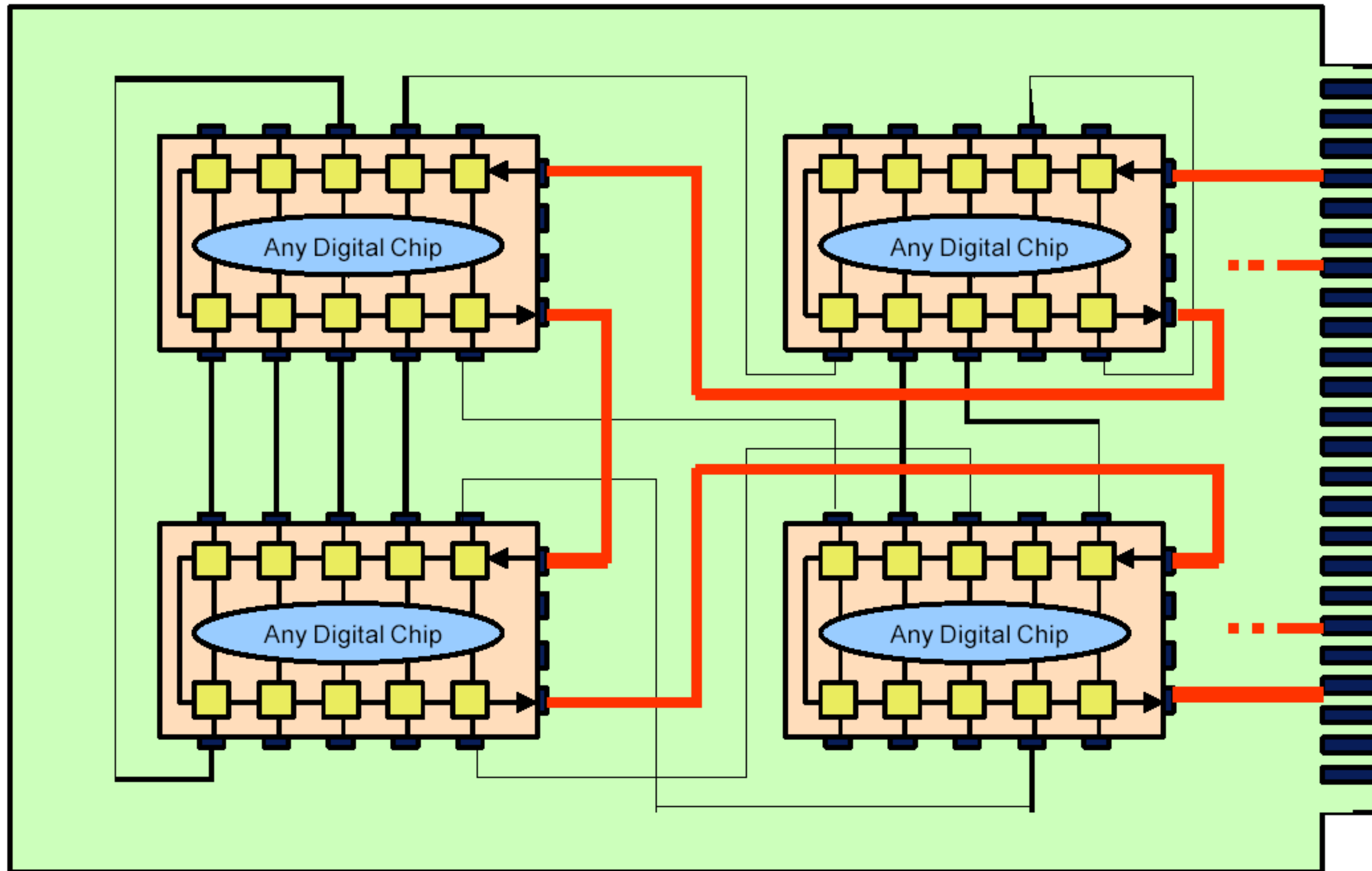


Scan-Path design strategy

DFT: Reconfiguration for Scan-Path

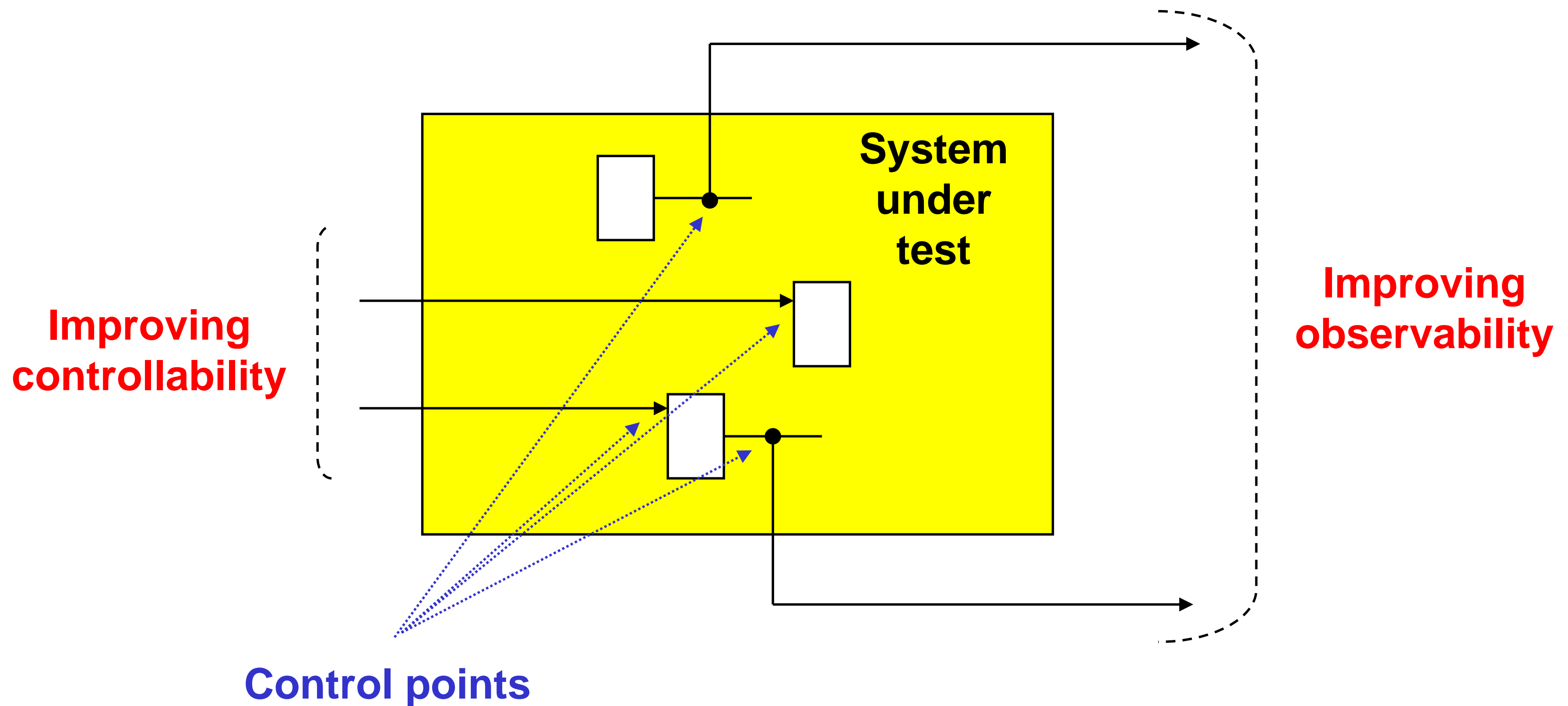


DFT: Boundary Scan Standard



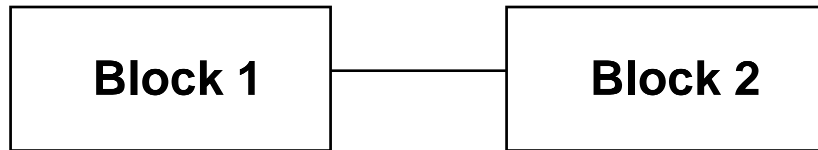
Two Tasks of DFT

To ways for improving testability with inserting of control points:



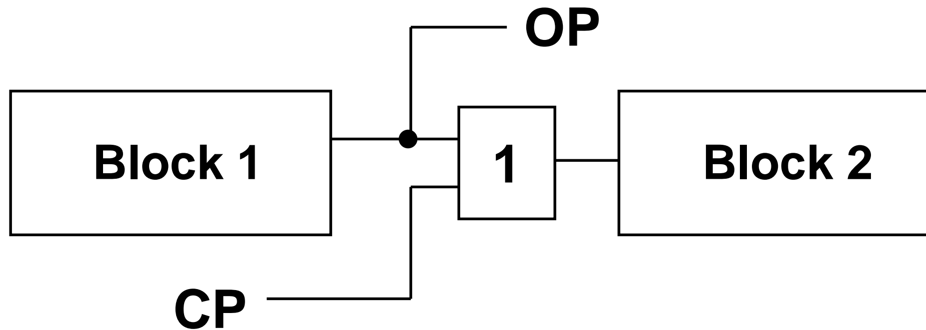
Two Tasks of DFT

Method of Test Points:

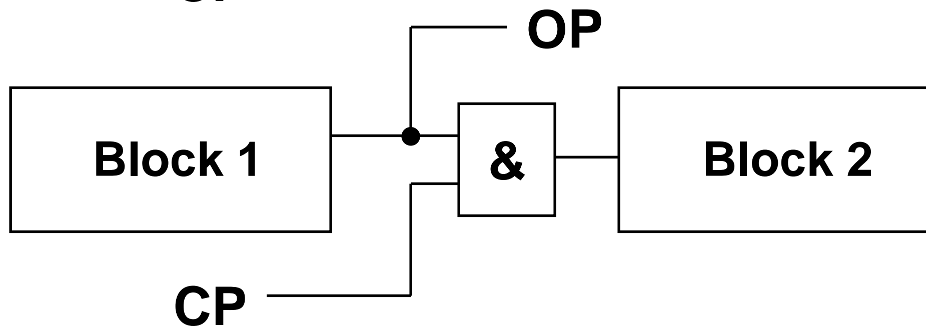


Block 1 is not observable,
Block 2 is not controllable

Improving controllability and observability:



1- controllability:
CP = 0 - normal working mode
CP = 1 - controlling Block 2 with signal 1



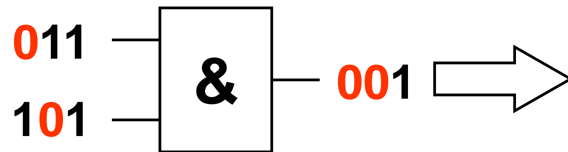
0- controllability:
CP = 1 - normal working mode
CP = 0 - controlling Block 2 with signal 0

Tradeoff Problems of DFT

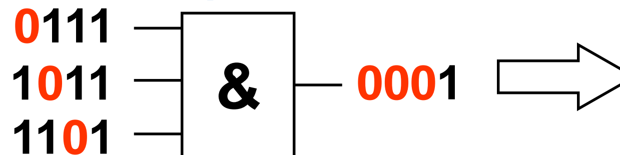
Amusing testability:

Theorem: You can test an arbitrary digital system by only 3 test patterns if you design it approprietly

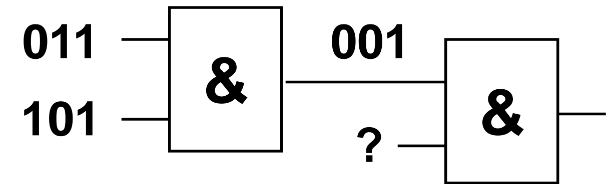
Proof: a) The case of 2-input AND

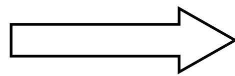


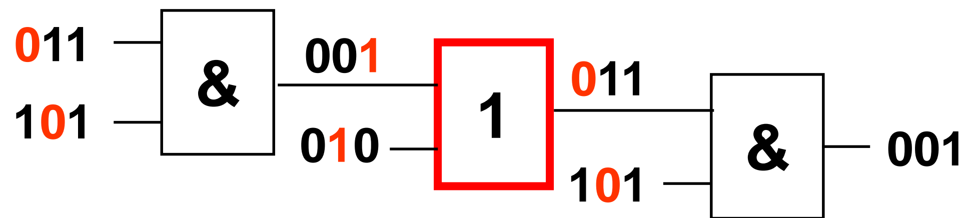
b) The case of 3-input AND



c) The case of a network







c) The general case of a digital system?

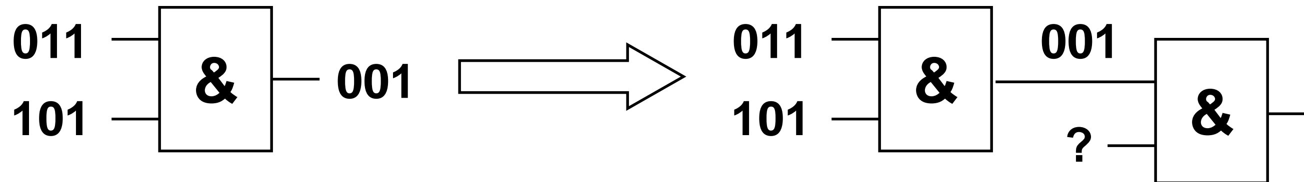
Any system ← Scan-Path ← FSM (?) ← Any CC ← NAND

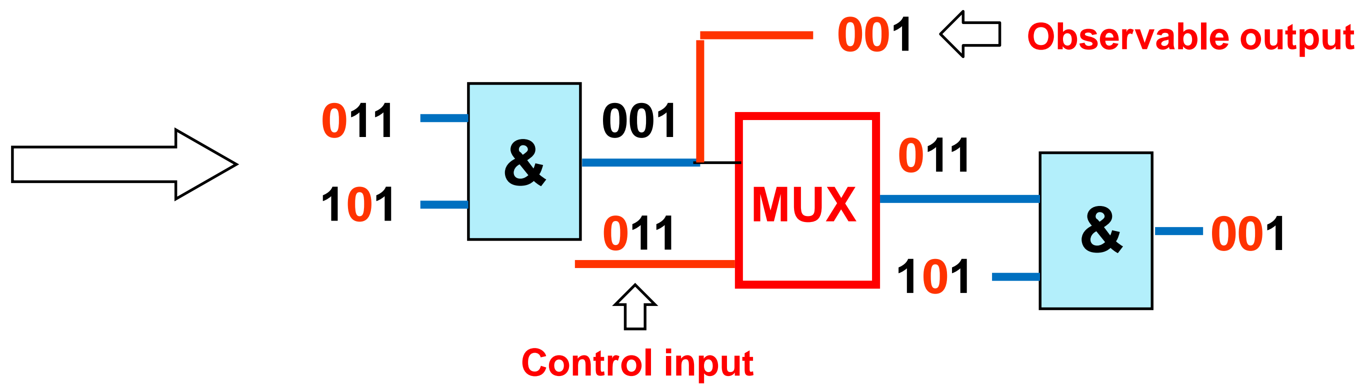
Tradeoff Problems of DFT

Direct local Design for Testability:

You can improve controllability and observability using MUX

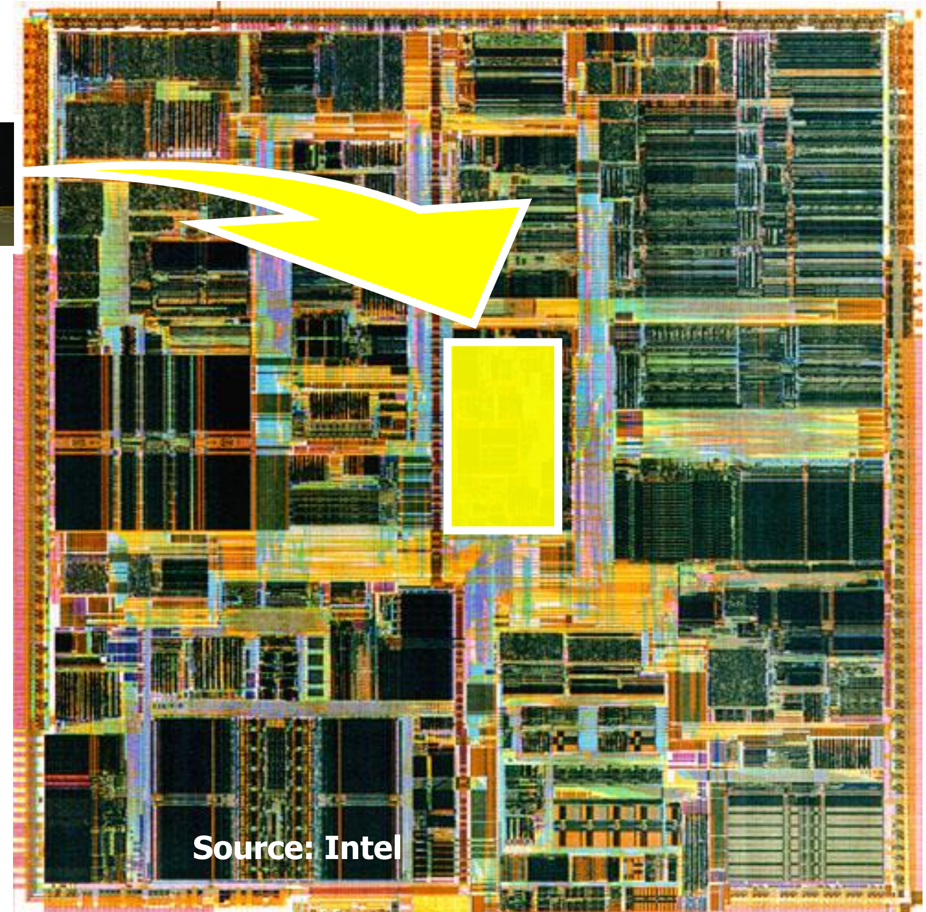
Proof:



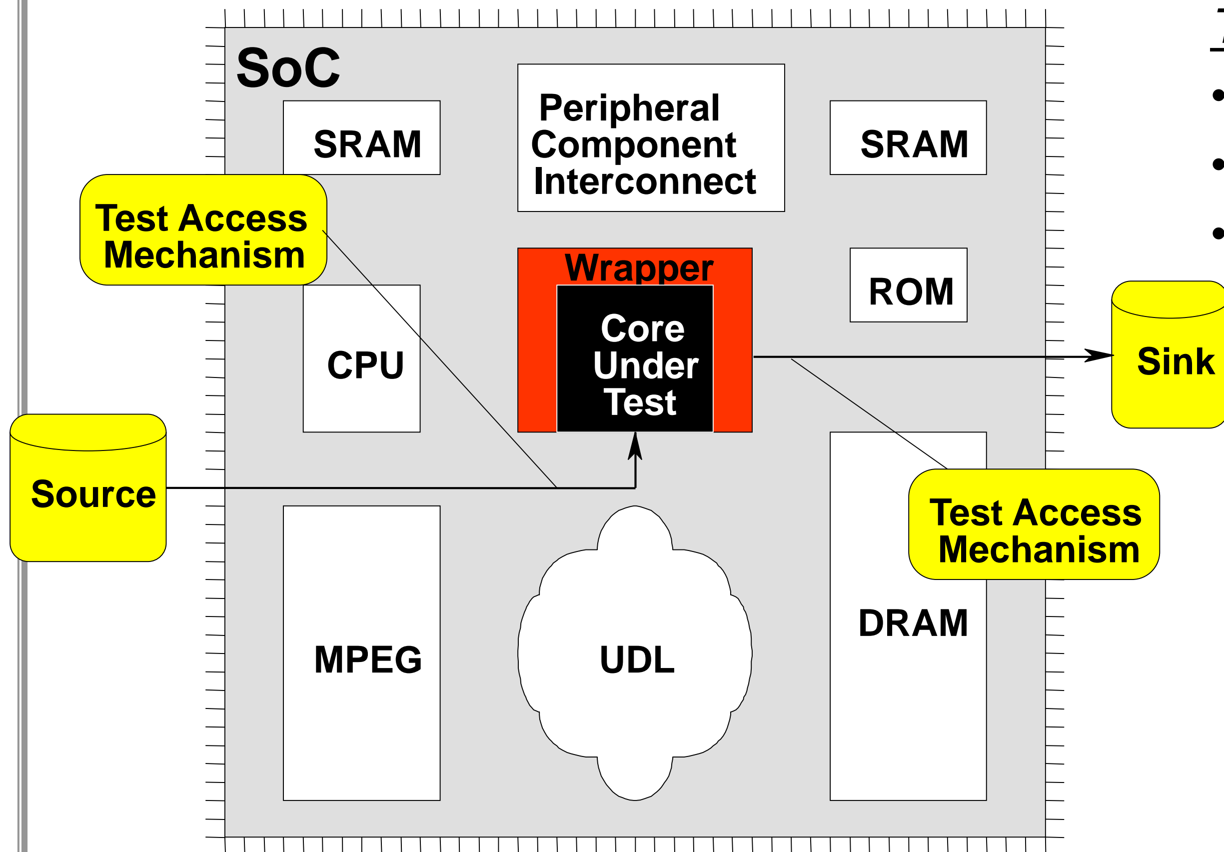


DFT: Built-in Self-Test (BIST)

Cores have to be tested on chip



BIST Components



Test architecture components:

- Test pattern source & sink
- Test Access Mechanism
- Core test wrapper

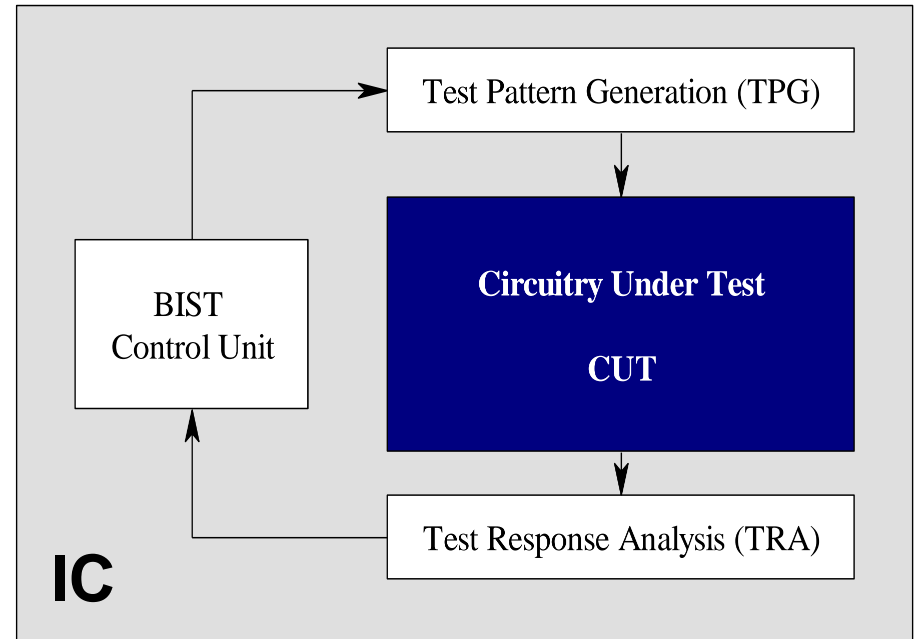
Solutions:

- Off-chip solution
  - need for external ATE
- Combined solution
  - mostly on-chip, ATE needed for control
- On-chip solution
  - BIST

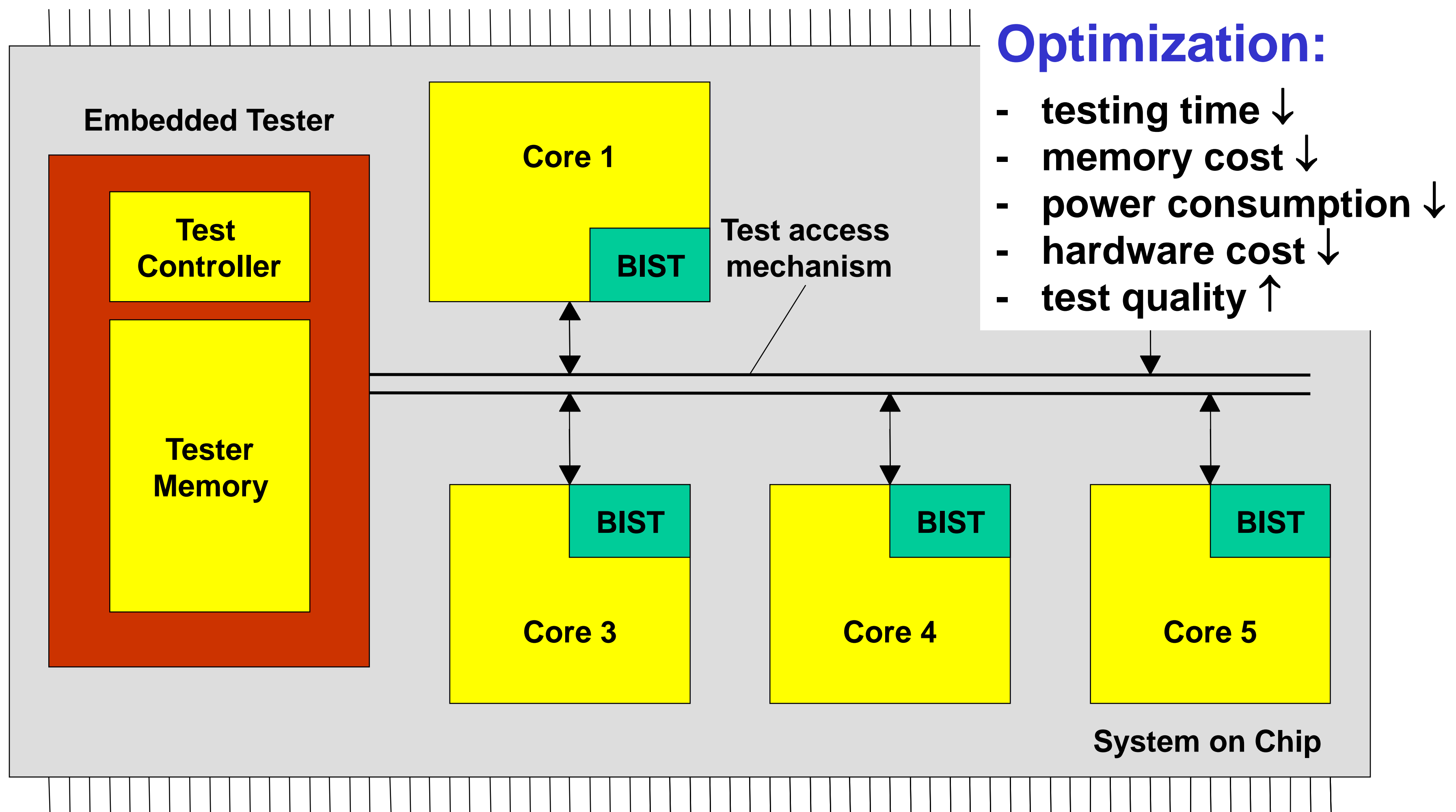
BIST Components

- On circuit
  - Test pattern generation
  - Response verification
- Random pattern generation,

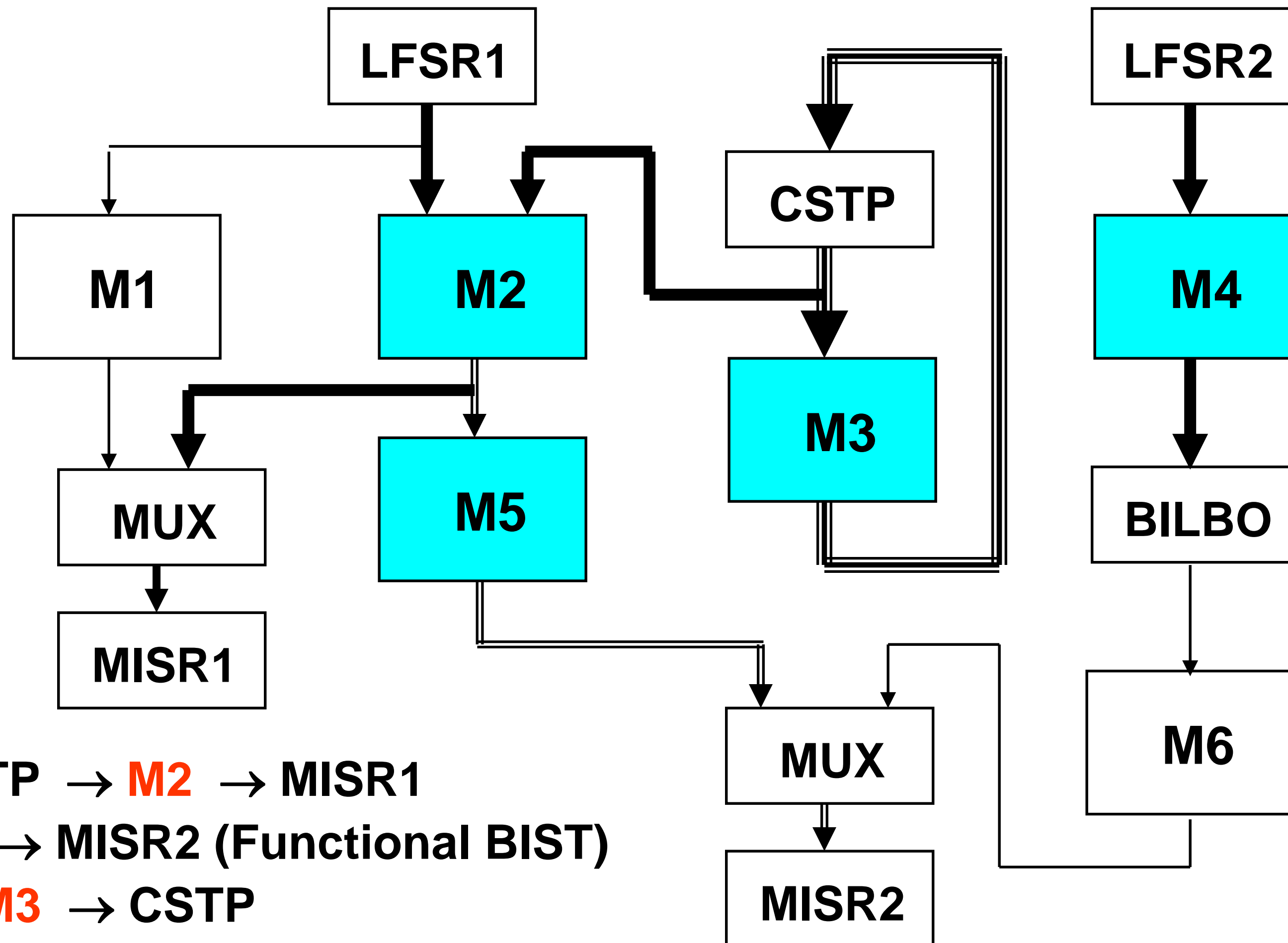
  very long tests
- Response compression



Distributed BIST and Criteria



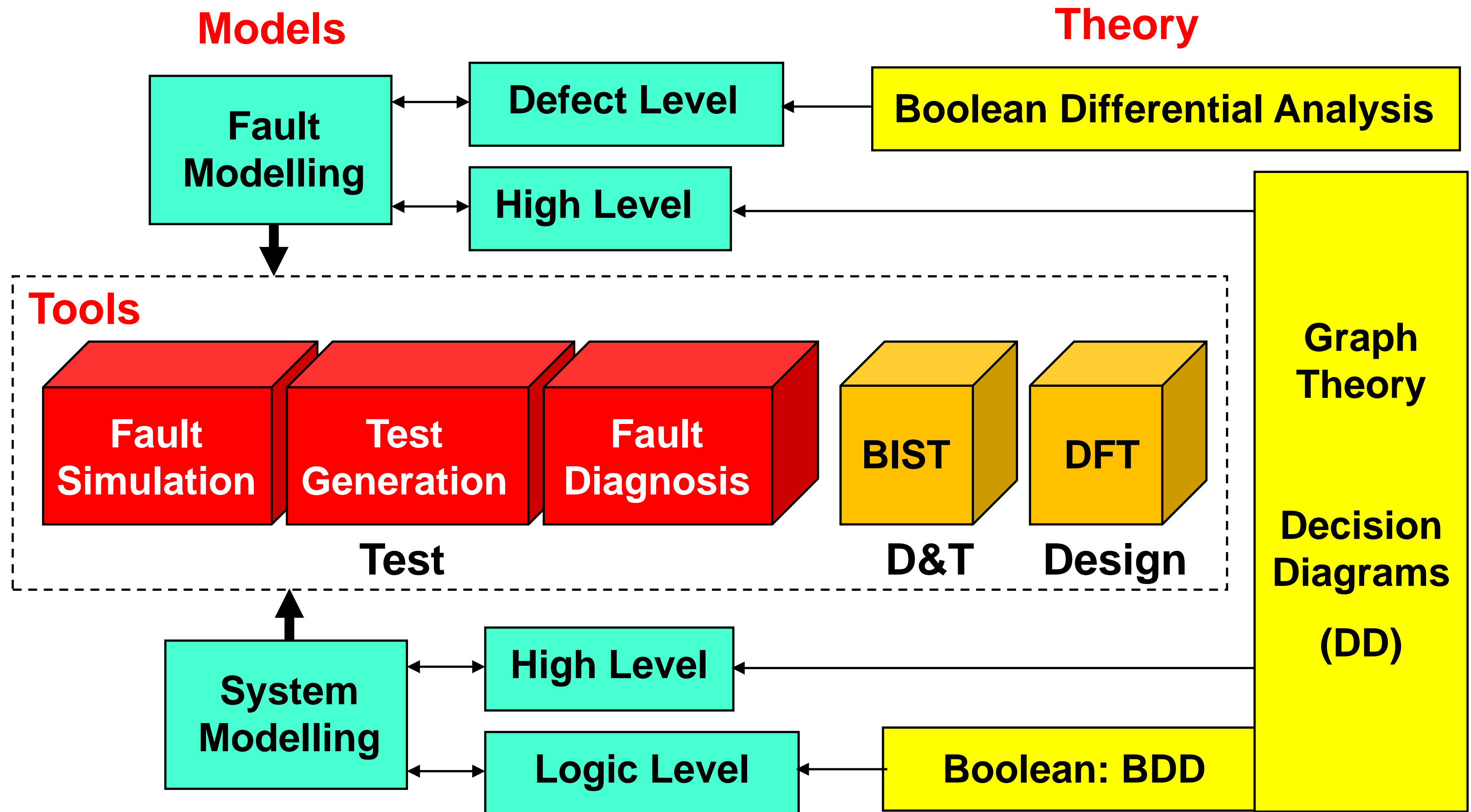
Distributed BIST Synthesis



Concurrent testing:

LFSR, CSTP → M2 → MISR1
M2 → M5 → MISR2 (Functional BIST)
CSTP → M3 → CSTP
LFSR2 → M4 → BILBO

Test Course Map



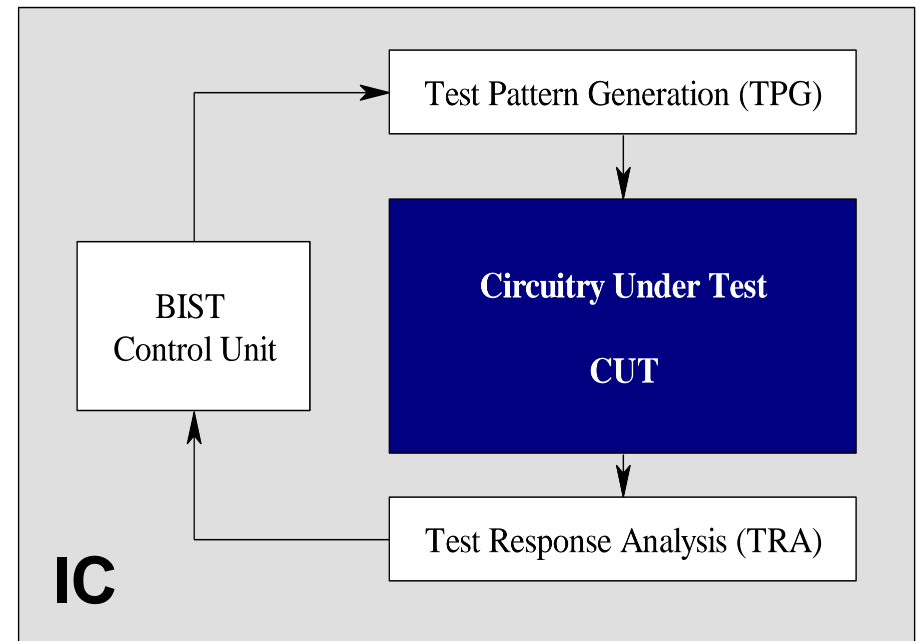
Course Work: Investigations of BIST

- **Design of a circuit**
- **Evaluation of the testability of the circuit**
  - Using fault simulation)
- **Redesign for testability**
  - Control points selection, optimization
- **Built-in self-test**
  - Design of solutions
- **Experimental research**

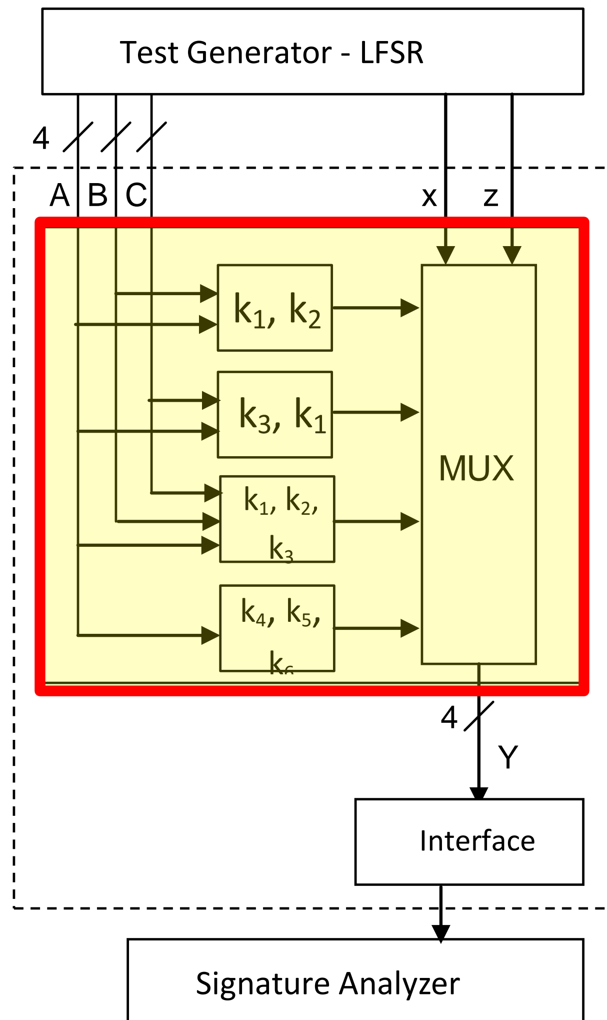
Course Work: Built-In Self-Test

Problems:

- BIST modeling and simulation
  - Test pattern generation (quality?, test length?)
  - Response verification
- Pseudorandom test
  - very long tests?
- Hybrid test solutions
- Response compression



Course Work: Research Object



1. Design of a combinational circuit for the following functionality

If $x = 0$, $z = 0$, then $Y = k_1A + k_2B$, else

if $x = 0$, $z = 1$, then $Y = k_3A - k_1C$, else

if $x = 1$, $z = 0$, then

$$Y = (k_1A \vee k_1B \wedge k_2C) \oplus (k_3C \vee \text{NOT}\,(k_3A) \wedge k_1B),$$

else

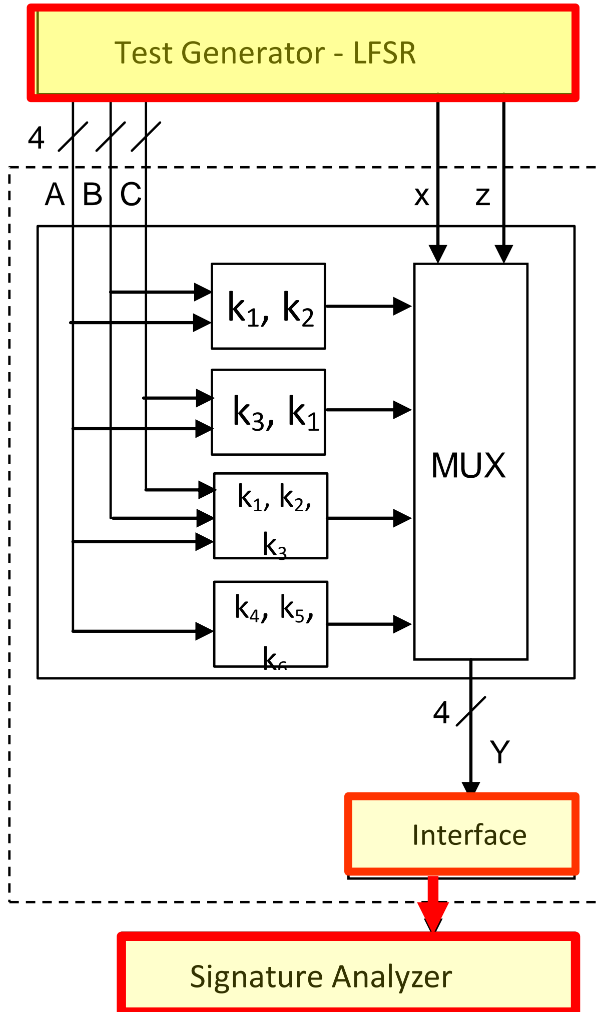
if $x = 1$, $z = 1$, then $Y = k_4A^2 + k_5A + k_6$

Coefficients k_i can be found on the next slide

Course Work: Versions of Research Object

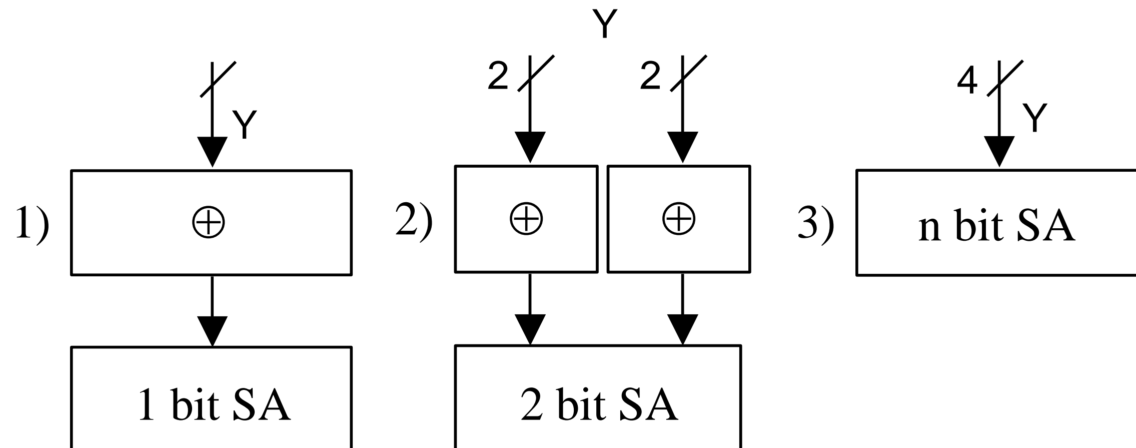
Vers. No.	k_1	k_2	k_3	k_4	k_5	k_6	Vers no.	k_1	k_2	k_3	k_4	k_5	k_6
1	1	1	1	0,1	0,2	0,5	8	1	1	1	1,5	0,1	0,5
2	1	1	0	0,1	0,2	1,0	9	1	1	0	1,5	0,1	1,0
3	1	0	1	0,1	0,2	2,0	10	1	0	1	1,5	0,4	2,0
4	1	0	0	0,1	0,2	3,0	11	1	0	0	1,5	0,4	3,0
5	0	1	1	0,1	1,0	0,5	12	0	1	1	1,5	0,8	0,5
6	0	1	0	0,1	1,0	1,0	13	0	1	0	1,5	0,8	1,0
7	0	0	1	0,1	2,0	2,0	14	0	0	1	1,5	1,5	2,0
15	1	1	1	0,2	0,4	1,0	20	0	0	1	0,5	0,4	1,0
16	1	1	1	0,1	0,4	1,2	21	0	0	1	1,0	0,8	1,5
17	1	1	0	0,2	0,4	0,5	22	0	1	1	0,5	1,2	0,5
18	1	0	1	0,2	0,4	1,0	23	0	1	1	1,0	0,3	1,0
19	0	0	1	0,3	2,0	1,0	24	1	0	1	0,4	1,0	1,5

Course Work: Design of Interface



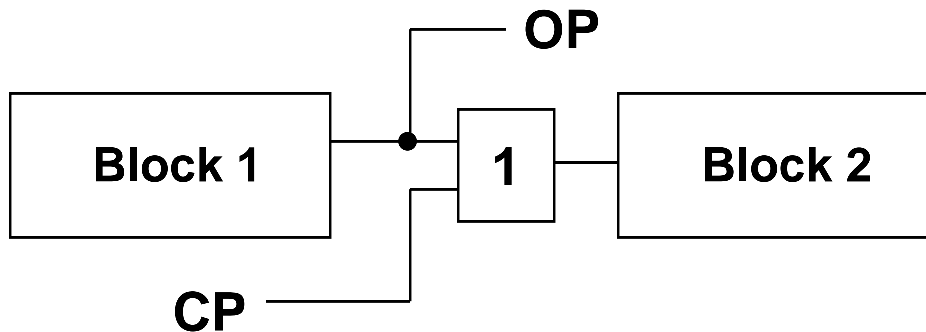
2. Design of the experimental bench. Use three different interface versions for experiments: 1 bit, 2-bit and 4- or more bit interfaces for respective n-bit Signature Analyzers

The types of interface:



Course Work: Design of a Testable Circuit

3. **Enter the** designed gate-level (AND, OR, NOT) **circuit into the computer** by CADENCE circuit editor
4. **Testability analysis**. Generate test patterns with Turbo-Tester (TT) ATPG. If the fault coverage is 100%, remove one or more patterns from the test set, so that at least two faults remain undetected
5. **Improve the testability of the circuit** to reach again 100% fault coverage with the updated test set



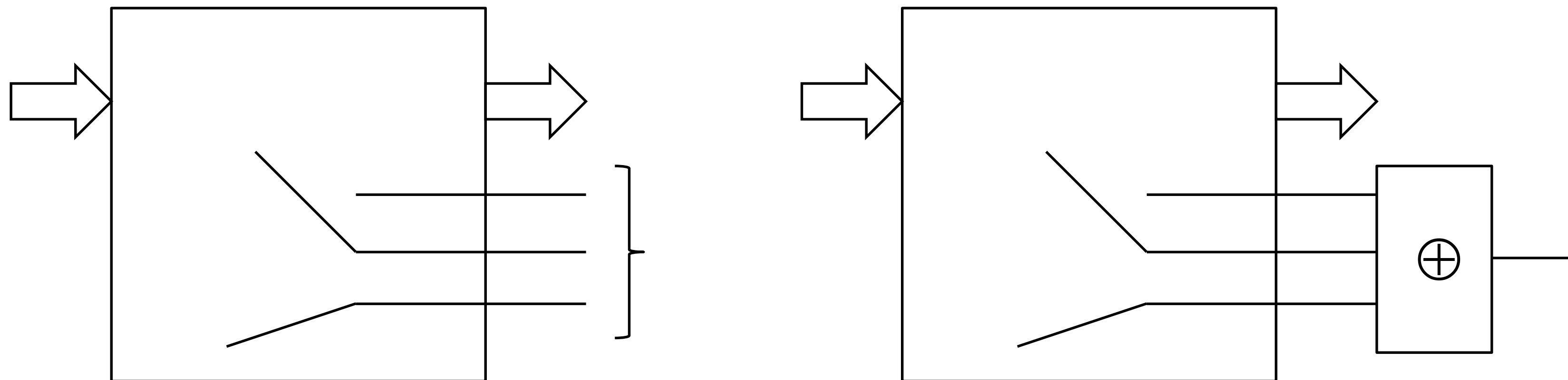
1- controllability:
CP = 0 - normal working mode
CP = 1 - controlling Block 2 with signal 1

Course Work: Observability Investigation

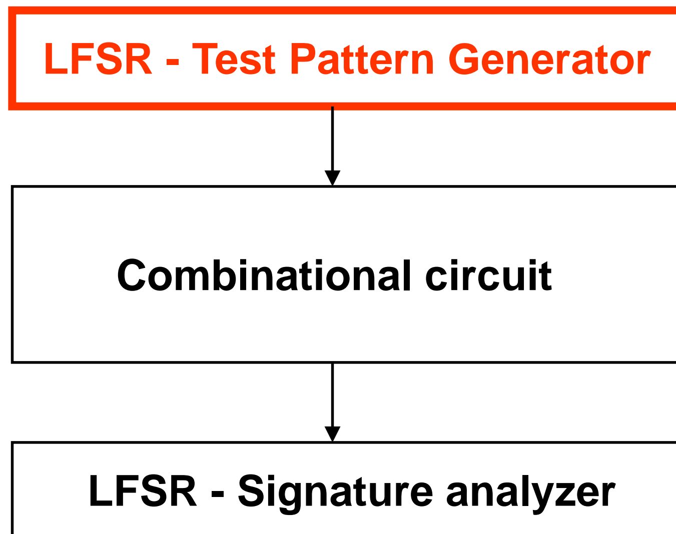
6. Analyze two different testability improvement solutions:

- **Separate pins for all observability points**
- **Single joint pin for all observability points**

Draw the graphics for both cases for the function $P = f(T)$ where P is fault coverage, and T is test length



Course Work: Design of a Test Generator



7. Generate test patterns by the BILBO tool for 10 different polynomials, and find the best structure for the LFSR

   Choose 5 primitive, and 5 non-primitive polynomials

   Report for all 10 experiments

   a) the maximum achievable fault coverage, and

   b) fix the minimum test length needed for that

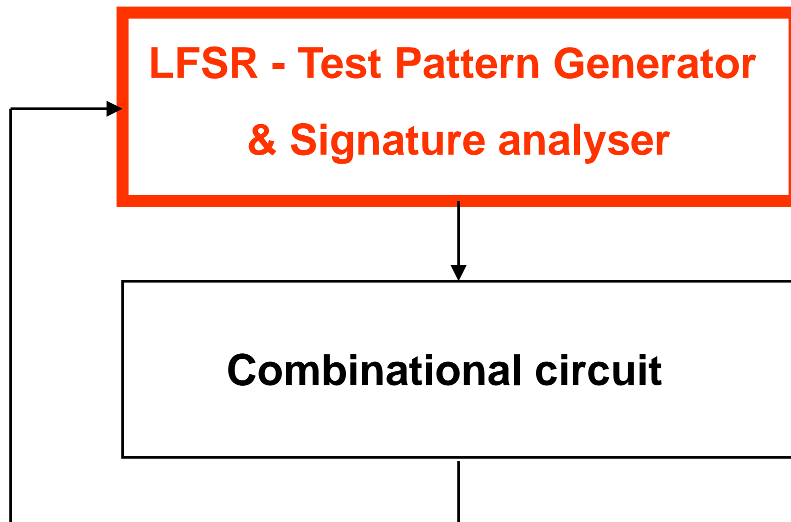
   Calculate the increase of the circuit size

   (in number of 2-input gates)

   due to adding of self-test circuitry

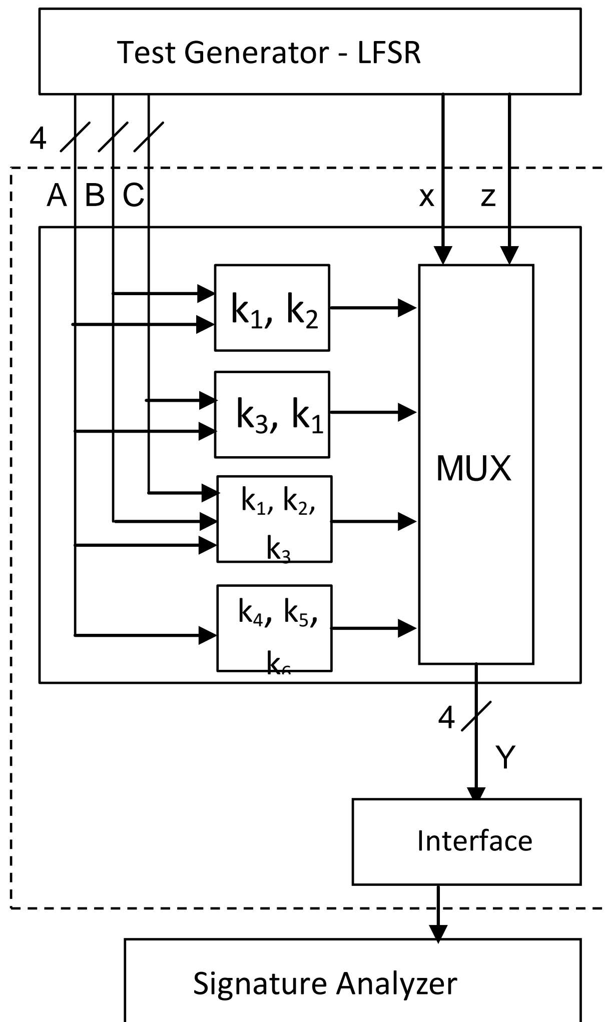
Course Work: Design of a Test Generator

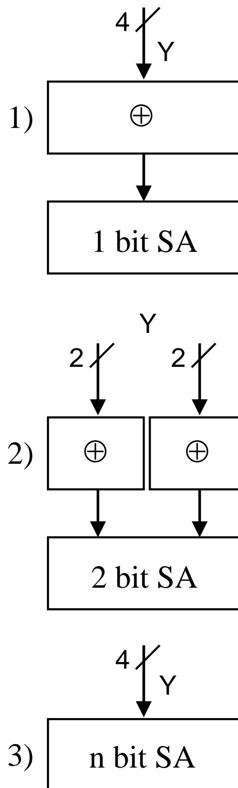
CSTP - Circular Self-Test Path:



8. Repeat the previous task for the case of using CSTP
   ("Circular Self Test Path")

Course Work: Design of a Response Analyzer





9. Carry out experiments with the best test set found in task 7

for 4 Signature Analyzers (SA):
1-bit, 2-bit, 4-bit, and 8-bit

Calculate the fault coverages

Draw the graphic *P* = *f*(*SA*)

P – is the fault coverage and
SA – is the number of bits in the Signature Analyzer

Draw 4 graphics *P* = *f*(*T*) for 4 SAs

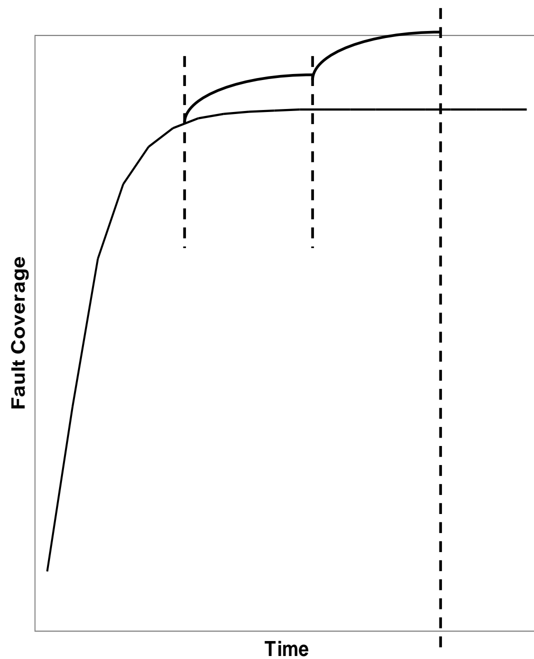
T (test length) – changes from 0% to 100% of fault coverage

Explain the graphics (dependence on two factors: test quality, and response analysis quality)

Course Work: Store-and-Generate BIST

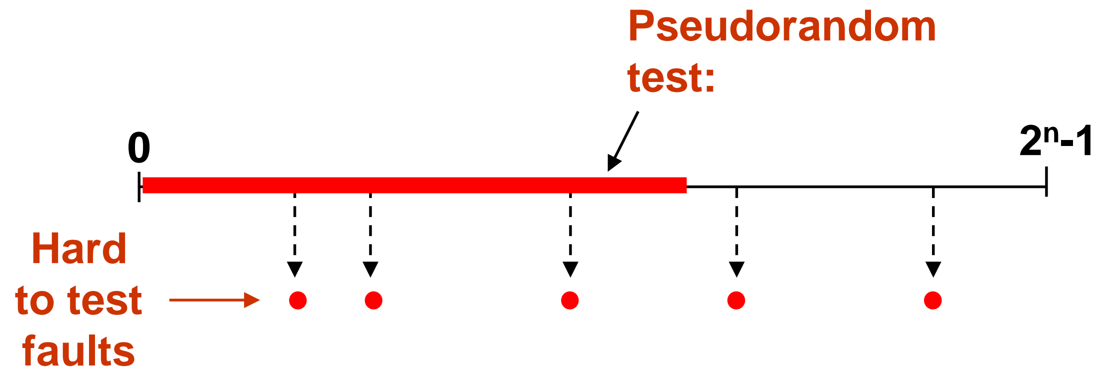
The main motivations of using random patterns are:

- low generation cost
- high initial efeciency

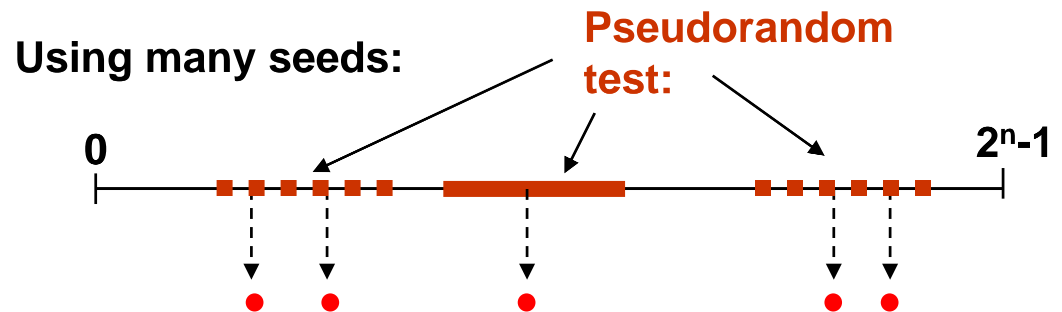


Problem: low fault coverage

Long PR test:

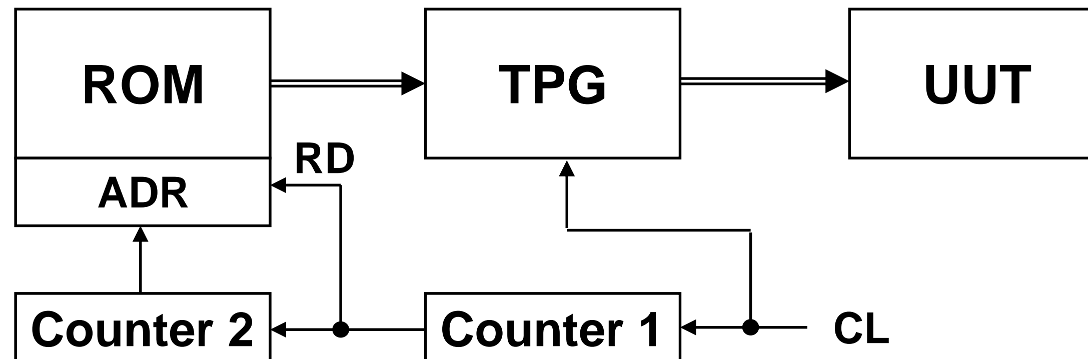


Using many seeds:



Course Work: Store-and-Generate BIST

10. **Synthesize an optimal BIST**, using "store & generate“ architecture. Choose for that the best BILBO structure and the 100% test with length *n*. Find the tradeoff between the number of seeds to be stored in the memory, and the test length



11. Compare the results in tasks 4, 5, 7, 8 and 10. Which solution is the best and why? Draw the block-level final structure of the selected best BIST solution.

12. Present a the results of experiments