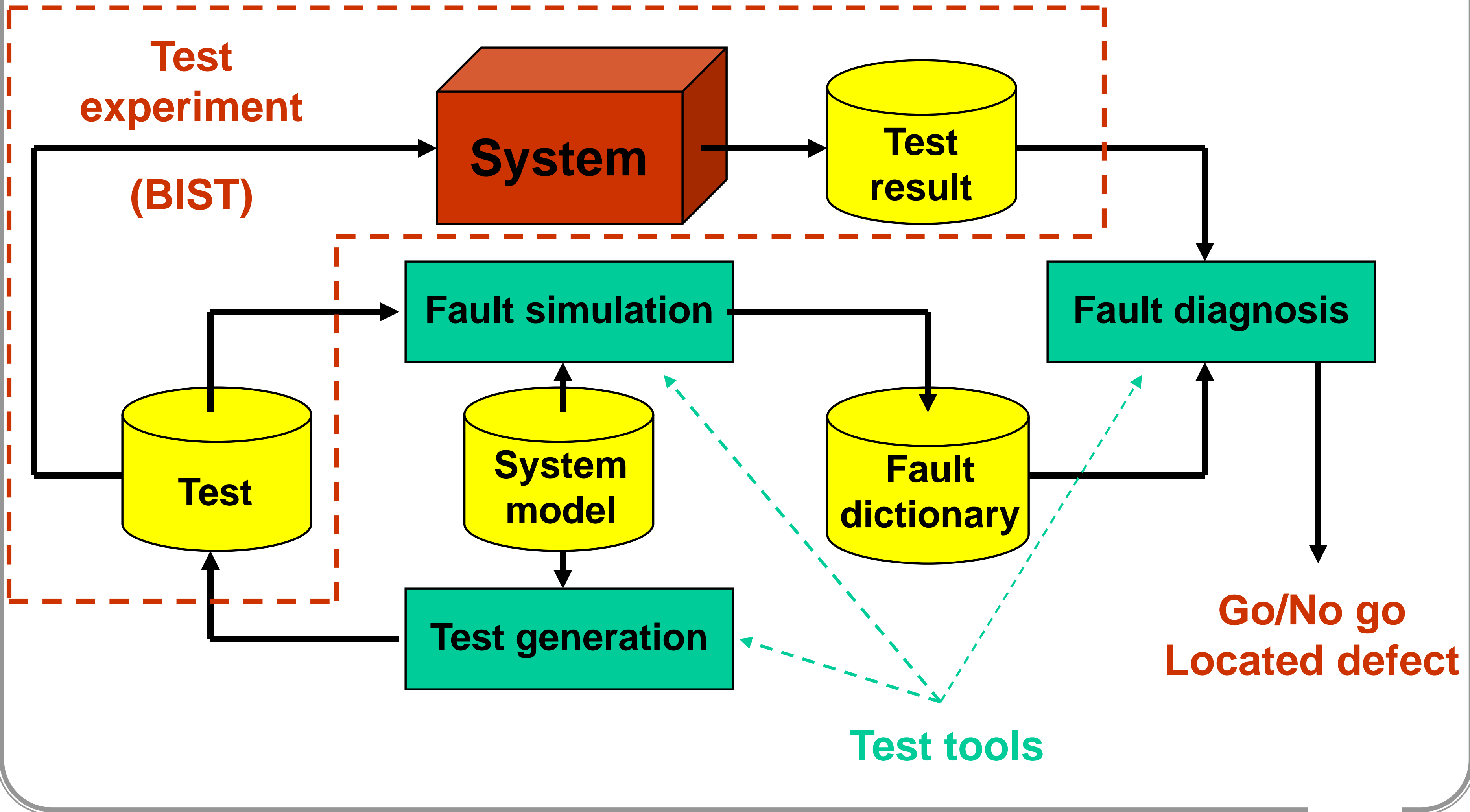
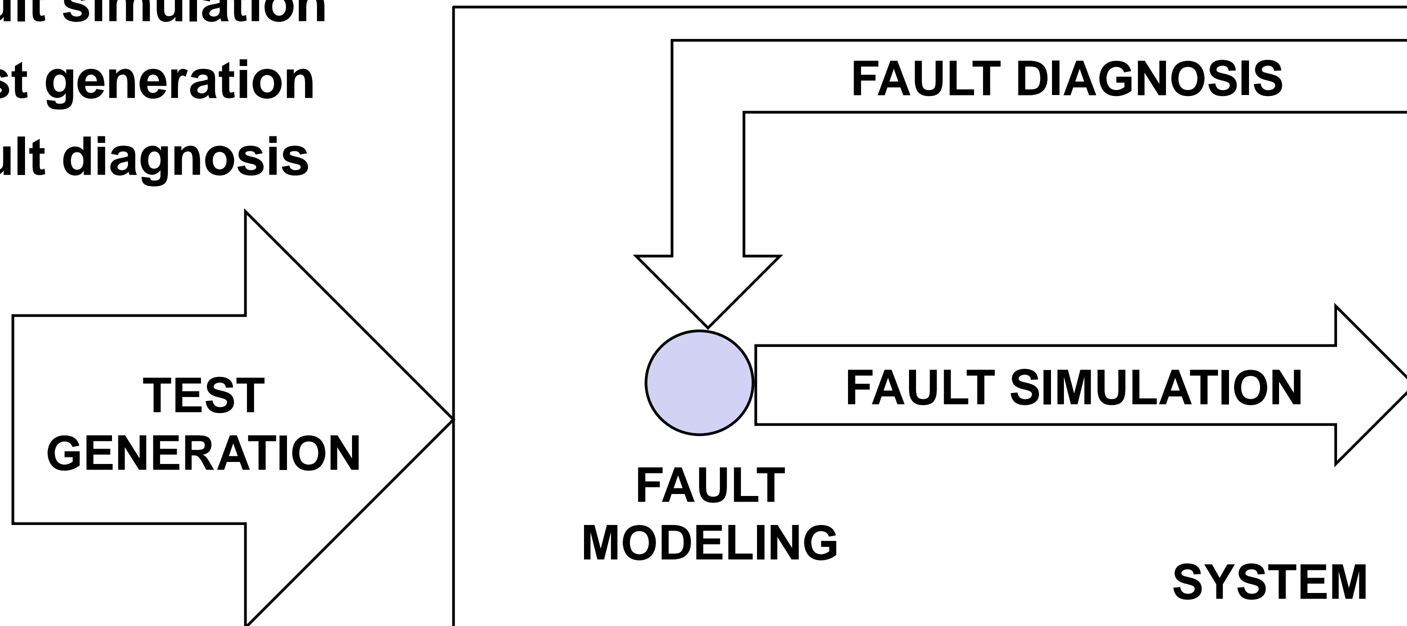
Overview about TESTING: Testing World



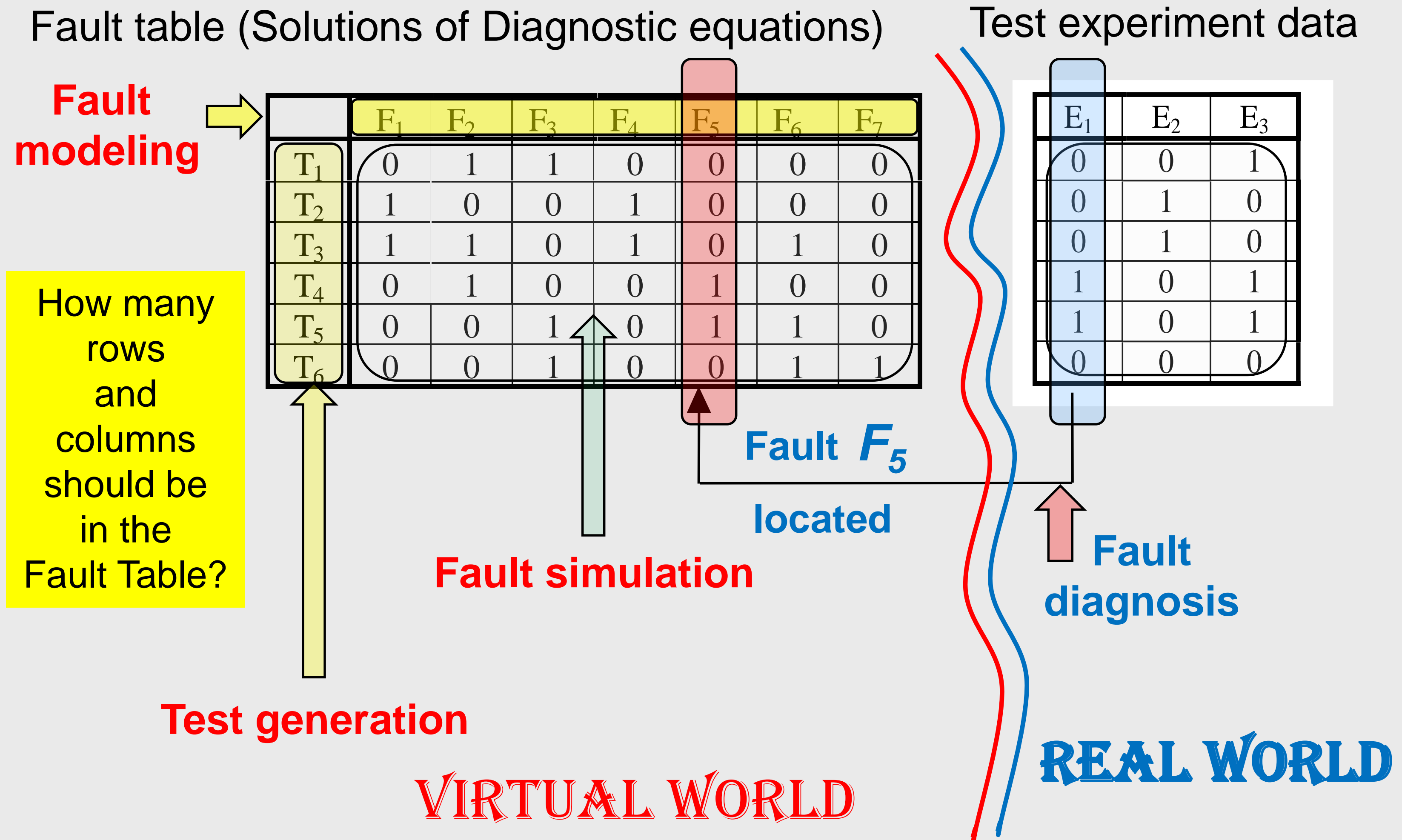
Test Related Basic Problems

Relationships between diferent test tasks

- **Fault modeling**
- **Fault simulation**
- **Test generation**
- **Fault diagnosis**



Test Related Basic Problems



	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

E_1	E_2	E_3
0	0	1
0	1	0
0	1	0
1	0	1
1	0	1
0	0	0

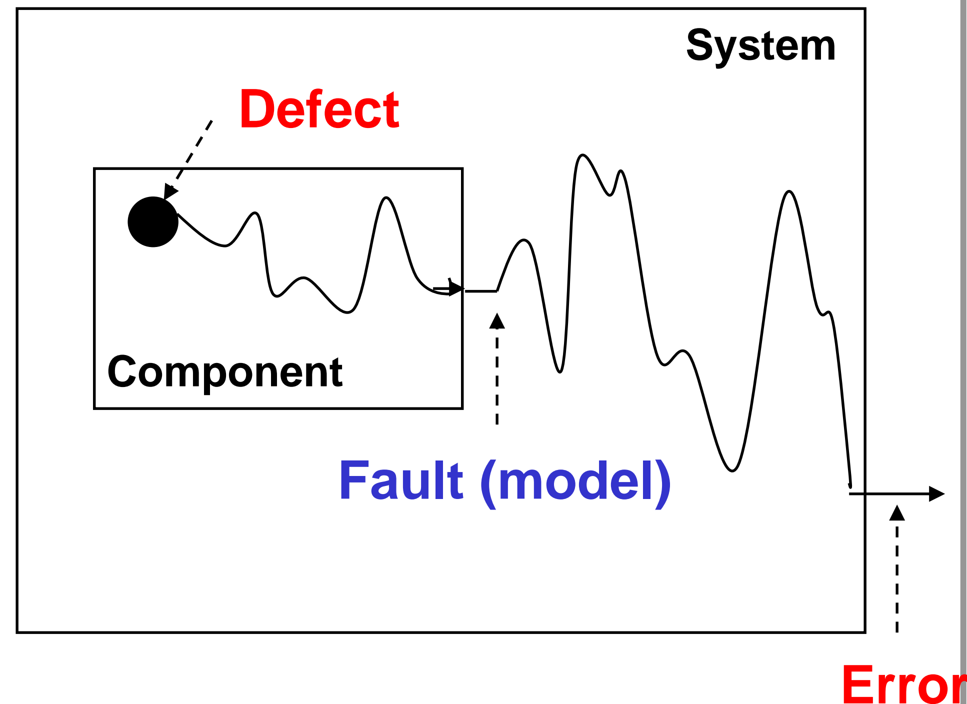
Why We Need Fault Models?

- Fault models are needed for
  - test generation,
  - test quality evaluation and
  - fault diagnosis
- To handle real physical defects is too difficult
- The fault model should
  - reflect accurately the behaviour of defects, and
  - be **computationably efficient**
- Usually combination of different fault models is used
- Fault model free approaches (!)

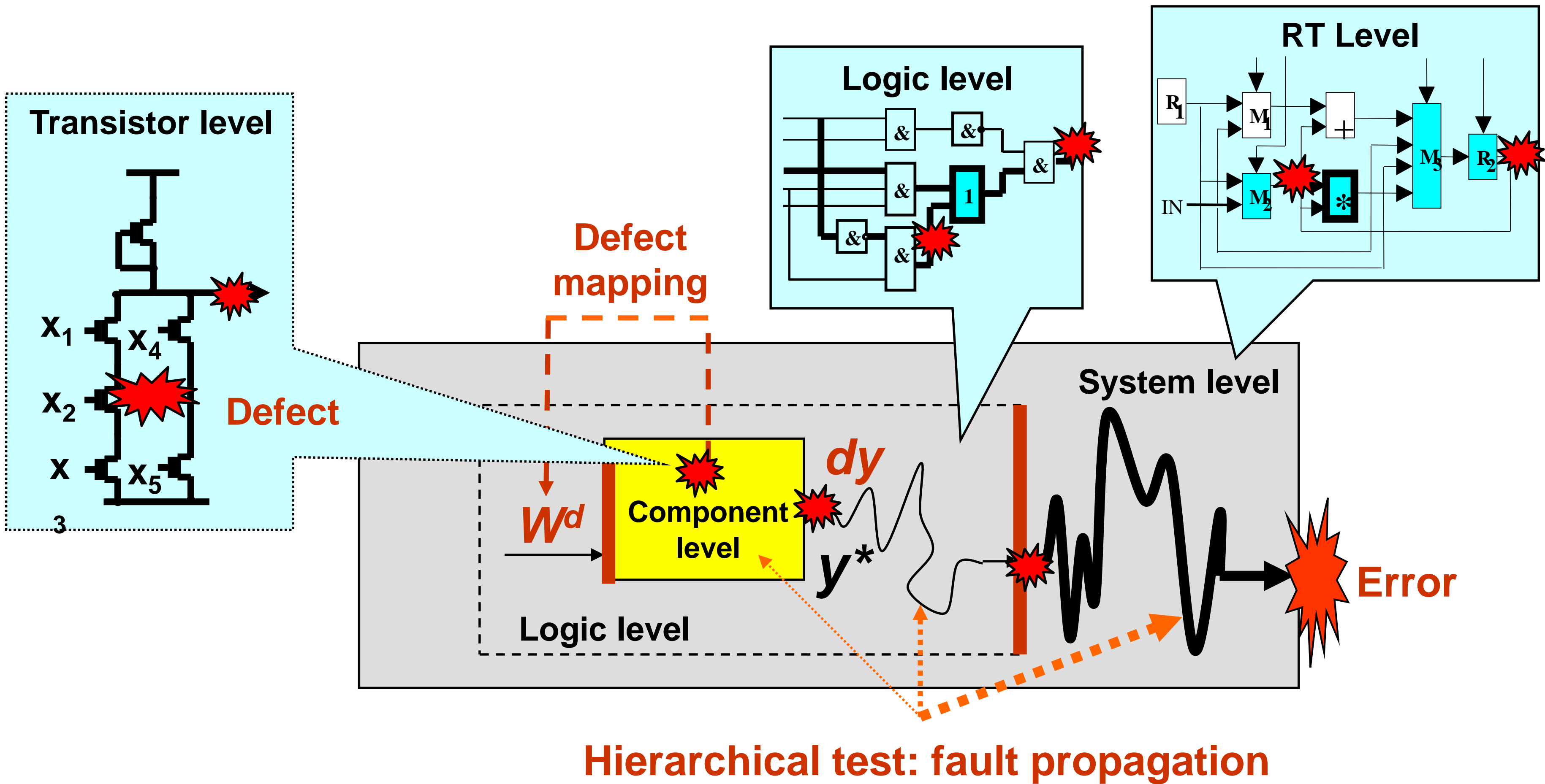
Fault Modeling: Defects and Faults

Defects, faults and errors

- An instance of an incorrect operation of the system being tested is referred to as an **error**
- The causes of the observed errors may be **design errors** or **physical faults (defects)**
- Physical defects do not allow a direct mathematical treatment of testing and diagnosis
- The solution is to deal with **logical fault models**



Divide & Conquer: Hierarchy is the Solution

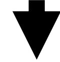


N-Dimensional Space of (**46?**) Test Problems

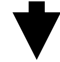
1 selected problems field

Test Generation:
Deterministic
Random
Genetic

Test Optimization:
Test Compaction
Testability Design
Signature Analyzis



Testing:
External Test
Built-In Self-Test



Fault Diagnosis:
Combinational
Sequential
Fault dictionaries

Fault Simulation:
Single Fault
Parallel Fault
Parallel Pattern

Fault Analysis:
Deductive
Concurrent
Reversive

System classes:
Combinational circuits
Sequential circuits
Finite State Machines
Data Flow Circuits
Microprocessors
Memories
Systems on Chip

Timing:
Synchronous
Asynchronous
Multi-Clocking

Models:
Structural
Functional
Procedural
Behavioral

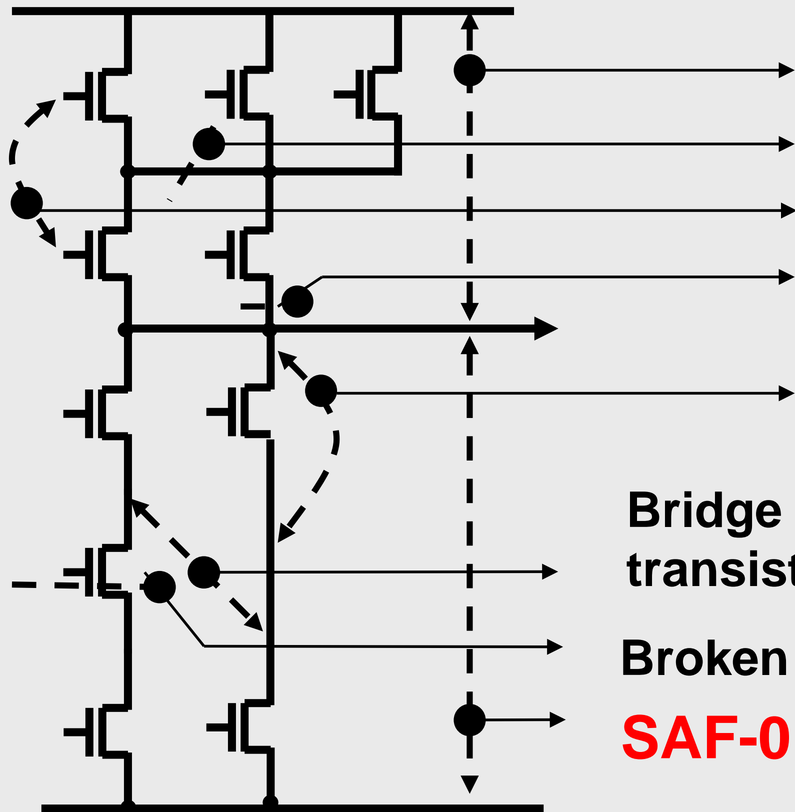
Hierarchy:
Bottom-up
Top-down
Jo-jo

Fault classes:
Stuck-at-Faults
Bridging faults
Opens
Delay Faults
Physical defects
Functional Faults

Fault Problems:
Multiple Faults
Fault Masking
Fault Dominance
Fault Equivalence
Fault Collapsing
Redundancy

12 dimensions

Faults and fault models



SAF-1

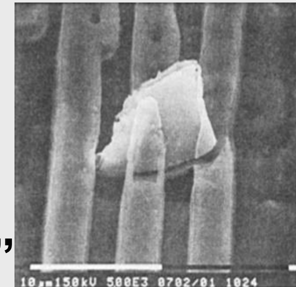
Broken wire

Bridge between wires

Wire "is hanging in then air "

Capacity → Sequential behavior

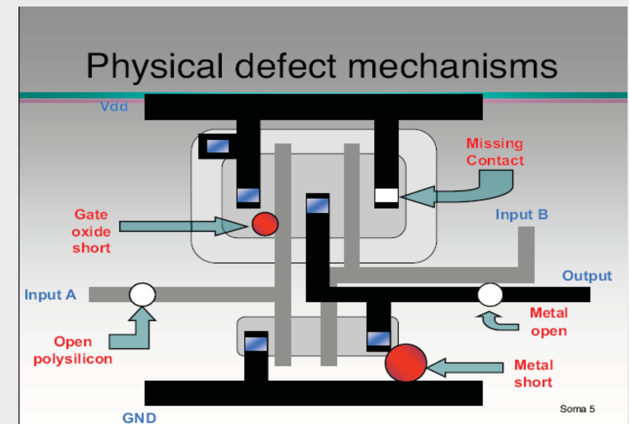
Short in the transistor



Bridge between transistors

Broken transistor

SAF-0

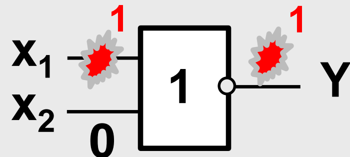


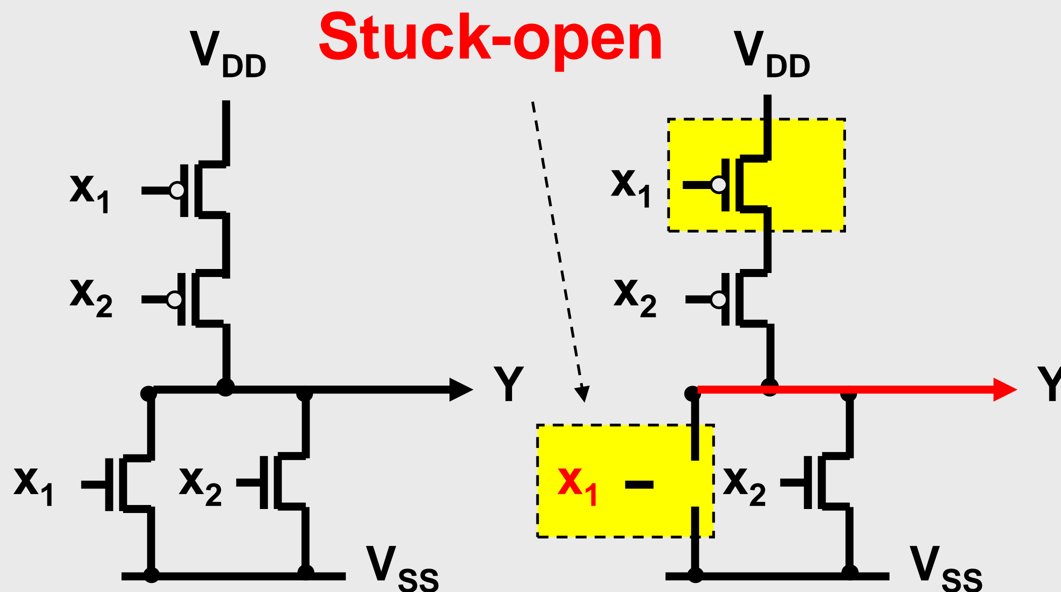
© Mani Soma

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Transistor Level Stuck-open Faults

NOR gate





x_1	x_2	y	y^d
0	0	1	1
0	1	0	0
1	0	0	Y'
1	1	0	0

Test sequence is needed: 00,10

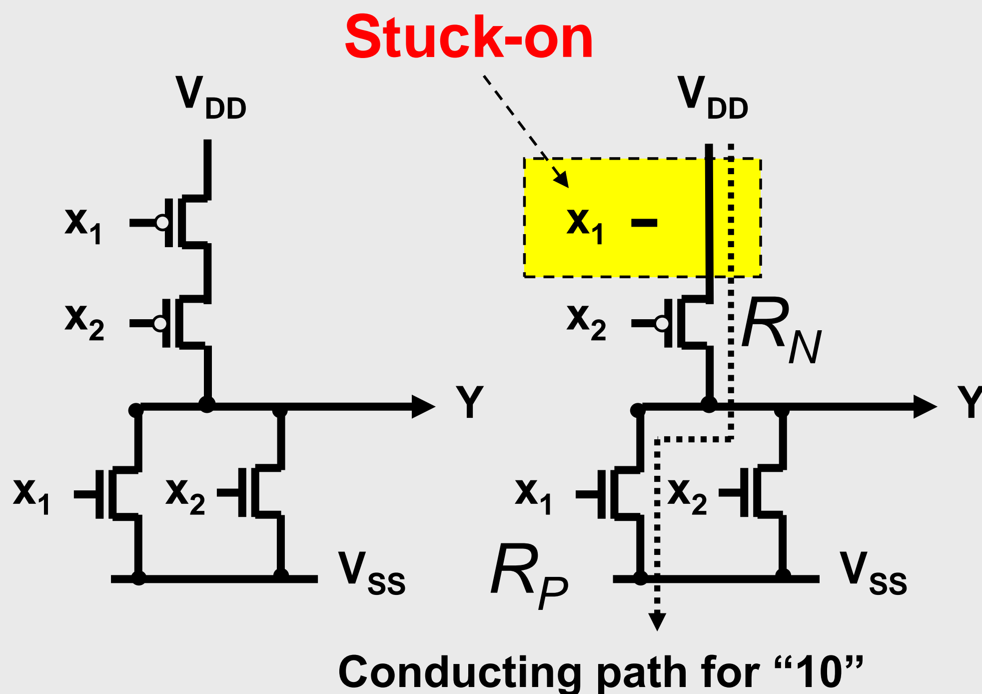
No conducting path from V_{DD} to V_{SS} for "10"

The wire Y is floating → Capacity is working as a memory

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Transistor Level Stuck-On Faults

NOR gate



x_1	x_2	y	y^d
0	0	1	1
0	1	0	0
1	0	0	V_Y
1	1	0	0

$$V_Y = \frac{V_{DD} R_P}{(R_P + R_N)}$$

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Fault Modeling: Logic Level Faults - SAF

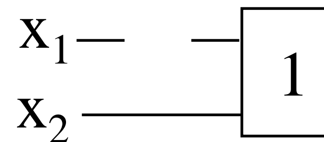
Why logic fault models?

- complexity of simulation reduces (many physical faults may be modeled by the same logic fault)
- one logic fault model is applicable to many technologies
- logic fault tests may be used for physical faults whose effect is not completely understood
- **they give a possibility to move from the lower physical level to the higher logic level**

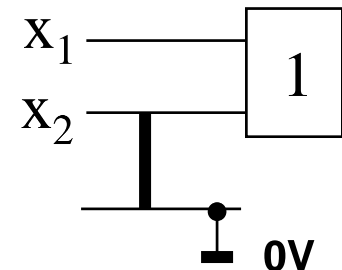
Stuck-at fault model:

Stuck-at-0

Broken line

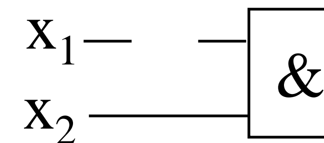


Bridge to ground

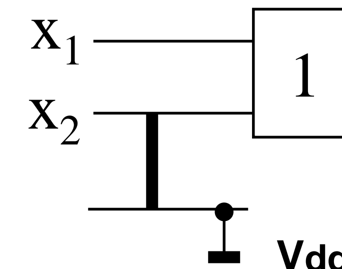


Stuck-at-1

Broken line



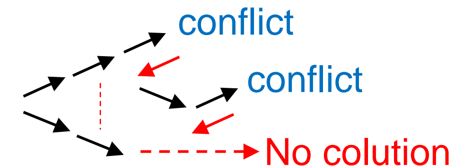
Bridge to Power Supply



Fault Cover and Fault Redundancy

- **Fault cover is a measure of the number of detectable faults**
- **Why fault redundancy is an important and troublesame problem**
  - **It makes test generation (search for a proper test pattern for the given fault extremely time consuming)**
    - *n* – number of inputs of the circuit
    - The search space is 2^n where **backtracks are needed**
    - If 64 inputs, then the search space is $2^{64} = 10^{19}$
    - **Redundant fault needs the search throughout the full search space**

      If not done, the fault cover cannot be calculated trustworthy
    - F – number of all faults
    - F_D – number of detected faults
    - FC – fault coverage
    - F_R – number of redundant faults
    - TE – test efficiency
  - **Fault coverage:** $FC = F_D / F$
  - **Test efficiency:** $TE = F_D / (F - F_R)$



Example:

Faults: $F = 1000$

Redundant faults: $FR = 100$

Detected faults: $FD = 880$

Fault coverage: $FC = 880/1000 = 88\%$

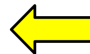
Test efficiency: $TE = 880/900 = 98\%$

Contradiction: between fault tolerance and fault coverage

Problems with Testing: Multiple Faults

- Multiple stuck-fault (MSF) model is an extension of the single stuck-fault (SSF) where several lines can be simultaneously stuck
- If n - is the number of possible SSF sites, there are $2n$ possible SSFs, but there are

But 3^n -1 possible MSFs



Wire a 0,1,x

Wire b 0,1,x

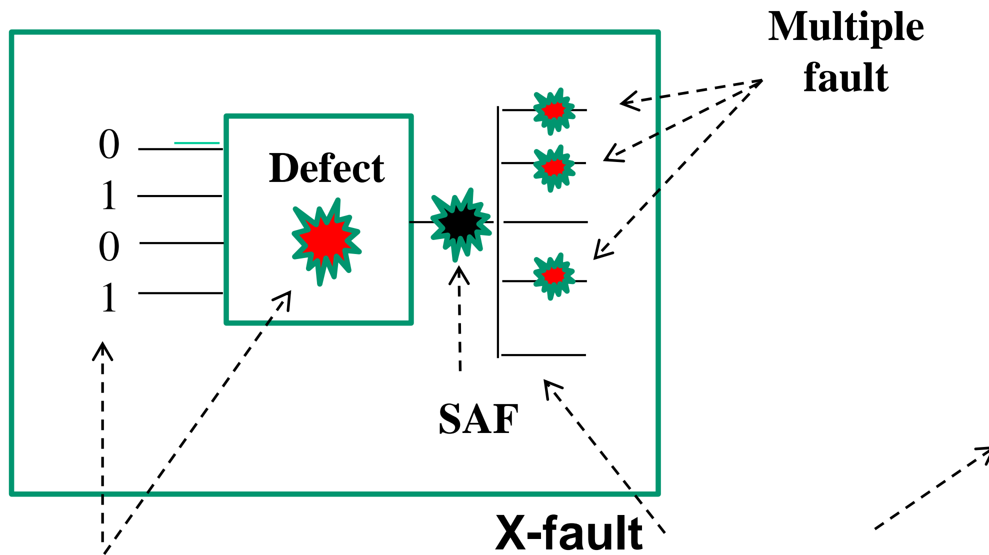
- If we assume that the multiplicity of faults is no greater than k, then the number of possible MSFs is

$$N = \sum_{i=1}^{k} \{C_n^i\} 2^i \ll 3^n - 1 \qquad C_n^i = \frac{n!}{i!(n-i)!}$$

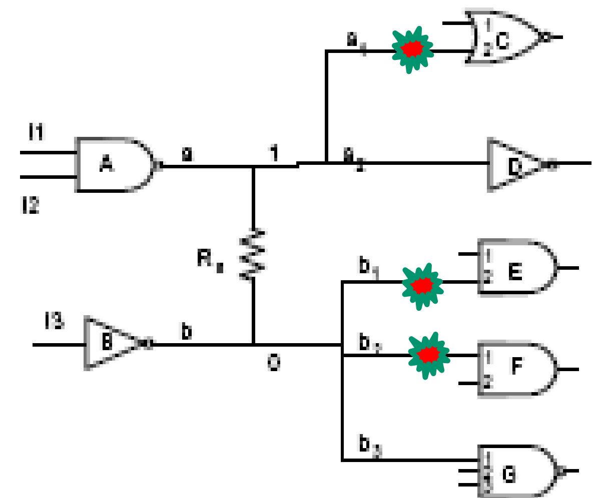
- C^i_n - number of sets of i lines, 2^i – number of faults on the set

Fault Modeling: Conditional SAF

Very complex faults can be translated (reduced) to SAF model:



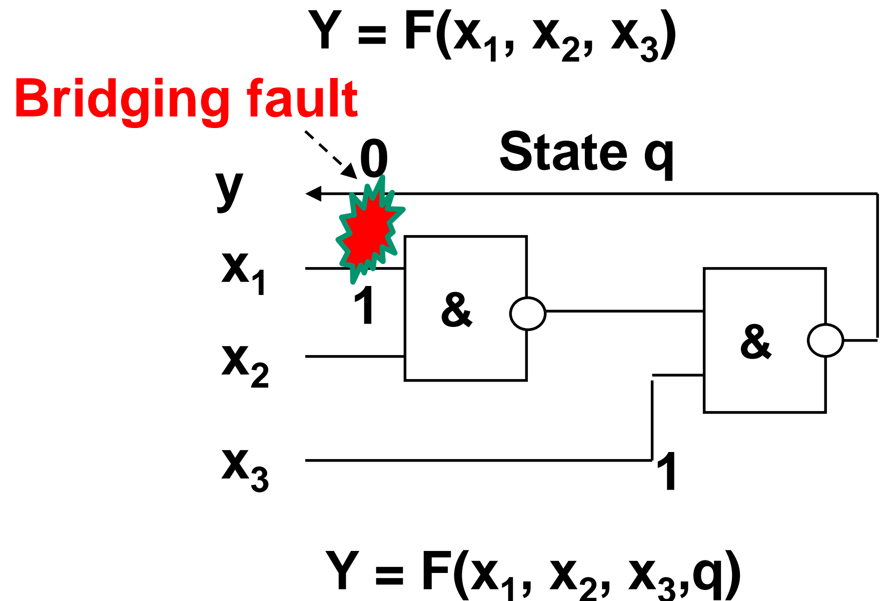
Resistive bridge fault



Conditional fault
Pattern fault
Constrained SAF
Single faulty signal

X-fault
Byzantine fault
Bridges
Stuck-opens
Multiple faulty signal

Gate-Level Faults: SAF Model



Complexity problem:

Number of single faults – **2n**
Number of **multiple** faults – **3^n -1**
n – is number of wires

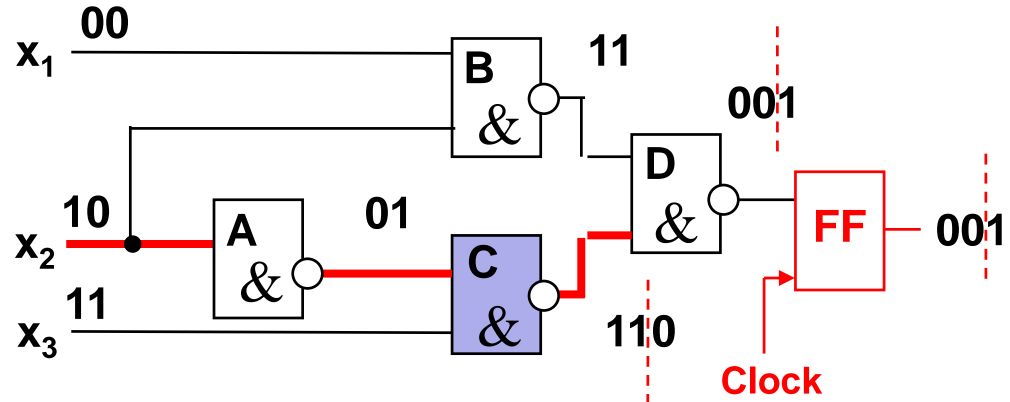
Things are even worse

Delay Fault Models

Delay faults are tested by test pattern pairs:

- the first test pattern **initializes** the circuit, and

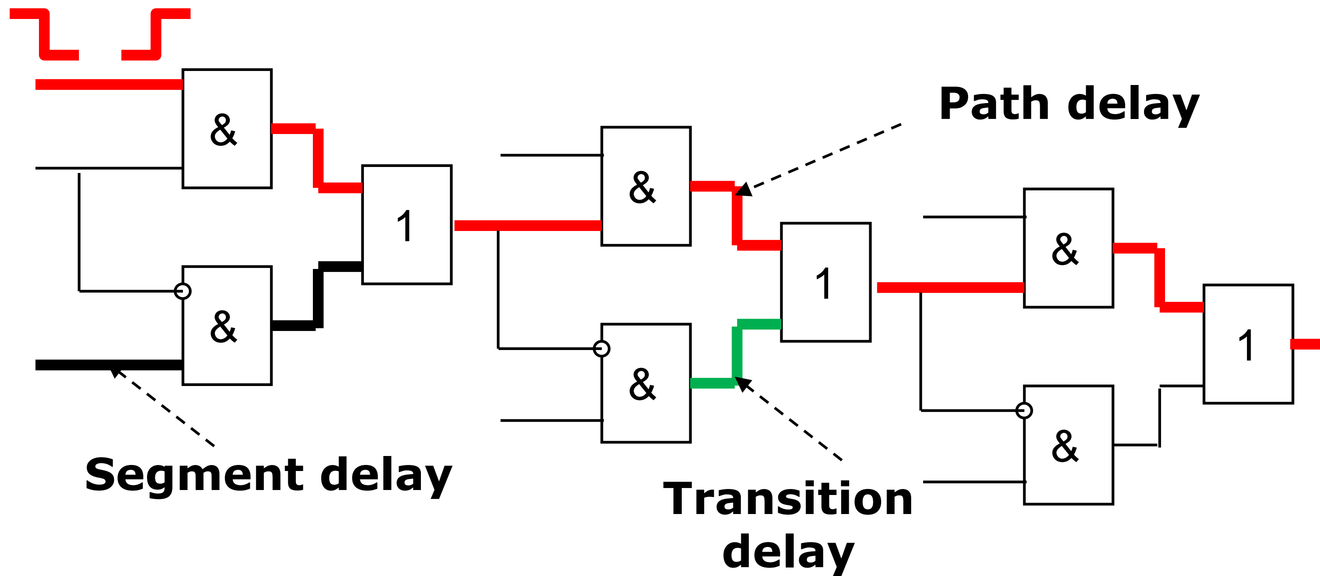
- the second pattern **sensitizes** the fault



Delay fault models:

- Gate delay fault (delay fault is lumped at a single gate, quantitative model)
- Transition fault **(qualitative model, gross delay fault model, independent of the activated path)**
- Path delay fault **(sum of the delays of gates along a given path)**
- Line delay fault (is propagated through the longest senzitizable path)
- Segment delay fault (tradeoff between the transition and the path delay fault models)

Transition Delay and Path Delay Faults



Numbers of faults:

- **Transition faults**: 2*(18 lines)=36
- Segment delay faults: 2*(12 segments) = 24
- **Path delay daults**: 2*(22 paths) 44

Number of Paths in Circuits

ISCAS'85 Family of benchmark circuits:

Circuit	Inputs	Outputs	Gates	Levels	Paths
c17	5	2	13	4	11
c32	36	7	203	18	83926
c499	41	32	275	12	9440
c880	60	26	469	25	8642
c1355	41	32	619	25	4173216
c1908	33	25	938	41	729057
c2670	233	140	1566	33	679960
c3540	50	22	1741	48	28676671
c5315	178	123	2608	50	1341305
c6288	32	32	2480	125	**98943441738294937238**
c7552	207	108	3827	44	726494

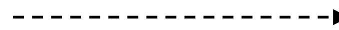
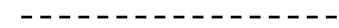
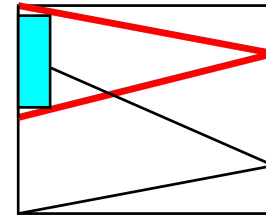
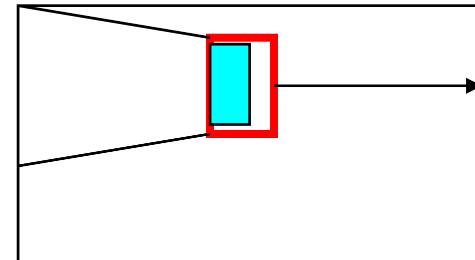
Comparison of Delay Faults

Fault models	Advantages	Limitations
Gate delay	All gates can be modeled	• Distributed failures not considered • Exact defect size not possible
Transition fault	Easy to model all gates	Distributed failures not considered
Path delay	Distributed failures considered	Impossible to enumerate all paths
Line delay	• All gates are modeled • Distributed failures considered • Better coverage metric • Additional fault coverage by using multi-path technique	• Existence of nonrobust test • May fail for some shorter paths
Segment delay	Considers general delay defect from spot to distributed failures	Longest delay path may not be tested

Copyright © A.K.Majhi, V.D.Agrawal 1997

Universal Functional Faults

Exhaustive combinational fault model:

- exhaustive test patterns 
- pseudoexhaustive test patterns
  - exhaustive **output line oriented** test patterns  
  - exhaustive **module oriented** test patterns 

Advantage: The way to hierarchical approach and to „conquer and divide“ strategy

Fault Modeling: Register Level Faults

vs. Boolean formulas with Boolean variable and SAF model ($x \equiv 0$, $x \equiv 1$)

RTL statement:

$$K: (\mathit{If}\ T, C)\quad R_D \leftarrow F(R_{S1}, R_{S2}, \ldots R_{Sm}), \rightarrow N$$

Components of the statement: (variable types)

K	- label
T	- timing condition
C	- logical condition
R_D	- destination register
R_S	- source register
F	- operation (microoperation)
$\leftarrow$	- data transfer
$\rightarrow N$	- jump to the next statement

RT level faults:

$K \rightarrow K'$	- label faults
$T \rightarrow T'$	- timing faults
$C \rightarrow C'$	- logical condition faults
$R_D \rightarrow R_D$	- register decoding faults
$R_S \rightarrow R_S$	- data storage faults
$F \rightarrow F'$	- operation decoding faults
$\leftarrow$	- data transfer faults
$\rightarrow N$	- control faults
$(F) \rightarrow (F)'$	- data manipulation faults

Disadvantage: Too many fault dedicated models, abstraction, formalization and tool support are missing, test program generation is a manual work today

Microprocessor Fault Model

Faults affecting the operation of microprocessor can be divided into the following classes:

- **addressing faults affecting register decoding**
- **addressing faults affecting the instruction decoding and – sequencing functions;**
- **faults in the data-storage function;**
- **faults in the data-transfer function;**
- **faults in the data-manipulation function**

Disadvantage: Formalization and tool support are missing, test program generation is a manual work today

Summary of Overview

- Main tools and tasks
  - Test generation
  - Fault Simulation
  - Fault Diagnosis
- Different fault types
  - Logic faults, X-faults, Timing (delay) faults, High-Level faults
- Fault modeling
  - Low-level fault models (SAF, transistor faults, bridging, delays)
  - High-level fault models (RTL, microprocessors)

Fault Simulation

Fault table

Test experiment data

Fault modeling

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

E_1	E_2	E_3
0	0	1
0	1	0
0	1	0
1	0	1
1	0	1
0	0	0

How many rows and columns should be in the Fault Table?

Fault F_5 located

Fault simulation

Testing

Fault diagnosis

Test generation

Comparison of Fault Simulation Methods

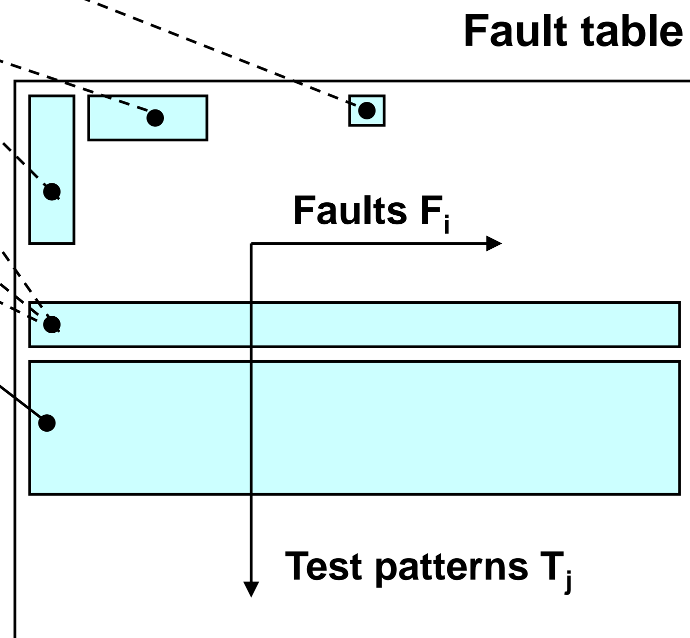
Fault simulation techniques:

- serial fault simulation
- parallel fault simulation
- deductive fault simulation
- concurrent fault simulation
- critical path analysis
- parallel critical path analysis

Common concepts:

- fault specification (fault collaps)
- fault insertion
- fault effect propagation
- fault discarding (dropping)

Comparison of methods:

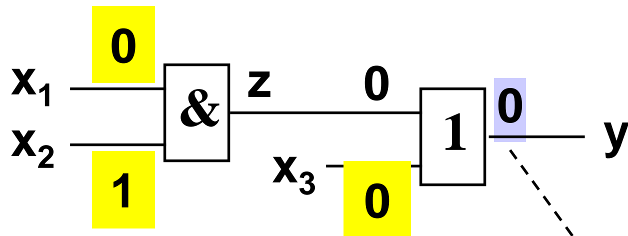


Entry (i,j) = 1(0) if F_i is detectable (not detectable) by T_j

Single and Parallel Fault Simulation

Single pattern

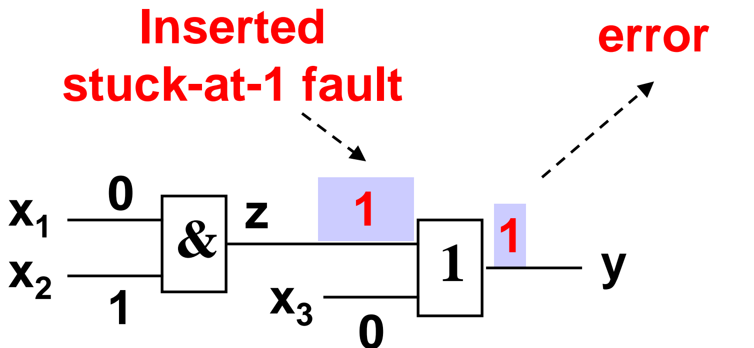
Fault-free circuit:



Faulty circuit:

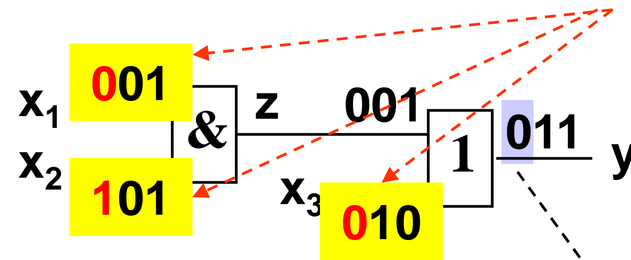
Detected error

Inserted stuck-at-1 fault



Parallel patterns

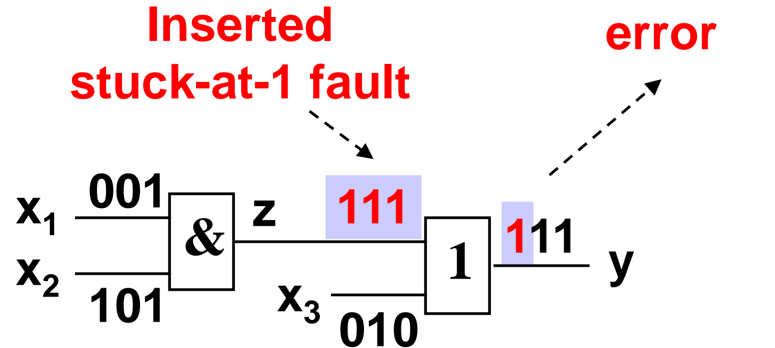
Fault-free circuit: Three test patterns



Faulty circuit:

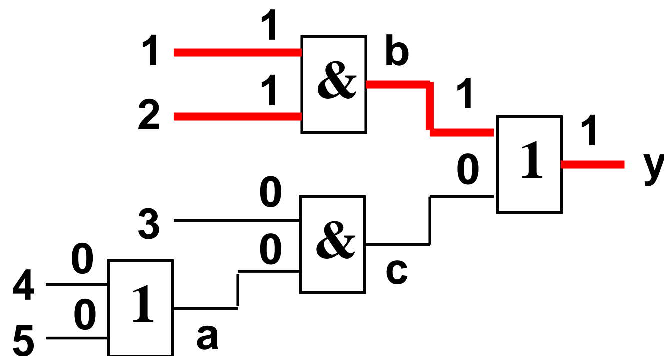
Inserted stuck-at-1 fault

Detected error

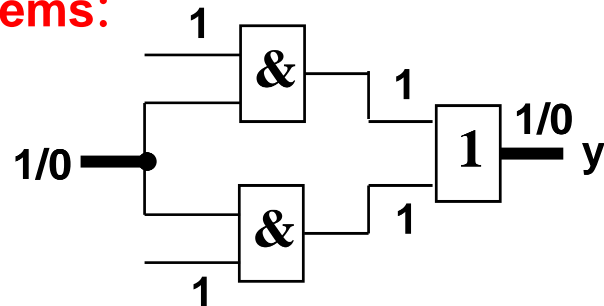


Fault Simulation: Critical Path Tracing

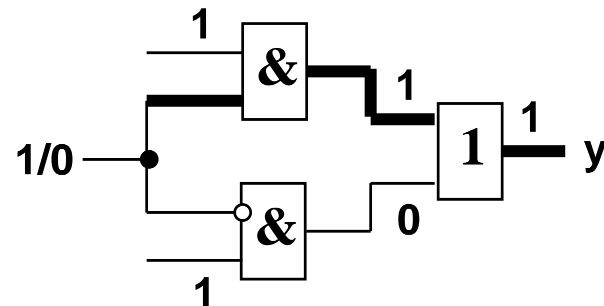
Activated (critical) path
is traced backwards



Problems:

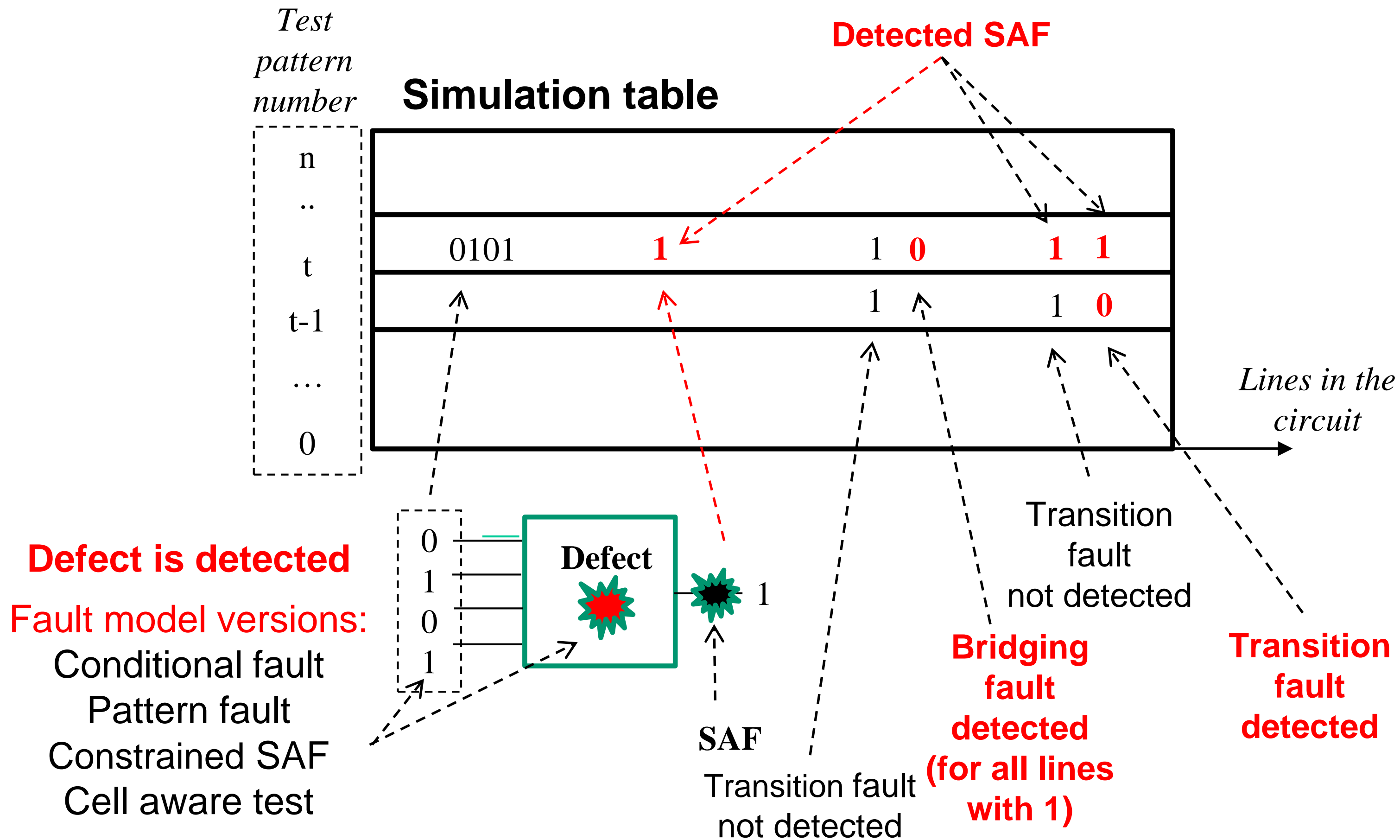


The critical path is not continuous

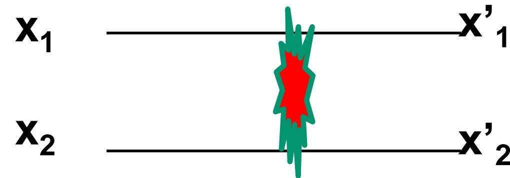


The critical path breaks on the fan-out

Simulation of Different Classes of Faults

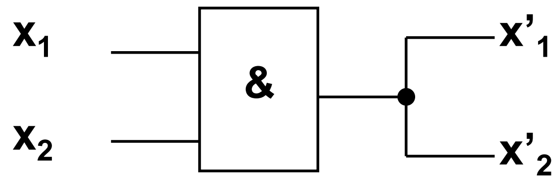


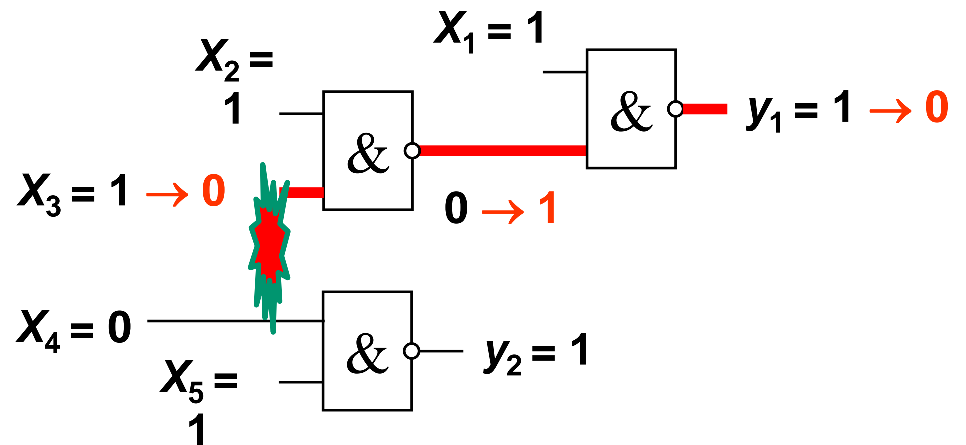
Testing of Bridging Fault Models



Wired AND model

W-AND:





Simulation of Bridging Faults

Nodes and variables:

Bridging fault table

	1	2	3	4	5	6	7
1		0	0			0	
2	01			1	01		1
3	1			1	1		1
4	0		0		0		
5		1		1			1
6		1		1			1
7		0	0			0	

Simulation table for SAF faults (variables)

x1	x2	x3	x4	x5	x6	x7
1	0	1	0	1	1	0
0	1	1	0	0	1	0

SAF fault table (nodes)

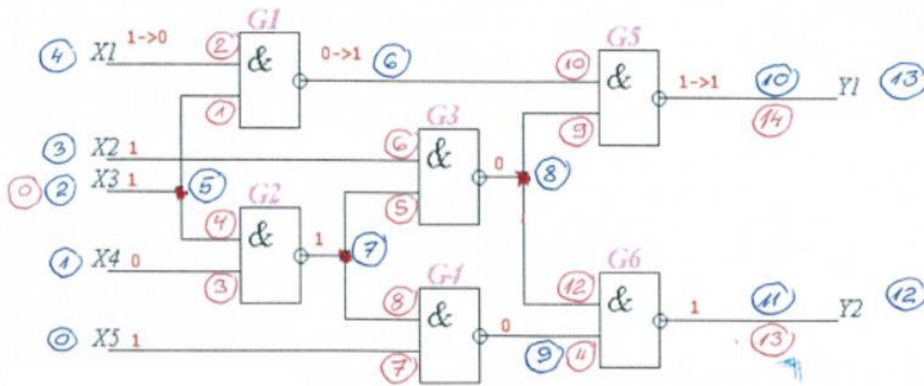
1	2	3	4	5	6	7
x	1	x	1	0	0	x
1	0	0	x	x	x	1

Converting the SAF fault table into Bridging fault table

Turbo-Tester Data Formats

Circuit

Blue colour – variables
Red colour – nodes (fault locations)



Circuit model

```
STAT#   15 Nods,  14 Vars,  9 Grps,  5 Inps,  0 Cons,  2 Outs

MODE#   STRUCTURAL

VAR#   0:  (i_______)  "i_5"
VAR#   1:  (i_______)  "i_4"
VAR#   2:  (i_______)  "i_3"
VAR#   3:  (i_______)  "i_2"
VAR#   4:  (i_______)  "i_1"

VAR#   5:  (________)  "i_3"
GRP#    0: BEG =   0, LEN =   1 -----
   0   0:  (____)    (    0     0)  V = 2     "i_3"

VAR#   6:  (________)  "inst_0>o"
GRP#    1: BEG =   1, LEN =   2 -----
   1   0:  (I___)    (    1     0)  V = 5     "inst_0>i_1"
   2   1:  (I___)    (    0     0)  V = 4     "i_1"

VAR#   7:  (________)  "inst_1>o"
GRP#    2: BEG =   3, LEN =   2 -----
   3   0:  (I___)    (    1     0)  V = 1     "i_4"
   4   1:  (I___)    (    0     0)  V = 5     "inst_1>i_2"

VAR#   8:  (________)  "inst_2>o"
GRP#    3: BEG =   5, LEN =   2 -----
   5   0:  (I___)    (    1     0)  V = 7     "inst_2>i_1"
   6   1:  (I___)    (    0     0)  V = 3     "i_2"

VAR#   9:  (________)  "inst_3>o"
GRP#    4: BEG =   7, LEN =   2 -----
   7   0:  (I___)    (    1     0)  V = 0     "i_5"
   8   1:  (I___)    (    0     0)  V = 7     "inst_3>i_2"

VAR#  10:  (________)  "inst_4>o"
GRP#    5: BEG =   9, LEN =   2 -----
   9   0:  (I___)    (    1     0)  V = 8     "inst_4>i_1"
  10   1:  (I___)    (    0     0)  V = 6     "inst_0>o"

VAR#  11:  (________)  "inst_5>o"
GRP#    6: BEG =  11, LEN =   2 -----
  11   0:  (I___)    (    1     0)  V = 9     "inst_3>o"
  12   1:  (I___)    (    0     0)  V = 8     "inst_5>i_2"

VAR#  12:  (_o______)  "o_2"
GRP#    7: BEG =  13, LEN =   1 -----
  13   0:  (____)    (    0     0)  V = 11     "inst_5>o"

VAR#  13:  (_o______)  "o_1"
GRP#    8: BEG =  14, LEN =   1 -----
  14   0:  (____)    (    0     0)  V = 10     "inst_4>o"
```

Simulation table (variables)

```
01234 5678901 23
----------------
11110 hhlhhll LL
11001 lhhhllh HL
00011 lhhlhhh HH
10101 hlhhlhh HH
01000 lhhhhll LL
```

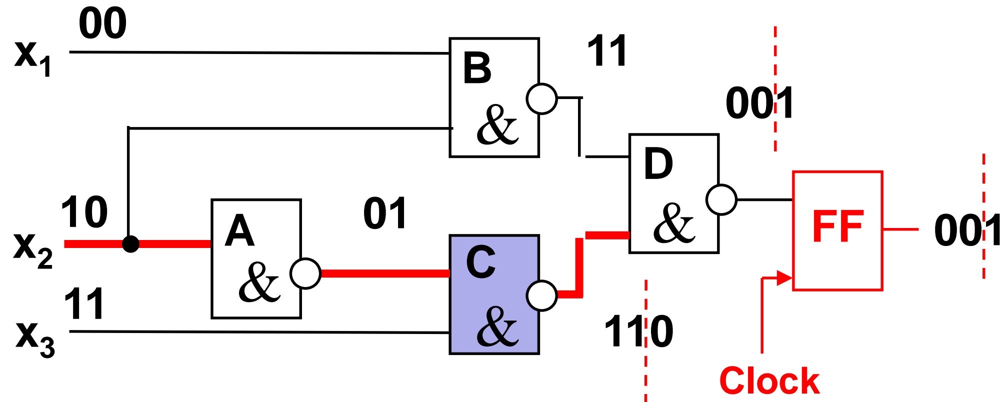
Fault table (nodes)

```
25415  73078  69810 < variables
01234  56789  01234 < nodes
---------------------
0X100  1XX10  00011
11XX1  X1000  01X01
XXXXX  00XX1  XX100
0001X  XX00X  11X00
XXXXX  X11X0  00011
```

Delay Fault Models

Delay faults are tested by test pattern pairs:

- the first test pattern **initializes** the circuit, and
- the second pattern **sensitizes** the fault



Delay fault models:

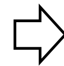
- Gate delay fault (delay fault is lumped at a single gate, quantitative model)
- Transition fault **(qualitative model, gross delay fault model, independent of the activated path)**
- Path delay fault **(sum of the delays of gates along a given path)**
- Line delay fault (is propagated through the longest senzitizable path)
- Segment delay fault (tradeoff between the transition and the path delay fault models)

Simulation of Delay Faults

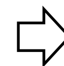
Optimization of the test sequence by reordering the patterns

Example: Converting of SAF fault table into the delay fault table

	1	2	3	4
T1	0	1	1	0
T2	1	1	1	0
T3	1	0	1	1
T4	0	1	0	0
7	1	2	2	2



	1	2	3	4
T1	0	1	1	0
T3	1	0	1	1
T2	1	1	1	0
T4	0	1	0	0
6	1	2	1	2



	1	2	3	4
T1	0	1	1	0
T3	1	0	1	1
T4	0	1	0	0
T2	1	1	1	0
8	2	2	2	2

Numbers of detected delay faults in columns

Number of all detected delay faults

Full fault coverage is achieved

Procedure (greedy algorithm):

1) Find the test pair with maximum of the Hamming distance and set this pair as the first two patterns
2) Find for the second pattern the companion from the rest of patterns with maximum of the Hamming distance
3) Continue the algorithm till the next to the last pattern

Fault Table based Diagnosis

Combinational fault diagnosis

Two phases:
1) The full test (all test patterns) is executed, and the **result vector E_k** is fixed
2) A match for E_k in the fault table with a column vector F_j is found
3) The colmn vector F_j refers to the **fault F_j**

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

E_1	E_2	E_3
0	0	1
0	1	0
0	1	0
1	0	1
1	0	1
0	0	0

Fault F_5 located

Faults F_1 and F_4 are not distinguishable

No match, diagnosis not possible

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Fault Diagnosis

Sequential (adaptive) fault diagnosis by **Edge-Pin Testing**

Test patterns are executed in a sequence one-by-one. Depending on the test result, the next pattern will be selected and executed. Beforehand a diagnostic tree can be constructed

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

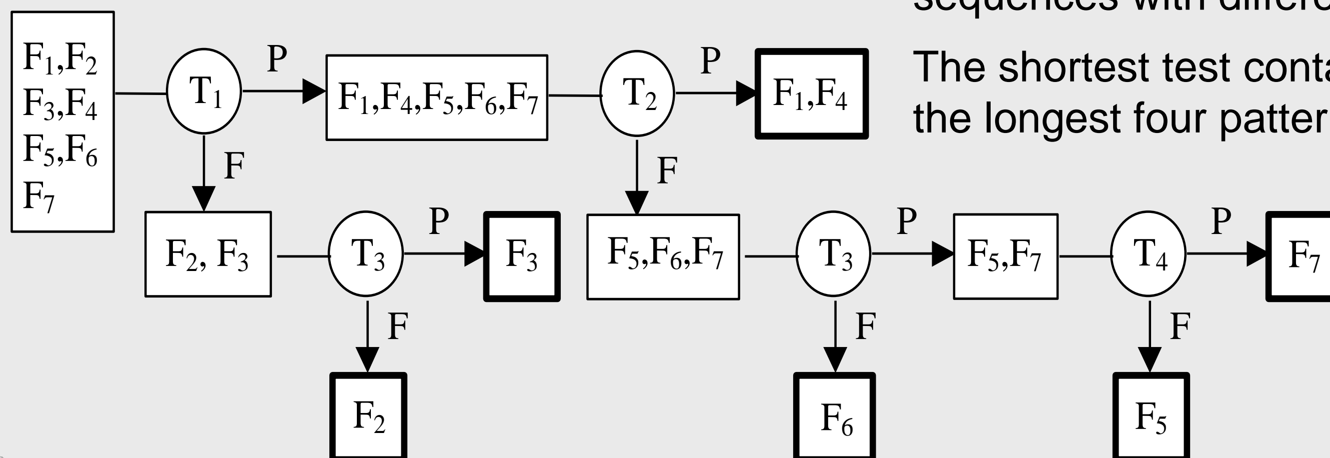
Diagnostic tree:

Two faults F_1,F_4 remain indistinguishable

Not all test patterns used in the fault table are needed

Different faults need for identifying test sequences with different lengths

The shortest test contains two patterns, the longest four patterns

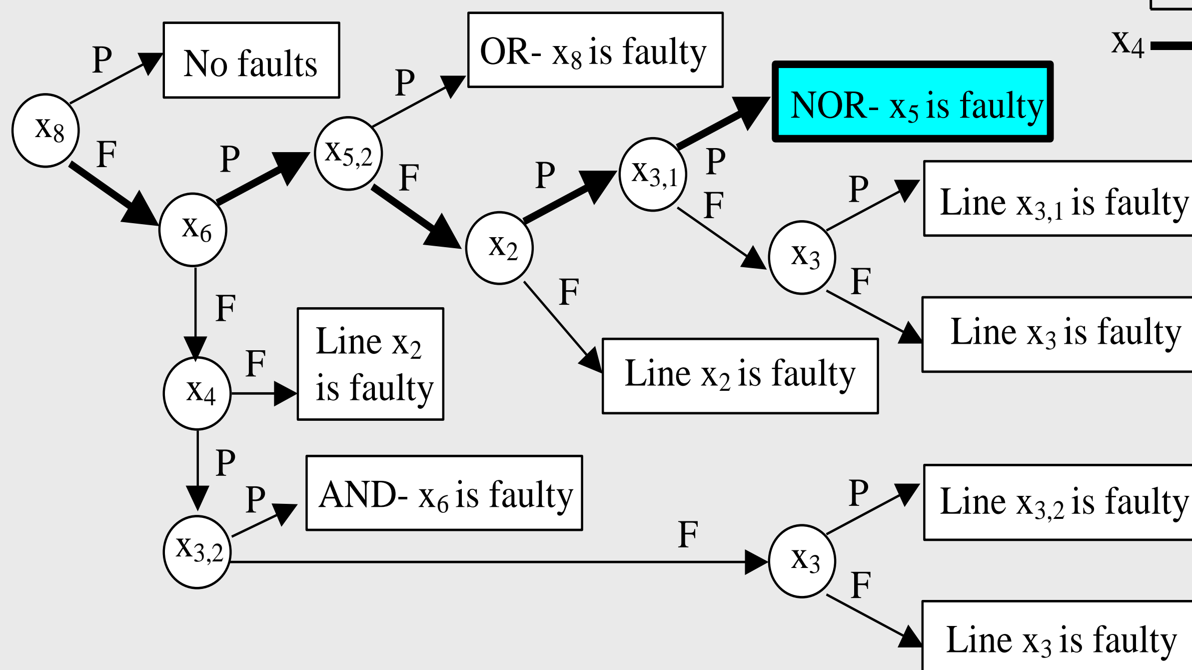


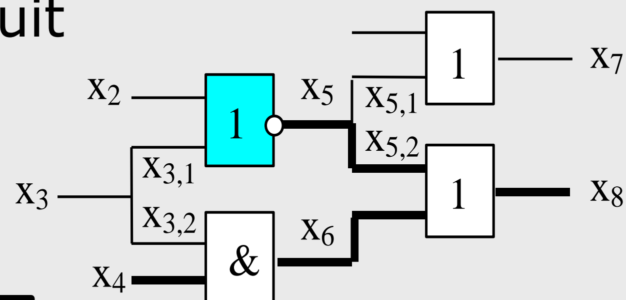
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Sequential Fault Diagnosis

Guided-probe testing inside the circuit

Searh tree:



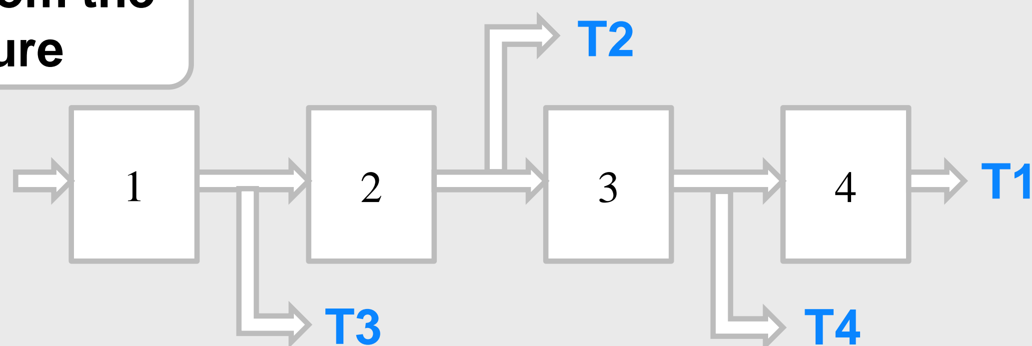


Faulty circuit

Differently from the previous methods, in this approach multiple faults can be independently located

The Problem of Diagnosis: A Good Strategy

Let us begin from the whole picture



Fault candidates

1,2,3,4
T1

+ OK

– 1,2
T2

+ 3,4
T4

+ 4
– 3

– 1
T3

+ 2
– 1

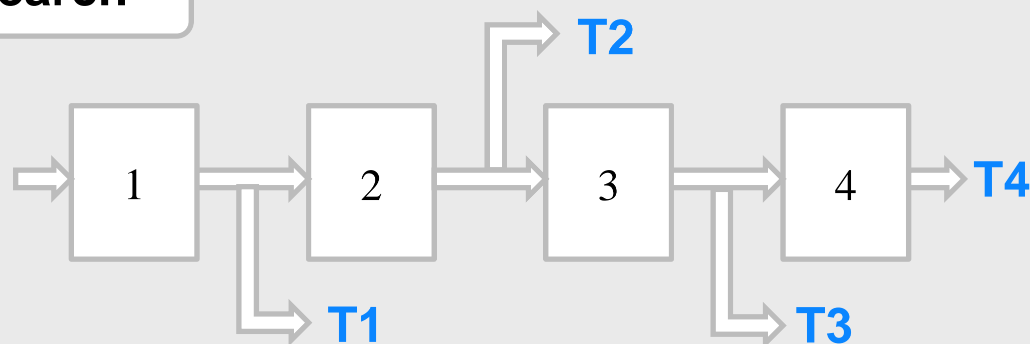
Thereafter: divide and conquer

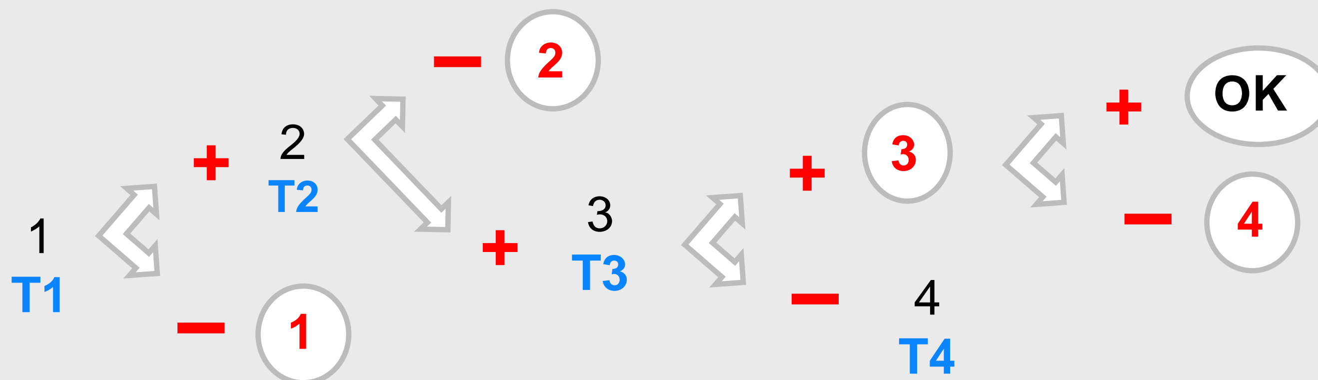
The average length of the fault location procedure:

1 + 4 * 3 = 13/5 = **2,6**

An Example of a Bad Strategy

One-by-one search





The previous result was :

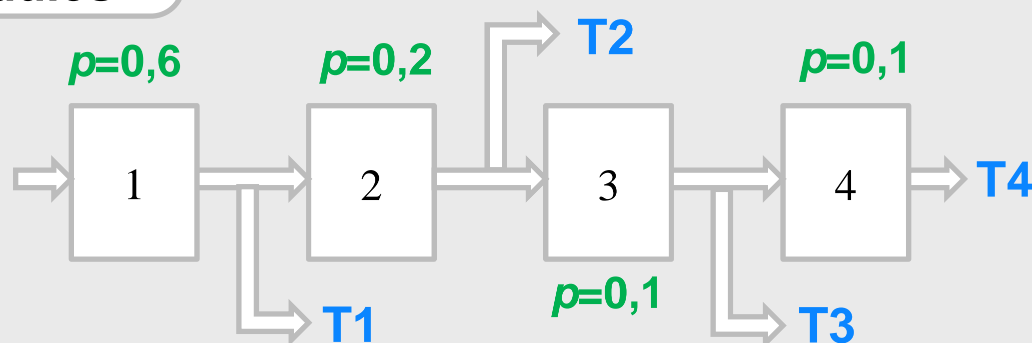
2,6

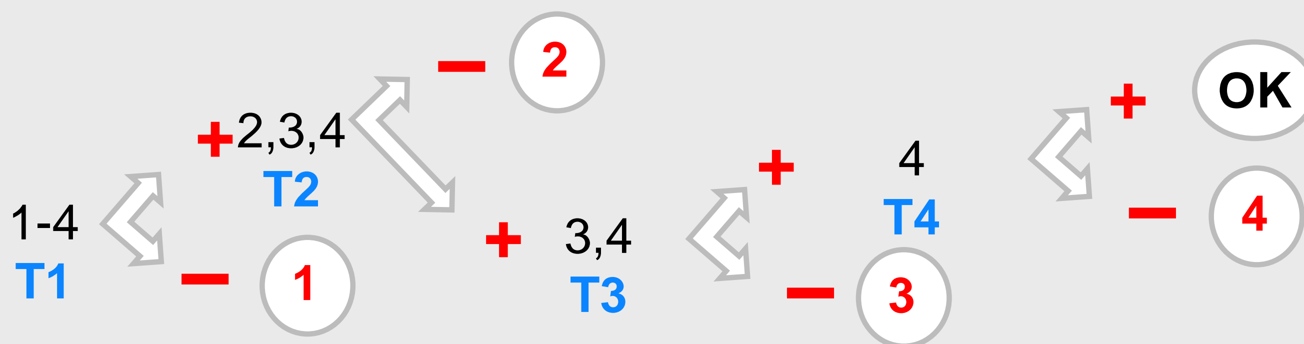
The average length of the fault location procedure:

$$(1 + 2 + 3 + 4 * 2)/5 = 14/5 = 2{,}8$$

Optimization Using Statistics

Probability of faults may be different in different modules





The previous results were :

2,6 & 2,8

The average length of the fault location procedure:

$$(1 * 0,6 + 2 * 0,2 + 3 * 0,1 + 4 * 0,1 * 2) = \mathbf{2,1}$$

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation

Fault table

Test experiment data

Fault modeling

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
T_1	0	1	1	0	0	0	0
T_2	1	0	0	1	0	0	0
T_3	1	1	0	1	0	1	0
T_4	0	1	0	0	1	0	0
T_5	0	0	1	0	1	1	0
T_6	0	0	1	0	0	1	1

E_1	E_2	E_3
0	0	1
0	1	0
0	1	0
1	0	1
1	0	1
0	0	0

How many rows and columns should be in the Fault Table?

Fault F_5 located

Fault simulation

Fault diagnosis

Testing

Test generation

Test Generation Methods

Gate-level methods

- **Functional testing:** universal test sets
- **Structural test** generation
  - Path activation conception
  - Algorithms: D, Podem, Fan
  - Test generation for multiple faults
  - Test generation for sequential circuits
- **Random test** generation
- Genetic algorithms for test generation

High-level and hierarchical methods

- Test generation for **digital systems**
- Test generation for **microprocessors**

Exhaustive Testing

Universal test sets

1. Exhaustive test (trivial test)

2. Pseudo-exhaustive test

Properties of (pseudo)exhaustive tests

1. Advantages (concerning the stuck at fault model):

- test pattern generation is not needed

- fault simulation is not needed

- no need for a fault model

- easily generated on-line by hardware

- redundancy problem is eliminated

- single and multiple stuck-at fault coverage is 100%

2. Shortcomings:

- long test length (2^n patterns are needed, n - is the number of inputs)

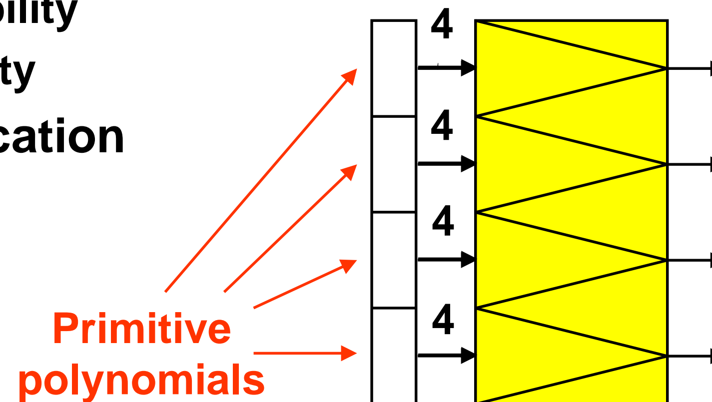
- CMOS stuck-open fault problem

Pseudoexhaustive Testing

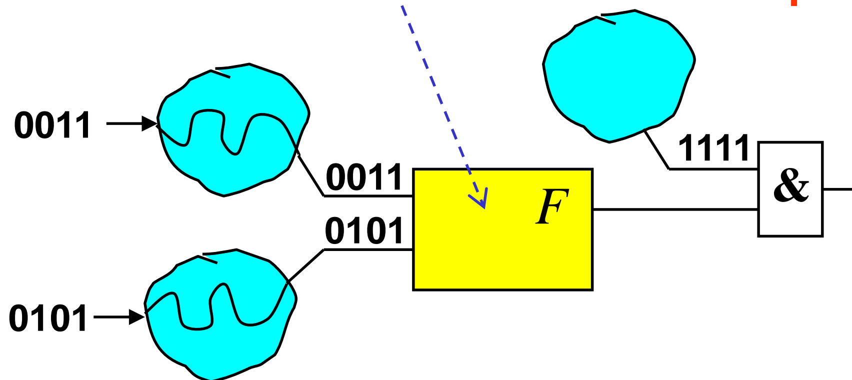
Pseudo-exhaustive test sets:

- Output function verification
  - maximal parallel testability
  - partial parallel testability
- Segment function verification

Output function verification



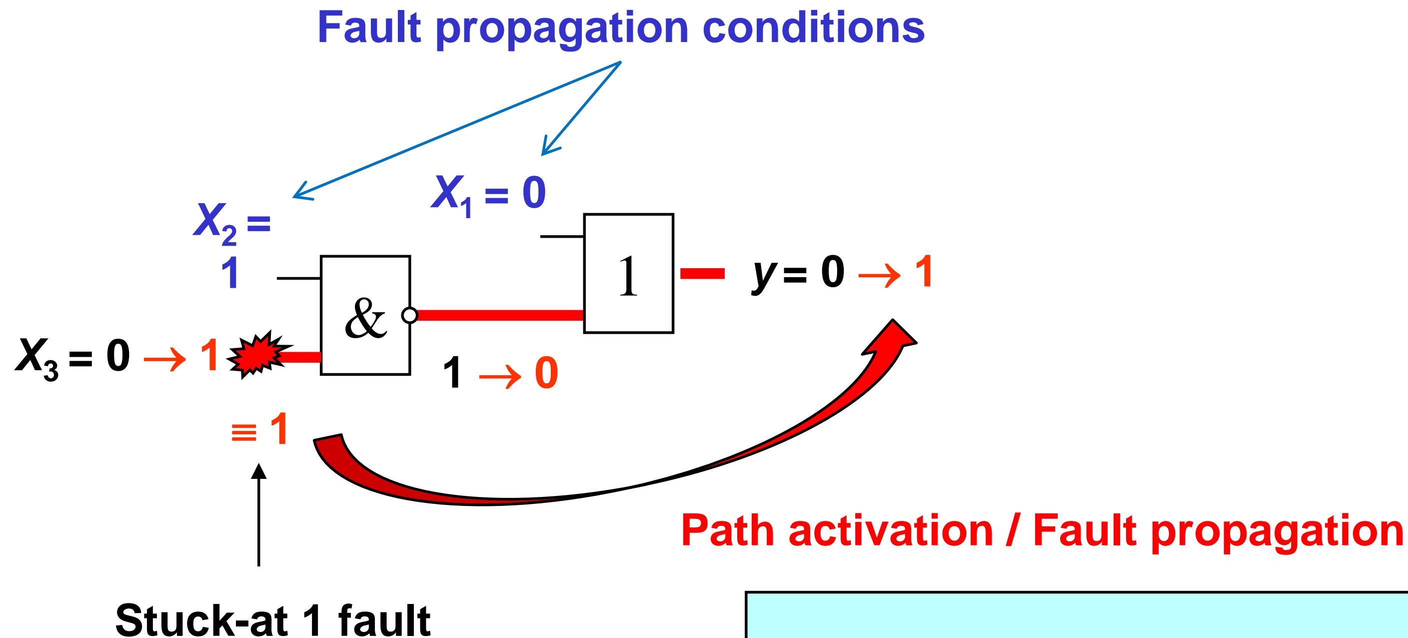
Segment function verification



$2^{16} = 65536$ >> $4 \times 16 = 64$ > 16

Exhaustive test	Pseudo-exhaustive sequential	Pseudo-exhaustive parallel

Faults as Test Generation Objectives



(x_3 = 0 → 1) → (y = 0 → 1)

Output is depending on input change

Test Generation: Stuck-at-Faults (SAF)

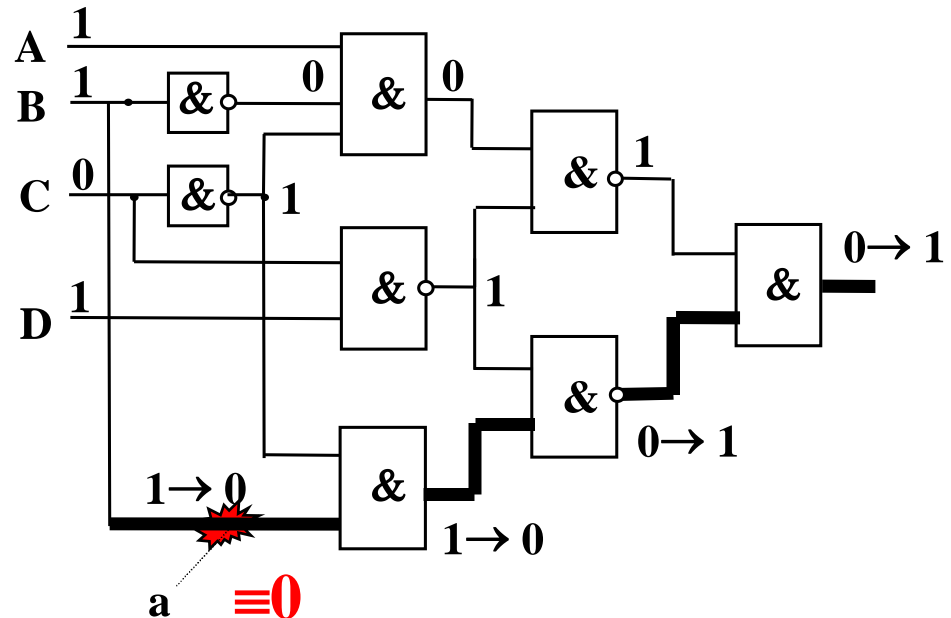
Structural gate-level testing: fault sensitization

Fault detection

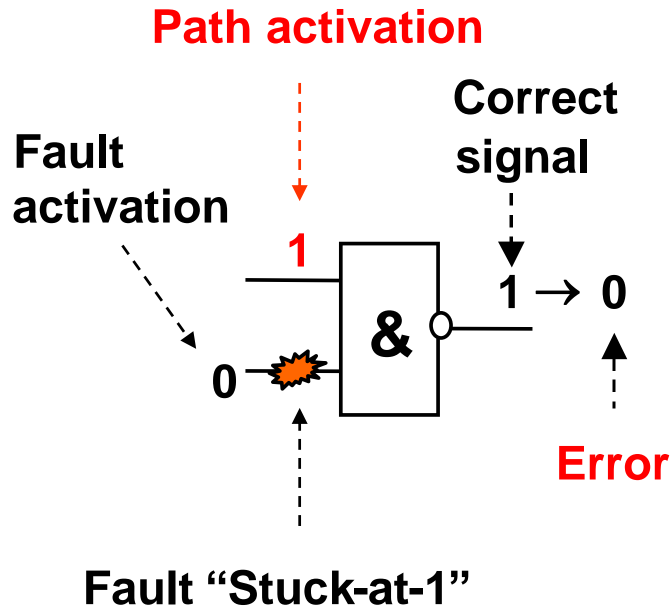
- A test *t* = 1101 is simulated, both without and with the fault a/0
- The fault is detected since the output values in the two cases are different

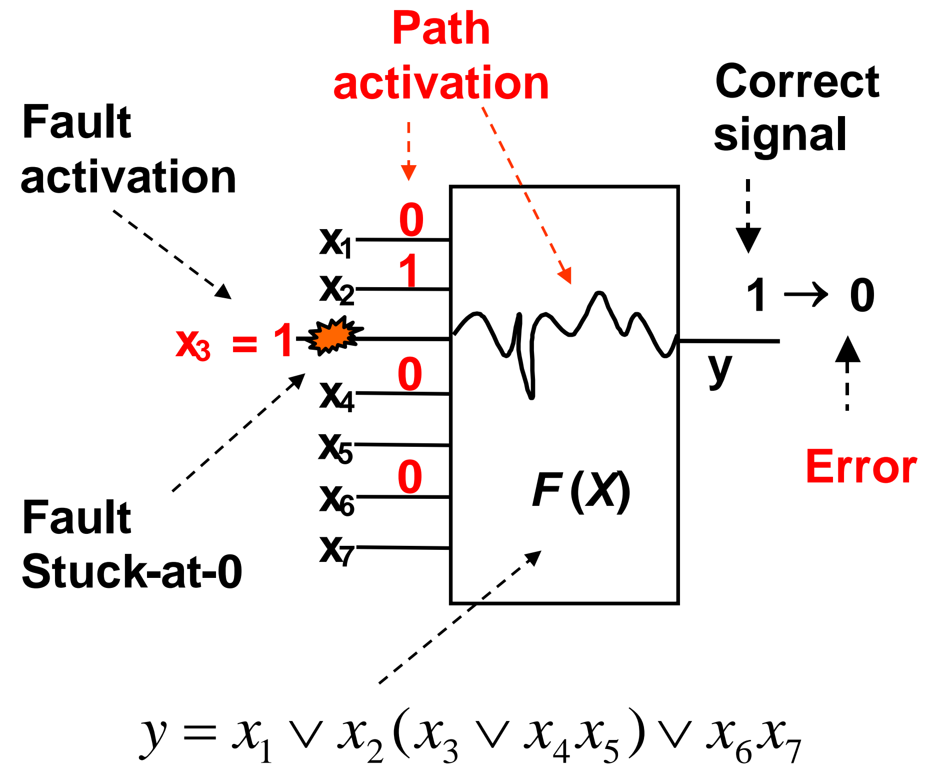
Why is fault detected?

- A fault a/0 is sensitized by the value 1 on a line a
- A path from the faulty line *a* is sensitized (bold lines) to the primary output



Fault Propagation Problem

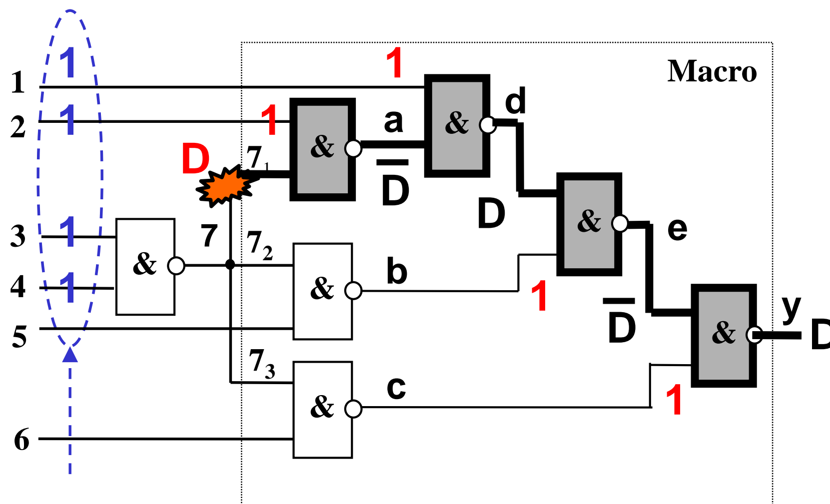




$$y = x_1 \vee x_2(x_3 \vee x_4 x_5) \vee x_6 x_7$$

Structural Test Generation: Main Principles

Single path fault propagation:



Test pattern

$$y = x_6 x_{7,3} \vee (\overline{x_1} \vee x_2 x_{7,1})(\overline{x_5} \vee \overline{x_{7,2}})$$

Fault sensitisation:

$x_{7,1} = D$

Fault propagation:

$x_2 = 1,\ x_1 = 1,\ b = 1,\ c = 1$

Line justification:

$x_7 = D = 0$: $x_3 = 1,\ x_4 = 1$

b = 1: (already justified)

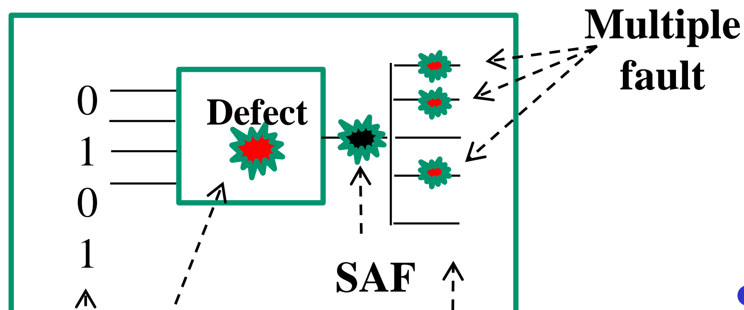
c = 1: (already justified)

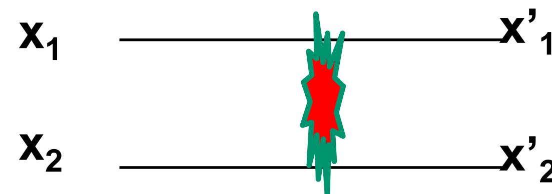
Symbolic fault modeling:

D = 0 - if fault is missing
D = 1 - if fault is present

Test Generation: Conditional SAF

Generalization: Bridging Fault > Conditional SAF

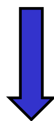




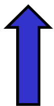
Conditional fault
Pattern fault
Constrained SAF
Single faulty signal

X-fault
Byzantine fault
Bridges
Stuck-opens
Multiple faulty signal

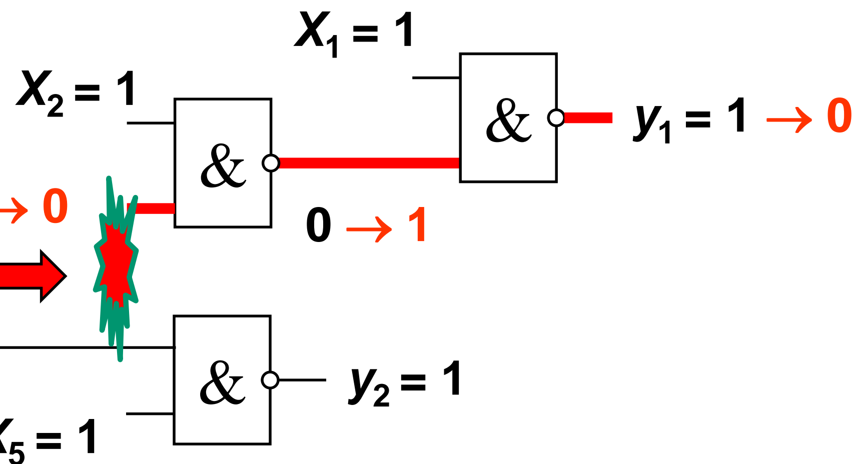
SAF:



Bridge

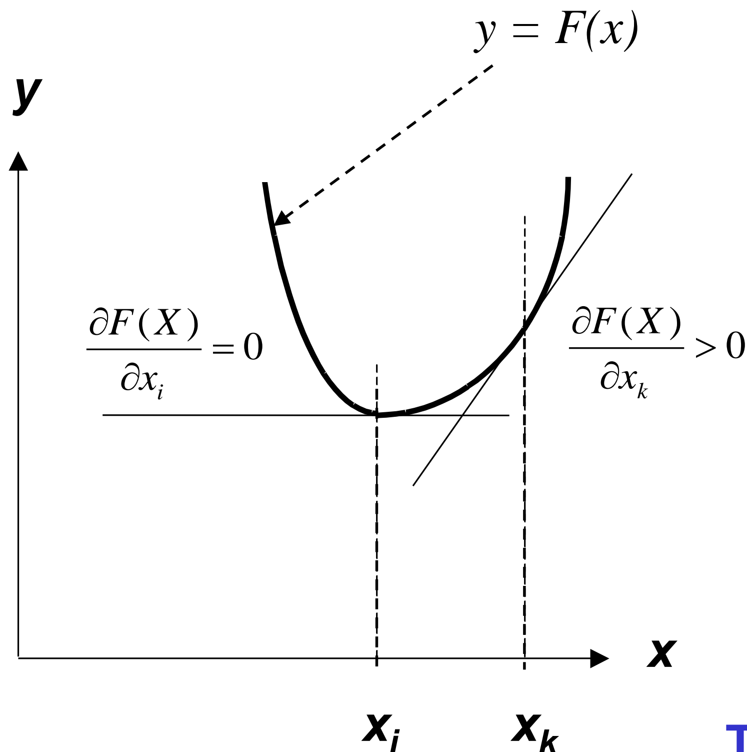


Condition



Boolean Derivatives

Traditional algebra: **speed**



Boolean algebra: **change**

$$x \in \{0,1\},\ Y=F(X)$$
$$Y \in \{0,1\}$$

$$\frac{\partial F(X)}{\partial x_i} = 1$$

F(X) **will change if** x_i changes

$$\frac{\partial F(X)}{\partial x_i} = 0$$

F(X) **will not change if** x_i changes

Reverse function: ***Y=F(X, dX)***

Test generation:
(Calculation of the value of reverse function)

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

Boolean derivatives

Boolean function:

$$Y = F(x) = F(x_1, x_2, \ldots , x_n)$$

Boolean partial derivative:

$$\frac{\partial F(X)}{\partial x_i} = F(x_1,\ldots x_i,\ldots x_n) \oplus F(x_1,\ldots \bar{x}_i,\ldots x_n) = 1$$

Simplified equation:

$$\frac{\partial F(X)}{\partial x_i} = F(x_1,\ldots x_i = 0,\ldots x_n) \oplus F(x_1,\ldots x_i = 1,\ldots x_n) = 1$$

Test generation: $X = g^{-1}(dY = 1, dX)$

Boolean Derivatives

Useful properties of Boolean derivatives:

For *F(x)* not depending on x_i

$$\frac{\partial[F(X)G(X)]}{\partial x_i} = F(X)\frac{\partial G(X)}{\partial x_i}$$

$$\frac{\partial[F(X) \vee G(X)]}{\partial x_i} = \overline{F(X)}\frac{\partial G(X)}{\partial x_i}$$

Example: $x_1 x_4 (x_2 x_3 \vee x_2 \overline{x_3})$

$$F(X) = x_1 x_4 \quad G(X) = x_2 x_3 \vee x_2 \overline{x_3}$$

$$\frac{\partial[F(X)G(X)]}{\partial x_2} = x_1 x_4 \frac{\partial G(X)}{\partial x_i}$$

Continue the same for $\frac{\partial G(X)}{\partial x_i}$

These properties allow to simplify the Boolean differential equation to be solved for generating test pattern for a fault at x_i

Calculation of Boolean Derivatives

Special cases for differential equations:

$$\frac{\partial F(X)}{\partial x_i} \equiv 0$$ - **if** *F(x)* **is independent of** x_i

$$\frac{\partial F(X)}{\partial x_i} \equiv 1$$ - **if** *F(x)* **depends always on** x_i

$$\frac{\partial F(X)}{\partial x_i} = F(x_1,\ldots x_i = 0,\ldots x_n) \oplus F(x_1,\ldots x_i = 1,\ldots x_n)$$

Examples:

$$F(X) = x_1(x_2 x_3 \vee x_2 \overline{x_3}): \quad \frac{\partial F(X)}{\partial x_3} = x_1 x_2 \oplus x_1 x_2 \equiv 0$$

$$F(X) = x_1 \oplus x_2 : \quad \frac{\partial F(X)}{\partial x_1} = \overline{x_2} \oplus x_2 \equiv 1$$

Calculation of Boolean Derivatives

Example:

Given: $y = x_1 x_2 \vee x_3(\overline{x_2} x_4 \vee \overline{x_1}(x_4 \vee (x_5 \vee \overline{x_2} x_6))) \vee x_1 \overline{x_3}$

Calculation of the Boolean derivative:

$$\frac{\partial y}{\partial x_5} = \overline{x_1 x_2 \vee x_1 \overline{x_3}} \, \frac{\partial(x_3(\overline{x_2} x_4 \vee \overline{x_1}(x_4 \vee (x_5 \vee \overline{x_2} x_6))))}{\partial x_5} =$$

$$= \overline{x_1 x_2 \vee x_1 \overline{x_3}} \, x_3 \, \frac{\partial(\overline{x_2} x_4 \vee \overline{x_1}(x_4 \vee (x_5 \vee \overline{x_2} x_6)))}{\partial x_5} =$$

$$= \overline{x_1 x_2 \vee x_1 \overline{x_3}} \, x_3 \, \overline{\overline{x_2} x_4} \, \frac{\partial(\overline{x_1}(x_4 \vee (x_5 \vee \overline{x_2} x_6)))}{\partial x_5} =$$

$$= \overline{x_1 x_2 \vee x_1 \overline{x_3}} \, x_3 \, \overline{\overline{x_2} x_4} \, \overline{x_1} \, \frac{\partial(x_4 \vee (x_5 \vee \overline{x_2} x_6))}{\partial x_5} = \overline{x_1 x_2 \vee x_1 \overline{x_3}} \, x_3 \, \overline{\overline{x_2} x_4} \, \overline{x_1} \, \overline{x_4} \, \overline{\overline{x_2} x_6} \, \frac{\partial x_5}{\partial x_5}$$

Calculation of Boolean derivatives

Finding a solution of the differential equation:

$$\frac{\partial y}{\partial x_5} = \overline{(x_1 x_2 \vee x_1 \overline{x_3})}\, x_3 \overline{(\overline{x_2} x_4)}\, \overline{x_1}\, \overline{x_4} \overline{(\overline{x_2} x_6)} \frac{\partial x_5}{\partial x_5} =$$

$$= (\overline{x_1} \vee \overline{x_2})(\overline{x_1} \vee x_3) x_3 (x_2 \vee \overline{x_4}) \overline{x_1}\, \overline{x_4} (x_2 \vee \overline{x_6}) =$$

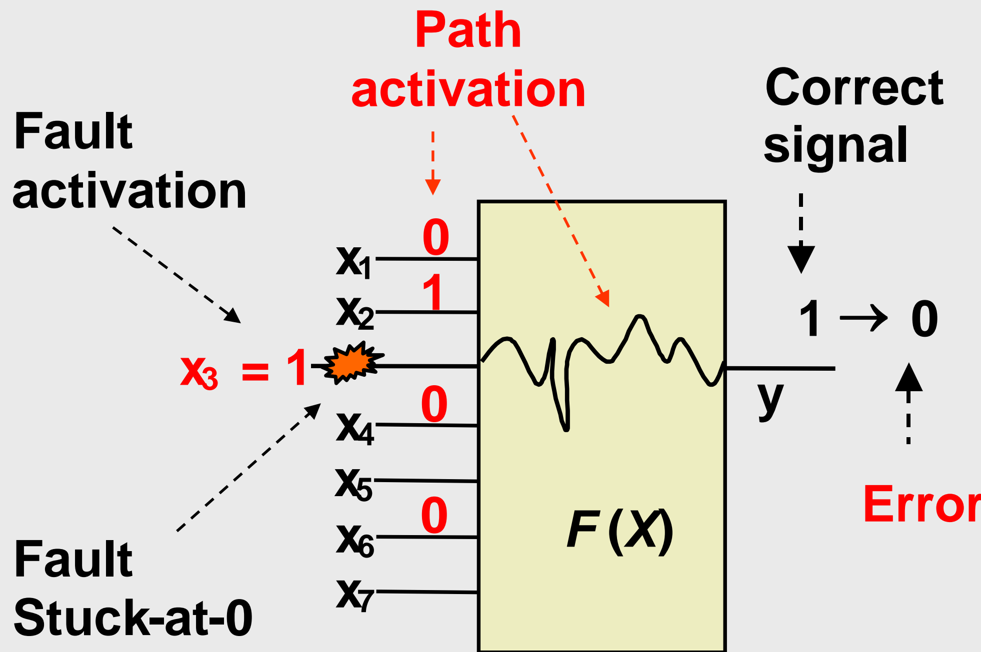
$$= \overline{x_1}\, \overline{x_4} x_3 x_2 \vee \ldots = 1$$

The DNF will include all solutions

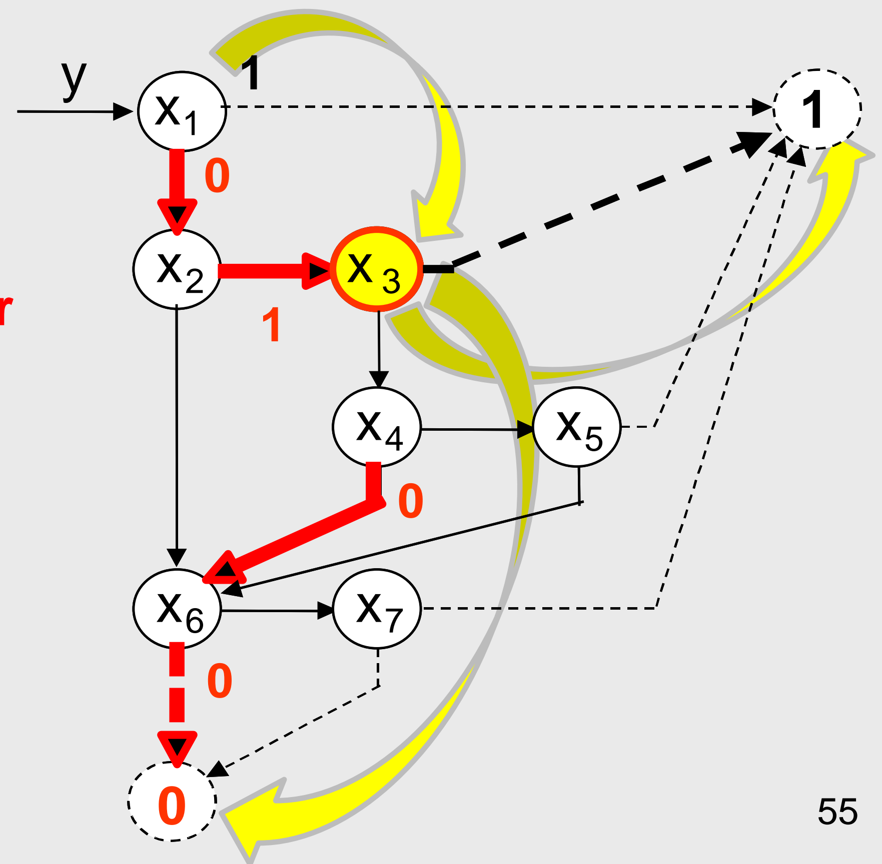
$$\frac{\partial y}{\partial x_5} = \overline{x_1}\, \overline{x_4}\, x_3 (x_2 \vee \overline{x_6}) = \overline{x_1}\, \overline{x_4}\, x_3 x_2 = 1 \Longrightarrow \begin{cases} x_1 = 0 \\ x_4 = 0 \\ x_3 = 1 \\ x_2 = 1 \end{cases}$$

Test pattern is found:

BDDs and Testing of Logic Circuits



Topological view on Binary Decision Diagrams:



$$y = x_1 \vee x_2 (x_3 \vee x_4 x_5) \vee x_6 x_7$$

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation with BD and BDD

BD:

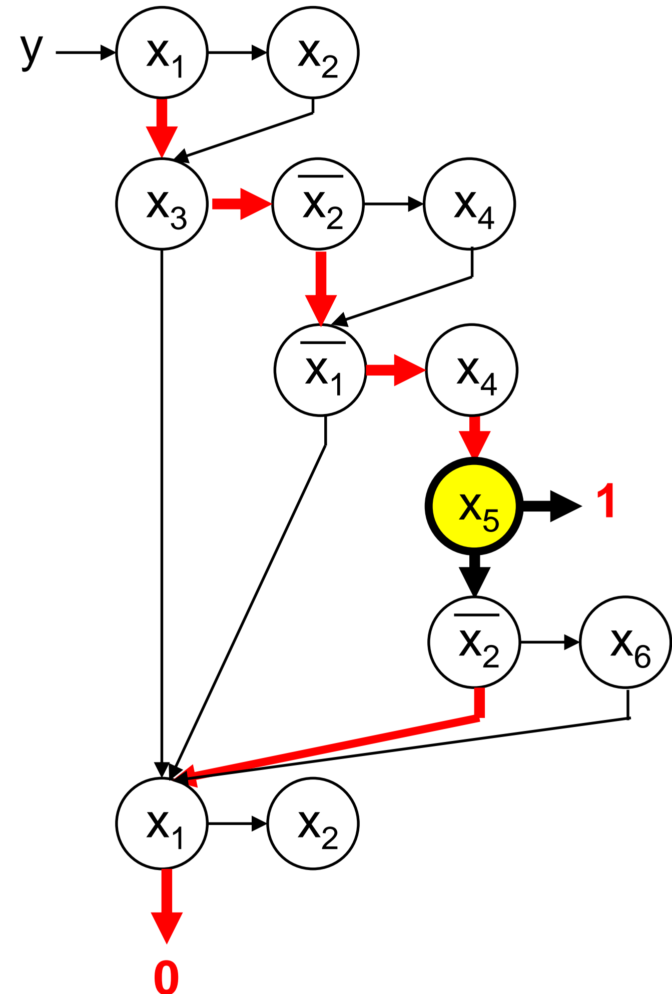
$$y = x_1 x_2 \vee x_3(\overline{x_2} x_4 \vee \overline{x_1}(x_4 \vee (x_5 \vee \overline{x_2} x_6)) \vee x_1 \overline{x_3}$$

$$\frac{\partial y}{\partial x_5} = \overline{(x_1 x_2 \vee x_1 \overline{\overline{x_3}})}\, x_3 \overline{(\overline{\overline{x_2}} x_4)}\, \overline{x_1}\, \overline{x_4}\, \overline{(\overline{\overline{x_2}} x_6)} \frac{\partial x_5}{\partial x_5} =$$

$$= (\overline{x_1} \vee \overline{x_2})(\overline{x_1} \vee x_3) x_3 (x_2 \vee \overline{x_4}) \overline{x_1}\, \overline{x_4} (x_2 \vee \overline{x_6}) =$$

$$= \overline{x_1}\, \overline{x_4} x_3 x_2 \vee \ldots = 1$$

BDD:



Test pattern:

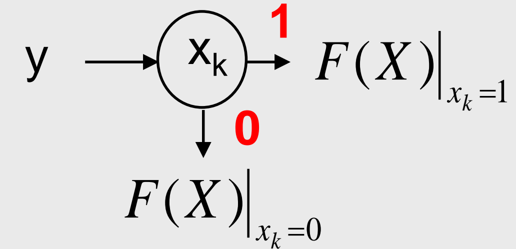
x_1	x_2	x_3	x_4	x_5	x_6	y
0	1	-	0	D	-	D

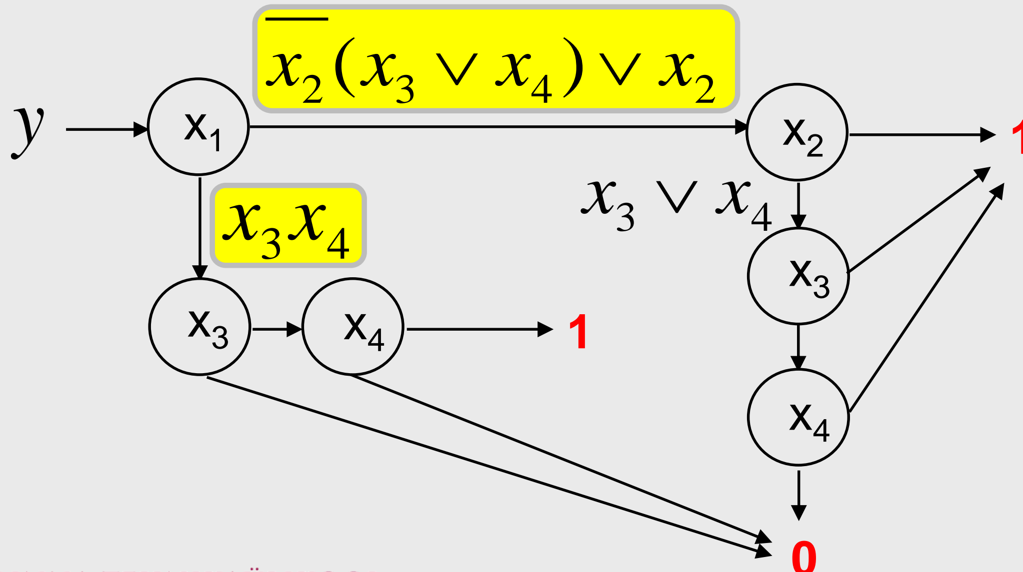
Functional Synthesis of BDDs

Shannon's Expansion Theorem: $y = F(X) = x_k F(X)\big|_{x_k=1} \vee \overline{x_k} F(X)\big|_{x_k=0}$

$$y = \underbrace{x_1 \left(\overline{x_2}(x_3 \vee x_4) \vee x_2\right)}_{x_1 F(X)|_{x_1=1}} \vee \underbrace{\overline{x_1}\, x_3 x_4}_{\overline{x_1} F(X)|_{x_1=0}}$$

Using the Theorem for BDD synthesis:



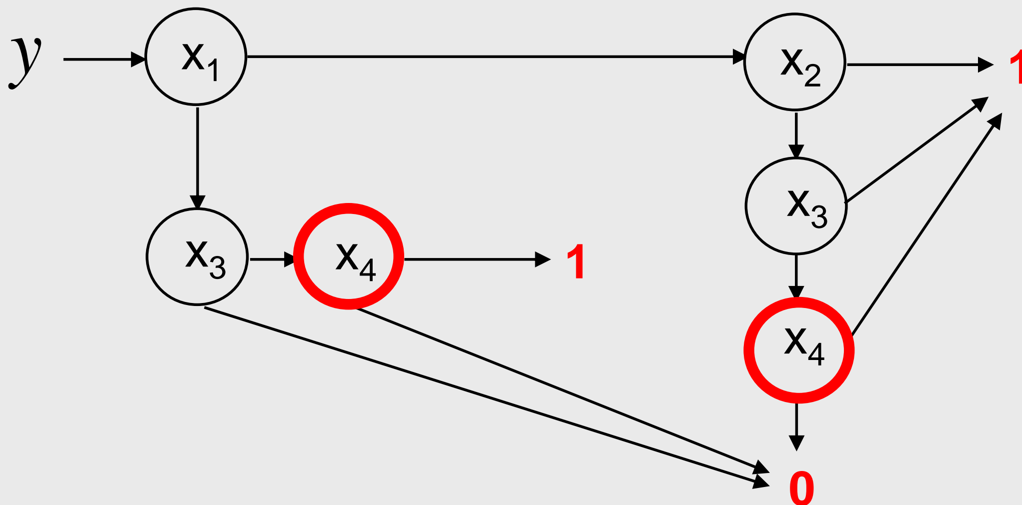


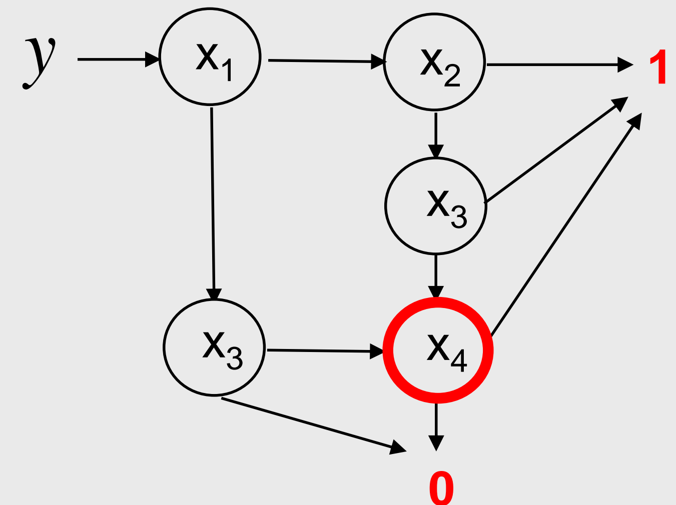
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Functional Synthesis of BDDs

Shannon's Expansion Theorem: $y = F(X) = x_k F(X)\big|_{x_k=1} \vee \overline{x_k} F(X)\big|_{x_k=0}$

$$y = x_1(\overline{x_2}(x_3 \vee x_4) \vee x_2) \vee \overline{x_1} x_3 x_4$$





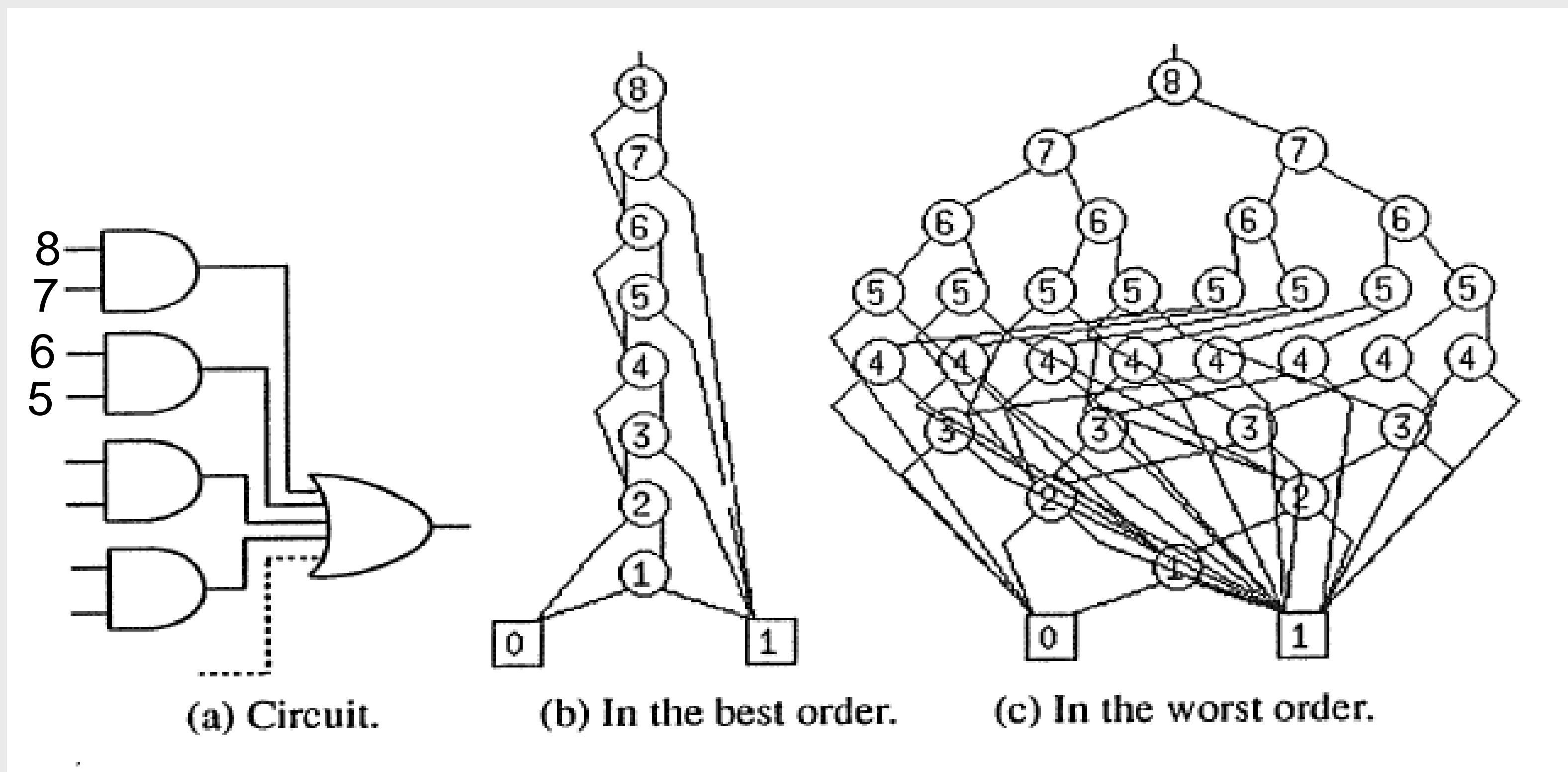
Optimization possibilities:

- ✓ Several terminal nodes with the same variables can be merged
- ✓ Equivalent terminal subgraphs can be also merged

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

BDDs and Complexity

Optimization (by ordering of nodes): BDDs for a 2-level AND-OR circuit



(a) Circuit. (b) In the best order. (c) In the worst order.

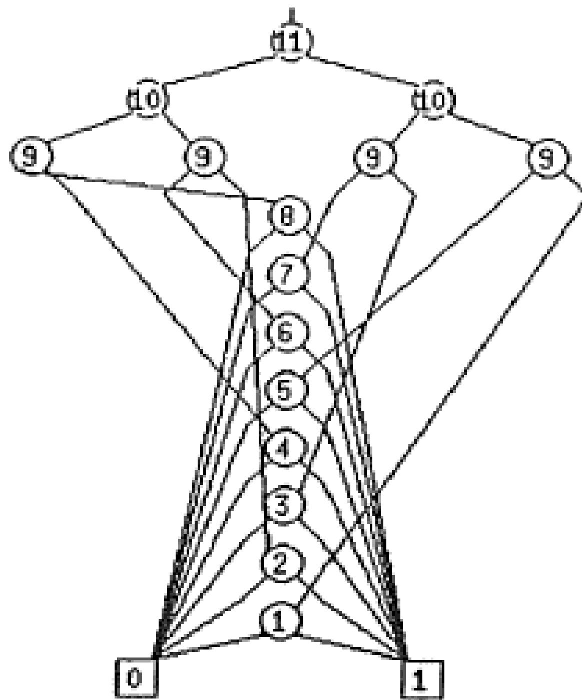
(2n) nodes

$(2*2^n - 2)$ nodes

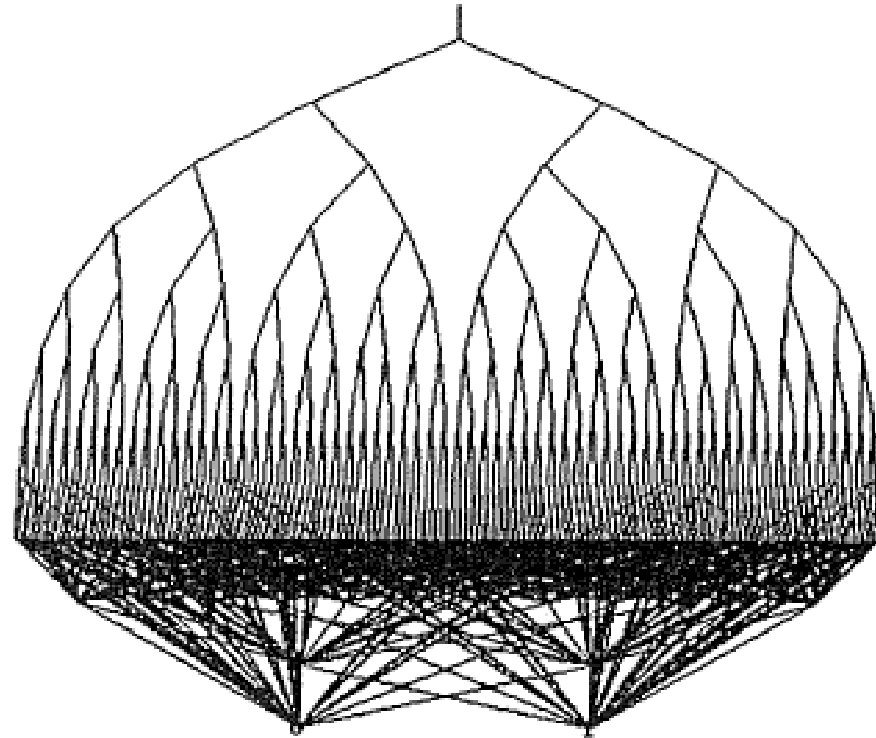
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

©S.Minato, 1996

BDDs and Complexity



(a) In the best order.



(b) In the worst order.

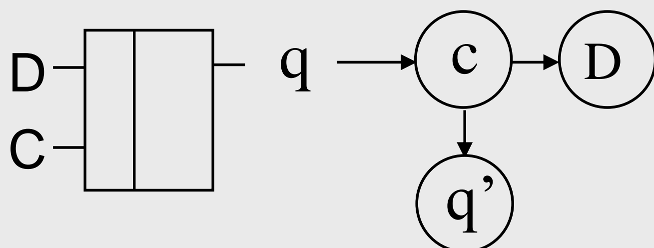
BDDs for an 8-bit data selector

©S.Minato, 1996

BDDs and Complexity

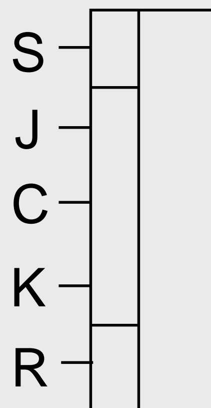
Elementary BDDs

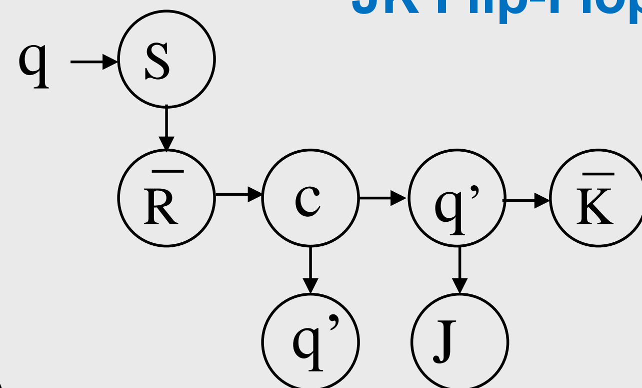
D Flip-Flop



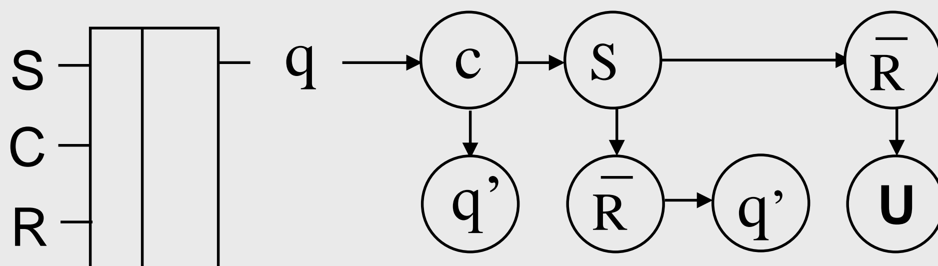
$$q = cD \vee \overline{c}q'$$

JK Flip-Flop





RS Flip-Flop



$$q = c(S \vee q'\overline{R}) \vee \overline{c}q'$$

$$SR = 0$$

U - unknown value

BDD optimization:

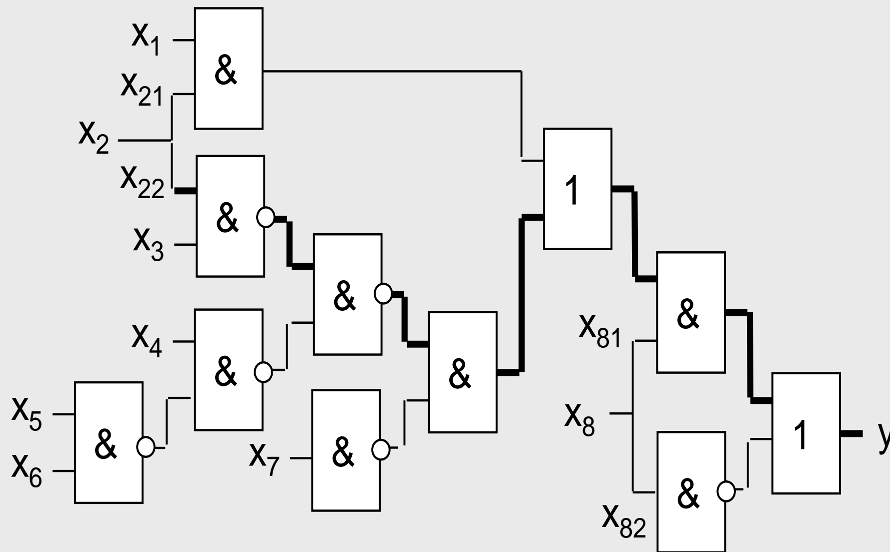
We may start synthesis:

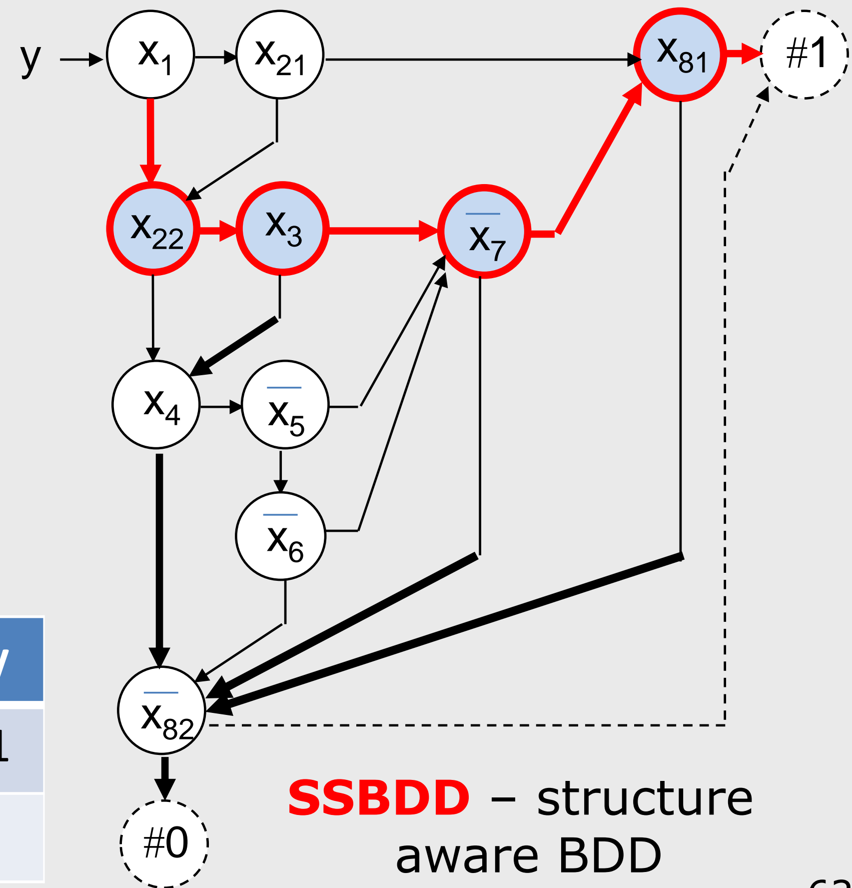
- from the **most important** variable, or
- from the **most repeated** variable

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation with SSBDDs

$$y = f(X) = (x_1 x_{21} \vee (x_{22} x_3 \vee x_4(\overline{x_5} \vee \overline{x_6}))\overline{x_7}) x_{81} \vee \overline{x_{82}}$$





X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
X^t	0	1	1	0	-	-	0	1	1

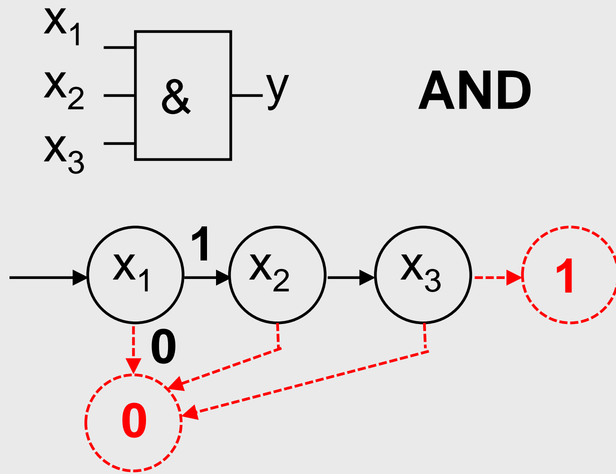
Tested: $x_{22} \equiv 0$, $x_3 \equiv 0$, $x_7 \equiv 1$, $x_{81} \equiv 0$

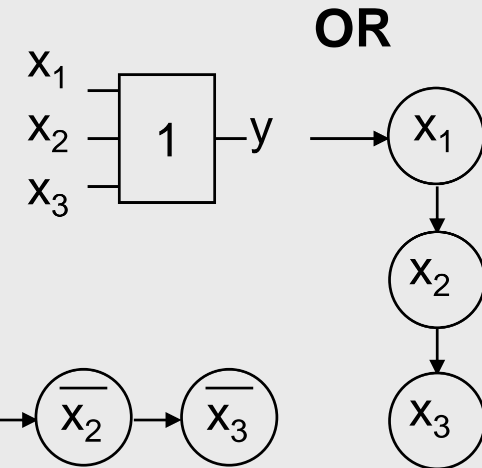
SSBDD – structure aware BDD

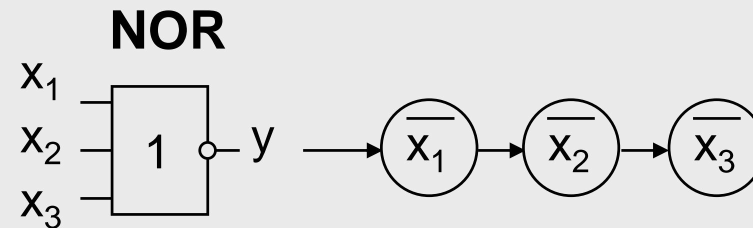
1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

BDDs for Logic Gates

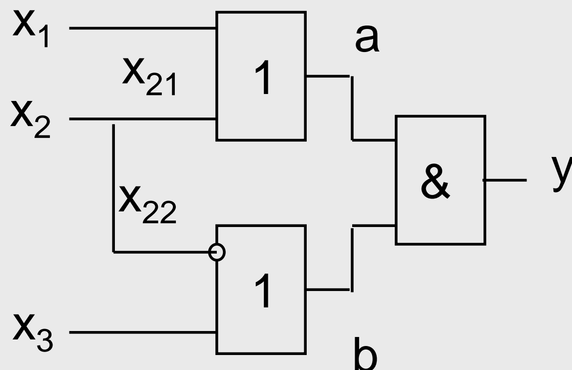
Elementary BDDs:







How about a circuit?

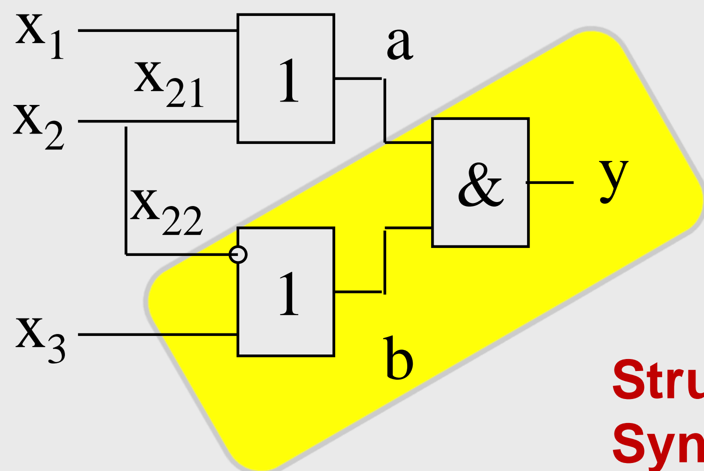


SSBDD synthesis:

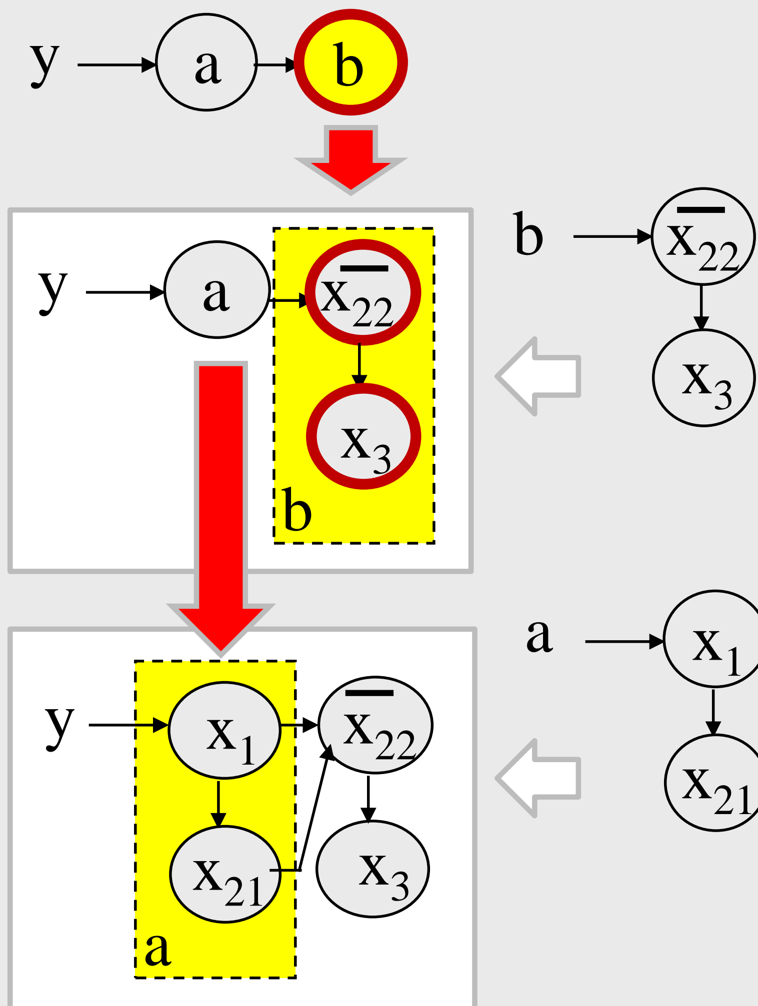
SSBDDs for a given circuit are built by **superposition** of BDDs for gates

Synthesis of BDD for a Circuit

Given circuit:



Superposition of BDDs:



Structurally Synthesized BDD:

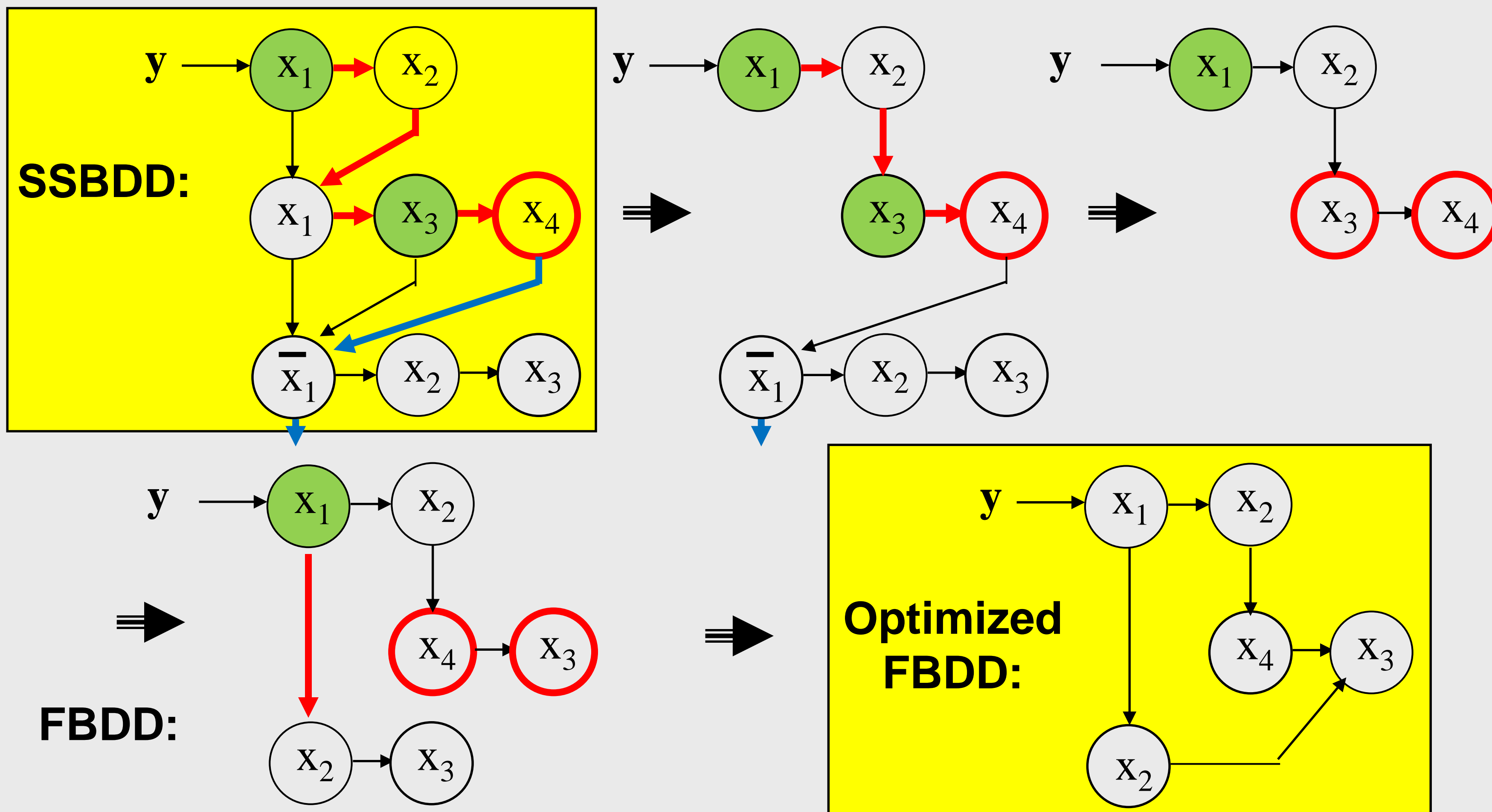
Compare to

Superposition of Boolean functions:

$$y = a \& b = (x_1 \vee x_{21})(\overline{x_{22}} \vee x_3)$$

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

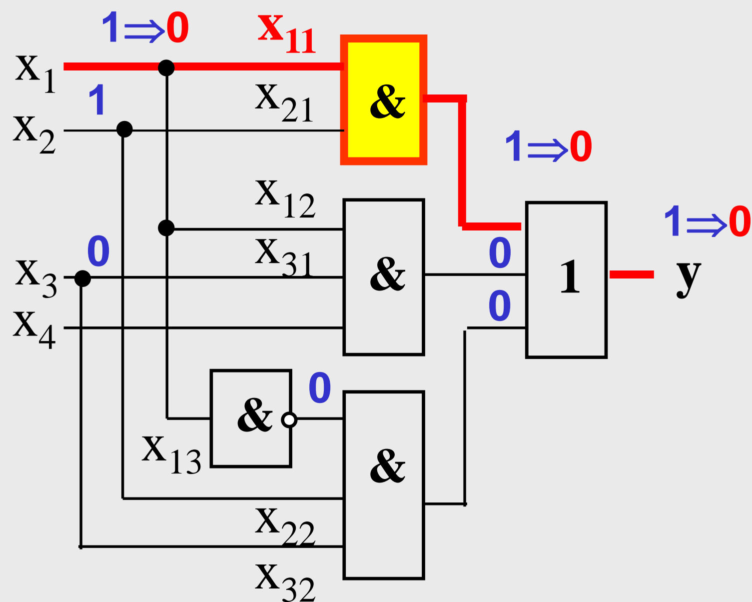
Transformation of SSBDDs to FBDDs



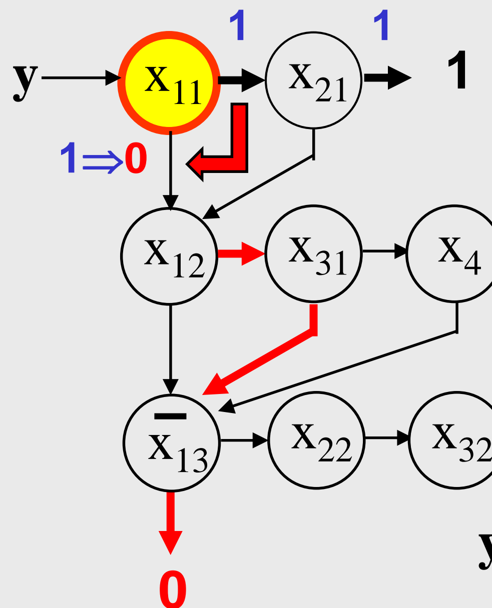
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation with SSBDDs and FBDDs

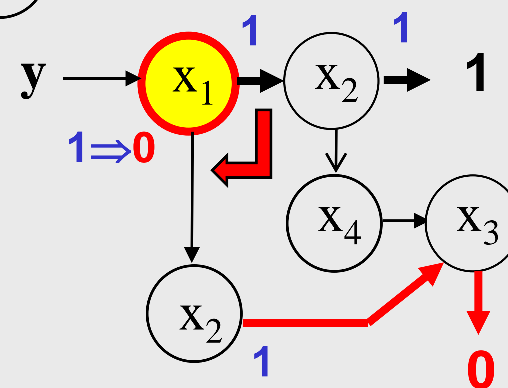
Test generation for: $x_{11} \equiv 0$



Structural BDD:



Functional BDD:

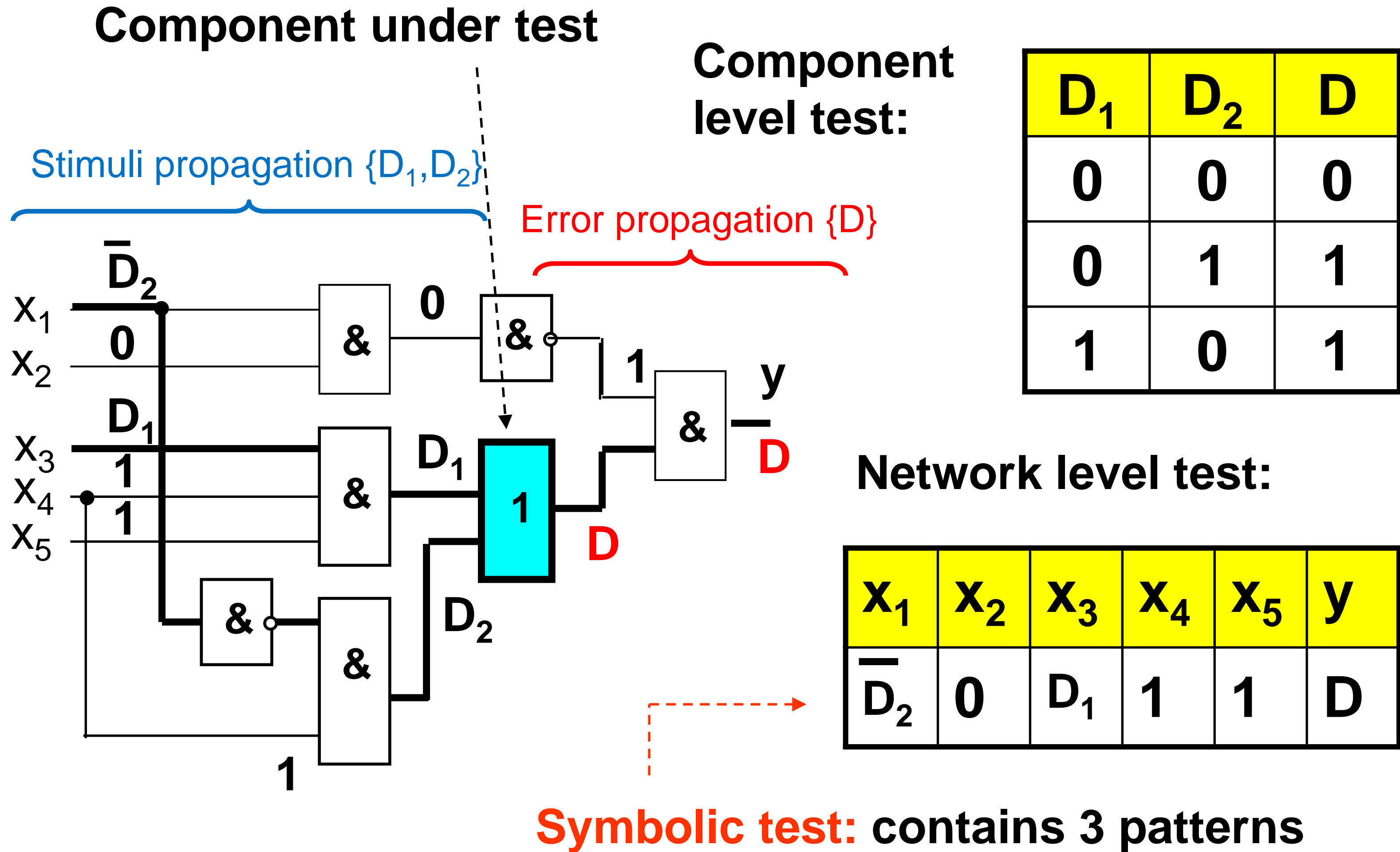


Test pattern:

x_1	x_2	x_3	x_4	y
1	1	0	-	1 ⇒0

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

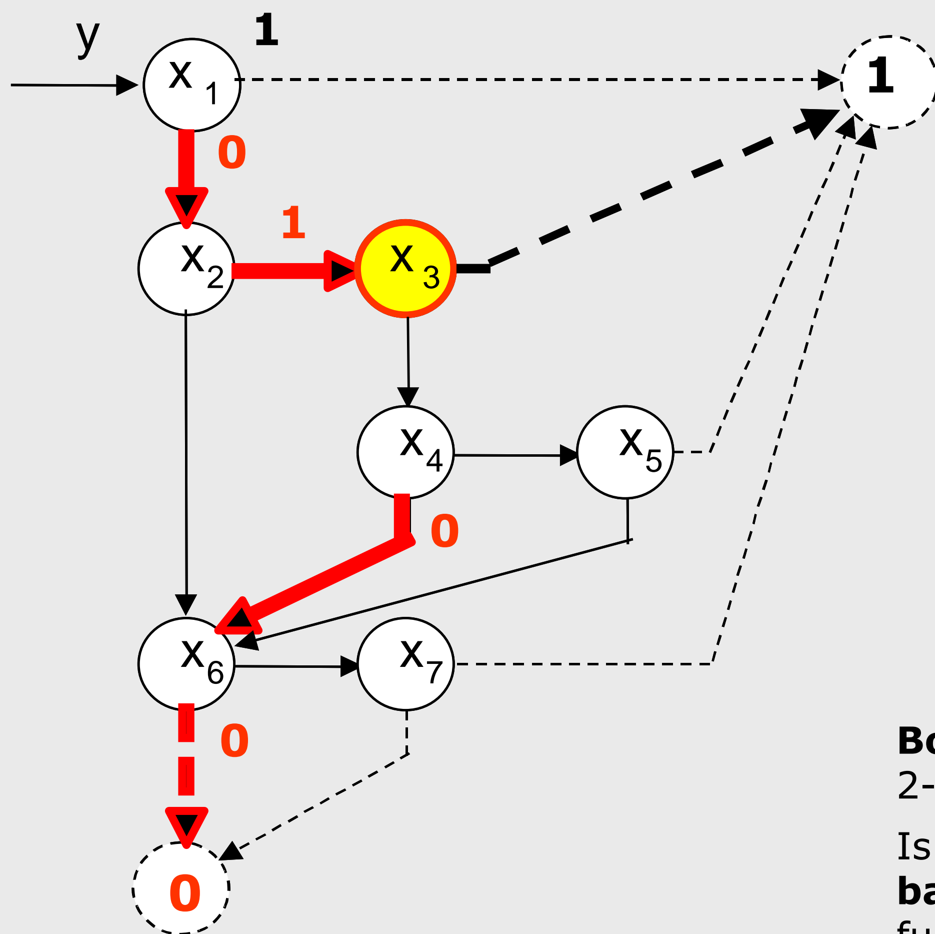
Complexity: Hierarchical Test Generation



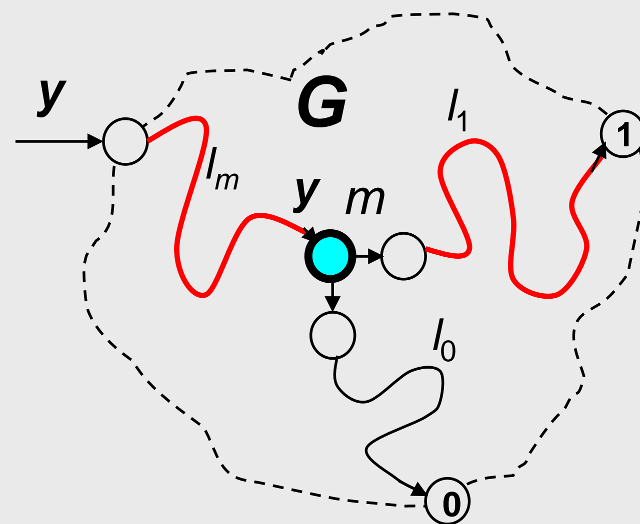
D_1	D_2	D
0	0	0
0	1	1
1	0	1

x_1	x_2	x_3	x_4	x_5	y
$\overline{D_2}$	0	D_1	1	1	D

Generalization of BDDs



The basic idea of simulation with BDDs:



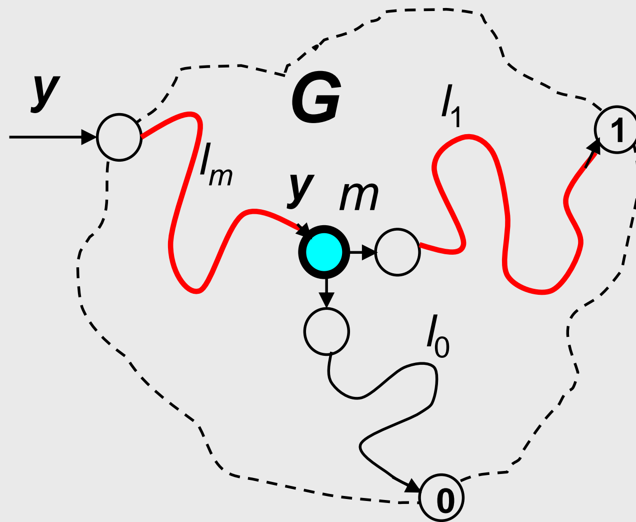
Boolean functions represent the **low** 2-valued logic **level**

Is it possible to use the topological **graph-based modeling** to generalize to **higher** functional **levels** like RTL

Generalization of BDDs

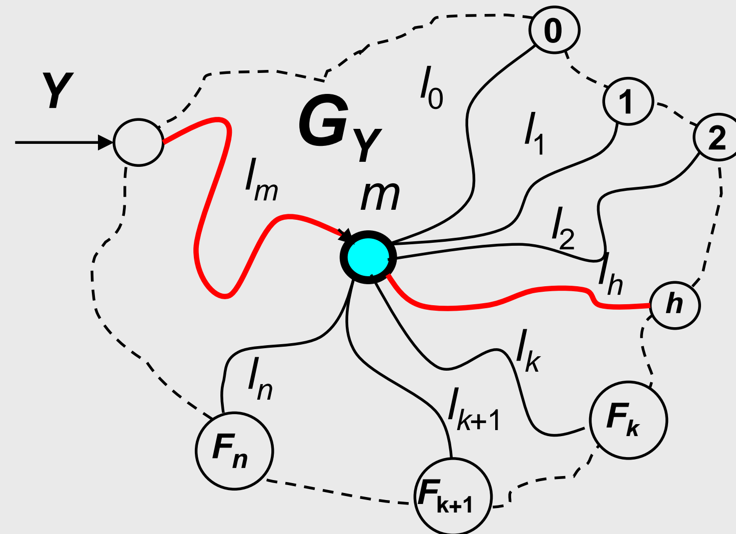
Binary DD

2 terminal nodes and
2 edges from each node



General case of DD

n ≥ 2 terminal nodes and
n ≥ 2 edges from each node



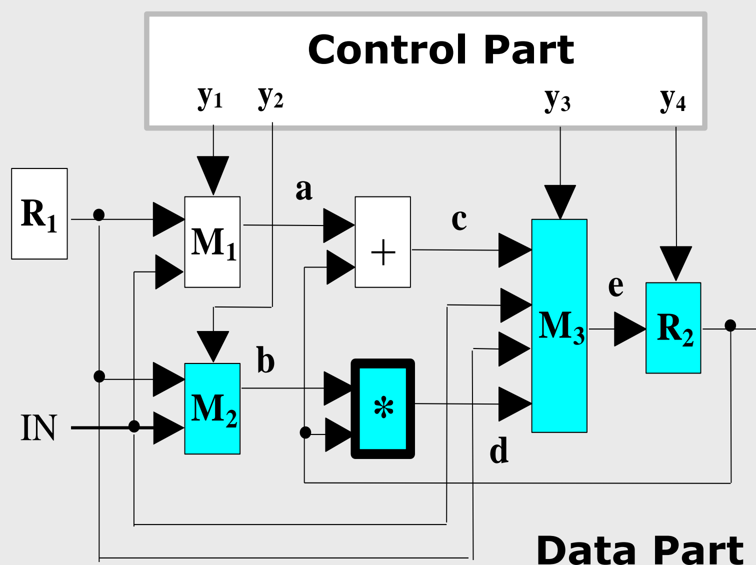
Novelty: **Boolean methods can be generalized in a straightforward way to higher functional levels**

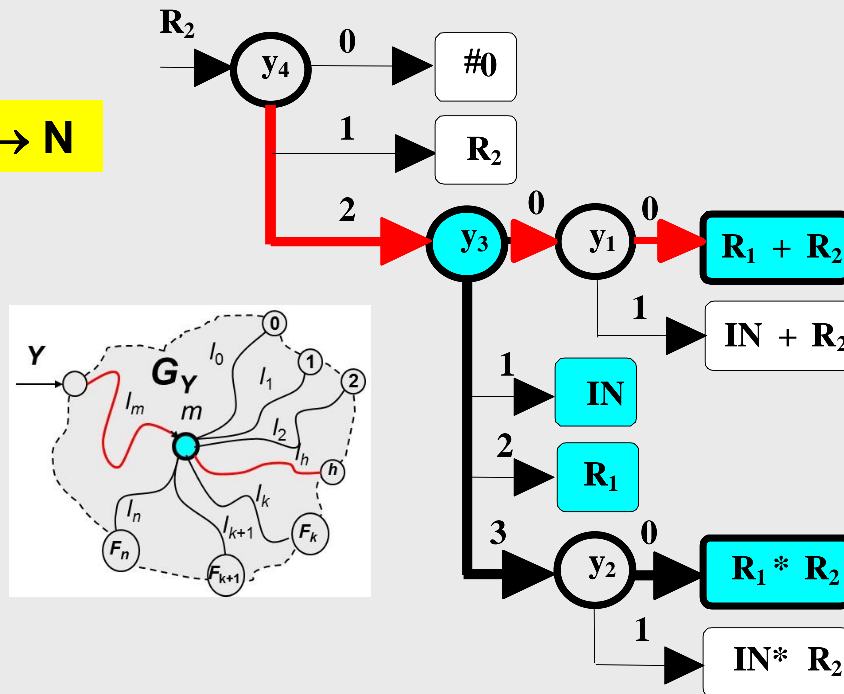
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

HLDD - High-Level Decision Diagrams

RTL-statement:

K: (*If* T,C) $R_D \leftarrow F(R_{S1},R_{S2},\ldots R_{Sm}), \rightarrow N$





Control part:
Internal nodes in the **HLDD**
Data Part:
Terminal nodes

Interpretation of HLDDs – Nodes and Faults

RTL-statement:

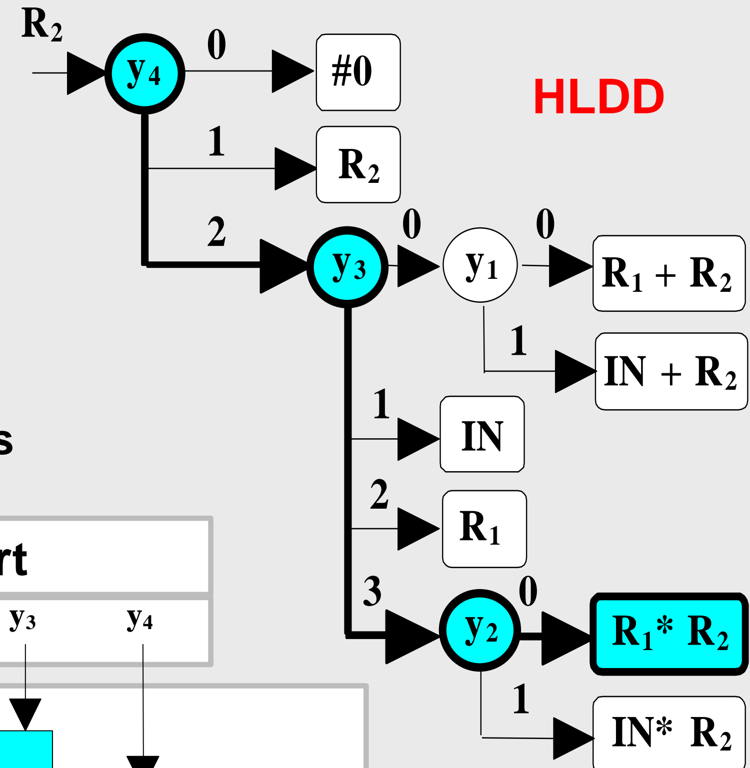
K: (*If* T,C) $R_D \leftarrow F(R_{S1},R_{S2},\ldots R_{Sm}), \rightarrow N$

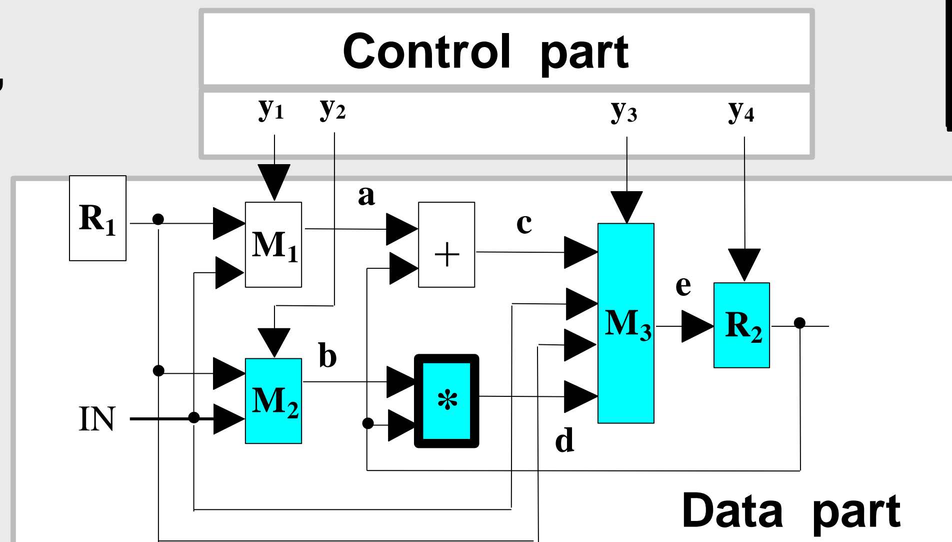
Nonterminal nodes

RTL-statement faults:
label,
timing condition,
logical condition,
register decoding,
operation decoding,
control faults

Terminal nodes

RTL-statement faults:
data storage,
data transfer,
data manipulation faults





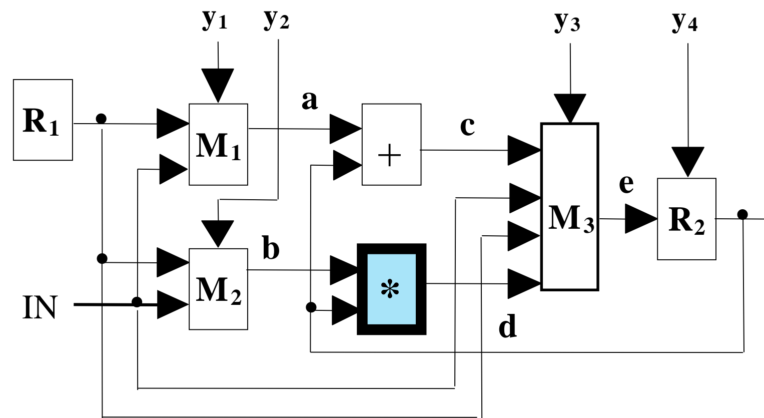
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

High Level Test Generation for Systems

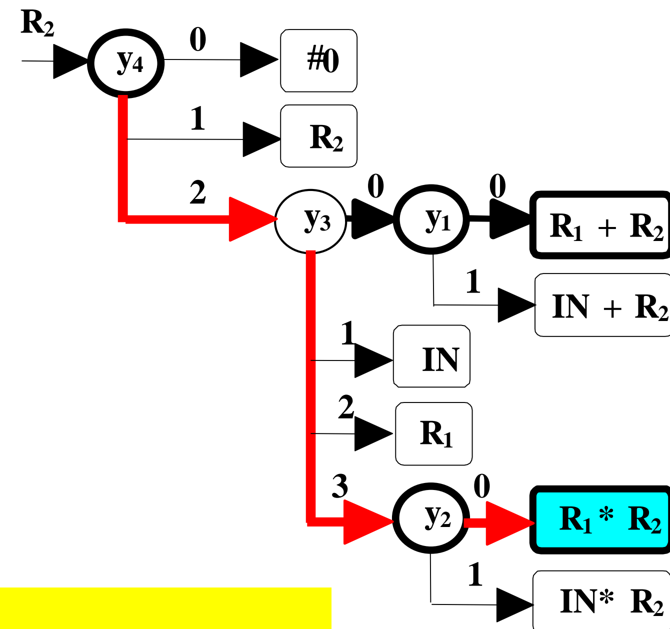
RTL test generation with DDs: Scanning test

Single path activation in a single DD
Data function $R_1 * R_2$ is tested

Data path



Decision Diagram



Test program: **Control:** $y_1 y_2 y_3 y_4$ = - 032

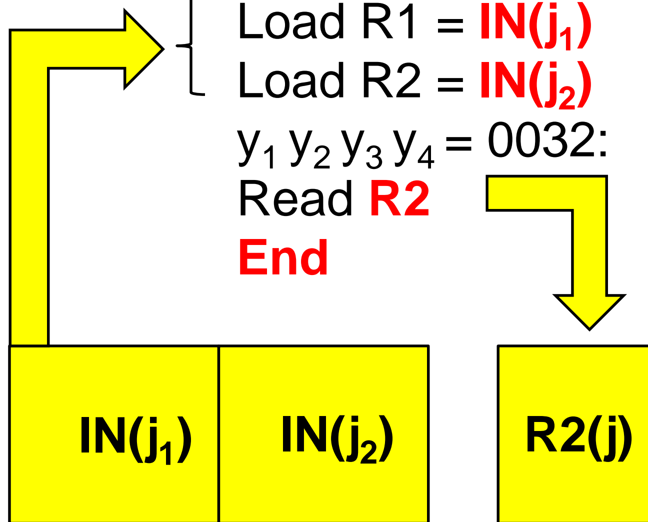
Data: *For all specified pairs of* (R_1, R_2)

Test Program Synthesis for Digital Systems

High-level test generation with DDs: Scanning test program

Test program:

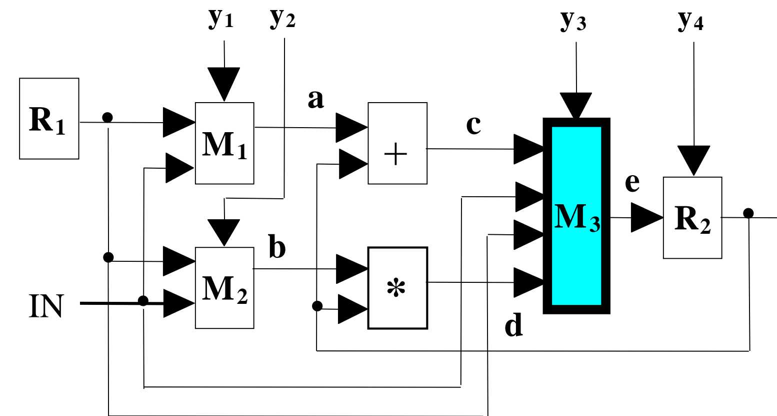
For j=1,n
  Begin
  Load R1 = **IN(j_1)**
  Load R2 = **IN(j_2)**
  $y_1\ y_2\ y_3\ y_4$ = 0032:
  Read **R2**
  End



Test data   Test results

Test template:

Control: $y_1\ y_2\ y_3\ y_4$ **= 0032**
Data: *For all specified pairs of* **(R_1, R_2)**

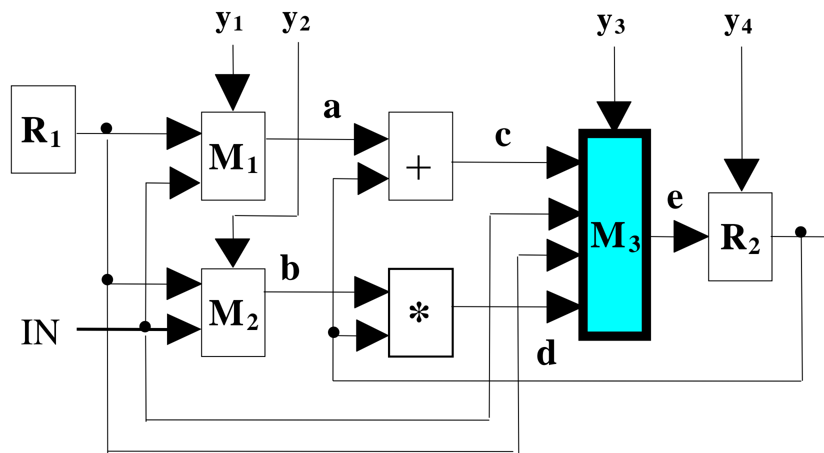


Test Generation with HLDDs for Systems

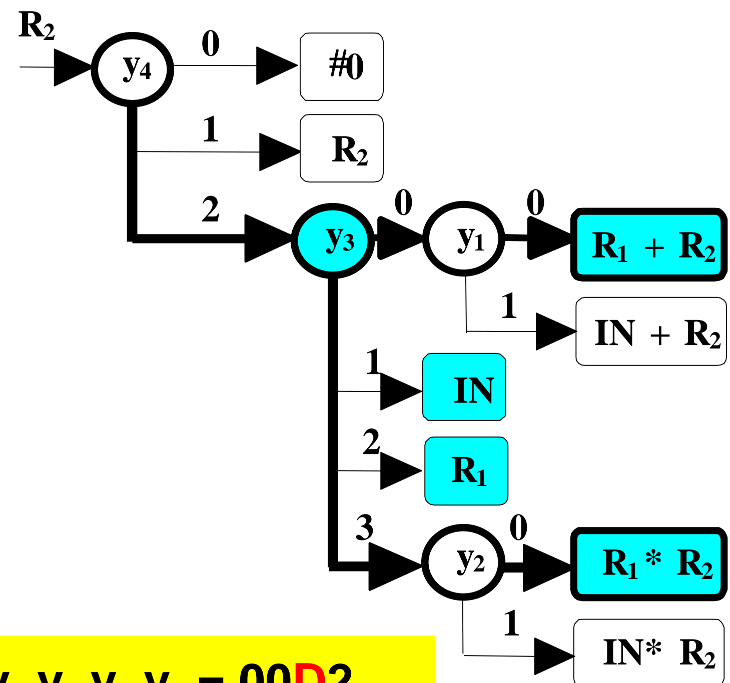
High-level test generation with DDs: Conformity test

Multiple paths activation in a single DD
Control function y_3 is tested

Data path



Decision Diagram



Test program:

Control: *For* **D** = 0,1,2,3: $y_1\, y_2\, y_3\, y_4$ = 00D2

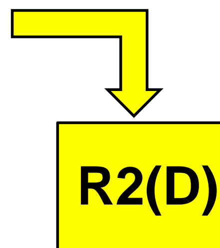
Data: *Solution of* $R_1 + R_2 \neq IN \neq R_1 \neq R_1 * R_2$

Test Generation with HLDDs for Systems

High-level test generation with DDs: Conformity test

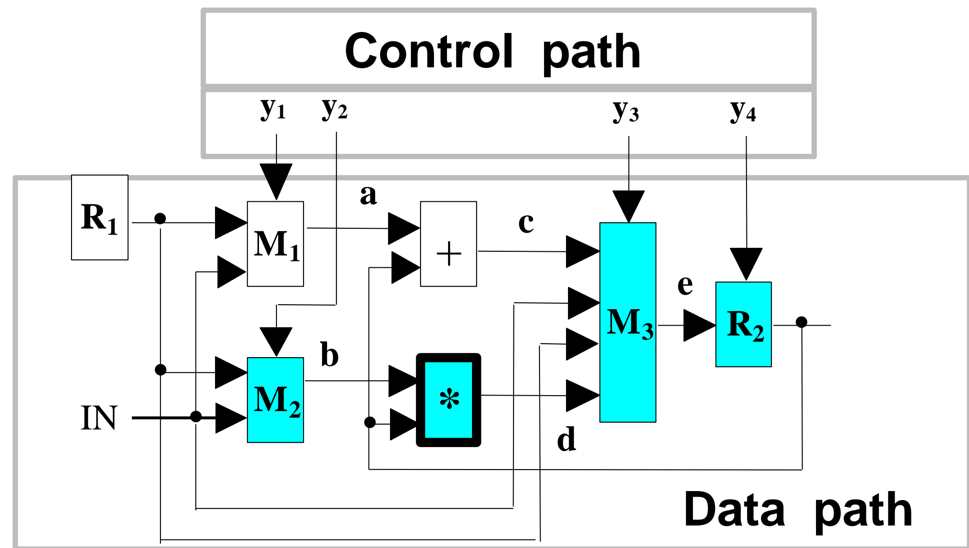
Test program:

For D = 0,1,2,3
Begin
Load R1 = **IN1**
Load R2 = **IN2**
Apply
IN = **IN3**
$y_1 y_2 y_3 y_4$ = 00**D**2
Read **R2**
End



Test template:

Control: *For* **D = 0,1,2,3:** $y_1 y_2 y_3 y_4 = 00D2$
Data: *Solution of* $R_1 + R_2 \neq IN \neq R_1 \neq R_1 * R_2$



Scan-Path for Making Systems Transparent

Hierarchical test generation with Scan-Path:

Scan-In

Control Part

y_1 y_2 y_3 y_4

Scan-Out

R_1 M_1 M_2 M_3 R_2 IN a b c d e + *

Bus

Data Part

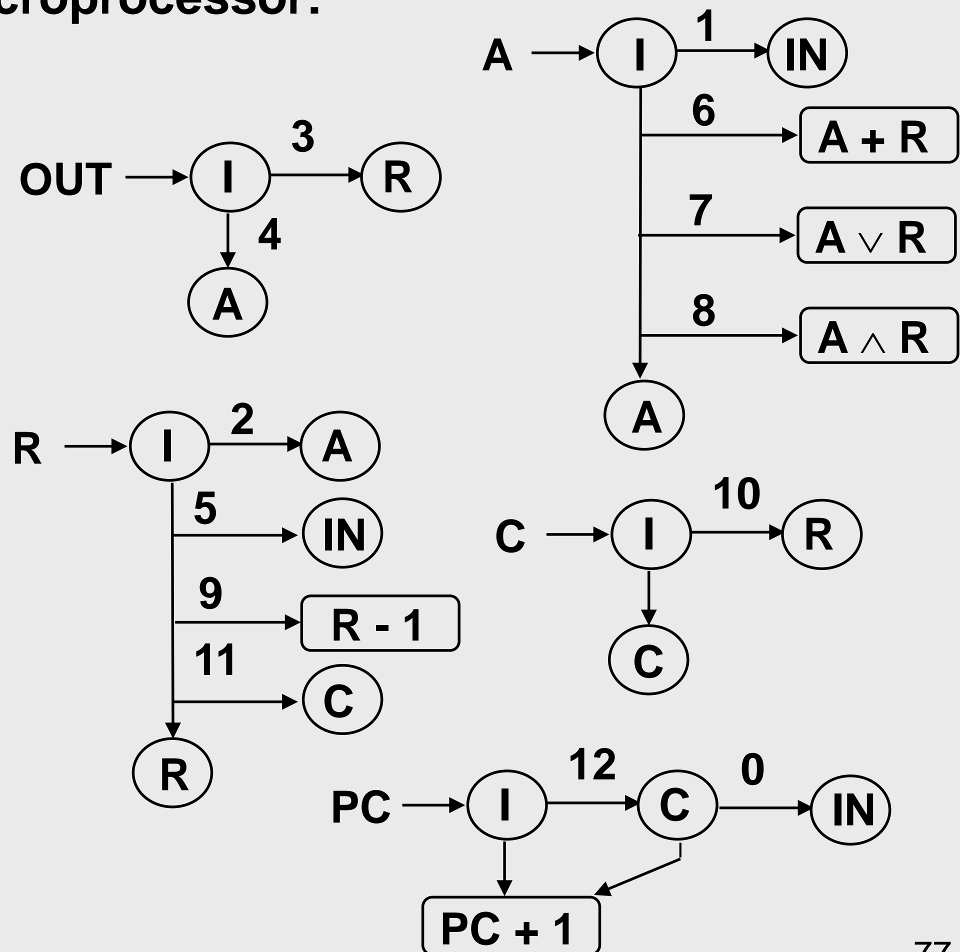
R_2 y_4 0 #0 1 R_2 2 y_3 0 y_1 0 $R_1 + R_2$ 1 $IN + R_2$ 1 IN 2 R_1 3 y_2 0 $R_1 * R_2$ 1 $IN * R_2$

Microprocessor Modeling with HLDDs

HLDD-model of a microprocessor:

Instruction set:

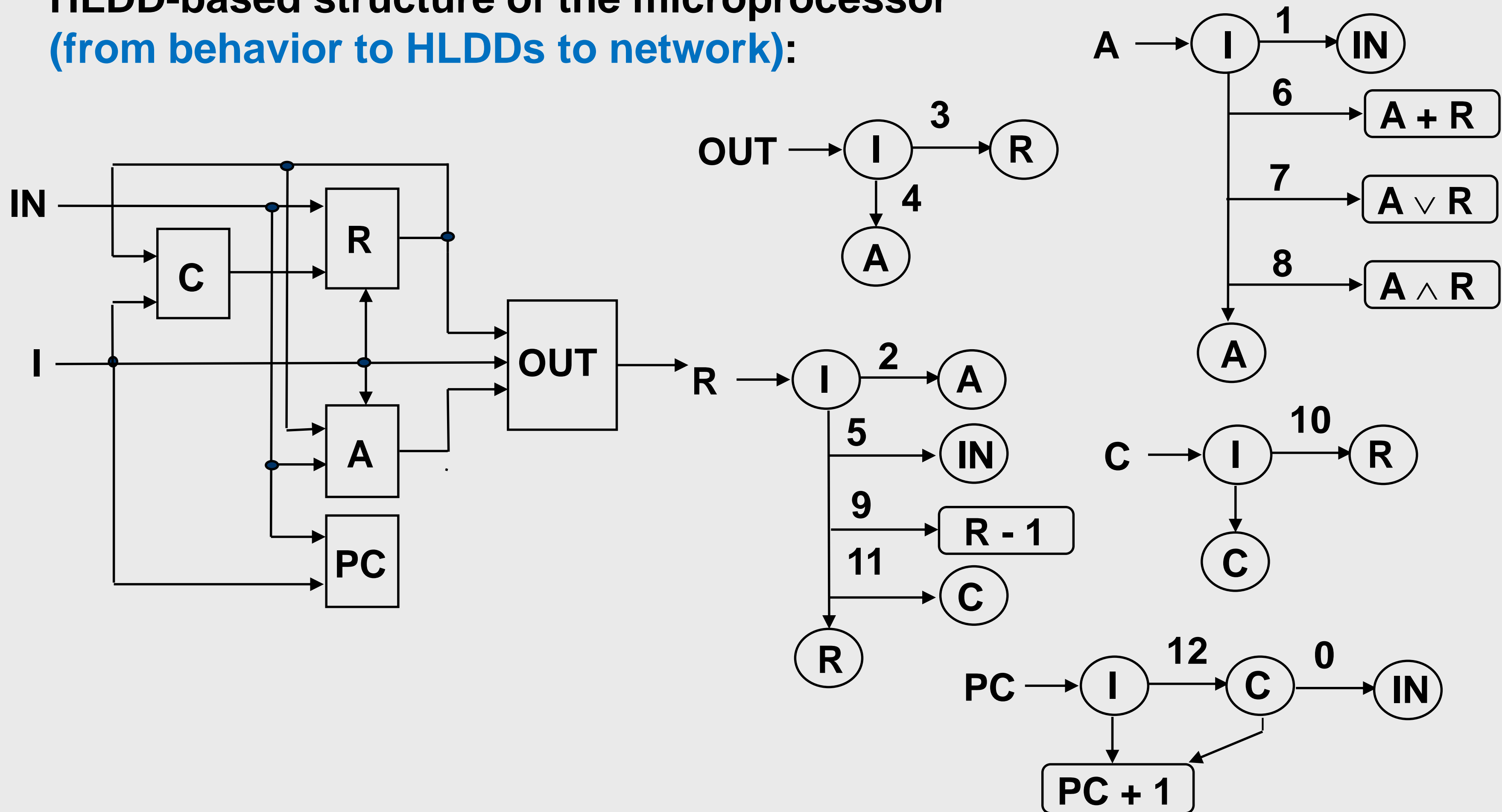
I_1:	MVI A,M	A ← IN
I_2:	MOV R,A	R ← A
I_3:	MOV M,R	OUT ← R
I_4:	MOV M,A	OUT ← A
I_5:	MOV R,M	R ← IN
I_6:	ADD R	A ← A + R
I_7:	ORA R	A ← A ∨ R
I_8:	ANA R	A ← A ∧ R
I_9:	SUB R	R ← R - 1
I_{10}:	MOV C,R	C← R
I_{11}:	CMA R,C	R ← C
I_{12}:	JMP PC, C	PC = IN IF C=0
	For all I_k	PC = PC + 1



1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Microprocessor Modeling with HLDDs

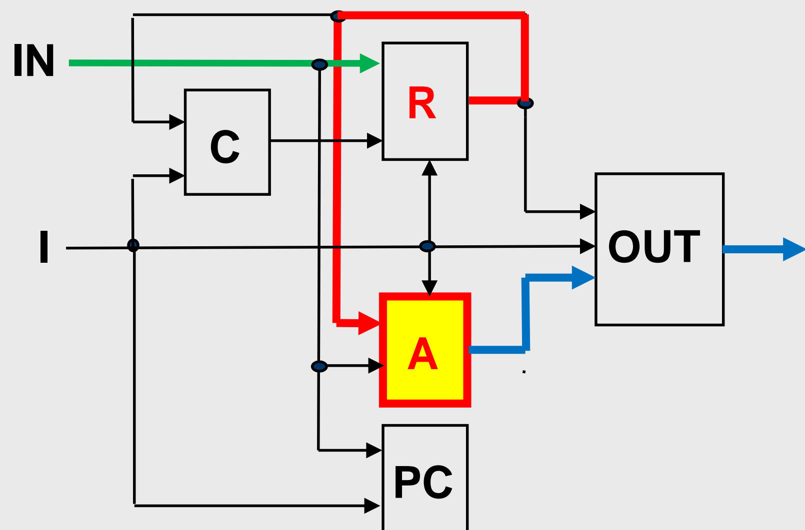
HLDD-based structure of the microprocessor (from behavior to HLDDs to network):

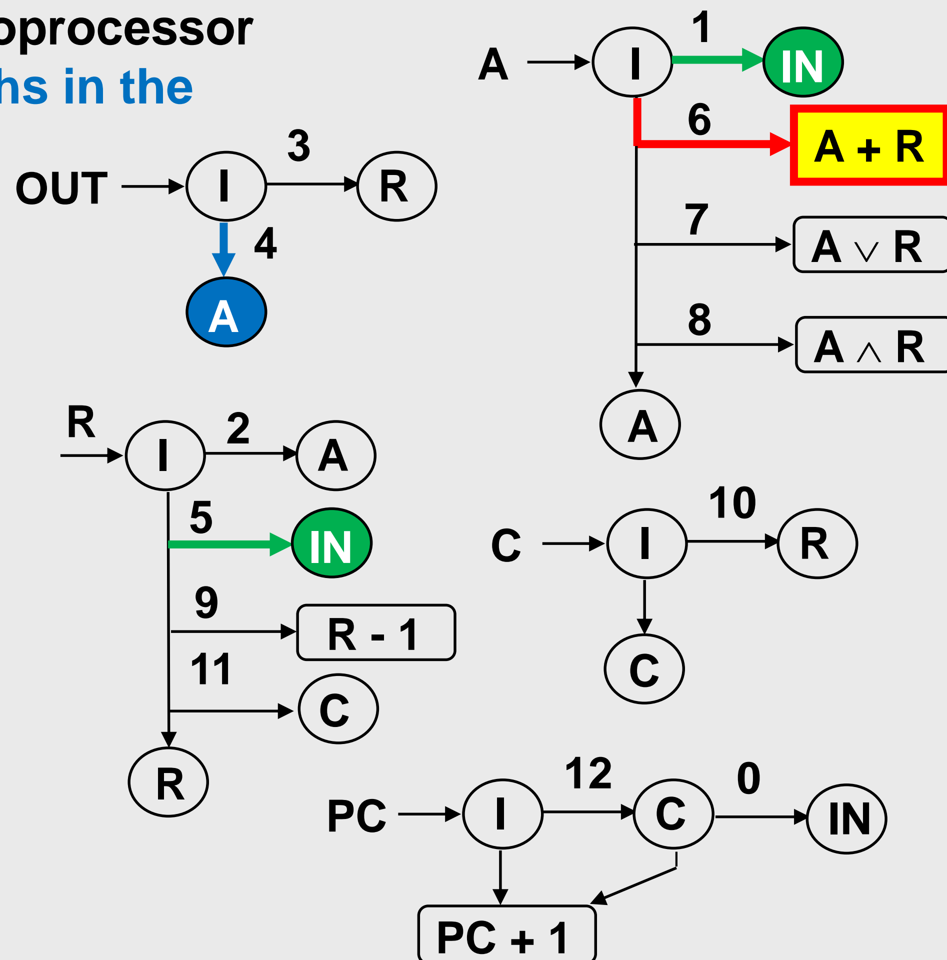


1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation for Microprocessors

HLDD-based structure of the microprocessor (propagation of faults through paths in the network):

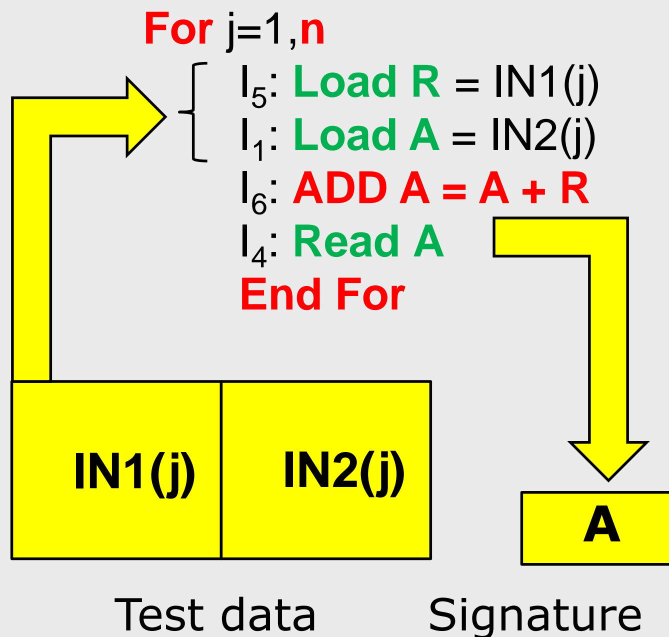


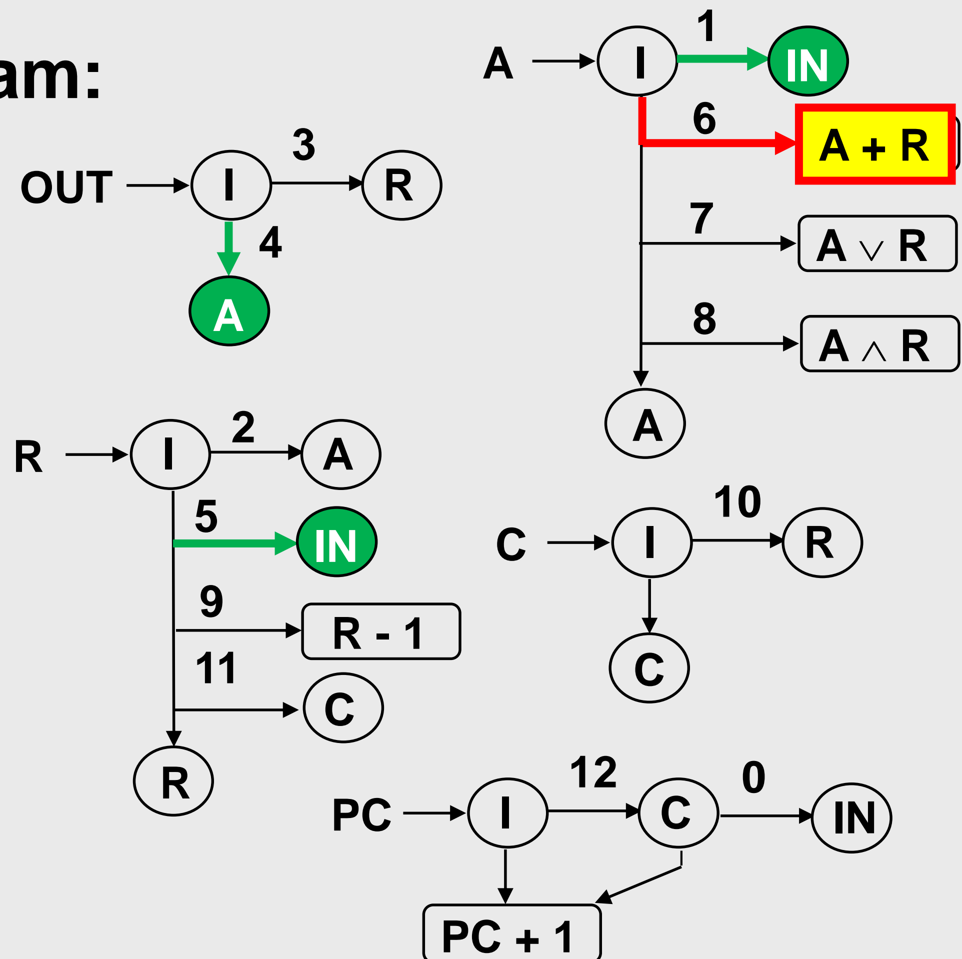


1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Test Generation for Microprocessors

Scanning test program:



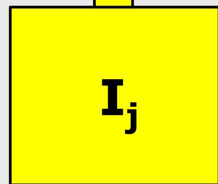


1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

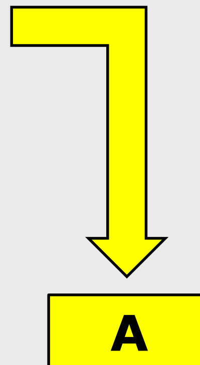
Test Generation for Microprocessors

Conformity test program:

For D=1,n
I_5: **Load R** = IN(1)
I_1: **Load A** = IN(2)
I_D: D
I_4: **Read A**
End For

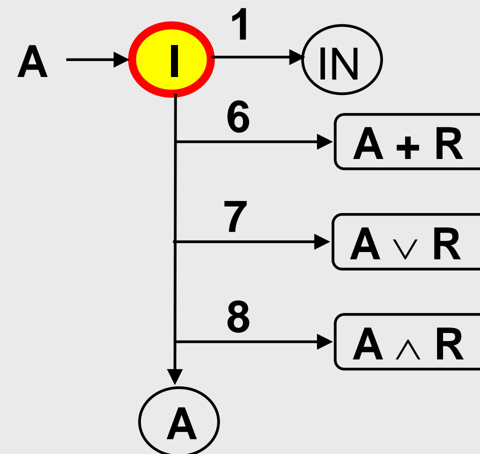


Test data



A

Signature



Added effects:

1) Special type of test compaction:
   - DD model
   - Test program template
   - Automated TPG
2) When testing all the functions of A with the same LOAD and READ conditions, the probability of fault masking will reduce
3) The faults of type "added erroneous actions" are as well easily tested

Algorithm:

Control: *For* **D = 0,1,….12: I_D**
Data: *Solution of* **IN ≠ (A + R) ≠ (A ∨ R) ≠ (A ∧ R)**

Modeling of Microprocessors with HLDDs

Instruction set

OP	B	Mnemonic	Semantic	RT level operations
0	0	LDA A1, A	READ memory	$R(A1) = M(A)$, $PC = PC + 2$
	1	STA A2, A	WRITE memory	$M(A) = R(A2)$, $PC = PC + 2$
1	0	MOV A1,A2	Transfer	$R(A1) = R(A2)$, $PC = PC + 1$
	1	CMA A1,A2	Complement	$R(A1) = \neg R(A2)$, $PC = PC + 1$
2	0	ADD A1,A2	Addition	$R(A1) = R(A1) + R(A2)$, $PC = PC + 1$
	1	SUB A1,A2	Subtraction	$R(A1) = R(A1) - R(A2)$, $PC = PC + 1$
3	0	JMP A	Jump	$PC = A$
	1	BRA A	Conditional jump (Branch instruction)	IF C=1, THEN $PC = A$, ELSE $PC = PC + 2$

Instruction code:

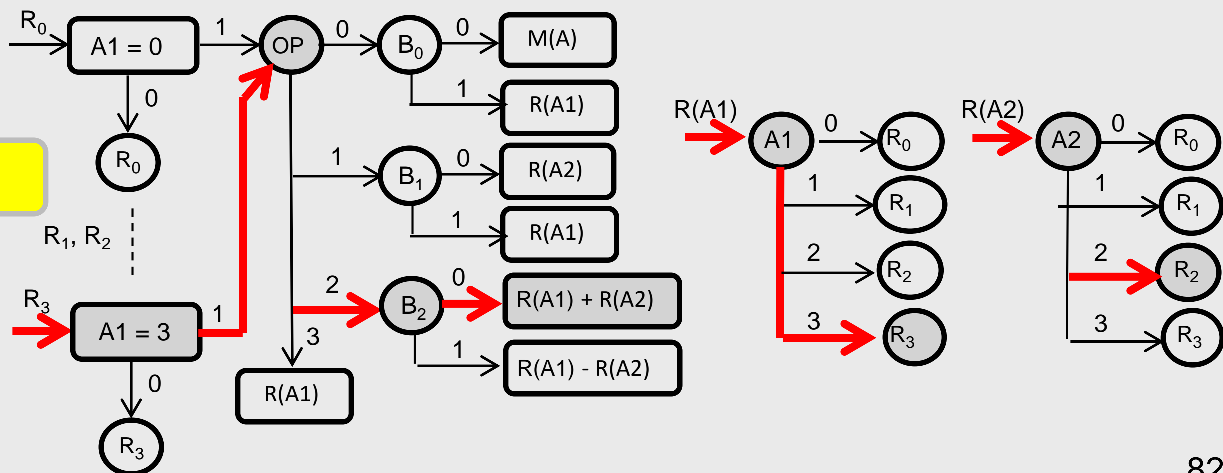
ADD A1 A2

OP=2. B=0. A1=3. A2=2

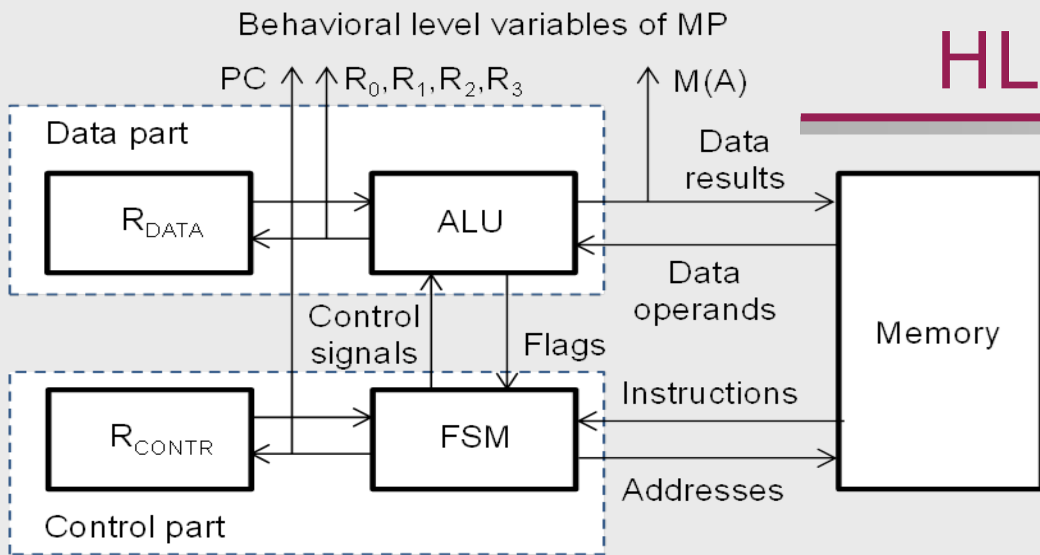
R(A1) = R(A1) + R(A2)

$R_3 = R_3 + R_2$

$PC = PC+1$



HLDDs for MP InstrSet

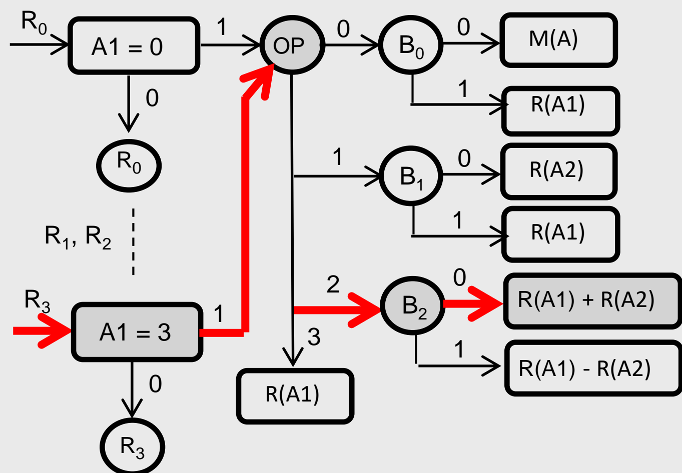


Instruction code:
ADD A1 A2
OP=2. B=0. A1=3. A2=2

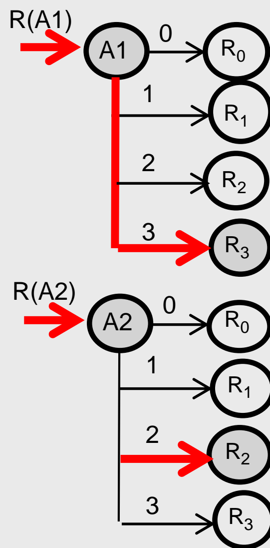
$R_3 = R_3 + R_2$

PC = PC+1

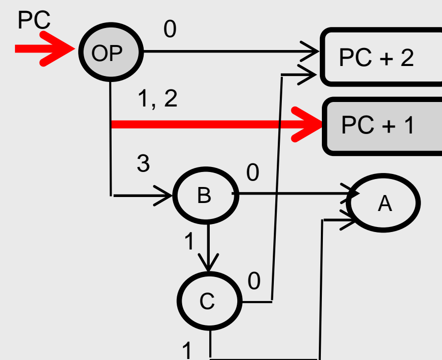
Registers and ALU



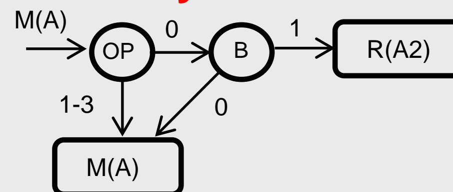
Register Decoding



Program Counter



Memory Access



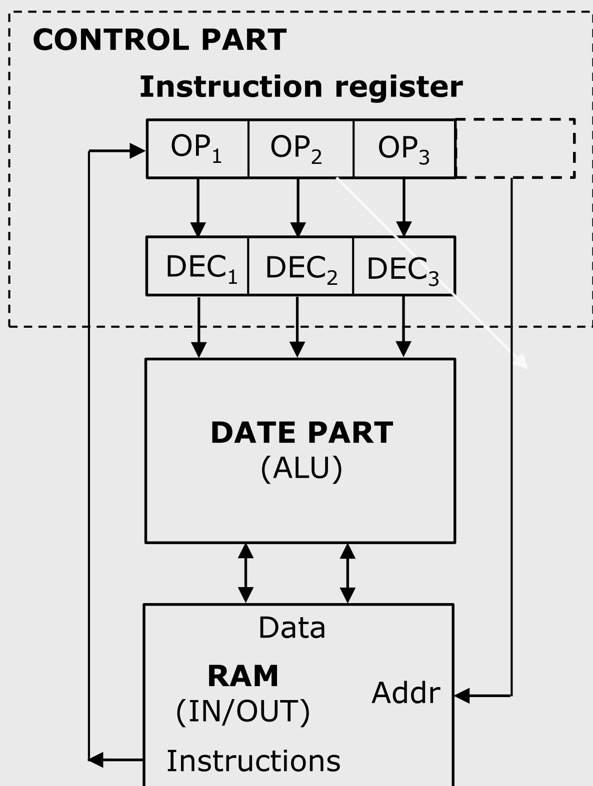
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Modeling of Microprocessors with HLDDs

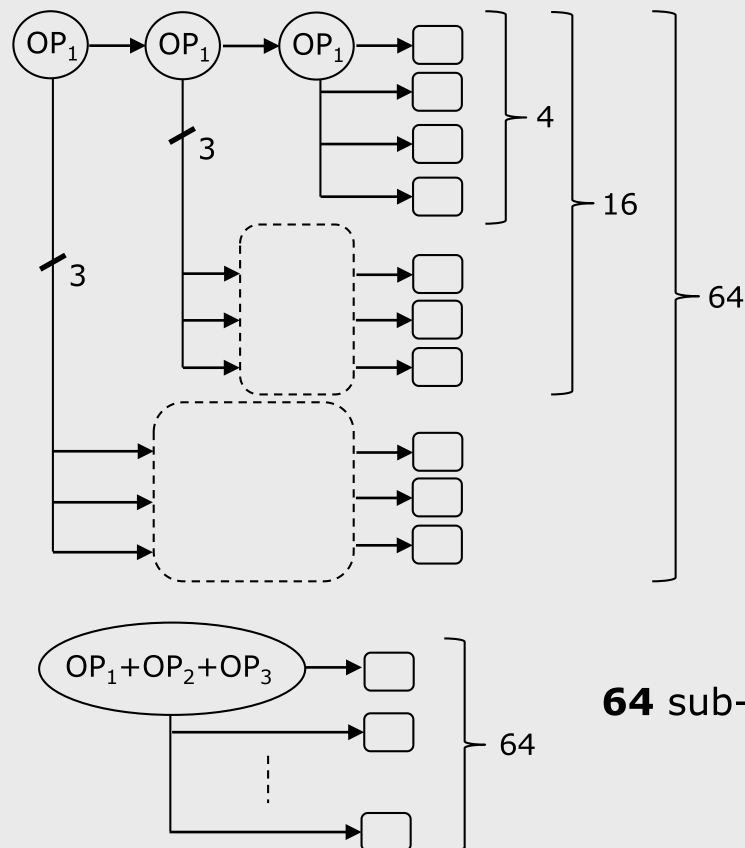
Microprocessor instruction set architecture

HLDD

12 control sub-tests
64 data part sub-tests



?



64 sub-tests

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

© Raimund Ubar

Fault Coverage Table and FC Measure

Functional fault model:

$\forall j\ [f_j \neq \text{ZERO})]$

$\forall i,j\text{: } \forall k\ [(f_{i,k} < f_{j,k}]$

Distinguished Operations f_i		Distinguished Operations f_j								
		f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
		MOV	ADD	SUB	CMP	AND	OR	XOR	NOT	NOP
f_0	MOV	00000000	00110000	00010000	00100000	00000000	00100000	00100000	00110000	00000000
f_1	ADD	01001000	00000000	00001000	01001000	01001000	01001000	00000000	00000000	00000000
f_2	SUB	11000110	10100110	00000000	11100000	11000000	11100110	00100110	00100000	00000000
f_3	CMP	00000111	00010111	00010001	00000000	00000000	00000111	00000111	00010000	00000000
f_4	AND	00000111	00110111	00010001	00100000	[illegible]	[illegible]	[illegible]	[illegible]	00000000
f_5	OR	00000000	00010000	00010000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
f_6	XOR	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
f_7	NOT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
f_8	NOP	11001111	10110111	00011001	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

$$f_{i,k} < f_{j,k}$$

Fault coverage measure:
Percentage of 1-s
in the fault coverage table

1 – means that
a constraint is satisfied
by at least one pair of data operands

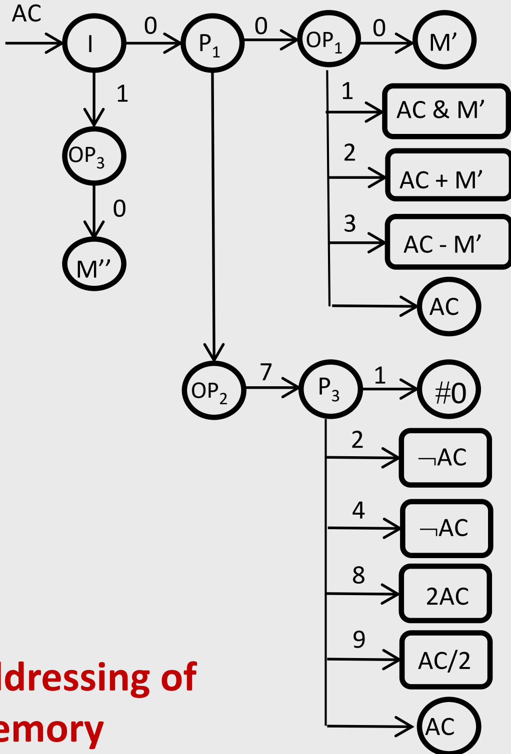
Parwan Microprocessor : Instruction Set

	OP	
LDA	0	AC=M
AND	1	$AC=AC \wedge M$
ADD	2	$AC=AC \wedge M$
SUB	3	$AC=AC \wedge M$
JMP	4	PC=A
STA	5	M=AC
JSR	6	PC=A Jump to subroutine

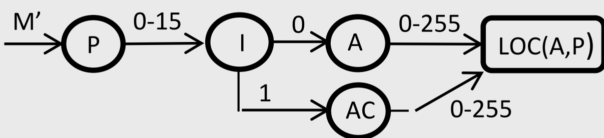
	OP	I	P	
CLA	7	0	1	AC=0
CMA	7	0	2	$AC= \neg AC$
CMC	7	0	4	$C= \neg C$
ASL	7	0	8	AC=2AC
ASR	7	0	9	AC=AC/2
BRA_N	7	1	0	If negative
BRA_Z	7	1	2	If zero
BRA_C	7	1	4	If carry
BRA_V	7	1	8	If overflow

Parwan: HLDD Model

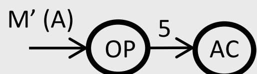
ALU Data Path



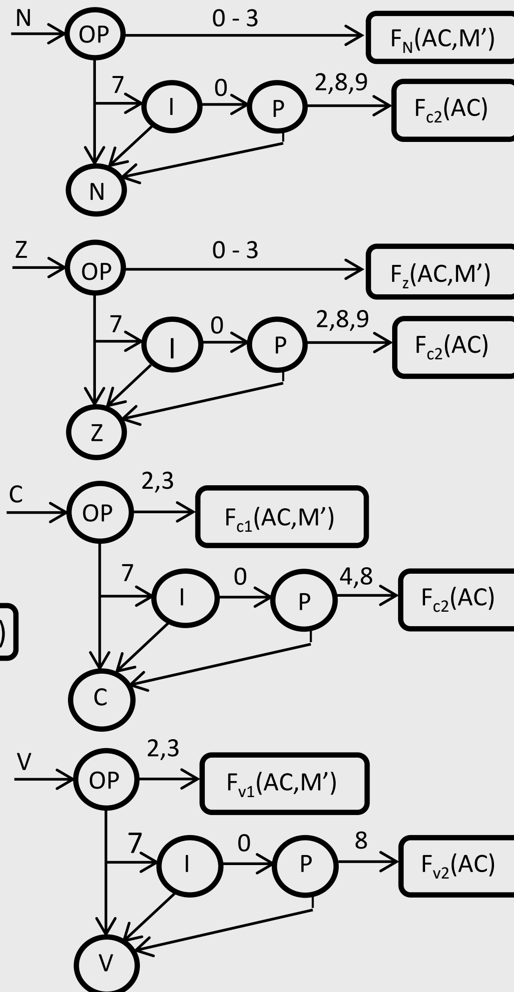
Addressing of memory



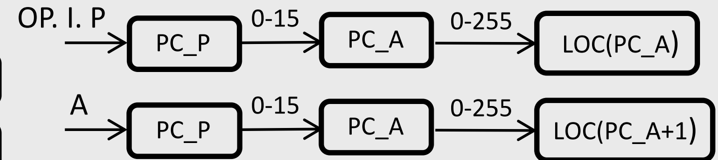
Output behaviour



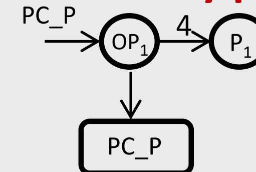
ALU Flags



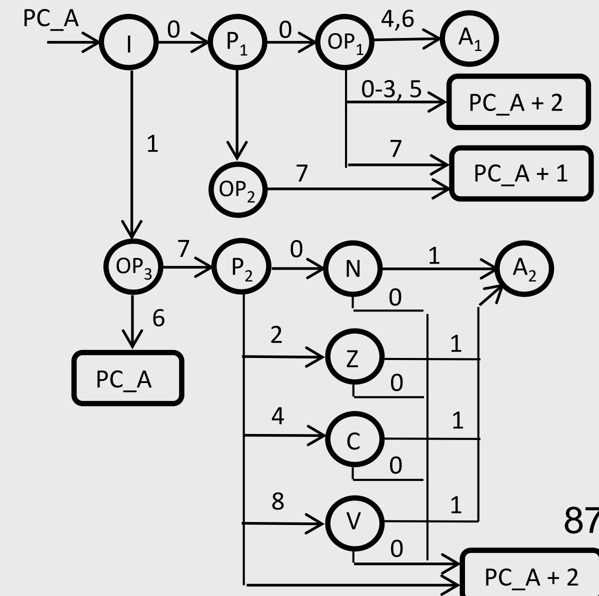
Instruction addressing



Next memory page calculation

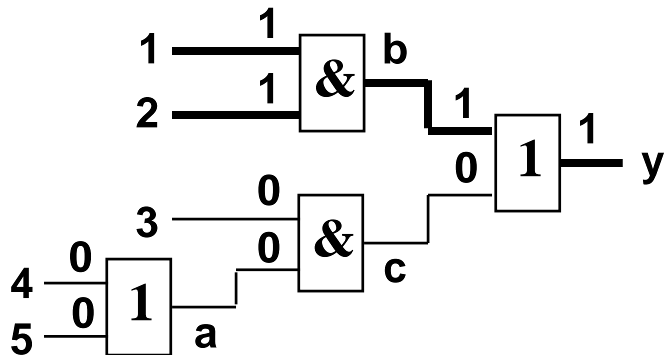


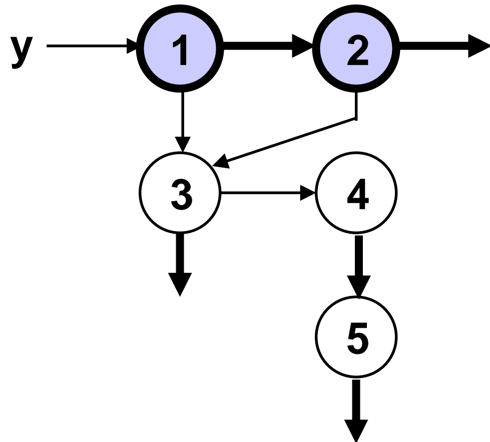
Next PC offset calculation



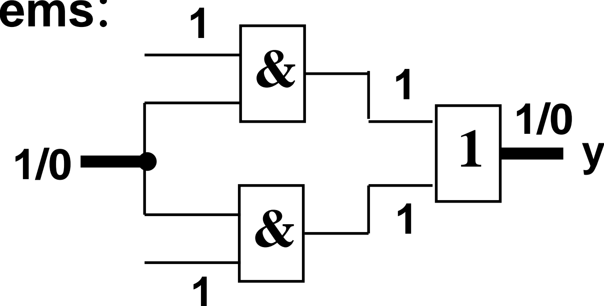
1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Fault Simulation: With BDDs

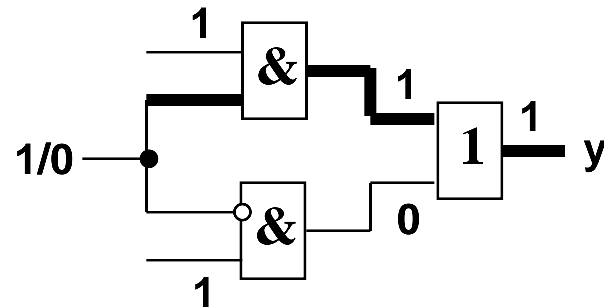




Problems:



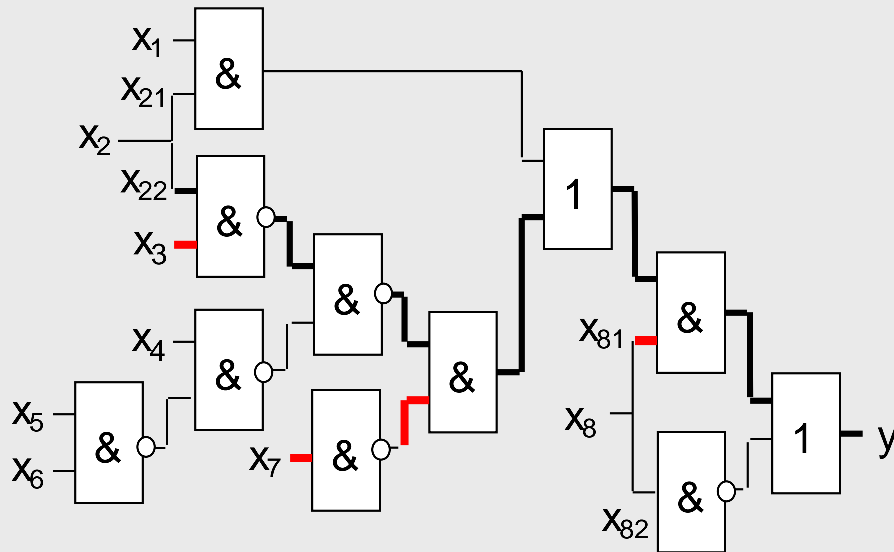
The critical path is not continuous

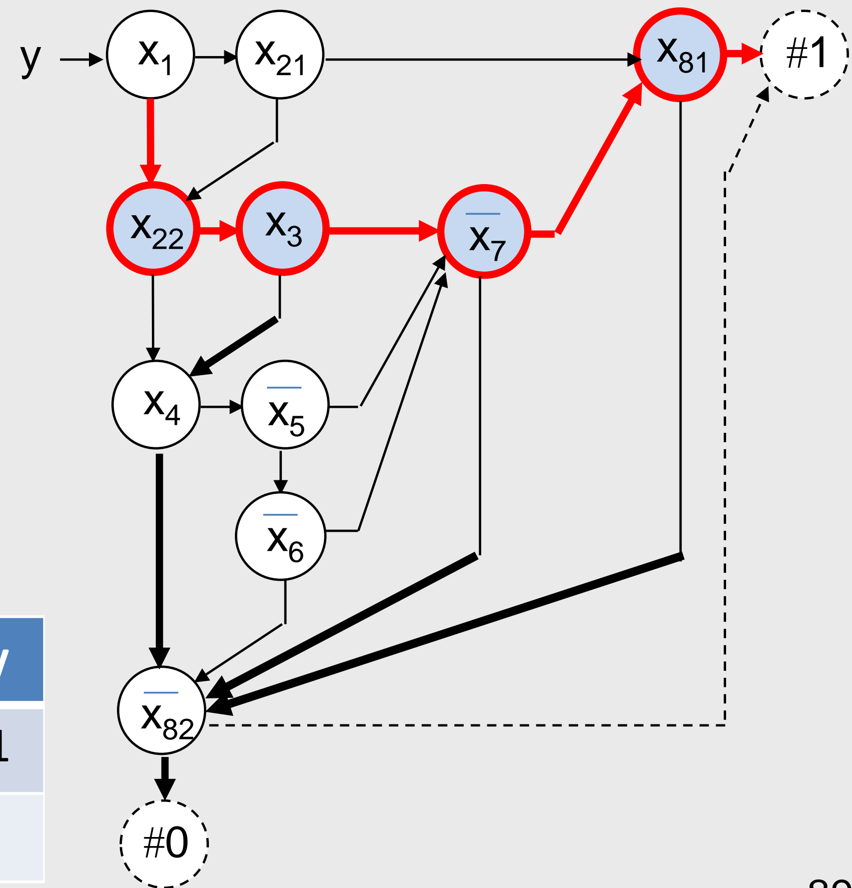


The critical path breaks on the fan-out

Fault Simulation with BDDs

$$y = f(X) = (x_1 x_{21} \vee (x_{22} x_3 \vee x_4(\overline{x_5} \vee \overline{x_6}))\overline{x_7})x_{81} \vee \overline{x_{82}}$$





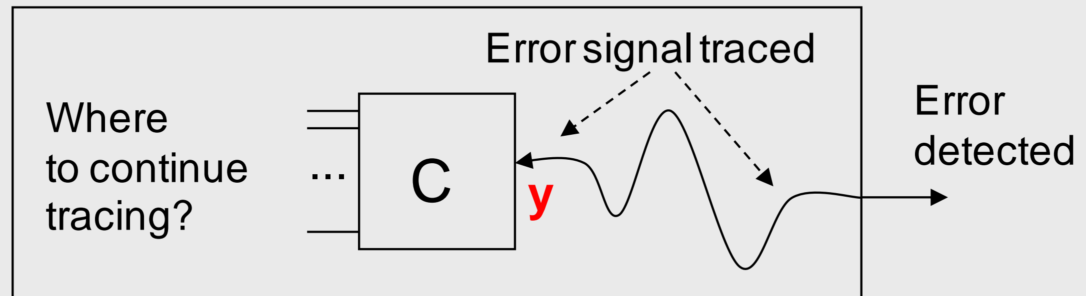
X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
X^t	0	1	1	0	-	-	0	1	1
Tested: $x_{22} \equiv 0$, $x_3 \equiv 0$, $x_7 \equiv 1$, $x_{81} \equiv 0$									

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

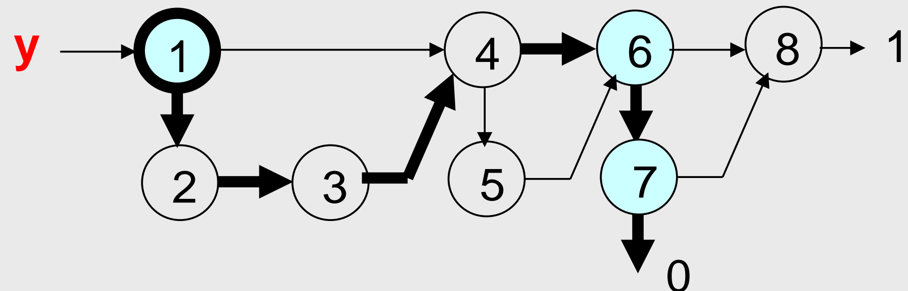
Fault Diagnosis with BDDs

Sequential fault diagnosis by **Signal Pinpointing**

Fault diagnosis / Fault simulation:



Property 2:
If a test vector X
activates in SSBDD a **0**-path (**1**-path)
which travers a subset of nodes M,
then only **0**-nodes
(**1**-nodes) have to be considered as fault
candidates



Fault diagnosis and fault simulation can be speed-up by using Property 2

Speeding-up simulation:
M = {1,2,3,4,6,7}
M* = {1,6,7} – by Property 2
M** = {**6,7**} – by Property 1

Only **6** and **7** have to be considered

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY