Overview: Testability Evaluation

Outline

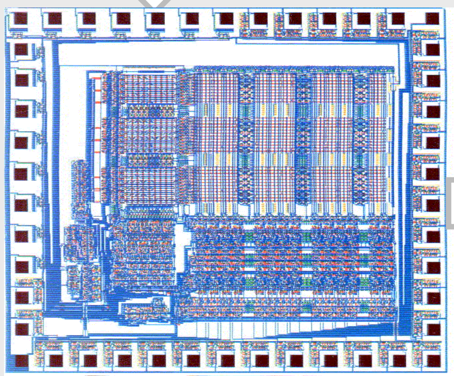
- Quality Policy of Electronic Design
- Tradeoffs of Design for Testability
- Testability measures
  - Heuristic measures
  - Probabilistic measures
- Calculation of testability
  - Parker - Mc Cluskey method
  - Cutting method
  - Conditional probabilities based method

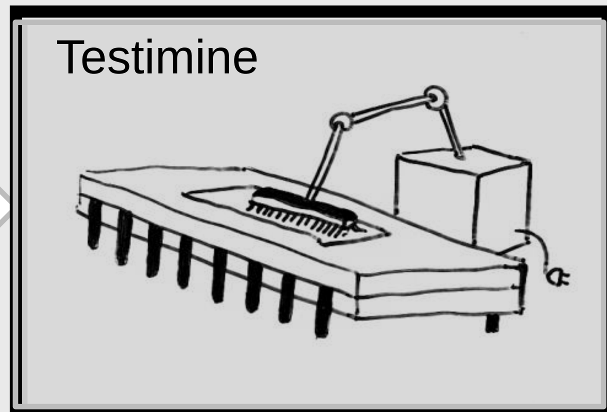
Quality Policy

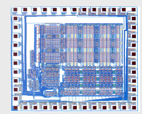
Chips from manufactory

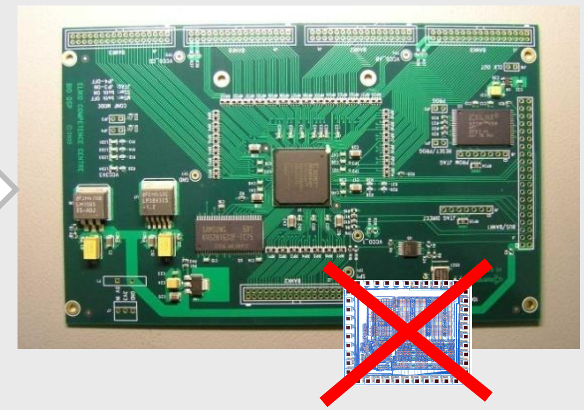
Yield

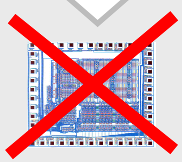
For example, yield is 60%. Other chips are faulty







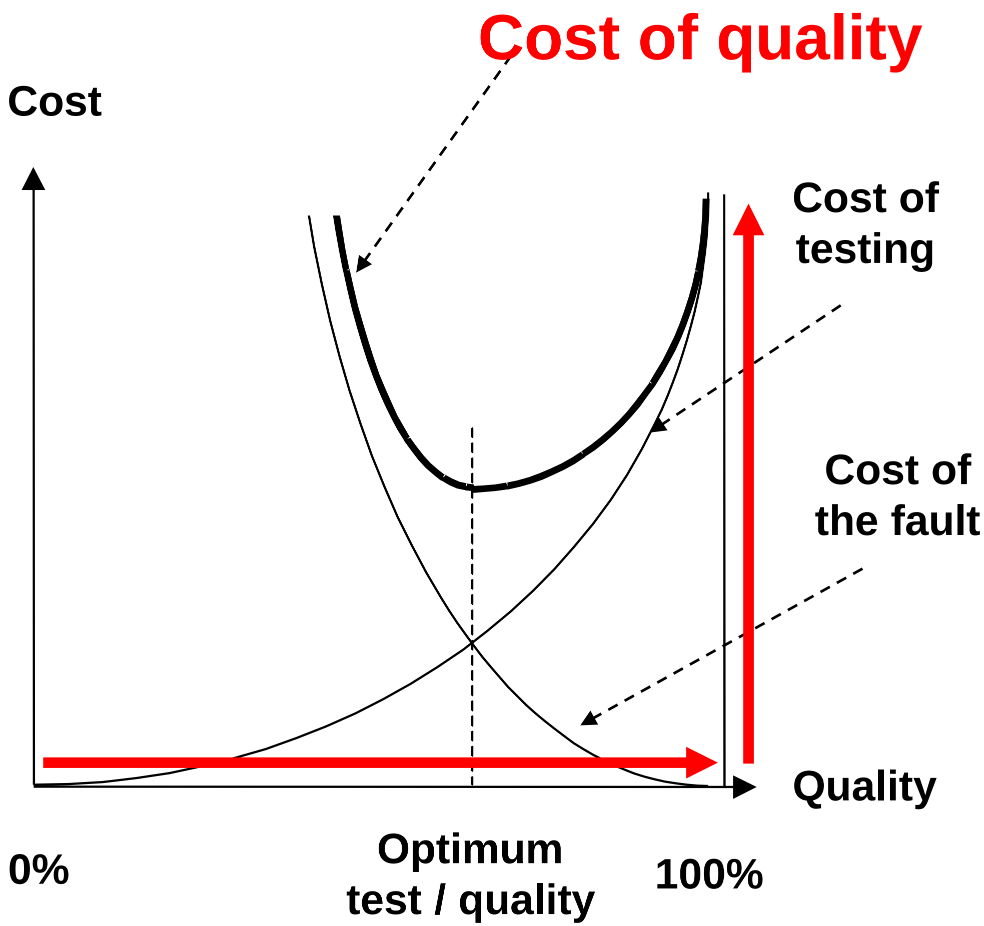




Defect level means:

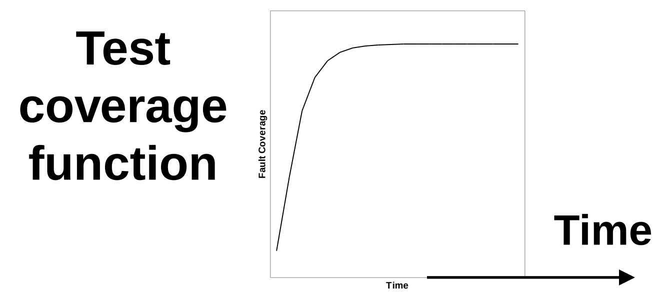
How many faulty chips from 40% escape?

Introduction: The Problem is Money?



How to succeed?
Try too hard!

How to fail?
Try too hard!
(From American Wisdom)



<u>*Conclusion:*</u>

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

T.Williams

Design for Testability

The problem is - QUALITY:

Yield (Y) → **Quality policy** → Defect level (DL)

P, n

P_a

Design for testability

Testing

P - probability of a defect
n - number of defects
P_a - probability of accepting a bad product

$$Y = (1 - P)^n$$ - probability of producing a good product

Design for Testability

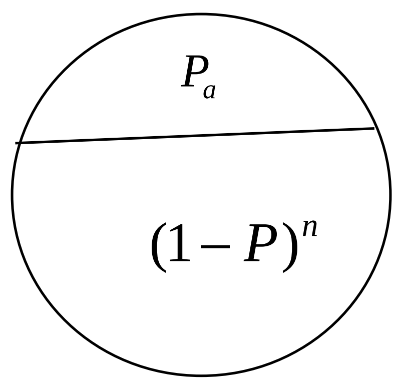
The problem is - QUALITY:

Yield (*Y*) → **Quality policy** → **Defect level (*DL*)**

P*,*n ... P_a

$$\boxed{DL} = \frac{P_a}{(1-P)^n + P_a} = 1-(1-P)^{n-m} = 1-Y^{\frac{n-m}{n}} = 1-Y^{(1-\frac{m}{n})} = \boxed{1-Y^{(1-T)}}$$

$$P_a = (1-P)^m - (1-P)^n$$



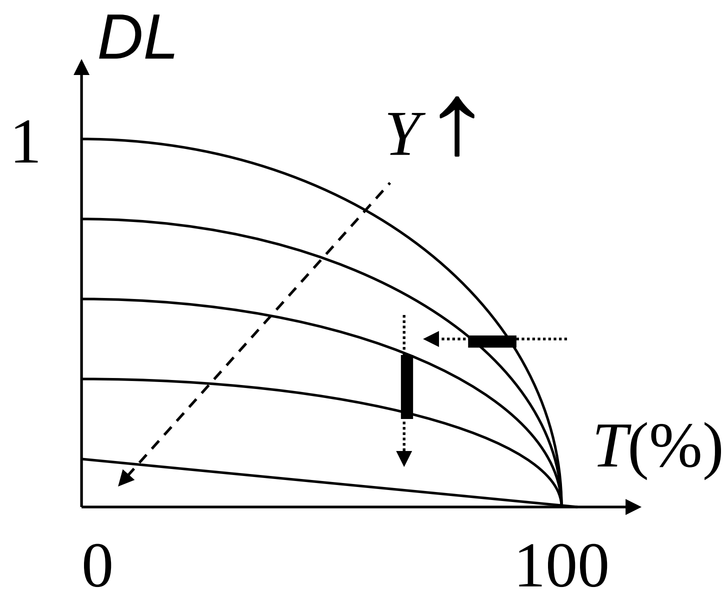
$$\boxed{Y = (1-P)^n}$$

n - number of defects
m - number of faults tested
P - probability of a defect
P_a - probability of accepting a bad product
T - test coverage

Design for Testability

The problem is - Money:

$$DL = 1 - Y^{(1-T)}$$



Y(%) \ T(%)	10	50	90
90	**8**	**5**	**1**
50	**45**	**25**	**5**
10	**81**	**45**	**9**

Goal: $DL \downarrow \leftarrow T \uparrow \leftarrow$ **Testability** $\uparrow$

Paradox: **Testability** $\uparrow \rightarrow DL \uparrow (Y \downarrow)$

Design for Testability

Technical tradeoff:

Goal: $DL \downarrow \leftarrow T \uparrow \leftarrow$ **Testability** $\uparrow$

Paradox: **Testability** $\uparrow \rightarrow DL \uparrow (Y \downarrow)$

DFT: **Resynthesis or adding extra hardware**

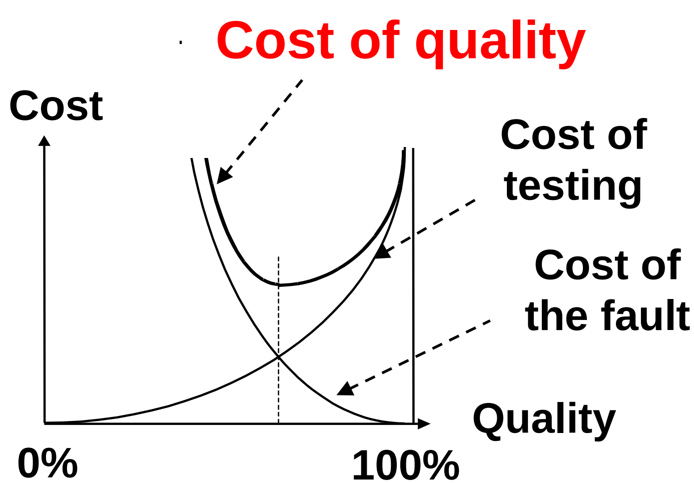
Logic complexity $\uparrow$
Area $\uparrow$
Number of I/O $\uparrow$

Performance $\downarrow$

Power consumption $\uparrow$
Yield $\downarrow$

Economic tradeoff:

$$C\,(\text{Design} + \text{Test}) < C\,(\text{Design}) + C\,(\text{Test})$$



Design for Testability

Economic tradeoff:

$$C\ (\text{Design} + \text{Test}) < C\ (\text{Design}) + C\ (\text{Test})$$

$$C\ (\text{DFT}) + C\ (\text{Test'}) < C\ (\text{Design}) + C\ (\text{Test})$$

$$C\ (\text{Test}) = C_{TGEN} + (C_{APLIC} + (1 - Y)\ C_{TS})\ Q$$

Test generation

Testing

Troubleshooting

Volume

Design

Product

$$C\ (\text{DFT}) = (C_D + \Delta C_D) + Q(C_P + \Delta C_P)$$

Testability Criteria

Qualitative criteria for Design for testability:

Testing cost:

- Test generation time
- Test application time
- Fault coverage
- Test storage cost (test length)
- Availability of Automatic Test Equipment

The cost of re-design for testability:

- Performance degradation
- Area overhead
- I/O pin demand

Testability of Design Types

General important relationships:

1. T (Sequential logic) < T (Combinational logic)

   Solutions: **Scan-Path design strategy**

2. T (Control logic) < T (Data path)

   Solutions: **Data-Flow design, Scan-Path design strategies**

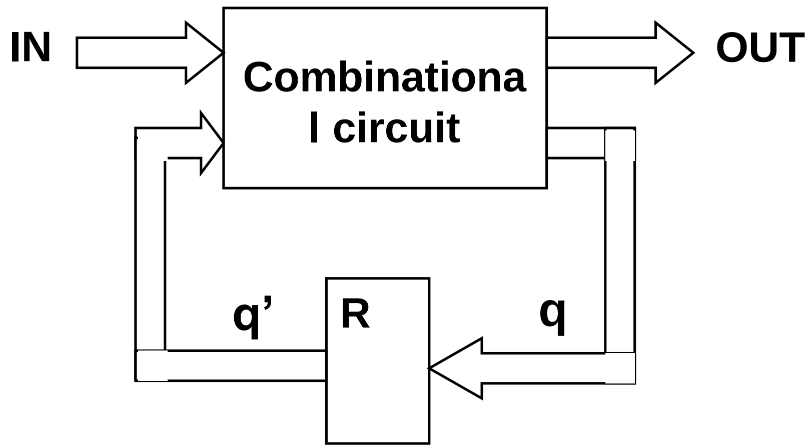
3. T (Random logic) < T (Structured logic)

   Solutions: **Bus-oriented design, Core-oriented design**

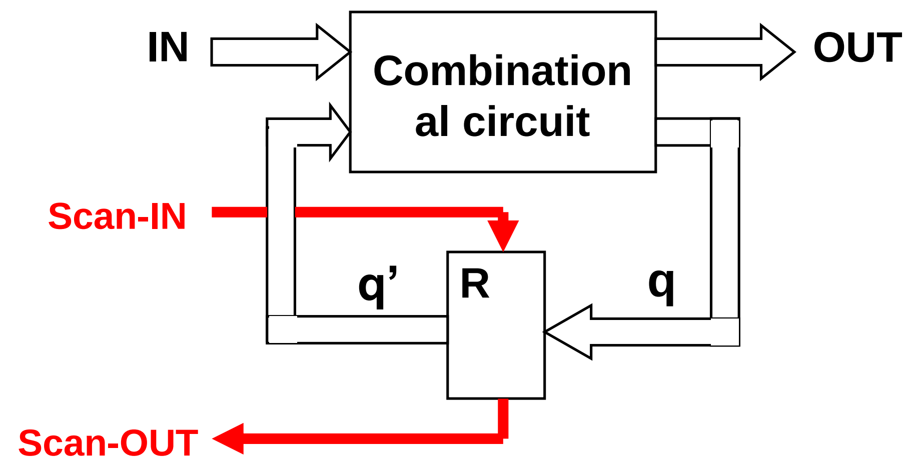
4. T (Asynchronous design) < T (Synchronous design)

Testability of Design Types

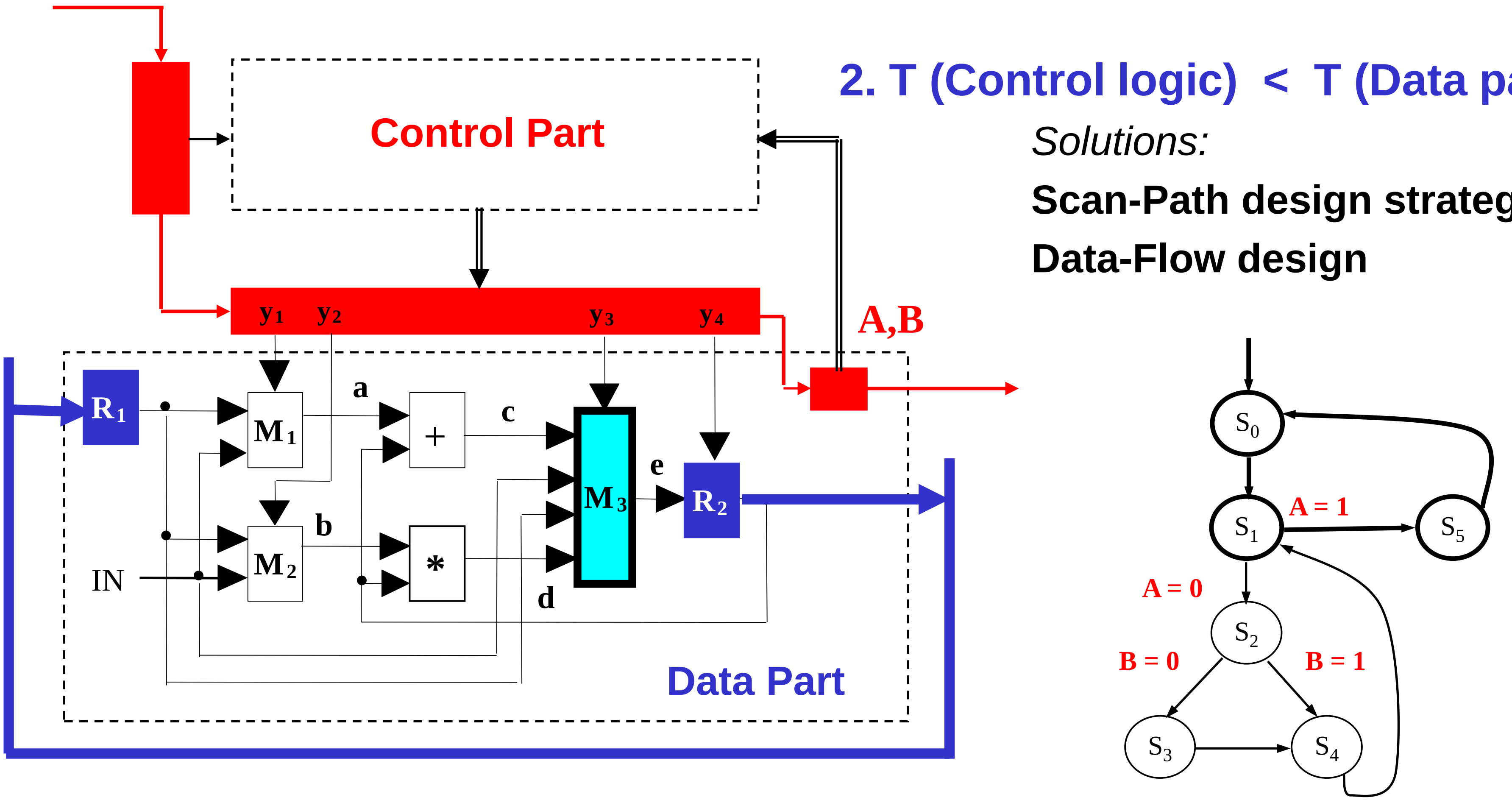
1. T (Sequential logic) < T (Combinational logic



Solution: Scan-Path design strategy



Testability of Design Types



2. T (Control logic) < T (Data path)

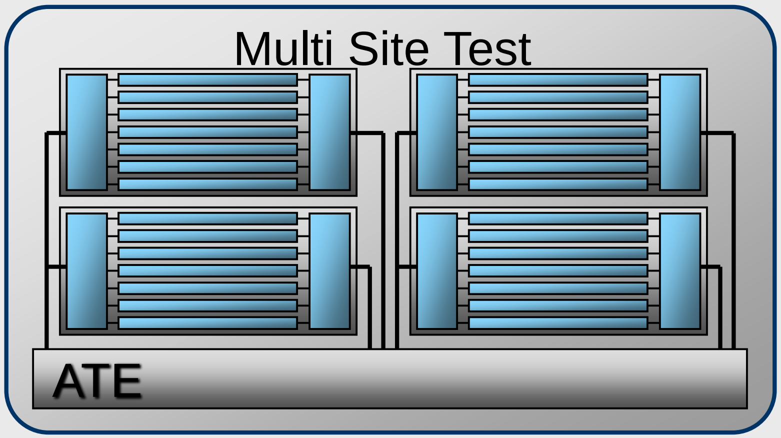
Solutions:

Scan-Path design strategie

Data-Flow design

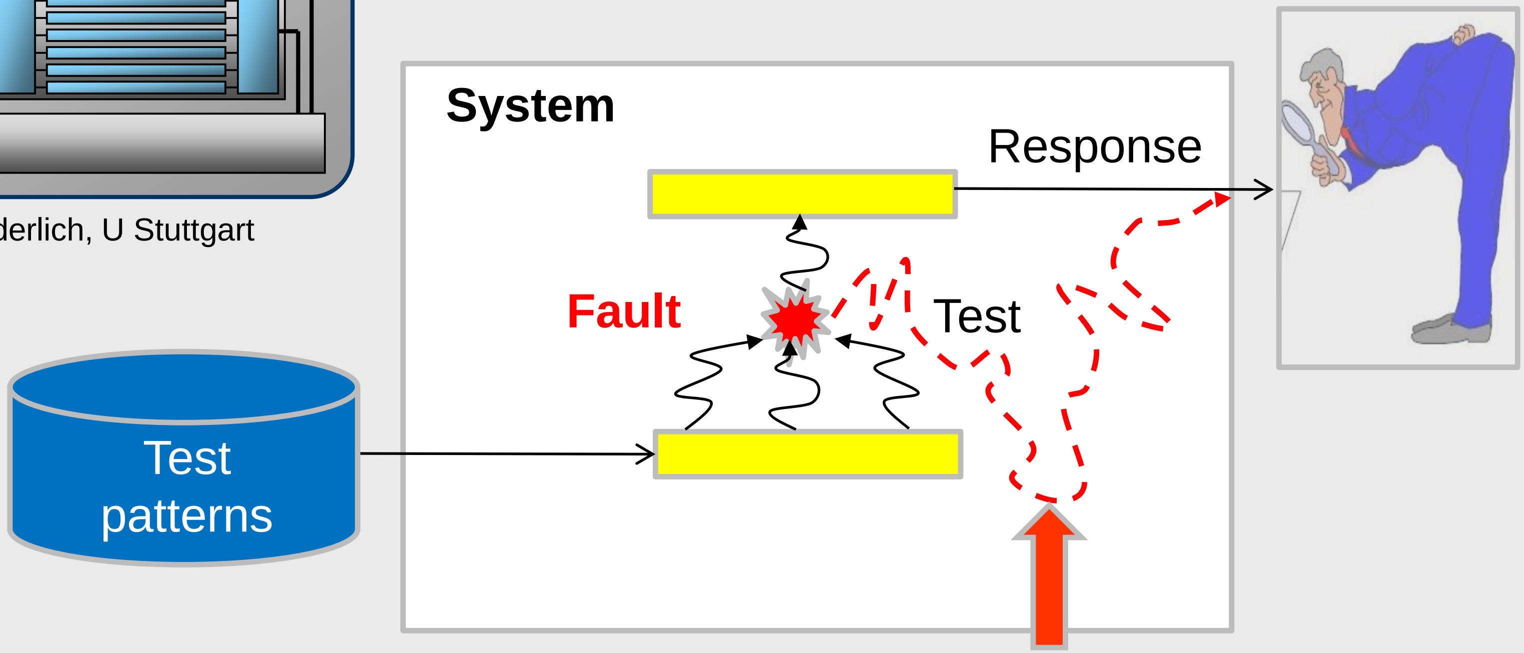
Scan-Path Based Testing

How to test million transistors?



H.-J.Wunderlich, U Stuttgart

All **memory components** are made **“transparent”** via shift registers



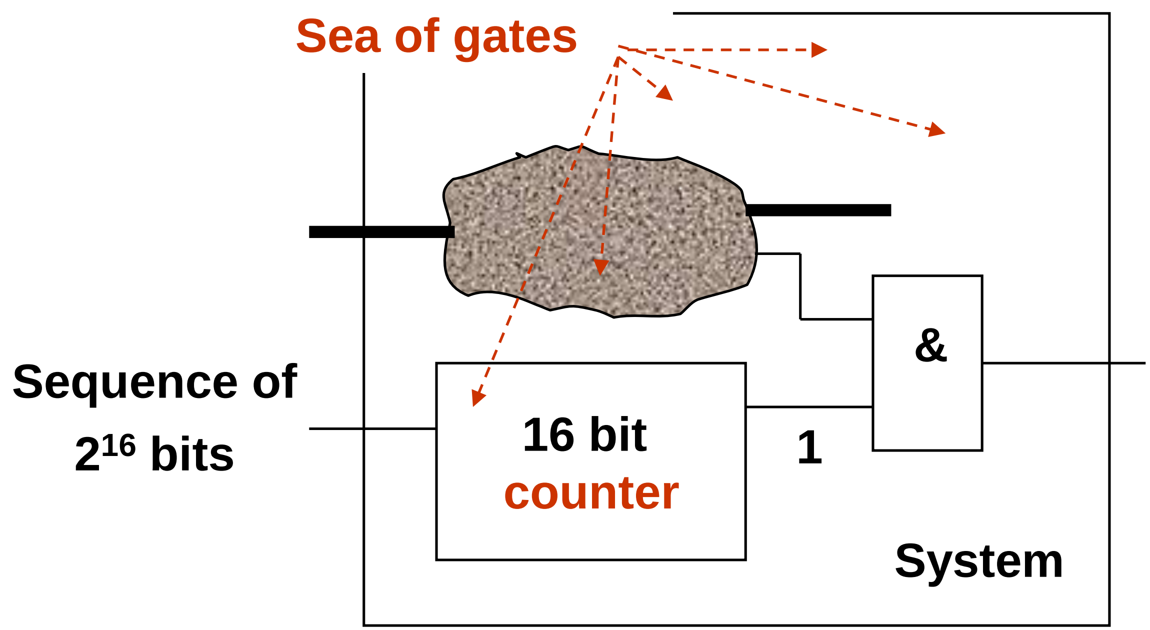
Signal path through millions transistors

Testability of Design Types

3. T (Random logic) < T (Structured logic)

Solutions: **Bus-oriented design, Core-oriented design**

Random logic, structure is hidden



Testability Estimation Rules of Thumb

Circuits less controllable

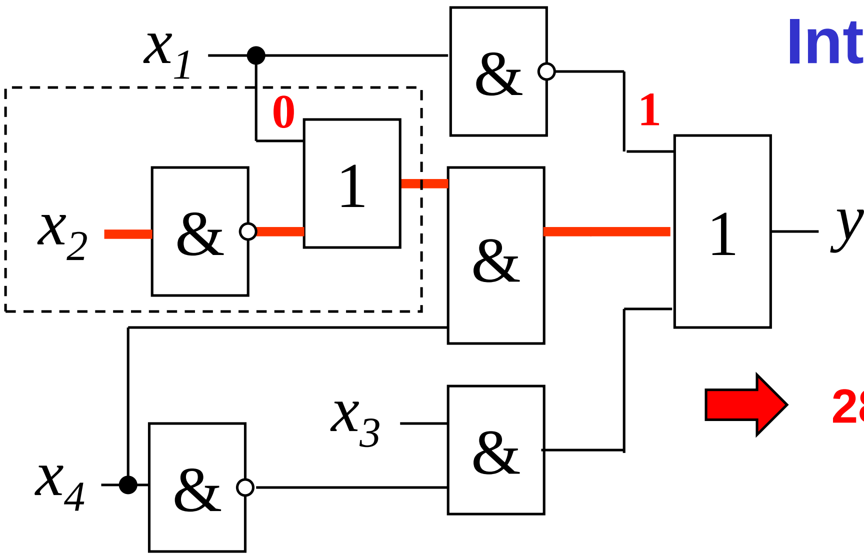
- Decoders
- Circuits with feedback
- Counters
- Clock generators
- Oscillators
- Self-timing circuits
- Self-resetting circuits

Circuits less observable

- Circuits with feedback
- Embedded
  - RAMs
  - ROMs
  - PLAs
- Error-checking circuits
- Circuits with redundant nodes

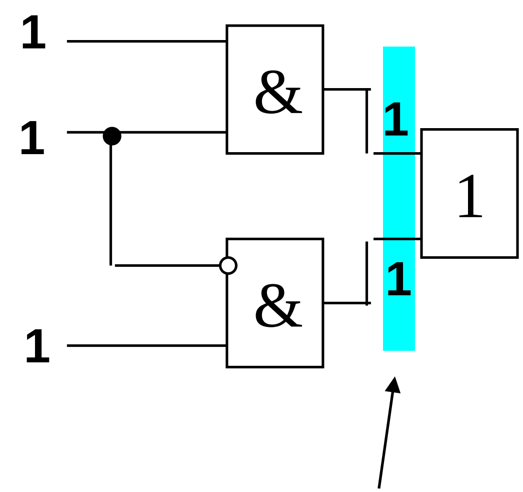
Bad Testability: Fault Redundancy

Redundant gates (bad design):



28 faults

Internal signal dependencies:



Impossible pattern,
OR → XOR is not testable

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

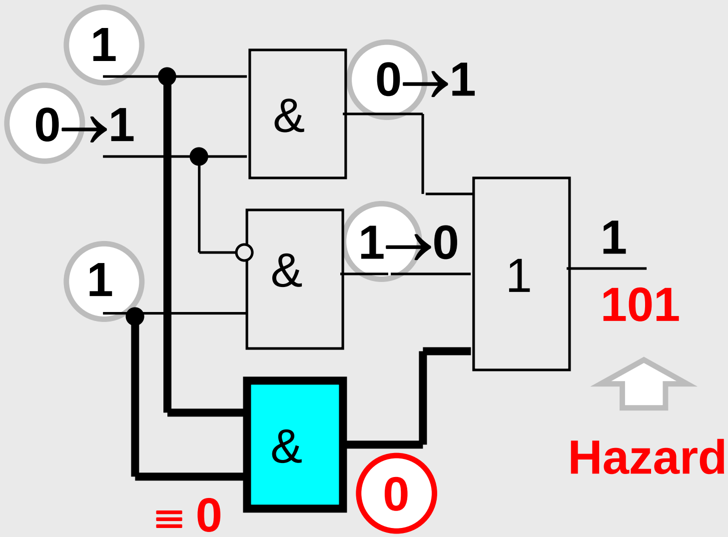
$$\frac{\partial y}{\partial x_2} \equiv 0$$

Faults at x_2 is not testable

Optimized function: $y = \overline{x_1} \vee x_4 \vee x_3$ → **6 faults (21%) !**

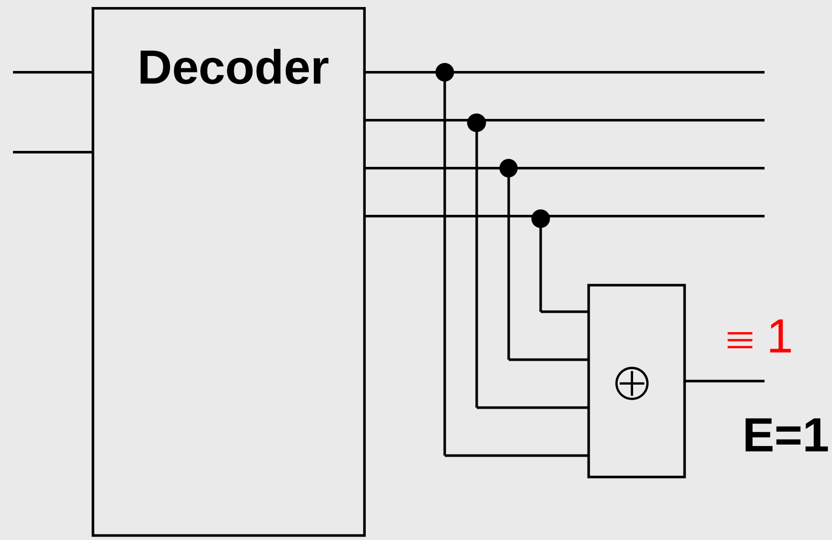
Fault Redundancy

Hazard control circuit:



Redundant AND-gate
Fault ≡ 0 is not testable

Error control circuitry

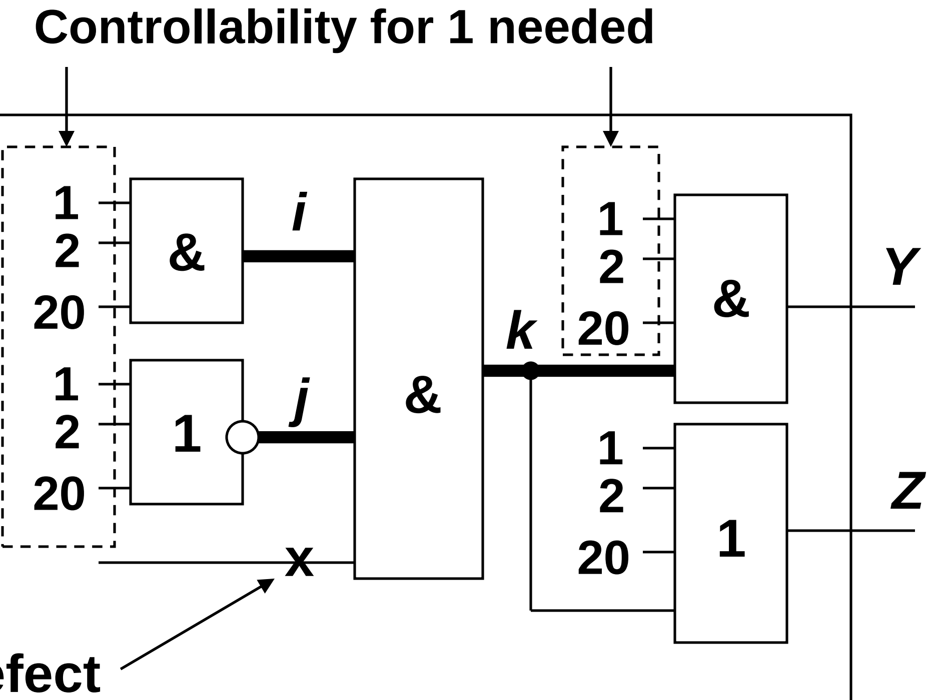


E = 1 if decoder is fault-free
Fault ≡ 1 is not testable

Hard to Test Faults

Evaluation of testability:

- Controllability
  - C_0 (*i*)
  - C_1 (*j*)
- Observability
  - O_Y (*k*)
  - O_Z (*k*)
- Testability



Probability of detecting $1/2^{20*20*20} = 1/2^{60}$

Consequeces of Bad Testability

- **Expected impact** of good testability
  - Reduces cost of deterministic test generation
  - Reduces cost of testing (time, memory space, length of test)
- **Redundant faults**
  - don’t need to be tested, because the functionality of the circuit remains correct
  - if you don’t know that the not-covered fault is redundant, the lower fault coverage will mean **ambiguiety** – under-estimating the test result
- **Hard-to-test faults**
  - cause reduction of the test quality in random testing
  - in deterministic testing the problem is solved at higher cost

Testability Analysis Methods

- ✓ **Fault simulation** – a very slow but **exact** method
- ✓ **Toggling** – a faster method, but only **approximate**
  - Logic simulation can be repeated a number of times with different data sets
  - Toggling on a node of a digital circuit means „to switch from 0 to 1, or from 1 to 0"
  - Circuit activity can be measured by **counting the toggles**
- ✓ Disadvantage:
  - Toggling is only a means to characterize the **controllability** of signals, but not the real testability
- ✓ **Calculation of testability measures**
  - Heuristic methods
  - Probabilistic methods

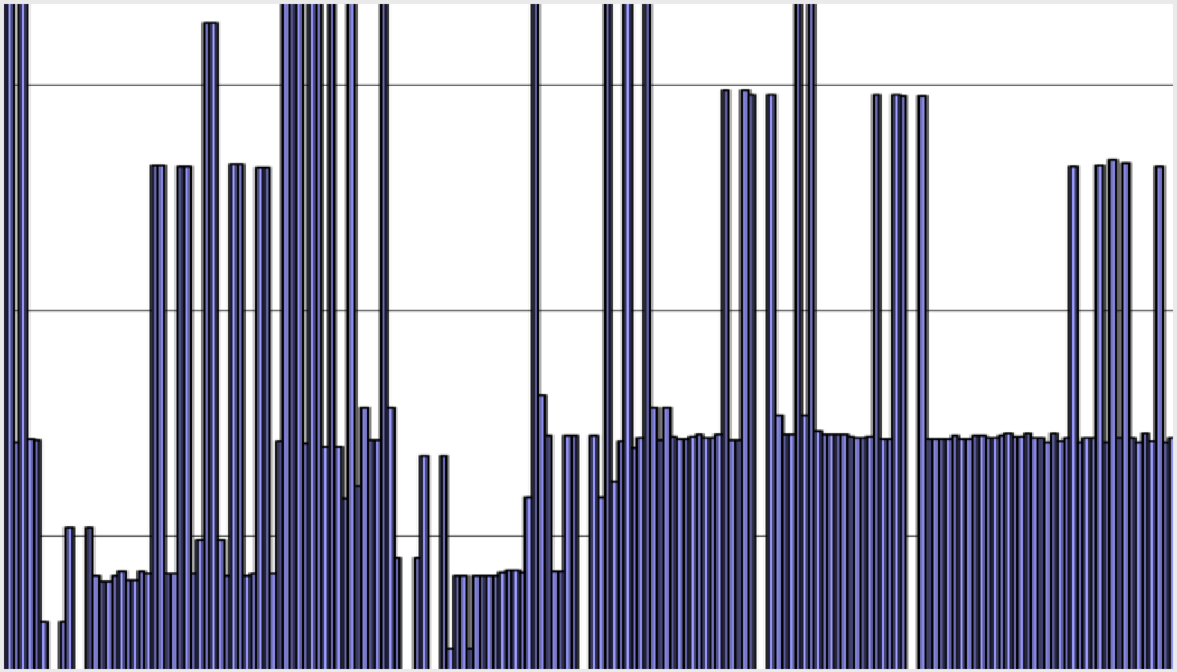
Testability Analysis with Fault Simulation

Fault table
As a result of
fault simulation

	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

Fault detectability spectrogramm

Fault detection frequency
as
testability
measure

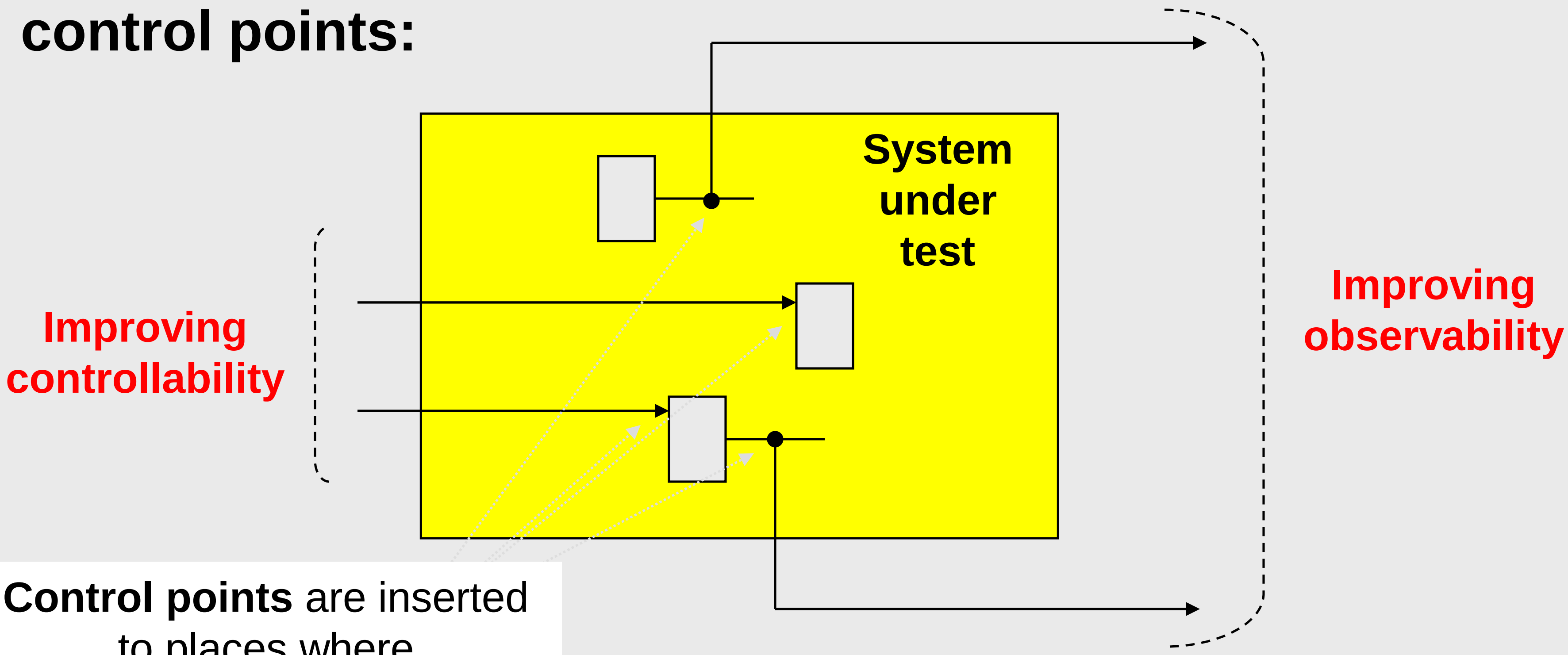


Different faults

The lower
the frequence,
the lower
the testability

DFT Using Control Points

To ways for improving testability with inserting of control points:



Control points are inserted to places where controllability or observability are low

To select control points, the testability **measures** are to be calculated

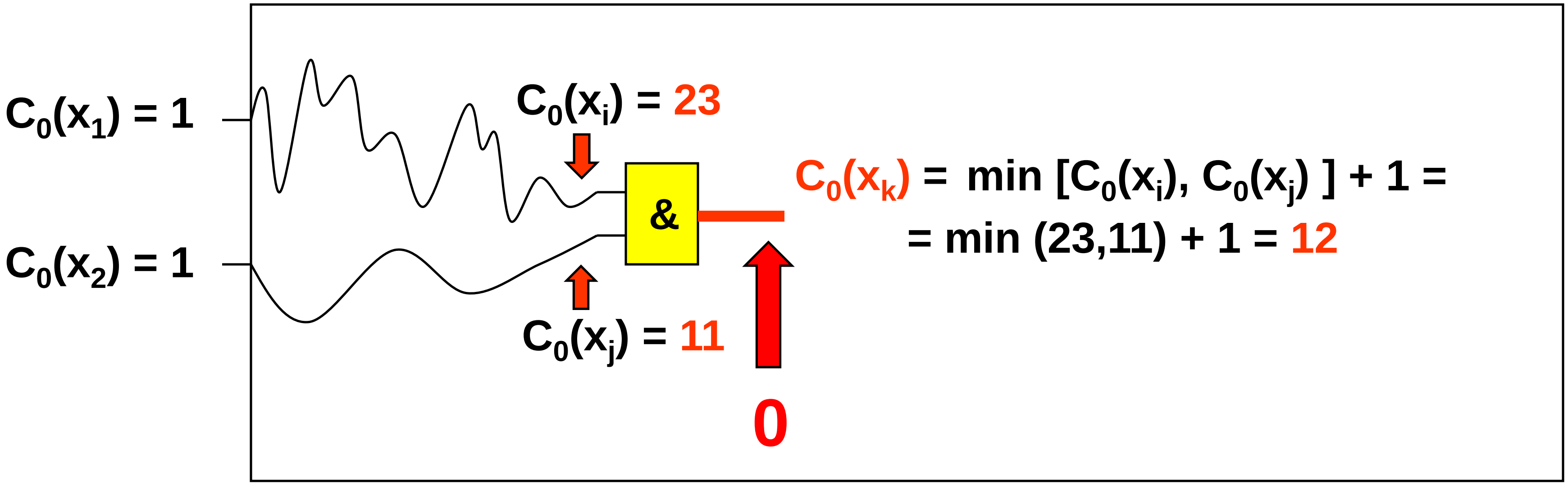
Heuristic Testability Measures

Controllability calculation: AND gate

Measure: minimum number of nodes that must be set to produce 0 or 1

For inputs: $C_0(x) = C_1(x) = 1$

For other signals: recursive calculation starting from inputs



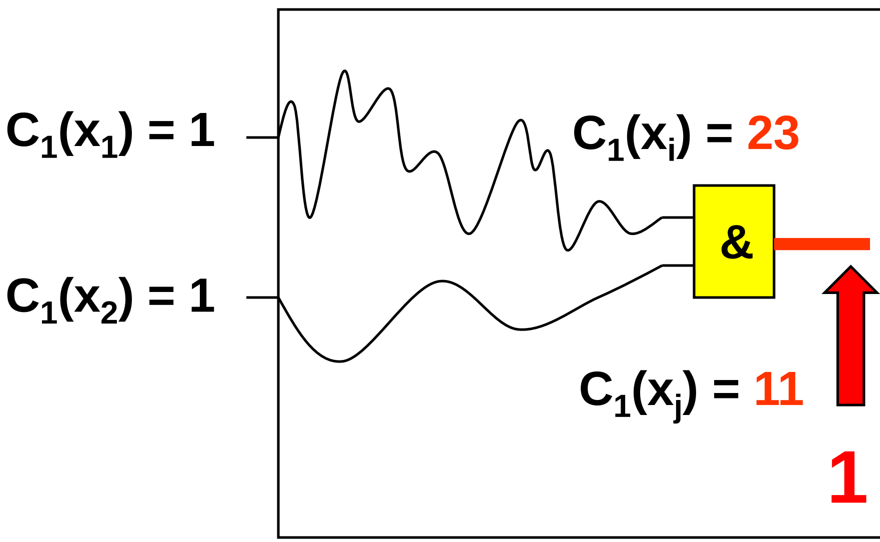
Heuristic Testability Measures

Controllability calculation: AND gate

Measure: minimum number of nodes that must be set to produce 1

For inputs: $C_0(x) = C_1(x) = 1$

For other signals: recursive calculation starting from inputs



$C_1(x_k) = C_1(x_i) + C_1(x_j) + 1 =$

$= 23 + 11 + 1 = 35$

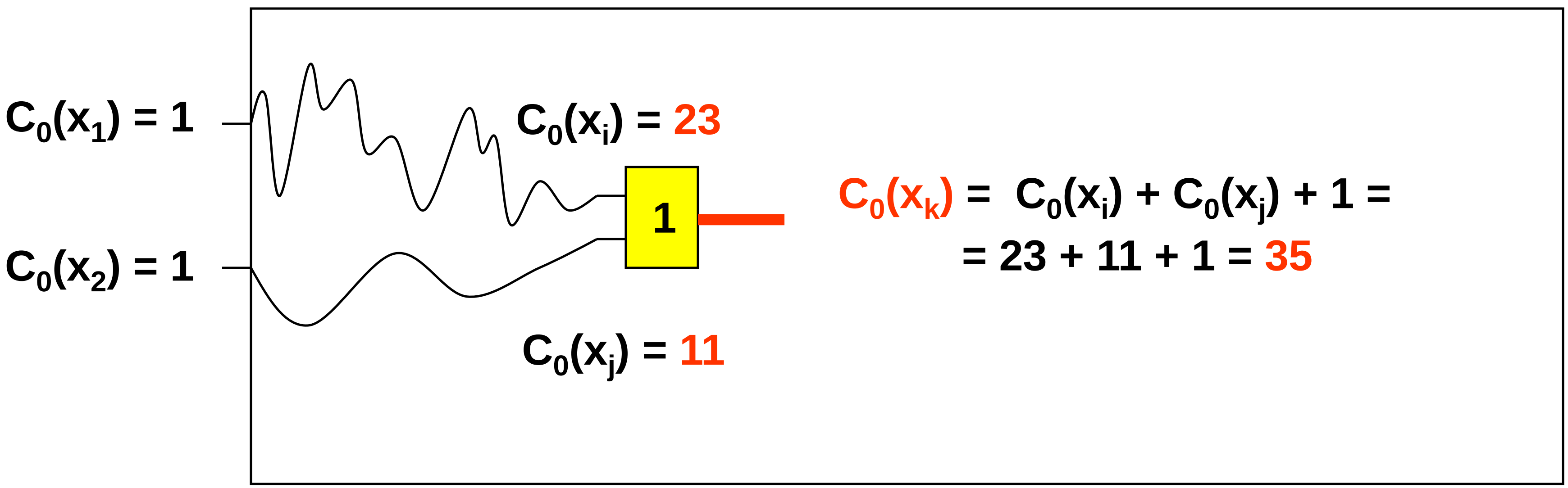
Heuristic Testability Measures

Controllability calculation: OR gate

Measure: minimum number of nodes that must be set to produce 0

For inputs: $C_0(x) = C_1(x) = 1$

For other signals: recursive calculation starting from inputs



Heuristic Testability Measures

Controllability calculation: EXOR gate

Measure: minimum number of nodes that must be set in order to produce 0

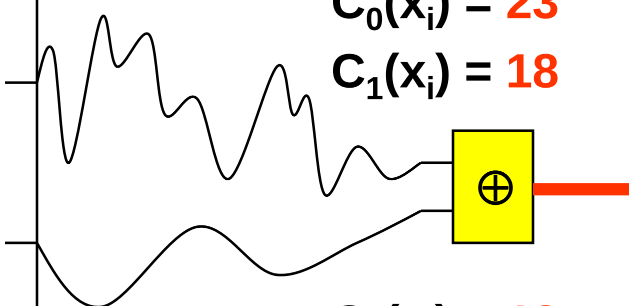
For inputs: $C_0(x) = C_1(x) = 1$

For other signals: recursive calculation starting from inputs

$C_0(x_i) = 23$

$C_1(x_i) = 18$

$C_0(x_1) = 1$



$C_0(x_2) = 1$

$C_0(x_j) = 12$

$C_1(x_j) = 20$

$$C_0(x_k) = \min \{ [C_0(x_i) + C_0(x_j)], [C_1(x_i) + C_1(x_j)] \} + 1 =$$

$$\min\{ (23 + 12), (18 + 20) \} + 1 =$$

$$\min (35,38) + 1 = 36$$

$$C_1(x_k) = \min (30,43) + 1 = 31$$

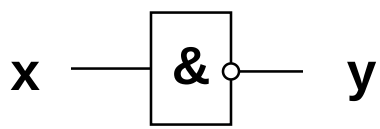
Heuristic Testability Measures

Controllability calculation:

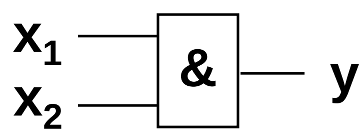
Measure: minimum number of nodes that must be set in order **to produce 0** or **1**

For inputs: $C_0(x) = C_1(x) = 1$

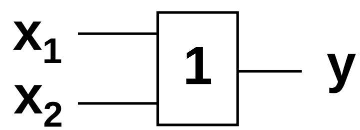
For other signals: recursive calculation rules:



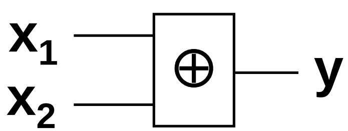
$$C_0(y) = C_1(x) + 1$$
$$C_1(y) = C_0(x) + 1$$



$$C_0(y) = \min\{C_0(x_1), C_0(x_2)\} + 1$$
$$C_1(y) = C_1(x_1) + C_1(x_2) + 1$$



$$C_1(y) = \min\{C_1(x_1), C_1(x_2)\} + 1$$
$$C_0(y) = C_0(x_1) + C_0(x_2) + 1$$



$$C_0(y) = \min\{(C_0(x_1) + C_0(x_2)), (C_1(x_1) + C_1(x_2))\} + 1$$
$$C_1(y) = \min\{(C_0(x_1) + C_1(x_2)), (C_1(x_1) + C_0(x_2))\} + 1$$

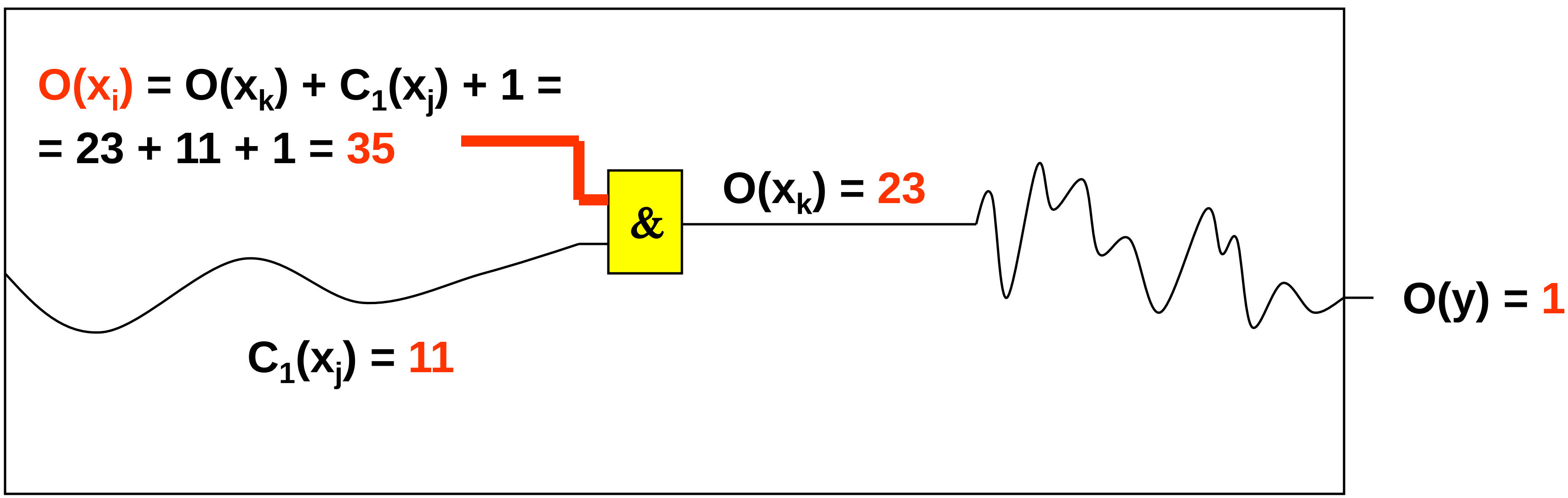
Heuristic Testability Measures

Observability calculation:

Measure: minimum number of nodes which must be set for fault propagating

For outputs: $O(y) = 1$

For other signals: recursive calculation starting from outputs



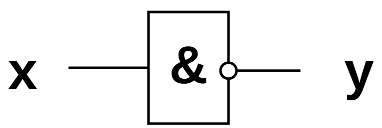
Heuristic Testability Measures

Observability calculation:

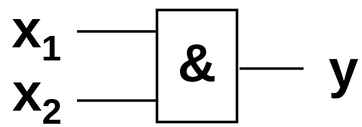
Measure: minimum number of nodes which must be set for fault propagating

For outputs: $O(y) = 1$

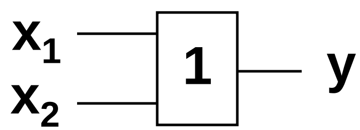
For other signals: recursive calculation rules:



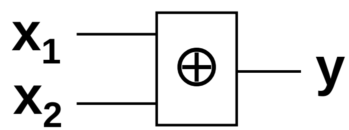
$$O(x) = O(y) + 1$$



$$O(x_1) = O(y) + C_1(x_2) + 1$$



$$O(x_1) = O(y) + C_0(x_2) + 1$$



$$O(x_1) = O(y) + 1$$

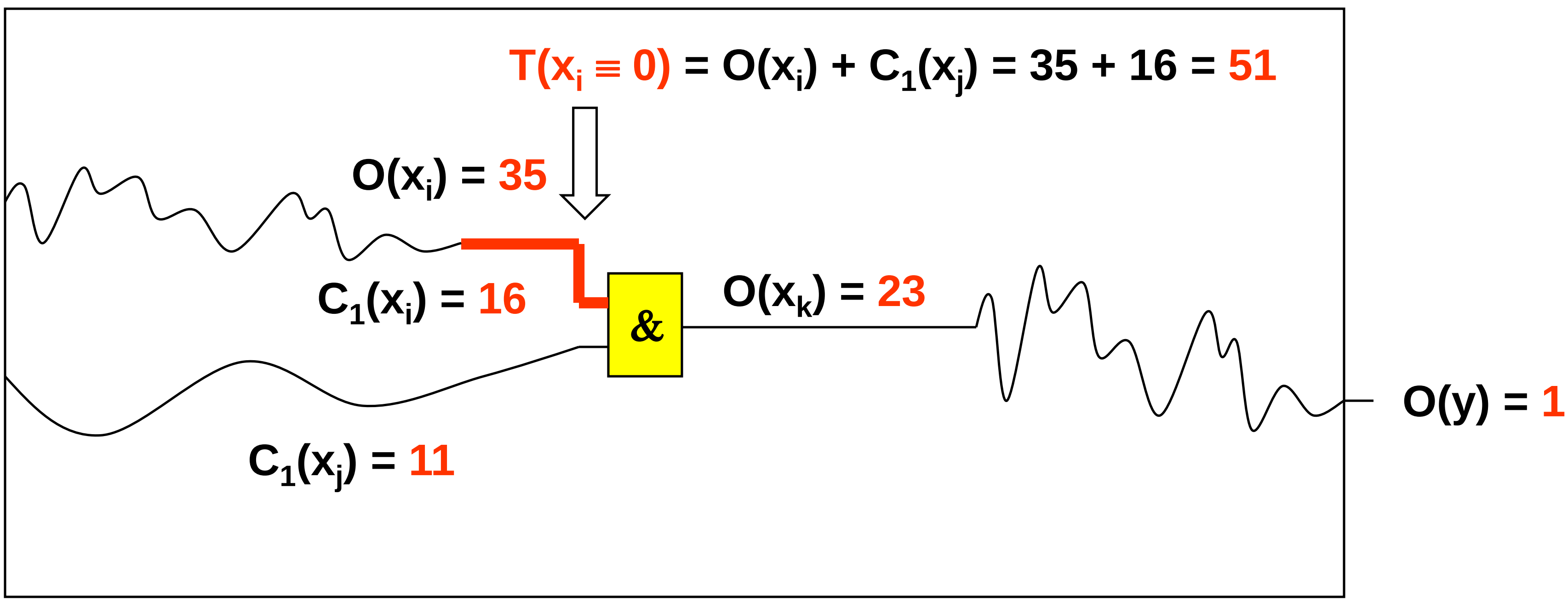
Heuristic Testability Measures

Testability calculation:

Measure: sum of controllability and observability

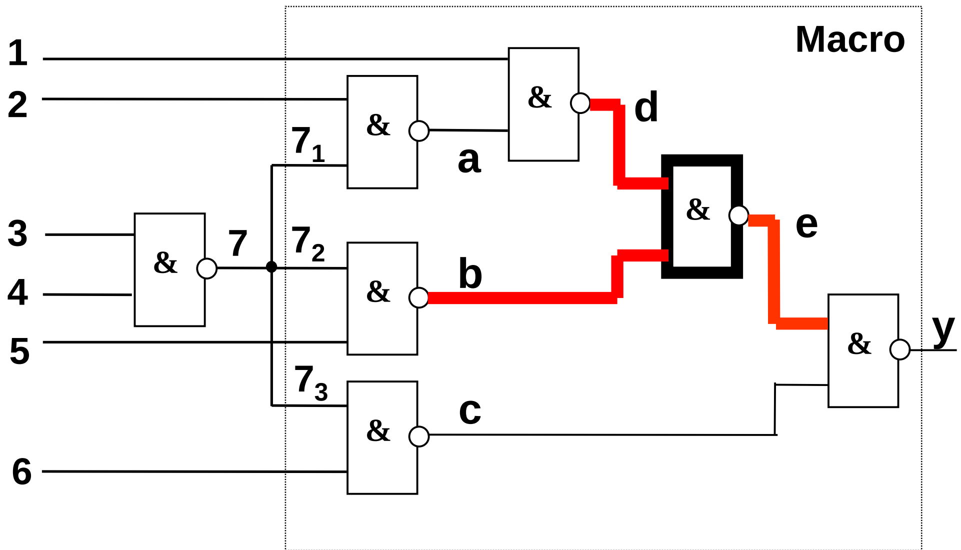
$$T(x \equiv 0) = C_1(x) + O(x)$$

$$T(x \equiv 1) = C_0(x) + O(x)$$



Heuristic Testability Measures

Controllability and observability:



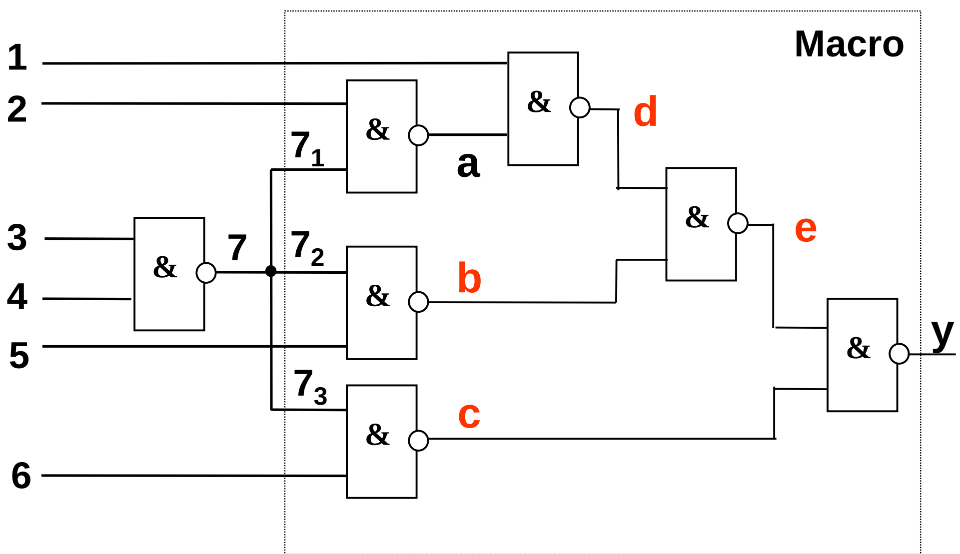
	Controllabilies		Obs.
x	$C_0(x)$	$C_1(x)$	$O(x)$
1	1	1	10
2	1	1	12
3	1	1	11
4	1	1	11
5	1	1	10
6	1	1	10
7	3	2	9
7_1	3	2	11
7_2	3	2	9
7_3	3	2	9
a	4	2	9
b	4	2	7
c	4	2	7
d	4	2	7
e	5	5	4
y	8	5	1

Heuristic Testability Measures

Testability calculation:

$$T(x \equiv 0) = C_1(x) + O(x)$$

$$T(x \equiv 1) = C_0(x) + O(x)$$



	Controllabilies		Obs.	Testab.
x	$C_0(x)$	$C_1(x)$	O(x)	$T(x\equiv 0)$
1	1	1	10	11
2	1	1	12	13
3	1	1	11	12
4	1	1	11	12
5	1	1	10	11
6	1	1	10	11
7	3	2	9	11
7_1	3	2	11	13
7_2	3	2	9	11
7_3	3	2	9	11
a	4	2	9	11
b	4	2	7	9
c	4	2	7	9
d	4	2	7	9
e	5	5	4	9
y	8	5	1	6

Why the testability of y is the lowest?

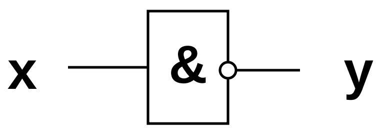
Probabilistic Testability Measures

Controllability calculation:

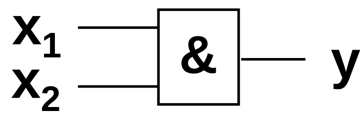
Measure: probability to produce 0 or 1 at the given nodes $p_{xi} = p(x_i=1) = 1$

For inputs: $C_0(i) = 1 - p_{xi}$ $C_1(i) = p_{xi}$

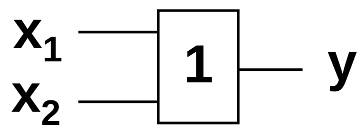
For other signals: recursive calculation rules:



$p_y = 1 - p_x$

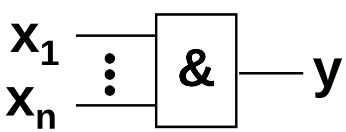


$p_y = p_{x1}\, p_{x2}$

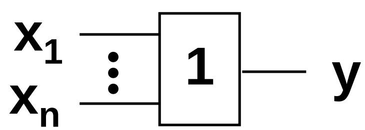


$p_y = 1 - (1 - p_{x1})(1 - p_{x2})$

$p_y = p_{x1} + p_{x2} - p_{x1}\, p_{x2})$



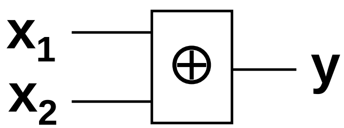
$$p_y = \prod_{i=1}^{n} p_{xi}$$



$$p_y = 1 - \prod_{i=1}^{n} (1 - p_{xi})$$

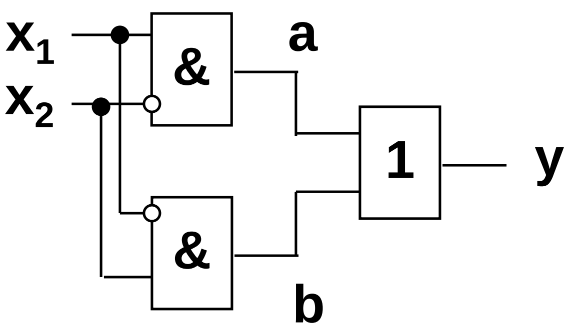
Paradoxes of Probabilistic Measures

Probabilities of reconverging fanouts:



$$p_y = (1 - p_{x1})\, p_{x2} + (1 - p_{x2})\, p_{x1}$$

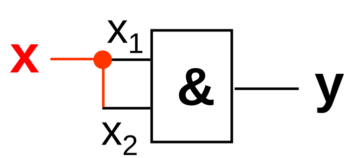
$$= 0{,}25 + 0{,}25 = 0{,}5$$



$$p_y = 1 - (1 - p_a)\,(1 - p_b)$$

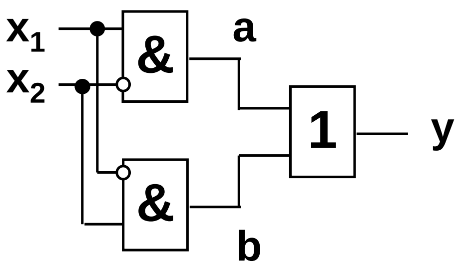
$$= 1 - 0{,}75 * 0{,}75 = 0{,}44$$

Signal correlations:

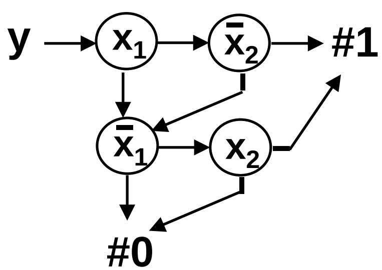


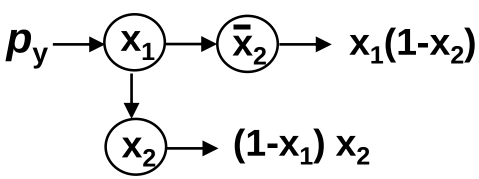
$$p_y = p_{x1}\, p_{x1} = p_x^2 = p_x$$

Calculation of Signal Probabilities



SSBDD based algorithm:





$p_y = x_1(1-x_2) + (1-x_1)\,x_2 = 0{,}5$

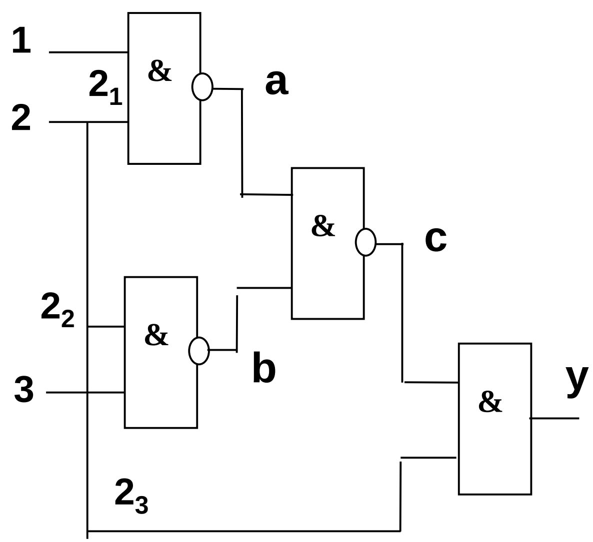
Parker - McCluskey algorithm:

$$
\begin{aligned}
p_y &= 1 - (1 - p_a)(1 - p_b) = \\
&= 1 - (1 - p_{x1}(1 - p_{x2}))(1 - p_{x2}(1 - p_{x1})) = \\
&= 1 - (1 - p_{x1} + p_{x1}p_{x2})(1 - p_{x2} + p_{x1}p_{x2}) = \\
&= 1 - (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p^2_{x1}p_{x2} + \\
&\qquad + p_{x1}p_{x2} - p_{x1}p^2_{x2} + p^2_{x1}p^2_{x2}) = \\
&= 1 - (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p_{x1}p_{x2} + \\
&\qquad + p_{x1}p_{x2} - p_{x1}p_{x2} + p_{x1}p_{x2}) = \\
&= p_{x2} - p_{x1}p_{x2} + p_{x1} - p_{x1}p_{x2} + p_{x1}p_{x2} - \\
&\qquad - p_{x1}p_{x2} + p_{x1}p_{x2} - p_{x1}p_{x2}) = \\
&= p_{x1} + p_{x2} - 2p_{x1}p_{x2} = 0{,}5
\end{aligned}
$$

The exponents are removed

Calculation of Signal Probabilities

Two methods: (1) from INP to OUT, (2) from OUT to INP



For all inputs: $p_k = 1/2$

Fast calculation gate by gate:

$p_a = 1 – p_1p_2 = 0{,}75,$

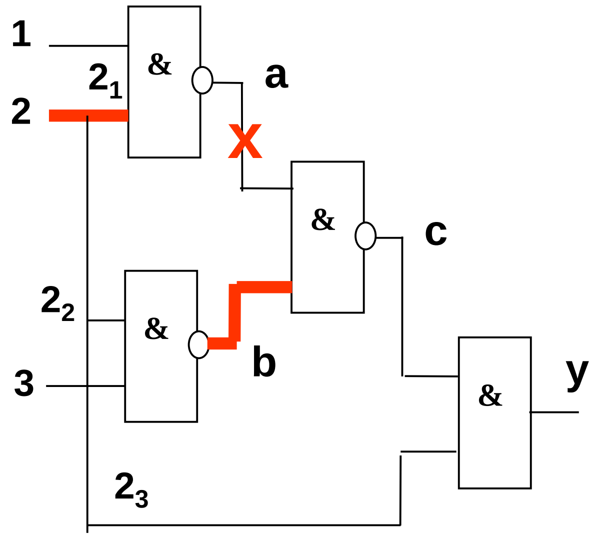
$p_b = 0{,}75, \quad p_c = 0{,}43, \quad p_y = 0{,}22$

Parker - McCluskey algorithm:

$$p_y = p_cp_2 = (1- p_ap_b)\, p_2 =$$
$$= (1 – (1- p_1p_2)\,(1- p_2p_3))\, p_2 =$$
$$= p_1p_2^{\,2} + p_2^{\,2}p_3 - p_1p_2^{\,3}p_3 =$$
$$= p_1p_2 + p_2\, p_3 - p_1p_2p_3 = 0{,}38$$

Probabilistic Testability Measures

Parker-McCluskey:



For all inputs: $p_k = 1/2$

Observability:

$$p(\partial y/\partial a = 1) = p_b\, p_2 =$$

$$= (1 - p_2 p_3)\, p_2 = p_2 - p_2^2 p_3$$

$$= p_2 - p_2 p_3 = 0,25$$

Testability:

$$p(a \equiv 1) = p(\partial y/\partial a = 1)\,(1 - p_a) =$$

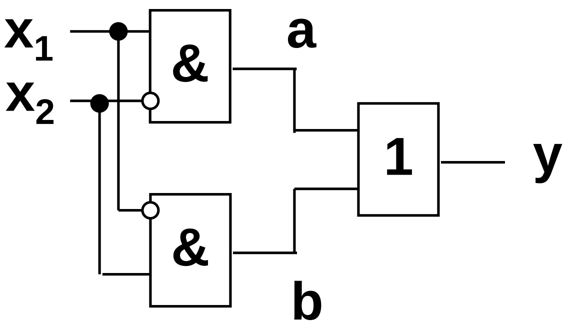
$$= (p_2 - p_2 p_3)(p_1 p_2) =$$

$$= p_1 p_2^2 - p_1 p_2^2 p_3 =$$

$$= p_1 p_2 - p_1 p_2 p_3 = 0,125$$

Calculation of Signal Probabilities

Parker - McCluskey algorithm:



$$p_y = 1 - (1 - p_a)(1 - p_b) =$$

$$= 1 - (1 - p_{x1}(1 - p_{x2}))(1 - p_{x2}(1 - p_{x1})) =$$

$$= 1 - (1 - p_{x1} + p_{x1}p_{x2})(1 - p_{x2} + p_{x1}p_{x2}) =$$

$$= 1 – (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p^2_{x1}p_{x2} +$$

$$+ p_{x1}p_{x2} - p_{x1}p^2_{x2} + p^2_{x1}p^2_{x2}) =$$

$$= 1 – (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p_{x1}p_{x2} +$$

$$+ p_{x1}p_{x2} - p_{x1}p_{x2} + p_{x1}p_{x2}) =$$

$$= p_{x2} - p_{x1}p_{x2} + p_{x1} - p_{x1}p_{x2} + p_{x1}p_{x2} -$$

$$- p_{x1}p_{x2} + p_{x1}p_{x2} - p_{x1}p_{x2}) =$$

$$= p_{x1} + p_{x2} - 2p_{x1}p_{x2} = 0,5$$

Conclusions:

The P-McC method has a high complexity

In tree-like circuits, the gate-by-gate method works accurately

Calculation of Signal Probabilities

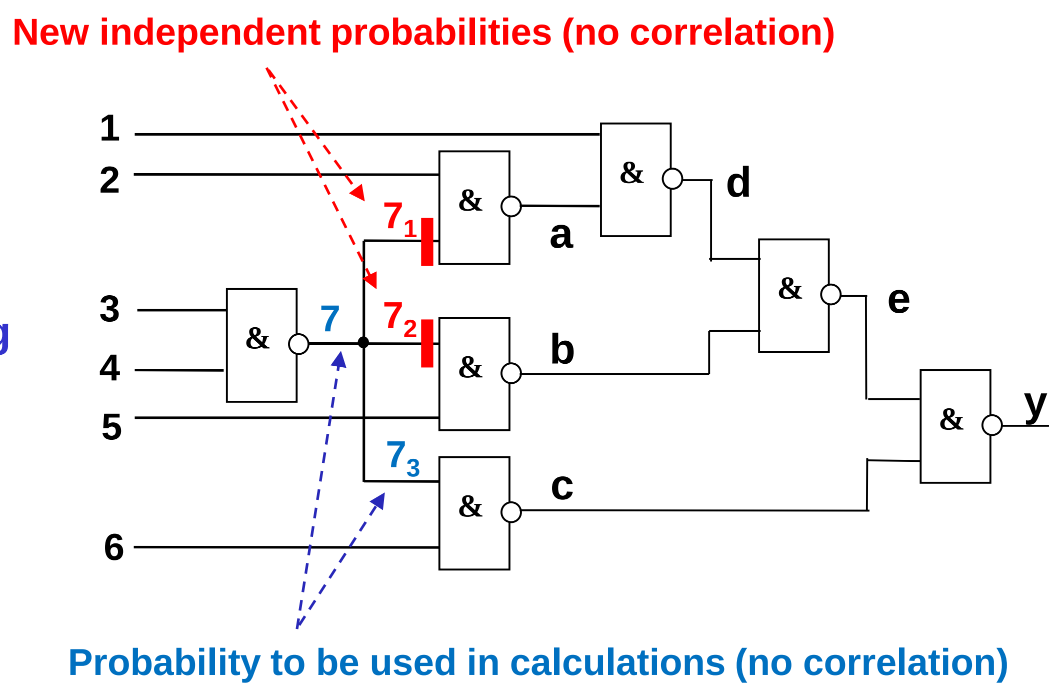
Cutting method

Idea:

- The complexity of exact calculation is reduced by using lower and higher bounds of probabilities

Technique:

- Reconvergent fan-outs are cut except of one
- Probability range of [0,1] is assigned to all the cut lines
- The bounds are propagated by straightforward calculation

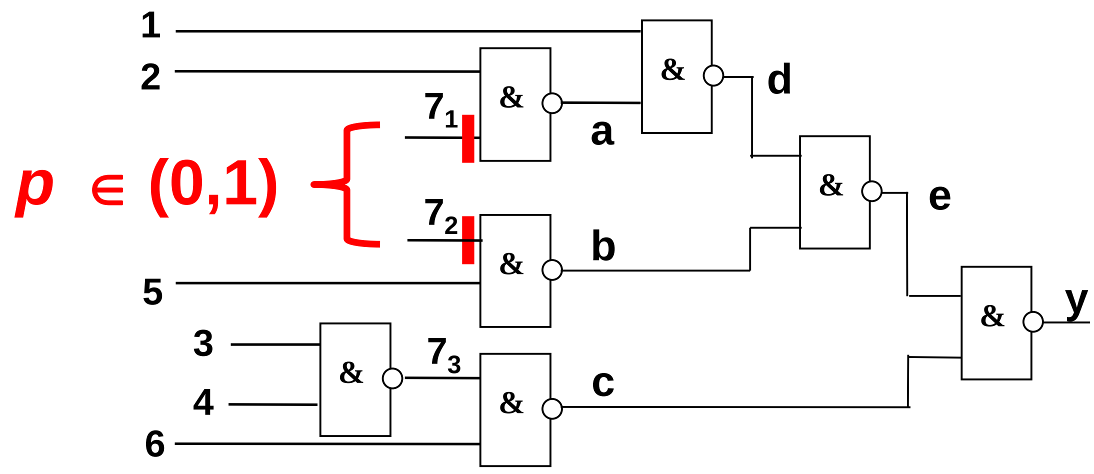


Lower and higher bounds for the probabilities of the cut lines:

$$p_{71} := (0;1),\ \ p_{72} := (0;1),\ p_{73} := 0{,}75$$

Calculation of Signal Probabilities

Cutting method



Technique:

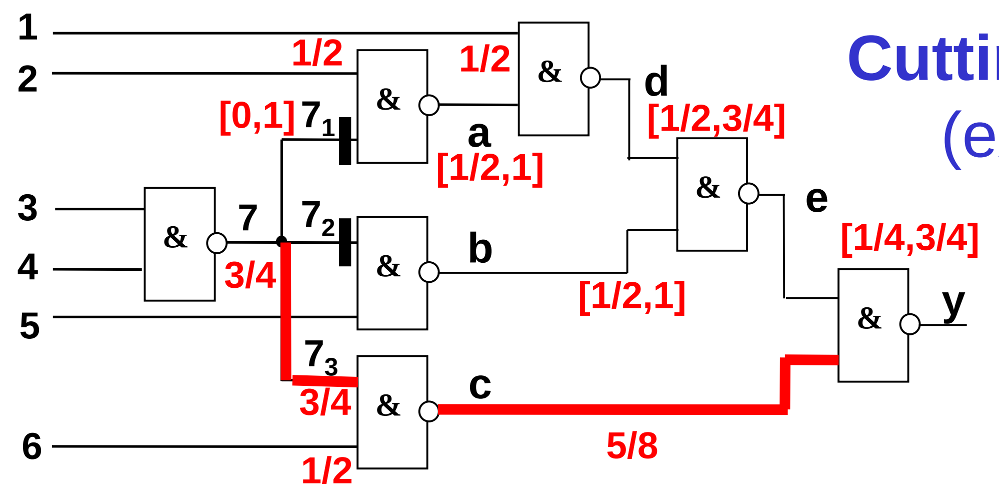
- Reconvergent fan-outs are cut except of one
- Probability range of [0,1] is assigned to all the cut lines
- The new probabilities are propagated by straightforward calculation

Lower and higher bounds for the probabilities of the cut lines:

p_{71} := (0;1), p_{72} := (0;1), p_{73} := 0,75

Calculation of Signal Probabilities

- For all inputs: $p_k = 0{,}5$
- Reconvergent fan-outs are cut except of one – 7_1 and 7_2
- Probability range of [0,1] is assigned to all the cut lines - 7_1 and 7_2
- The bounds are propagated by straightforward calculation



Cutting method (example)

[34/64 - 54/64]
[0.5 - 0.8]

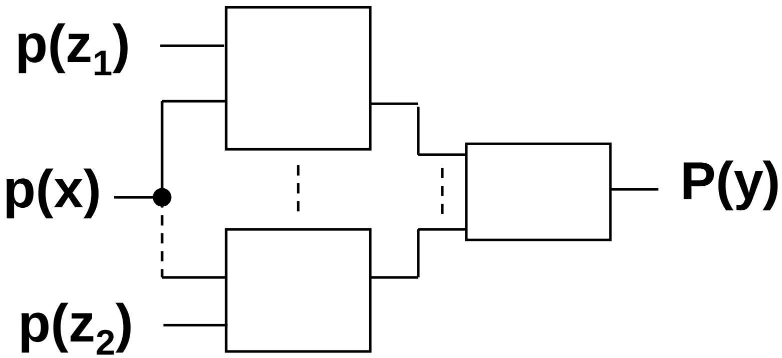
Exact value:
41/64 = **0.6**

Calculation steps:

p_k	$[p_{LB}, p_{HB})$	Exact p_k	p_k	$[p_{LB}, p_{HB})$	Exact p_k
p_7	3/4	3/4	p_b	[1/2, 1]	5/8
p_{71}	[0, 1]	3/4	p_c	5/8	5/8
p_{72}	[0, 1]	3/4	p_d	[1/2, 3/4]	11/16
p_{73}	3/4	3/4	p_e	[1/4, 3/4]	19/32
p_a	[1/2, 1]	5/8	p_y	[34/64, 54/64]	41/64

Calculation of Signal Probabilities

Method of conditional probabilities



$P(y) = p(y/x{=}0)\ p(x{=}0) + p(y/x{=}1)\ p(x{=}1)$

Probabilitiy for – *y*

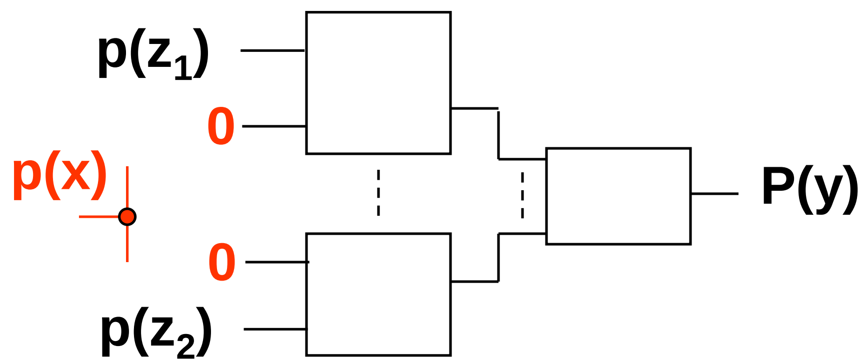
Conditions – *x* ∈ set of conditions

$$p(y) = \sum_{i \in (0,1)} p(y/(x = i)\, p(x = i)$$

Conditional probabilitiy

Calculation of Signal Probabilities

Method of conditional probabilities



Probabilitiy for – *y*

Conditions – *x* ∈ set of conditions

$$p(y) = \sum_{i \in (0,1)} p(y/(x=i)\,p(x=i)$$

Conditional probabilitiy

P(y) = p(y/x=0) p(x=0) + p(y/x=1) p(x=1)

Idea of the method:

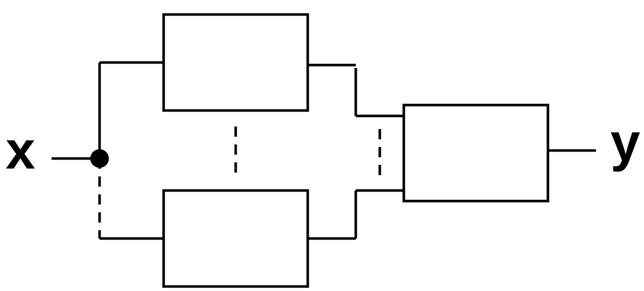
Two conditional probabilities are calculated along the paths (NB! not bounds as in the case of the cutting method)

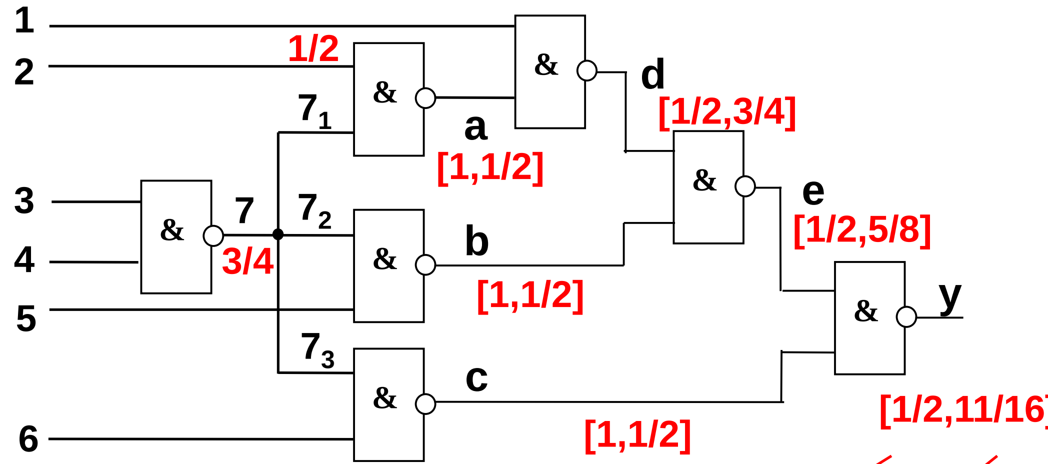
Since no reconvergent fanouts are on the paths, no danger for signal correlations

Calculation of Signal Probabilities

Method of conditional probabilities

$$p(y) = \sum_{i\in(0,1)} p(y/(x=i)\,p(x=i)$$





NB! Conditional probabilities

$P_k \sim [P_k^0 = p(x_k/x_7=0),\ P_k^1 = p(x_k/x_7=1)]$

are propagated, not bounds

as in the cutting method.

For all inputs: $p_k = 1/2$

$$p_y = p(y/x_7=0)(1 - p_7) + p(y/x_7=1)p_7 = (1/2 \times 1/4) + (11/16 \times 3/4) = 41/64$$

Random Pattern Test Length

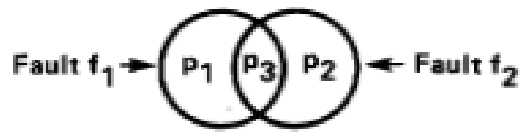
Terminology

- **Detection probabilities**:
  - n-th step detection probability of a fault
  - n-th step detection probability of a fault set
- **Worst fault** - that fault having the lowest detection probablity of any fault in the Network (independently of probabilities for other faults)
- **Escape Probability of a Fault Set:** the probability that at least one member of the fault set will not be detected by the random pattern test
  - For an *n*-step test, the escape probability of a fault set is denoted e_n
- The *escape probability* and the *detection probability* sum to one
- **Escape probability** is the **quality measure** of the random pattern test

Higher escape probabilities will require *longer* random pattern tests

Random Pattern Test Length

The faults are said to have **conjoint** test sets if their corresponding test sets share at least one vector



A Venn diagram showing the detection probabilities of two faults.

p_1 – is the probability that a randomly selected vector will detect fault f_1 but not f_2

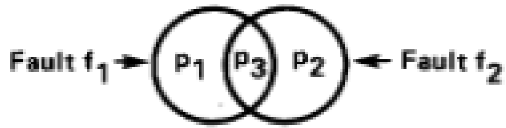
p_2 – is the probability that a randomly selected vector will detect fault f_2 but not f_1

p_3 – is the probability that the vector will detect both faults

The detection probability of fault f_1 is $p_1 + p_3$, and that of fault f_2 is $p_2 + p_3$

If $p_3 = 0$, then the corresponding test sets are **disjoint**

Random Pattern Test Length



A Venn diagram showing the detection probabilities of two faults.

The **escape probability** of the fault-set for the **disjoint case** is given by

$$e_n = (1 - p_1)^n + (1 - p_2)^n - [1 - (p_1 + p_2)]^n.$$

In order to compute the minimum random test length ***n*** that detects the fault set with escape probability **no larger than** threshold e_t, it is necessary to compute n that satisfies

$$e_n < e_t$$

Random Pattern Test Length

The random test length due to ***k*** faults with disjoint test sets, each with detection probability ***p*** is bounded by

$$N_{kD}^{U} = \left\lceil \frac{\ln(e_t/k)}{\ln(1-p)} \right\rceil$$

$$N_{kD}^{L} = \left\lceil \frac{\ln(e_t/k) - \ln(1 - e_t/2)}{\ln(1-p)} \right\rceil$$

For practical values of ***p*** and e_t (values much less than one) the formula can be approximated by

$$N_{kD}^{U} \cong \left\lceil \frac{\ln k - \ln e_t}{p} \right\rceil$$

For the case where $e_t = 10^{-3}$ (the confidence 99,9%

$$N_{kD}^{U} \cong \left\lceil \frac{6.9 + \ln k}{p} \right\rceil$$

Example: Random pattern test **11/*p*** can detect as many as **k = 50** hard faults, each having a detection probability with a **conficence of 99,9 %**

Literature: Jacob Savir, Paul H. Bardell „On random pattern test length". IEEE Trans. on Comp., 1984

Random Pattern Test Length

Examples

for the case of confidence of

e_t = 99,9%

$$N_{kD}^{U} \cong \left\lceil \frac{\ln k - \ln e_t}{p} \right\rceil$$

Test length as the function of the size of the set of hard faults ***k*** and for practical values of ***p* < 1**

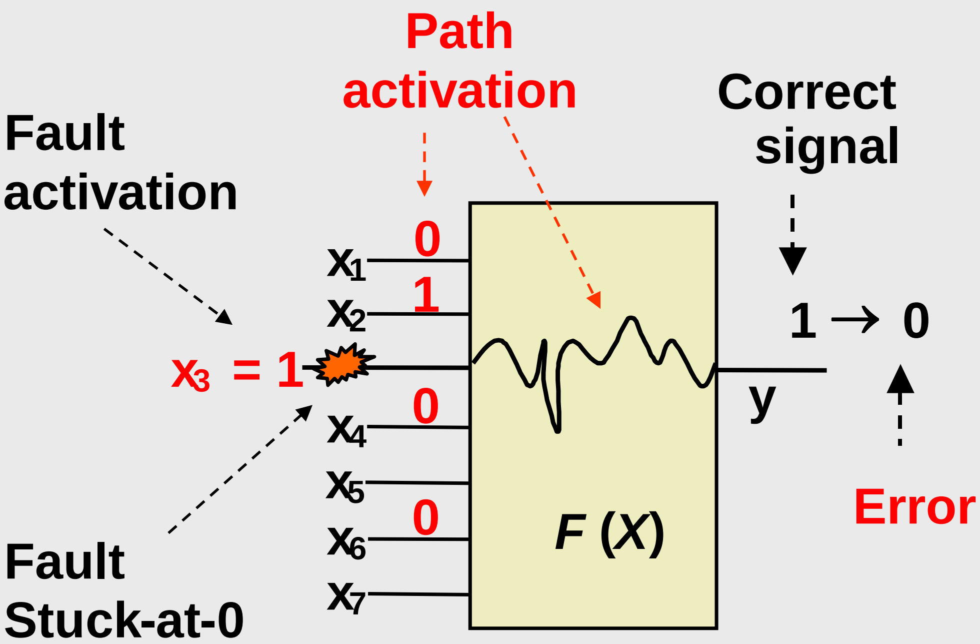
k	ln k	ln k - ln e	(Lnk-Lne)/p
1	0	7	690776
10	2	9	921034
20	3	10	990349
30	3	10	1030895

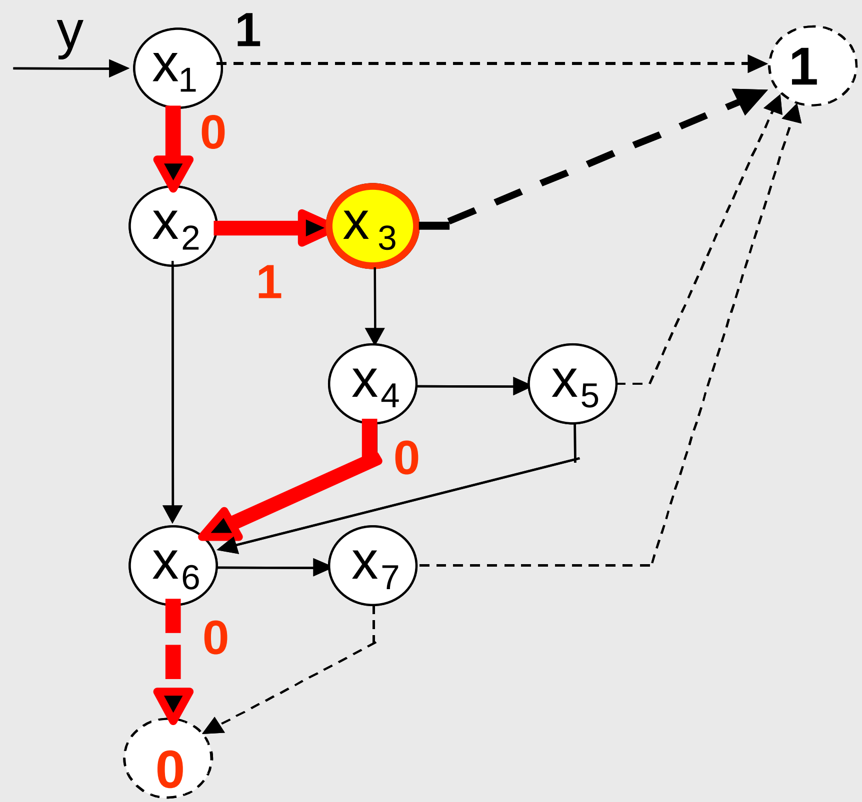
Test length as the function of the fault detection probability ***p*** for the size of the set of hard faults ***k* = 1**

k	ln k	p	lne/ln(1-p)
1	0	0,01	687
1	0	0,001	6904
1	0	0,0001	69074
1	0	0,00001	690772

BDDs and Testing of Logic Circuits

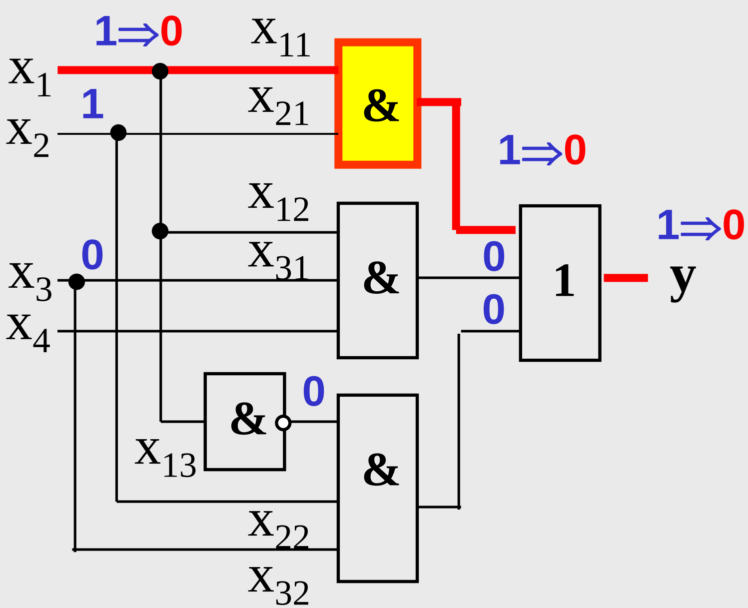
$$y = x_1 \vee x_2(x_3 \vee x_4 x_5) \vee x_6 x_7$$



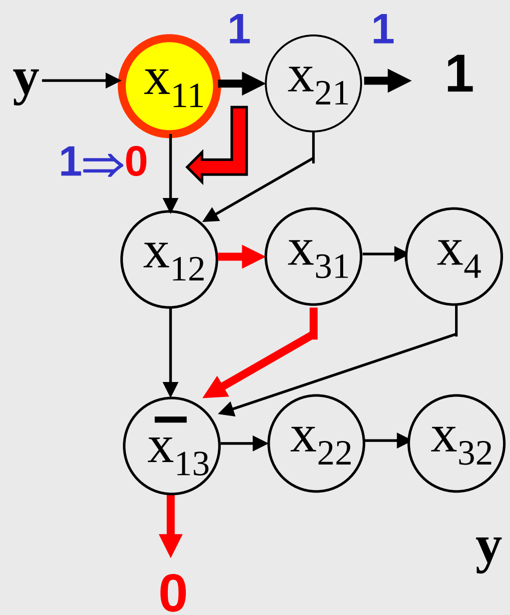


Two types of BDDs

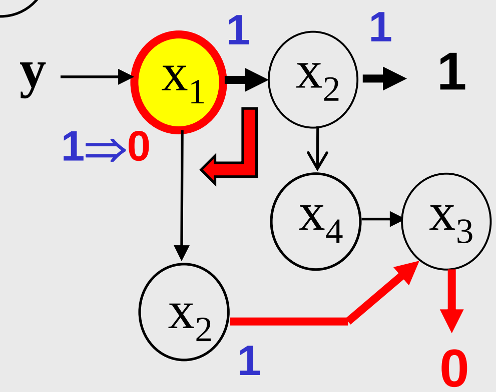
Test generation for: $x_{11} \equiv 0$



Structural BDD:



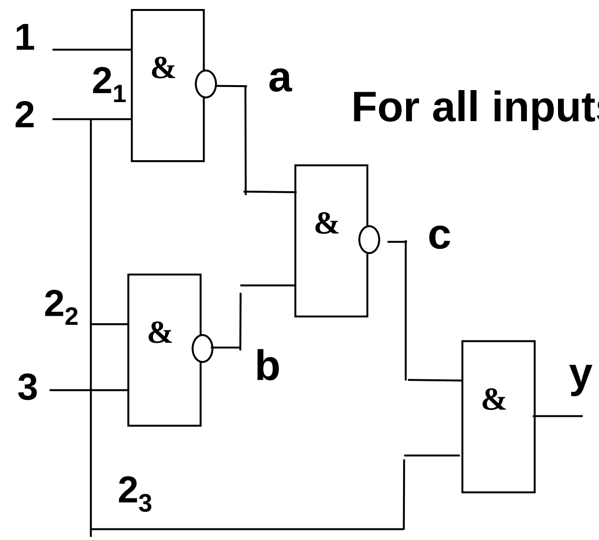
Functional BDD:



Test pattern:

x_1	x_2	x_3	x_4	y
1	1	0	-	$1 \Rightarrow 0$

Calculation of Signal Probabilities with BDDs



For all inputs: $p_k = 1/2$

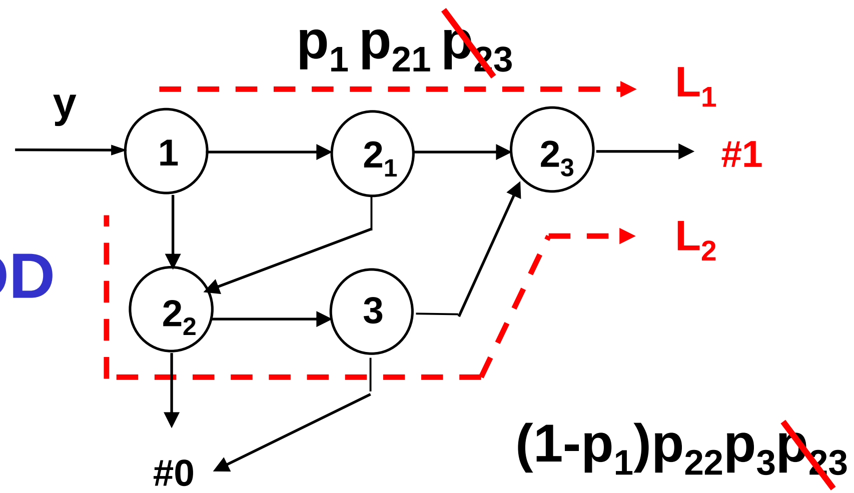
Using BDDs:

$$p_y = p(L_1) + p(L_2) =$$

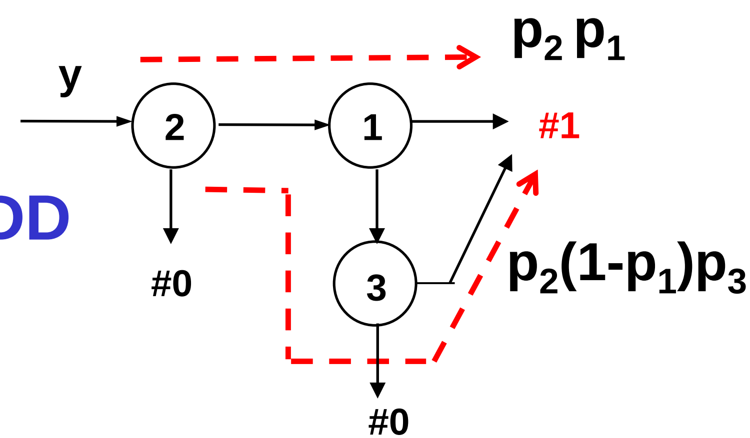
$$= p_1\, p_{21}\, \cancel{p_{23}} + (1 - p_1)\, p_{22}\, p_3\, \cancel{p_{23}} =$$

$$= p_1\, p_2 + (1 - p_1) p_2\, p_3 = 0{,}38$$

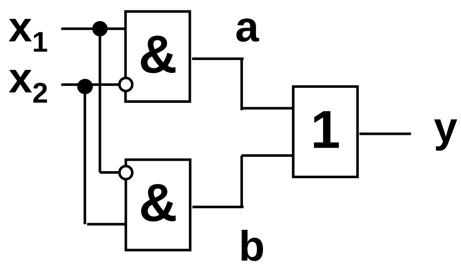
SSBDD



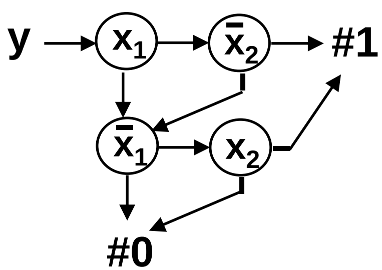
FBDD

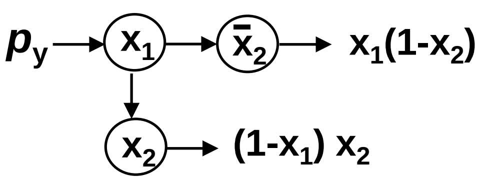


Calculation of Signal Probabilities



SSBDD based algorithm:





$p_y = x_1(1-x_2) + (1-x_1)\,x_2 = 0{,}5$

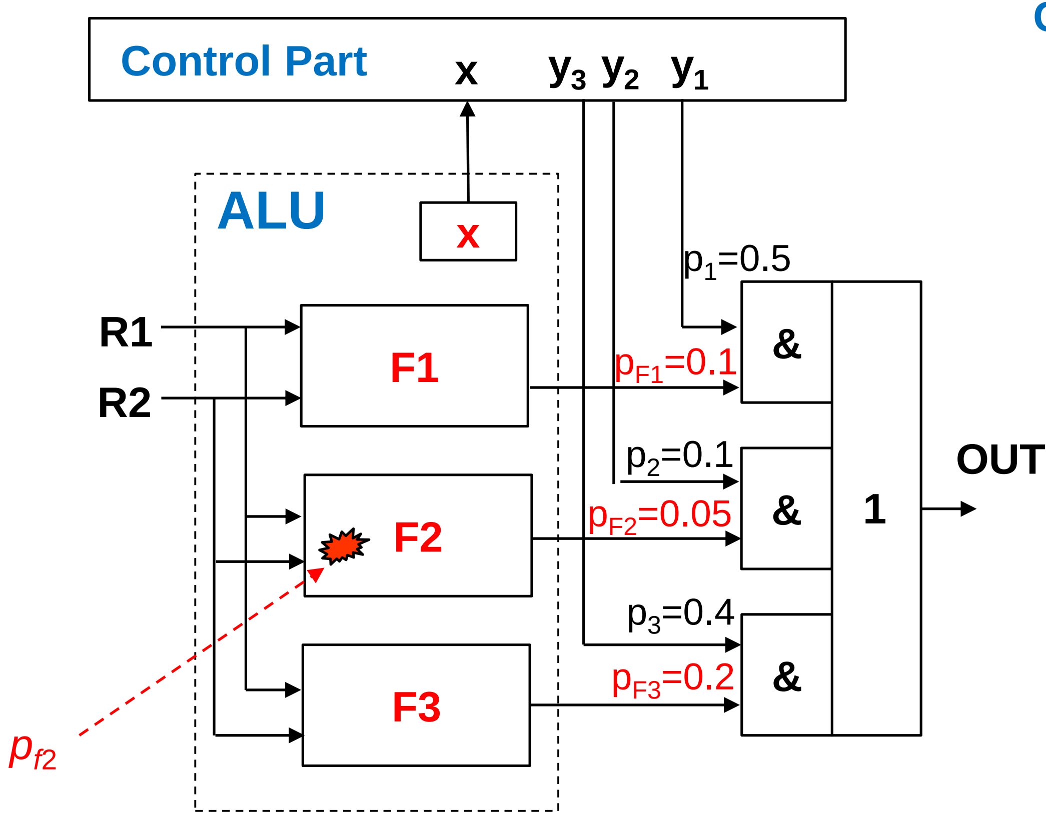
Parker - McCluskey algorithm:

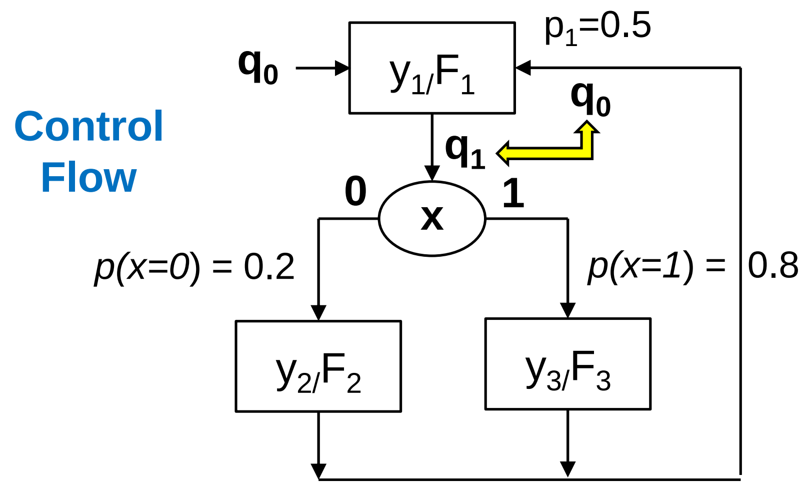
$$
\begin{aligned}
p_y &= 1 - (1 - p_a)(1 - p_b) = \\
&= 1 - (1 - p_{x1}(1 - p_{x2}))(1 - p_{x2}(1 - p_{x1})) = \\
&= 1 - (1 - p_{x1} + p_{x1}p_{x2})(1 - p_{x2} + p_{x1}p_{x2}) = \\
&= 1 - (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p^2_{x1}p_{x2} + \\
&\qquad + p_{x1}p_{x2} - p_{x1}p^2_{x2} + p^2_{x1}p^2_{x2}) = \\
&= 1 - (1 - p_{x2} + p_{x1}p_{x2} - p_{x1} + p_{x1}p_{x2} - p_{x1}p_{x2} + \\
&\qquad + p_{x1}p_{x2} - p_{x1}p_{x2} + p_{x1}p_{x2}) = \\
&= p_{x2} - p_{x1}p_{x2} + p_{x1} - p_{x1}p_{x2} + p_{x1}p_{x2} - \\
&\qquad - p_{x1}p_{x2} + p_{x1}p_{x2} - p_{x1}p_{x2}) = \\
&= p_{x1} + p_{x2} - 2p_{x1}p_{x2} = 0{,}5
\end{aligned}
$$

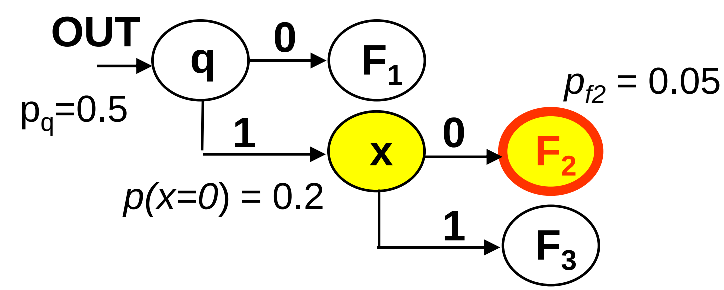
The exponents are removed

Example: RT Level Probabilistic Testing

Functional testing







$P(OUT = F_1) = p_1 = 0.5$

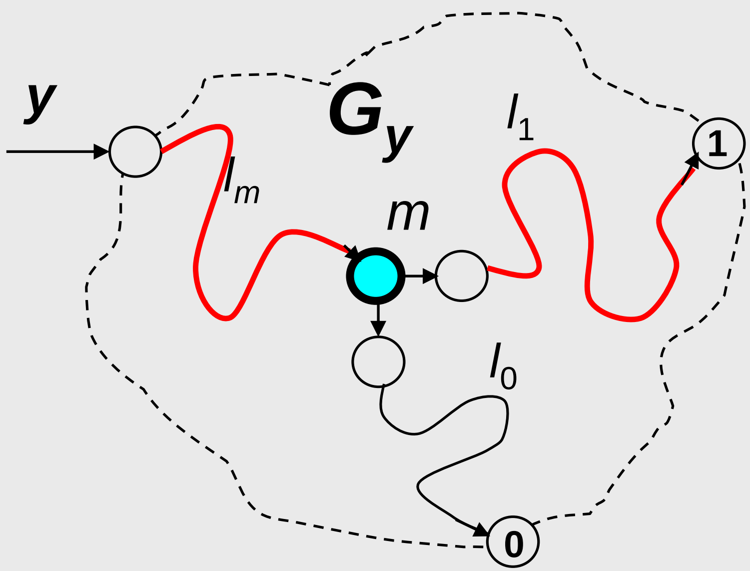
$P(OUT = F_2) = p_2 = p_q * p(x=0) = 0.5 * 0.2 = 0.1$

***P*(Det of *f2*)** **= $p_2 * p_{F2}$ = 0.1 * 0.05 = 0.005**

Generalization of BDDs

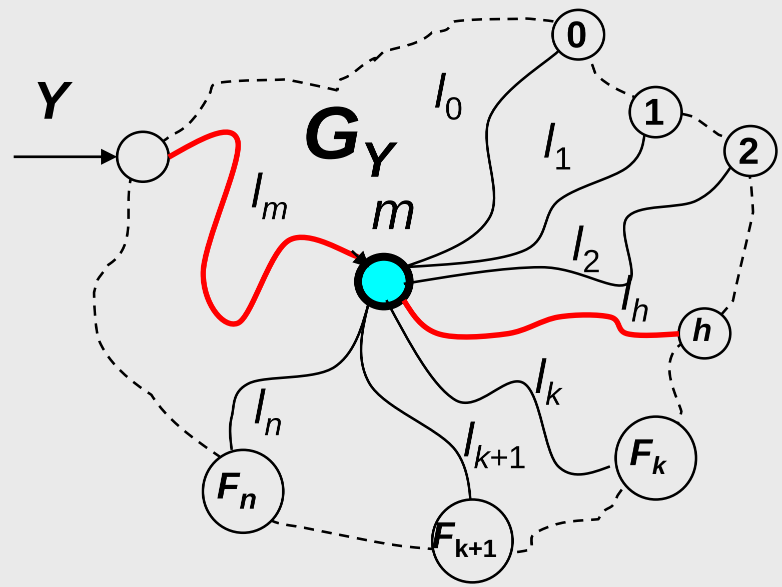
Binary DD

2 terminal nodes and
2 edges from each node



General case of DD

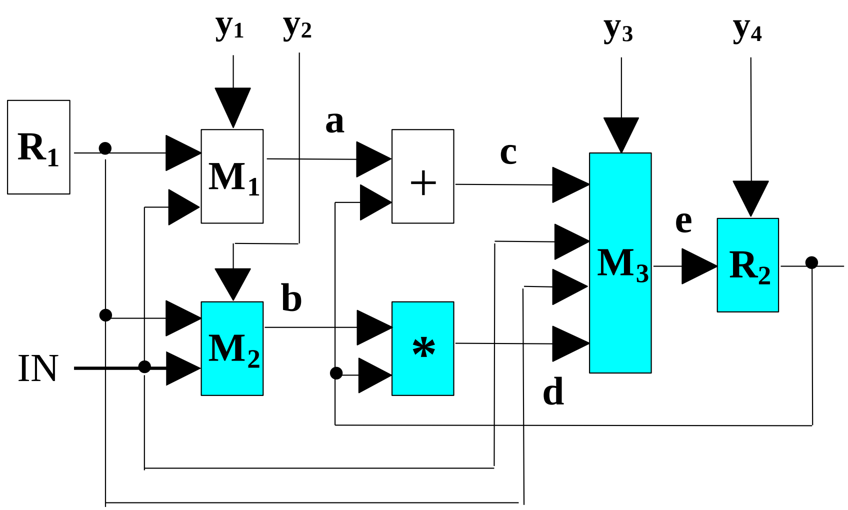
n ≥ 2 terminal nodes and
n ≥ 2 edges from each node

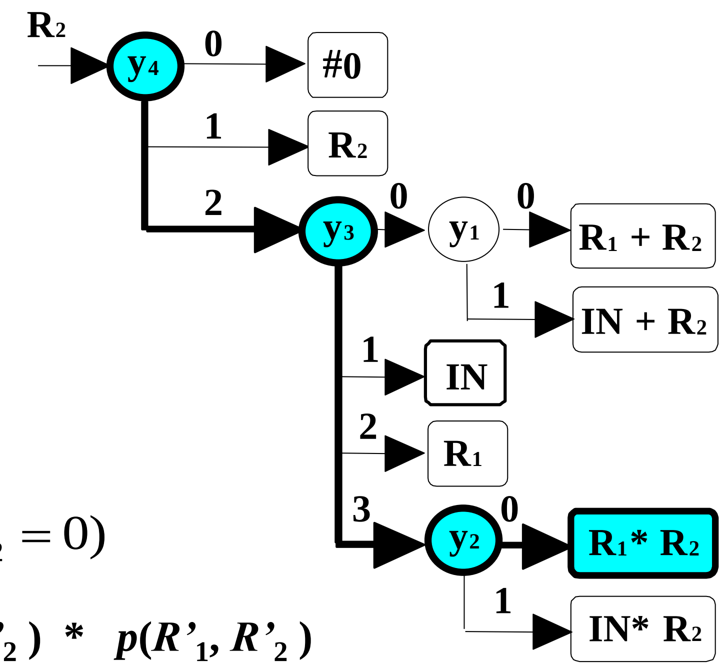


Novelty: **Boolean methods can be generalized in a straightforward way to higher functional levels**

Register Transfer Level DDs

Hierarchical calculation of probabilities:





Probability of $p(R_2 = R_1' \times R_2')$

$$p(R_2 = R_1' \times R_2') = p(y_4 = 2) \times p(y_3 = 3) \times p(y_2 = 0)$$

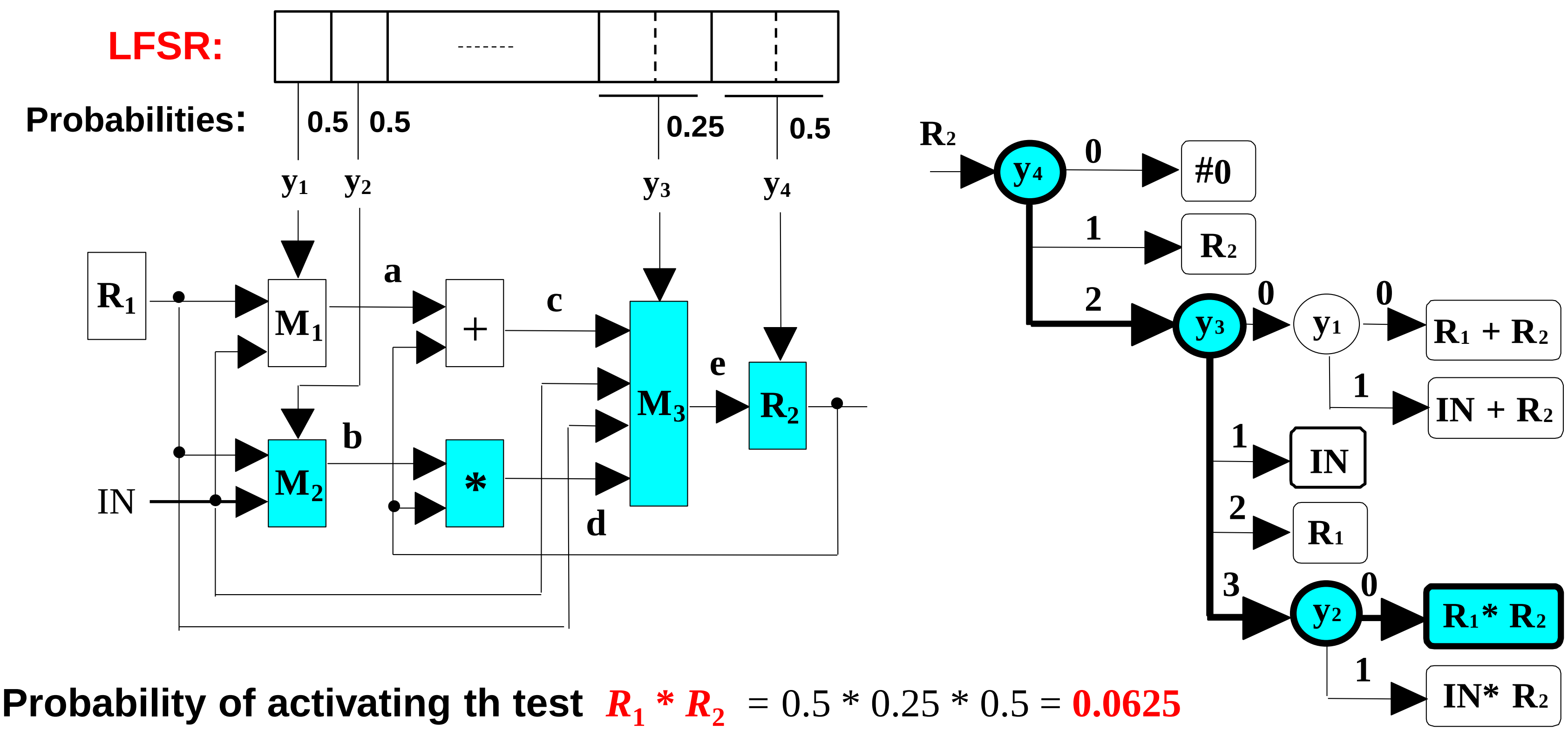
Probability of testing $R_1 * R_2 \rightarrow$ $\boldsymbol{p(R_2 = R'_1 * R'_2) \; * \; p(R'_1, R'_2)}$

$p(R_1, R_2)$ – can be calculated for the low gate-level multiplier model

All $p(y = k)$ – can be calculated for the low level control part model

Register Transfer Level DDs

LFSR-based BIST



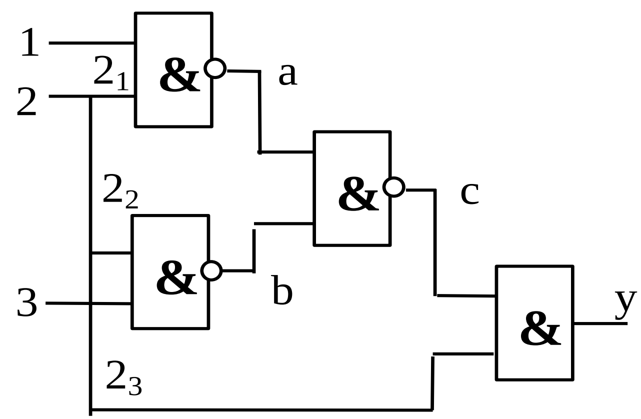
Probability of activating th test $R_1 * R_2$ = 0.5 * 0.25 * 0.5 = **0.0625**

Calculating Probabilities on BDDs

$$p_y = \sum_{L_k \in L(1)} \prod_{x \in X_k} p_x$$

Set of paths

Set of nodes on the path

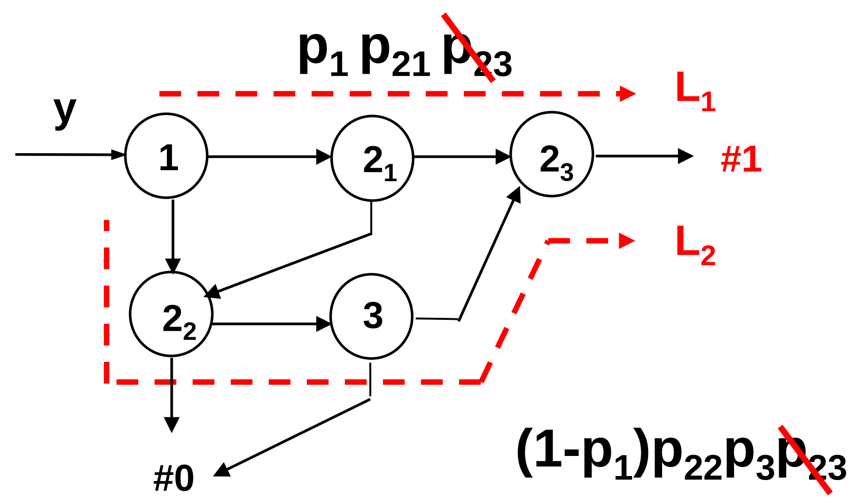


Example:

$L_1 = (1, 2_1, 2_3)$

$L_2 = (1, 2_2, 3, 2_3)$

$p_y = p_1 p_2 + (1-p_1) p_2 p_3 = 0{,}375$



Calculating Probabilities for RT-Level Circuits

Gate-level calculation:

$$p_y = \sum_{L_k \in L(1)} \prod_{x \in X_k} p_x$$

Compare

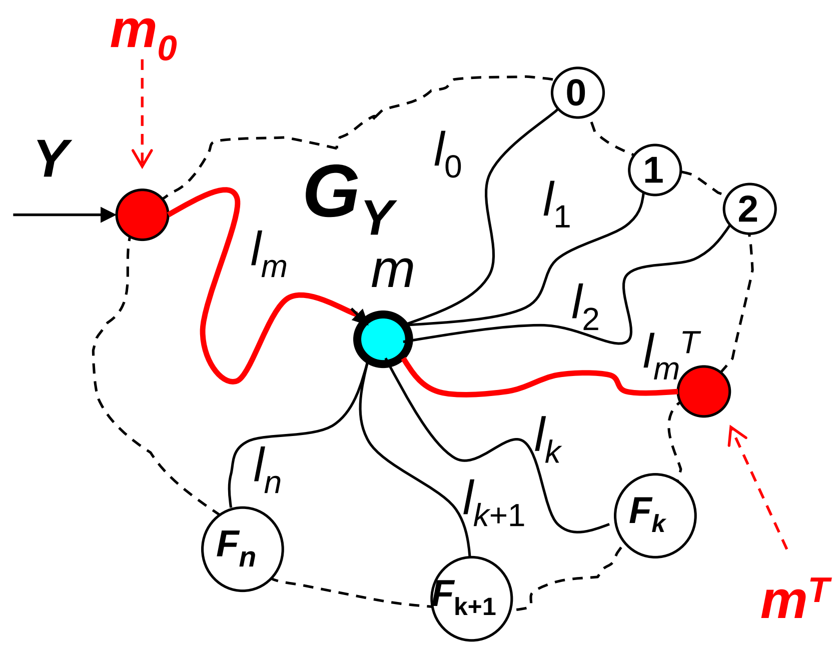
RT-level calculation:

$$P(y = z(m^T)) = \sum_{L_i \in L(m_0, m^T)} \prod_{x \in X_i} P(x)$$

Set of paths

Set of nodes on the path

DD for RT Level Data Path:



Calculating Observabilities RT-Level Data Paths

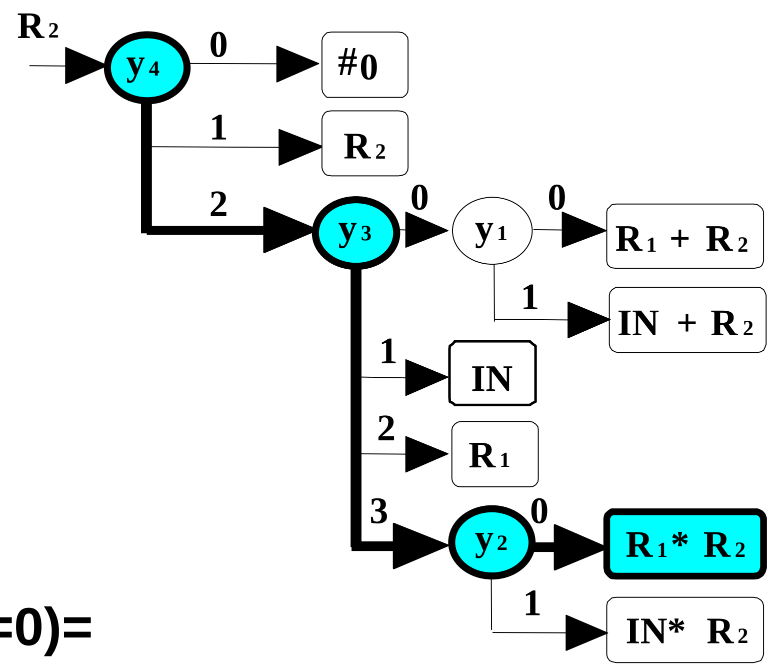
Gate-level calculation:

$$p_y = \sum_{L_k \in L(1)} \prod_{x \in X_k} p_x$$

RT-level calculation:

$$P(y=z(m^T)) = \sum_{L_i \in L(m_0,m^T)} \prod_{x \in X_i} P(x=e)$$

DD for RTL Data Path:



Example:

$$P(R_2 = R_1 * R_2) = P(y_4=2)\,P(y_3=3)\,P(y_2=0) =$$
$$= 0.3 * 0.25 * 0.5 = 0.04$$