

Course 2: IC Design Challenges and Methods for Contemporary
Technologies



**Timing Analysis and Characterization for
Full Custom IP-blocks**

Sergey Gavrilov
IPPM RAS / MIET

Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems

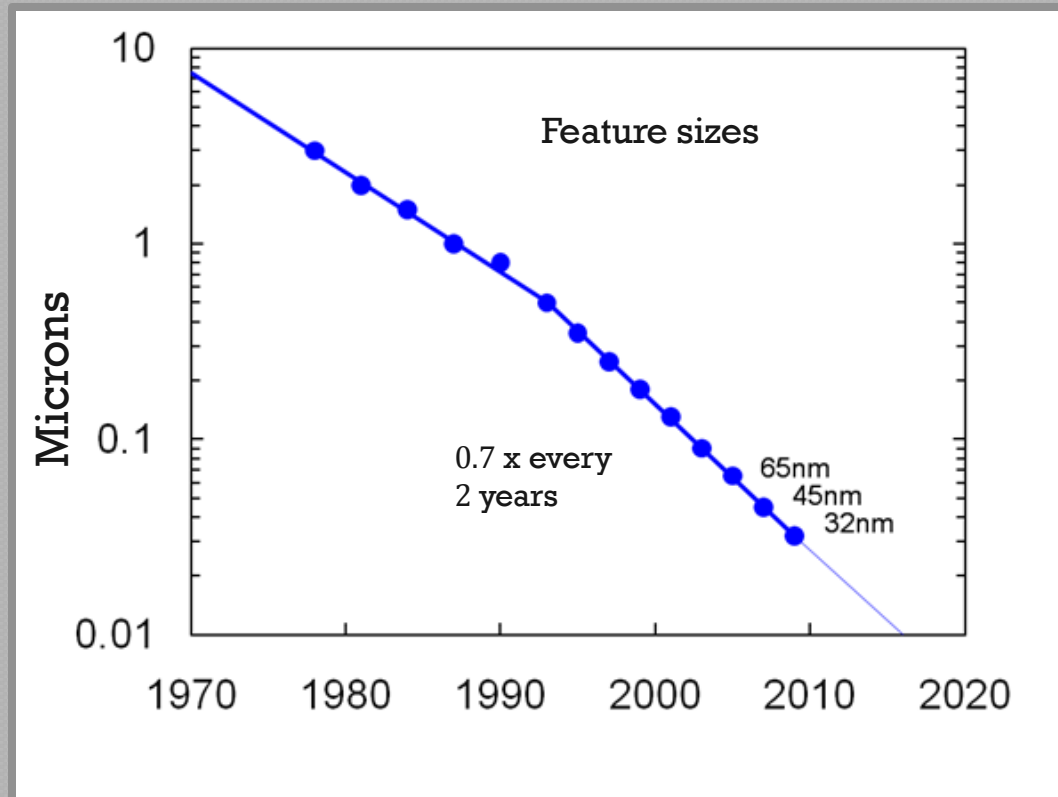


Outline

1. **Contemporary technologies & IP blocks design problems**
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems



Transistor Scaling Trends



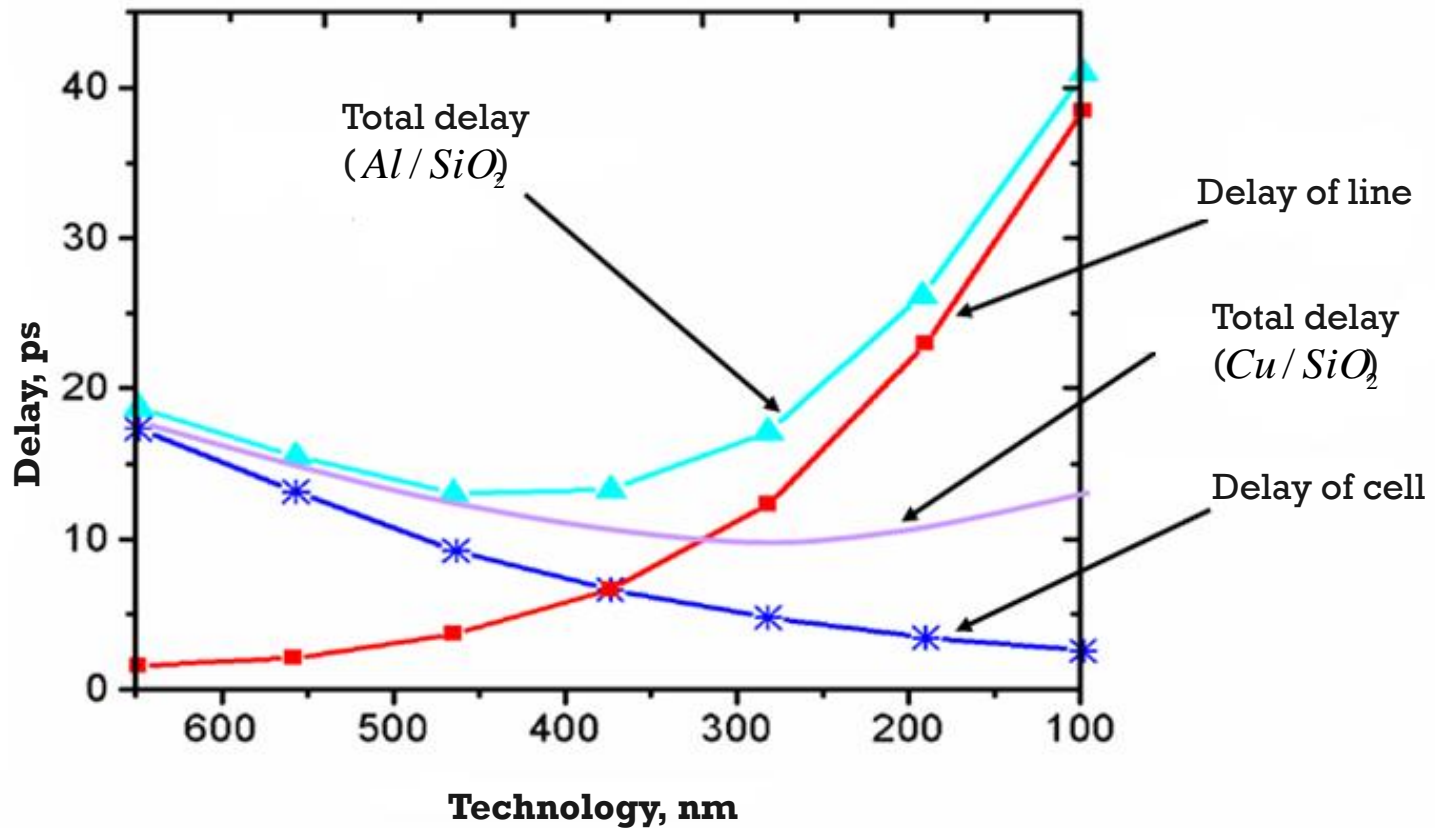
- ✓ Transistor density approximately doubles every two years. - **Moore's Law**.
- ✓ Smaller transistors give **improved performance, reduced power** and **lower cost** per transistor.



Interconnects Dominate the Transistors: Delay (ITRS Data)

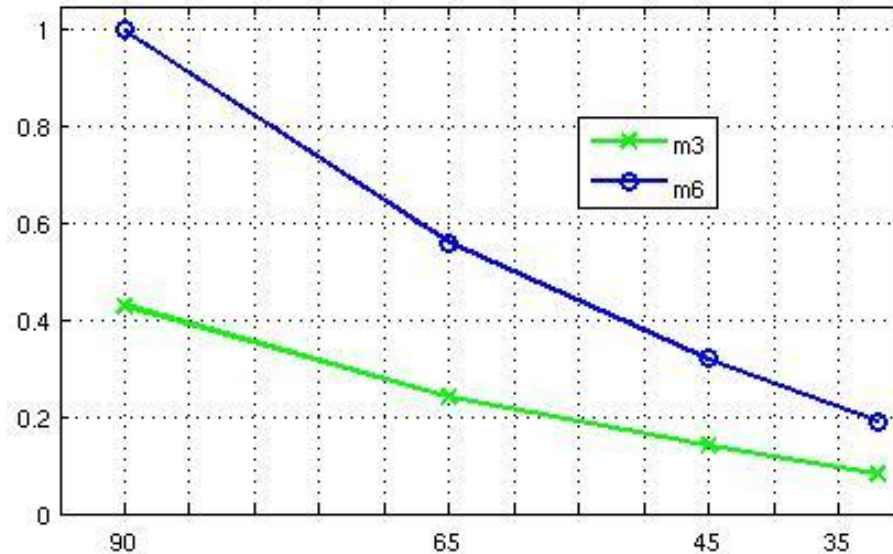
Technology	Gate / Transistor Delay	Delay of line, $L_{int}=1\text{mm}$
1.0 μm (Al, SiO ₂)	~ 20 ps	~ 1 ps
100 nm (Cu)	~ 5 ps	~ 30 ps
35 nm (Cu)	~ 2.5 ps	~ 250 ps

Data ITRS, Delay



Buffered Interconnects:

Critical path without buffered interconnects

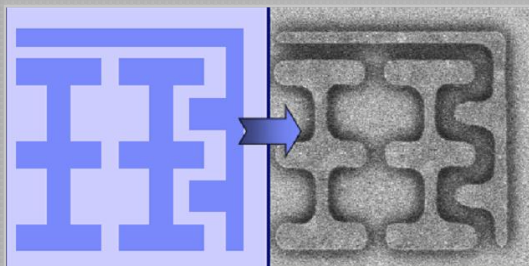


Metal Layers

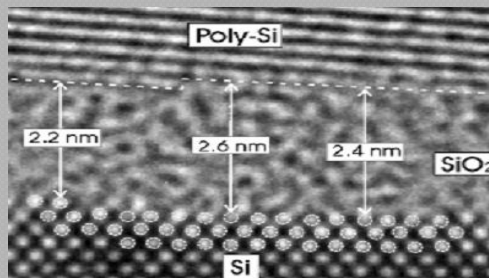
	90nm	65nm	45nm	32nm
m3	0.43	0.24	0.14	0.08
m6	1	0.56	0.32	0.19



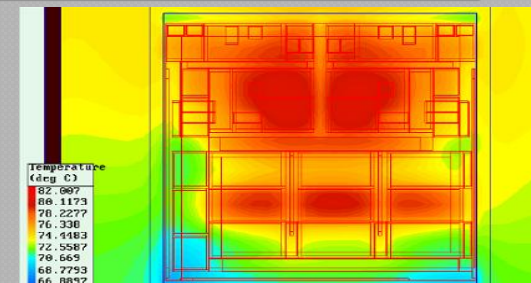
Parameter Variations in the Nanometer Range



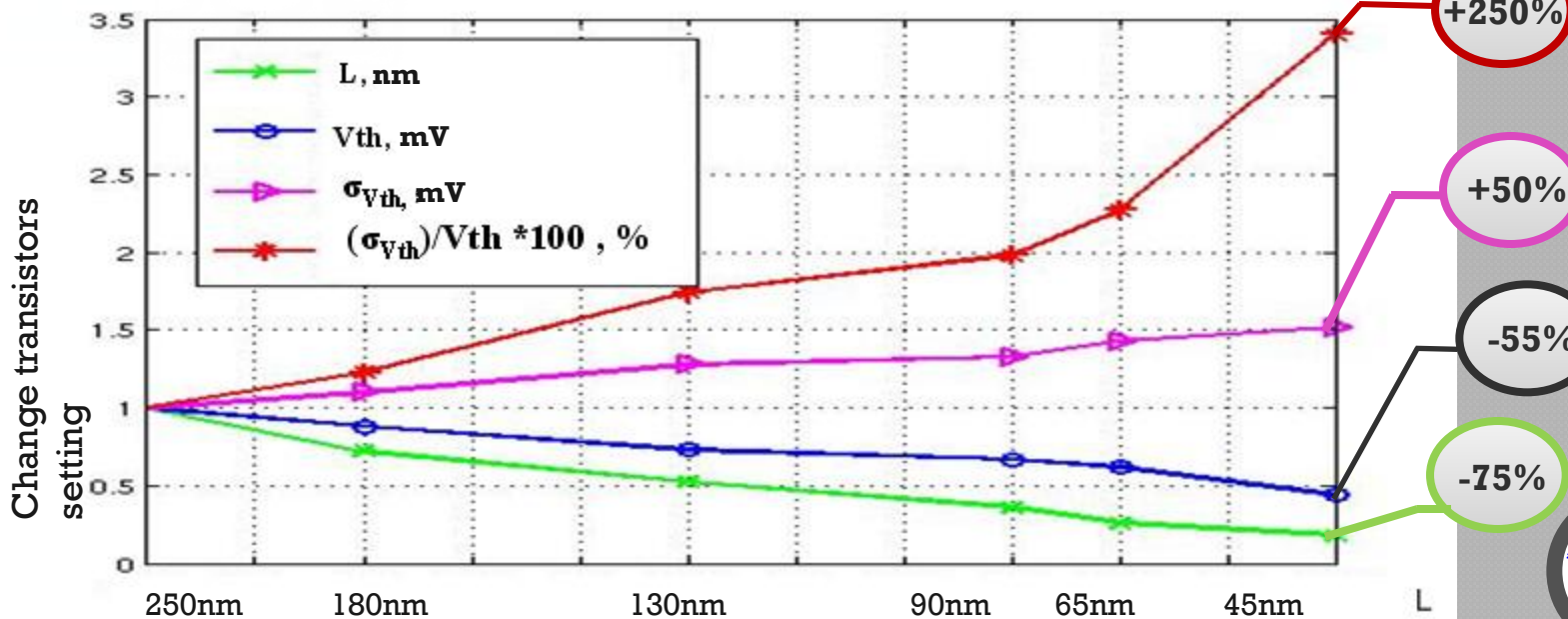
channel length (L_{eff})



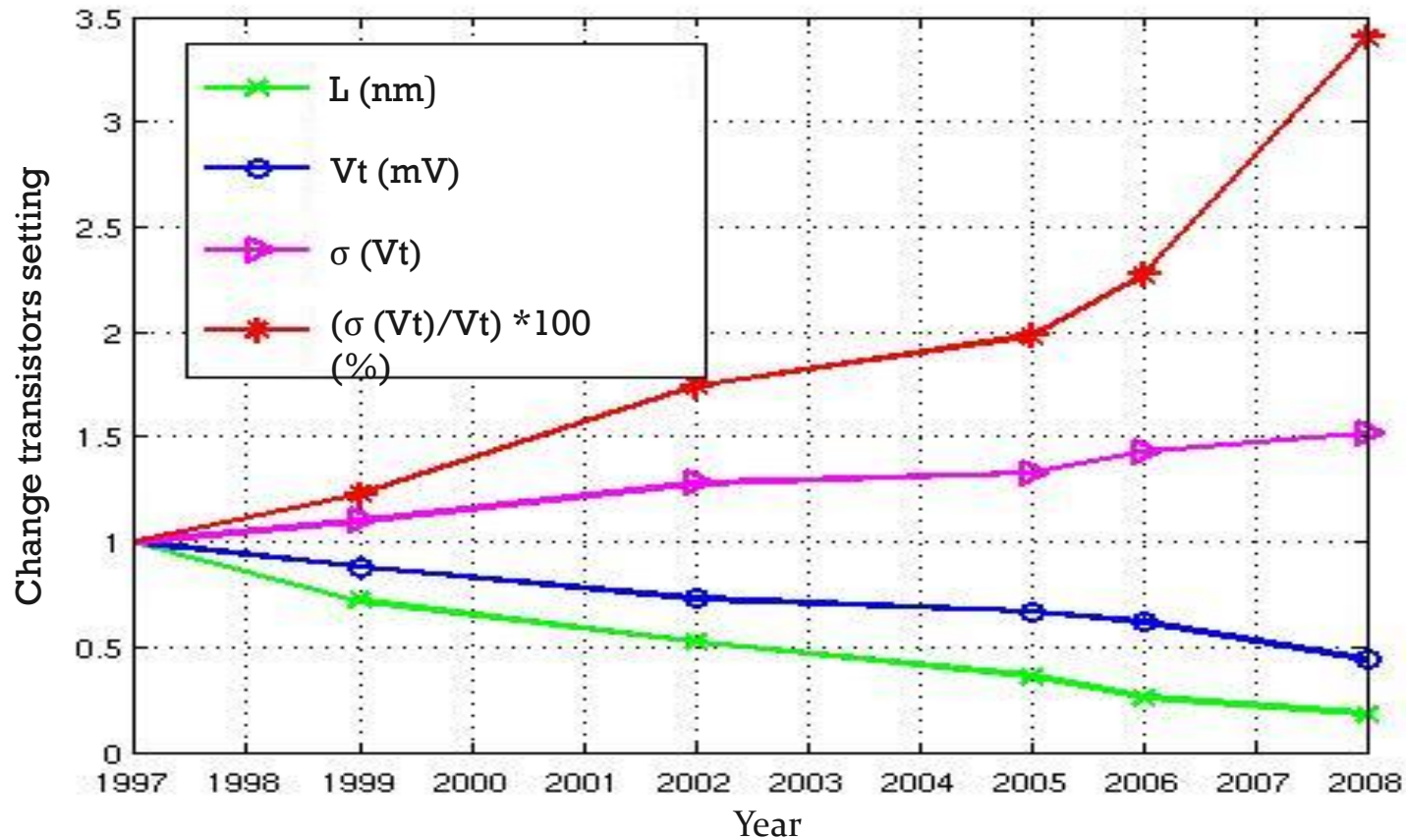
thickness of the oxide layer (T_{ox})



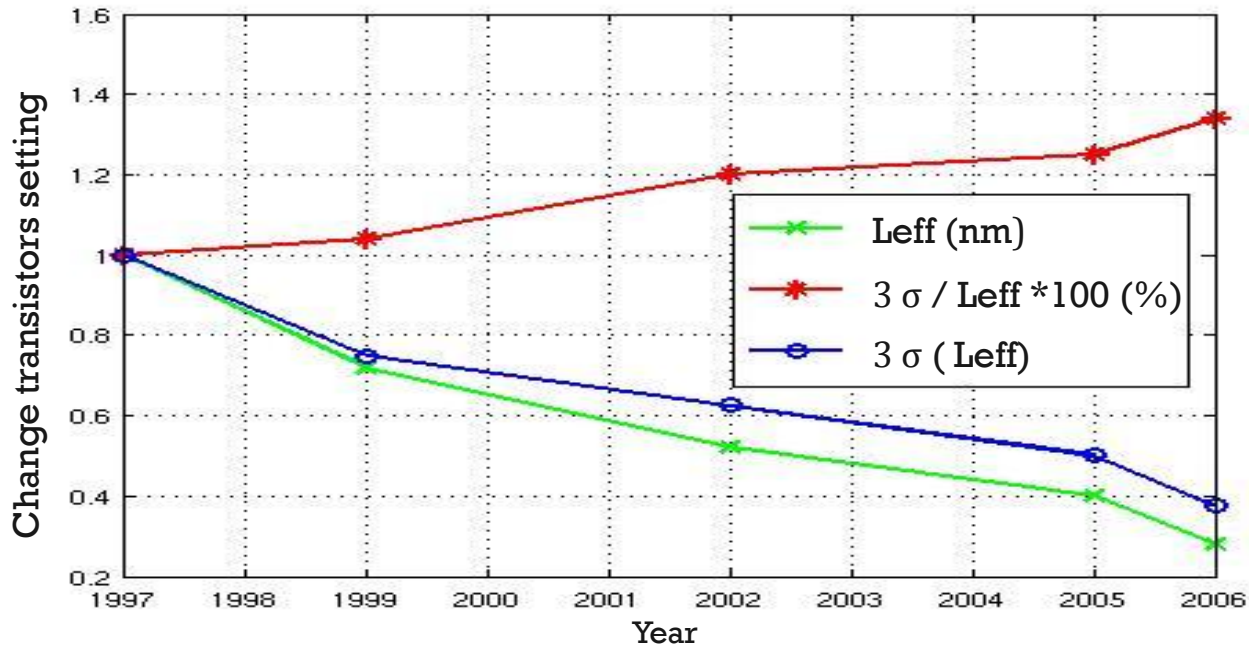
temperature (T)



Threshold Voltage and its Variations Scaling



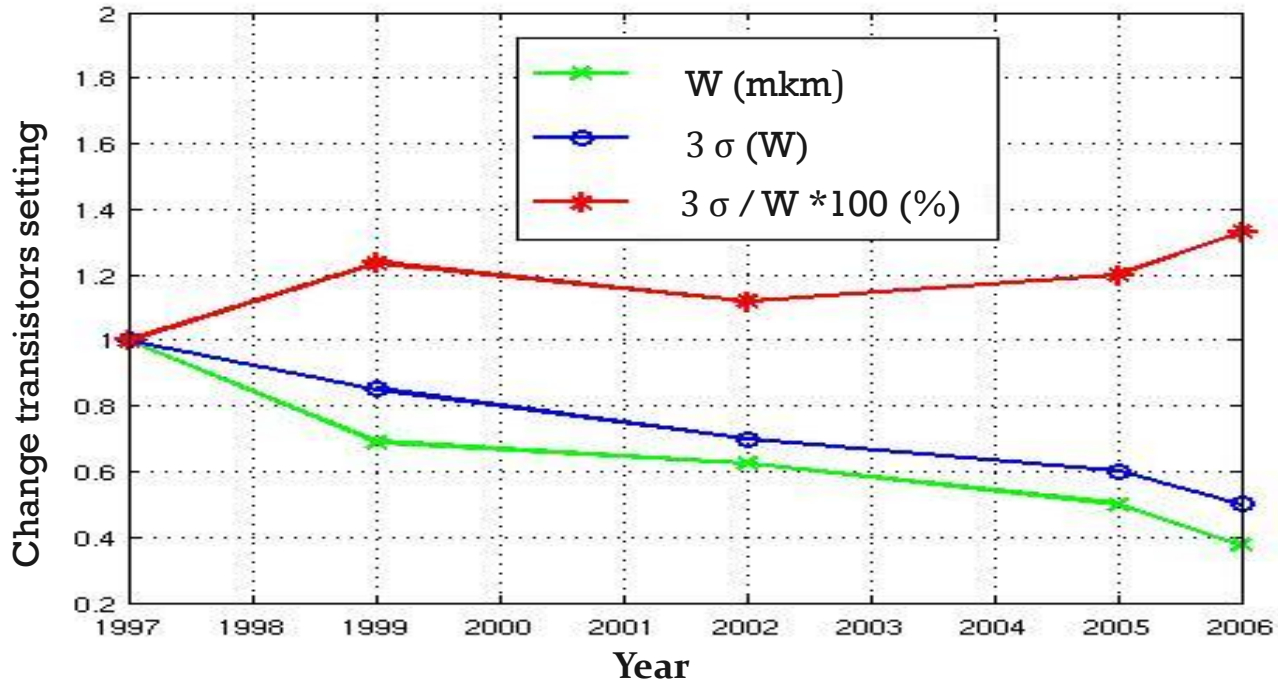
Leff and its Variations Scaling



Year	1997	1999	2002	2005	2006
L_{eff} (nm)	250	180	130	100	70
3σ (Leff)	80	60	50	40	30
$3\sigma / L_{eff} * 100$ (%)	32%	33.3%	38.5%	40%	43%



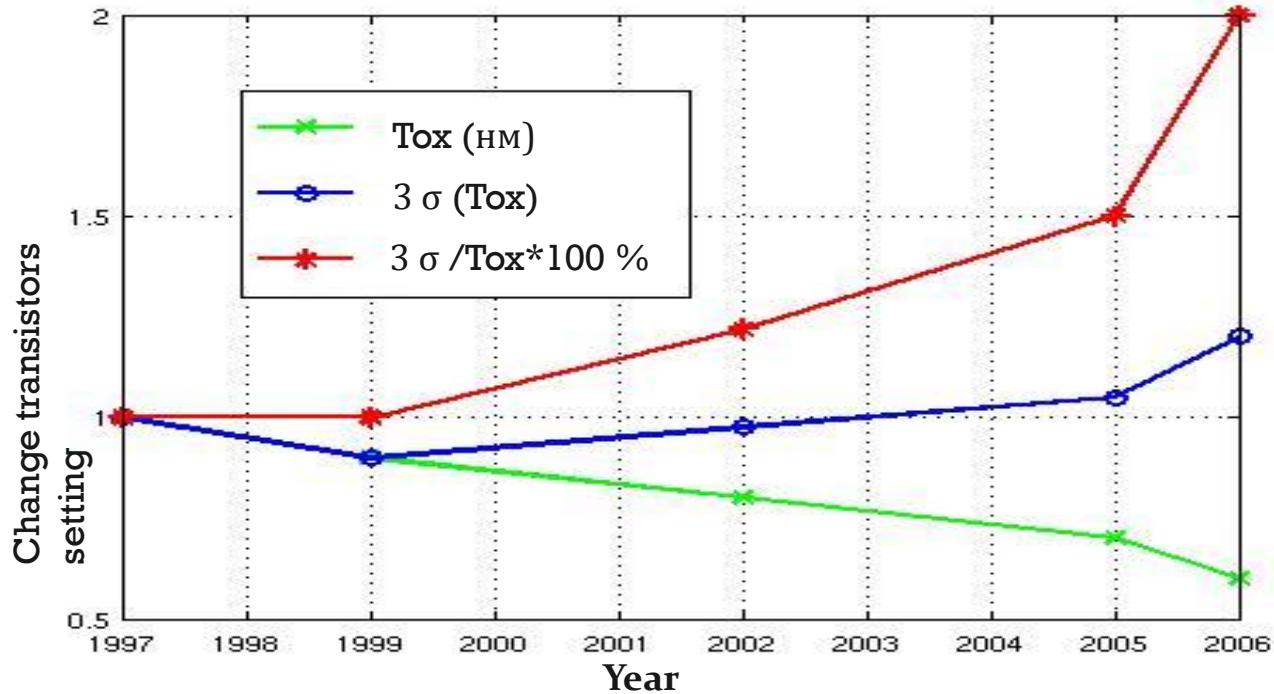
Transistor Width W and its Variation Scaling



Year	1997	1999	2002	2005	2006
W (mkm)	0.8	0.55	0.5	0.4	0.3
3σ (W)	0.2	0.17	0.14	0.12	0.1
3σ / W * 100 (%)	25%	31%	28%	30%	33.3%



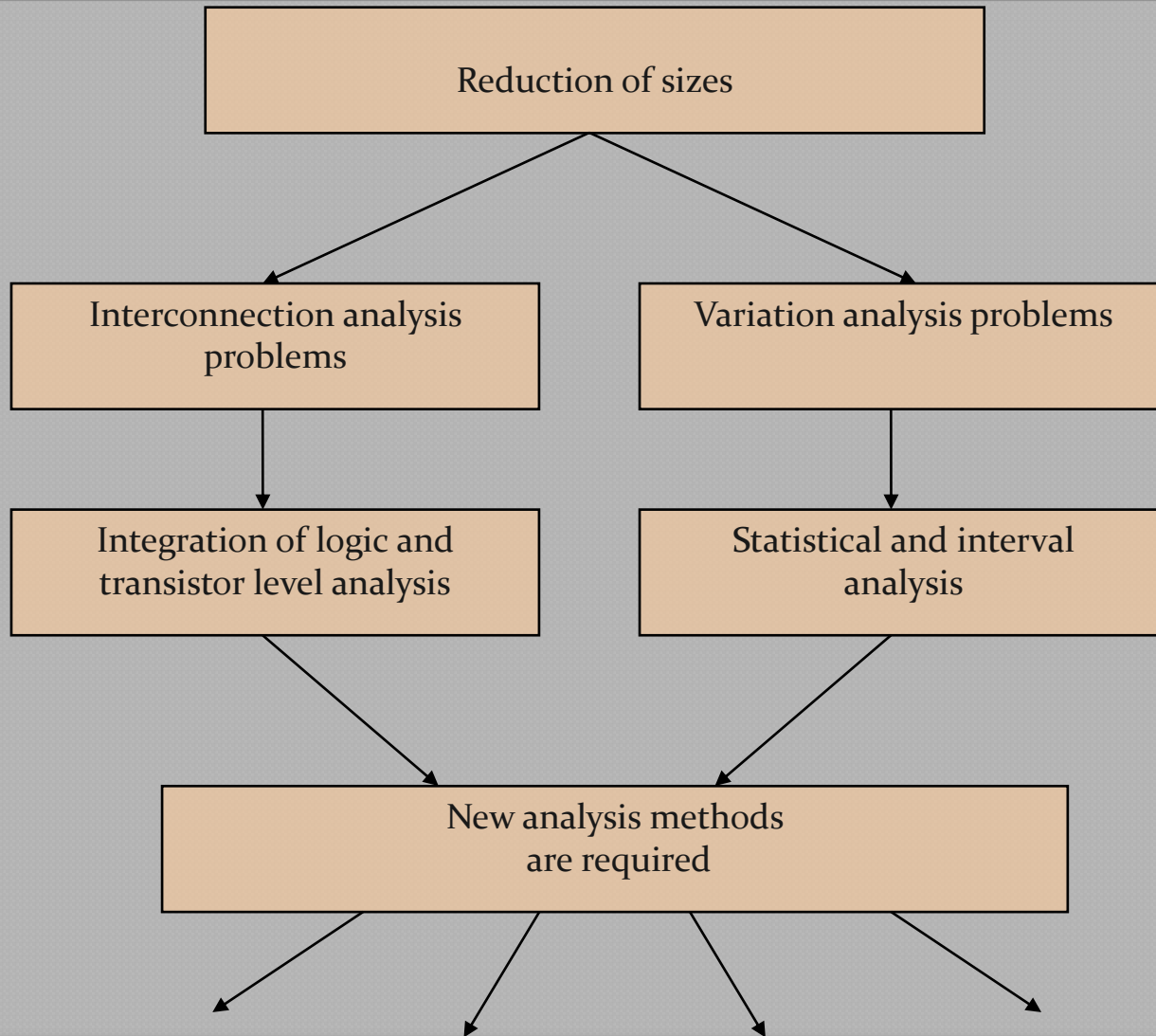
Tox and it's Variation Scaling



Year	1997	1999	2002	2005	2006
Tox (nm)	5.0	4.50	4.00	3.50	3.0
3σ (Tox)	0.4	0.36	0.39	0.42	0.48
3σ / Tox * 100(%)	8%	8%	9.75%	12%	16%



IP-Blocks Timing Analysis and Characterization Trends



Outline

1. Contemporary technologies & IP blocks design problems
- 2. Deterministic and statistical timing analysis**
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems



Static Timing Analysis (STA)

2 passes:

Forward propagation (from primary inputs to primary outputs)

LAT (Latest Arrival Time) for Setup restriction analysis

EAT (Earliest Arrival Time) for Hold restriction analysis.

Backward propagation (from primary outputs to primary inputs)

LRT (Latest Required Time)

ERT (Earliest Required Time).

The interval [**EAT,LAT**] – real arrival window

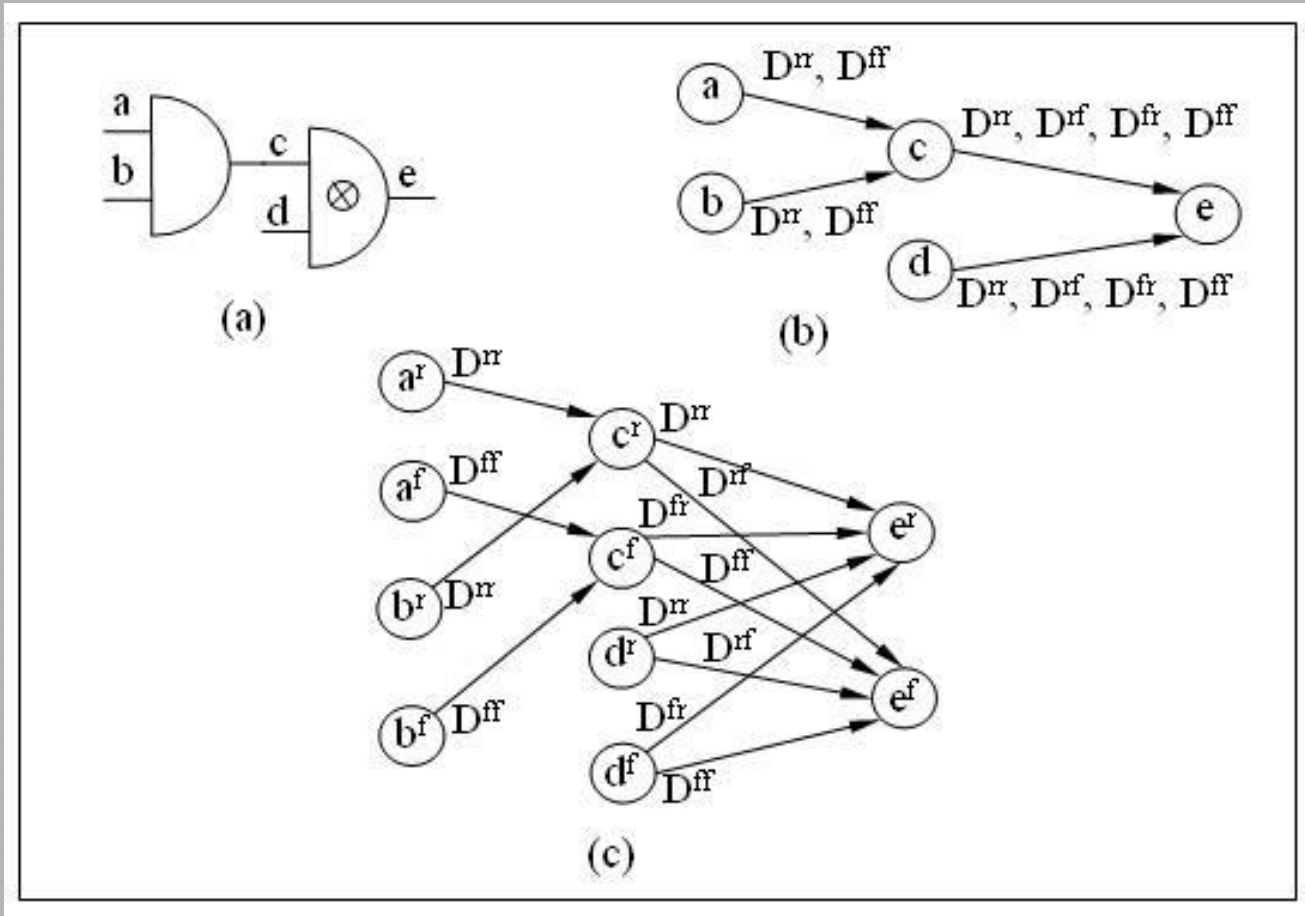
The interval [**ERT,LRT**] – required window.

$$\text{output}[j].\text{LAT} = \text{MAX} (\text{input}[i].\text{LAT} + \text{gate.delay}[i][j])$$

$$\text{input}[i].\text{LRT} = \text{MIN} (\text{output}[j].\text{LRT} - \text{gate.delay}[i][j])$$



Circuit Example(a), Delay Graph (b), Modified Delay Graph (c) – splitting fall / rise



Block Oriented Statistical Timing Analysis

X – normal distributed value (Delay, Slope, etc.) with mean value m_X and dispersion σ_X^2

$$X = m_X + \sigma_X \Delta X$$

ΔX – random value ($m_X=0$, $\sigma_X=1$)

Linear approximation (no correlations) assumption:

$$A = a_0 + \sum_{i=1}^n a_i \Delta X_i + r_a \Delta R_a$$

a_0 – mean nominal value,

ΔX_i , $i=1, \dots, n$, и ΔR_a – random values with normal distributions



Block Oriented Statistical Timing Analysis

Sum & Max operations:

$$1) C = A + B: c_o = a_o + b_o, c_i = a_i + b_i, r_c = \sqrt{r_a^2 + r_b^2}$$

$$2) C = \max(A, B)$$

$$\sigma_A = \sqrt{\sum_{i=1}^n a_i^2 + r_a^2}$$

$$\sigma_B = \sqrt{\sum_{i=1}^n b_i^2 + r_b^2}$$

Correlation coefficients: (for independent ΔX_i and ΔR_a):

$$\rho = \frac{1}{\sigma_A \sigma_B} \sum_{i=1}^n a_i b_i$$



Block Oriented Statistical Timing Analysis

Output mean value for C:

$$c_0 = a_0T + b_0(1-T) + \theta\varphi\left(\frac{a_0 - b_0}{\theta}\right)$$

Distribution for C:

$$\sigma_C^2 = (\sigma_A^2 + a_0^2)T + (\sigma_B^2 + b_0^2)(1-T) + (a_0 + b_0)\theta\varphi\left(\frac{a_0 - b_0}{\theta}\right)$$

Coefficient recalculation for C:

$$c_i = a_iT + b_i(1-T) \quad , i=1, \dots, n, \quad r_c = \sqrt{\sigma_C^2 - \sum_{i=1}^n c_i^2}$$



Increasing of Characterizations for Block Oriented Statistical Timing Analysis

$$D = D_0 + \sigma_D^X \cdot \Delta X$$

σ_D^X – delay sensitivity for a given parameter X

D_0 – nominal gate delay,

ΔX – random distribution

$$\sigma_D^X = \frac{D(X_{min}) - D(X_{max})}{2k}$$

X_{min} , X_{max} are side points of the interval $[-ks, +ks]$.



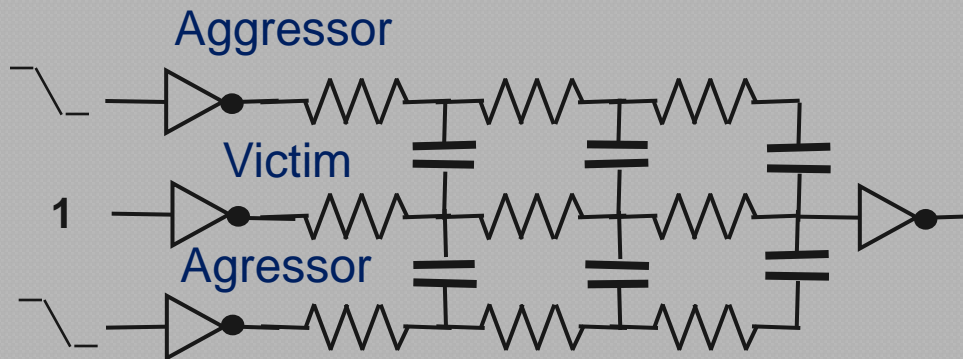
Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
- 3. Digital noise analysis problems**
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems

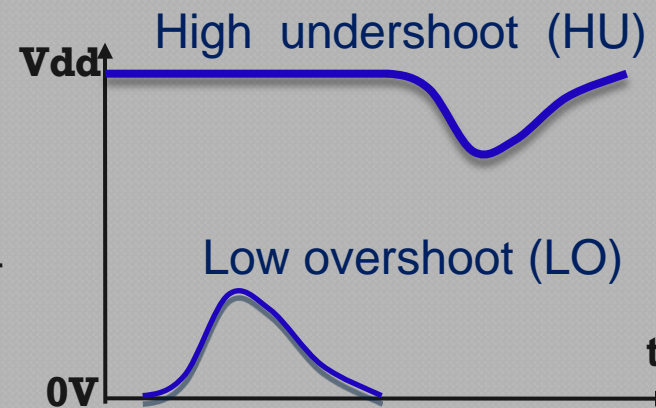


Interconnect Extraction and Cross Coupling Noise

- Aggressor nets affect victim net through coupling capacitances
- Functional Noise: changes logic state of the victim net
- Delay Noise: affects signal propagation delay
- Different types of functional noises:
 - victim state and aggressor switching direction
 - Low/High Overshoot/Undershoot

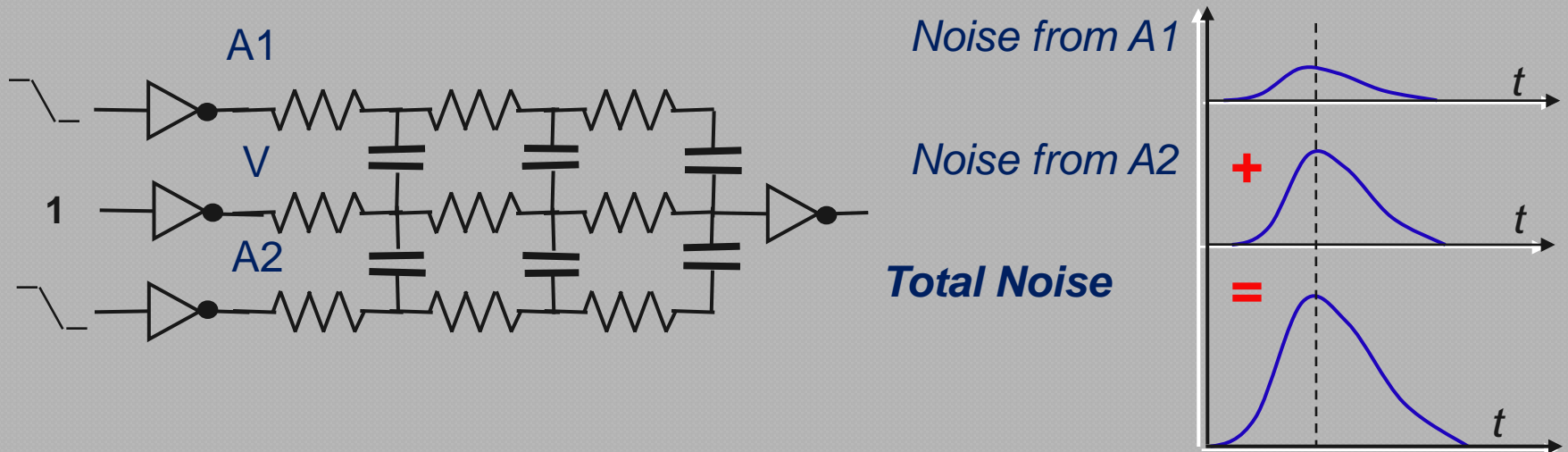


Noise cluster



Conservative Coupling Noise Analysis

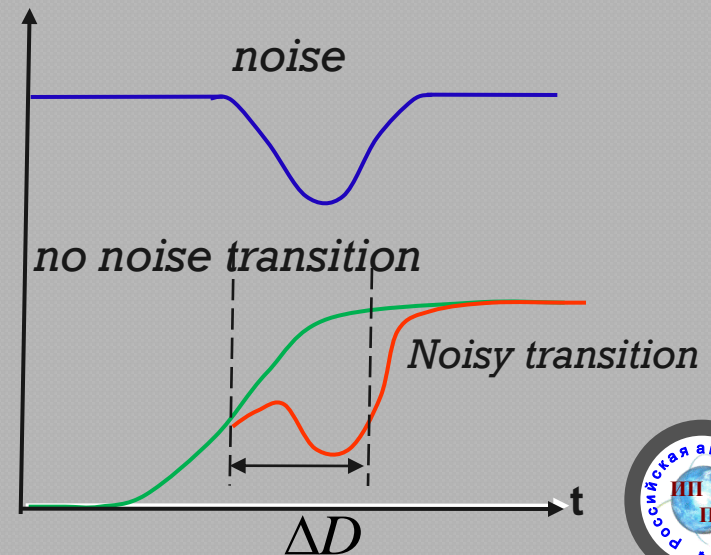
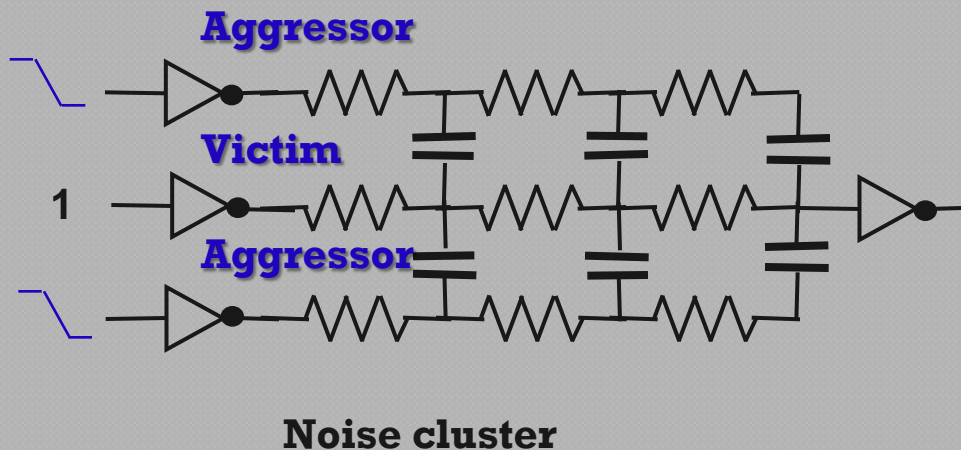
- All aggressor nets switch simultaneously in the same direction
- All aggressor noises combine to create maximum noise
- Aggressors switching times align to inject maximum noise



- ✓ Ignores correlation between circuit signals and may overestimate noise
- ✓ May produce *false noise violations*
- ✓ New method to reduce false noise violations by using logic implications

Cross Coupling Delay Noise

- ✓ Aggressor nets affect victim net through coupling capacitances
- ✓ Functional Noise: changes logic state of the victim net
 - Affects victim when it is in a stable state
- ✓ Delay Noise: changes signal propagation delay
 - Affects victim net when it transitions
 - Delay changes accumulate along the signal propagation paths



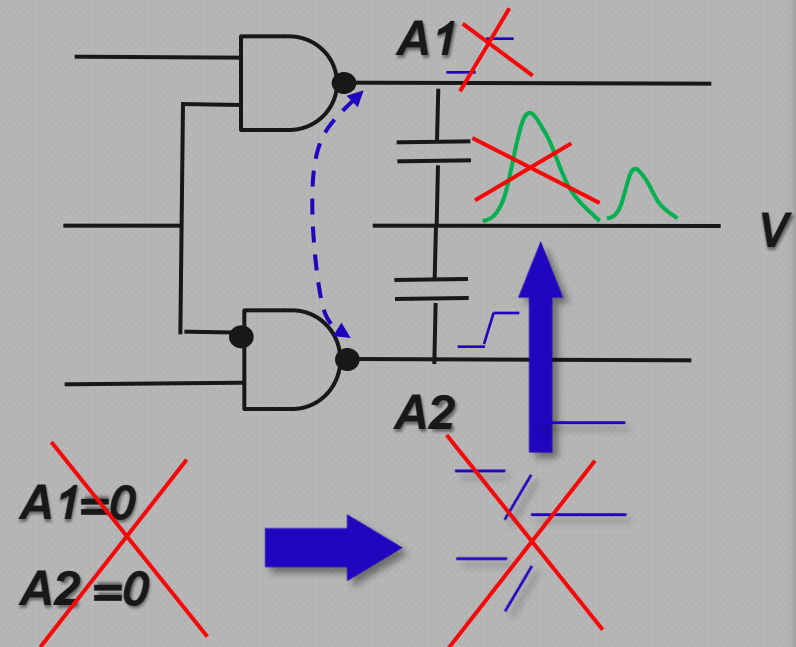
Signals Correlation and False Noise

✓ Timing correlation:

- nets switch at different clock cycles, etc.

✓ Logic correlation:

- circuit logic **prohibits** some combinations of nets signals
- it prohibits some aggressor nets from simultaneous switching



✓ Ignoring signal correlation overestimates noise and results in false noise violations

- makes difficult to recognize actual noise violations
- diminishes trust in noise analysis results

✓ False noise analysis is needed

Delay Noise Model

Linear approximation for small impulses:

$$\Delta D = \frac{\partial D}{\partial h_n} * \Delta h_n + \frac{\partial D}{\partial w_n} * \Delta w_n + \dots$$

The total delay increment is sum of independent aggressors increments:

$$\Delta D = \sum \Delta D_i$$

The total delay increment across the path from input to output:

$$\Delta D_p = \sum_{i \in P} \sum_{j \in A_i} \Delta D_{i,j}$$

$\Delta D_{i,j}$ – delay increment for i -th «victim» in the path P due to Noise from j -th «aggressor».



Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
- 4. Logic cell characterization**
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems



Non-Linear Delay Model (NLDM)

Table characterization

$$D_{out}(S_{inp}^k, C_{out}^l) , k \in [1: Ns], l \in [1: Nc]$$

$$S_{out}(S_{inp}^k, C_{out}^l) , k \in [1: Ns], l \in [1: Nc]$$

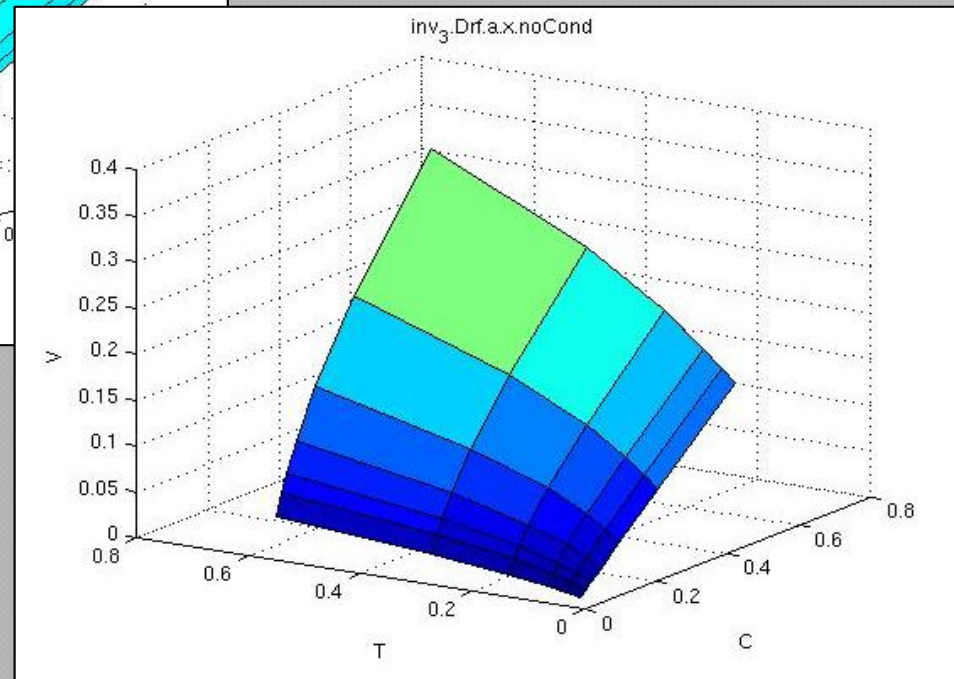
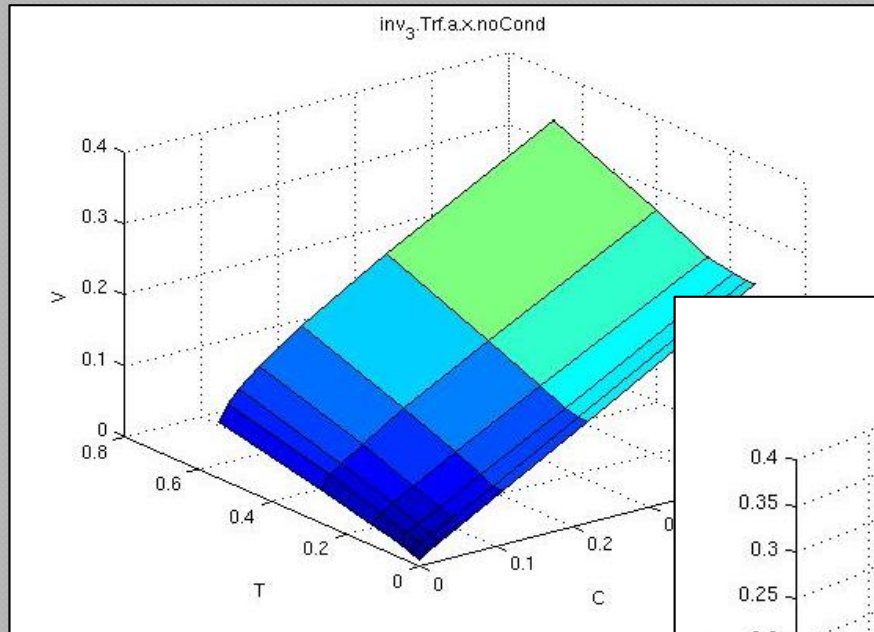
For each delay graph $arc_{ij} = (inp_i, out_j)$, when logic input state switch inp_i results in output switch out_j

Simplified NLDM input caps

C_{inp}^r, C_{inp}^f , for each input, can be different for $0 \Rightarrow 1$ (r), $1 \Rightarrow 0$ (f).



Results of Characterization



Composite Current Source (CCS)

CCS driver model:

$$I_{out} = F(t, S_{inp}, C_{out})$$

CCS efficient input caps for different arcij

$$C_{-1}^{(r/f)}(S_{inp}, C_{out}), C_{-2}^{(r/f)}(S_{inp}, C_{out}),$$

C₁ – the table of caps for the 1-th half of transition,]

C₂ – the table of caps for the 2-d half of transition



Efficient Current Source Model (ECSM)

1) ECSM table: $V_{out} = G(t, S_{inp}, C_{out})$.

$V_{out}(t) \in [0 + \varepsilon, Vdd - \varepsilon]$, ε - constant.

$V_{out}(t)$ is normilized: $\frac{1}{Vdd}$ To the interval (0, 1).

2) ECSM input caps: $C_{inp}^{(r|f)}$ (S_{inp}, C_{out})

CCS & ECSM models are equivalent theoretically:

$$I_{out}(t) = C_{out} * \frac{dV_{out}(t)}{dt}$$

Results are different practically.



Logic Characterization Input Data

Example for AND₂

```

-----
f_loop_set    slew_low_threshold          0.2*$vdd
f_loop_set    slew_upper_threshold      0.8*$vdd
# длительность периода входных сигналов
f_loop_set    time_slice                  20
# описание теста
f_loop_testcase      delay      test1
# описание формы сигналов на входах ЛЭ
      f_loop_waveform      i1      "rfr1rfrf"
      f_loop_waveform      i2      "1rfrforf"
# задание измерений задержек и фронтов
      f_loop_measure      delay      i1 r 1 x r
      f_loop_measure      delay      i1 f 2 x f
      f_loop_measure      delay      i2 r 4 x r
      f_loop_measure      delay      i2 f 5 x f
      f_loop_measure      delay      i1 r 7 x r
      f_loop_measure      delay      i2 r 7 x r
      f_loop_measure      delay      i1 f 8 x f
      f_loop_measure      delay      i2 f 8 x f
# задание измерений входных емкостей
      f_loop_measure      cap        i1 1
      f_loop_measure      cap        i2 4
      f_loop_measure      cap        i1 2
      f_loop_measure      cap        i2 5
f_loop_end
# запись результатов в выходной файл
f_macro_write dotlib
f_loop_destroy

```



Outline

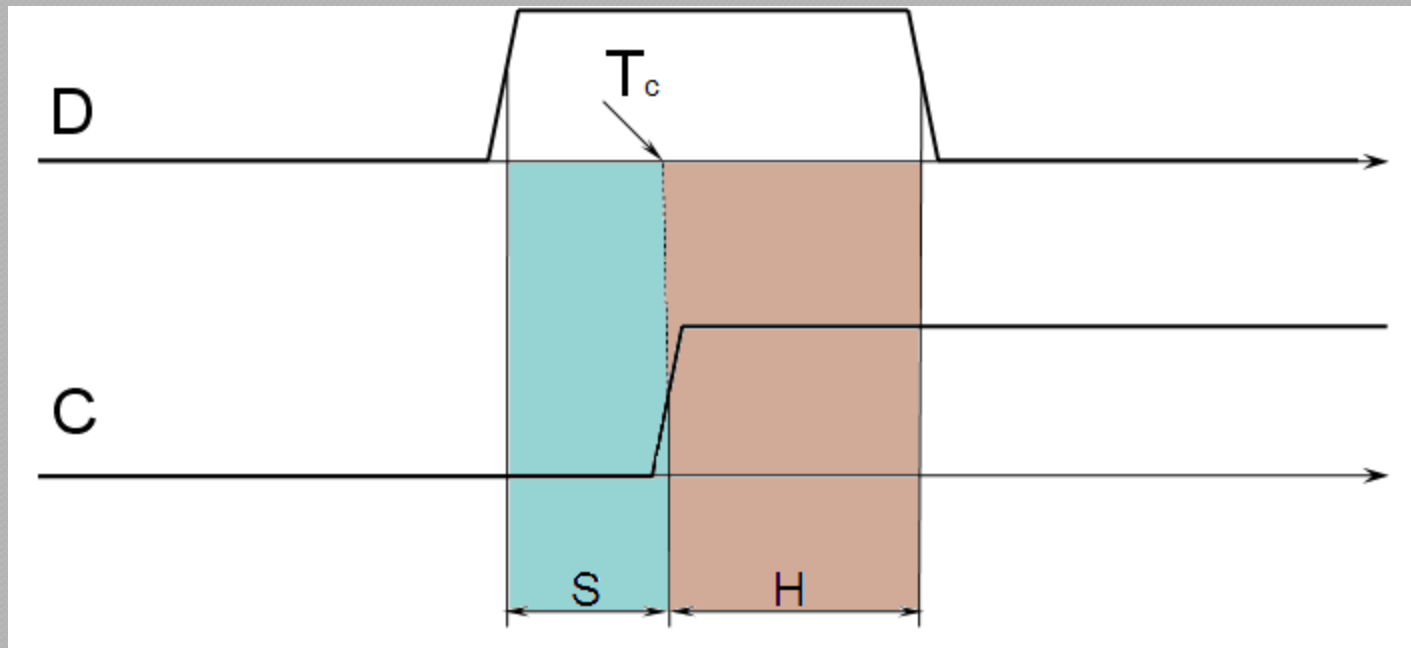
1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
- 5. Memory cell characterization**
6. Decomposition problems for IP blocks
7. Logic correlation analysis for timing and noise estimation
8. Input stimulus generation for IP blocks
9. IP blocks characterization speed-up
10. Future technologies problems



Setup & Hold Characterization

Restriction control:

- ✓ Correct output switching
- ✓ Delay degradation control



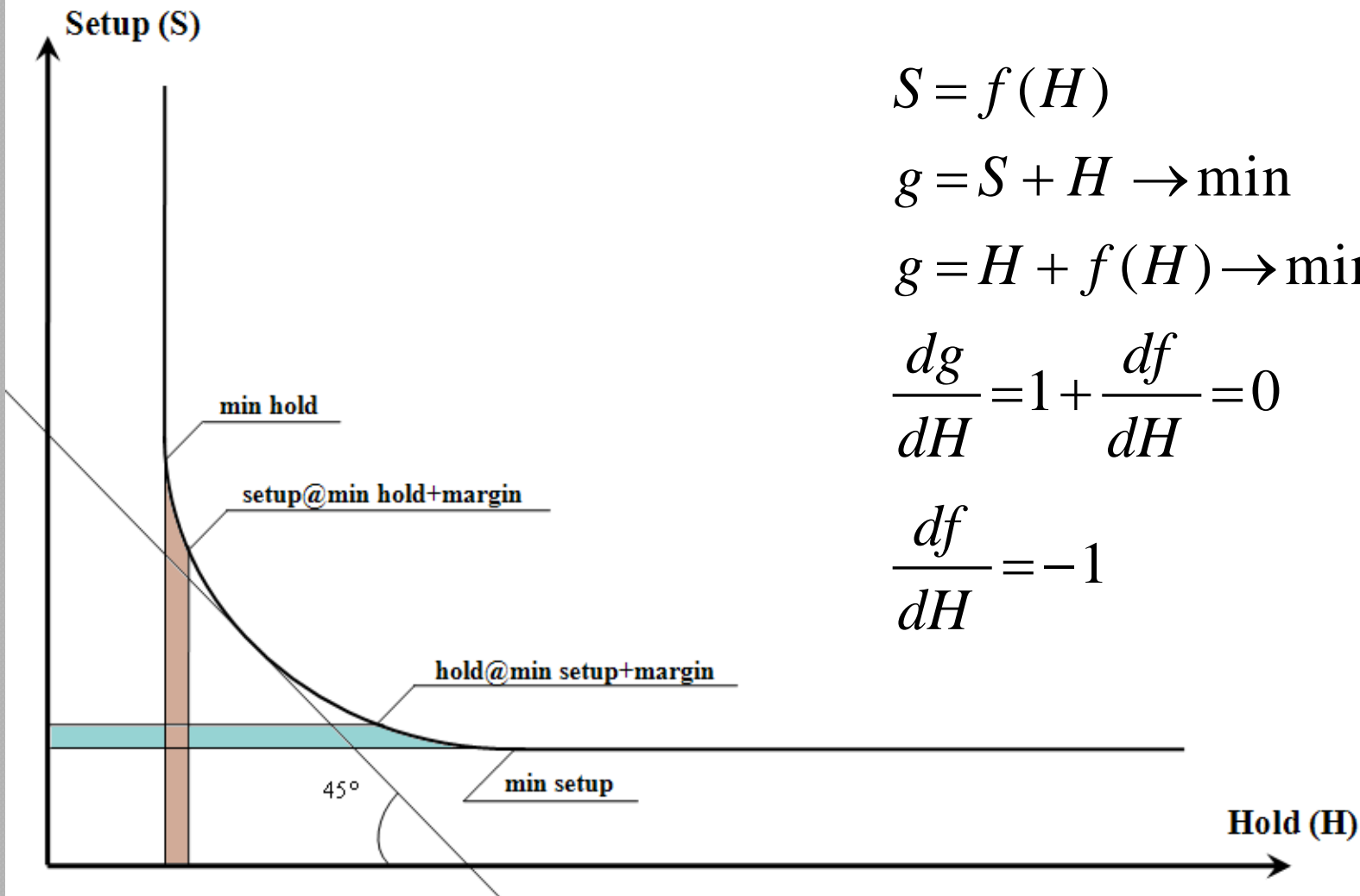
Different Types of Setup & Hold Characterization

- ✓ Independent Setup
- ✓ Independent Hold
- ✓ Dependent Setup (Hold first)
- ✓ Dependent Hold (Setup first)
- ✓ Minimal SUM = Setup + Hold
- ✓ 3D interdependent characterization

Delay(Setup, Hold)



Express Analysis of Setup and Hold



$$S = f(H)$$

$$g = S + H \rightarrow \min$$

$$g = H + f(H) \rightarrow \min$$

$$\frac{dg}{dH} = 1 + \frac{df}{dH} = 0$$

$$\frac{df}{dH} = -1$$

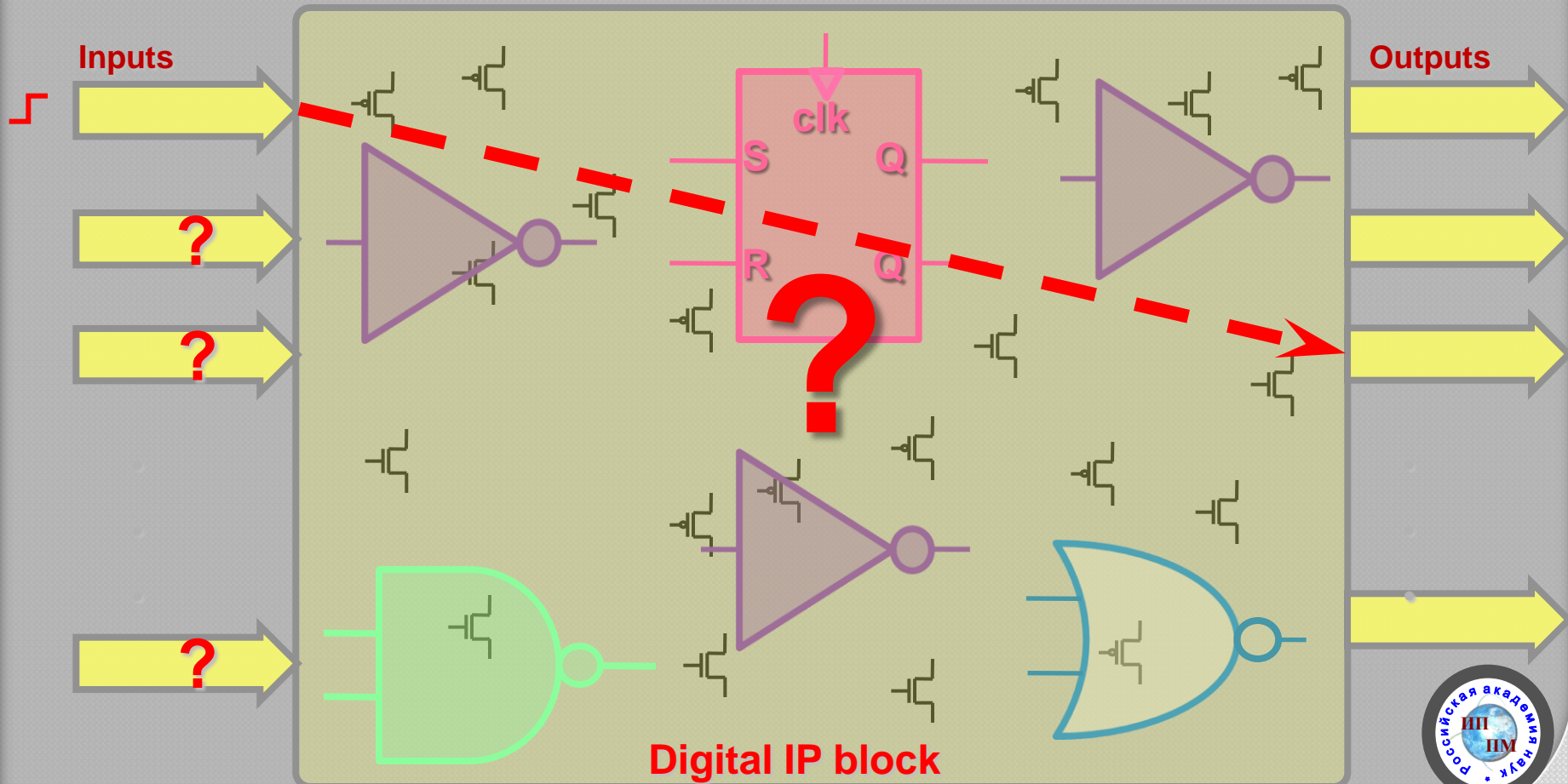
Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
- 6. Decomposition problems for IP blocks**
7. Input stimulus generation for IP blocks
8. Logic correlation analysis for timing and noise estimation
9. IP blocks characterization speed-up
10. Future technologies problems

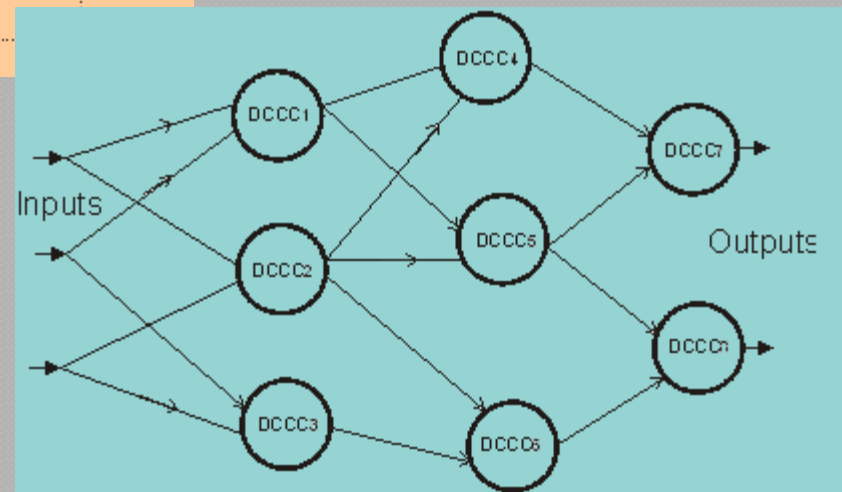
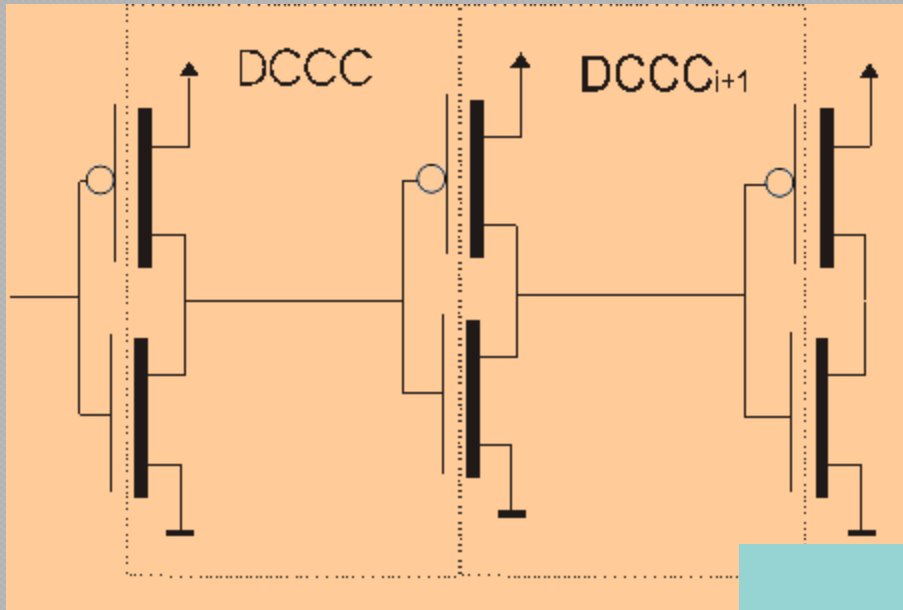


Characterization of the Full Custom IP block

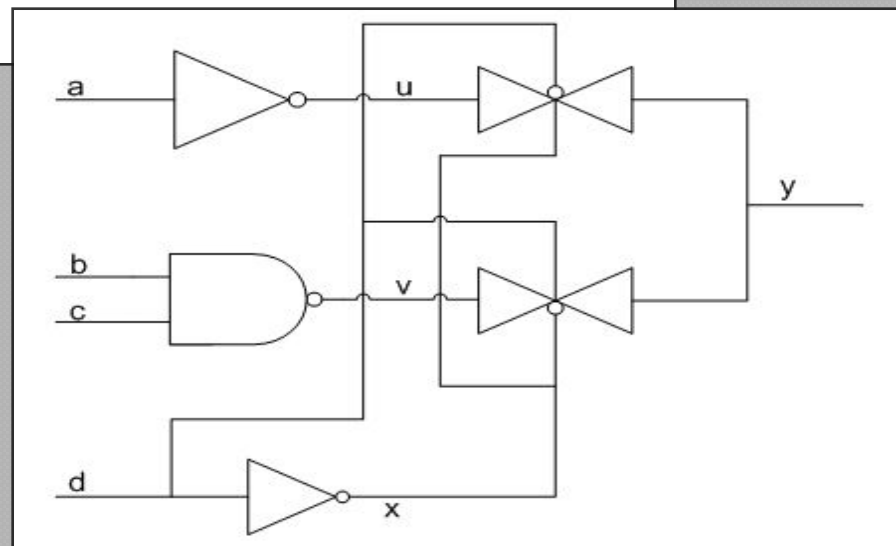
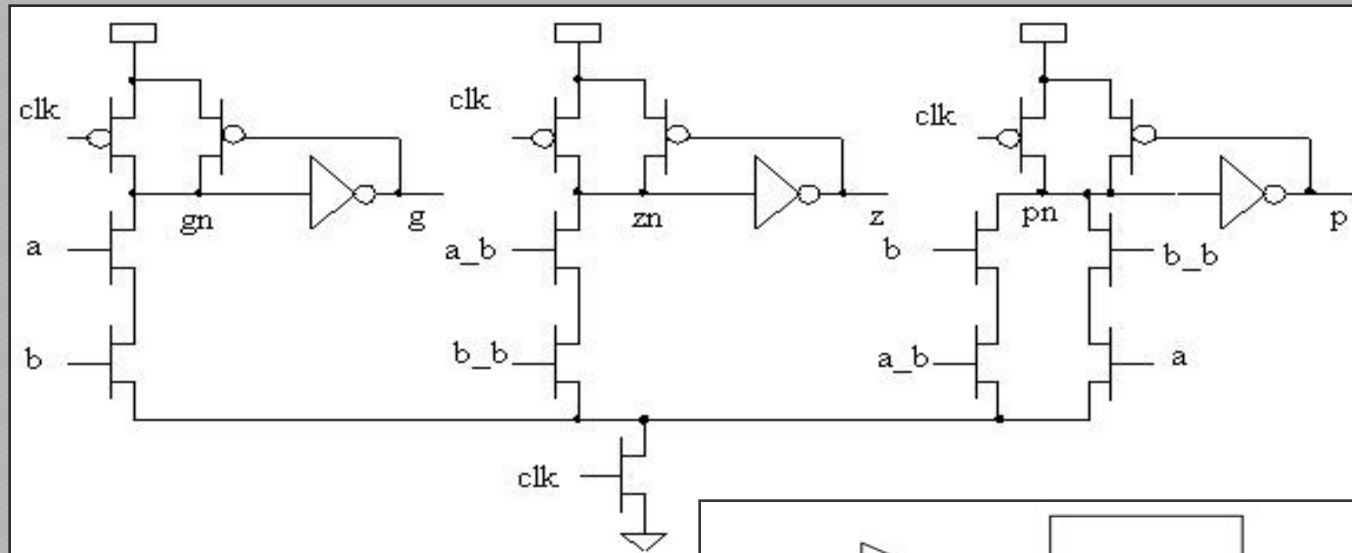
Find input stimulus for **maximal delay** from a given primary input to a given primary output



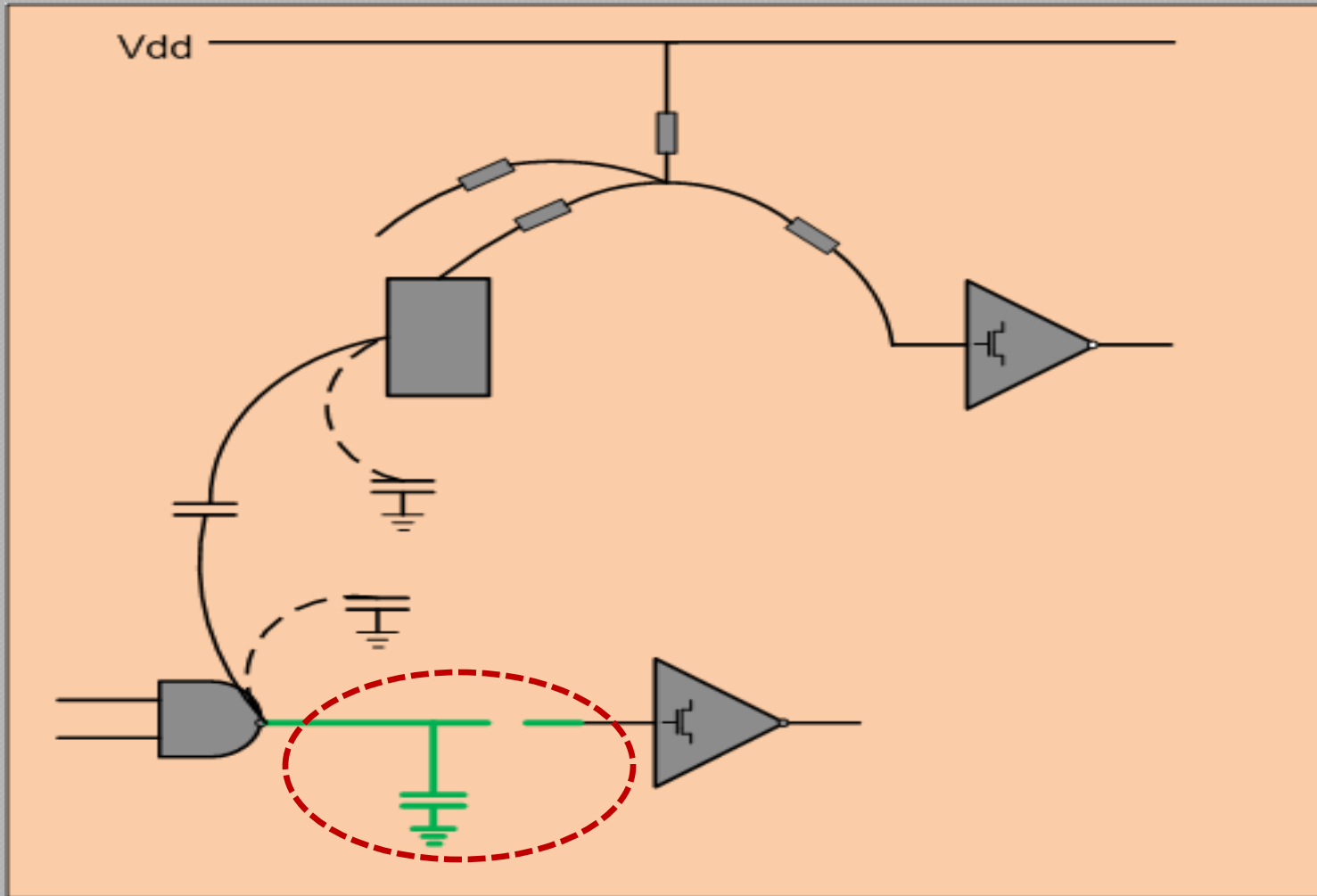
Decomposition Approach (DCCC = DSN = CCC...)



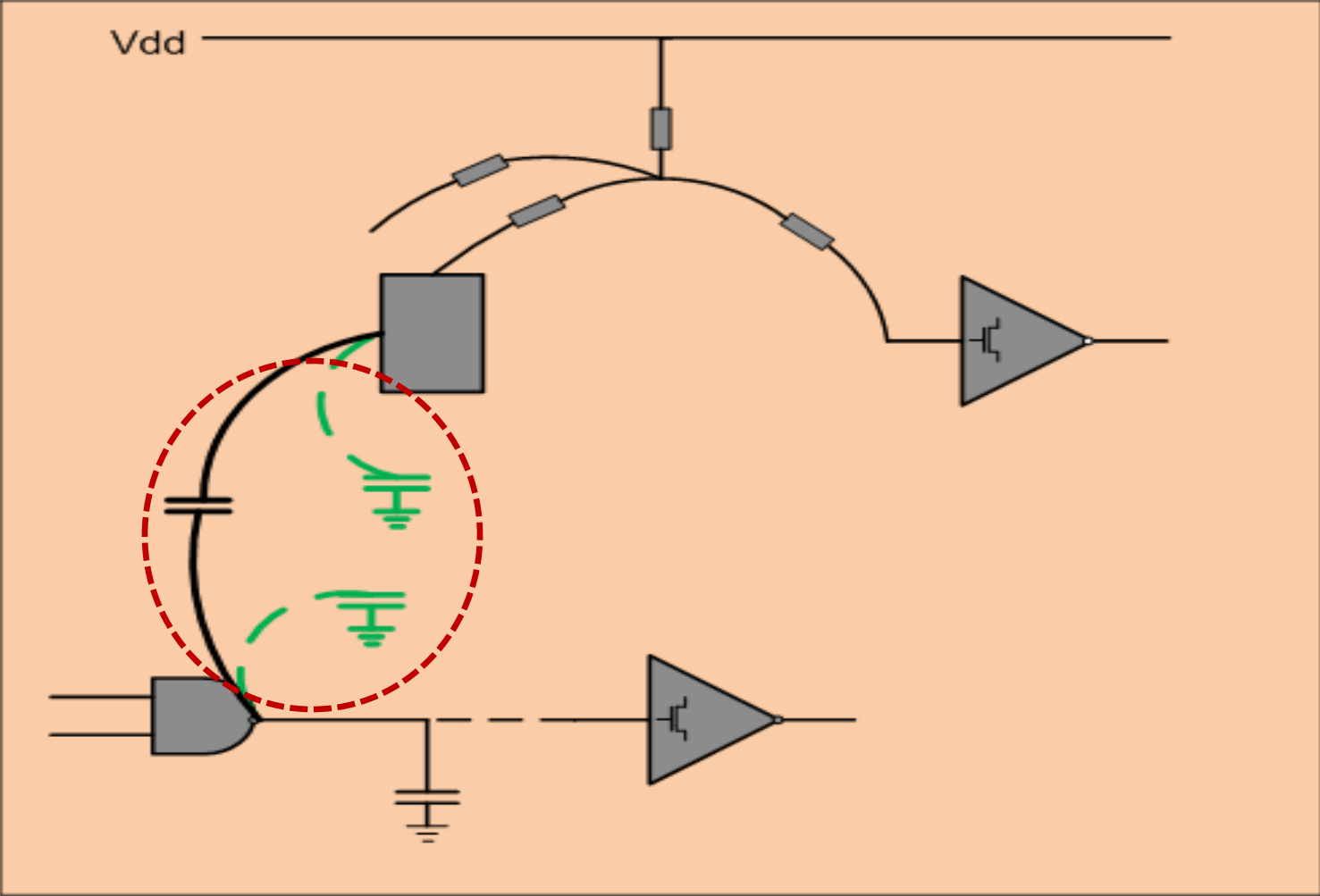
Decomposition for Full Custom IP-block: DCCC # Gate for Path Transistors & Domino Logic



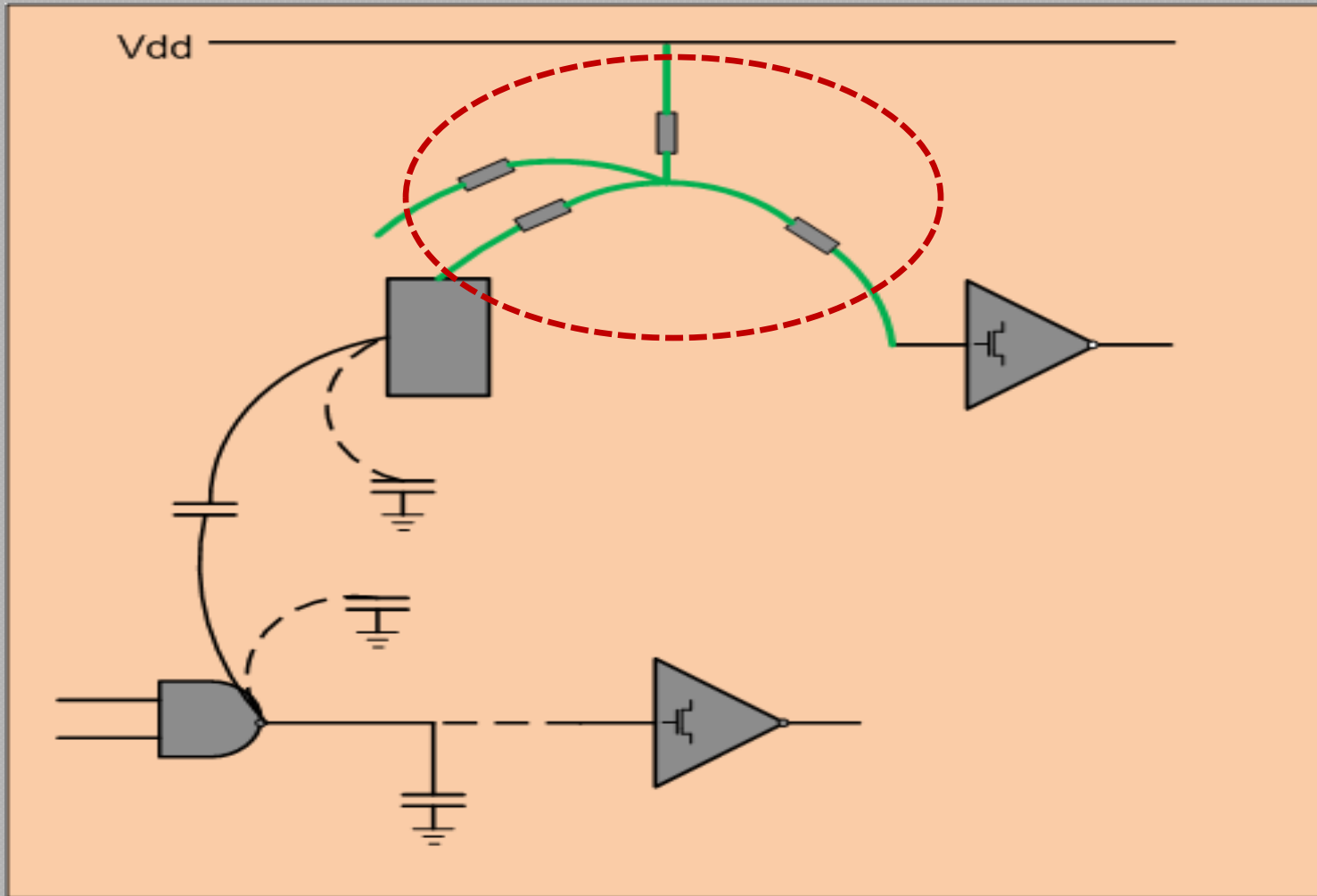
Decomposition Problem: (1) Cinp Error



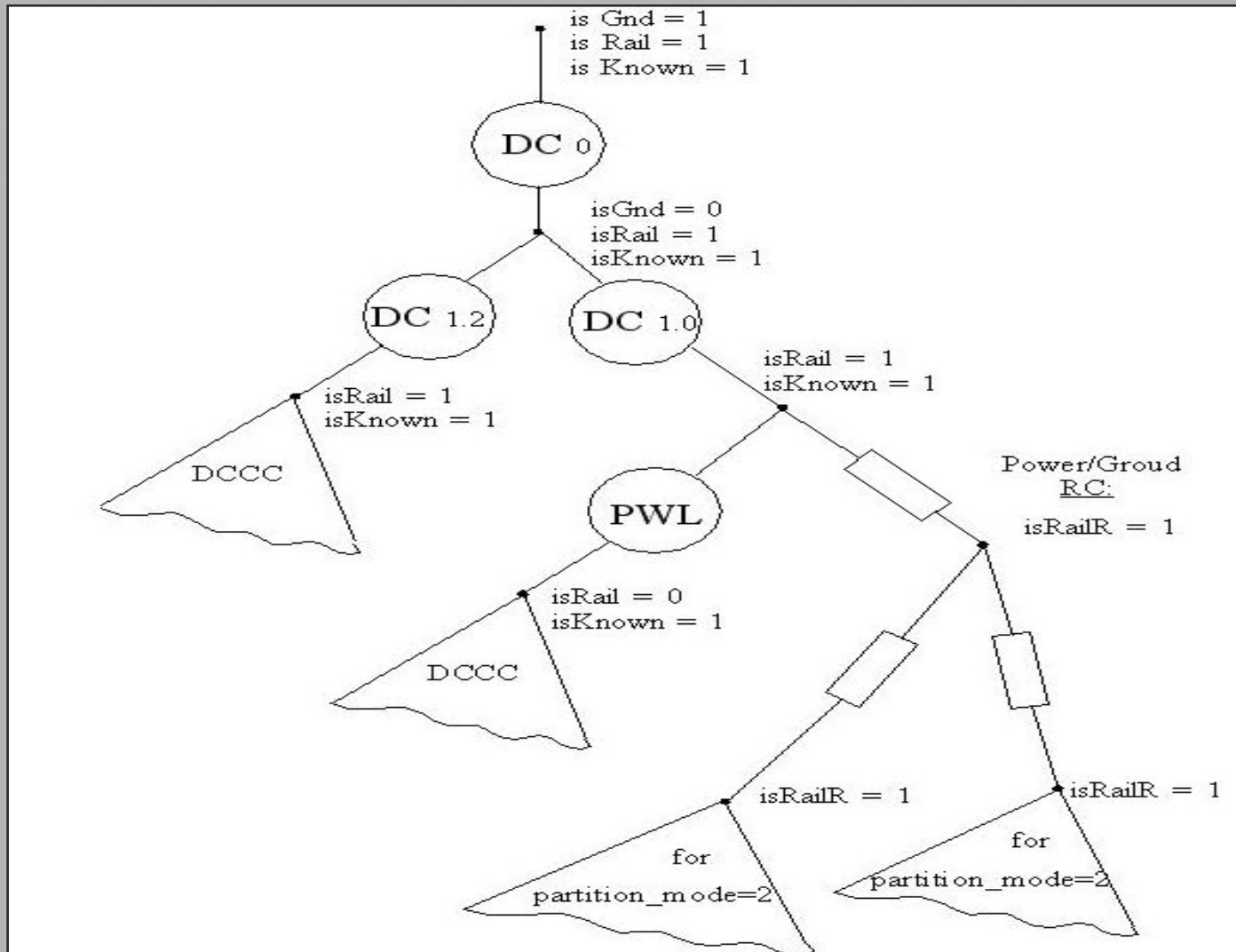
Decomposition Problem: (2) Coupling Cap Noise



Decomposition Problem: (3) IR-drop



Modified DCCC Decomposition



Decomposition and Correlations

- ✓ Delay: true path analysis - logic correlations results in false path
- ✓ Inputs stimulus for DCCC: correlations between DCCC inputs
- ✓ Coupling capacitances: correlations between aggressors and victim
- ✓ IR drop: max current estimation – correlations in different DCCC switching



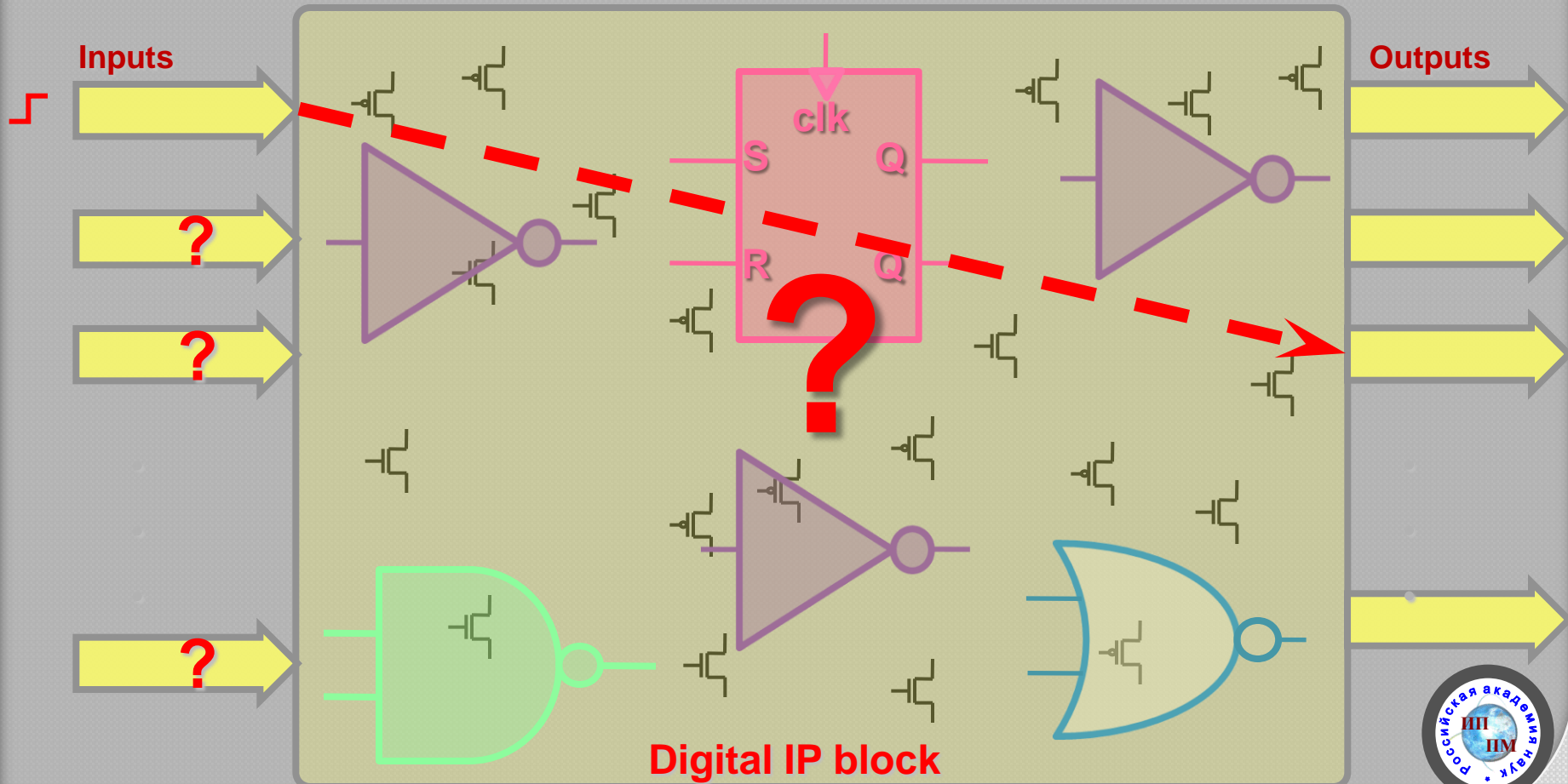
Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
- 7. Input stimulus generation for IP blocks**
8. Logic correlation analysis for timing and noise estimation
9. IP blocks characterization speed-up
10. Future technologies problems



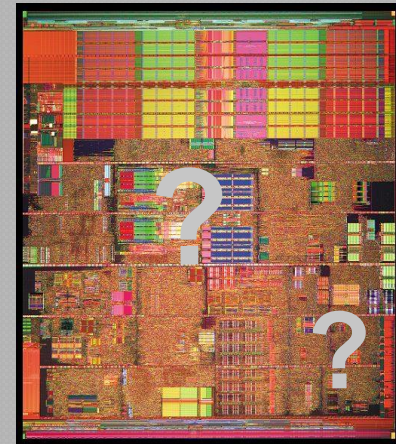
Input Stimulus Generation for the Full Custom IP

Find input stimulus for **maximal delay** from a given primary input to a given primary output



Timing Analysis for the Full Custom IP blocks

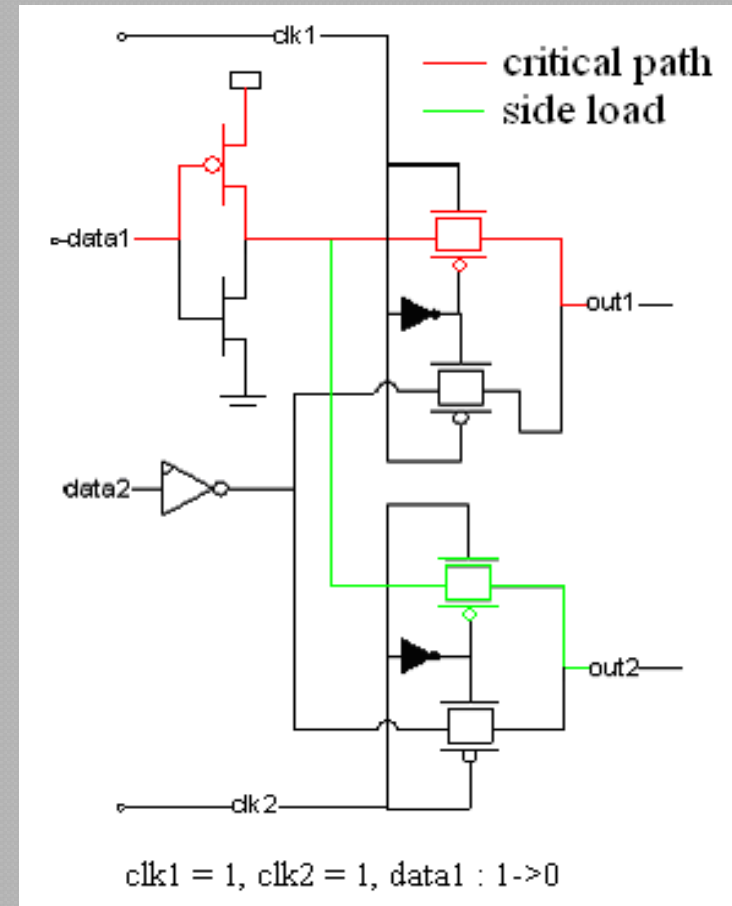
- Full custom IP block
 - Logic function of DCCC component is unknown
 - Library less analysis is required
 - Only transistor netlist is available
- Input stimulus search for maximal delay
 - Increasing number of DCCC inputs
 - Input logic correlations restrictions
 - Available methods:
full simulation; BDD / ADD approaches; critical path search



Critical Path and Side Load Conflict

✓ Critical path without side loads can be different from critical path with side loads

✓ Input logic correlation restrictions results in additional problems in critical path / side load analysis



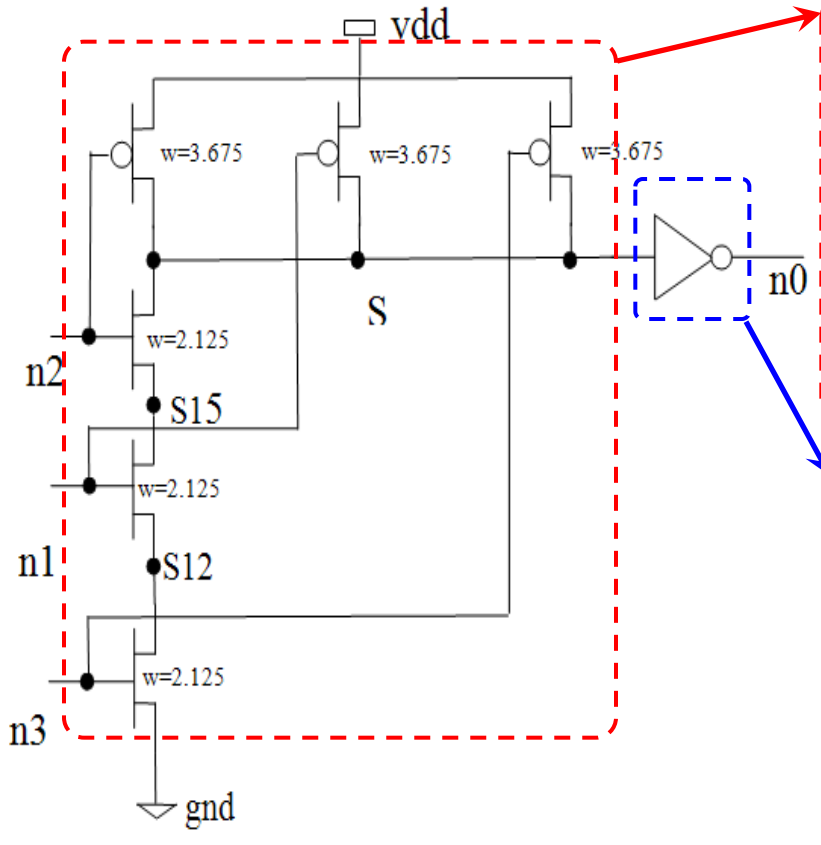
Stimulus Search: Proposed Approach:

- ✓ Input data should combine both logic and transistor data
- ✓ Generate *PU/PD-SP-DAG* for pull-up and pull-down networks from transistor netlist (logic extraction)
- ✓ Store the history of node resolutions “Resolution list”
- ✓ Form equivalent π -model in terms of Elmore delay.
- ✓ Delay analysis for particularly defined inputs
- ✓ Branch and bound approach for Max delay search

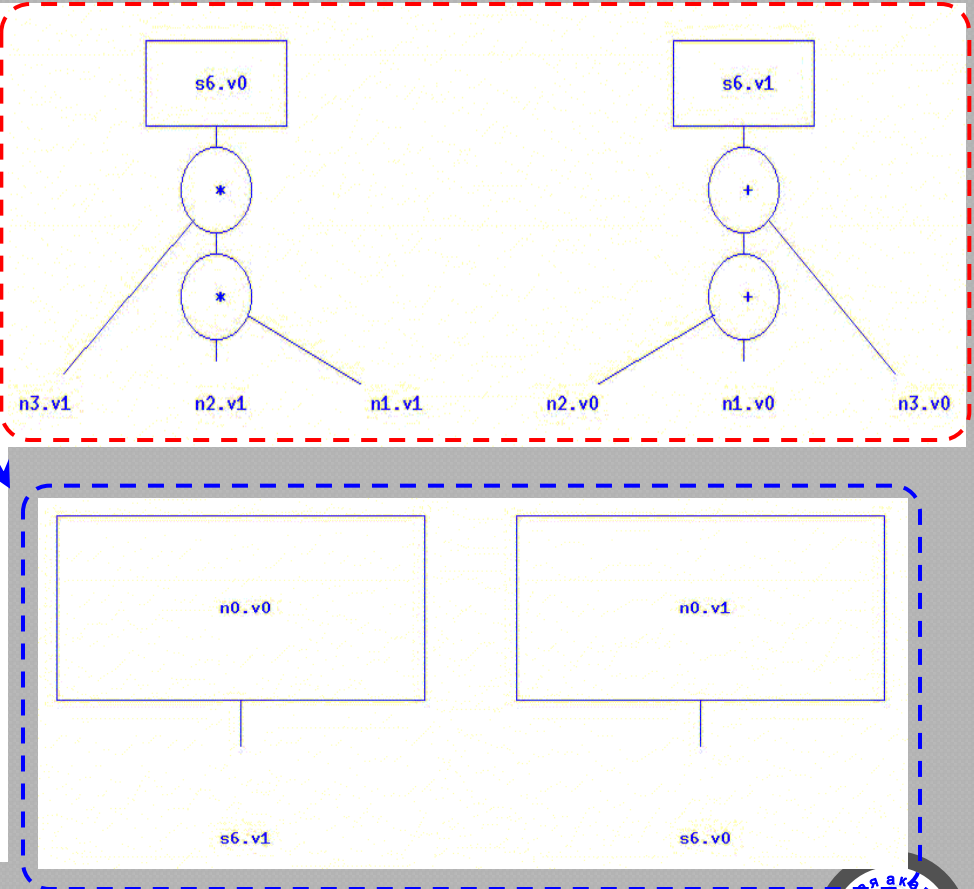


Logic Extraction and SP-DAG: CMOS element AND3

Transistor Circuit

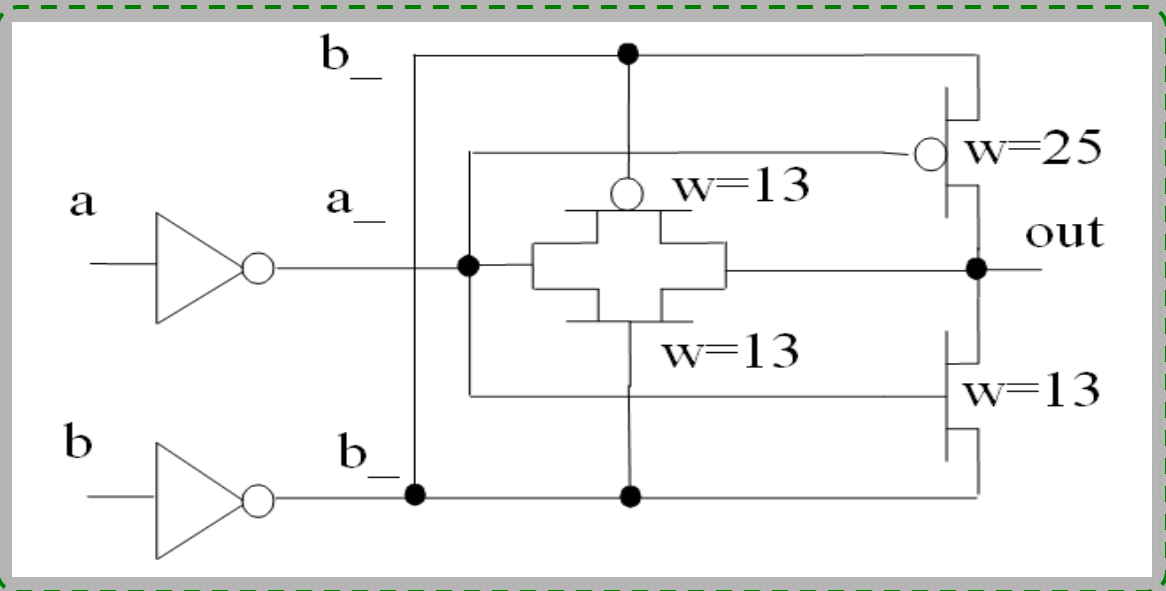


SP-DAG

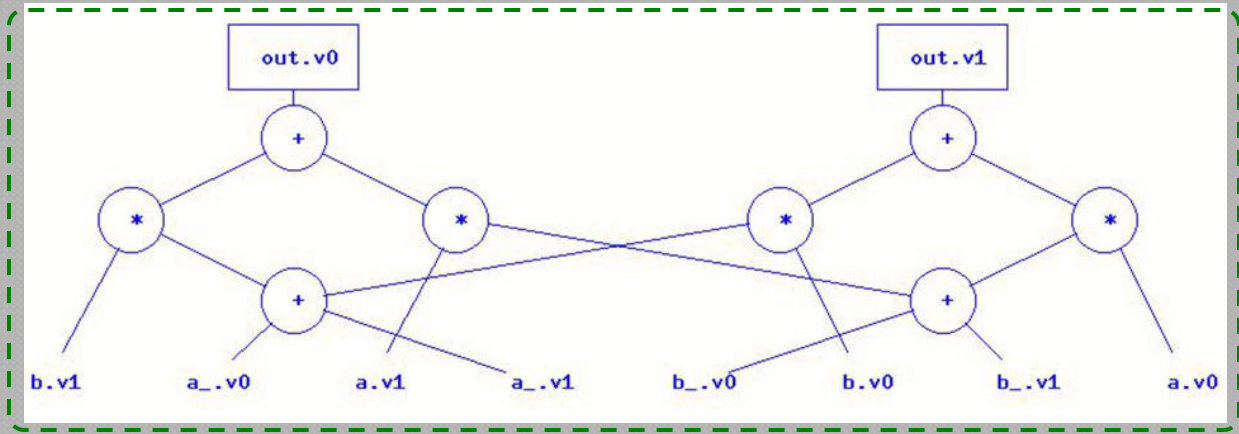


Logic Extraction and SP-DAG: CMOS element XOR2

Transistor Circuit

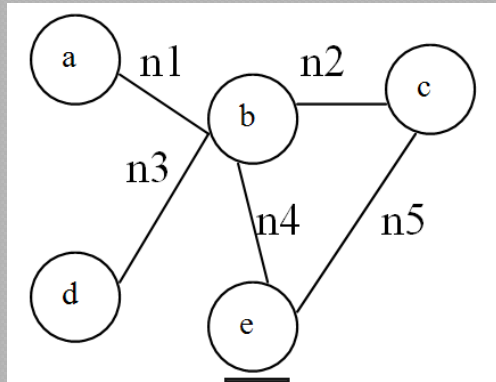


SP-DAG

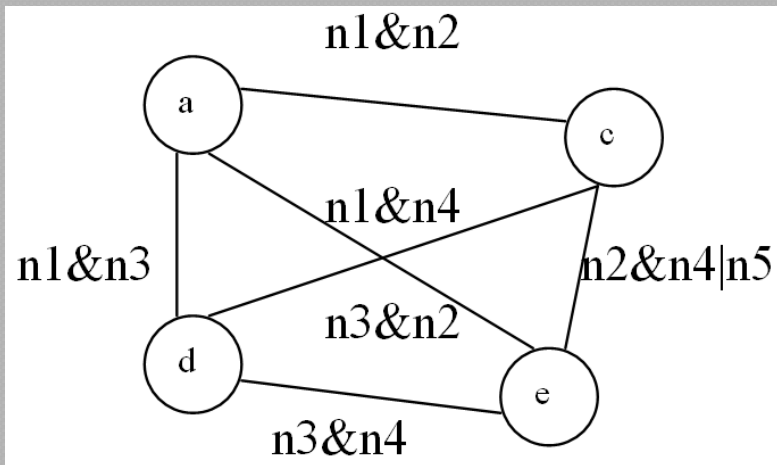


Gauss Elimination for non-SP Structure:

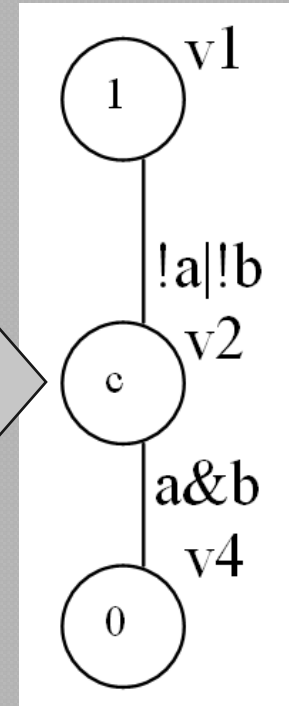
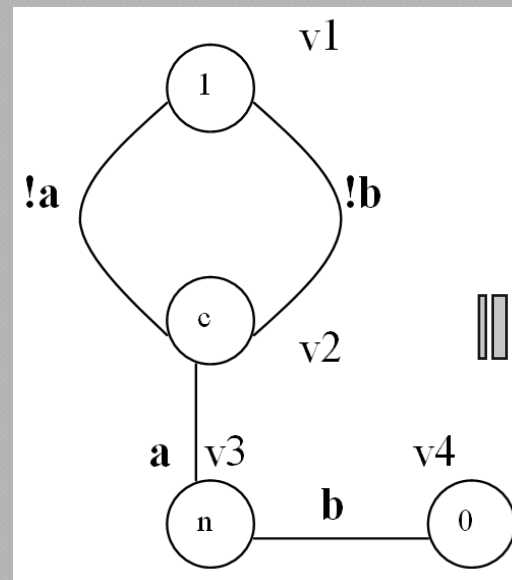
Before b resolution



After b resolution



NAND2



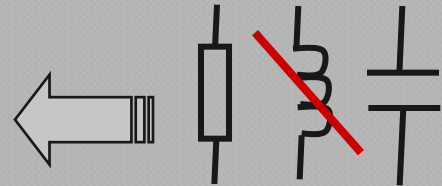
Pi-model in Gauss Elimination Approach (~Ticer)

$$Y_k = \sum_i y_{ki} = \frac{B_k}{s} + G_k + s \cdot C_k$$

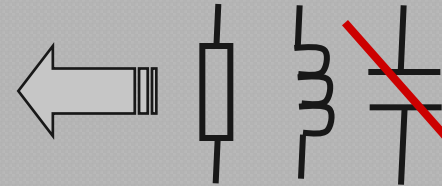
$$y_{ki} = \frac{1}{s} (b_i + s \cdot g_i + s^2 \cdot c_i) \quad y_{kj} = \frac{1}{s} (b_j + s \cdot g_j + s^2 \cdot c_j)$$

$$y_{ij} = \frac{1}{s^2 \cdot Y_k} \cdot (b_i b_j + s \cdot (b_i g_j + b_j g_i) + s^2 (g_i g_j + b_i c_j + b_j c_i) + s^3 (g_i c_j + g_j c_i) + s^4 c_i c_j)$$

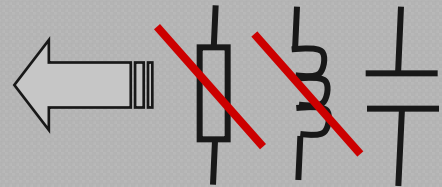
$$y_{ij} = \frac{1}{G_k} \cdot (g_i g_j + s(c_i g_j + c_j g_i) + \dots)$$



$$y_{ij} = \frac{1}{s \cdot B_k} \cdot (b_i b_j + s(b_i g_j + b_j g_i) + s^2 (g_i g_j + b_i c_j + b_j c_i) + \dots)$$



$$y_{ij} = \frac{1}{C_k} \cdot (c_i c_j)$$



Elmore Delay Estimation

Elmore delay:

$$d = C/g$$

C - equivalent ground capacitance;

$g = 1/R$ - equivalent internal conductance.

- ✓ Calculate logic states for internal and external nodes before and after switch;
- ✓ Calculate equivalent conductances for *pull-up* and *pull-down* networks;
- ✓ Calculate equivalent load capacitances for fall and rise switches
- ✓ Estimate switch delays (*fall delay*) and (*rise delay*).

$$\max(d) = \max(C) / \min(g)$$

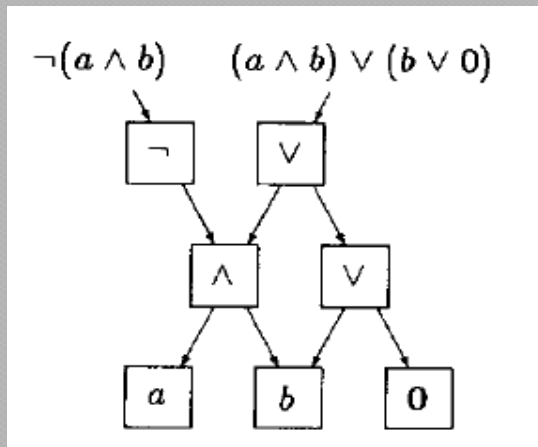


Prototypes vs PU/PD-SP-DAG

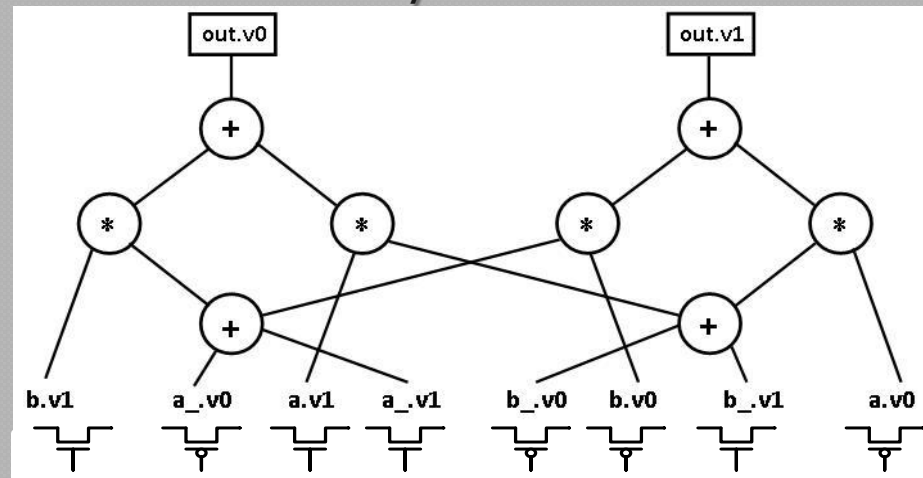
SP-DAG - [R.E. Bryant, Algorithmic Aspects of Symbolic Switch Network Analysis]

BDD - [R.E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation]

DAG-граф [Bryant]



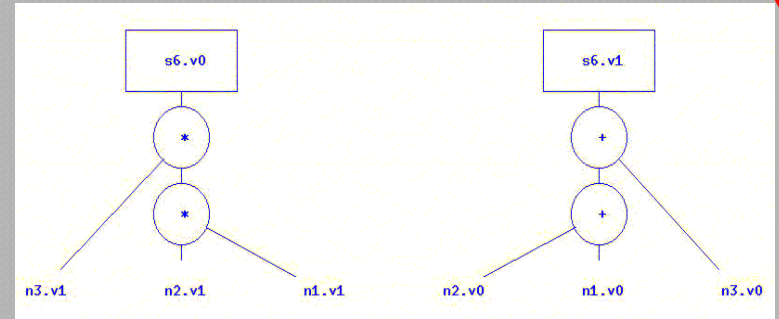
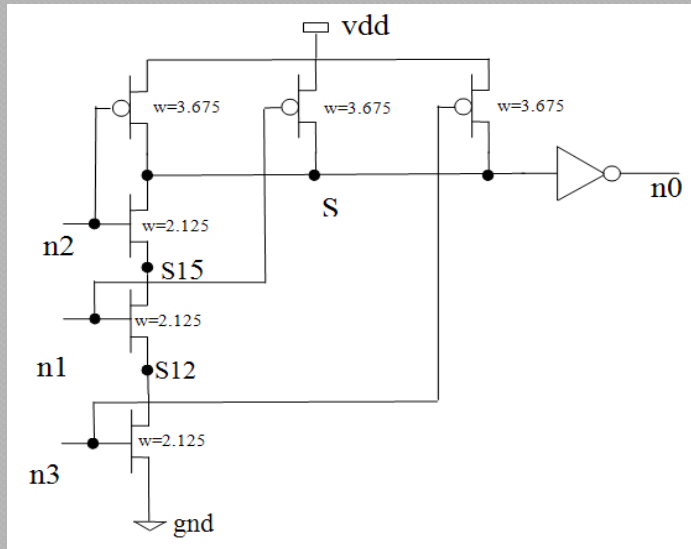
PU/PD-SP-DAG



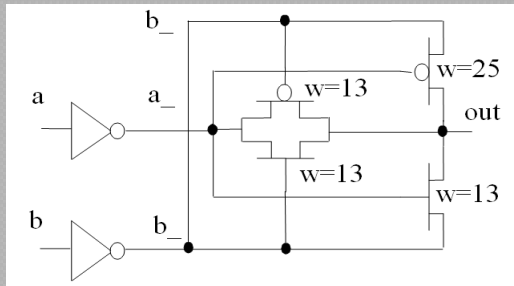
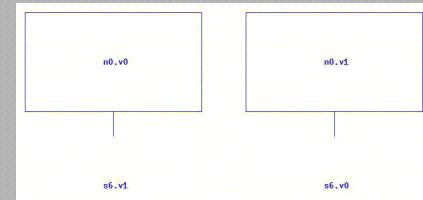
PU/PD-SP-DAG examples

Circuits

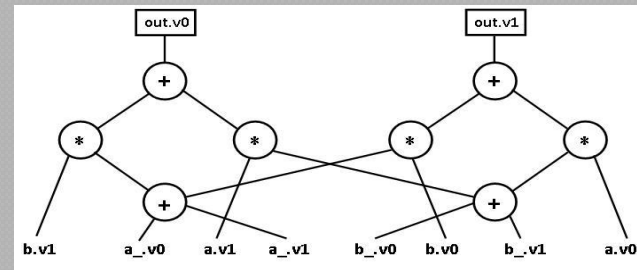
SP-DAG граф



AND3

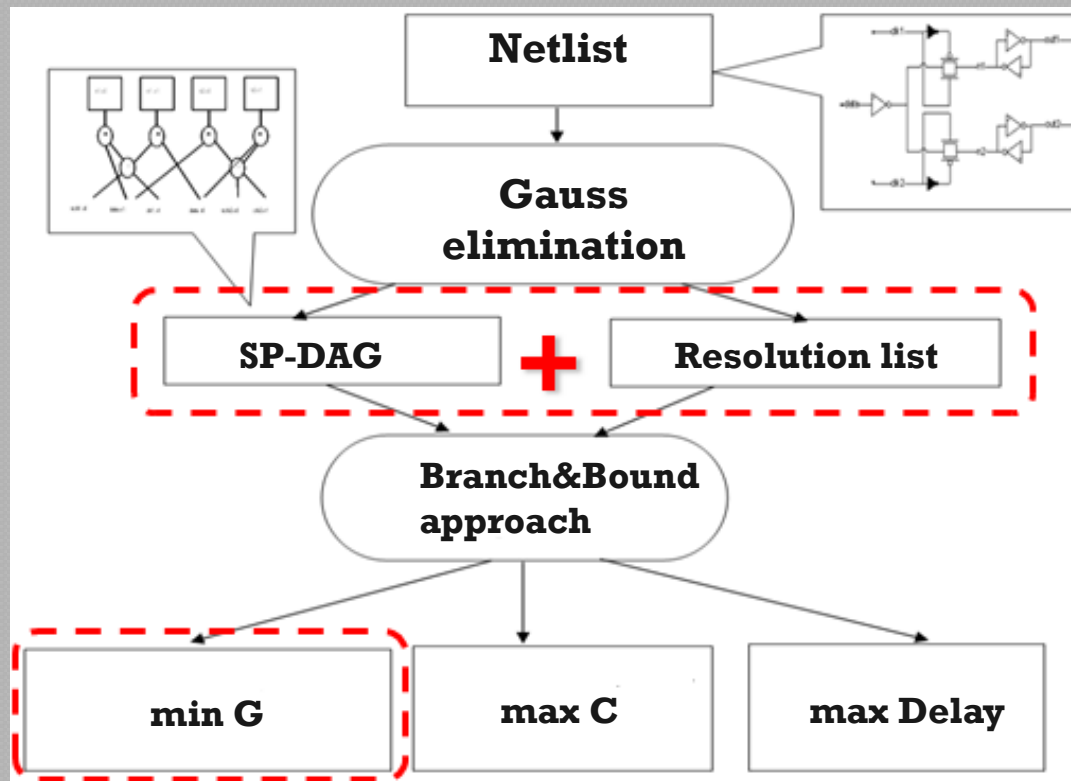


XNOR2



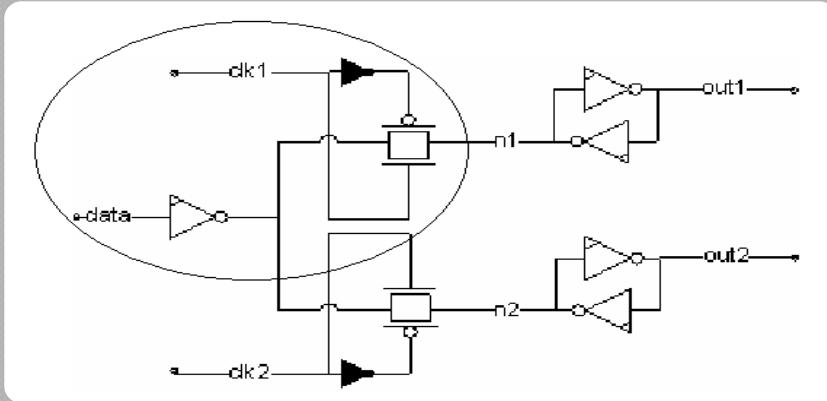
Input Stimulus Generation for Characterization

- ✓ Equivalent Pi-model estimation
- ✓ Max/Min estimation (min G, max C) $\max(D) = \max(C) / \min(g)$

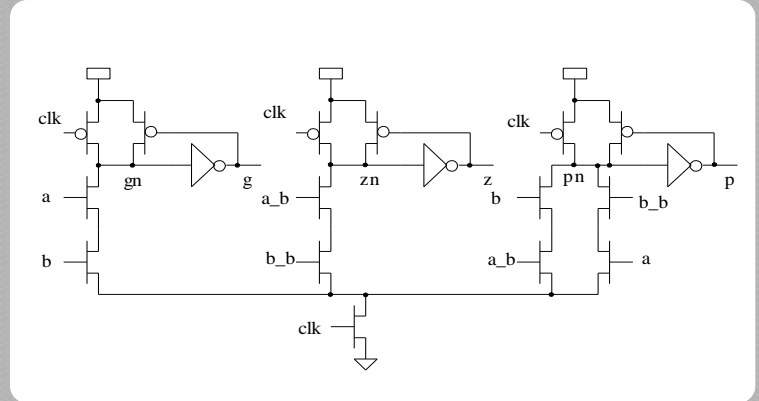


Example of Branch and Bound Search

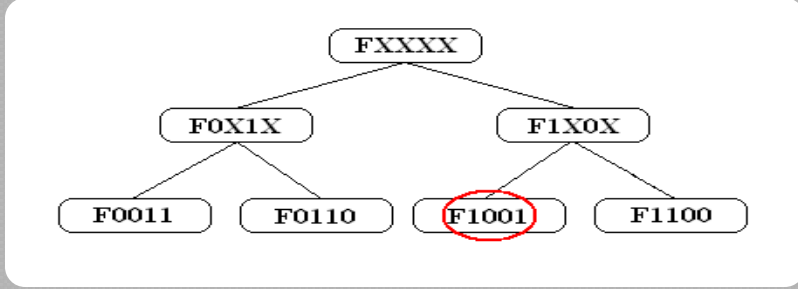
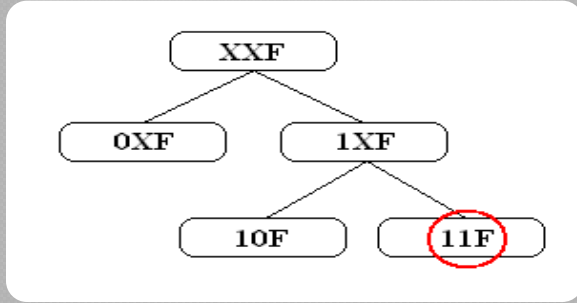
Account for input correlations



Inputs: clk₁, clk₂, data



Inputs: clk, a, b, a_b, b_b.
Logic restrictions: inverse a_b, inverse b_b



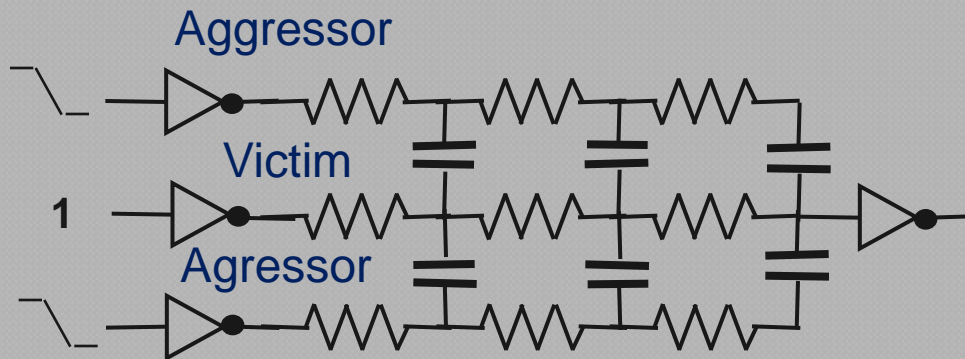
Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Input stimulus generation for IP blocks
- 8. Logic correlation analysis for timing and noise estimation**
9. IP blocks characterization speed-up
10. Future technologies problems

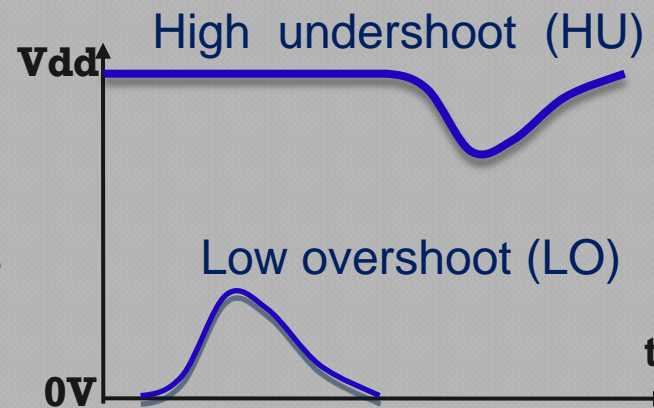


Cross Coupling Noise

- Aggressor nets affect victim net through coupling capacitances
- Functional Noise: changes logic state of the victim net
- Delay Noise: affects signal propagation delay
- Different types of functional noises:
 - victim state and aggressor switching direction
 - Low/High Overshoot/Undershoot

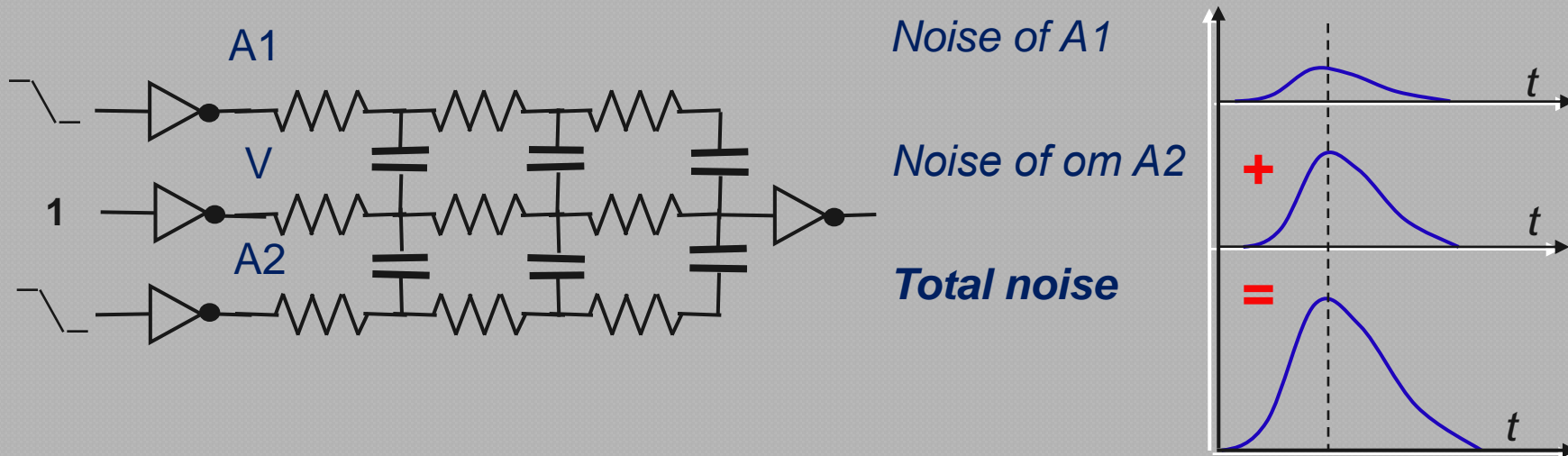


Noise cluster



Conservative Coupling Noise Analysis

- All aggressor nets switch simultaneously in the same direction
- All aggressor noises combine to create maximum noise
- Aggressors switching times align to inject maximum noise



- ✓ Ignores correlation between circuit signals and may overestimate noise
- ✓ May produce *false noise violations*
- ✓ New method to reduce false noise violations by using logic implications

Simple Logic Implication (SLI)

✓ Problem:

compute logic correlation & maximum realizable aggressors set

✓ Approach:

express logic correlation as simple logic implications

- Used in logic synthesis & peak current estimation [G. Hachtel 1988], [W.Kunz 1994], [R.I.Bahar 1996], [W.Long 2000], [S. Bobba 1998]

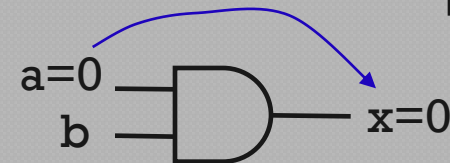
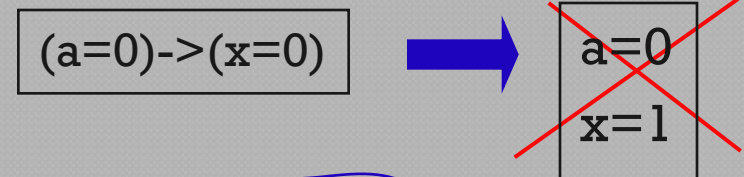
build constraint graph & find maximum realizable aggressors set

✓ SLI $(a=V_a) \rightarrow (x=V_x)$ means :

- if net a is at V_a then net x is at V_x

No timing information

Conservative only for glitch free circuits



SLI (Simple Logic Implication) Approach

Simple Logic Implication (SLI) for 2 nodes a, b :

$$(a = 0) \Rightarrow (b = 1)$$

Initial notation $(x = v) \Rightarrow (y = u)$

Short notation: $x^v \Rightarrow y^u$, where

x^v, y^u - are $(x, y$ or $\bar{x}, \bar{y})$:

$$x^v = \begin{cases} \bar{x} & \text{for } v = 0 \\ x & \text{for } v = 1 \end{cases}$$

SLI (Simple Logic Implication) Approach

Equivalent notations:

$$(a = 0) \Rightarrow (b = 0) \Leftrightarrow \bar{a} \Rightarrow \bar{b}$$

$$(a = 0) \Rightarrow (b = 1) \Leftrightarrow \bar{a} \Rightarrow b$$

$$(a = 1) \Rightarrow (b = 0) \Leftrightarrow a \Rightarrow \bar{b}$$

$$(a = 1) \Rightarrow (b = 1) \Leftrightarrow a \Rightarrow b$$

Implication set for a gate $y = \text{nand2}(a, b) = \overline{a \cdot b}$

$$(a = 0) \Rightarrow (y = 1) \Leftrightarrow \bar{a} \Rightarrow y$$

$$(b = 0) \Rightarrow (y = 1) \Leftrightarrow \bar{b} \Rightarrow y$$

$$(y = 0) \Rightarrow (a = 1) \Leftrightarrow \bar{y} \Rightarrow a$$

$$(y = 0) \Rightarrow (b = 1) \Leftrightarrow \bar{y} \Rightarrow b$$



SLI (Simple Logic Implication) Approach

Implication set for a gate $y = \text{nor2}(a,b) = \overline{a+b}$

$$(a = 1) \Rightarrow (y = 0) \quad \Leftrightarrow \quad a \Rightarrow \bar{y}$$

$$(b = 1) \Rightarrow (y = 0) \quad \Leftrightarrow \quad b \Rightarrow \bar{y}$$

$$(y = 1) \Rightarrow (a = 0) \quad \Leftrightarrow \quad y \Rightarrow \bar{a}$$

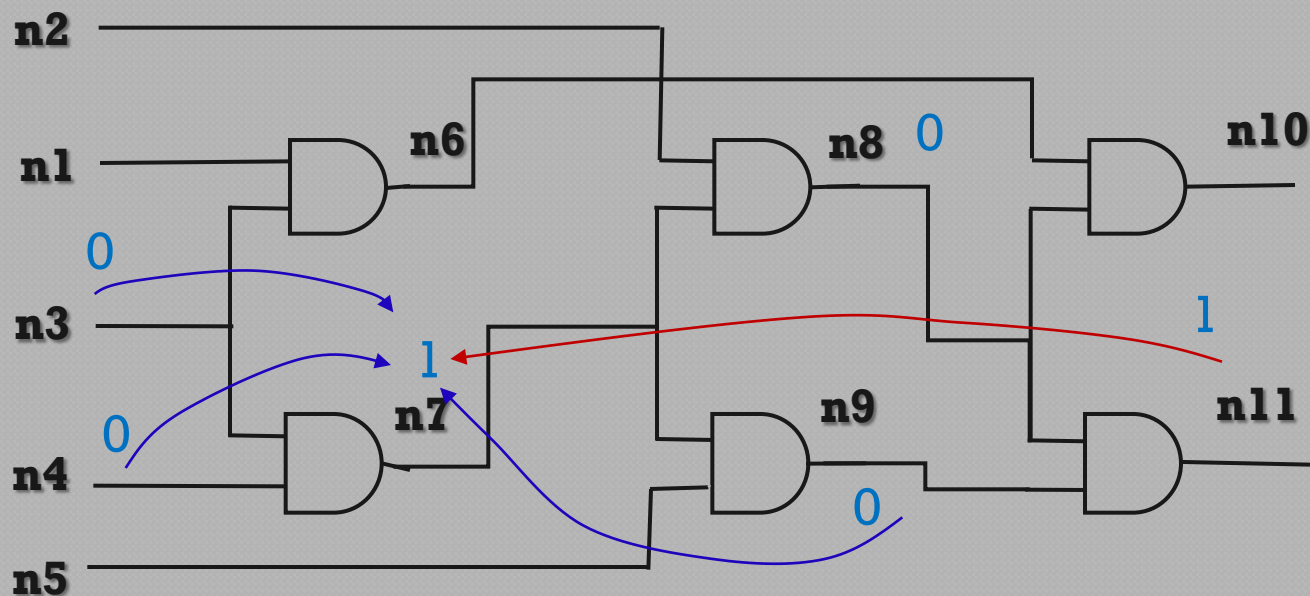
$$(y = 1) \Rightarrow (b = 0) \quad \Leftrightarrow \quad y \Rightarrow \bar{b}$$

Simple Logic Implication (SLI)

Compact representation used in implementation

4 implications lists $H^a_H, H^a_L, L^a_H, L^a_L$ for each circuit net a

implication list L^a_H consists of nets b_i such as SLI $(b_i = 1) \rightarrow (a = 0)$

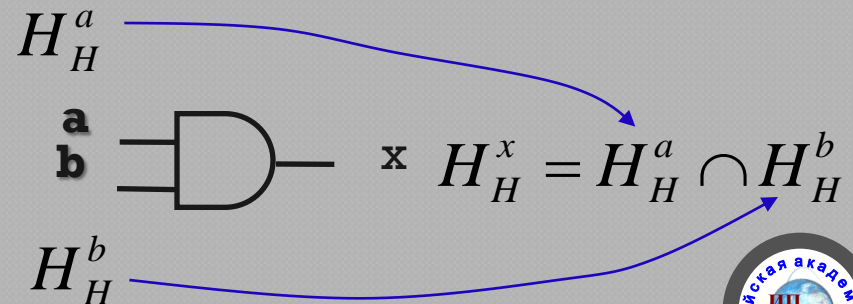
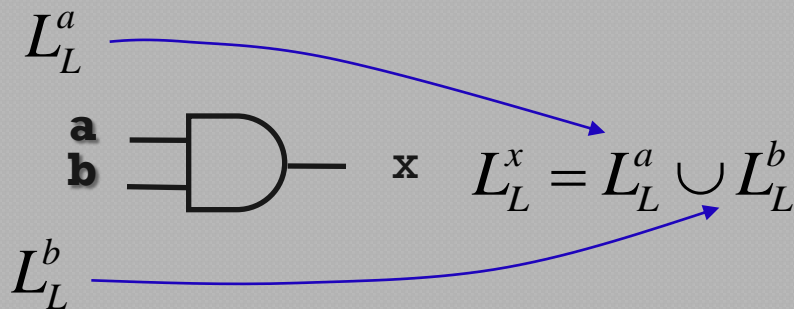


Implication lists: $H^{n7}_L = \{ n3, n4, n8, n9 \}$, $H^{n7}_H = \{ n11 \}$



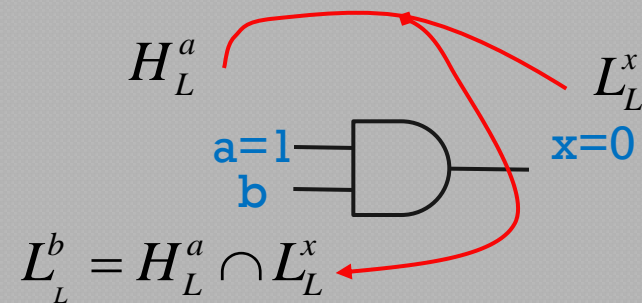
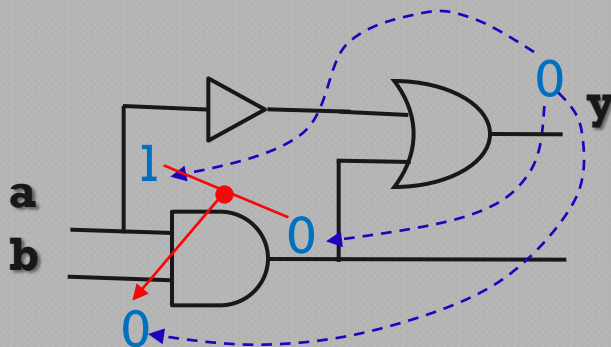
SLI Generation

- Compute SLIs for individual gate in circuit
- Propagate SLIs across the circuit
- Based on laws:
 - transitive: $(a = V_a) \rightarrow (b = V_b), (b = V_b) \rightarrow (c = V_c) \implies (a = V_a) \rightarrow (c = V_c)$
 - contra-positive: $(a = V_a) \rightarrow (x = V_x) \iff (x = \bar{V}_x) \rightarrow (a = \bar{V}_a)$
- Basic operations:
 - Implications lists union and intersection

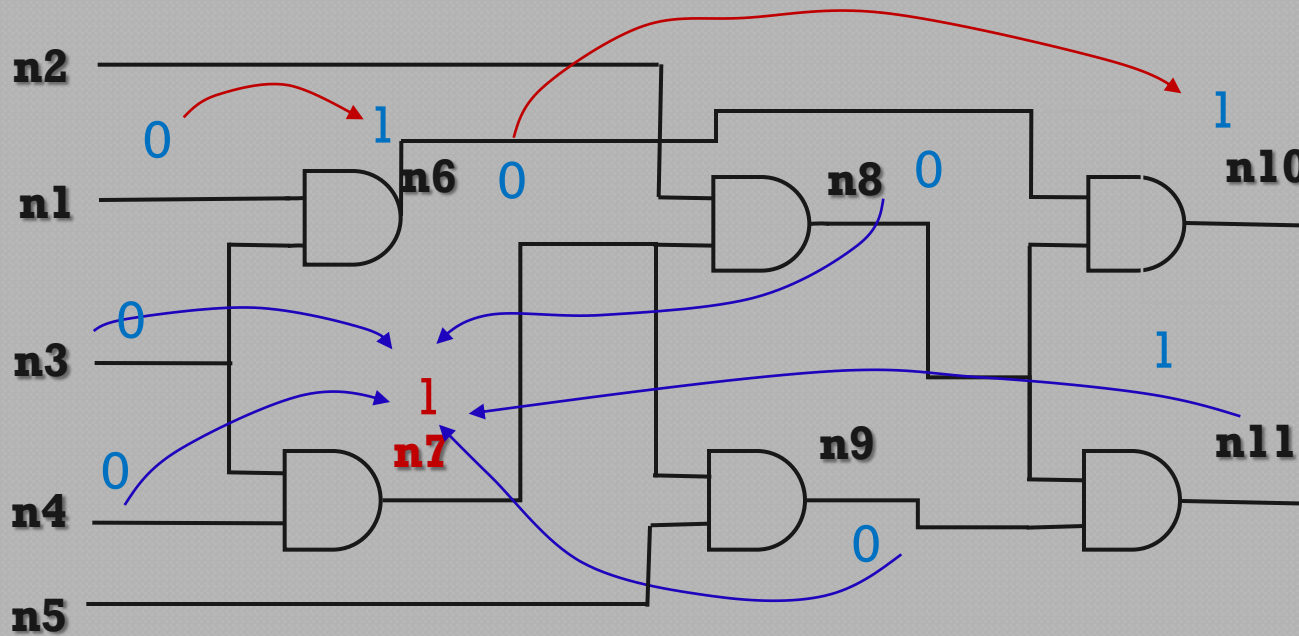


Lateral Propagation of SLI

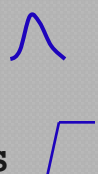

- ✓ Used in logic optimization [R.I.Bahar 1996], [W.Long 2000]
- ✓ Based on contra positive law
- ✓ AND gate:
 - implication $(a=1 \& x=0) \rightarrow b=0$
 - lateral implication lists propagation



Constraint Graph Construction from SLIs & MWIS Analysis

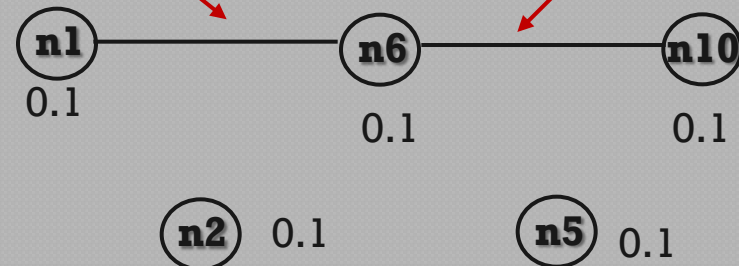


Noise cluster:

- victim: **n7**
- aggressors:
 - all others
- noise type: 
- victim at 0
- aggressors 

$(n1=0) \rightarrow (n6=1)$

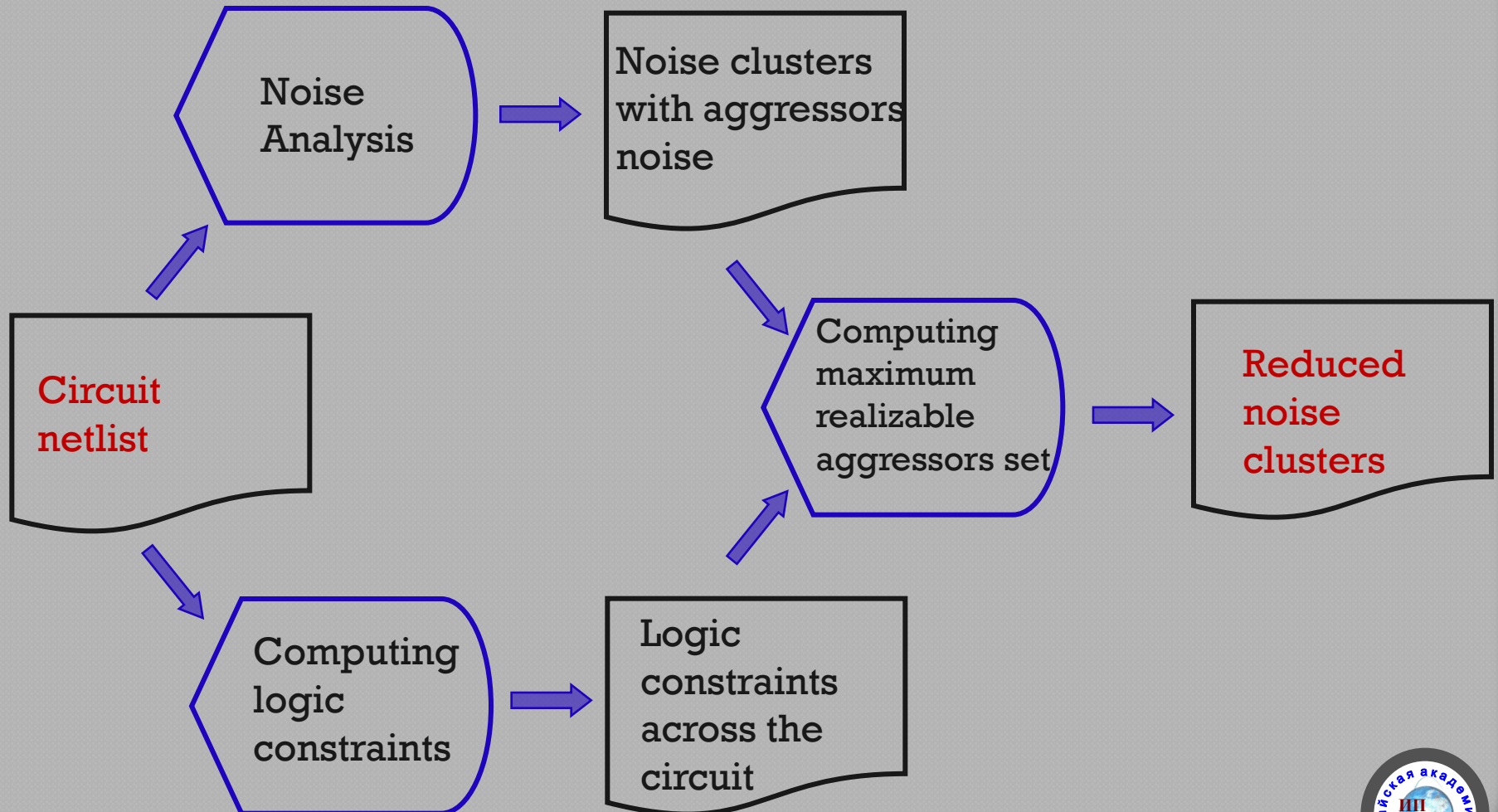
$(n6=0) \rightarrow (n10=1)$



•n3, n4, n8, n9, n11 are excluded because of victim/aggressor SLIs



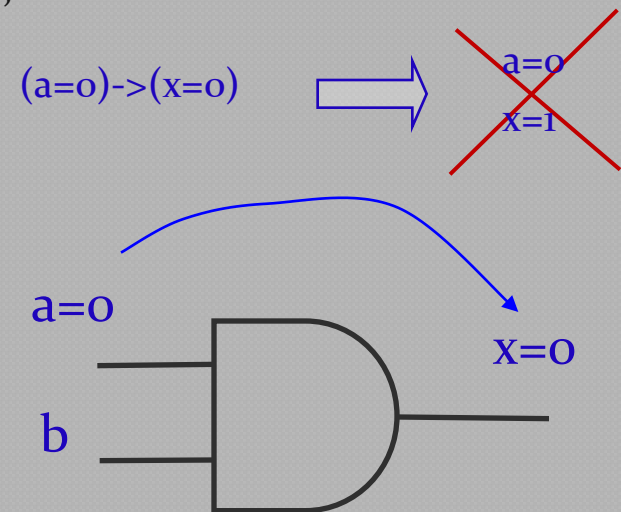
False Noise Analysis Data Flow



SLI and Other SAT Solutions Problems

- ✓ Logic correlation
 - SAT - problem
 - NP – complete
- ✓ Full analysis [A. Rubio, et al. 1997], [P. Chen, K. Keutzer, 1999]
 - For circuits ~100-300 nodes
- ✓ SLI heuristic approach [A. Glebov, S.Gavrilov, et al]
 - Fast but not full
 - Pair wise correlations only
 - Ignore 3-, 4- etc. correlations
 - Logic extraction is required

SLI:



Logic Constraints Representation

- System of equations or DNF:

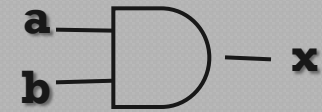
$$\left. \begin{array}{l} \bar{a} \cdot \bar{b} \cdot \bar{c} = 0 \\ \bar{a} \cdot \bar{b} \cdot d = 0 \\ \vdots \\ \vdots \\ \vdots \end{array} \right\} \iff \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot d + \dots = 0$$

✓ Set of conjunctive terms: $\bar{a} \cdot \bar{b} \cdot \bar{c}, \bar{a} \cdot \bar{b} \cdot d \dots$

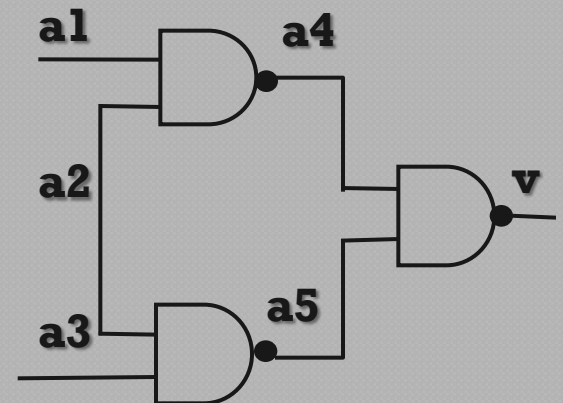
- Each conjunctive term prohibits one signal combination

- Term: $a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} = 0$

prohibits: $a=1, b=0, c=0, d=0$ and prohibits signals b, c, d from simultaneous switching if $a=1$



$$\bar{x} \cdot a \cdot b + x \cdot \bar{a} + x \cdot \bar{b} = 0$$



Constraints for low overshoot noise at v:

$$\begin{aligned} & \bar{v} \cdot \bar{a}_4, \bar{v} \cdot \bar{a}_5, \\ & \bar{a}_1 \cdot \bar{a}_4, \bar{a}_2 \cdot \bar{a}_4, \bar{a}_2 \cdot \bar{a}_5, \bar{a}_3 \cdot \bar{a}_5, \\ & a_1 \cdot a_2 \cdot a_4, a_2 \cdot a_3 \cdot a_5 \end{aligned}$$



Resolution Method

- Automatic theorem proving and SAT problem:
 - deriving new logic relations by **Resolution Rule**:

$$a + B = 1, \bar{a} + C = 1 \quad \longrightarrow \quad B + C = 1$$

$$\text{или } a + B, \bar{a} + C \quad \longrightarrow \quad B + C$$

- Resolution rule for logic constraints
 - constraints (false sentences) derivation

$$a \cdot B = 0, \bar{a} \cdot C = 0 \quad \longrightarrow \quad B \cdot C = 0$$

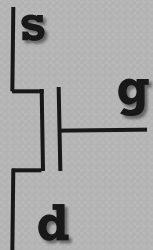
or

$$a \cdot B, \bar{a} \cdot C \quad \longrightarrow \quad B \cdot C$$

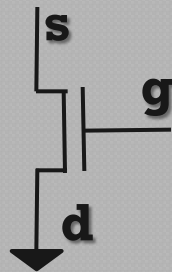


Transistor Level Logic Constraints Generation

- Initial logic constraints for transistors:



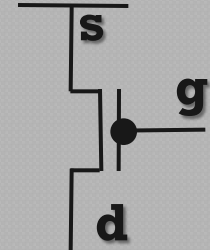
$$g \cdot s \cdot \bar{d}, g \cdot \bar{s} \cdot d$$



$$g \cdot s$$



$$\bar{g} \cdot s \cdot \bar{d}, g \cdot \bar{s} \cdot d$$

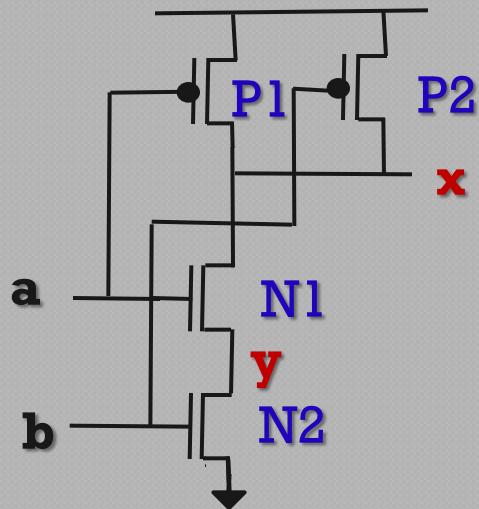


$$\bar{g} \cdot \bar{d}$$

- Deriving constraints for DCCCs (gates) at transistor level
 - compute constraints by resolution rule
 - try to eliminate variables not involved in noise clusters
 - remove tautologies: $a \cdot a \cdot B$
 - remove constraints covered by other ones: $(a \cdot \bar{b} \text{ covers } a \cdot \bar{b} \cdot \bar{c})$

Logic Constraints Derivation

Logic constraints for static NAND2



$$P1: \overline{a \cdot x}$$

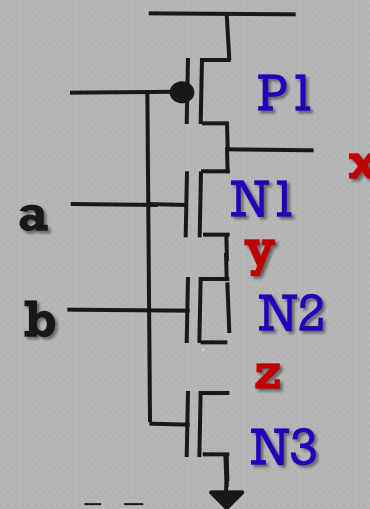
$$P2: \overline{b \cdot x}$$

$$N1: a \cdot \overline{y} \cdot x, a \cdot y \cdot \overline{x}$$

$$N2: b \cdot y$$

$$b \cdot y, a \cdot x \cdot \overline{y} \rightarrow a \cdot b \cdot x$$

Logic constraints for dynamic NAND2



$$P1: \overline{c \cdot x}$$

$$N1: a \cdot \overline{y} \cdot x, a \cdot y \cdot \overline{x}$$

$$N2: b \cdot \overline{y} \cdot z, b \cdot y \cdot \overline{z}$$

$$N3: c \cdot z$$

$$c \cdot z, b \cdot y \cdot \overline{z} \rightarrow c \cdot b \cdot y$$

$$c \cdot b \cdot y, a \cdot x \cdot \overline{y} \rightarrow c \cdot a \cdot b \cdot x$$



Constraints Derivation at Logic Level

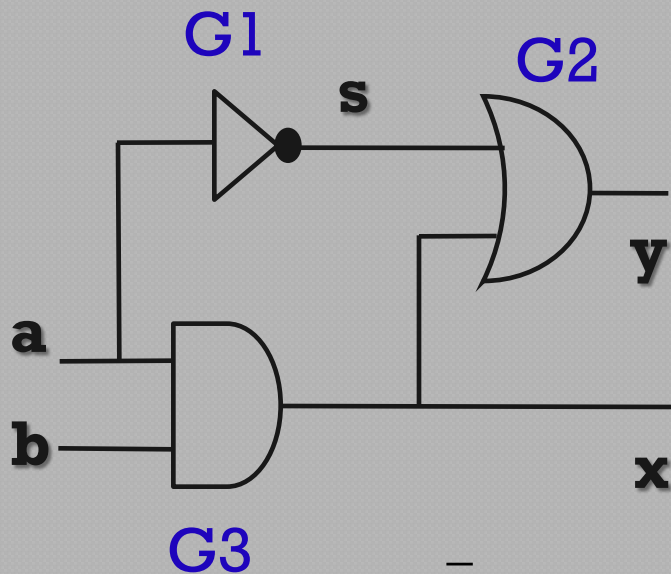
$$\overline{a \cdot s}$$

$$a \cdot s$$

$$\overline{x \cdot s \cdot y}$$

$$x \cdot \overline{y}$$

$$s \cdot y$$



$$\bullet \text{G1,G2: } \overline{a \cdot s}, \overline{x \cdot s \cdot y} \rightarrow \overline{a \cdot x} \cdot y \quad (\text{R1})$$

$$\overline{a \cdot s}, s \cdot y \rightarrow \overline{a \cdot y}$$

$$\bullet \text{G3,G2: } a \cdot b \cdot \overline{x}, x \cdot \overline{y} \rightarrow a \cdot b \cdot \overline{y} \quad (\text{R2})$$

$$\bullet \text{R1,R2: } \overline{a \cdot y}, a \cdot b \cdot \overline{y} \rightarrow \overline{y} \cdot b \cdot \overline{y} \quad (\text{R3})$$

$$\overline{y} \cdot b \cdot \overline{y} \rightarrow b \cdot \overline{y}$$

$$a \cdot b \cdot \overline{x}$$

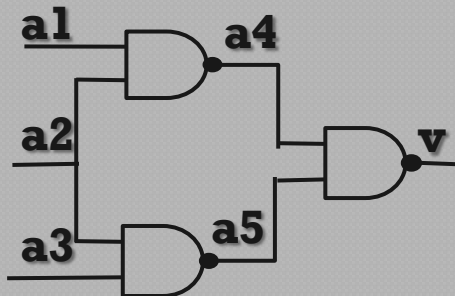
$$\overline{a \cdot x}$$

$$\overline{b \cdot x}$$

Characteristic ROBDD Construction

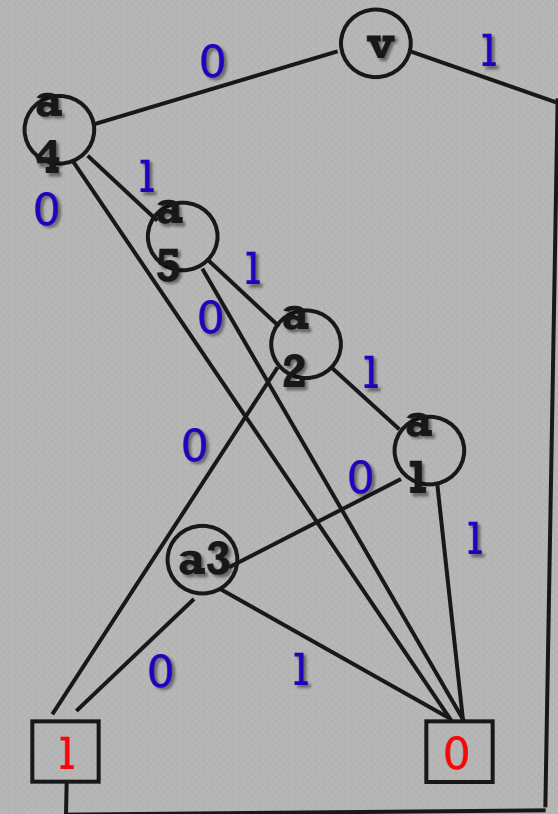
- ✓ Create root for victim
- ✓ Try $v=0,1$ assignments
- ✓ Make all conclusions from constraints
 - if constraints are satisfied create arc to 1
 - if constraints are at conflict create arc to 0
 - otherwise create arc to next aggressor vertex and repeat the analysis
- ✓ Repeat the procedure for aggressors

Low overshoot noise at v:



Constraints for low overshoot noise at v:

$$\begin{aligned} & \bar{v} \cdot \bar{a}_4, \bar{v} \cdot \bar{a}_5, \\ & \bar{a}_1 \cdot \bar{a}_4, \bar{a}_2 \cdot \bar{a}_4, \bar{a}_2 \cdot \bar{a}_5, \bar{a}_3 \cdot \bar{a}_5, \\ & a_1 \cdot a_2 \cdot a_4, a_2 \cdot a_3 \cdot a_5 \end{aligned}$$



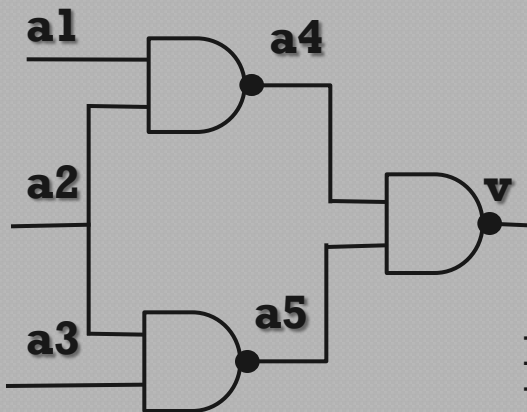
Maximum Realizable Noise Calculation

- ✓ Mark ROBDD vertices with noise value
- ✓ Find maximum aggressors set (a_{i1}, a_{i2}, \dots)
 - ROBDD has pair of paths to vertex 1

$(v=V, a_{i1}=0, a_{i2}=0, \dots)$ and

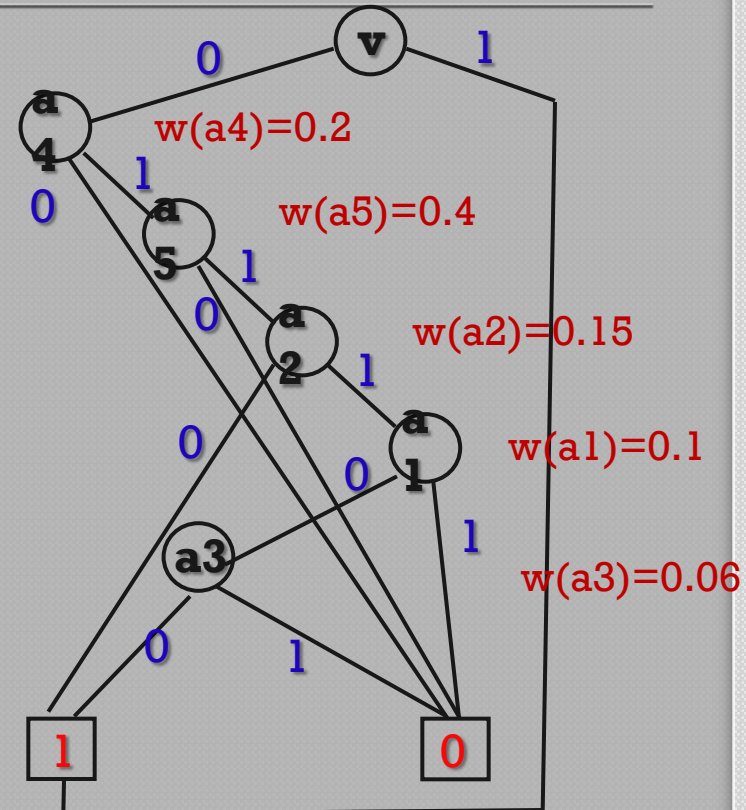
$(v=V, a_{i1}=1, a_{i2}=1, \dots)$

Low overshoot noise at v :



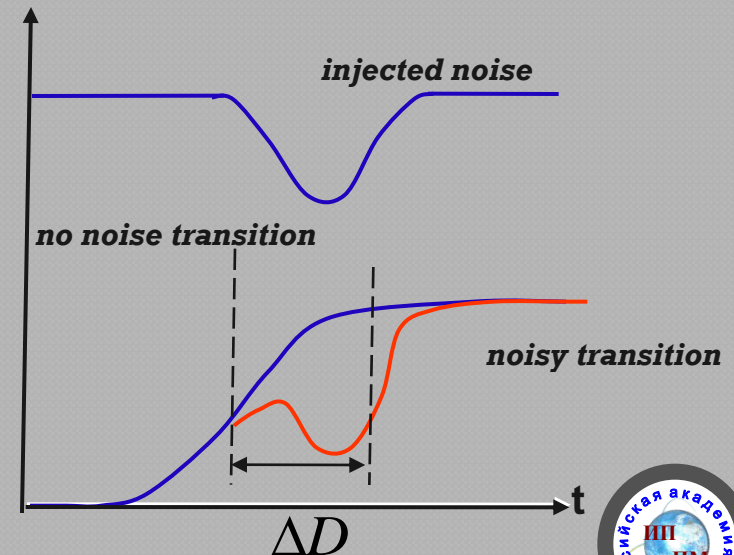
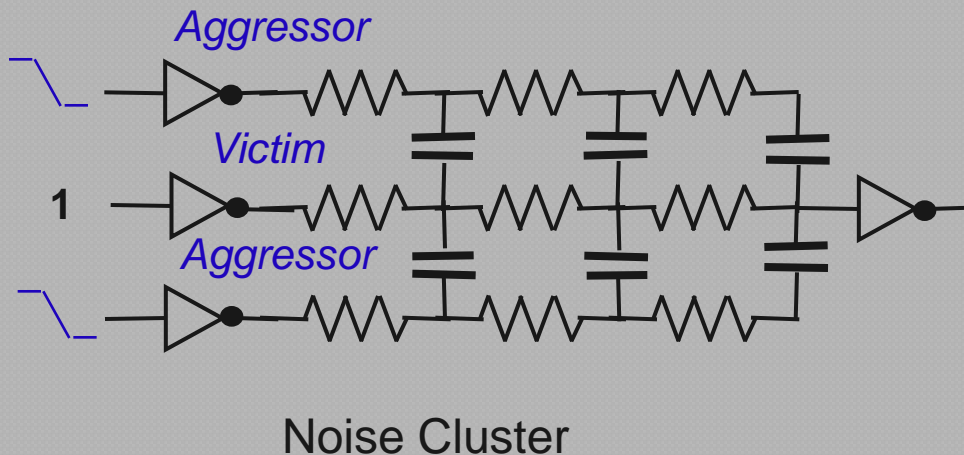
$v \cdot a_4, v \cdot a_5,$
 $a_1 \cdot a_4, a_2 \cdot a_4, a_2 \cdot a_5, a_3 \cdot a_5,$
 $a_1 \cdot a_2 \cdot a_4, a_2 \cdot a_3 \cdot a_5$

Maximum realizable aggressor set: a_1, a_3
 Maximum realizable noise 0.16



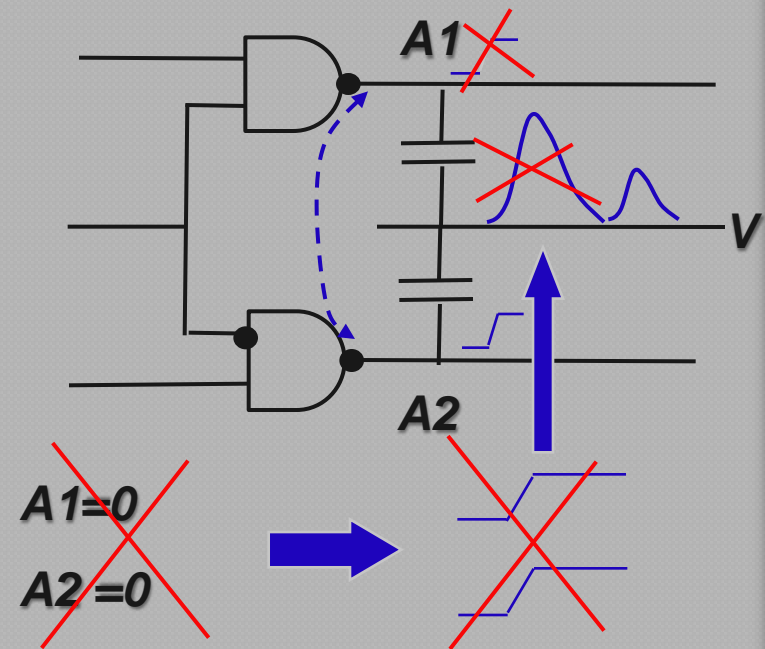
Cross Coupling Delay Noise

- ✓ Aggressor nets affect victim net through coupling capacitances
- ✓ Functional Noise: changes logic state of the victim net
 - Affects victim when it in a stable state
- ✓ Delay Noise: changes signal propagation delay
 - Affects victim net when it transitions
 - Delay changes accumulate along the signal propagation paths



Signal Correlation and False Noise Analysis

- ✓ Timing correlation:
 - nets switch at different clock cycles, etc.
- ✓ Logic correlation:
 - circuit logic *prohibits* some combinations of nets signals
 - it *prohibits* some aggressor nets from simultaneous switching



Two problems of false noise analysis:

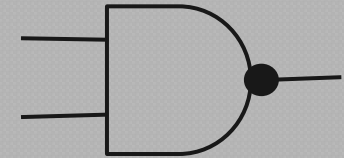
- Computing signal correlations
- Computing the worst possible noise and aggressors set injecting it
- Difficult optimization problem

Logic Constraints Representation and Derivation

✓ Set of conjunctive terms: $\bar{a} \cdot b \cdot \bar{c}$, $\bar{a} \cdot \bar{b} \cdot d$, ...

- Each term prohibits signal combination

$a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d}$ prohibits: $a=1, b=0, c=0, d=0$



- Resolution technique

- deriving sentences by resolution rule
- can handle multiple constraints
- can build approximate solutions
- works even at transistor level

- Simple Logic Implication (SLI)

- binary constraints only
- simpler implementation

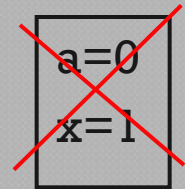
$$\bar{x} \cdot a \cdot b, x \cdot \bar{a}, x \cdot \bar{b}$$

$$a \cdot B, \bar{a} \cdot C \quad \longrightarrow \quad B \cdot C$$

$$(a=0) \rightarrow (x=0)$$



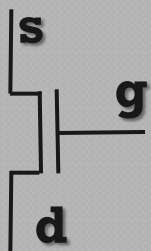
$$\bar{a} \cdot x$$



Transistor Level Logic Constraints Generation

- ✓ Set constraints for transistors
- ✓ Apply resolution rule
 - eliminate variables not involved in noise clusters
 - remove tautologies $a \cdot \bar{a} \cdot B$
 - remove constraints covered by other ones:

$(a \cdot \bar{b} \text{ covers } a \cdot \bar{b} \cdot \bar{c})$

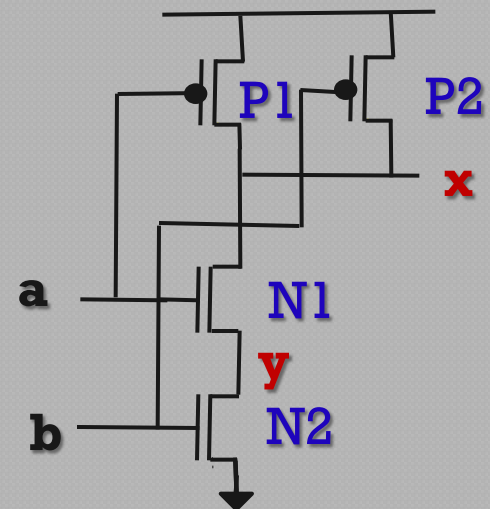


$$g \cdot s \cdot \bar{d}, g \cdot \bar{s} \cdot d$$



$$\bar{g} \cdot s \cdot \bar{d}, \bar{g} \cdot \bar{s} \cdot d$$

Logic constraints for NAND2



$$P1: \bar{a} \cdot \bar{x}$$

$$P2: \bar{b} \cdot \bar{x}$$

$$N1: a \cdot y \cdot x, a \cdot y \cdot \bar{x}$$

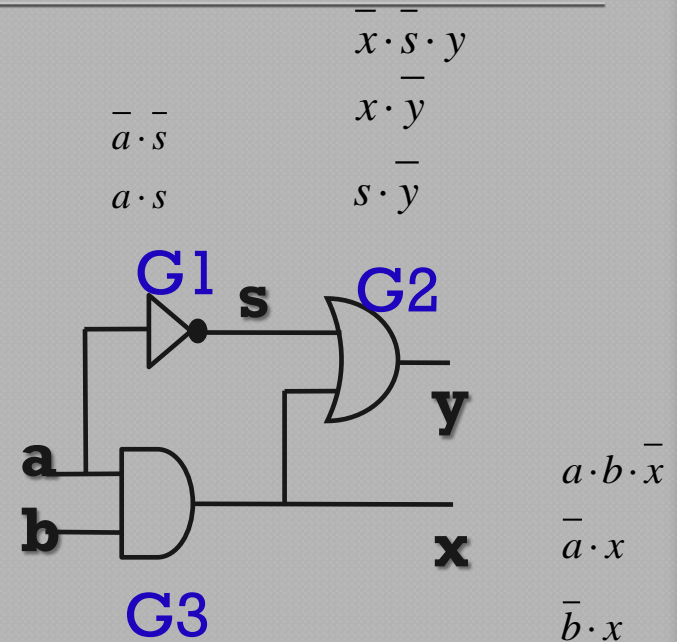
$$N2: b \cdot y$$

$$b \cdot y, a \cdot x \cdot \bar{y} \rightarrow a \cdot b \cdot x$$



Logic Constraints Derivation at Gate Level

- ✓ Start from transistor level constraints
- ✓ Apply resolution rule
 - until no constraints can be derived or all resources are exhausted
 - use propagation heuristic
 - propagate constraints through gates forward and backward
 - apply resolution rule to propagated and gate constraints
- ✓ Exclude tautologies and constraints covered by other ones



- G1, G2: $\frac{\bar{a} \cdot \bar{s}, \bar{x} \cdot \bar{s} \cdot y}{a \cdot s, s \cdot y} \rightarrow \bar{a} \cdot \bar{x} \cdot y$ (R1)
- G3, G2: $\frac{a \cdot b \cdot \bar{x}, x \cdot \bar{y}}{a \cdot s, s \cdot y} \rightarrow a \cdot b \cdot \bar{y}$ (R2)
- R1, R2: $\frac{\bar{a} \cdot \bar{y}, a \cdot b \cdot \bar{y}}{\bar{y} \cdot b \cdot \bar{y}} \rightarrow \bar{y} \cdot b \cdot \bar{y}$ (R3)



Linear Delay Noise Model

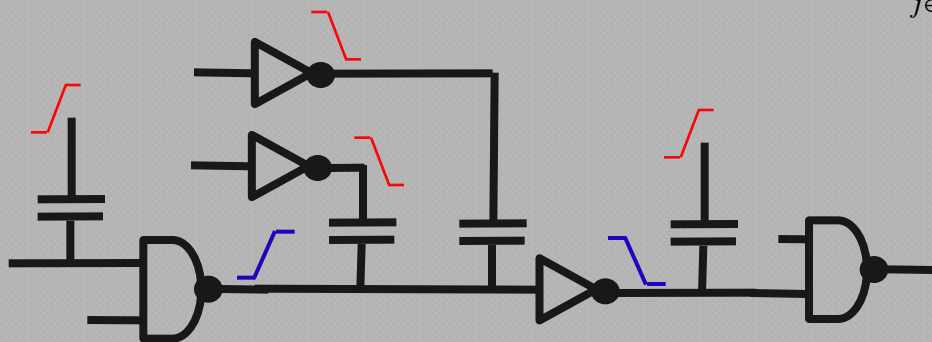
- ✓ Need for simple model to estimate each aggressor impact

Actual delay variation is verified by SPICE simulations

- ✓ Path delay variation is additive: $\Delta D_{Path} = \sum_{i \in Path} \Delta D_{Net,i}$

- ✓ Linearization of net delay: $D_{Net}(\Delta h_{noise}) = D_{Net, No_noise} + \frac{dD_{Net}}{dh_{noise}} \Delta h_{noise}$

- ✓ Additive model of net delay variation $\Delta D_{Net} = \sum_{j \in Net_aggr} \Delta D_{Net,j}$



$$\Delta D_{Path} = \sum_{i \in Path} \Delta D_{Net,i} = \sum_{i \in Path} \sum_{j \in Net_aggr} \Delta D_{agg,i,j}$$

- ✓ Linear model is used only for finding worst aggressor set
- ✓ Actual delay variation is verified by SPICE simulations

Computation of Linear Noise Model for Noise Cluster

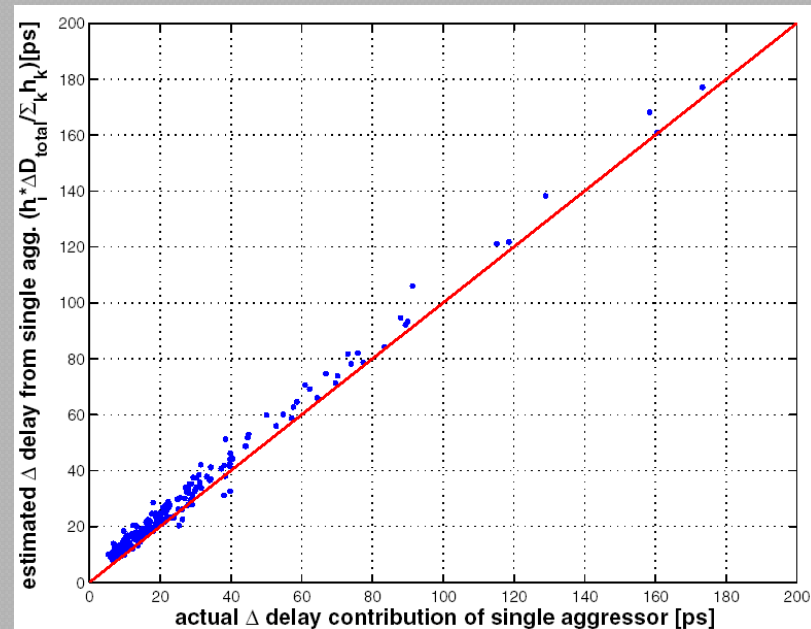
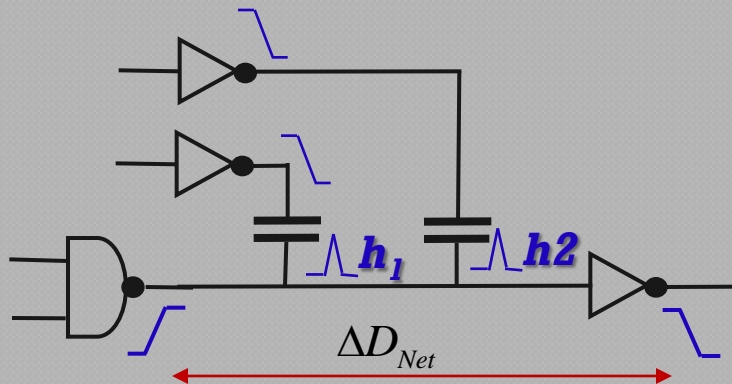
Compute noise pulse height h_i of each aggressor

Compute total noise pulse height: $H = \sum_{j \in \text{Net_Aggressors}} h_j$

Compute total net delay variation: ΔD_{Net}

Estimate delay variation due to each aggressor:

$$\Delta D_j = h_j \frac{\Delta D_{\text{Net}}}{\sum_{j \in \text{Net_Aggressors}} h_j}$$

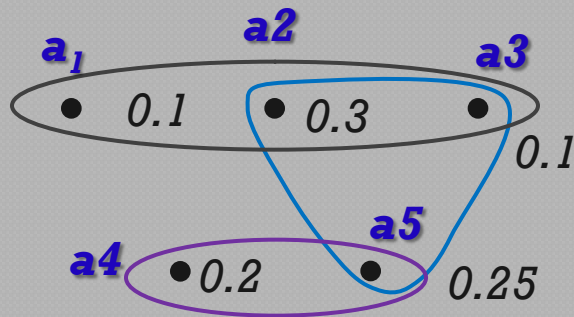


Error of delay noise additive model

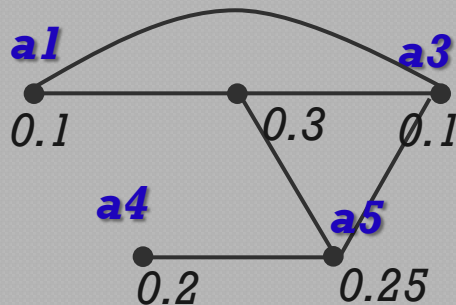


Constraint Graph and Hyper-Graph

- ✓ Vertices are aggressor nets
- ✓ Edges / hyper-edges are constraints
 - weight is injected noise
- ✓ Maximum weight independent set (MWIS) of vertices
 - does not have any edge/hyper-edge as subset



- Constraints/hyper-edges: $\{a_1a_2a_3, a_4a_5, a_2a_3a_5\}$
- MWIS = $\{a_1, a_2, a_5\}$, $w=0.65$

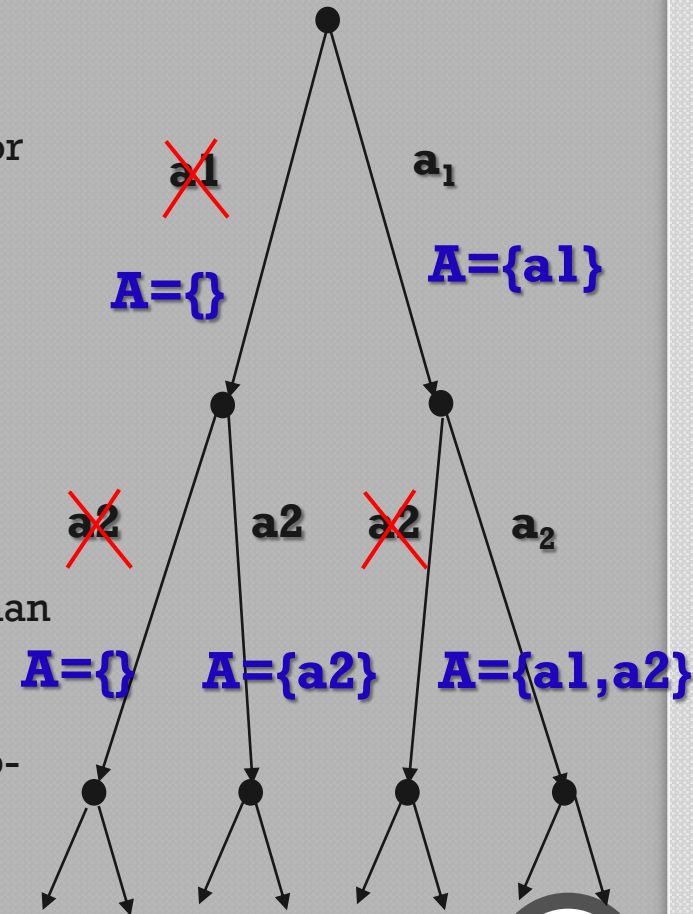


- Constraints (SLI/) edges: $\{a_1a_2, a_2a_3, a_3a_5, a_2a_5, a_4a_5, a_1a_3\}$
- MWIS = $\{a_1, a_4\}$, $w=0.35$



Branch and Bound Algorithm

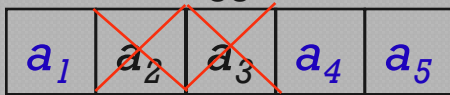
- ✓ Recursively traverses decision tree
 - branches by including or not including an aggressor
 - expands “current” aggressor set
- ✓ Accumulates partial solution
 - which is the worst found realizable aggressor set
- ✓ Cuts branches
 - sub-trees with non realizable aggressor sets
 - sub-trees with aggressor sets injecting less noise than the maximum found one
- ✓ Estimates upper bound of noise corresponding to a sub-tree



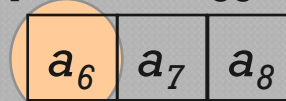
Data Flow in Branch and Bound Algorithm

Input data

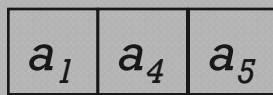
Processed aggressors



Unprocessed aggressors

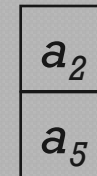


Current aggressor set



to process

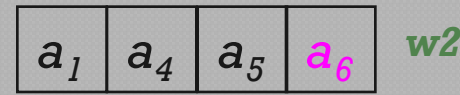
Worst found aggressor set



W_{worst}

Branching

New aggressor sets:



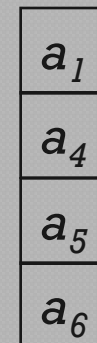
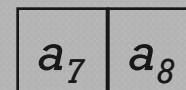
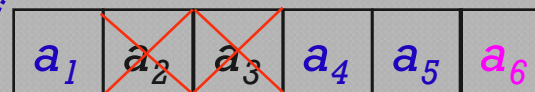
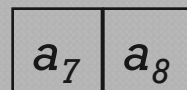
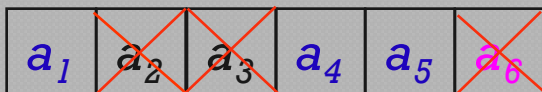
Updating worst found aggressor set

$$w_2 = W'_{worst} > W_{worst}$$

Cutting branches

Check constraints

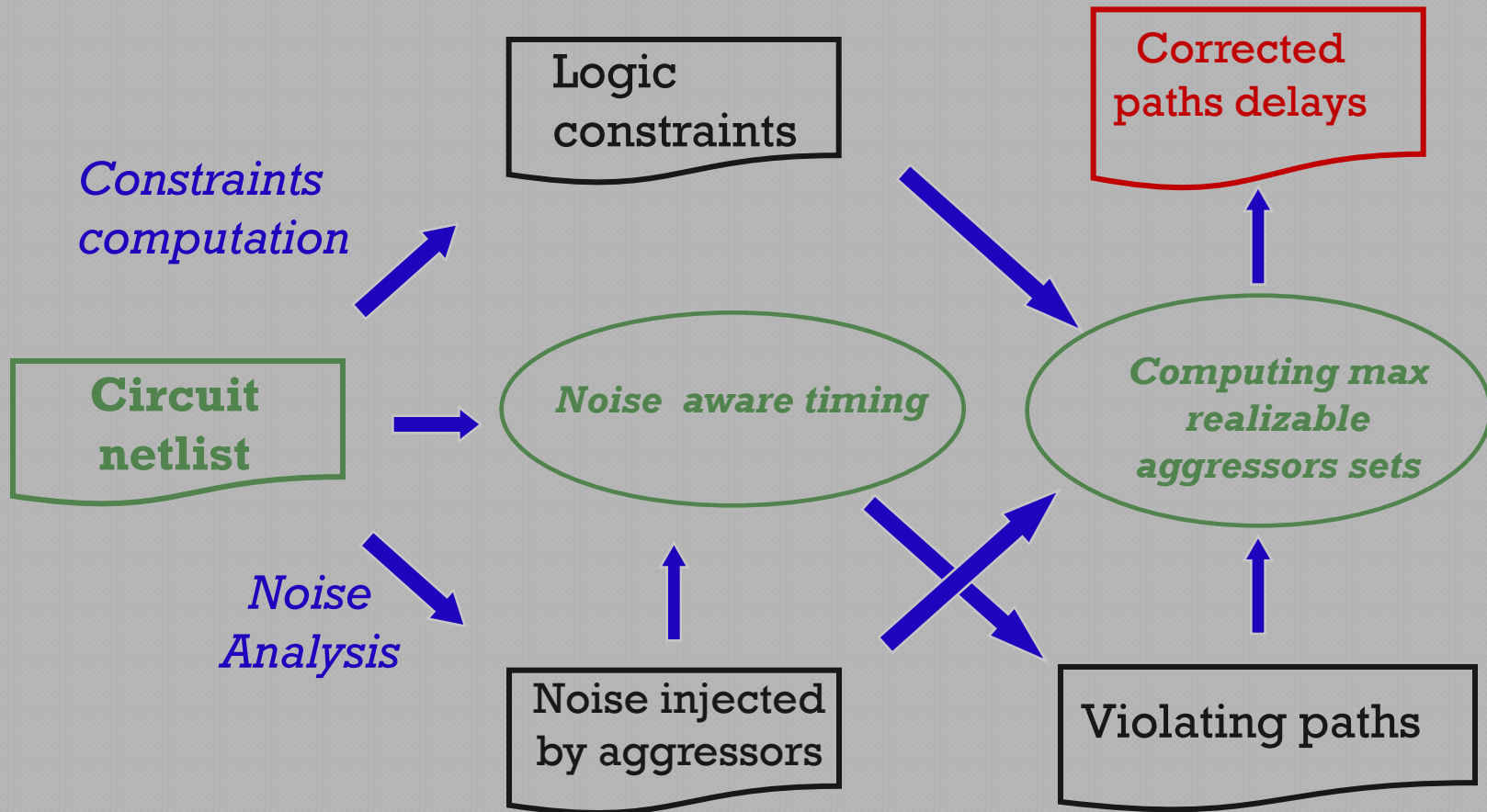
Compare $W_i + w_i$ with W_{worst}



Input data for recursive calls of B&B algorithm



Delay Noise Analysis Data Flow

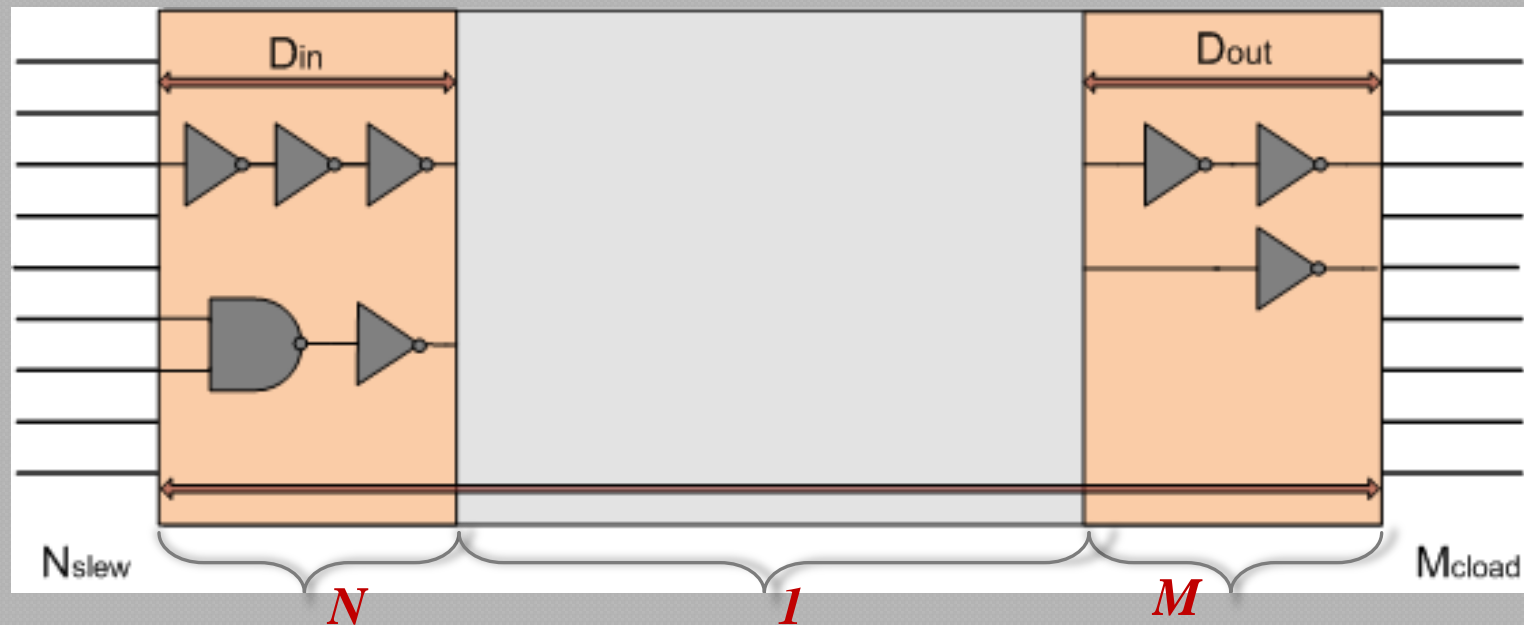


Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Input stimulus generation for IP blocks
8. Logic correlation analysis for timing and noise estimation
- 9. IP blocks characterization speed-up**
10. Future technologies problems



Analysis of the Dependences and Decomposition



$$D(S, C) = D(S_0, C_0) + (D_{in}(S) - D_{in}(S_0)) + (D_{out}(C) - D_{out}(C_0))$$

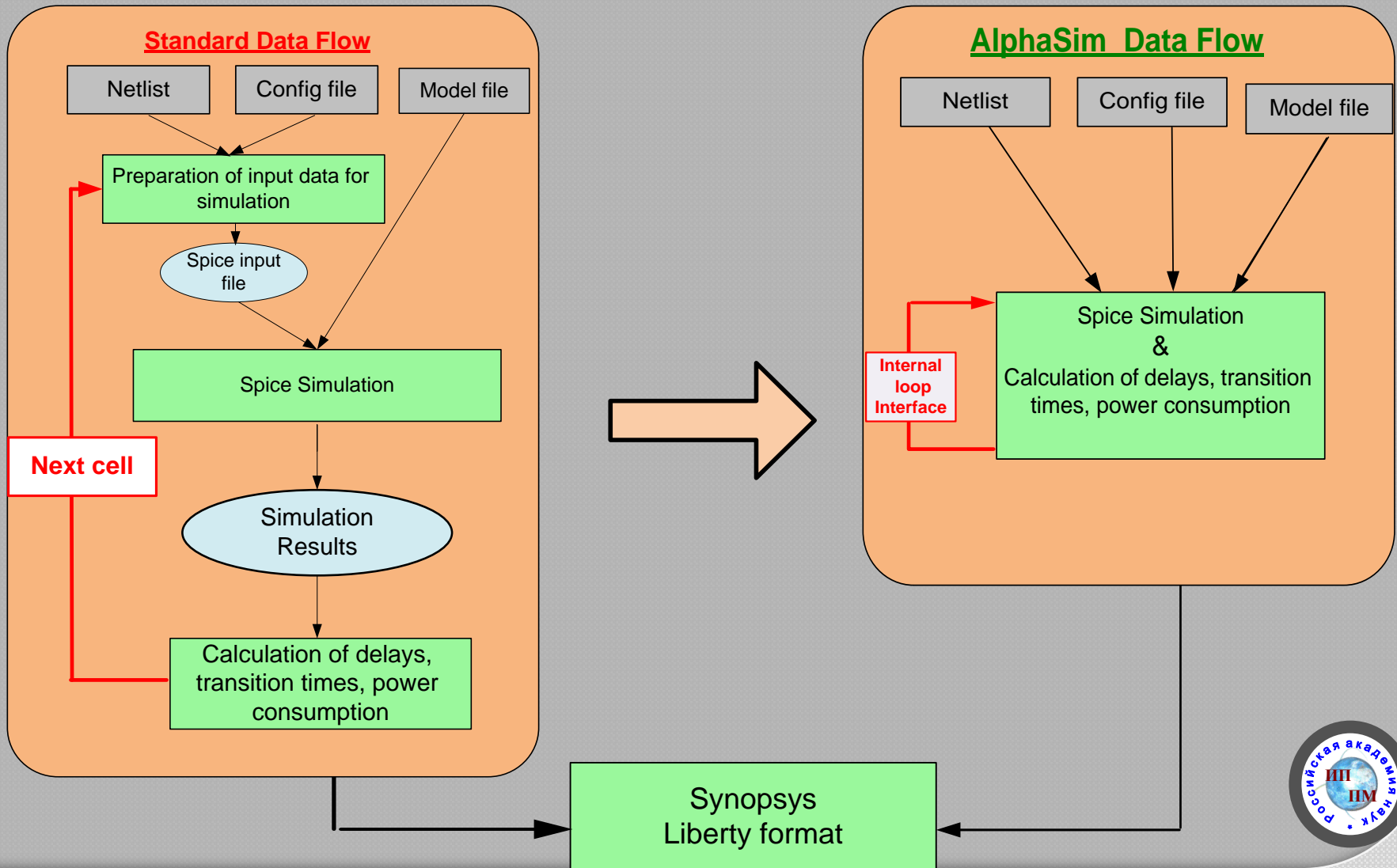
IP blocks Characterization Speed-up

Reduction of repeating Simulations

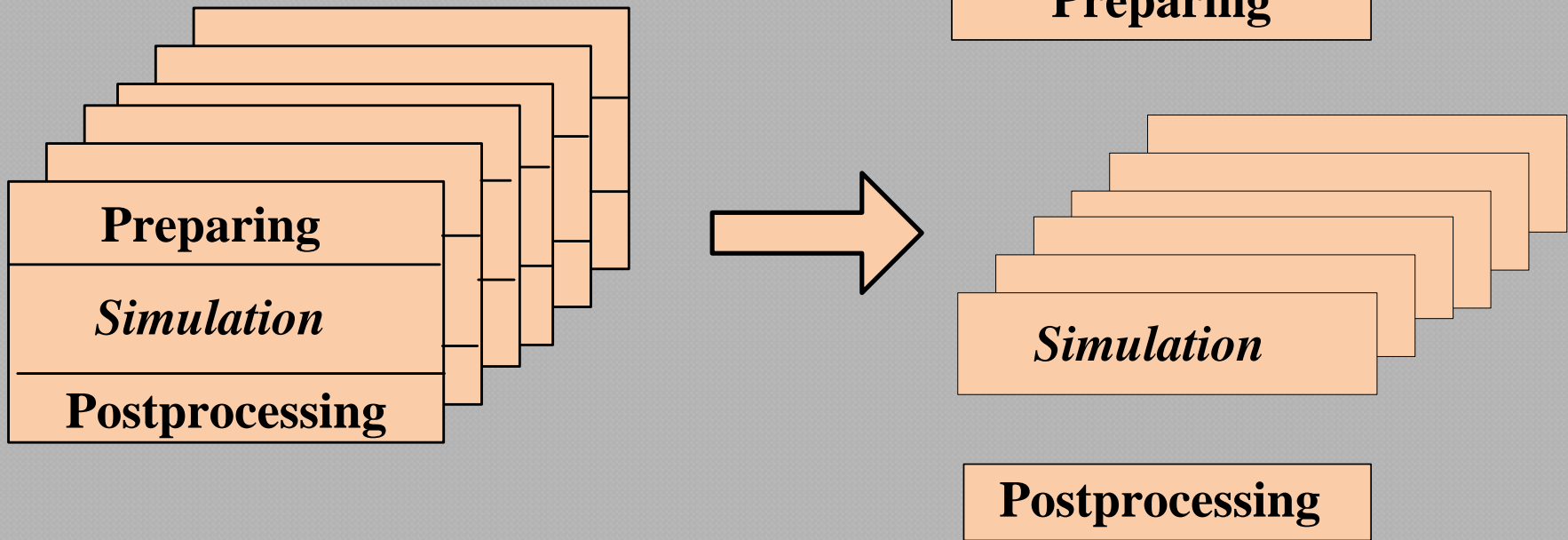
- ✓ Output DCCCs (1 or 2) cascades are simulated M times for different outputs.
- ✓ Input DCCCs (2 or 3) cascades are simulated N times for different input slope (transition time).
- ✓ The full circuit is simulated during single path.



AlphaSim vs Standard Data Flow



Parallel Simulations for Different C / S

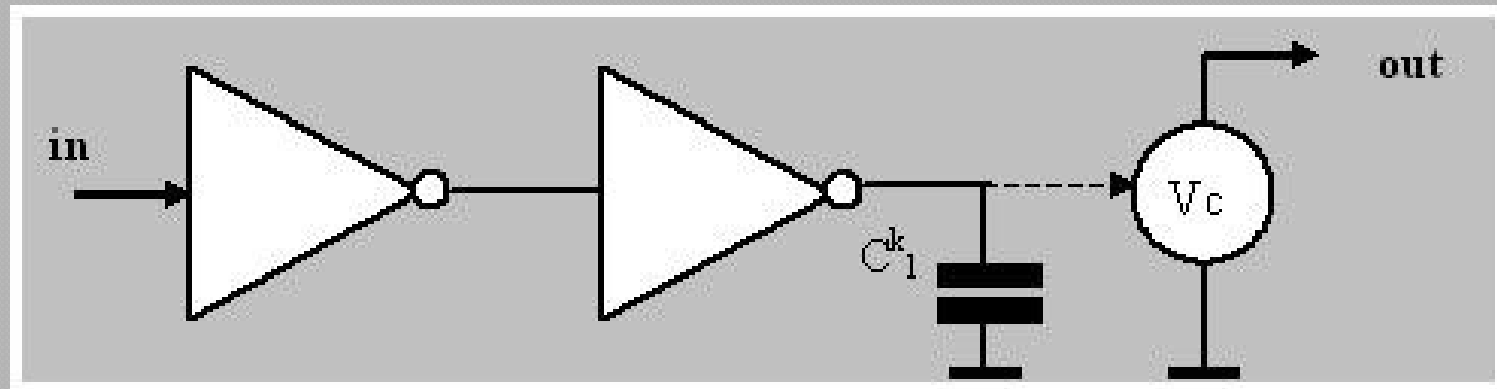


Ananalysis of Input Driver

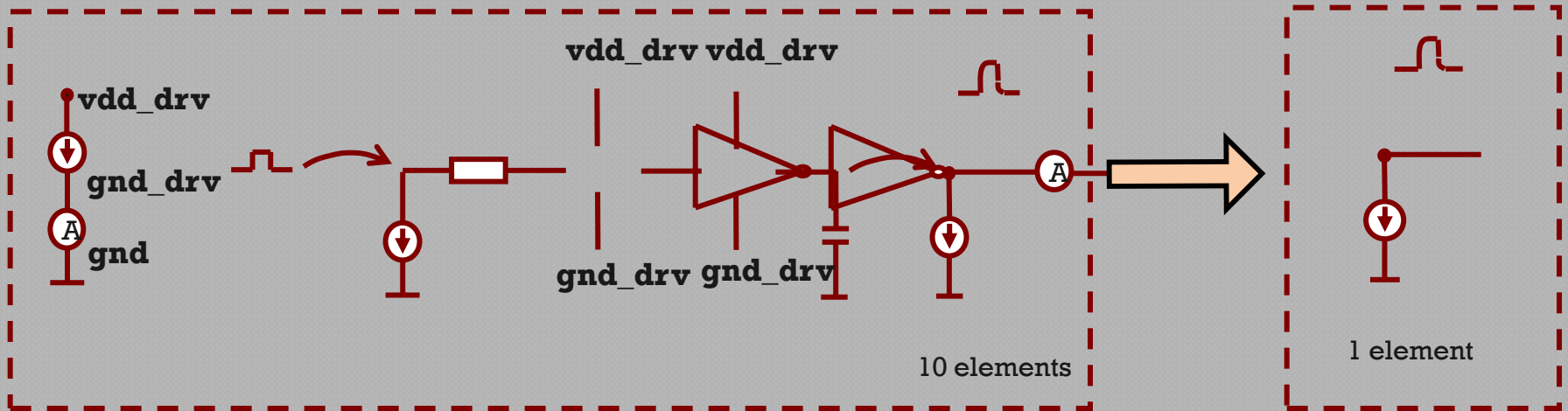
Input driver is used to generate smooth input waveform.

The voltage repeater is required to exclude direct contact between C_{inp} and driver output

The set of C_{k1} is chosen to generate the required input slope (transition time) $\{ S_{kinp} \}$.



Preliminary Driver Characterization and Speed-up



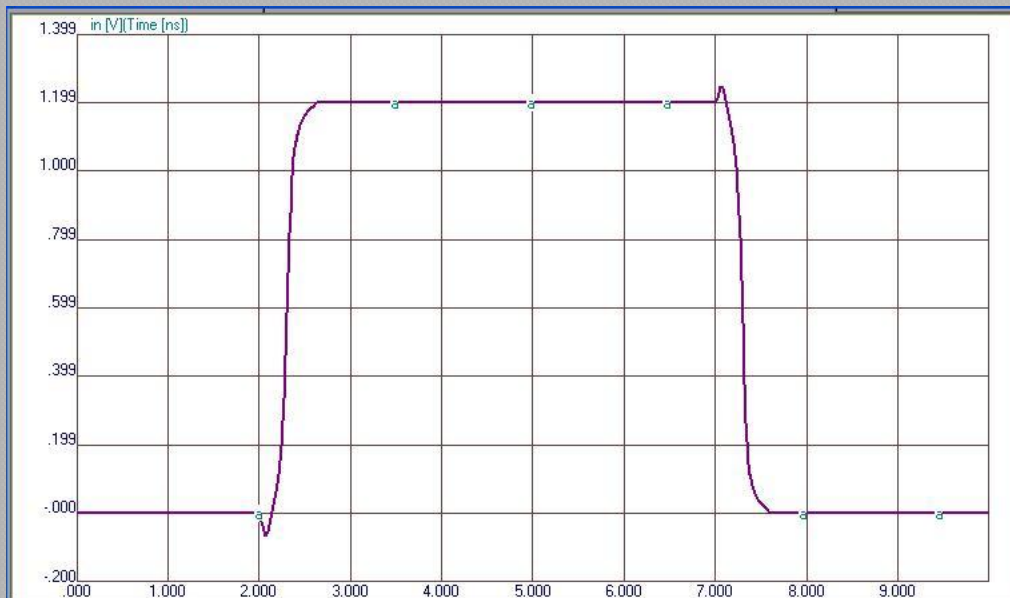
- ✓ Normal approach: Nand3 + 3 drivers: $4 + 3 \cdot 10 = 34$ elements
- ✓ Modified approach: Nand3: $4 + 3$ new characterized sources = 7 elements

Driver Characterization

$$y = \frac{m_1 \cdot r_1^3}{6h_i} + \frac{m_2 \cdot r_2^3}{6h_i} + \left(f_1 - \frac{m_1 \cdot h_i^2}{6} \right) \cdot \frac{r_1}{h_i} + \left(f_2 - \frac{m_2 \cdot h_i^2}{6} \right) \cdot \frac{r_2}{h_i}$$

Where : $\{x_i\}$ – node argument values, $\{f_i\}$ – node function values,

$r_1 = x_i - a$, $r_2 = x - x_{i-p}$, $f_1 = f_{i-p}$, $f_2 = f_i$, $h_i = x_i - x_{i-p}$, m_1 , m_2 – spline coefficients for each i-th interval.



Input spline = driver output



Outline

1. Contemporary technologies & IP blocks design problems
2. Deterministic and statistical timing analysis
3. Digital noise analysis problems
4. Logic cell characterization
5. Memory cell characterization
6. Decomposition problems for IP blocks
7. Input stimulus generation for IP blocks
8. Logic correlation analysis for timing and noise estimation
9. IP blocks characterization speed-up
- 10. Future technologies problems**



**Thank
You!**

