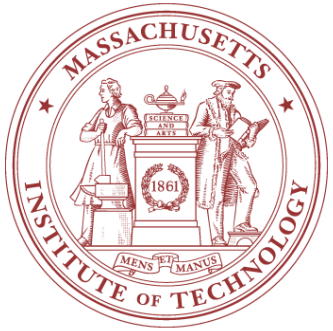
Design Methodologies and Automation for Relay-Based Circuits

Dejan Marković (UCLA)

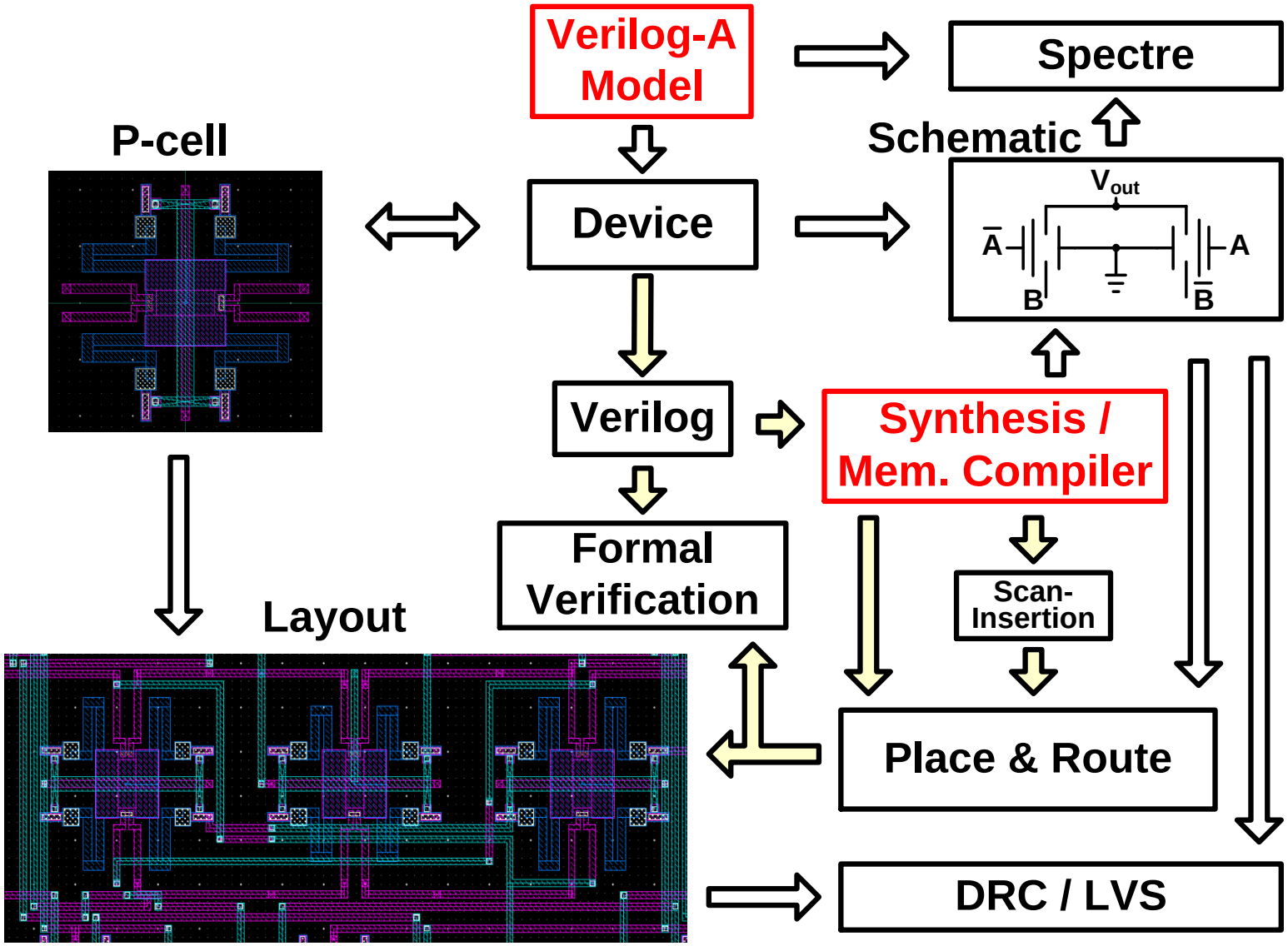
in collaboration with

Elad Alon, Tsu-Jae King Liu (UC Berkeley)

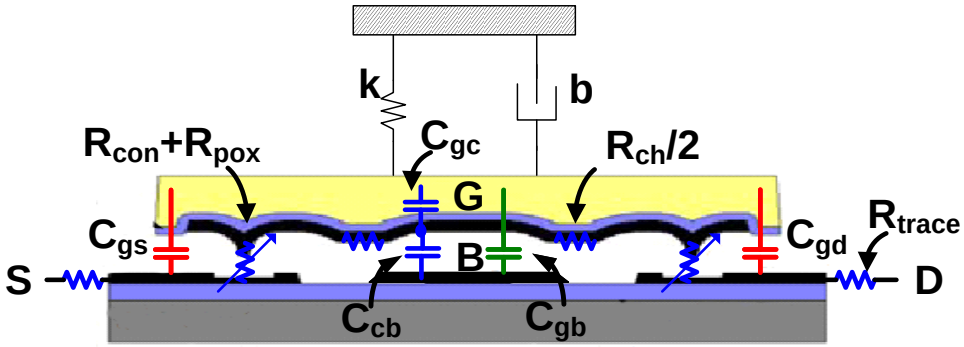
Vladimir Stojanović (MIT)

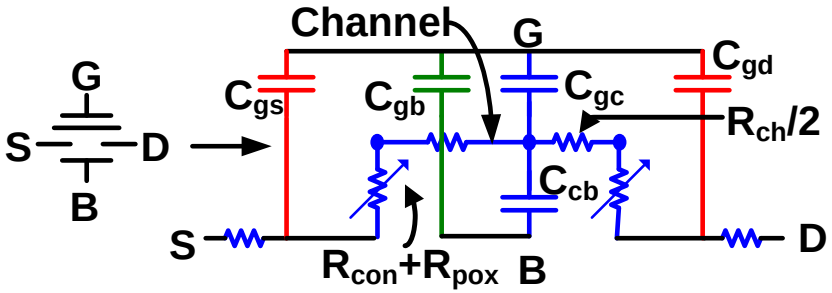


Relay VLSI Design Infrastructure



Compact Verilog-A Model

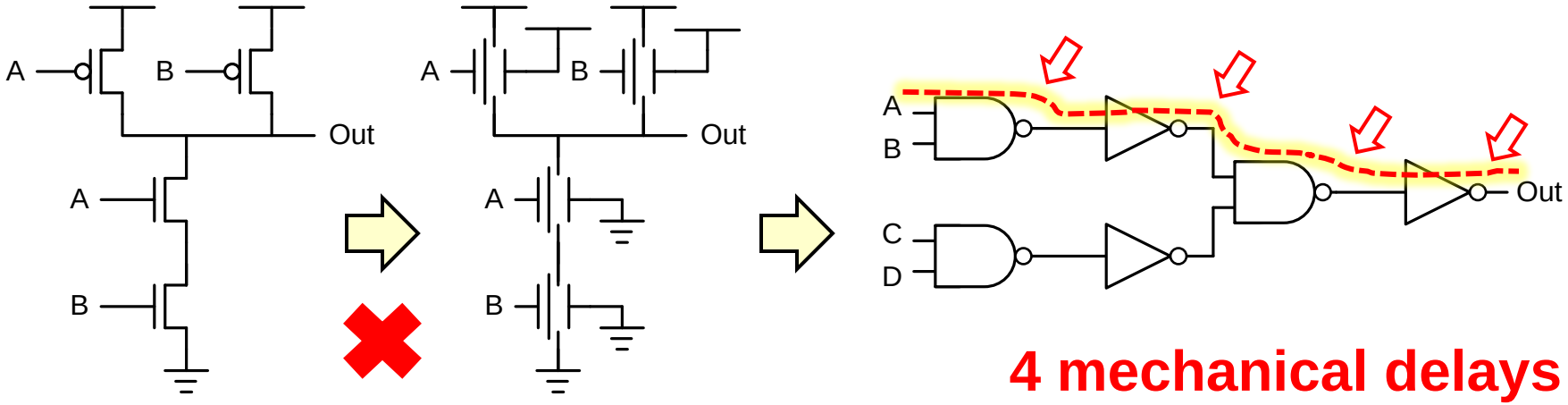




- Relay electromechanical behavior
- Calibrated against measured device
- Model used in Spectre simulations

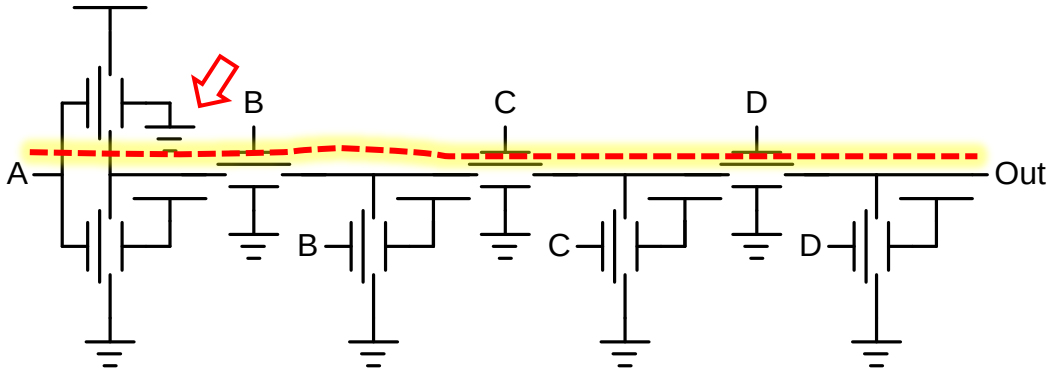
Relay Gate vs. CMOS

- **1 : 1 mapping unacceptable (1 mech. delay per stage)**



4 mechanical delays

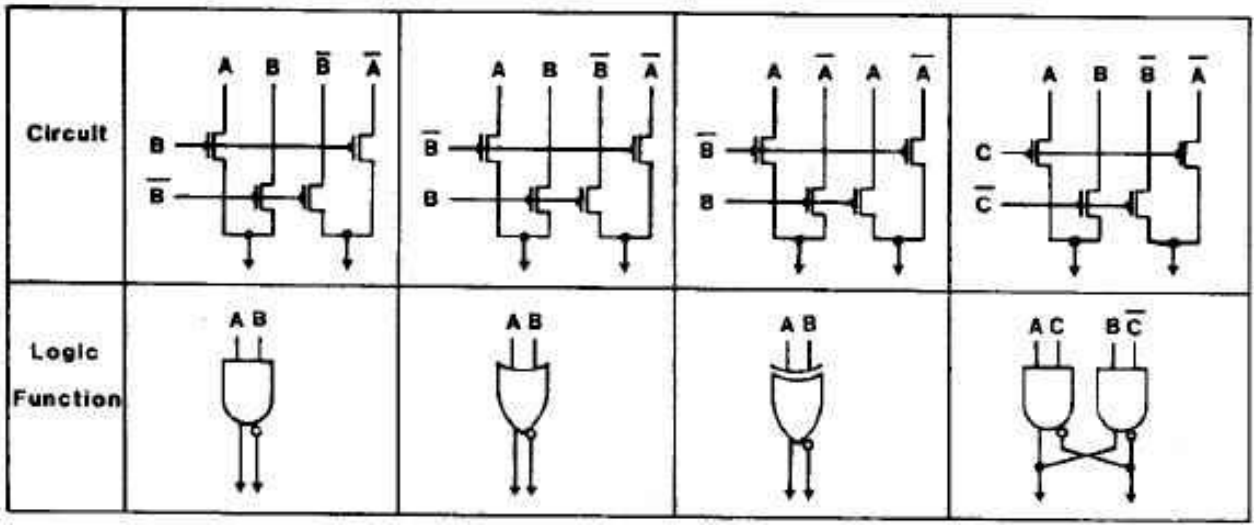
- **Signals propagate through S/D**



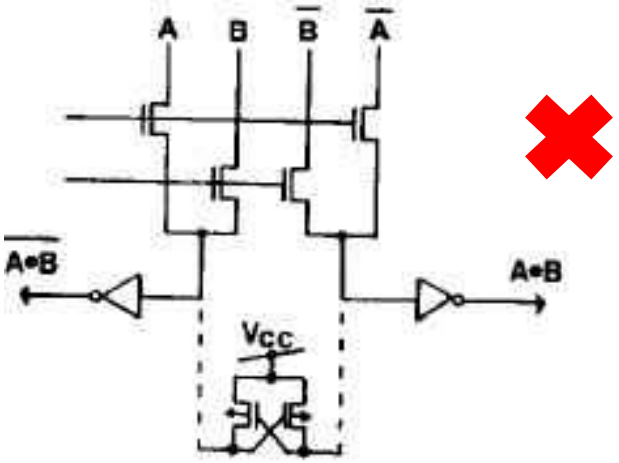
1 mechanical delay

Relay Gate vs. PTL

- **Intermediate signals can't drive relay gate**



- **Inverters cannot be used (add a mech. delay)**



(from Yano et al., JSSC 4/90)

Relay-based Synthesis

Goals:

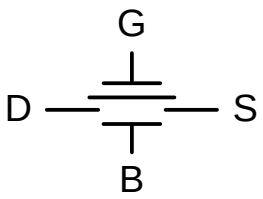
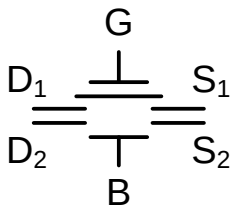
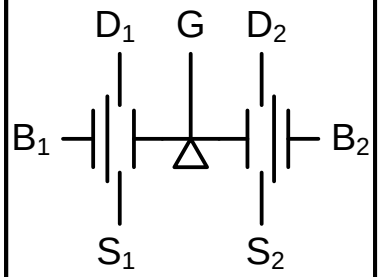
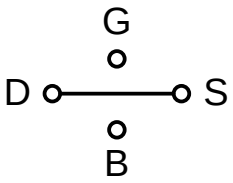
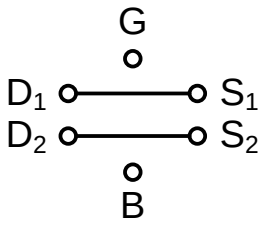
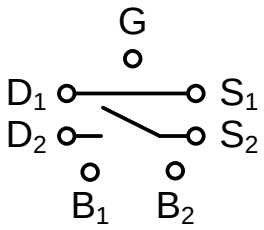
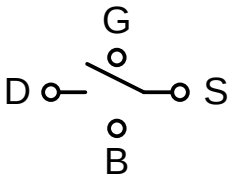
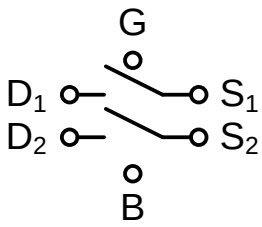
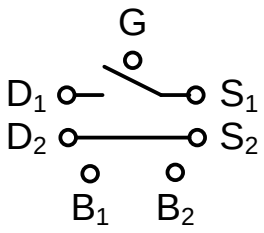
- Minimize the number of mechanical delays
- Minimize the number of relay devices

Approach:

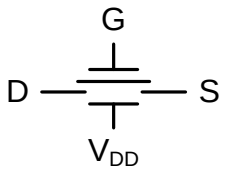
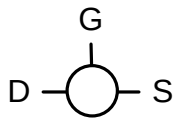
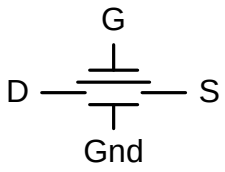
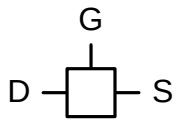
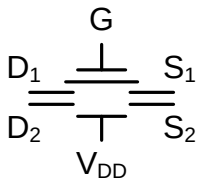
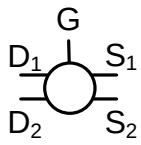
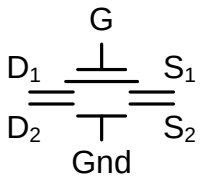
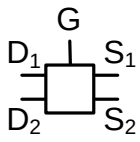
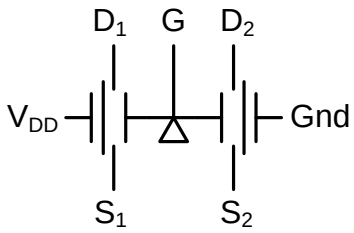
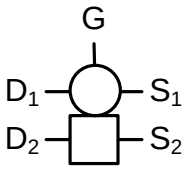
- Work with compound gates
- Implement node sharing

Types of Relay Devices

- **Support for three device types**

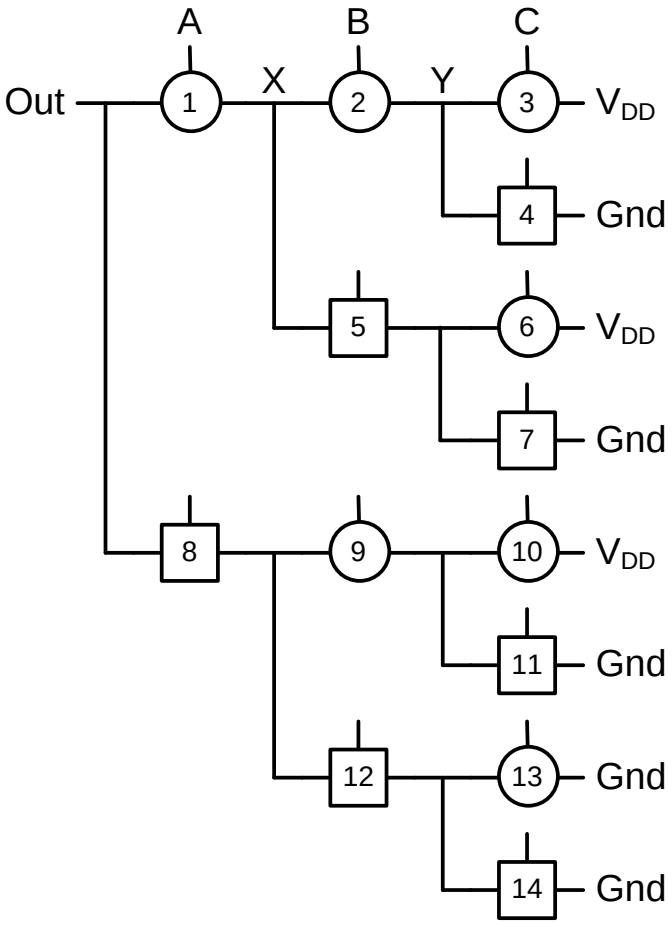
4-Terminal Relay	6-Terminal Relay	Seesaw Relay
		
On State: G≠B 	On State: G≠B 	State 1: G≠B1 
Off State: G=B 	Off State: G=B 	State 2: G≠B2 

Common Device Configurations

	Circuit Symbol	Abbreviated Symbol	Function
4T			0 Activated Pass Gate
			1 Activated Pass Gate
6T			0 Activated Double Pass Gate
			1 Activated Double Pass Gate
Seesaw			Compound Pass Gate

Tree-based Approach

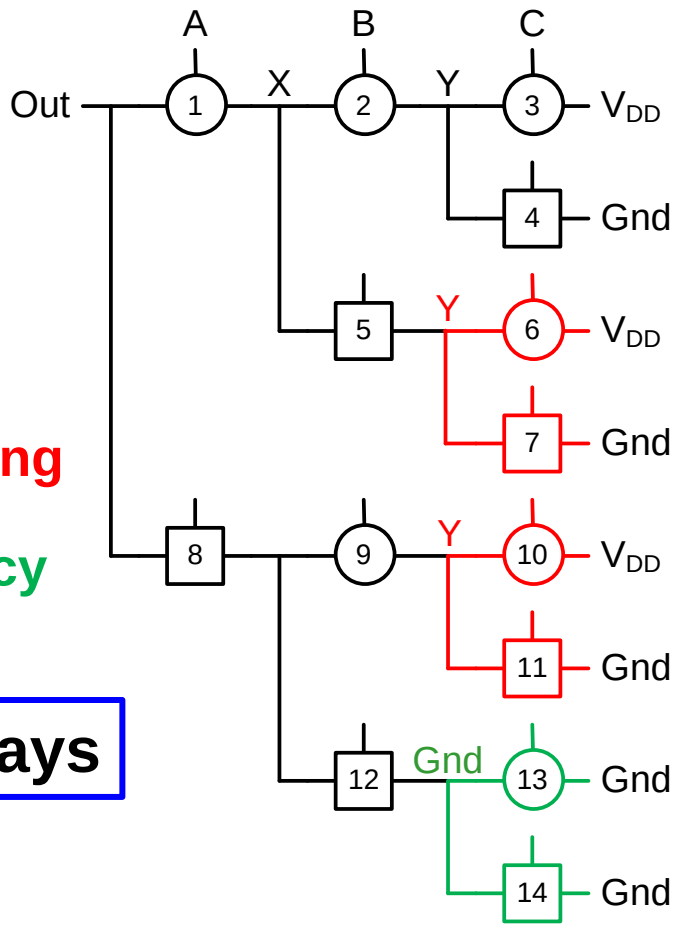
- **1 mechanical delay, large device count**



Out	A	B	C
1	0	0	0
0	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

Tree-based Approach

- 1 mechanical delay, large device count



Out	A	B	C
1	0	0	0
0	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

node sharing

redundancy

14 → 8 relays

Truth-Table Reduction

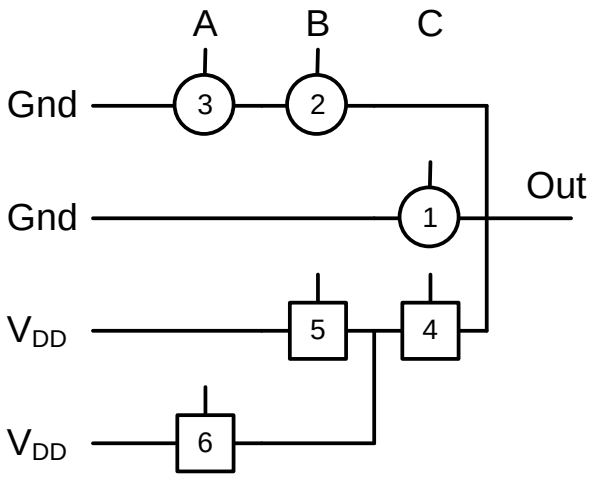
- **Redundancy pruning**
  - **Perform K-maps at the truth-table level**
  - **2^N table entries**

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
1	1	0	0

A	B	C	Out
0	1	1	1
1	0	1	1
1	1	1	1

Truth-Table Reduction

- **Redundancy pruning**
  - **Perform K-maps at the truth-table level**
  - **A "x" value is marked as -1**



A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
1	1	0	0

A	B	C	Out
0	1	1	1
1	0	1	1
1	1	1	1

A	B	C	Out
0	0	-1	0
0	-1	0	0
1	-1	0	0

A	B	C	Out
-1	1	1	1
1	-1	1	1

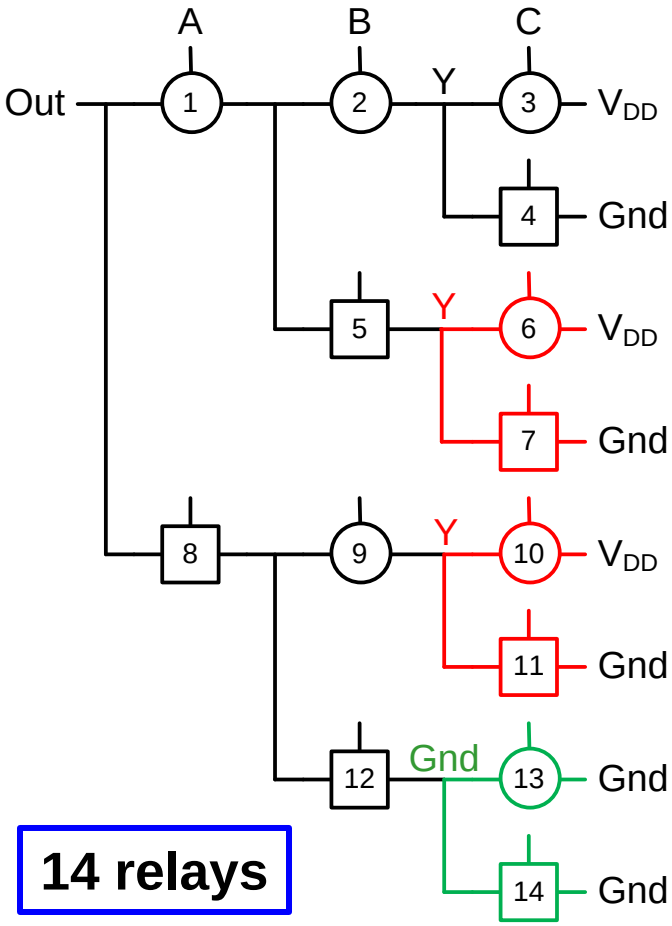
A	B	C	Out
0	0	-1	0
-1	-1	0	0

A	B	C	Out
-1	1	1	1
1	-1	1	1

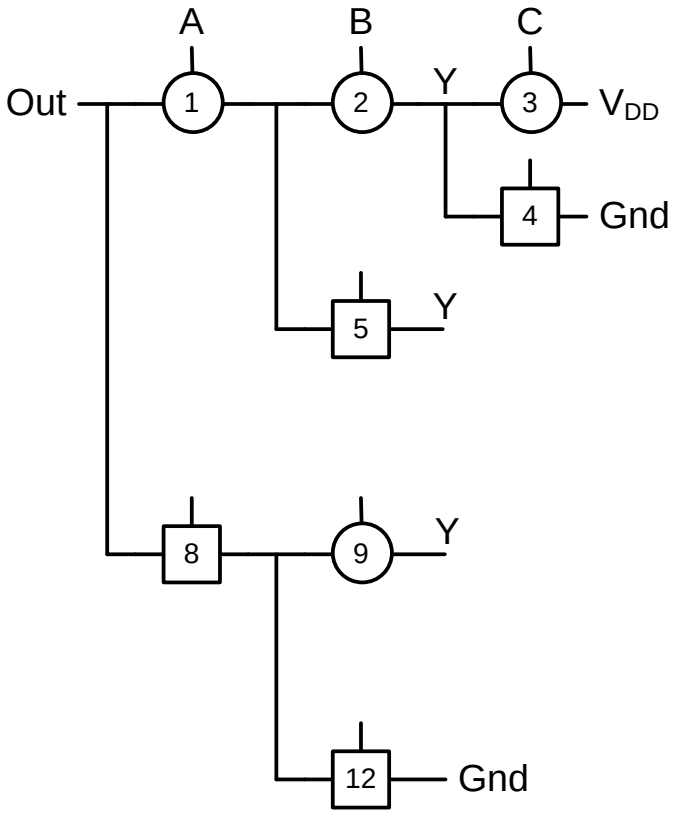
2^N table entries

Node Sharing

- **Utilizes common sub-expressions**
  - **One output shown for simplicity (could be separate trees)**

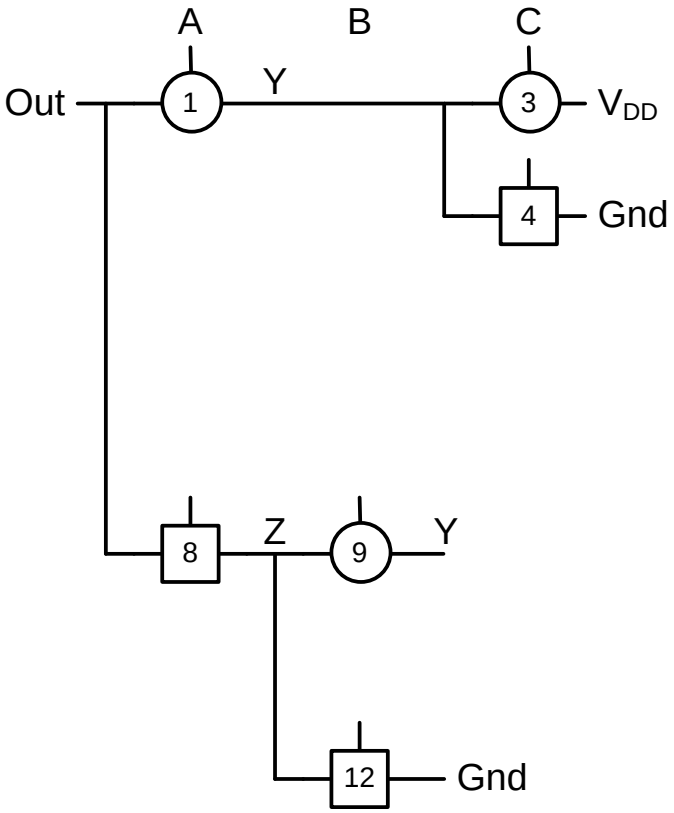


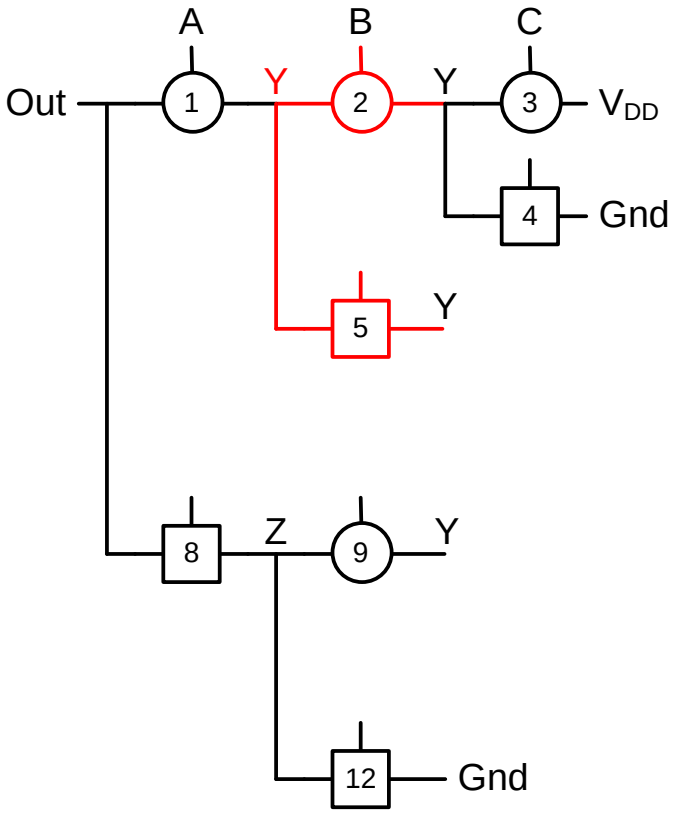
14 relays



Node Sharing

- **Utilizes common sub-expressions**
  - **One output shown for simplicity (could be separate trees)**

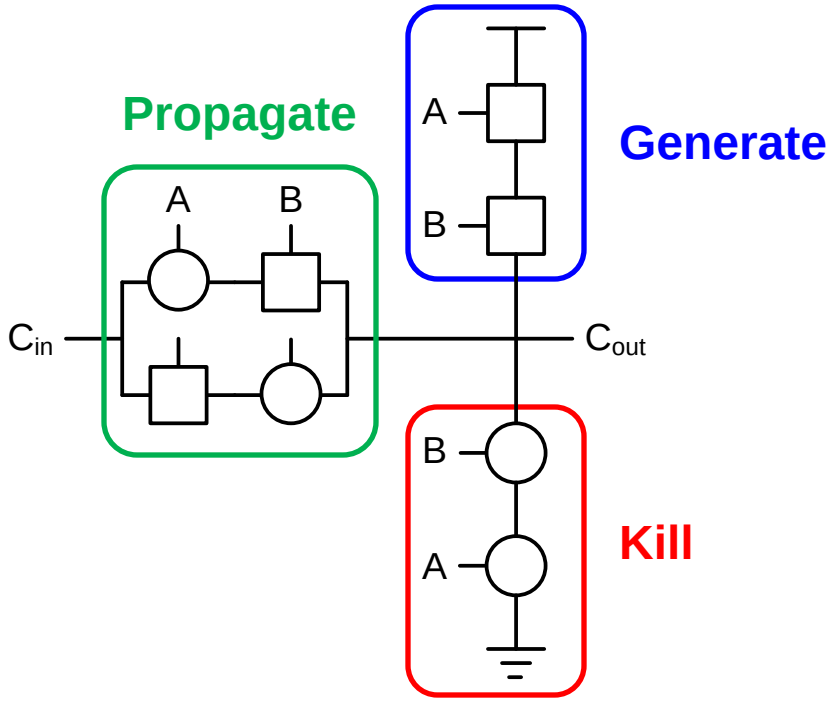




6 relays

Support for S/D Input

- **Essential for 0-mechanical-delay concatenation**

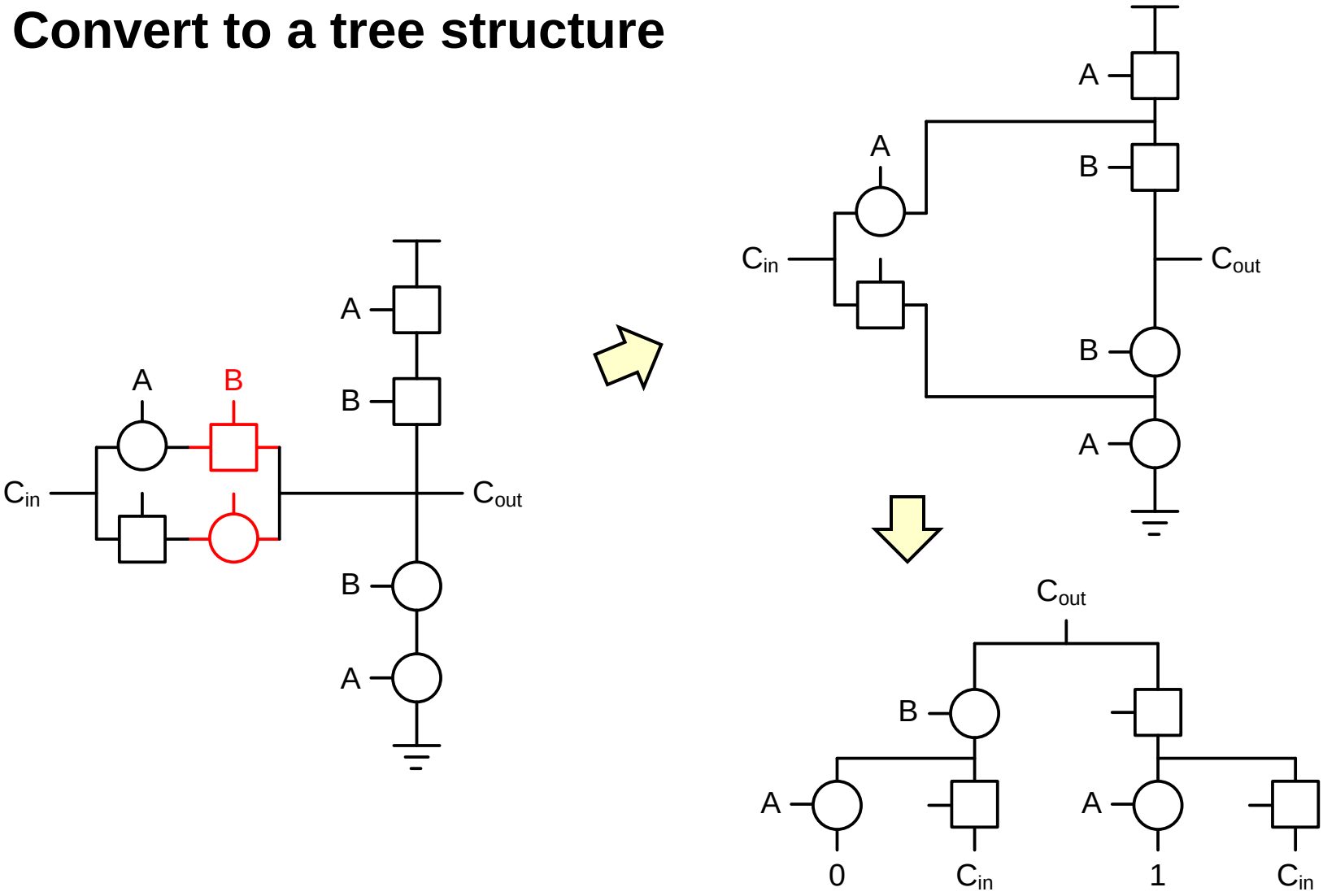


Adder example

How do we synthesize for S/D input?

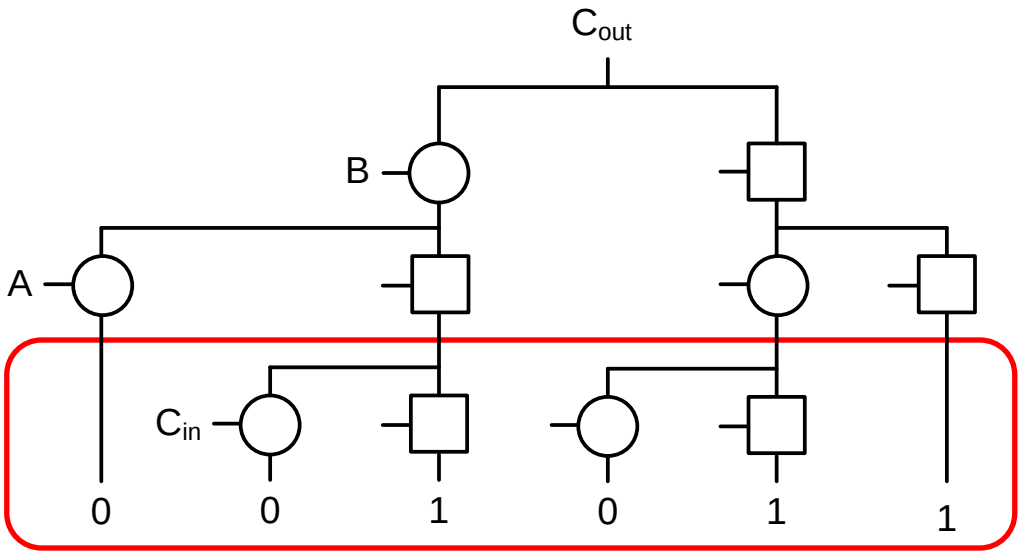
Support for S/D Input

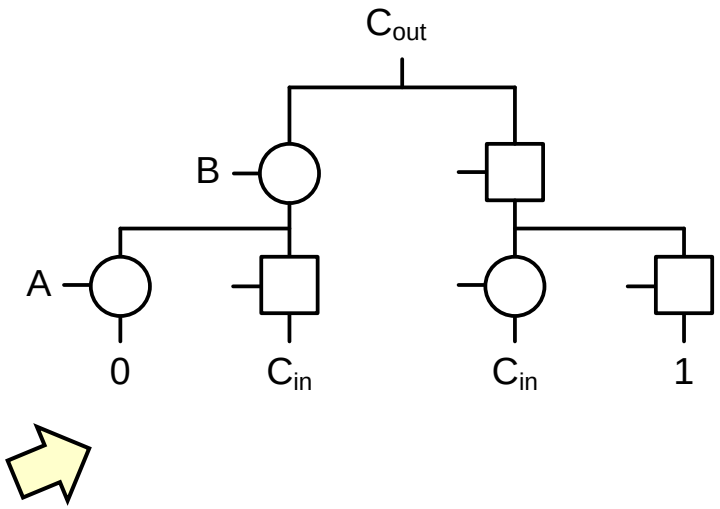
- **Convert to a tree structure**



BDD Representation

- **Input on the lowest hierarchy can be merged into S/D**



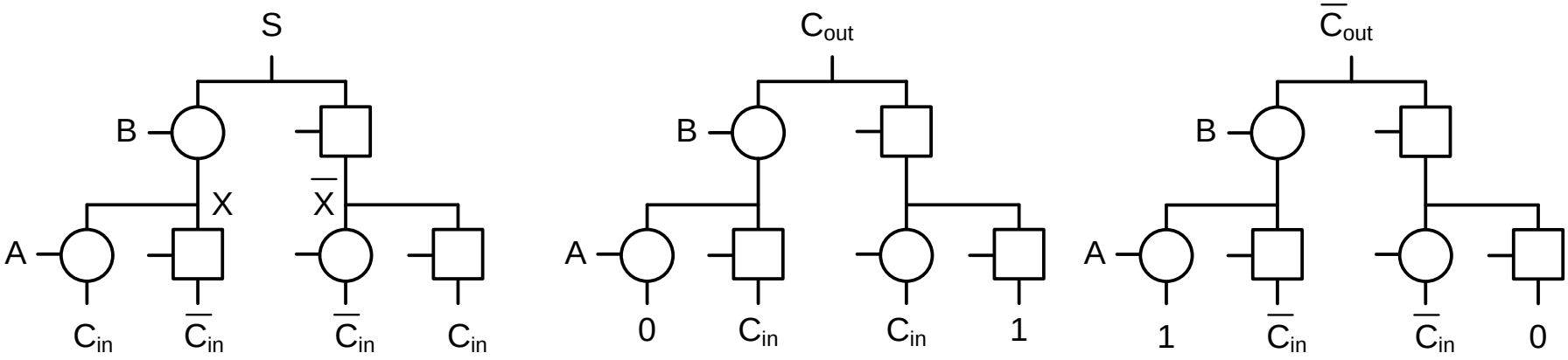


S/D mapping

00	01	10	11
0	C_{in}	$\overline{C_{in}}$	1

Modified-BDD Adder

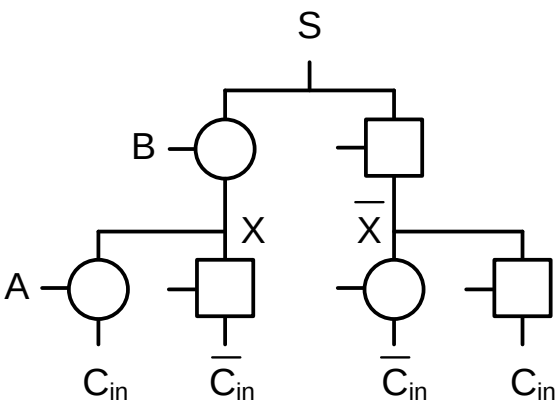
- Generate C_{out} and its complement for ripple carry
  - Cannot insert inverters (gate input = mechanical delay)

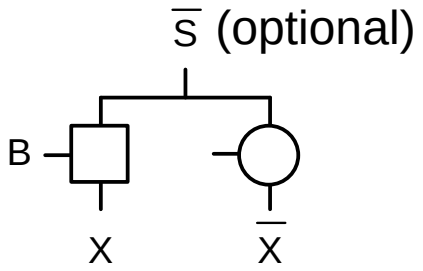


Inversion requires tree duplication

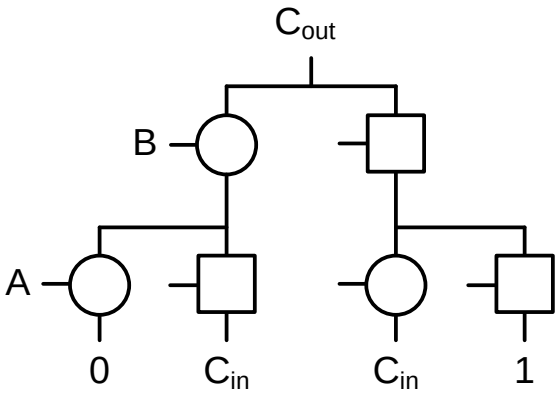
Modified-BDD Adder

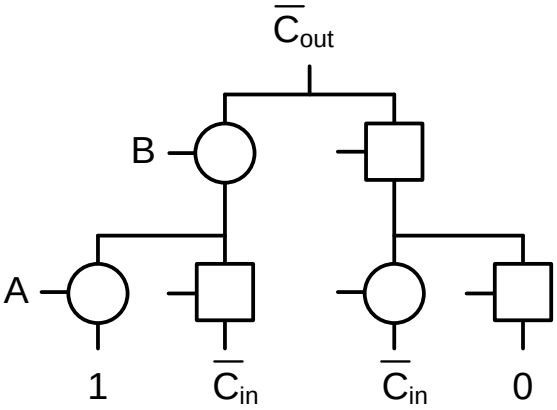
- **Node sharing can be exploited in some cases**





Suitable for node sharing





NOT suitable for node sharing

Matlab vs. ABC Tool

- **Runtime comparison (1 mechanical delay)**

Design	Matlab	ABC*
1-bit add	0.7 s	0.01 s
4-bit add	20.8 s	0.02 s
8-bit add	2.8 hrs	0.02 s
32-bit add	impractical	0.05 s
64-bit add	impractical	0.16 s
4:1 MUX	0.6 s	0.01 s
8:1 MUX	0.9 s	0.02 s
Instr. decoder	258 s	0.02 s

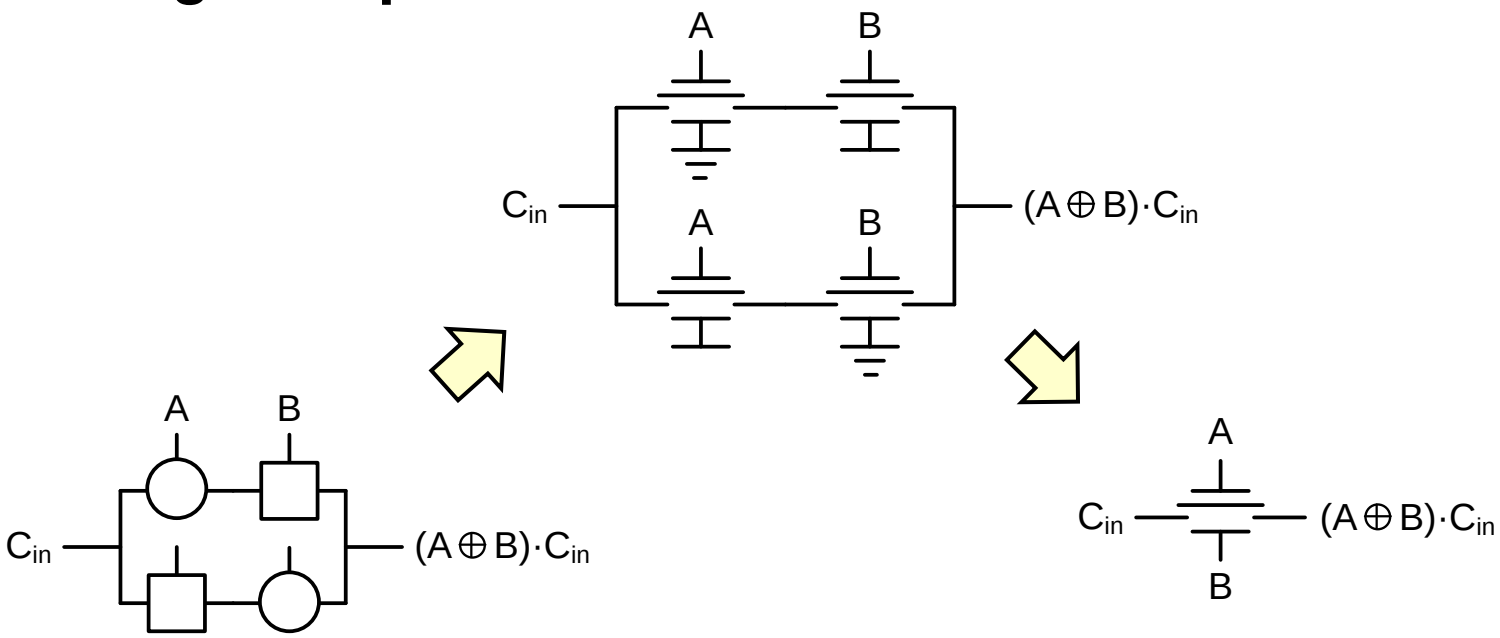
* w/o sharing algorithm

Matlab tool limited to logic cones with N ≤ 16 inputs

http://www.eecs.berkeley.edu/~alanmi/abc

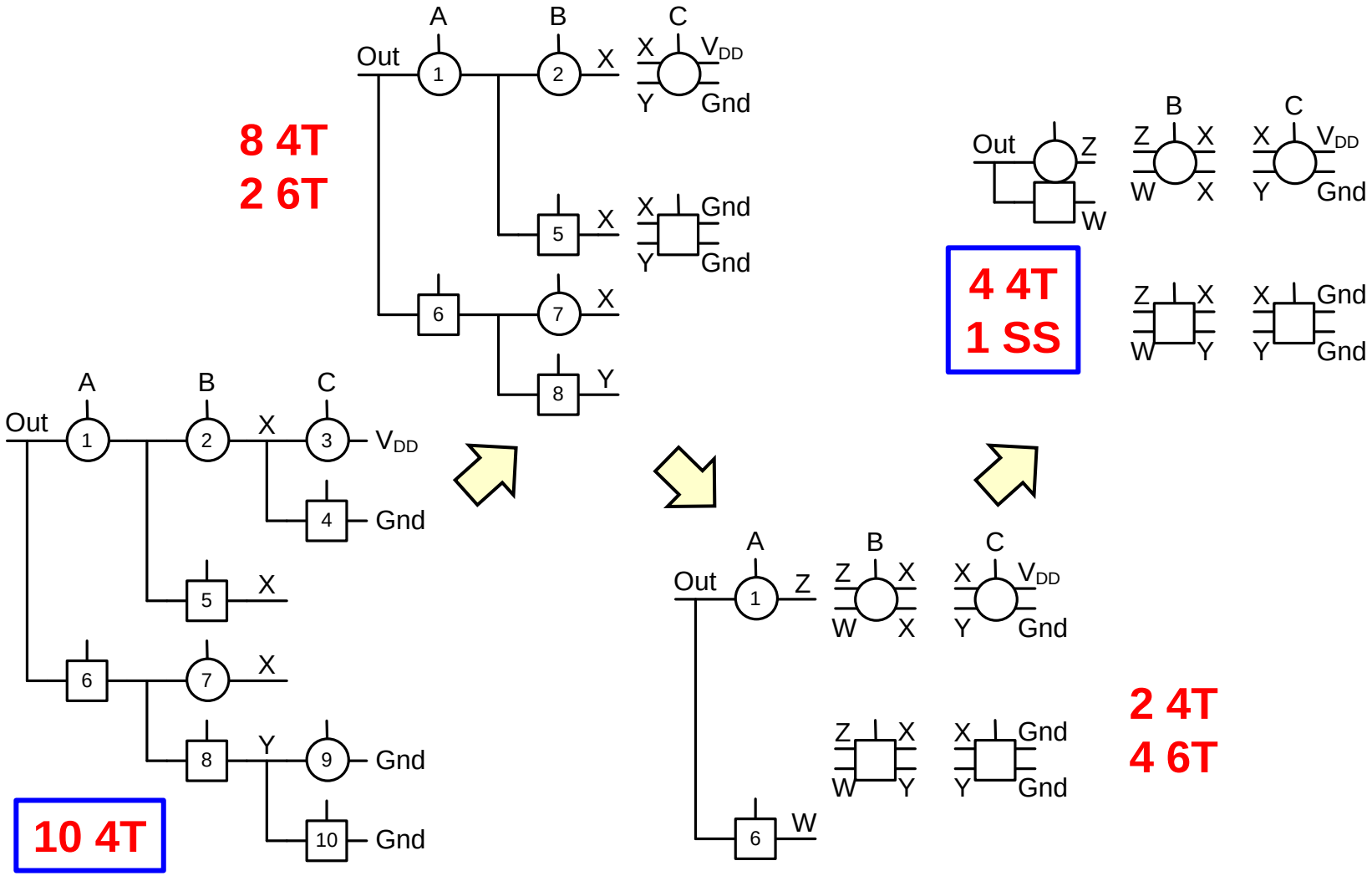
Synthesis: Remaining Challenges

- Node sharing in the BDD
- Area vs. mechanical delay tradeoff
- Back-gate input



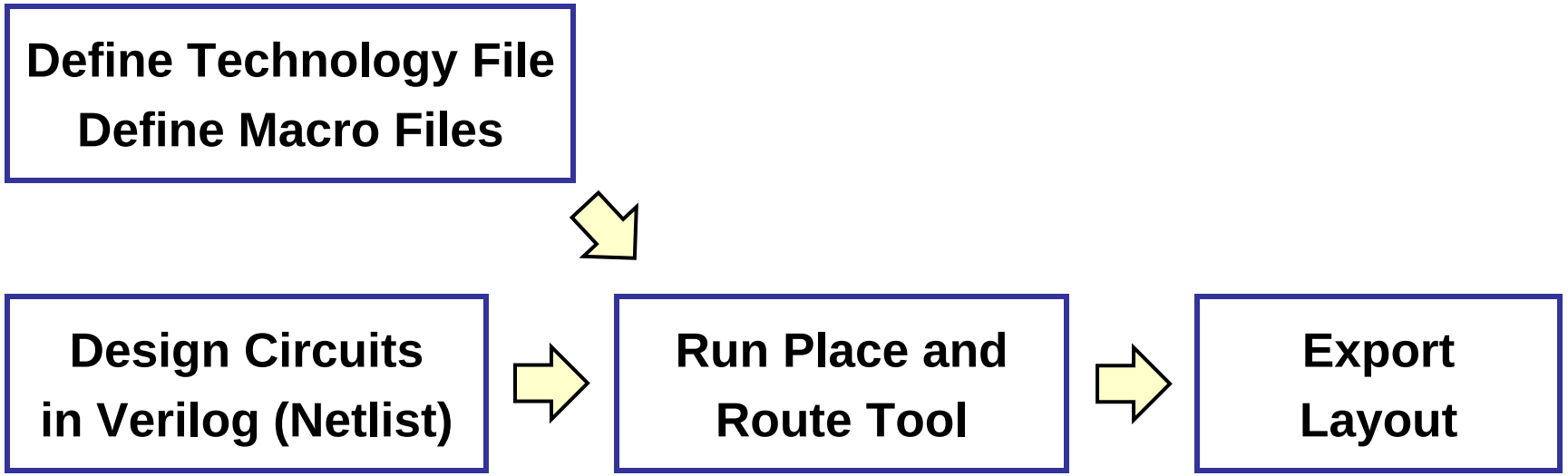
6T and Seesaw for Low Area

- 2x area reduction compared to 4T



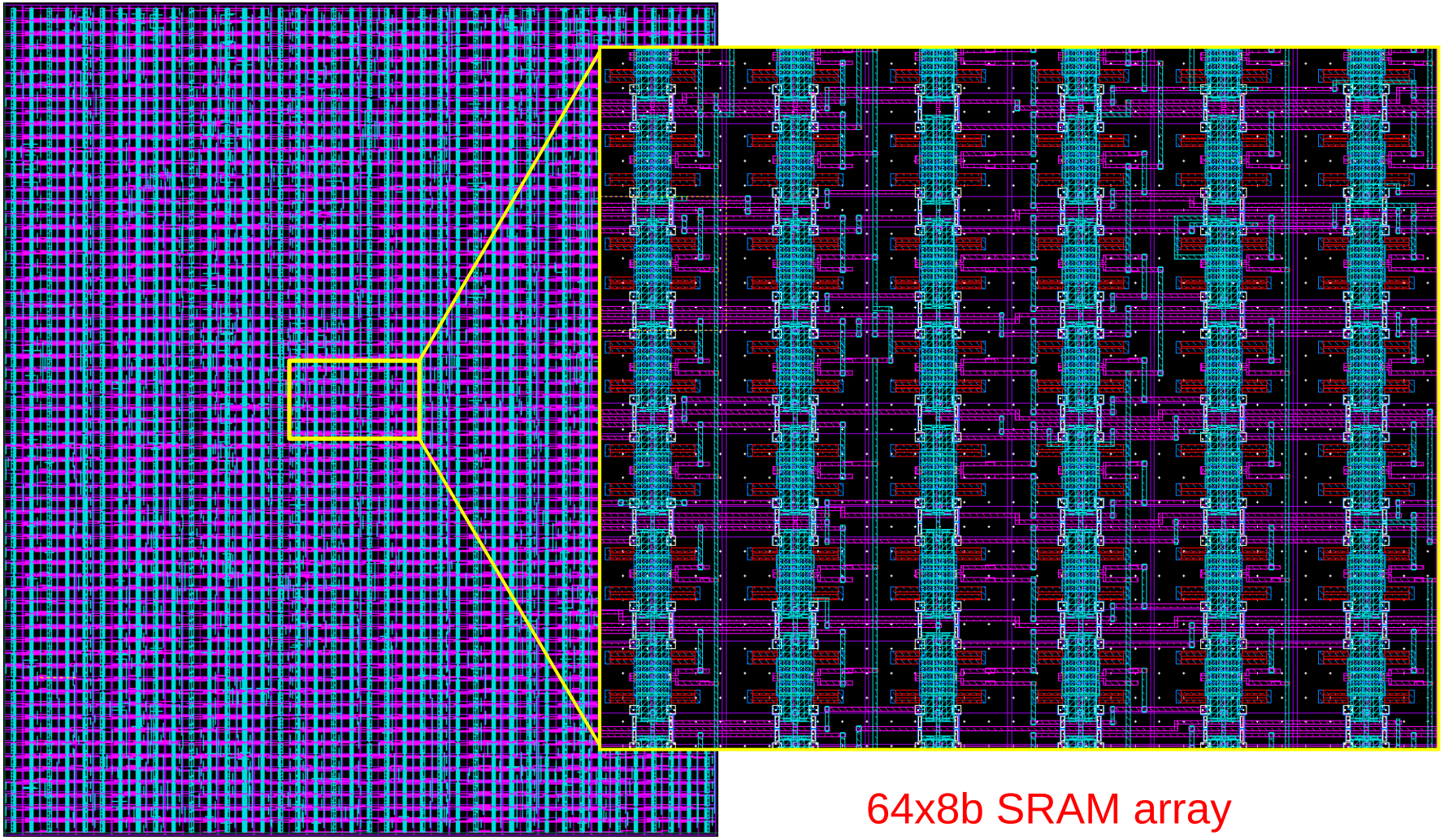
Layout Automation

- Using commercial P&R tool (Cadence Encounter)
  - Define technology file (LEF)
  - Define “standard-cell” macros (LEF)



Memory Compiler

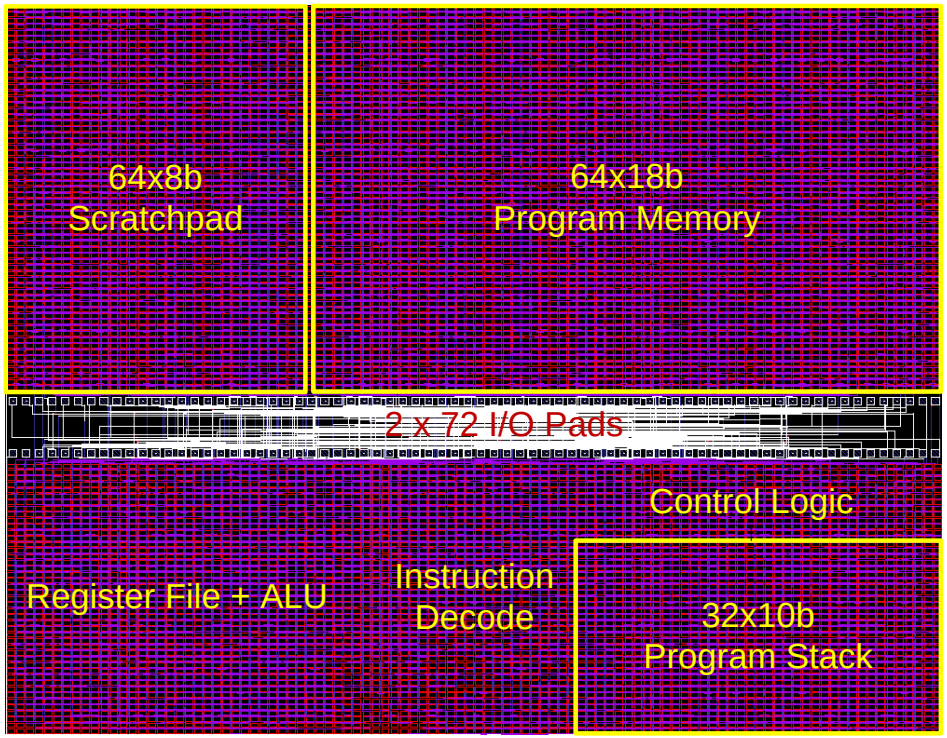
- **Auto-generate memory macros**



64x8b SRAM array

Chip Integration

- **Microcontroller design**
  - **Memory banks (80% area)**
  - **1 core**
- **Design flow**
  - **85% compiled memories**
  - **10% logic synthesis (Inst. decode)**
  - **5% custom design (ALU)**
- **Verifications**
  - **DRC, LVS, Formal**



12k relays

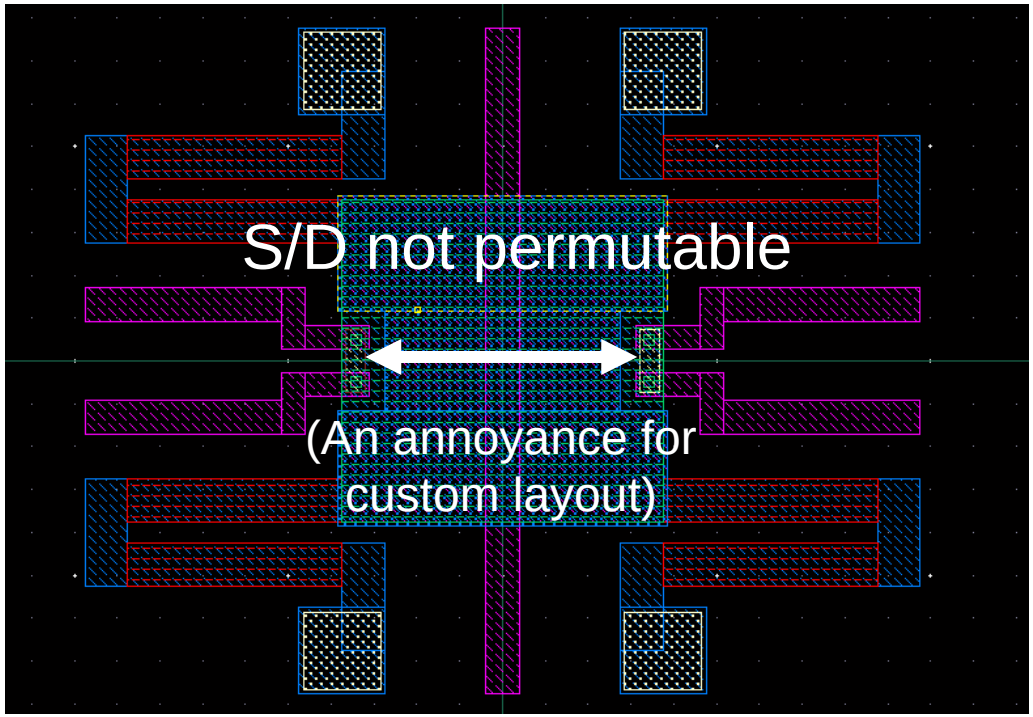
DRC Check

- **Automated DRC (compatible with Calibre)**
  - **Spacing and geometry rules**
  - **Layer-crossing release holes**

Example of Metal1 Spacing Rule

```
//*********************************
//***** METAL1 Layout Rules *****
//*********************************

m1NotGate   = METAL1 NOT GATE
m1NotDevice = (METAL1 NOT (SIZE GATE BY 50)) NOT (SIZE BODY BY 47)
m1Small     = m1NotGate WITH WIDTH <    3
m1Other     = m1NotDevice WITH WIDTH >=   3

M1Width1  {@ ERROR: Metal1 line width must be 3um or greater.
      INT m1Small < 3 ABUT>0<90 SINGULAR REGION }

M1Width2  {@ WARNING: Metal1 line width is recommended to be 10um or greater.
      INT (m1Other NOT (SIZE BODY BY 2)) < 10 ABUT>0<90 SINGULAR REGION }

M1Spacing {@ ERROR: Metal1 line spacing must be 2um or greater.
      EXT m1NotGate < 2 ABUT>0<90 SINGULAR REGION }
```

LVS Check



Use Calibre for LVS

- **Some limitations on what can be recognized/allowed (e.g. S/D permuataion)**

Conclusion

- Relay flow can leverage many features of CMOS design
- Minimization of mechanical delay and area essential
  - Biggest challenge for logic synthesis
- Modified BDD algorithm is a suitable candidate
  - Create sub-trees to eliminate inverters
  - Absorb critical-path input as S/D
  - Absorb common sub-trees through node sharing

Relay VLSI is here!