Design Flows for Optimal Power Designs

or

Toward Optimal Power Efficiency in Digital IC's

Prof. Carl Sechen

Department of Electrical Engineering

University of Texas at Dallas

Energy Efficiency of Contemporary Digital ICs

- Much of the concern these days in energy efficiency is seemingly at higher levels … which is good
- But, how much energy is wasted at the synthesis/physical design level?
  - At least 25% in dynamic power
  - At least 50% in total power
- Now that's significant!

Why So Much Energy Waste?

- Designers use the same library for both synthesis and physical design (place and route)
  - Very bad idea
  - The library that gives the best synthesis results is not the library that will give you the best post physical design results (power, delay)
- Ironically, the physical cell library can be very small, greatly reducing library upkeep
  - My observations are that cell libraries are like ant or rodent traps … once something gets in, it never gets out …

Why So Much Energy Waste? (cont'd)

- You have to make sure that each micron (ok, nm) of transistor size is actually doing something critical
  - Need an optimal and efficient optimal gate size selector, before physical design, after initial physical design and after later physical design
  - And of course you have to have appropriate drive strengths and beta ratios for the physical library cells.
- You won't want to use a V_T any higher than absolutely necessary for any cell

Toward Optimal Power Efficiency

- Major elements:
  1. Employ something that approaches an optimal synthesis library (very different from an optimal physical library)
  2. Employ something that approaches an optimal physical library
  3. Aim at near-optimal arithmetic networks (and thus must nail down THE way to implement a Full Adder (FA))
  4. Optimal continuous gate size selection
  5. Near-optimal discrete gate size selection
  6. Near-optimal V_T selection
- Tie-in with placement is important

Optimal Continuous Gate Size Selection

- First globally optimal, robust Lagrangian relaxation-based continuous gate sizer (Forge)
- Extremely accurate cell delay models are based on the actual .lib cell characterization data
- The delay we get is the same delay reported by leading STAs
- Two modes of operation
  - Generates the global minimum power for any desired delay
  - First generates the fast possible design, then the lowest power design, and finally a set of (e.g., 10-20) delay points in between, each with minimum power

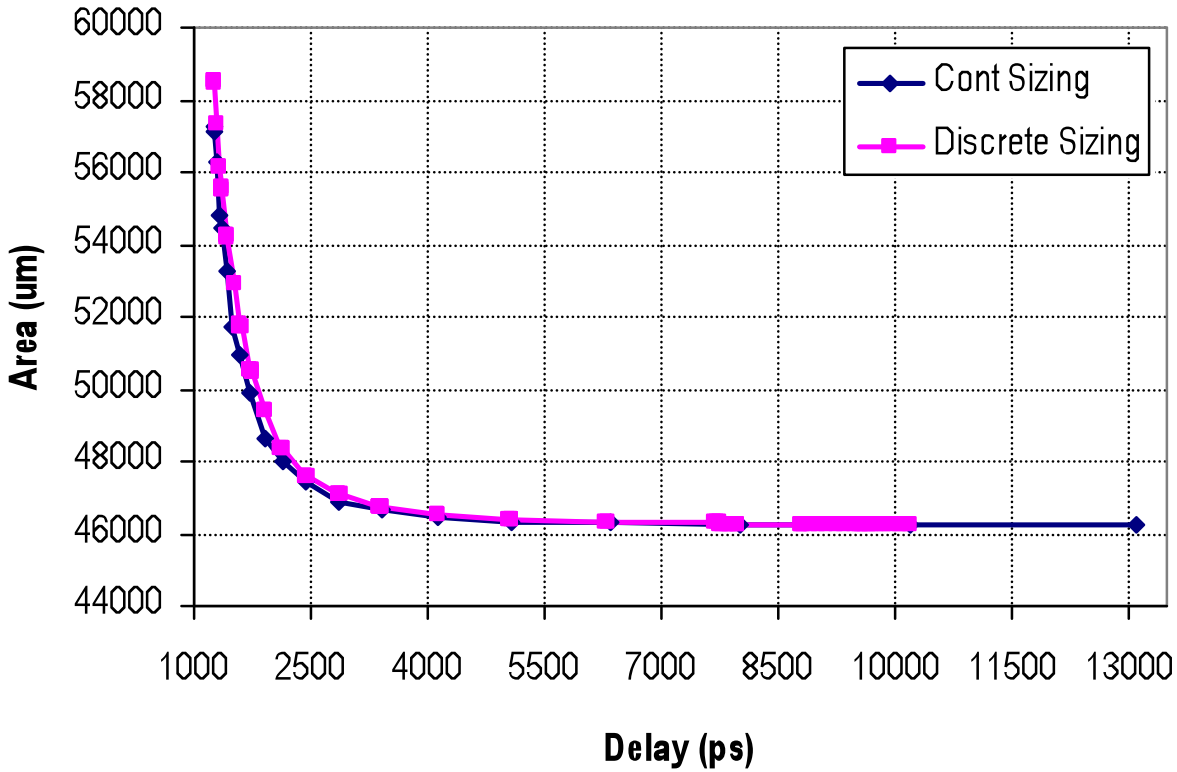
Near-Optimal Discrete Gate Size Selection

- The continuous-sized solutions are then converted to library-based, discrete gate size solutions
- Very first discrete approach to obtain results CLOSE to the global minimum continuous results
- First discrete gate sizer that nearly optimally assigns Vt's among the selections available in the library
- Handles multiple clock domains
- Handles commercial chips today, millions of cells
- Applicable to ANY structural (Verilog) netlist at any point in design flow
  - Pre layout
  - Post layout
  - With or without wire load model
  - With or without actual wiring parasitics

Results Summary

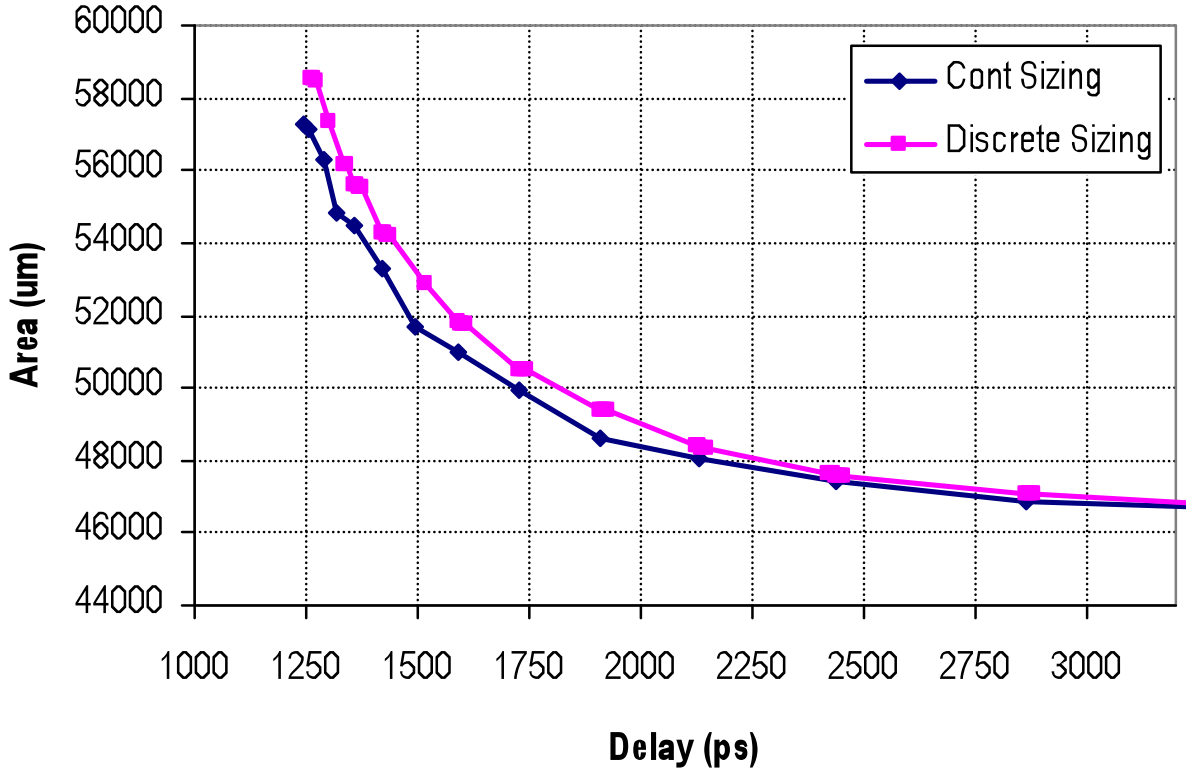
- Based on several large commercial blocks, we obtain > 35% power reduction for the same delay vs. leading EDA tools
  - > 50% leakage reduction for the same delay
  - Integrated with leading synthesis tools

Discrete Sizing – 300MHz Industrial Block



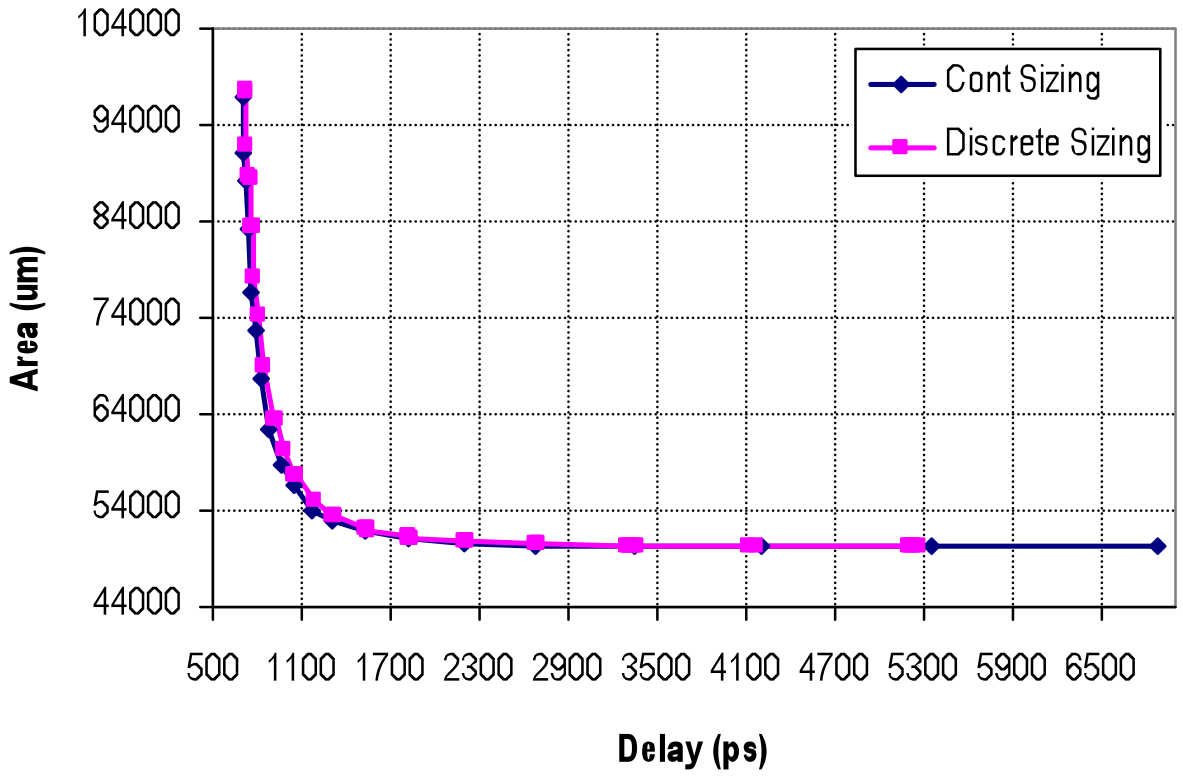
- 65 nm process – 125°C 0.9V SVT
- Total gates : 46,132 (excluding registers)

Discrete Sizing – 300MHz block (Zoomed)



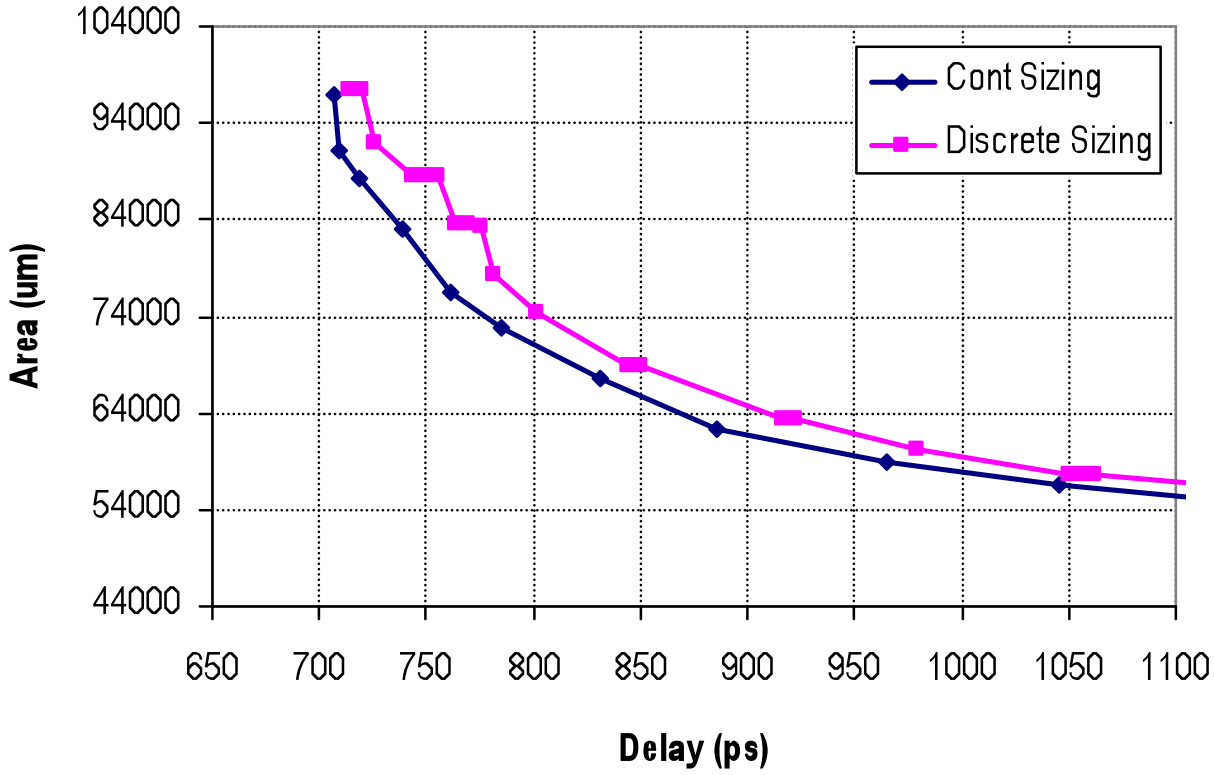
- Average area increment over continuous sizing 1.55%

Discrete Sizing – 900MHz block



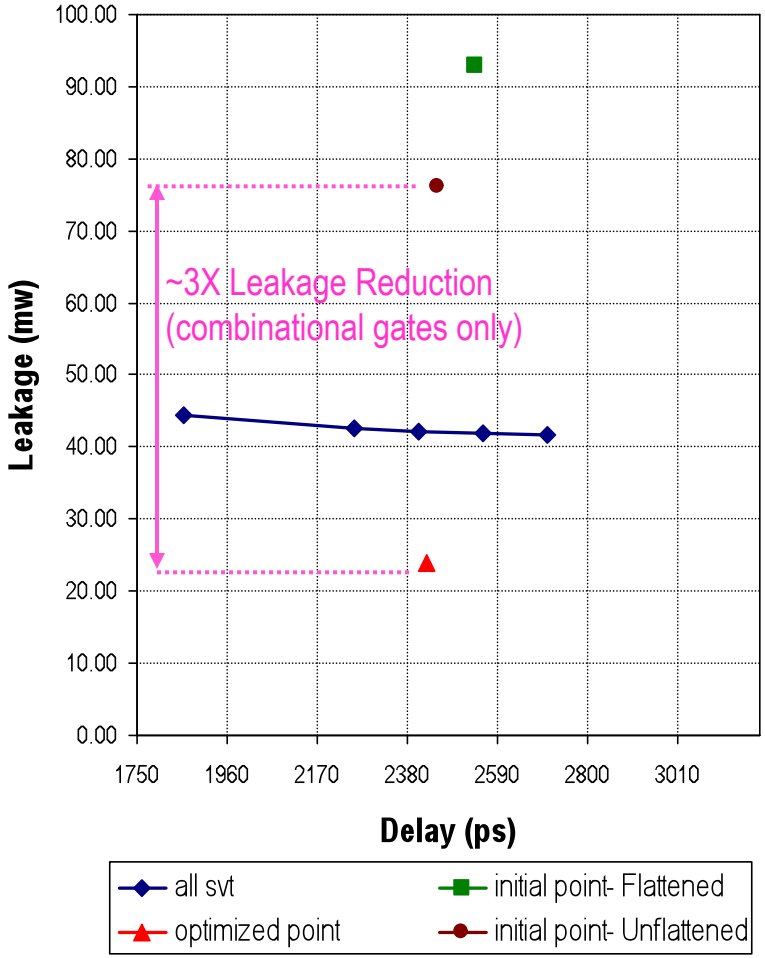
- Total gates : 49,614 (excluding registers)

Discrete Sizing – 900MHz block (Zoomed)

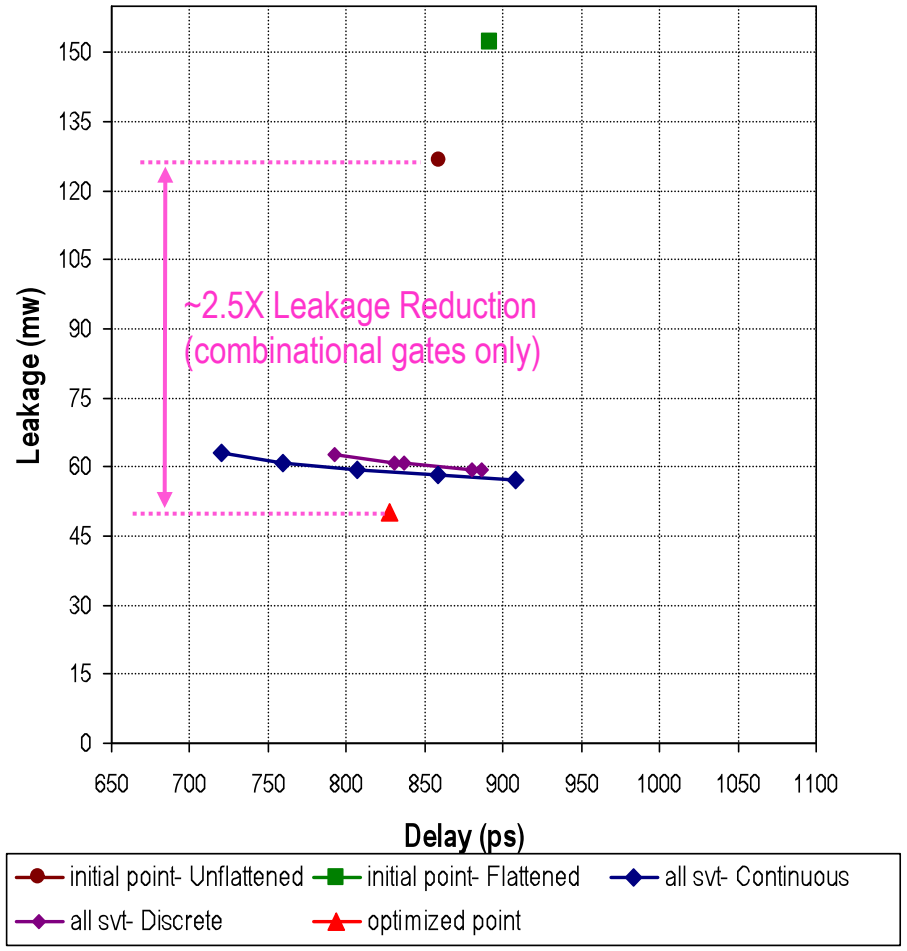


- Average area increment over continuous sizing 6.78%

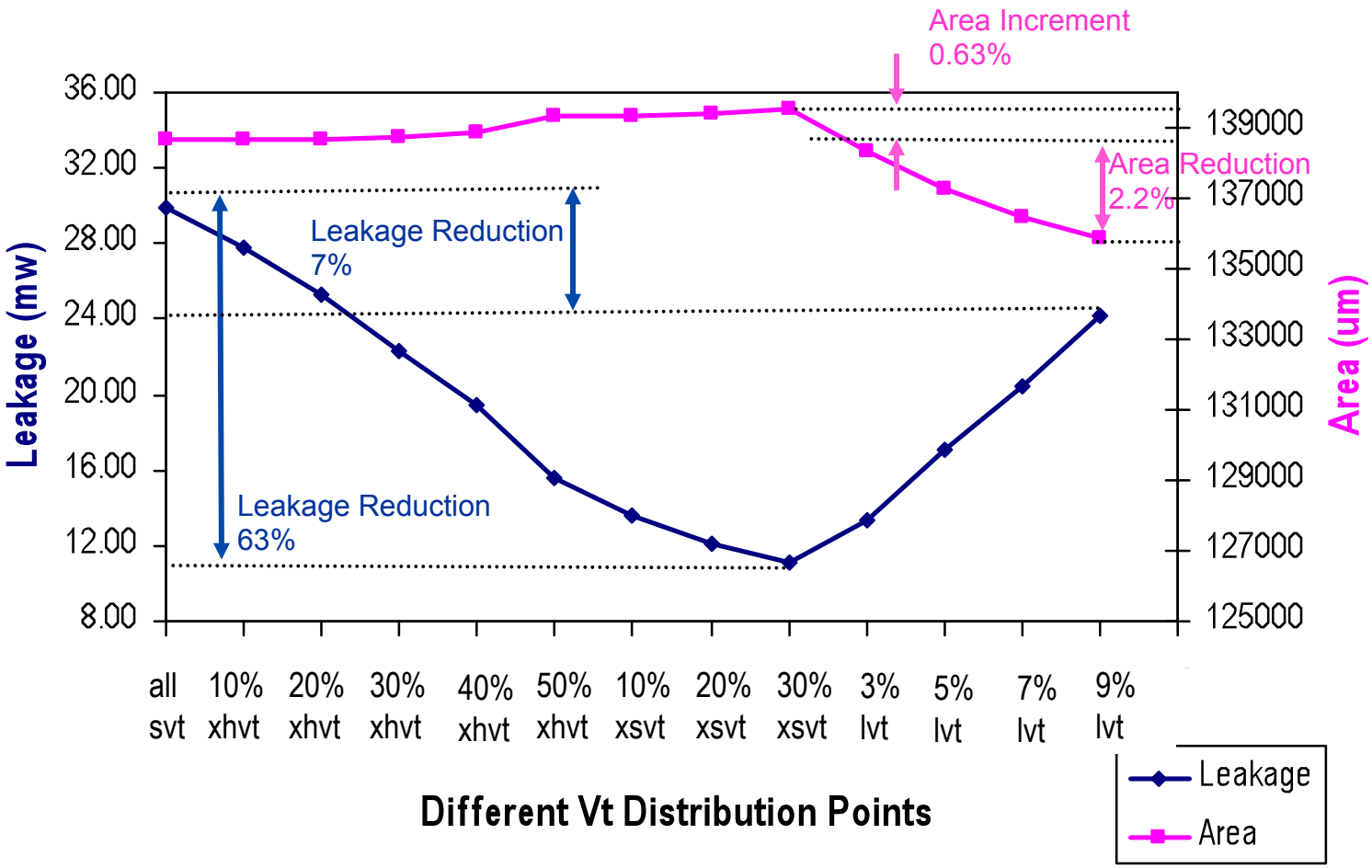
Leakage Power – 300MHz Block



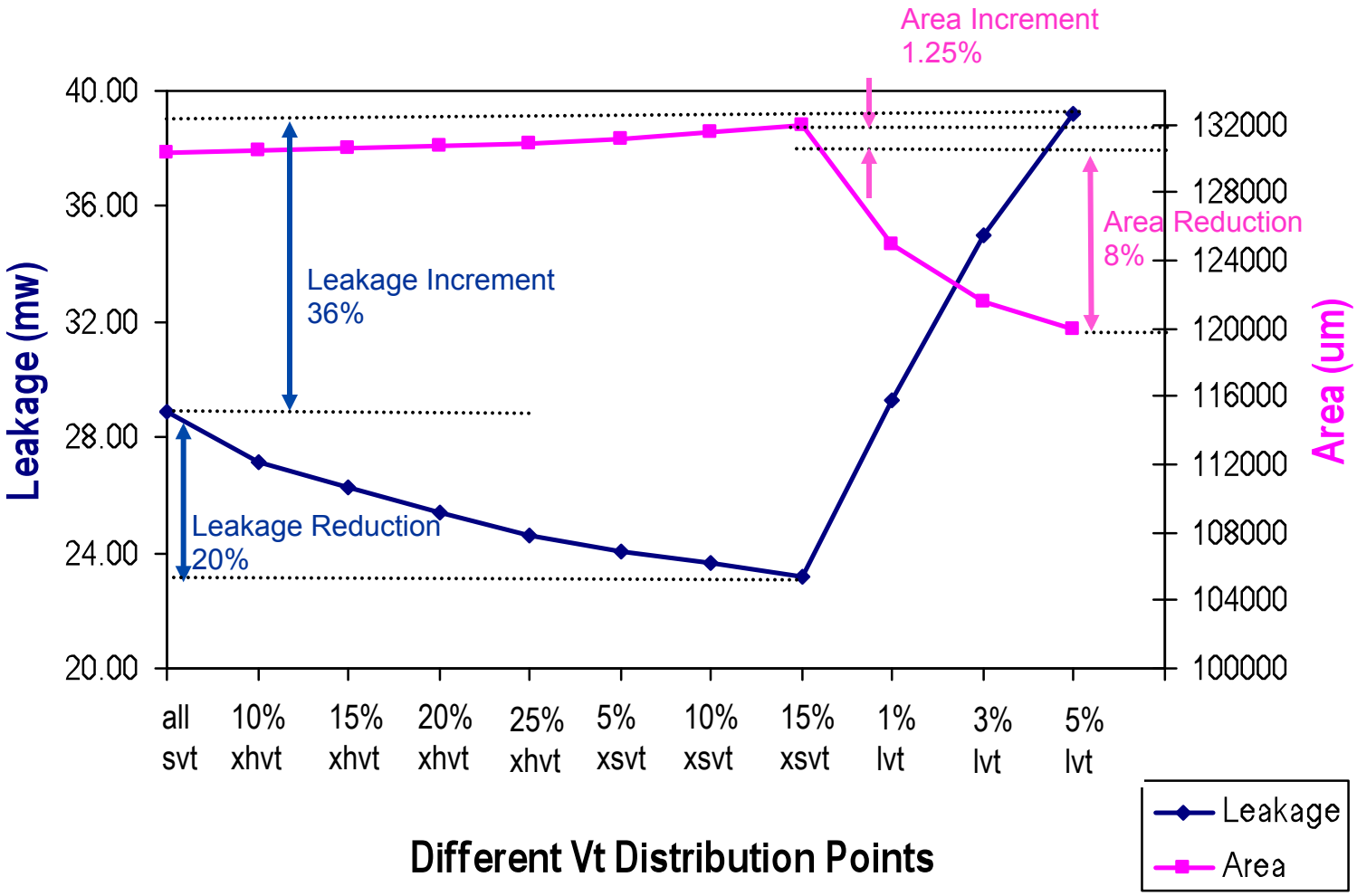
Leakage Power – 900MHz Block



V_T Selection – 300MHz Block (Delay 2250ps)



V_T Selection – 900MHz Block (Delay 750ps)



TI Benchmark

- Clock Period: 2.032 ns
- 125,000 sizeable cells
- Process: 40nm
  - Process corner
    - Delay optimization - 125^{o}C 0.9V
    - Leakage optimization - 105^{o}C 1.05V

Processing Steps

- Obtained verilog netlist from placed and routed design
- Flattened all positive unate, non-unate and complex gates
- Generated relative placement constraint for each flattened gate
  - Single row, multiple columns
- Updated component list in the .def file
- Read .def and legalize placement
- Route
- Extracted RC (wire load) for gate size selection

Sizeable Gate List

These are NOT all power efficient cells … but the “customer” used them …

- INV
- ND2
- ND3
- ND4
- NR2
- NR3
- AOI21
- AOI22
- OAI21
- OAI22
- AOI31
- AOI32
- AOI33
- AOI211
- AOI222
- OAI31
- OAI222
- AOAI211
- OAOI211
- MAJI3

Flattened Cells

Cell	Structure	Transistor Count	
		Flattened	Industrial
xnor2	nand2 + oai21	10	12
xor2	nor2 + aoi21	10	12
xnor3	xor2 + xnor2	20	22
xor3	xor2 + xor2	20	22
xnor4	xor2 + xor2 +xnor2	30	30
xor4	xor2 + xor2 + xor2	30	30
mux2	inv + aoi22 + inv	12	12
muxi2	inv + aoi22	10	10
mux3	(inv + aoi22) + (inv + inv + oai22)	22	20
mux4	(inv + aoi22) + aoi22 + (inv + oai22)	28	26
nand2B	inv + nand2	6	6
nor2B	inv + nor2	6	6

Flattened Cell Structure (Cont.)

Cell	Structure	Transistor Count	
		Flattened	Industrial
ao2bb2	nand2 + oai21	10	10
oa2bb2	nor2 + aoi21	10	10
ao21b	nand2 + nand2	8	8
mux2and2	nand2 + inv + inv + oai22	16	14
mux2or2B	inv + nor2 + inv + inv + oai22	18	16
buf	inv + inv	4	4
or2	nor2 + inv	6	6
an2	nand2 + inv	6	6
or3	nor3 + inv	8	8
an3	nand3 + inv	8	8
or4	nor2 + nor2 + nand2	12	12
an4	nand4 + inv	10	10

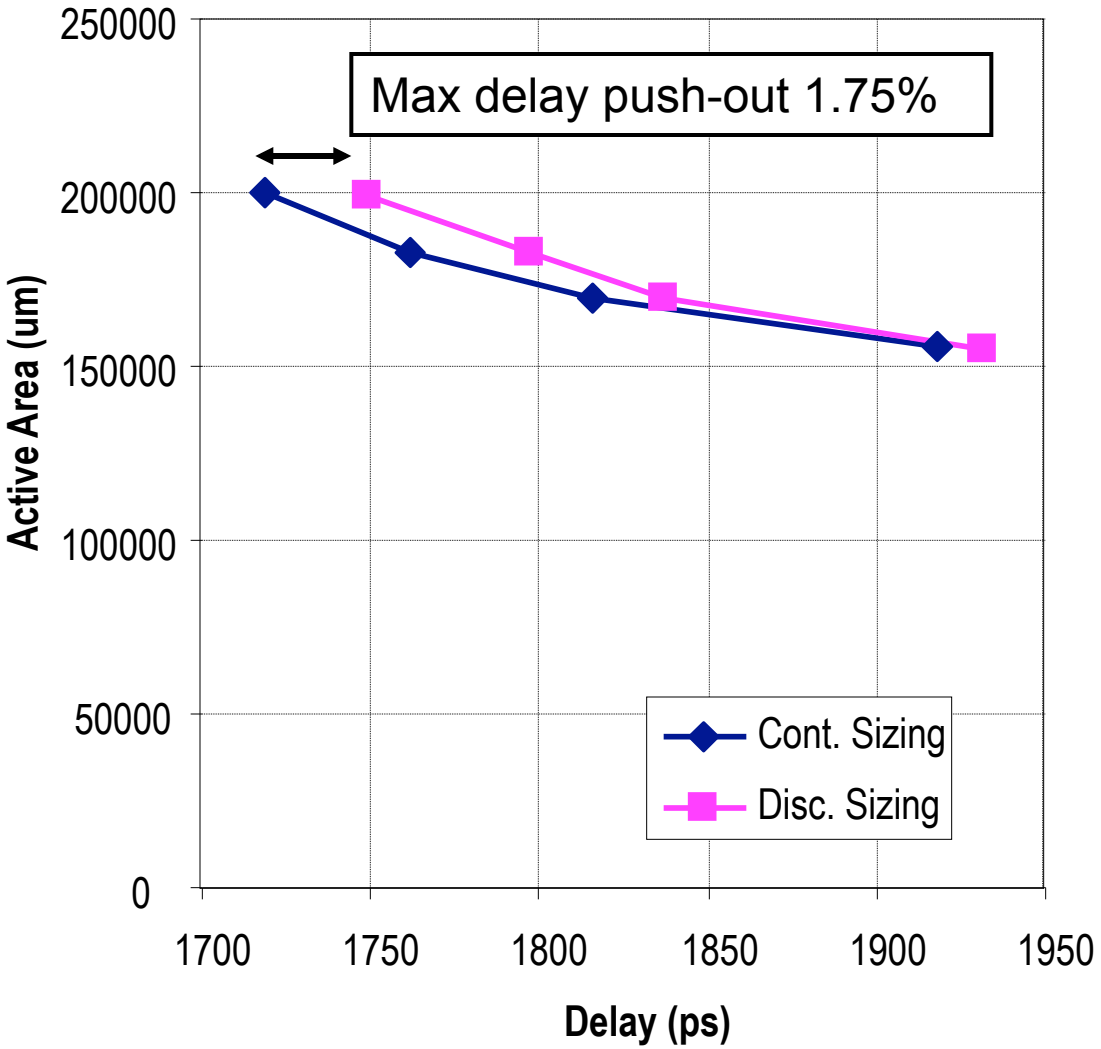
Flattened Cell Structure (Cont.)

Cell	Structure	Transistor Count	
		Flattened	Industrial
or5	nor3 + nor2 + nand2	14	14
an5	nand3 + nand2 + nor2	14	14
or6	nor2 + nor2 + nor2 + nand3	18	18
an6	nand3 + nand3 + nor2	16	16
ao21	aoi21 + inv	8	8
ao22	aoi22 + inv	10	12
ao222	aoi22 + nand2 + nand2	16	18
ao2222	aoi22 + aoi22 + nand2	20	24
ao31	aoi31 + inv	10	10
oa21	oai21 + inv	8	8
oa22	oai22 + inv	10	10
aoa211	aoai211 + inv	10	10
oao211	oaoi211 + inv	10	10

Flattened Cell Structure (Cont.)

Cell	Structure	Transistor Count	
		Flattened	Industrial
nand3B	inv + nand3	8	8
nor3B	inv + nor3	8	8
addh	xor2 + and2	16	16
addf	xor2, xor2, maji3, inv	32	32
addf42	addf + addf	56	56

Continuous vs. Discrete Sizing Result



V_T Assignment

V_T Type	Combinational Gates	Registers
Extended High V_T	47%	0%
High V_T	1%	0%
Extended Standard V_T	37%	100%
Standard V_T	14%	0%

Results

- **Performance**

	Original Base Line	Forge
Period (ps)	1950	1768

- **Power**

	Original Base Line	Forge
P_lkg comb (mW)	95.4	29.3
P_lkg_seq (mW)	22.3	9.7
Active_area (um)	317375	212594

Results (cont.)

	Original Base Line	Forge
w_gates (um)	317375	212594
cell_count	103494	125943
L_horiz_wire (um)	1400990	1434343
L_vert_wire (um)	1698424	1697836

Power Optimal Cell Library

- INV
- NAND2
- NAND3
- NOR2
- NOR3
- AOI21
- AOI22
- OAI12
- OAI22

ONLY 9 logic cells
(easy to maintain library)

- Plus **ONE** D-FF for each V_T !!!
  - (Not counting scan-related cells …)

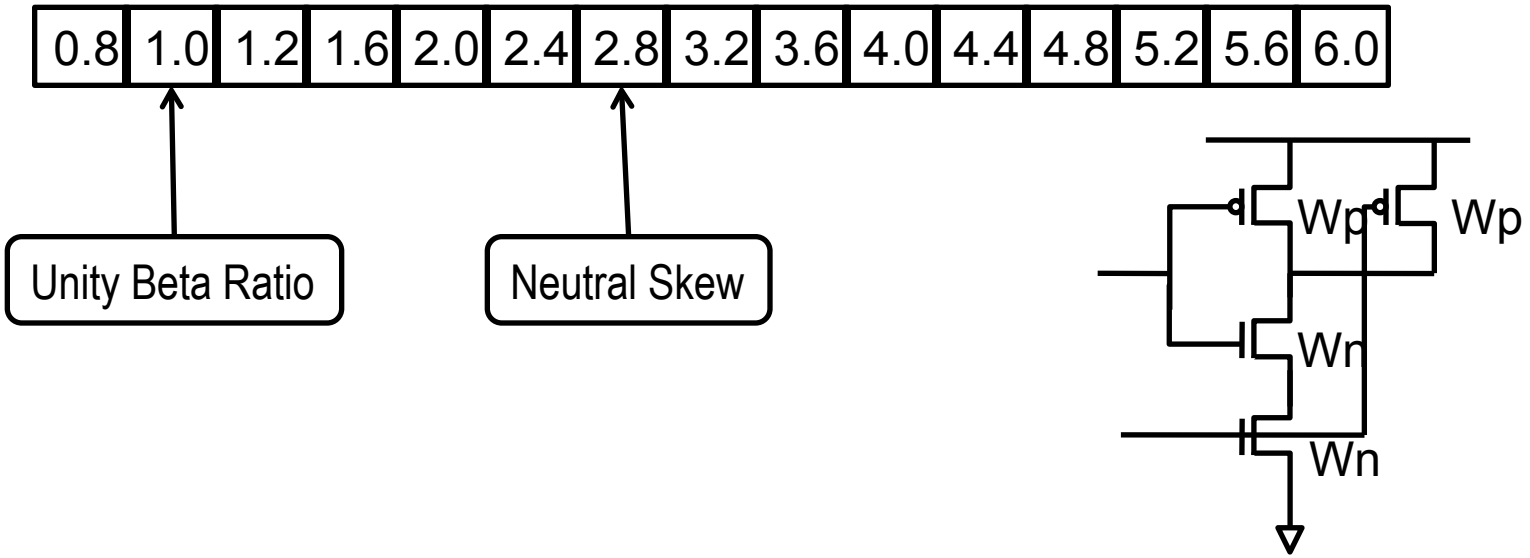
Power Optimal Cell Library (cont'd)

- INV
- NAND2
- NAND3
- NOR2
- NOR3
- AOI21
- AOI22
- OAI12
- OAI22

- Ok, only 9 sizeable cells
- Ah, but how many drive strengths?
- And how many beta ratios?
- Yeah, but what about the explosion in "global nets" with such "tiny" cells?
- Good questions … we'll take them one at a time

Our Base Standard Cell Library (LIB)

- IBM 130nm 1.2V 25C process technology
- Wn's in [0.28um, 7.84um], steps of 0.28um
- Wp's are any Wn times any of the beta ratios
- The full set of beta ratios (Wp/Wn) is:



- LIB is our full library and our objective is to find the smallest subset of these size and beta alternatives that yields similar power-delay curves

Same Effective Beta Range for Other Gates

GATE	β_{min}	$\beta_{neutral}$	β_{max}	Total Betas
INV	0.8	2.8	6.0	15
AOI21	0.8	2.8	6.0	16
OAI21	0.8	2.8	6.0	16
AOI22	0.8	2.8	6.0	15
OAI22	0.8	2.8	6.0	15
NAND2	0.4	1.4	3.0	15
NOR2	1.0	5.6	12.0	15
NAND3	0.26	1.0	2.2	14
NOR3	1.0	8.4	18.0	15

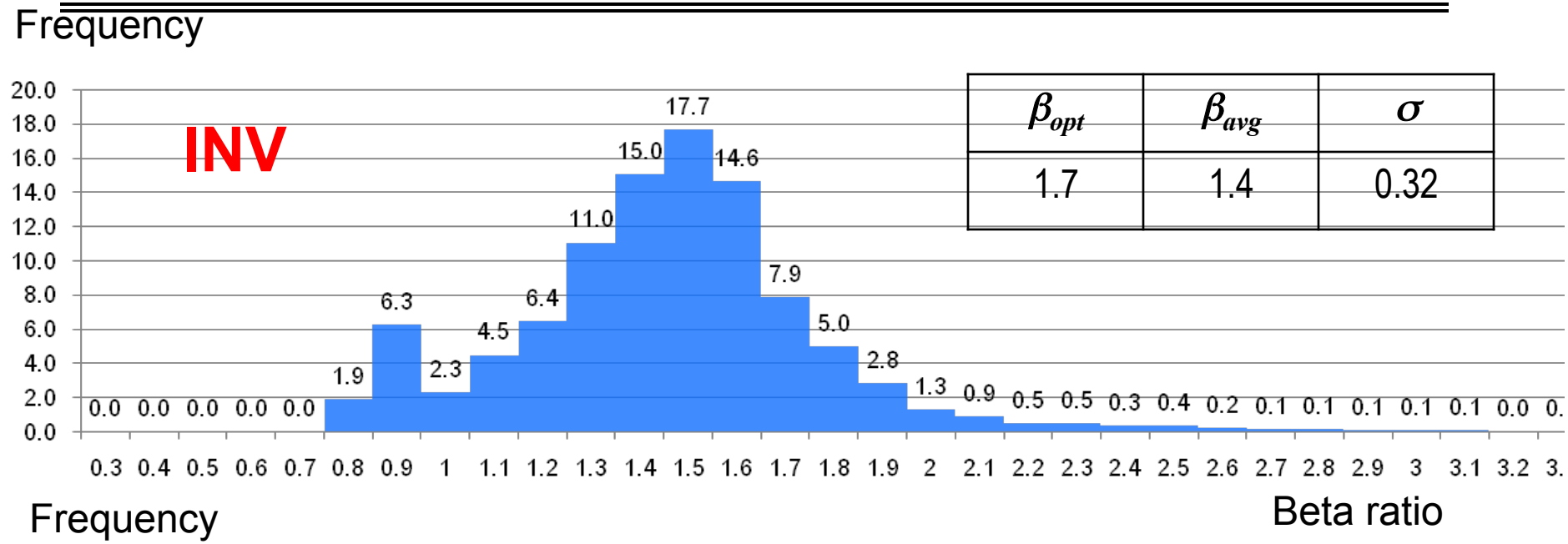
0.8	1.0	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0

Experimental Setup

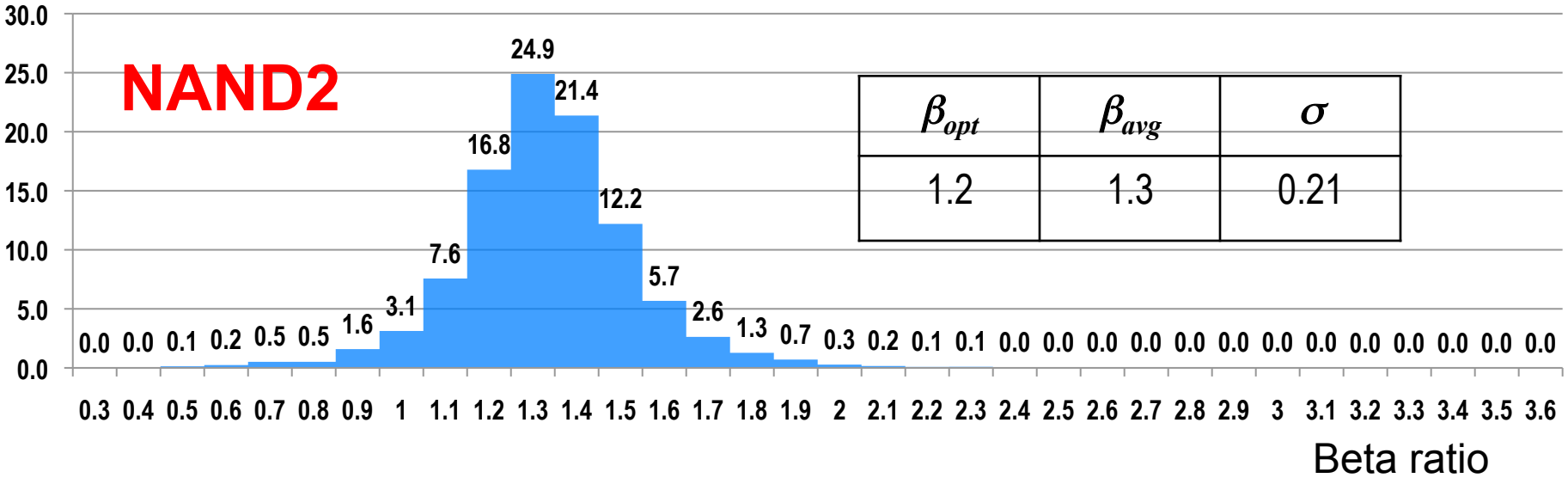
- Synthesize benchmarks with the leading commercial synthesis tool using the full library (LIB)
  - With wire load model
- Generate the continuous power-delay curves for all benchmarks
  - Select three points on the continuous curve for each benchmark: min delay and two points in the “knee” of the curve
- Then plot beta density function

$$\beta_{opt} = \sqrt{\frac{M_P \mu_N}{M_N \mu_P}}$$

Beta Density Functions

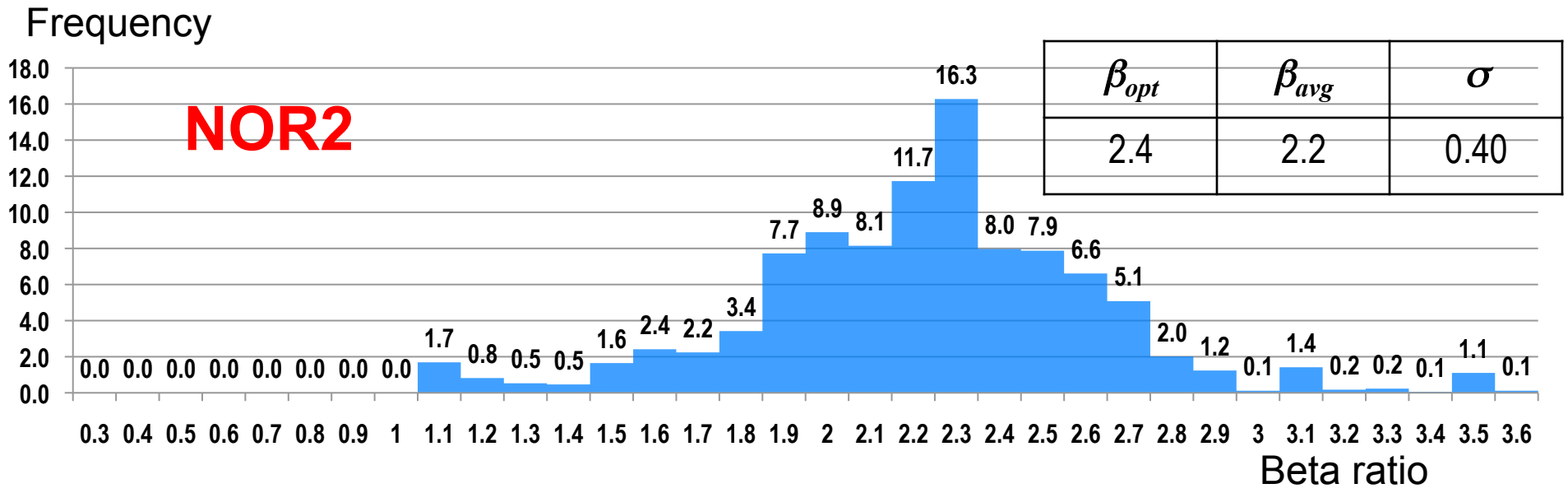


β_{opt}	β_{avg}	σ
1.7	1.4	0.32

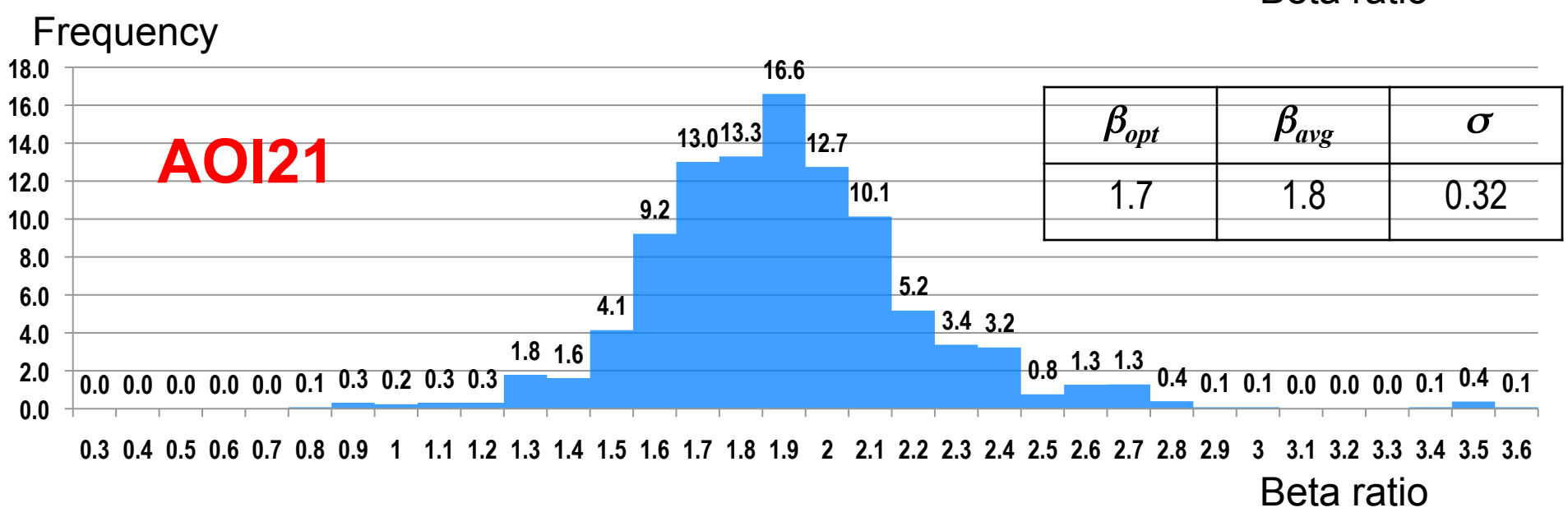


β_{opt}	β_{avg}	σ
1.2	1.3	0.21

Beta Density Functions (cont'd)



β_{opt}	β_{avg}	σ
2.4	2.2	0.40



β_{opt}	β_{avg}	σ
1.7	1.8	0.32

Which Cell Sizes Should We Pick?

- Generate 5 test discrete libraries
  1. LIB: full discrete library
  2. GEO: Geometrically spaced Wn sizes: 0.28um, 0.56um, 1.12um, 2.24um, 4.48um along with all beta options 0.5X, 1X, 2X, 4X, 8X drives
  3. LIN: Linearly spaced Wn sizes: 0.28um, [0.56um-2.24um] in steps of 0.56um along with all beta options 0.5X, 1X, 2X, 3X, 4X
  4. 3SIGMA: LIN but with discrete betas restricted to +/- 3 sigma of the mean beta computed for each gate
  5. 1.5SIGMA: LIN but with discrete betas restricted to +/- 1.5 sigma of the mean beta computed for each gate
     - Beta of 1 included in 3SIGMA and 1.5SIGMA for down sizing off critical paths
  6. 1BETA: Linearly spaced as above but with a single beta ratio
     - Beta chosen as the closest to the computed mean beta

0.8	1.0	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0	4.4	4.8	5.2	5.6	6.0

Library Beta Values

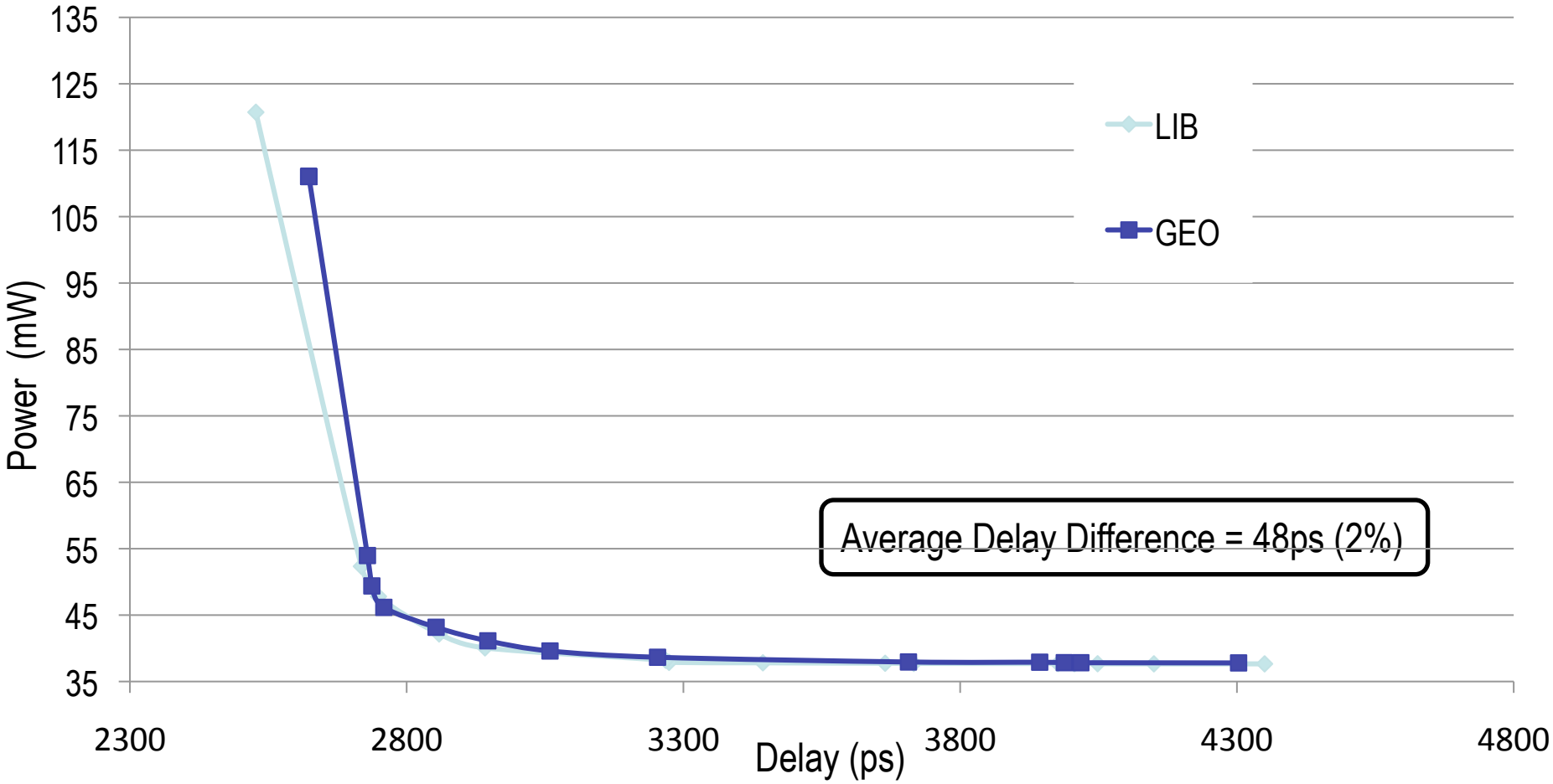
Gate	3SIGMA	1.5SIGMA	1BETA
INV	0.8 1.0 1.2 1.6 2 2.4	1.0 1.2 1.6 2.0	1.6
NAND2	0.6 0.8 1.0 1.2 1.4 1.6 1.8	1.0 1.2 1.4	1.2
NOR2	1.0 1.6 2.4 3.2 4.0	1.0 1.6 2.4 3.2	2.4
NAND3	0.6 0.7 0.86 1.0 1.16 1.31 1.46 1.61	0.86 1.0 1.16	1.16
NOR3	1.0 2.4 3.6 4.8	1.0 2.4 3.6	2.4
AOI21	0.8 1.0 1.6 2.4 3.2	1.0 1.6 2.4	1.6
OAI21	0.8 1.0 1.2 1.6 2.0 2.4	1.0 1.2 1.6 2.0	1.6
AOI22	0.8 1.0 1.2 1.6 2.0 2.4	1.0 1.2 1.6 2.0	1.6
OAI22	0.8 1.0 1.2 1.6 2.0 2.4	1.0 1.2 1.6 2.0	1.0

Cell Count for Each Library

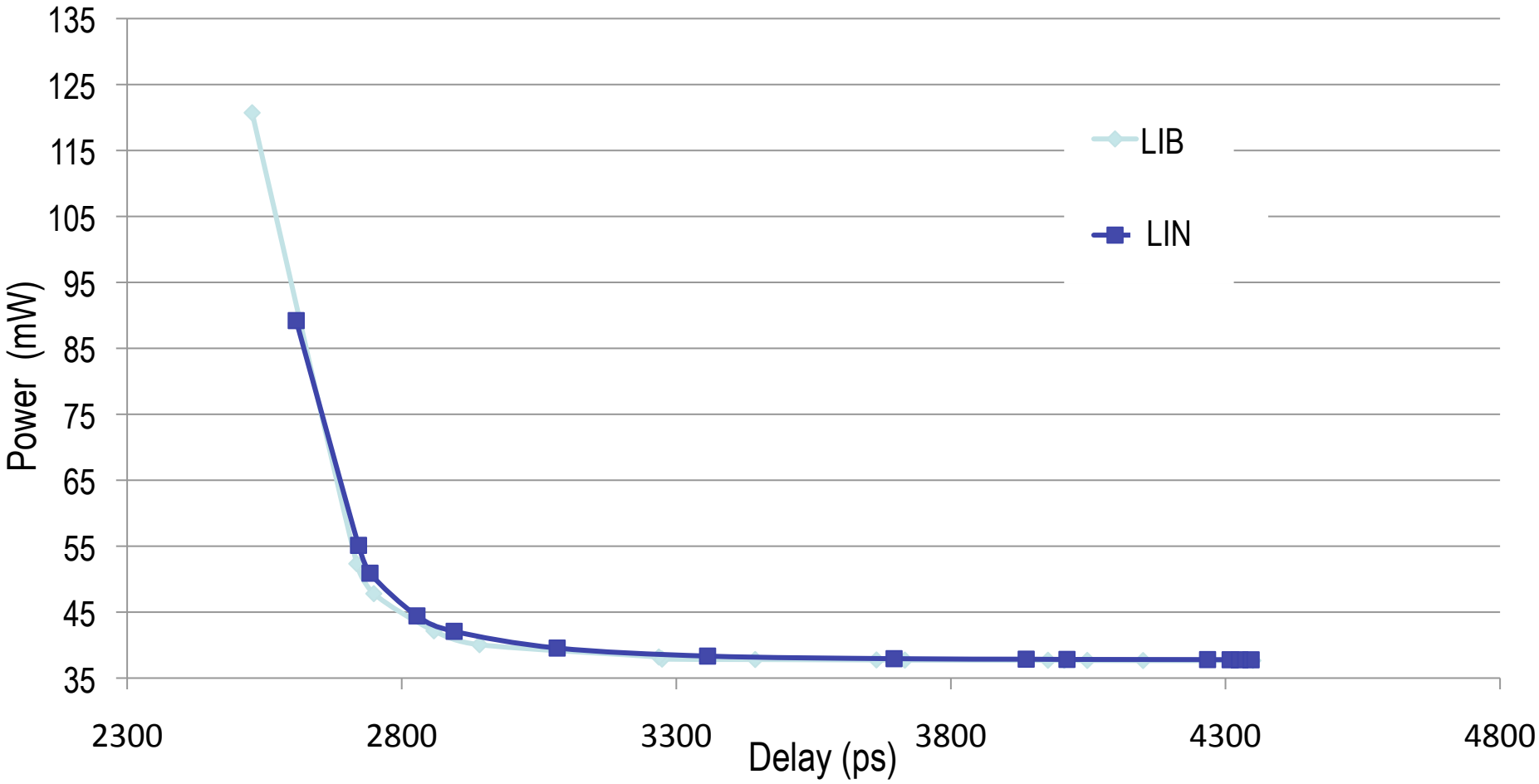
Gate	Number of Cells					
	LIB	GEO	LIN	3SIGMA	1.5SIGMA	1BETA
INV	420	75	75	30	20	5
NAND2	420	75	75	35	15	5
NOR2	420	75	75	25	20	5
NAND3	392	70	70	40	15	5
NOR3	420	75	75	20	15	5
AOI21	448	80	80	25	15	5
OAI21	448	80	80	30	20	5
AOI22	420	75	75	30	20	5
OAI22	420	75	75	30	20	5
TOTAL	**3808**	**680**	**680**	**265**	**160**	**45**

Benchmark *b20* Mapped to LIB and GEO

- Map points on the continuous power-delay curves to each of the 6 libraries – LIB, GEO, LIN, 3SIGMA, 1.5SIGMA, and 1BETA
- Measure performance deviation with respect to LIB

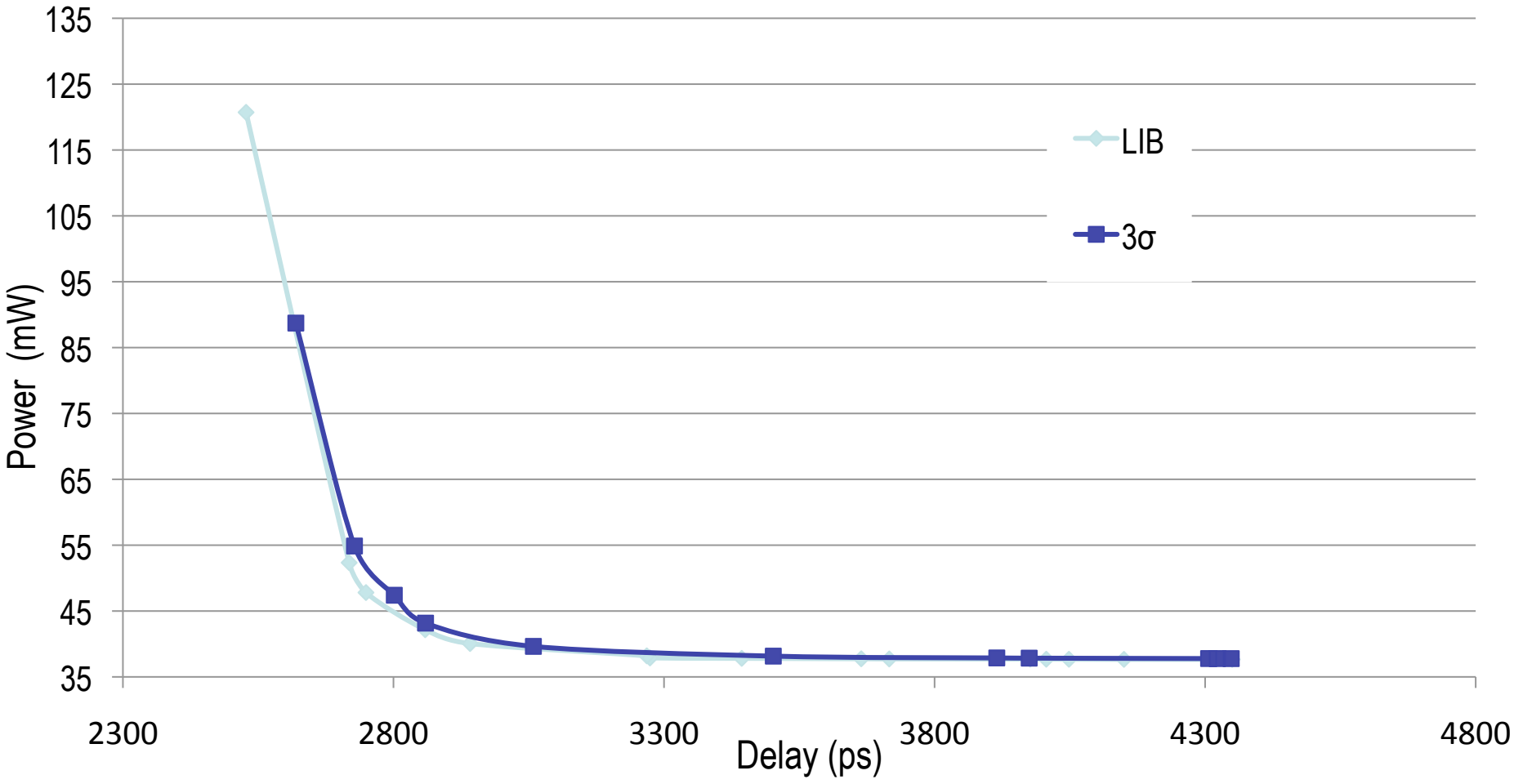


Benchmark *b20* Mapped to LIB and LIN



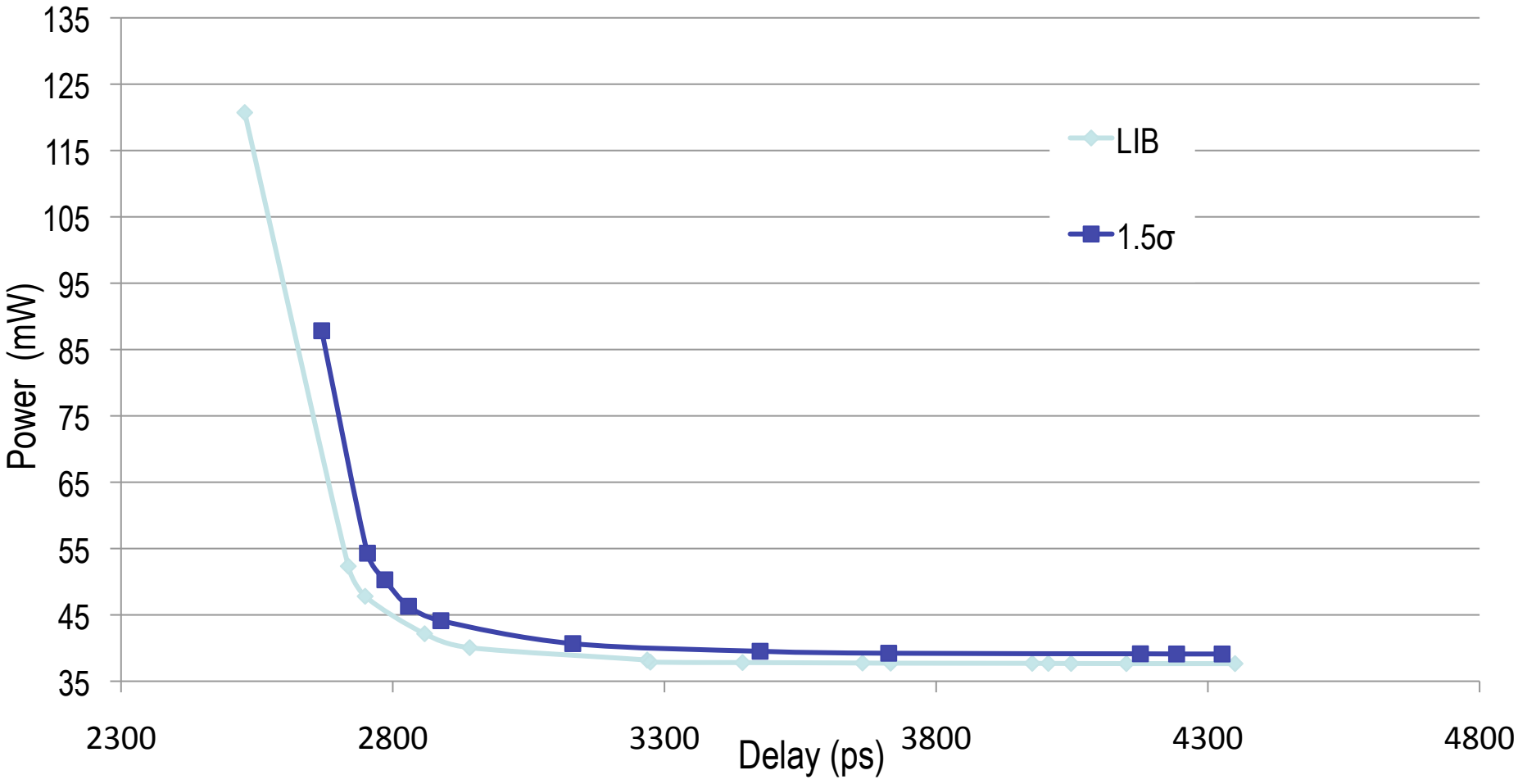
Average Delay Difference = 24ps (0.9%)

Benchmark *b20* Mapped to LIB and 3SIGMA



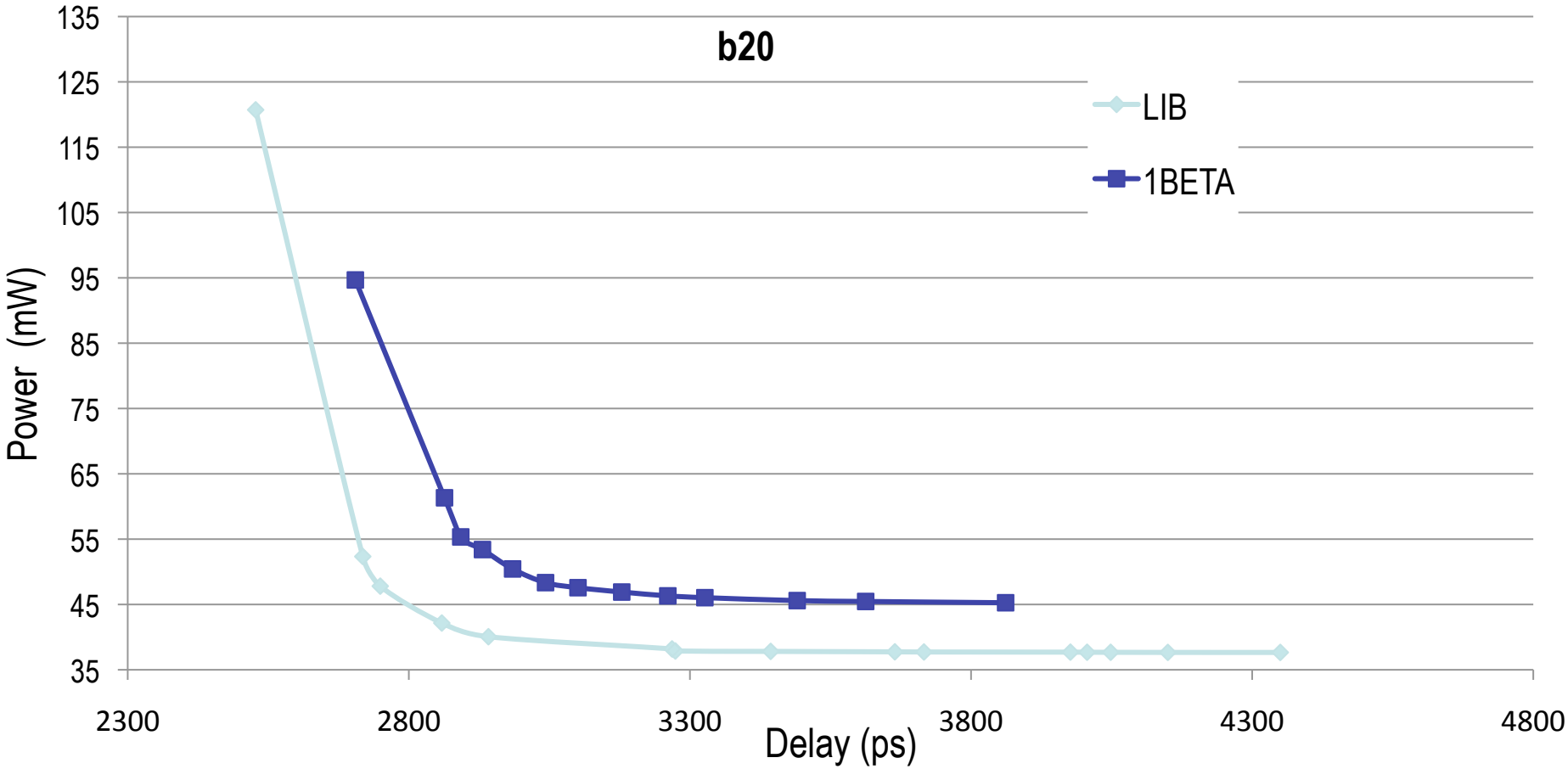
Average Delay Difference = 26ps (1.0%)

Benchmark *b20* Mapped to LIB and 1.5SIGMA



Average Delay Difference = 45ps (1.7%)

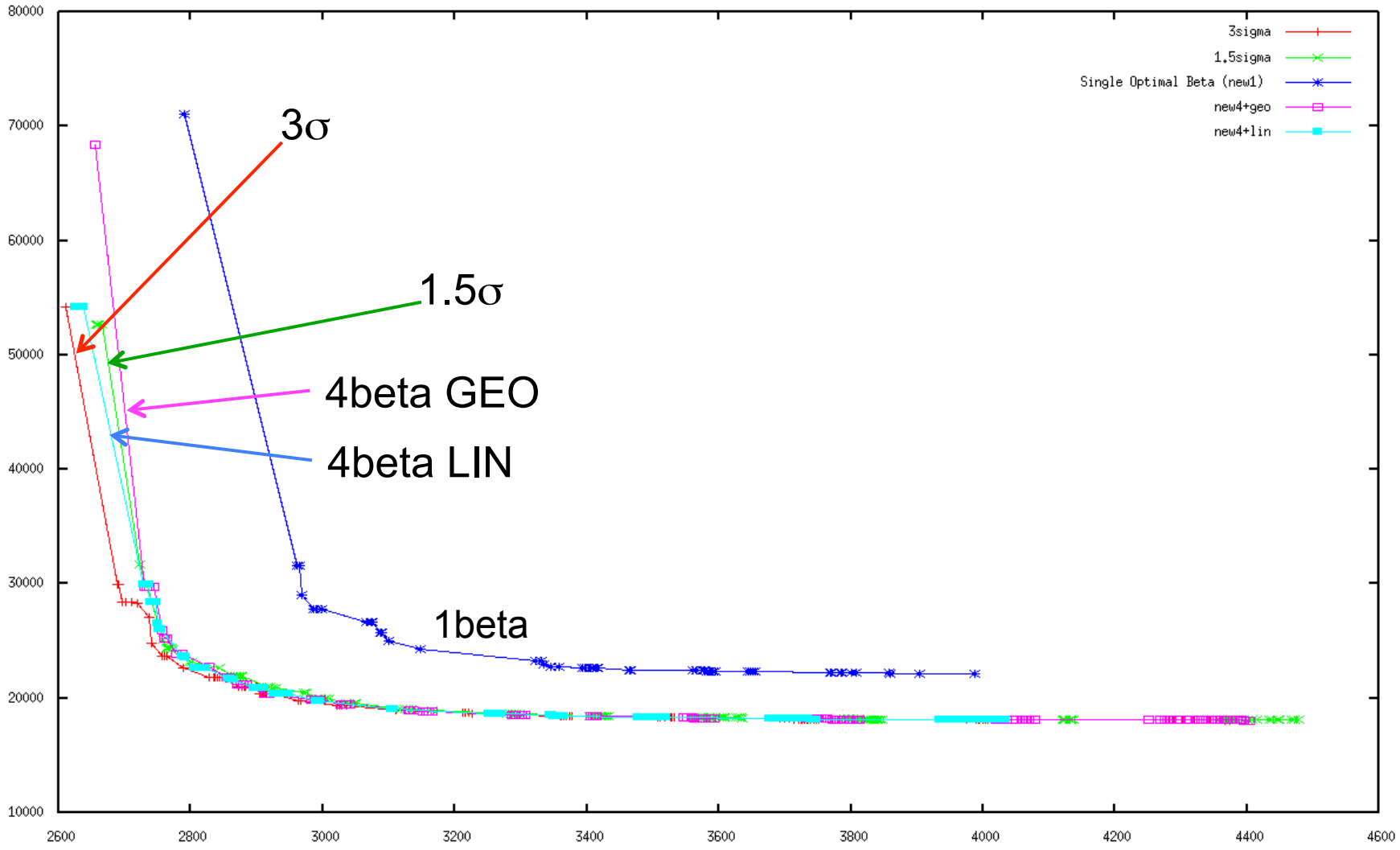
Benchmark *b20* Mapped to LIB and 1BETA



Average Delay Difference = 232ps (8.9%)

Average Power Difference = 10.7mW (24%)

40nm TI Process 1.0V 20k Cells



Library Conclusions

- Very low skew gates and high skew gates are rarely selected
- 3SIGMA library consisting of drive strengths 0.5X ,1X, 2X, 3X and 4X and beta ratios within $\beta_{opt} \pm 3\sigma$ (265 cells)
  - 14X smaller library compared to LIB
  - 0.8% performance loss
- 1.5SIGMA library consisting of drive strengths 0.5X ,1X, 2X, 3X and 4X and beta ratios within $\beta_{opt} \pm 1.5\sigma$ (160 cells)
  - 25X smaller library compared to LIB
  - 1.5% performance loss
- 1BETA library consisting of drive strengths 0.5X ,1X, 2X, 3X and 4X and a single beta ratio (45 cells)
  - 24% increase in power
  - 9.1% increase in delay

Global Net Issue

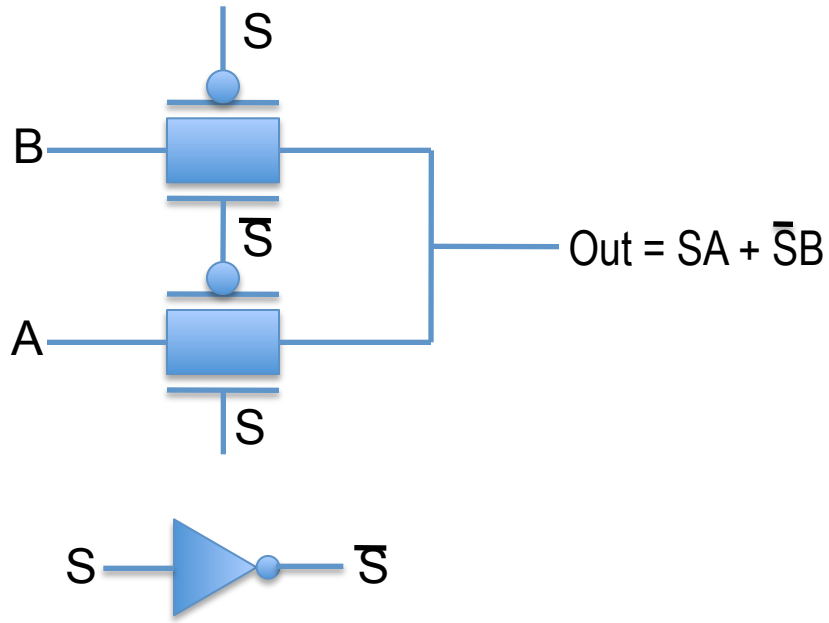
- Cell grouping in placement effectively knocks out the would-be extra global nets

Pass Transistor Logic (PTL)

- Pass transistor logic based cells should **NEVER** be in a standard cell library
- Provably no benefit whatsoever
- How so?
  - Let's go on a short tour …

Pass Transistor Logic (PTL)

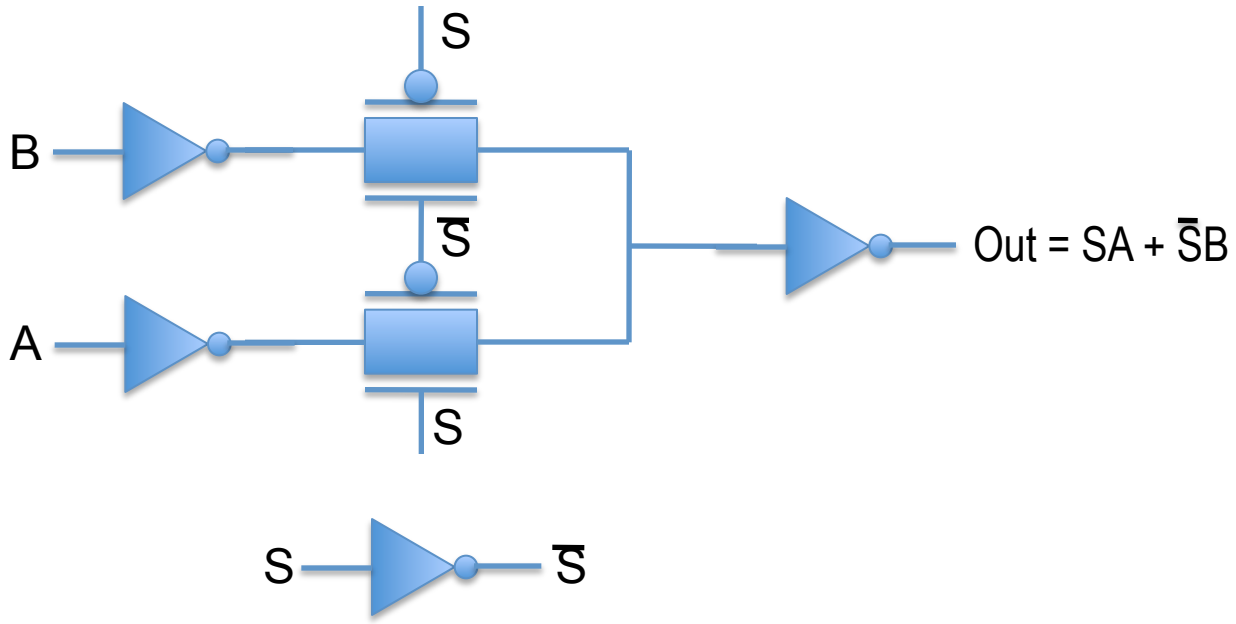
- Seemingly very efficient and fast for 2:1 multiplexors



- Only six transistors
- But, very poor layout efficiency, with several diffusion breaks
  - Standard cell width is considerably larger than it would be for a series-parallel six-transistor gate

Modern PTL

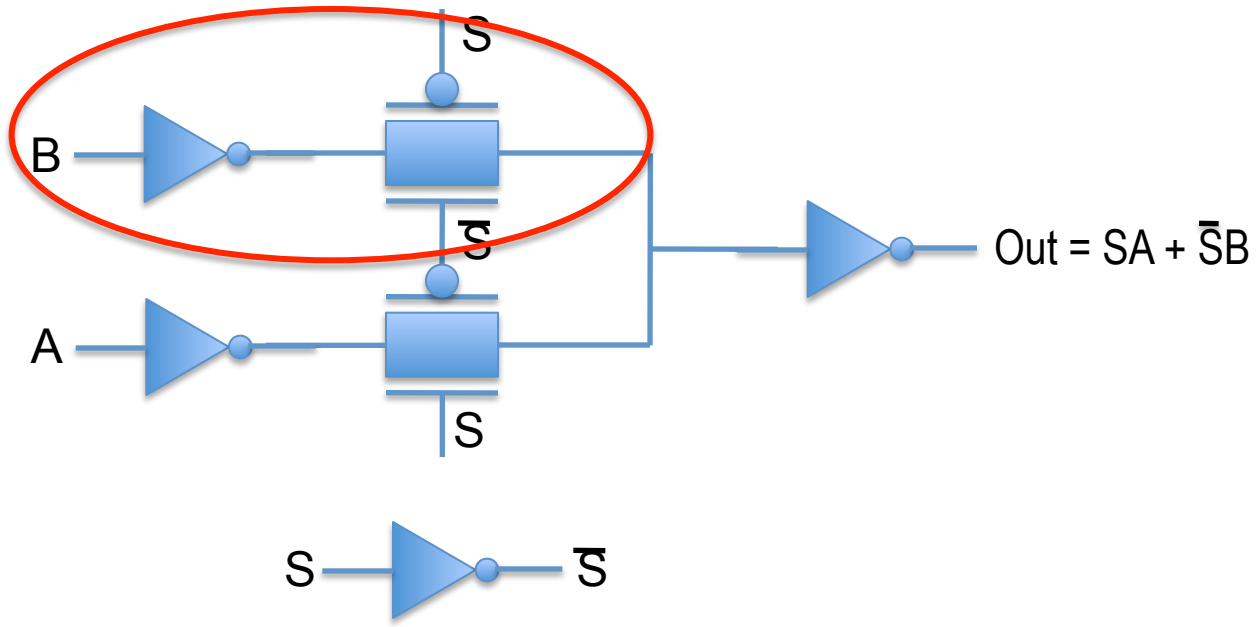
- Static timing analysis (STA) tools demand purely capacitive inputs
- The PTL MUX had to re-designed as follows:



- Now 12 transistors!
- Plus diffusion breaks and poor layout efficiency

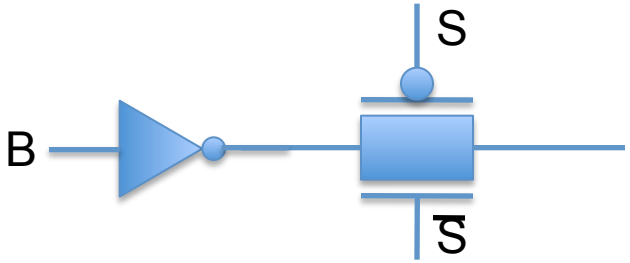
PTL MUX -- Closer Look

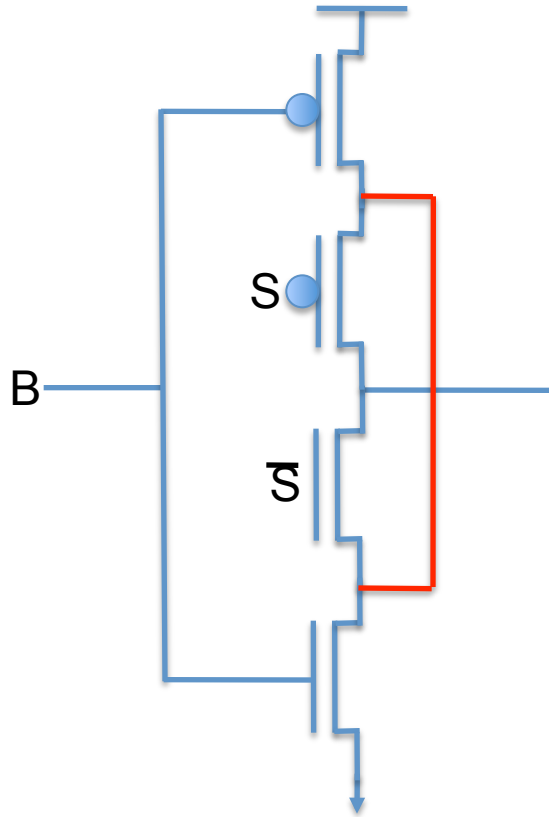
- Let's look at the encircled inverter-transmission gate pair



Inverter T-Gate Pair

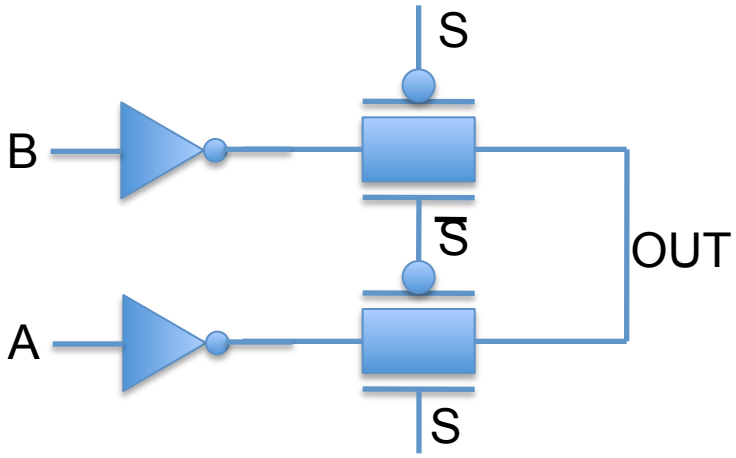
- The following two circuits are absolutely identical
- But, the red wire is of no use
  - It can only be used to help pull down the output by passing the 0 (gnd) through the pMOS (with gate input S), which is not effective (easily verified)
  - Similarly for the pull up case

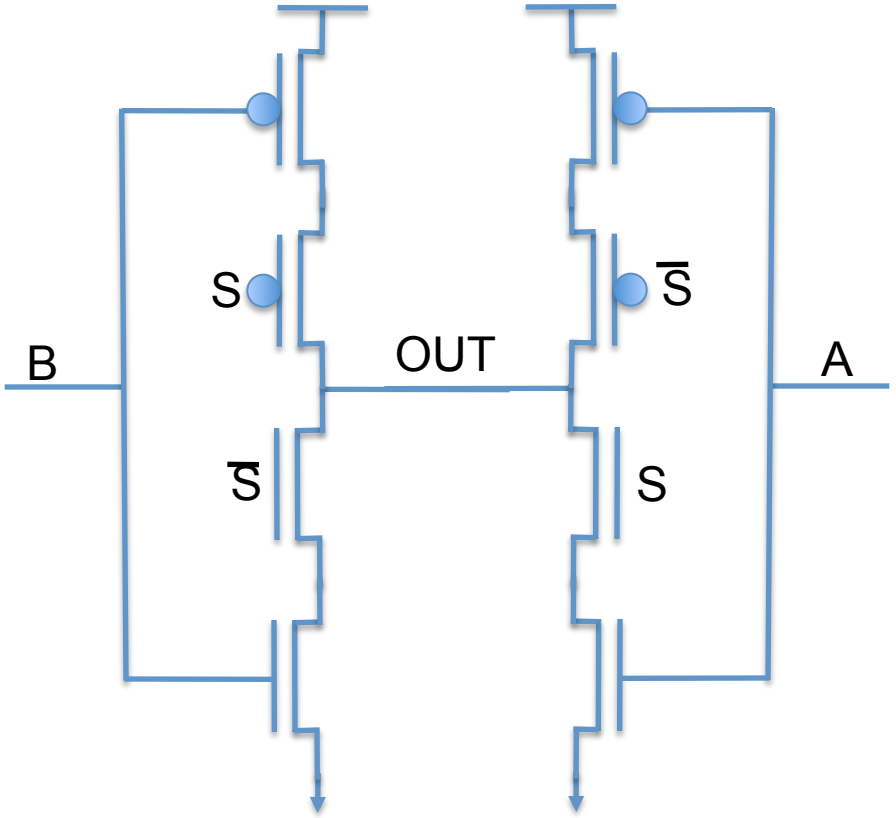




PTL MUX

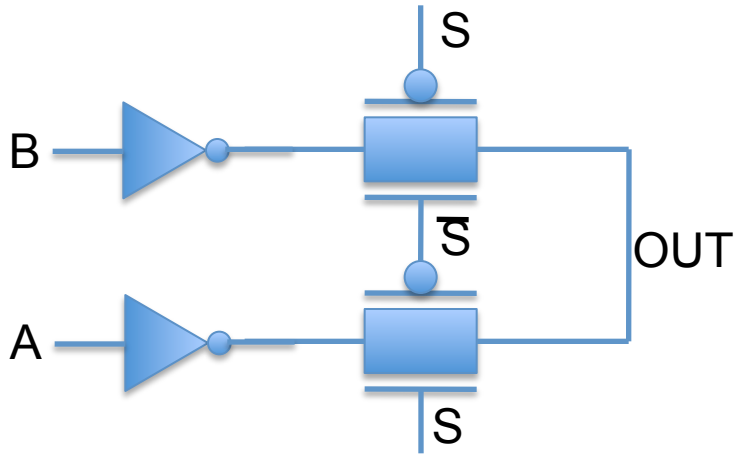
- Equivalence:

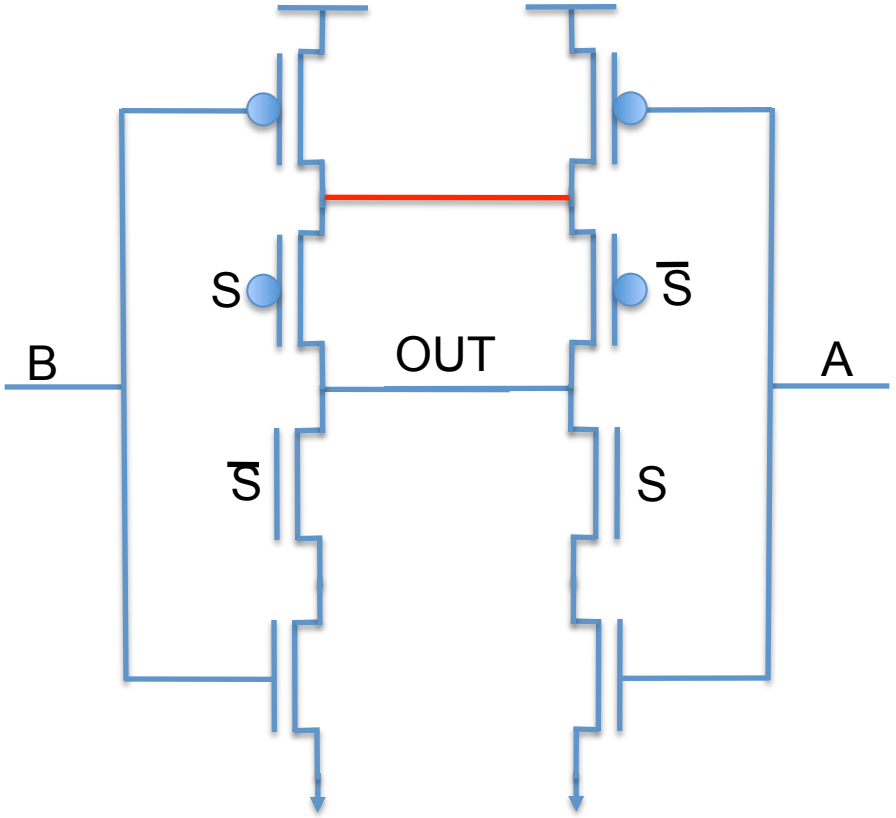




PTL MUX (cont'd)

- Equivalence:

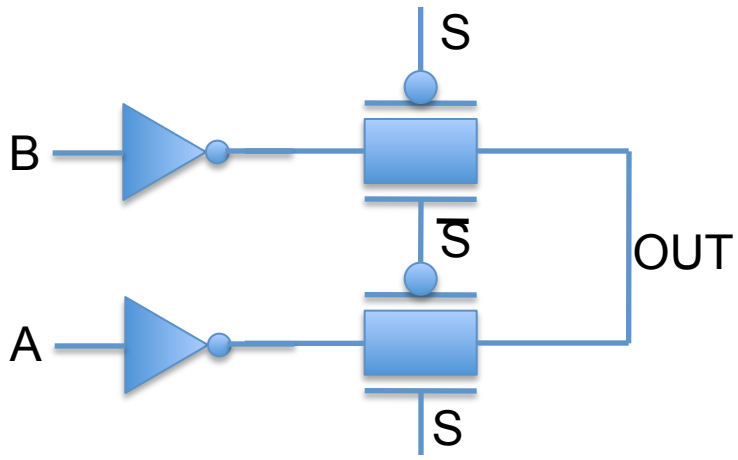


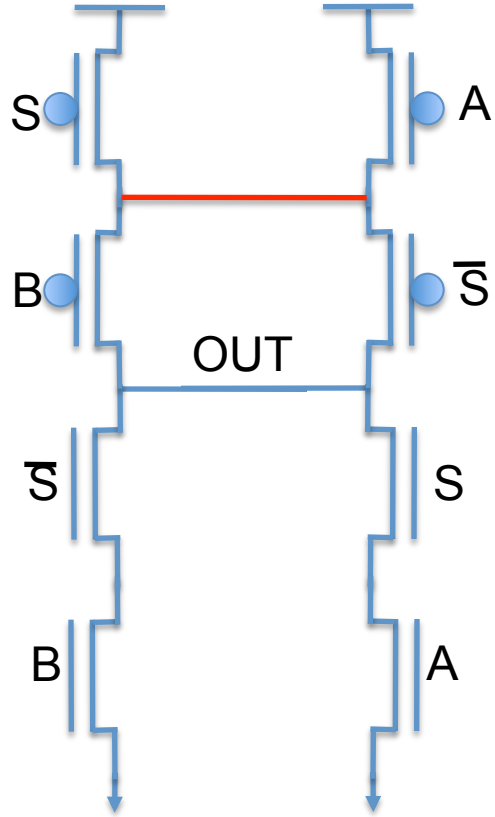


- With the red wire, it is an AOI22 gate
- But, is the red wire needed?

PTL MUX (cont'd)

- Equivalence:

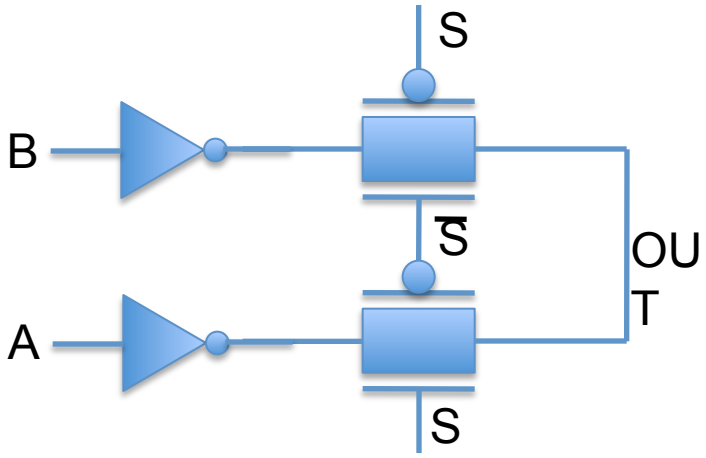


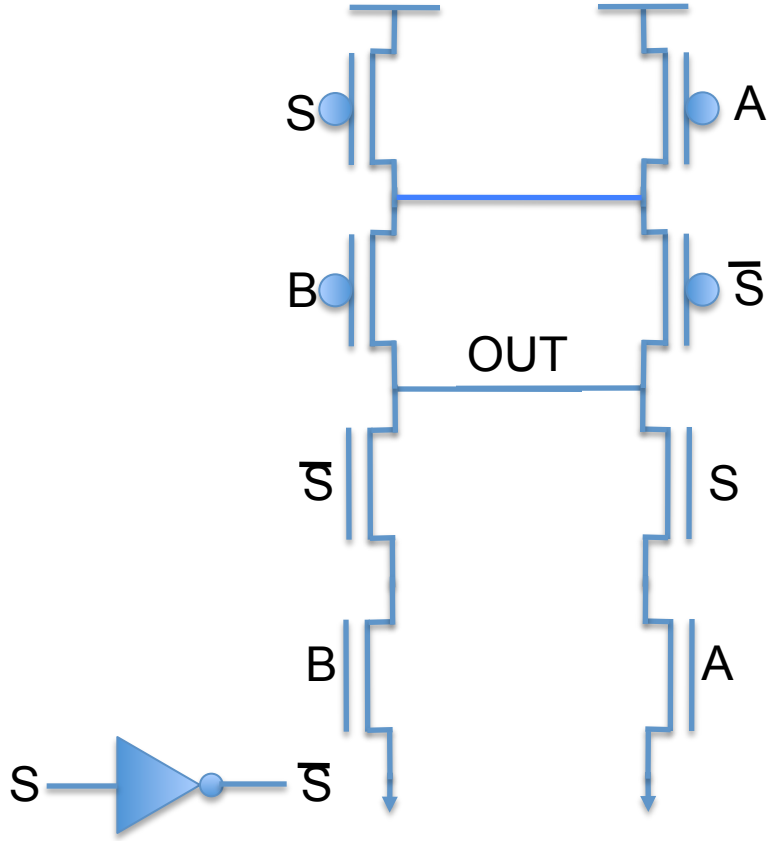


- But, is the red wire needed?
- No, if the inputs are re-ordered!
- If A=B=0, OUT should be 1 anyway

PTL MUX is EXACTLY an AOI22 Gate!!

- Equivalence:

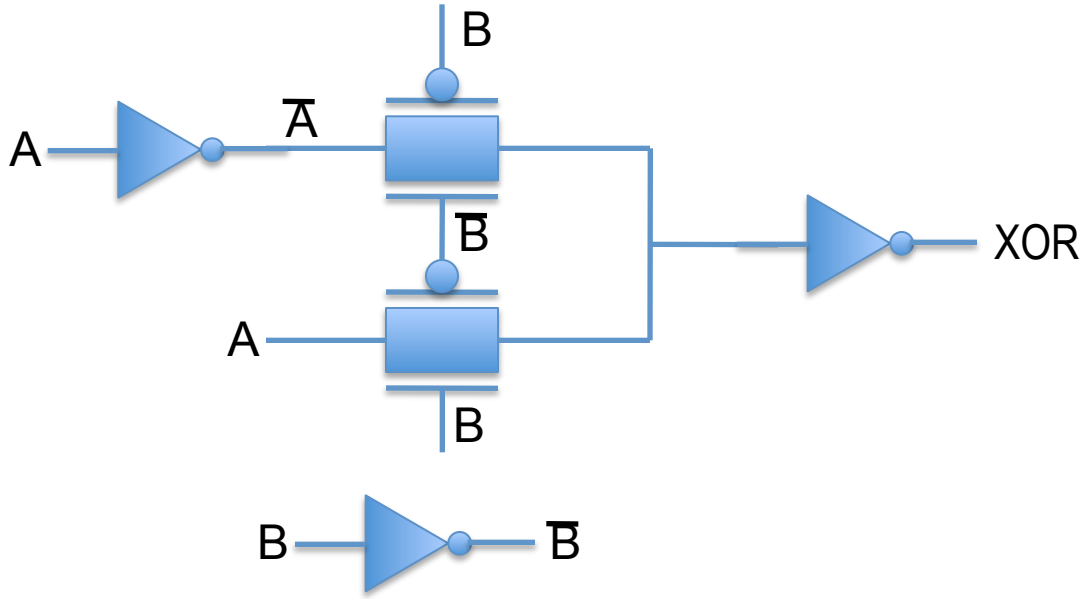




- Much less layout area

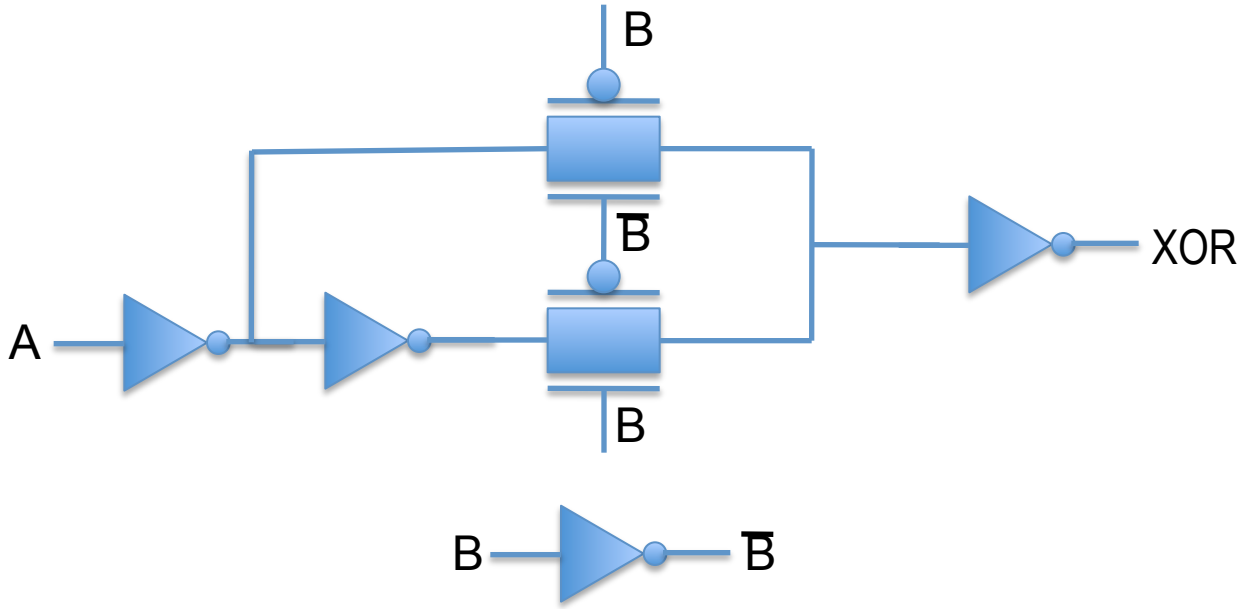
- Can easily add an output inverter to both if desired
- **Conclusion: DO NOT PUT EXPLICIT PTL MUX IN THE LIBRARY**

Ok, but what about PTL XOR?

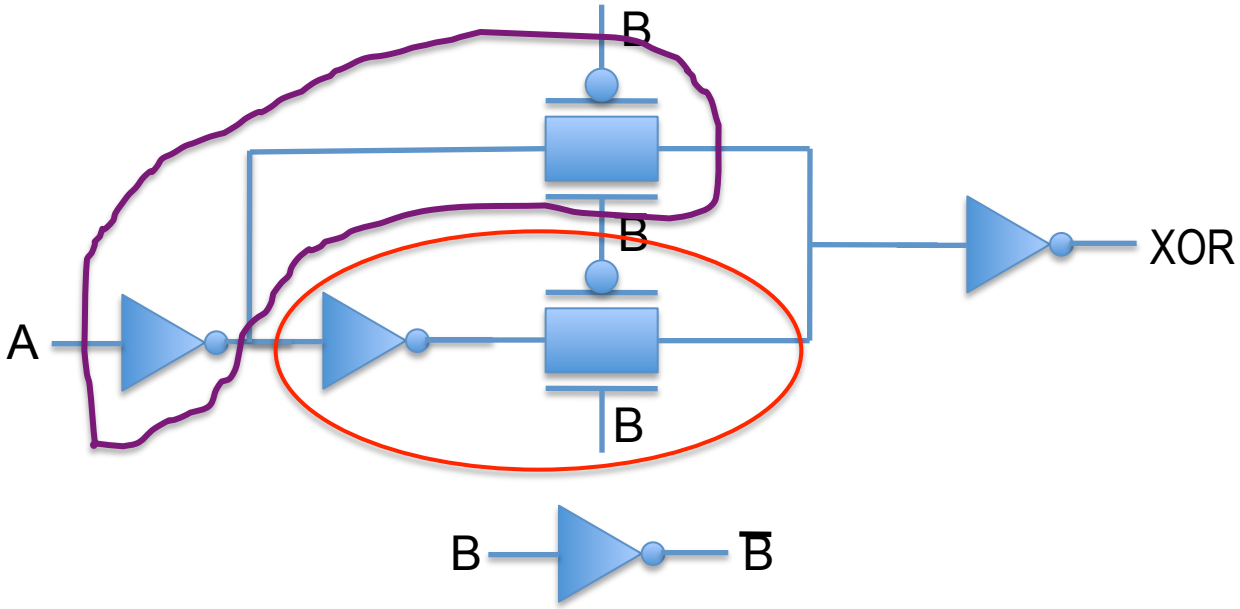


- But, need to have inputs with a purely capacitance load to enable static timing analysis

Revised PTL XOR

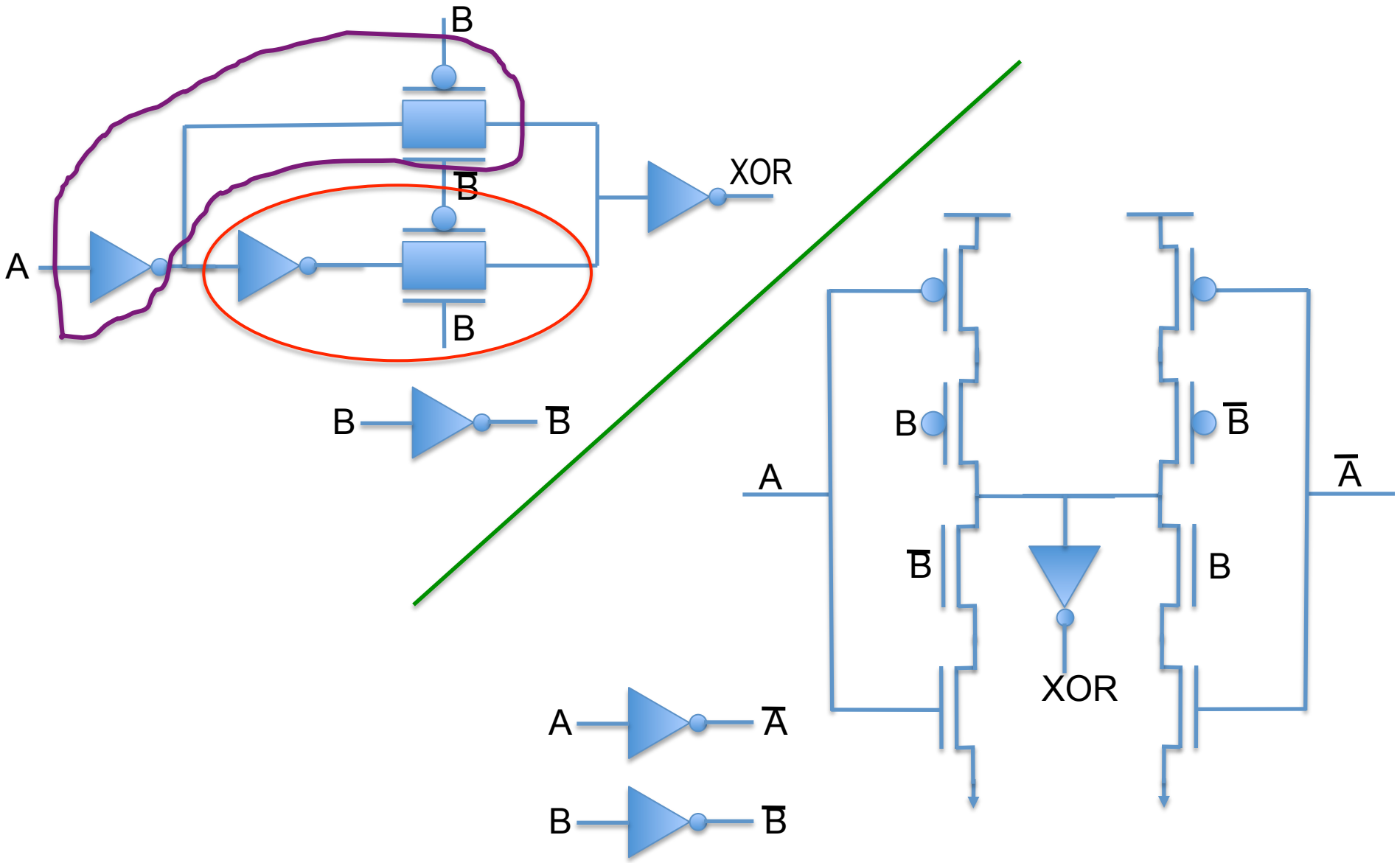


Closer Look at the PTL XOR

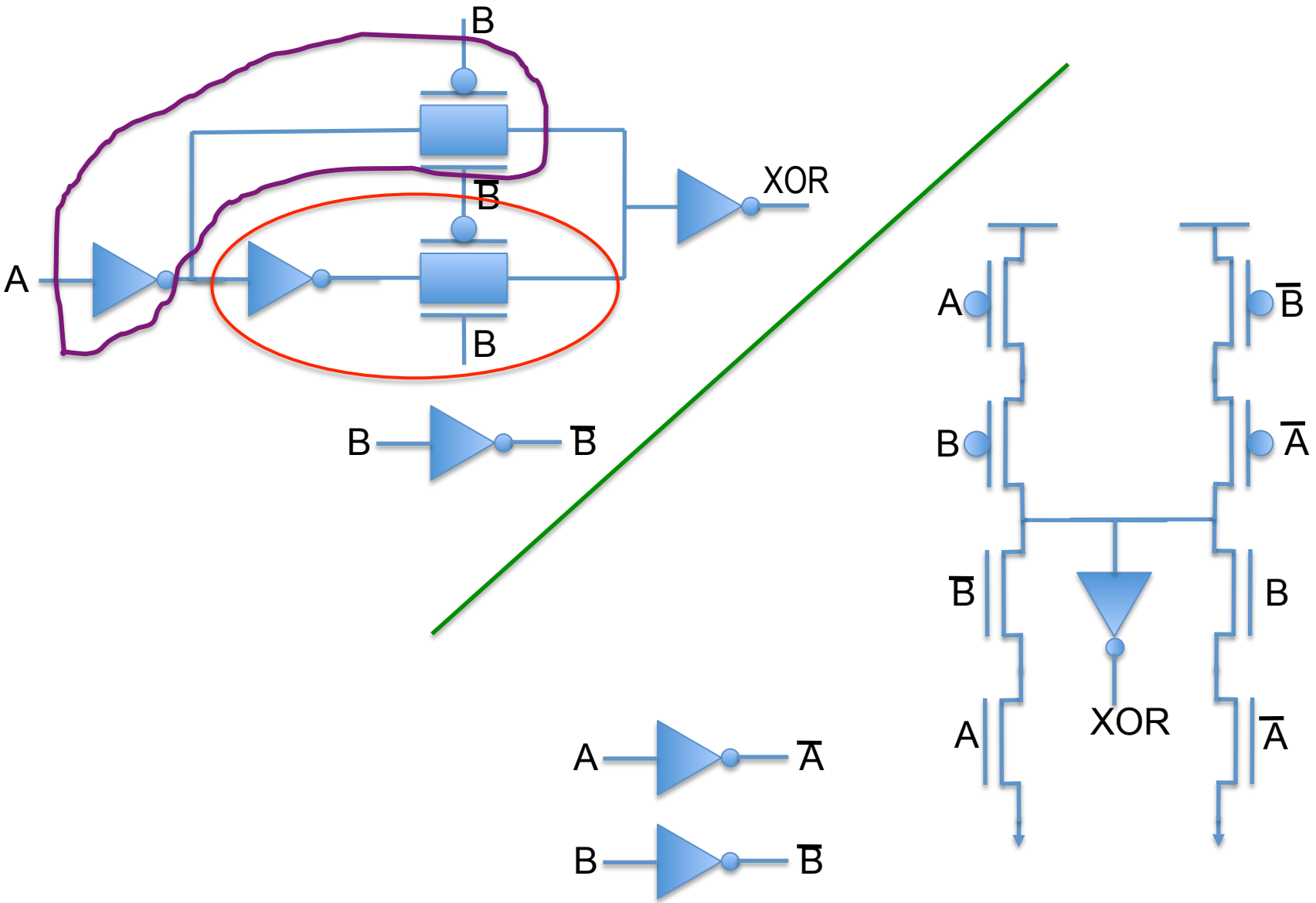


- Tri-state inverters are encircled

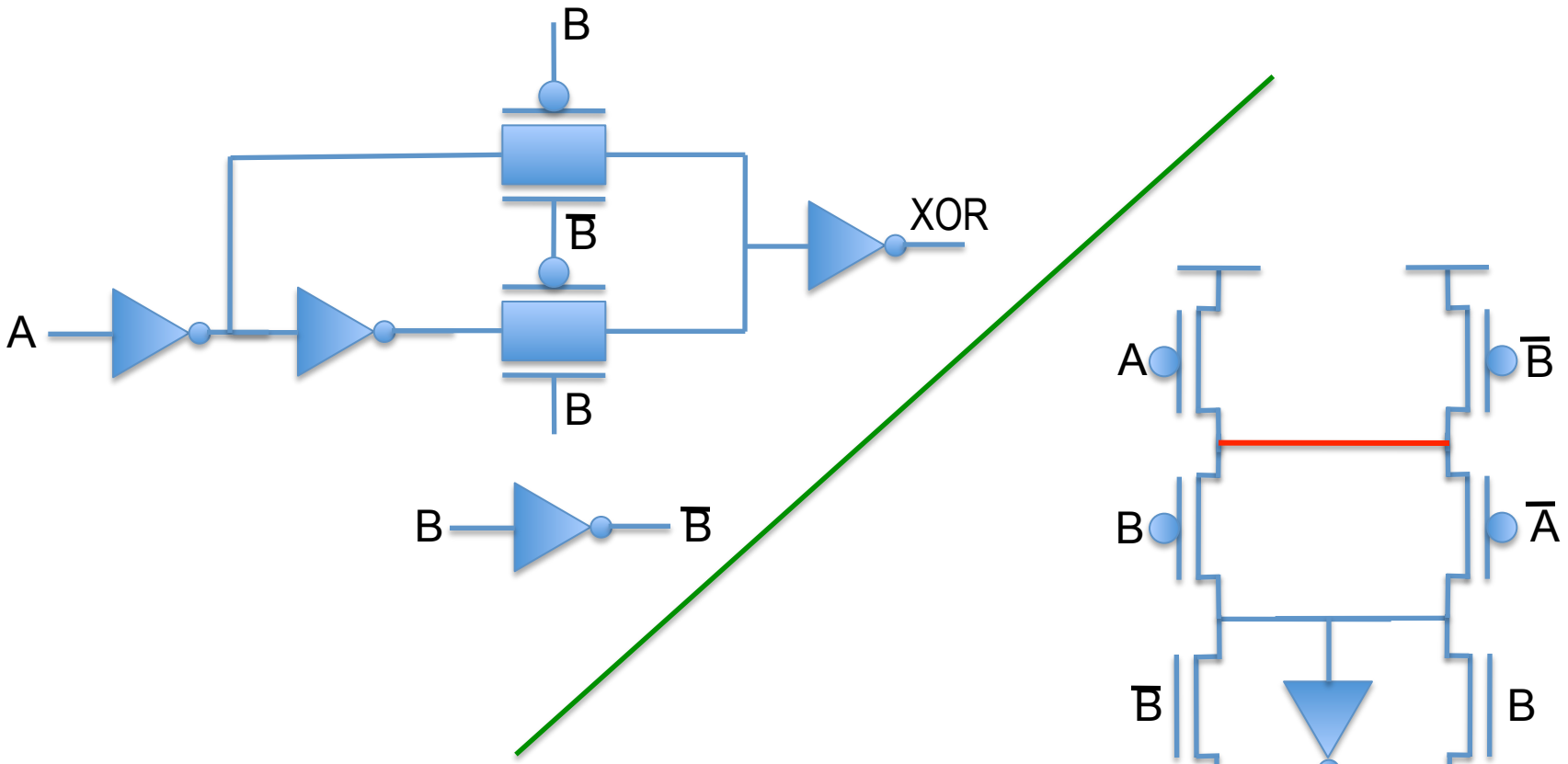
PTL XOR Equivalence



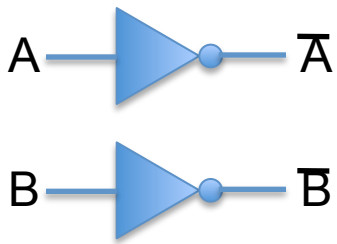
PTL XOR Equivalence – Inputs Reordered



PTL XOR Equivalence – Inputs Reordered

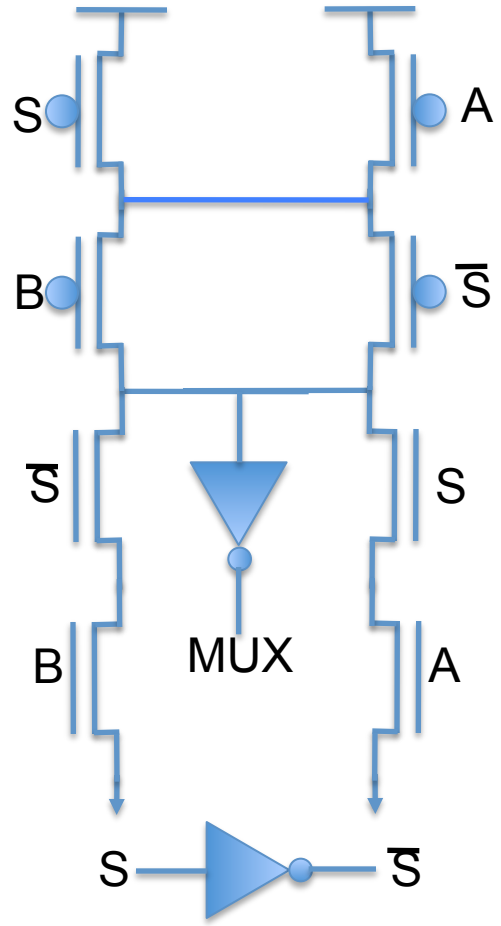


- The RED line does nothing!
- So, an AOI22 gate suffices to achieve equivalence



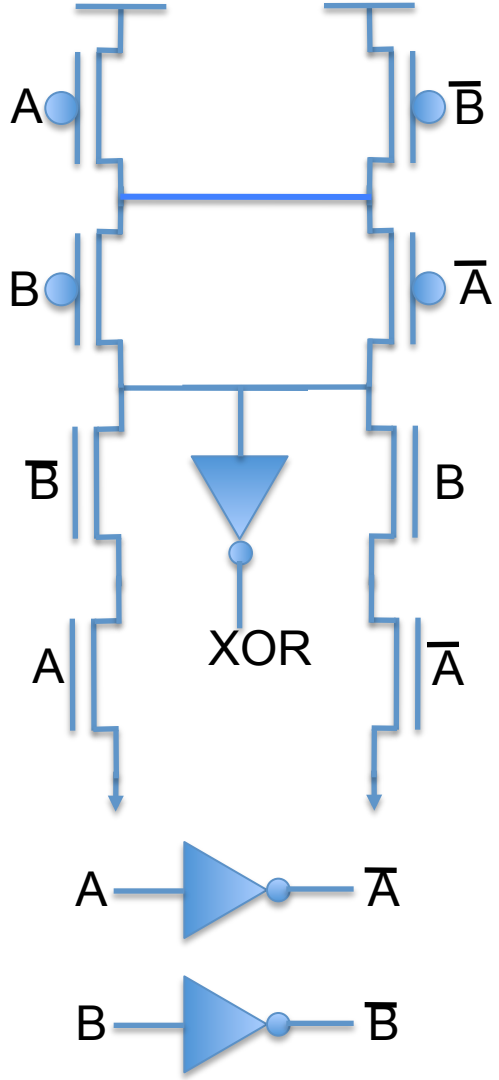
PTL–Summary

- PTL MUX is EXACTLY this:



- AOI22 plus inverters

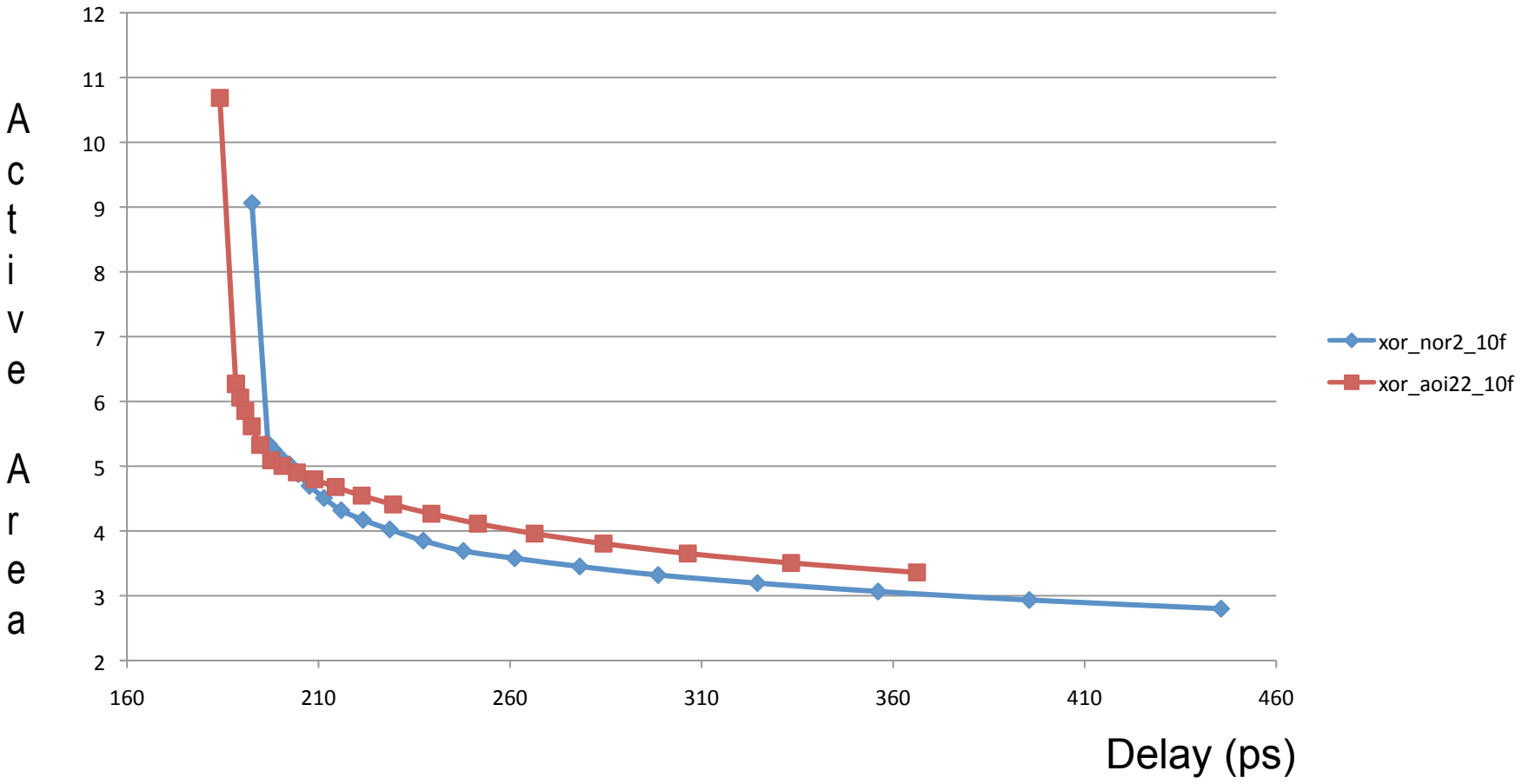
- PTL XOR is EXACTLY this:



- AOI22 plus inverters

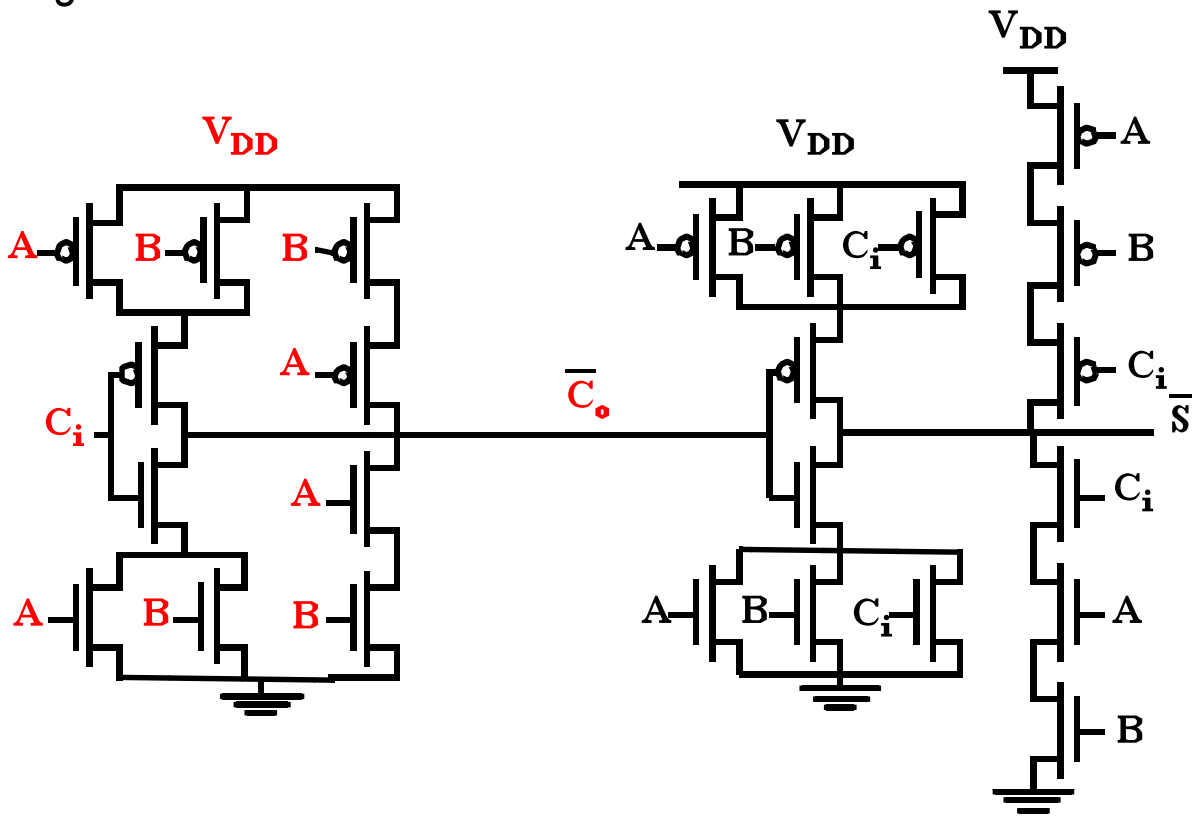
XOR Implementation

- But, AOI22-based XOR has 12 transistors plus 2 diffusion breaks, so a cell width equivalent to 16 transistors
- NAND2-AOI12 XOR has 10 transistors, no breaks … so it wins!!



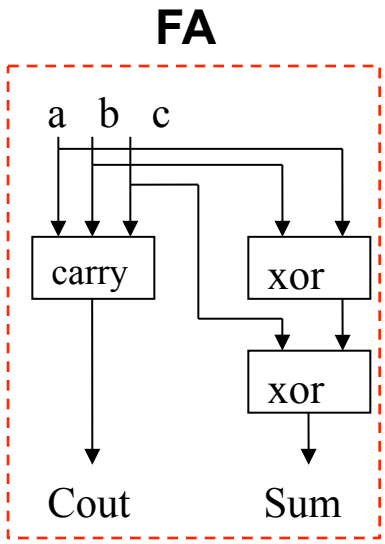
Full Adders

- The most area efficient adder:
  - 28 transistors (for non-inverted carry and sum)
  - 4 sizeable gates



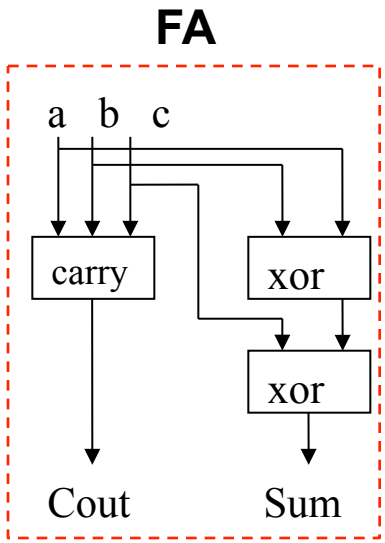
Full Adders (cont'd)

- The fastest full adder:
  - 2 XOR's and NAND-based carry (3 NAND2's and one NAND3)
  - 38 transistors
  - 8 sizeable gates

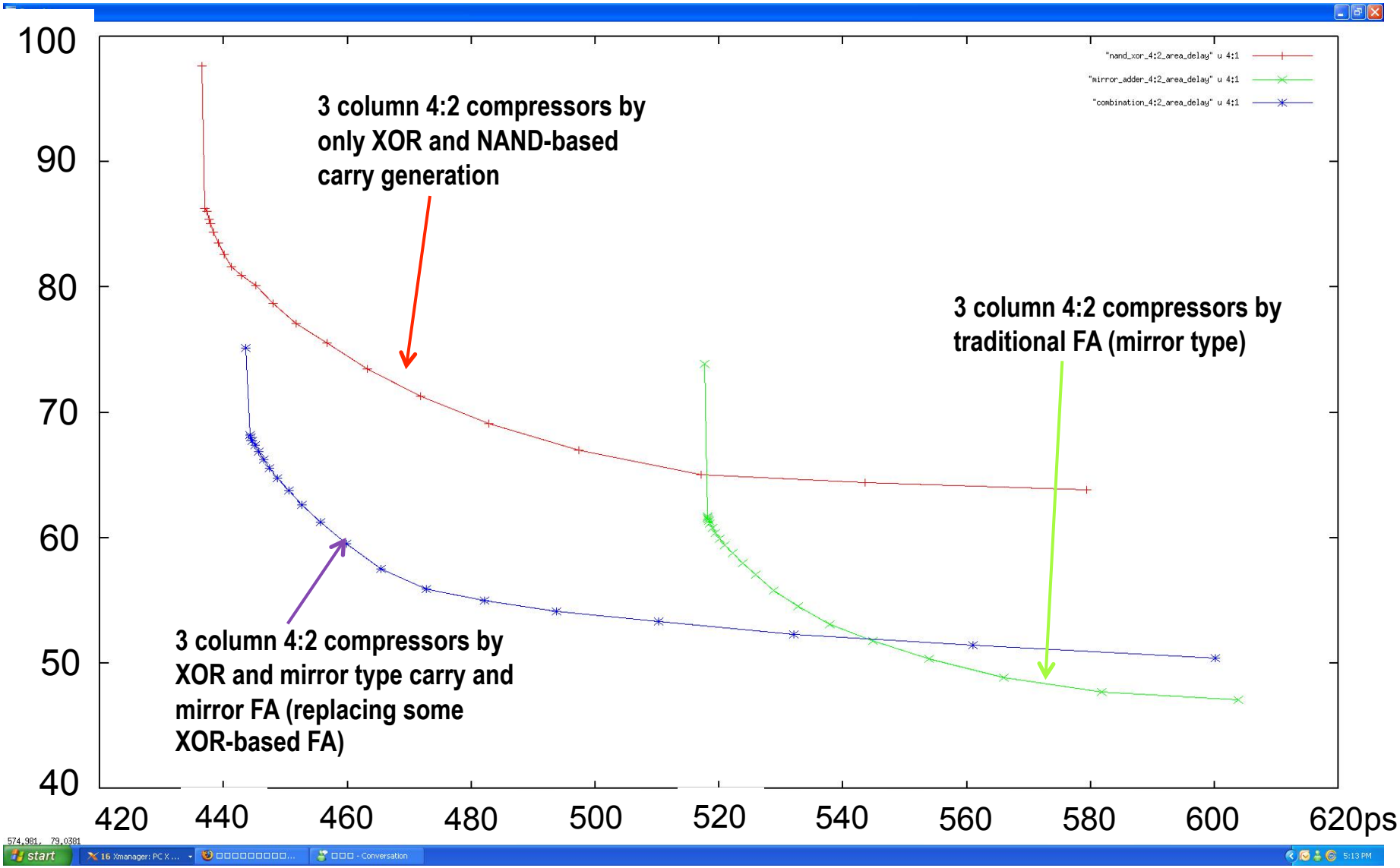


Full Adders (cont'd)

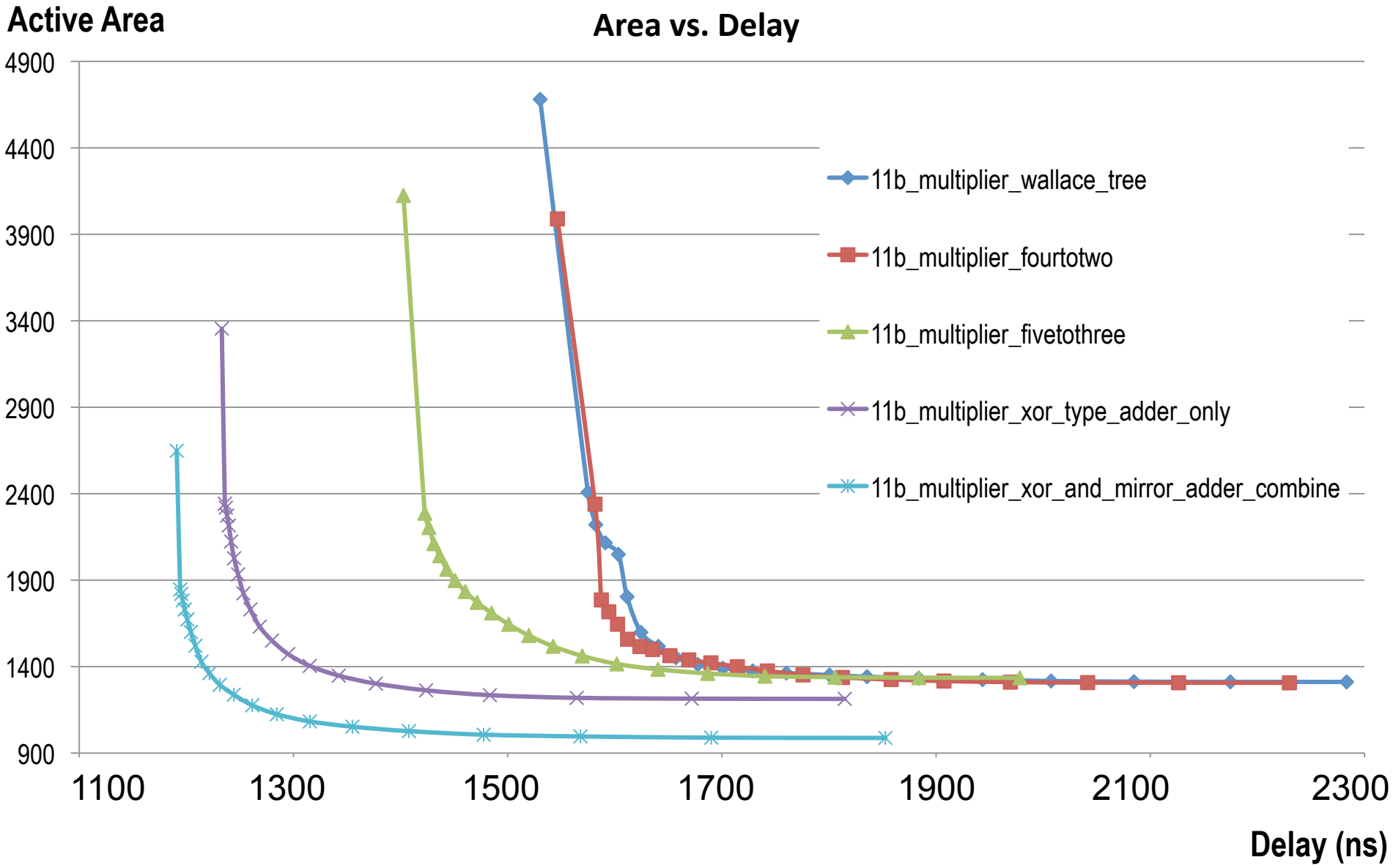
- The best full adder?
  - 2 XOR's and mirror-carry
  - 32 transistors
  - 9 sizeable gates



Different 4:2 Implementations

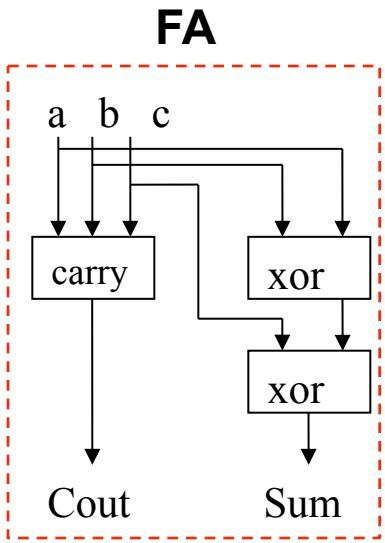


Comparison of Multiplier Implementations



Full Adders

- The clear winner:
  - 2 XOR's and mirror-carry
  - 32 transistors
  - 9 sizeable gates

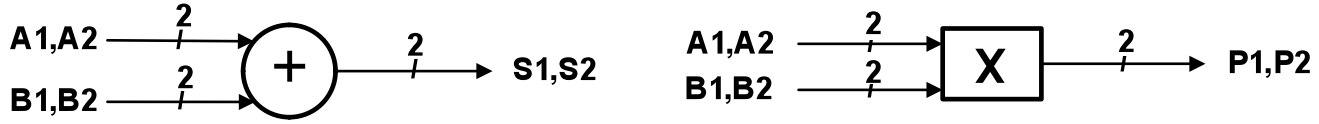


Why So Much Energy Waste? (cont'd)

- Many digital blocks include appreciable digital signal processing and/or arithmetic networks
- You have to make sure there are no carry-ripple delays and you have to make sure the compression trees use minimum power
  - The latter requires the most power efficient full adder cell
  - And one that is very "friendly" in gate size selection
  - The leading commercial tools do not hit the mark here …

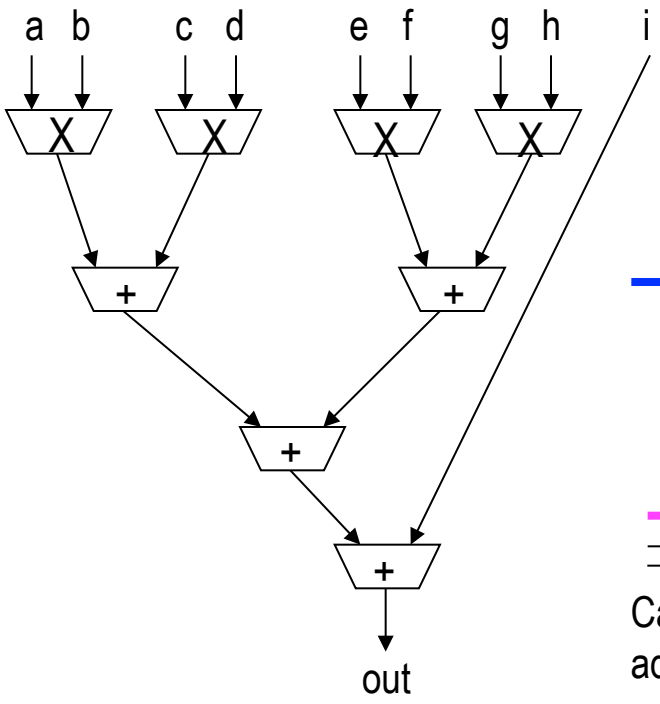
Generalized Carry-Save (GCS) Arithmetic

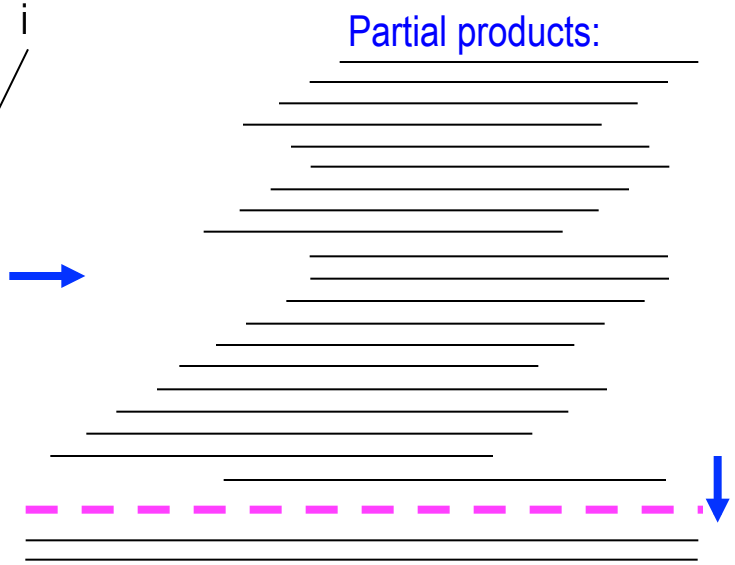
- DSP networks are largely an interconnection of adders, multipliers
  - Rippling of carries must be systematically avoided in arithmetic calculations
  - The carry operation is serial and thus many mathematical operations cannot be done in parallel
  - We have recently developed Generalized Carry-Save (GCS), which is a technique to parallelize addition and multiplication
  - All bits are added in parallel (essentially 4:2 compression) and in multiplication only the compression of partial products is employed



- Adder delay is therefore independent of operand widths
- Multiplier delay only depends on the logarithm of the multiplier width

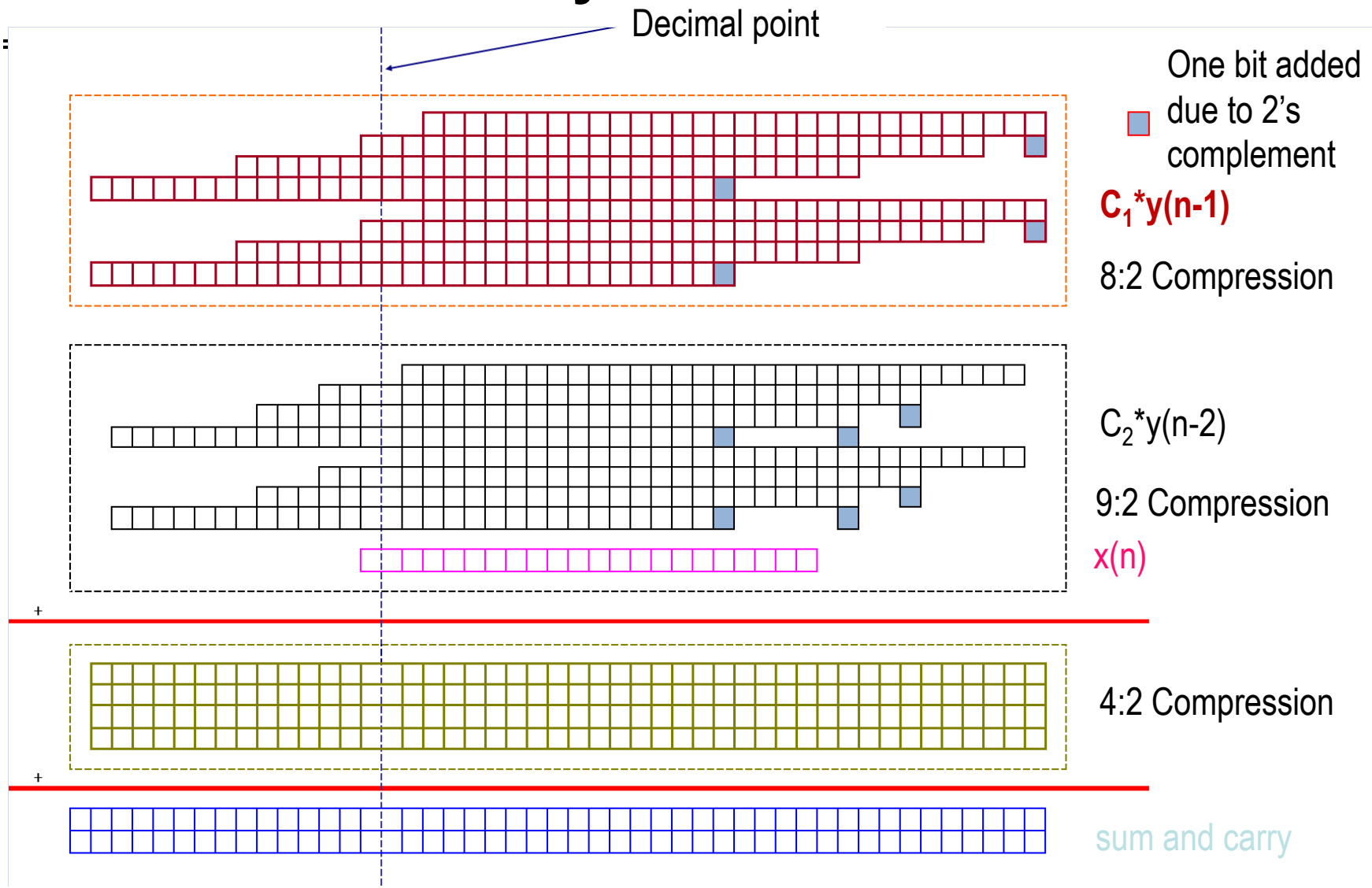
A Multiplier and Adder Network





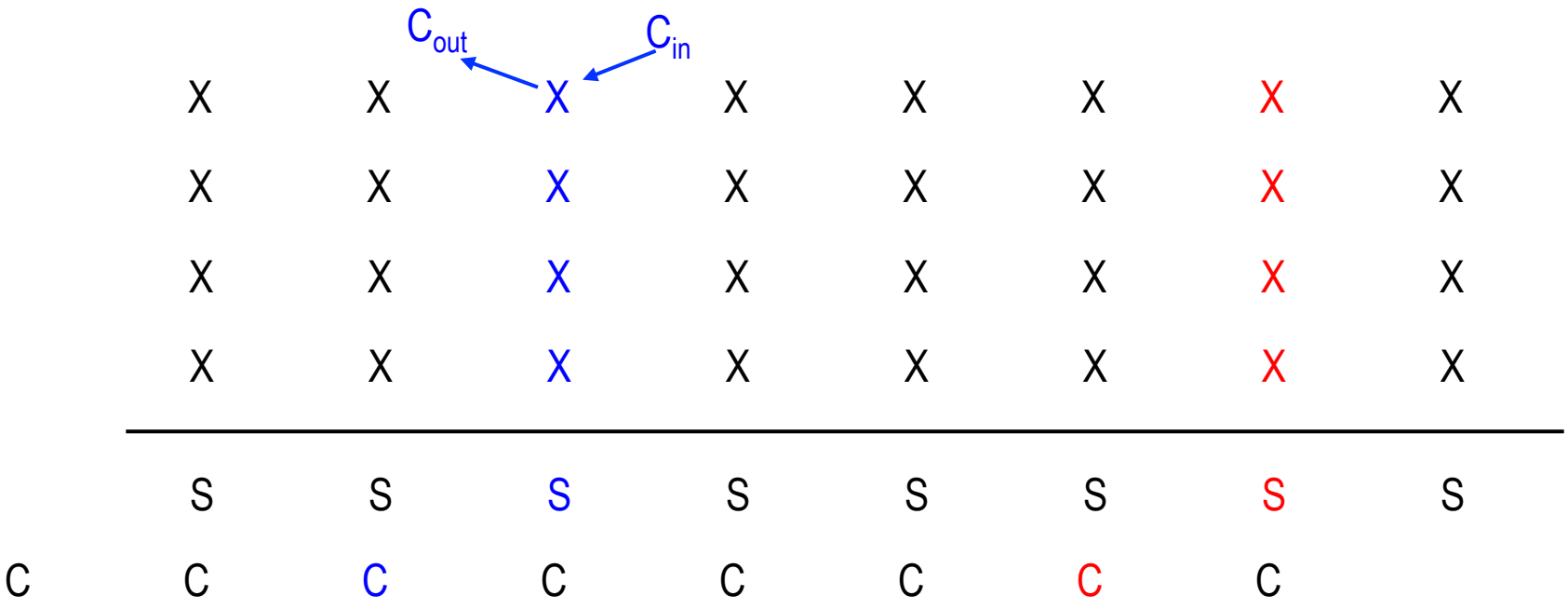
Carry-free compression to 2 final rows, whose addition equals “out”

Generalized Carry-Save Arithmetic for IIR



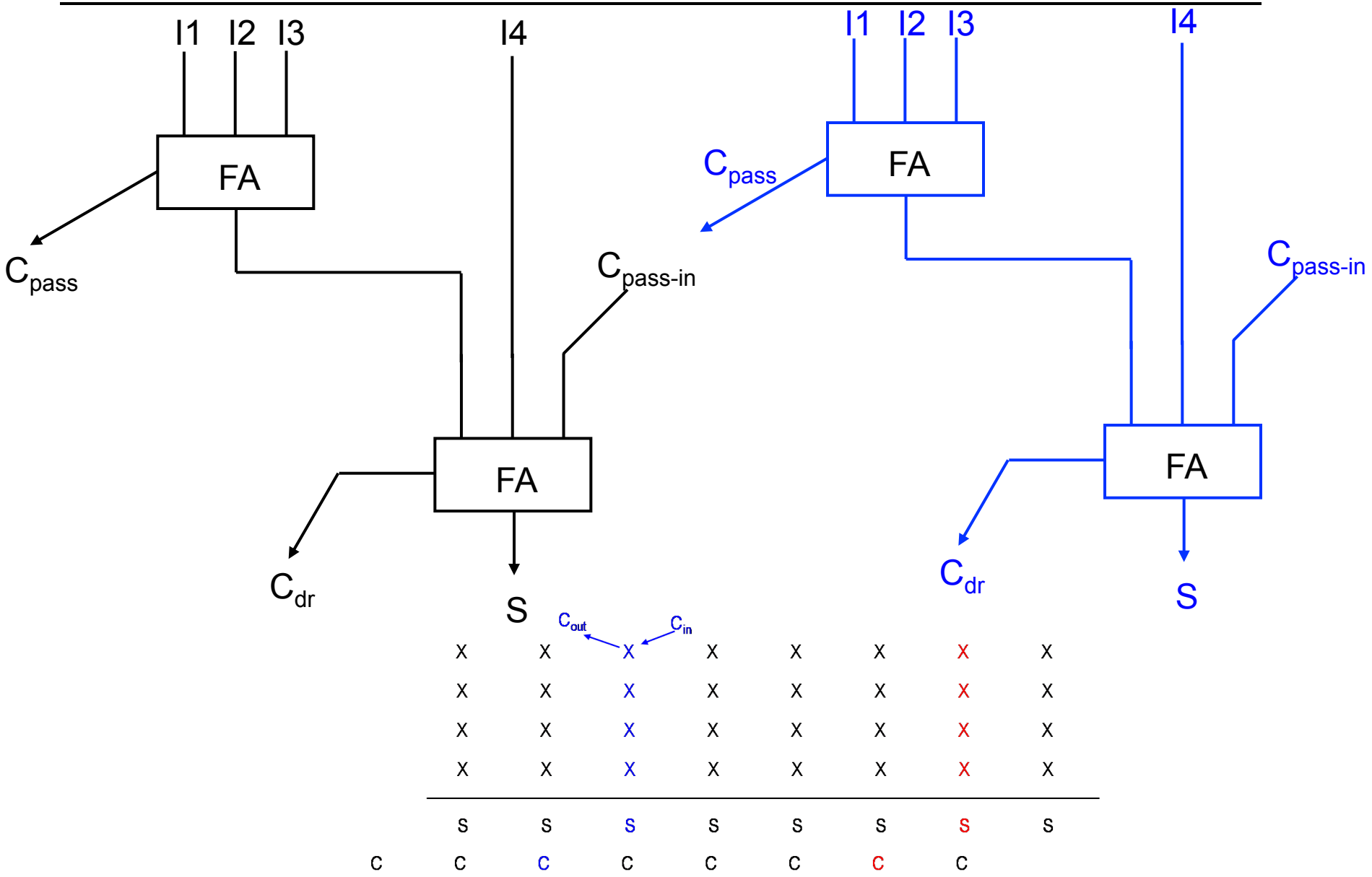
$$y(n) = C_1 * y(n-1) + C_2 * y(n-2) + x(n)$$

4:2 Compression

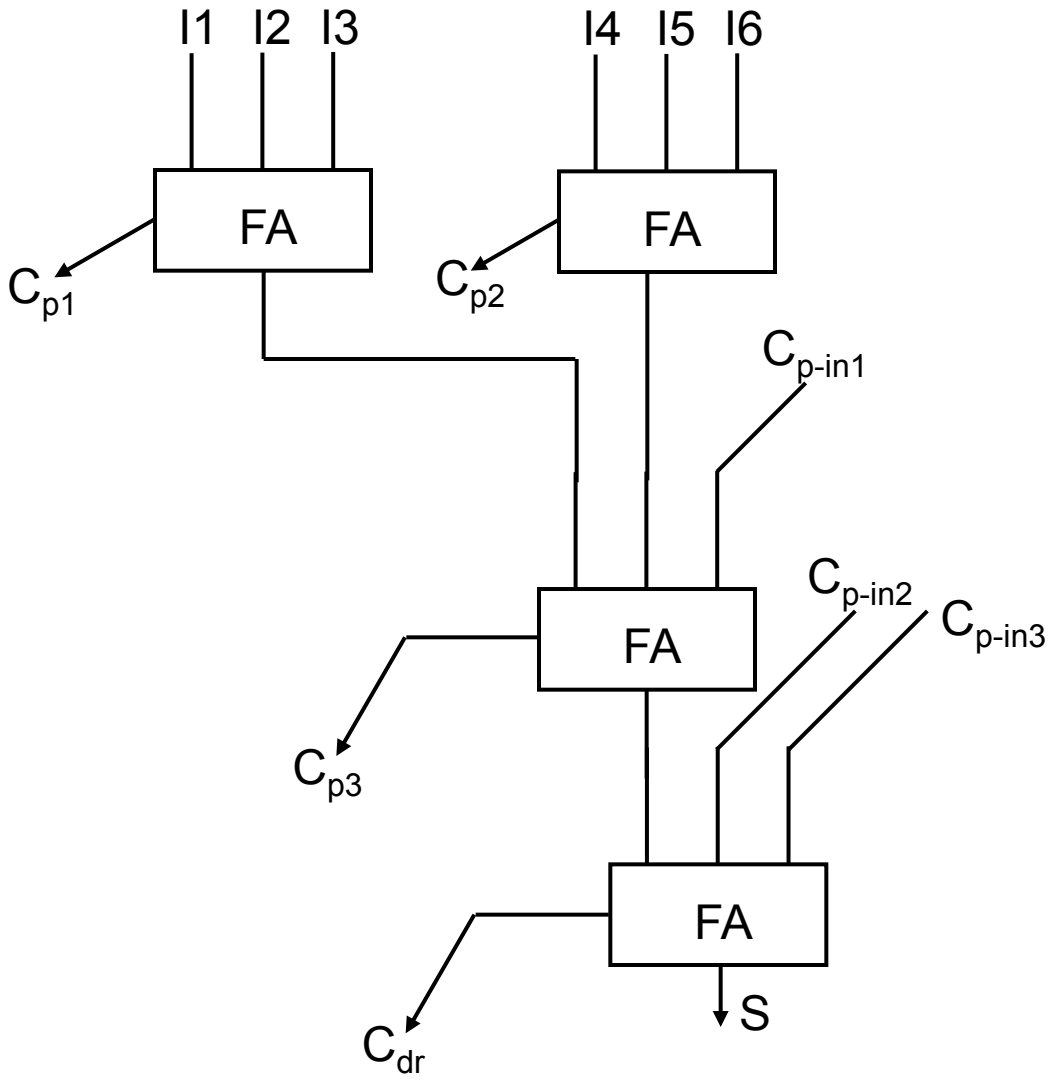


- Cout does not depend on Cin

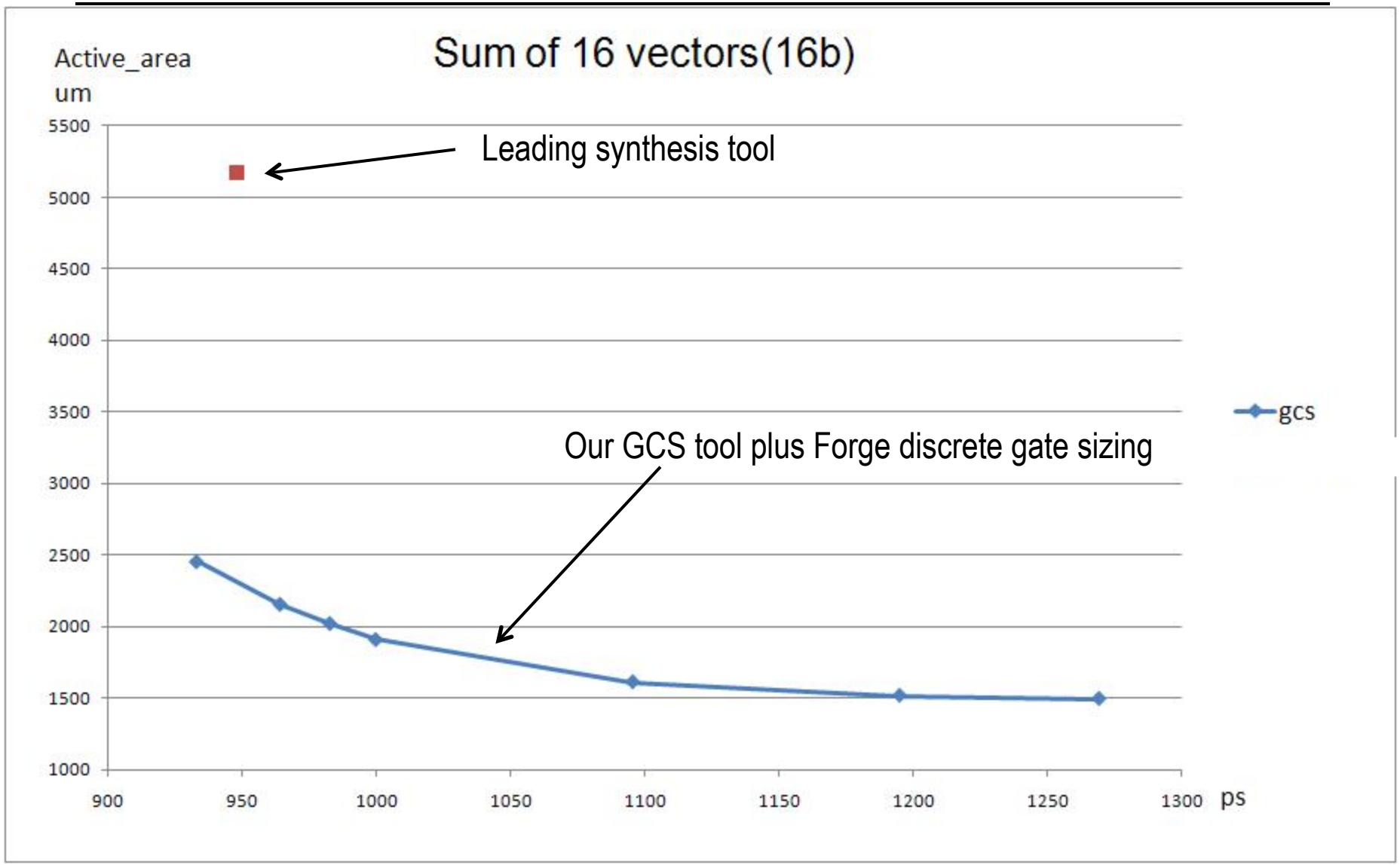
Ripple-Free 4:2 Compressor



Ripple-Free 6:2 Compressor

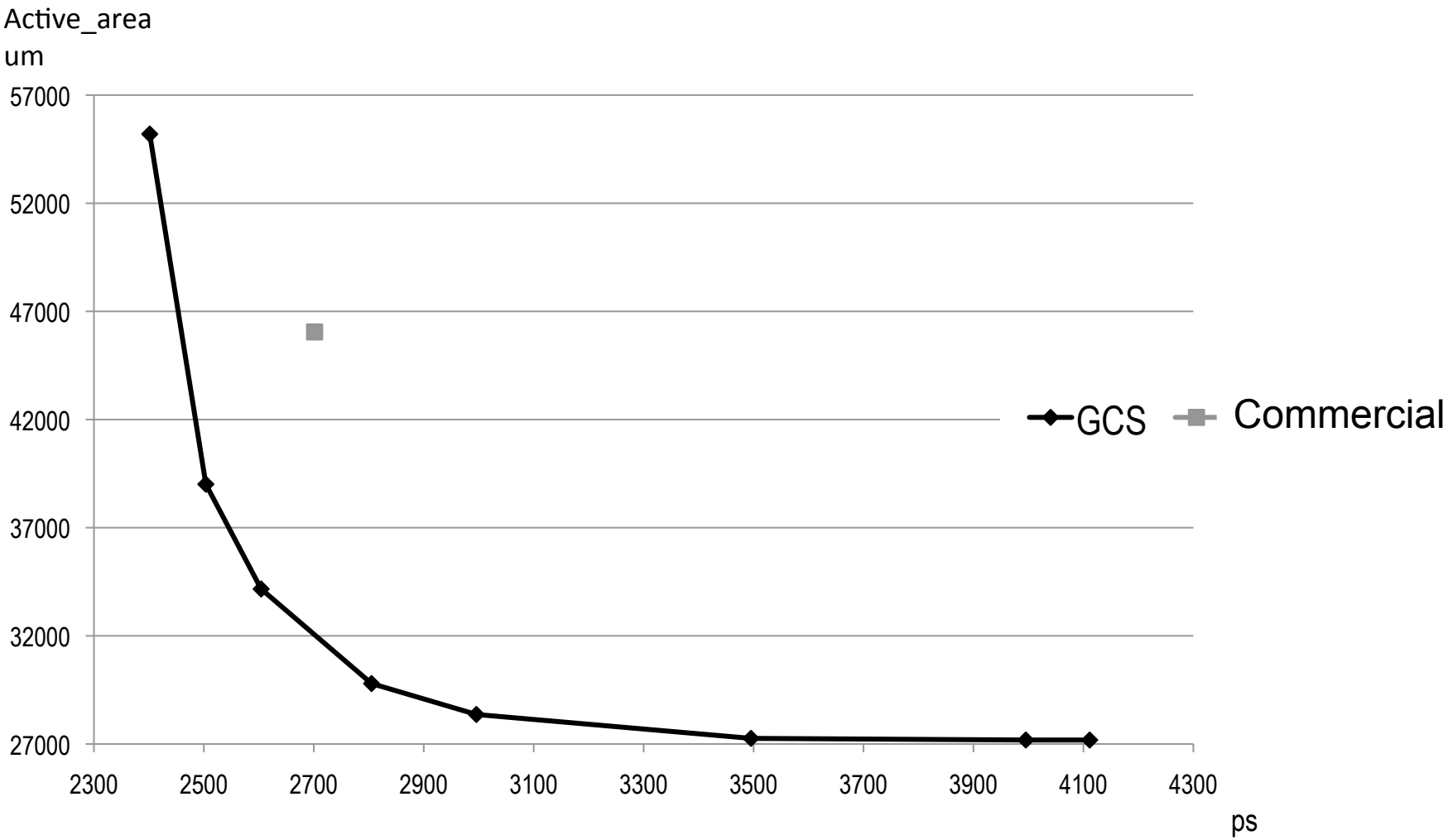


Sum of 16 16b vectors – TI 45nm 0.9V



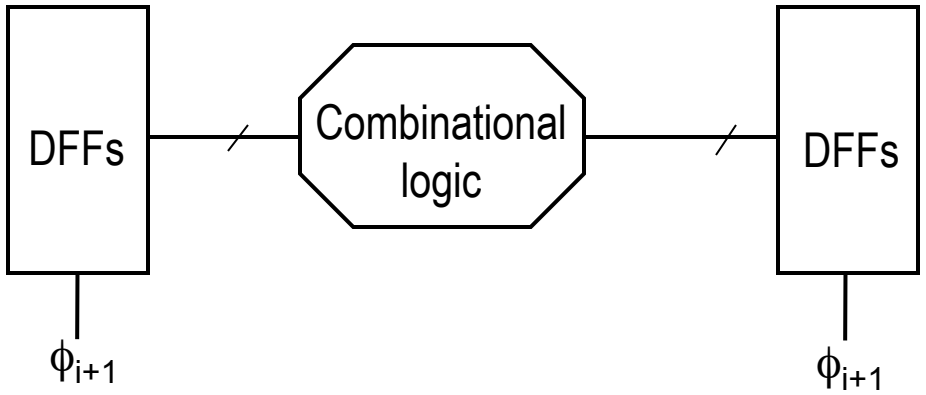
Nvidia FMA Example – TI 45nm 0.9V

- Multiplication and accumulation with absolute output
- A(54b), B(56b) and C(164b)
- Out = | *A* × *B* + *C* |



Why Do People Worry About Min-Delay?

- No, really, why do people worry about min-delay (hold time)??
- Let's consider the problem:



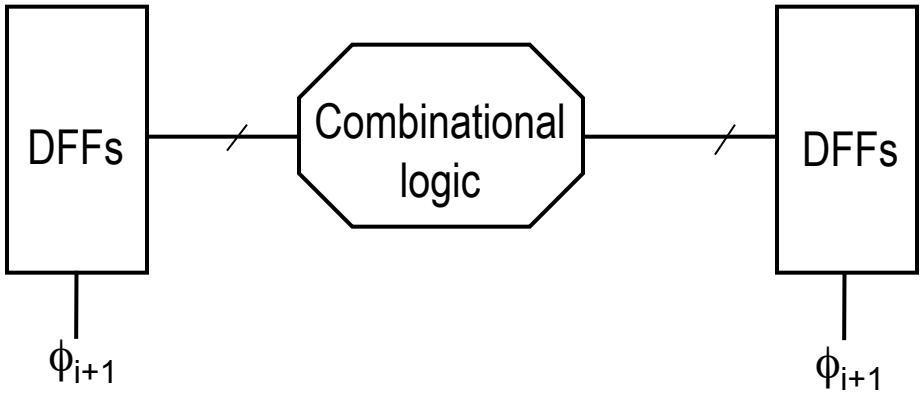
- Consider the "hold" for the i^{th} capturing event in the 2^{nd} bank of DFFs
- This "races" with the $(i+1)^{st}$ capturing event in the 1^{st} bank of DFFs

$$\phi_{i+1} + T_{clk->Q} + T_{CL\min} - T_{skew} > \phi_{i+1} + T_{hold}$$

- Thus:

$$T_{CL\min} > T_{hold} - T_{clk->Q} + T_{skew}$$

Min-Delay?



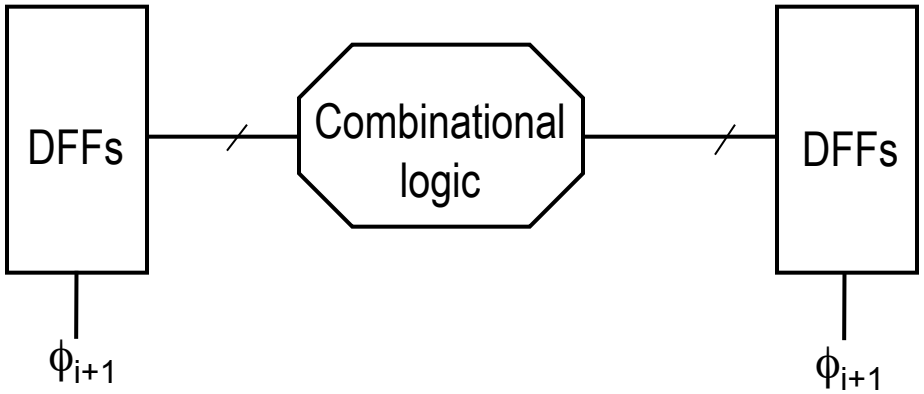
$$T_{CL\min} > T_{hold} - T_{clk->Q} + T_{skew}$$

- For contemporary DFFs, T_{hold} is negative and roughly two gate delays

$$T_{su} \approx |T_{hold}|$$

- So, in the absence of clock skew, T_{CLmin} is negative by 4 gate delays!!!
- You mean to tell me clock skew (in the worst case) is WAY more than 4 gate delays??
  - >> 4 gate delays?? (Now there's a problem the CAD industry hasn't addressed!!)

Min-Delay (cont'd)

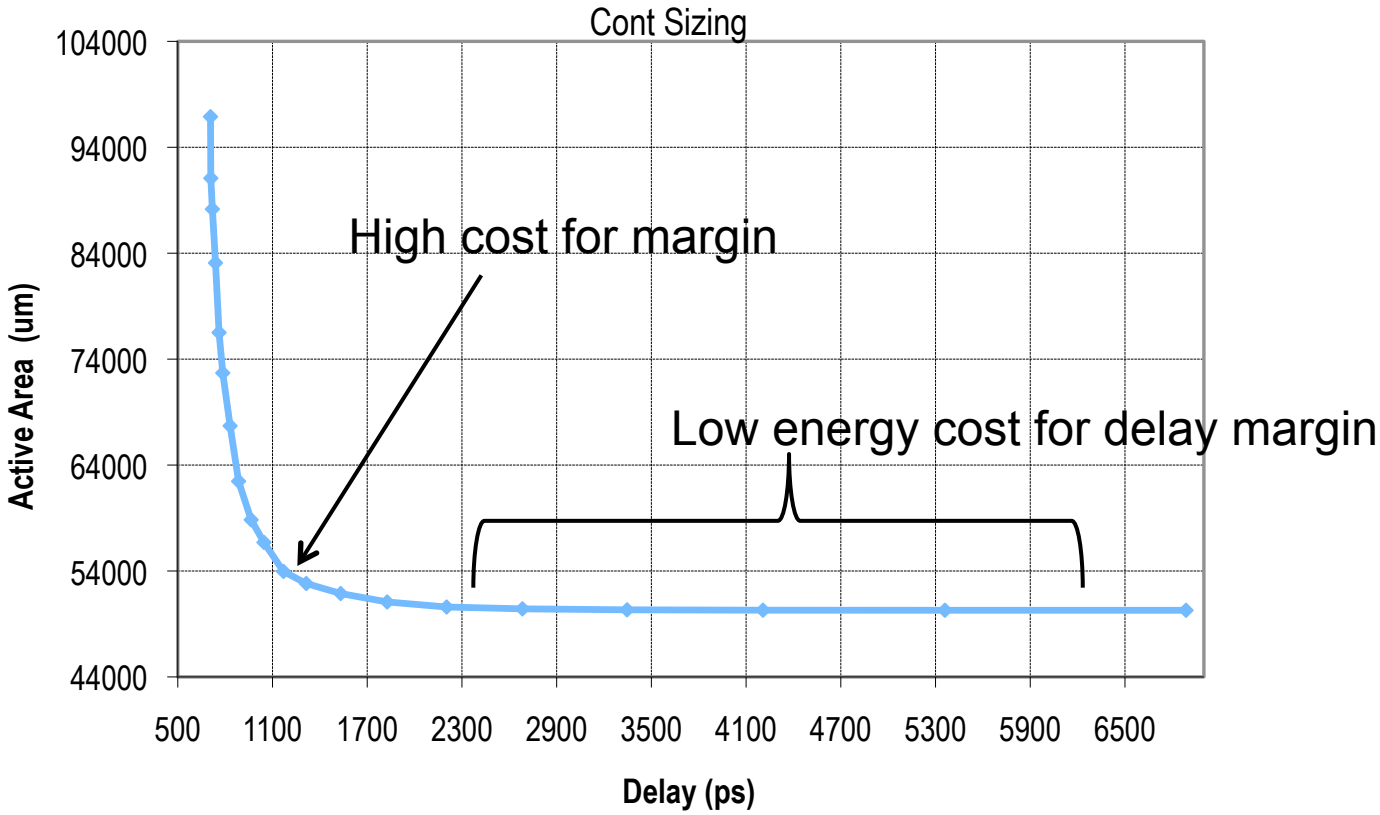


$$T_{CL\min} > T_{hold} - T_{clk->Q} + T_{skew}$$

- So, in the absence of clock skew, T_{CLmin} is negative by 4 gate delays!!!
- You mean to tell me clock skew (in the worst case) is WAY more than 4 gate delays??

- The good news is that our gate size selection tool (Forge) is able to identify the optimal points in the network to add delay (buffers) to satisfy any desired T_{CLmin}
  - And minimizes total area (and power) added in the process

Energy Cost for Margins (Energy Margins)



- Many today argue there is a HUGE energy cost for margins
  - So they say we need to go to asynchronous, software that tolerates errors, etc.
  - But they ignore the 4-100X overheads there …
- Definitely can substantially LOWER your energy cost for margin

Ongoing Work

- Still more work on defining the optimal synthesis library
- Major new initiative: adding analysis of local statistical variations into the power optimization

Conclusion

- Goal: Toward optimal power efficiency in digital IC's
- Message: Existing tools and libraries leave a LARGE amount of energy on the table
- Conclusion: We have devised means that seek to take a fair amount of that energy off the table