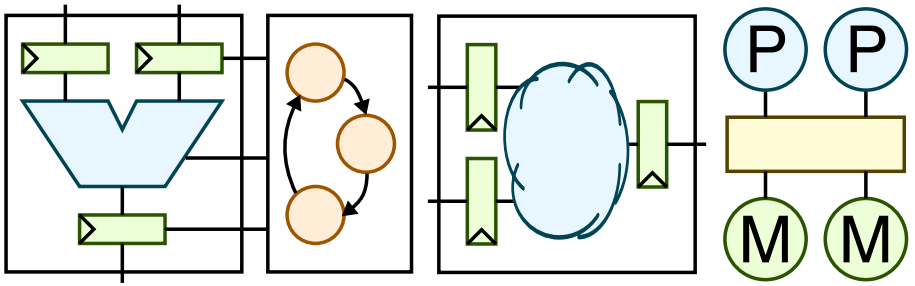
ECE 5745 Complex Digital ASIC Design
Topic 6: Closing the Gap Between ASIC and Custom

Christopher Batten

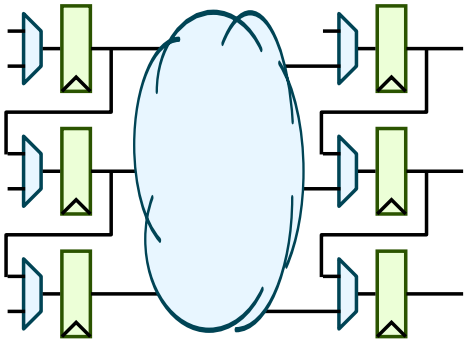
School of Electrical and Computer Engineering
Cornell University

http://www.csl.cornell.edu/courses/ece5745

Part 1: ASIC Design Overview



Topic 1
Hardware Description Languages

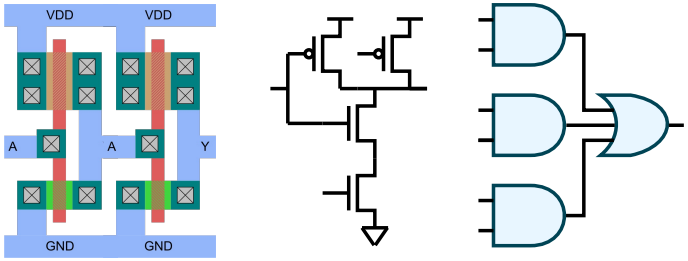


Topic 8
Testing and Verification

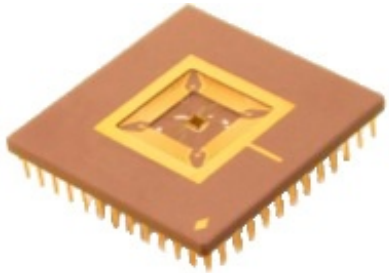
Topic 4
Full-Custom Design Methodology

Topic 6
Closing the Gap

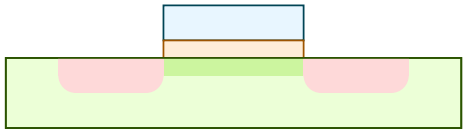
Topic 5
Automated Design Methodologies



Topic 3
CMOS Circuits



Topic 7
Clocking, Power Distribution, Packaging, and I/O



Topic 2
CMOS Devices

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

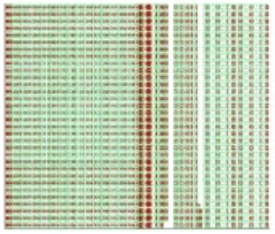
Data, Clock, and Power Gating

Voltage Scaling

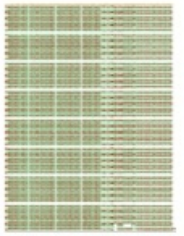
Datapath for Processor Register Read Stage

Synthesized Standard-Cell ASIC

Synthesized Single Bit Slice
(then tile these bit slices)



Manual Structural
Gate-Level Implementation
with Standard Cells and
7 Custom Cells



Full Custom

7-port 75x64-b regfile, 6-entry reservation station, bypass muxes, 18-b immediate logic, 13x1-b condition code regs in 0.5um

Adapted from [Chang'02]

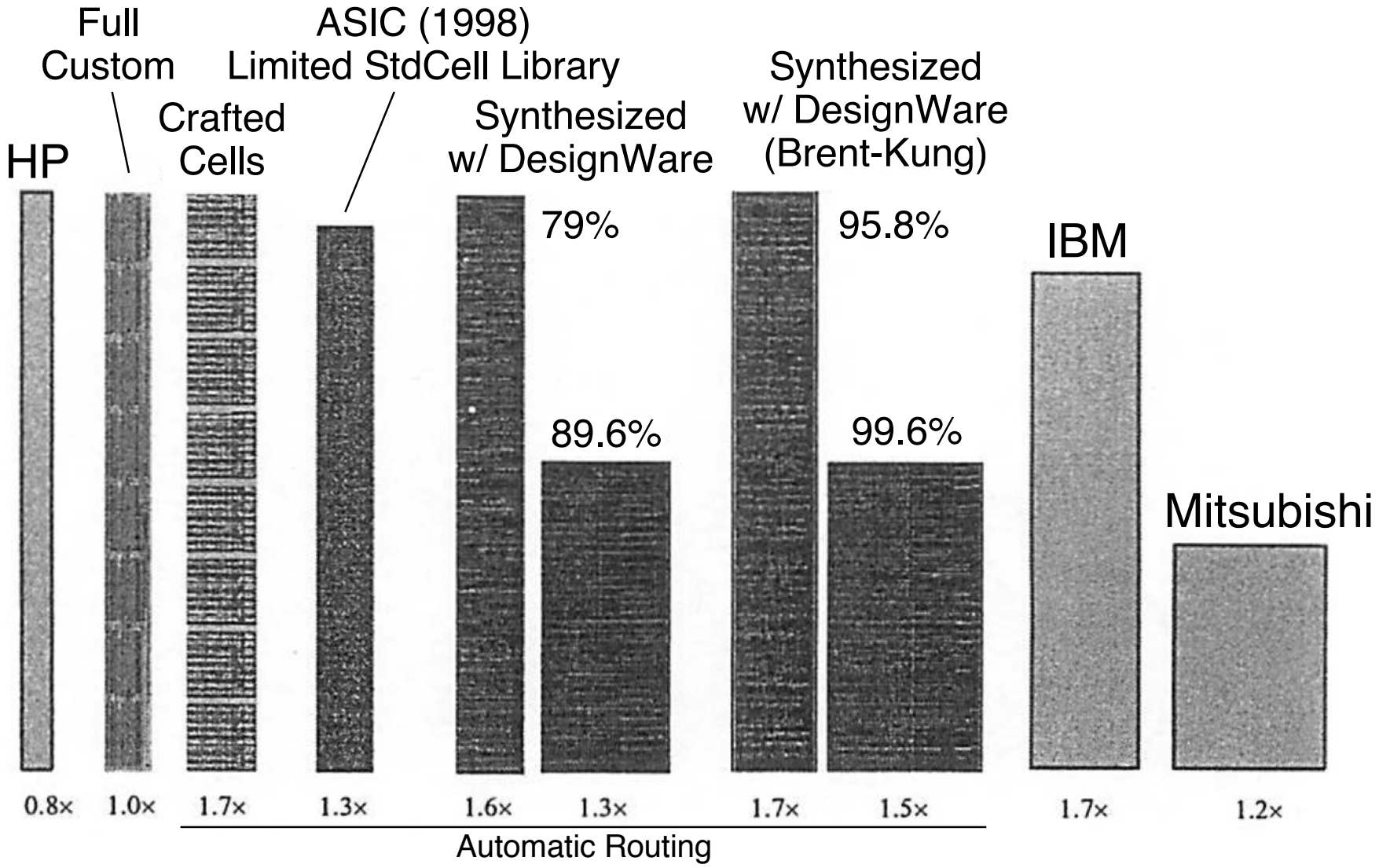
Area, Energy, Delay Gap between ASIC and Custom

Parameter	Custom	Crafted	Bit-sliced	Automatic P&R
Area	1	1.64	5.25	14.5
Delay	1	1.11	2.23	3.72
Gate Load	1	1.09	2.29	2.29
M2 Length	1	1.07	4.19	34.9
M3 Length	1	1.63	2.52	7.92
Effort	1	0.18	0.06	0.06

Although standard-cell ASIC design take a fraction of the effort the final design is many factors worse in terms of area, cycle time, and energy

Adapted from [Chang'02]

Datapath for 64-bit Adders



Adapted from [Chang'02]

Area, Energy, Delay Gap between ASIC and Custom

Adder	Process (L_{drawn})	FO4 (ps)	Delay (ns)	Delay (FO4)	Area (χ2)	Area (relative)	Number of Instances	Transistors (T) or Gates (G)	χ^2/T
HP (Naffziger)	0.50	140	0.9	6.6	61500	0.8	7000	T	8.8
IBM	0.50	125	1.5	12.0	135802	1.7			
Mitsubishi	0.50	250	4.7	18.8	98400	1.2	4280	T	23.0
MIT/Stanford									
Custom	0.50	250	3.7	14.8	81847	1.0	4666	T	17.5
Crafted	0.50	250	6.3	25.2	136684	1.7	162	Gate	
ASIC (1998)	0.50	250	10.0	40.0	105600	1.3	1041	Gate	
ASIC - DW	0.15	60	1.6	26.8	124848	1.5	1756	Gate	
ASIC - BK	0.15	60	1.1	18.3	105600	1.3	1097	Gate	

Adapted from [Chang'02]

ASIC vs. Custom High-Performance Processors

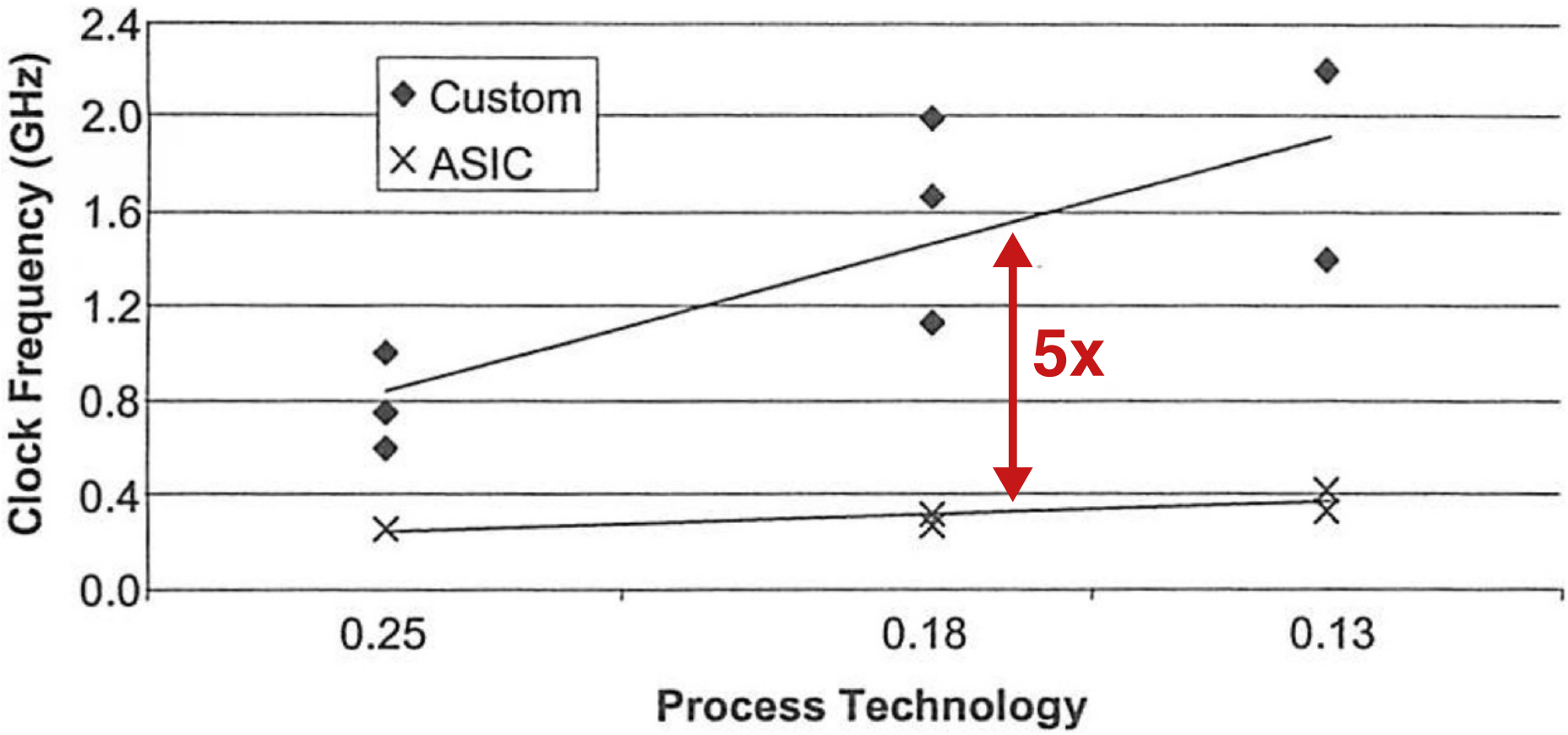
	Freq (GHz)	Tech (µm)	Vdd (V)	Pwr (W)	Area (mm^2)	Process Conditons	Operating Conditions
Pentium III (Katmai)	0.600	0.25	2.05	34.0	123.0		
Athlon (K7)	0.700	0.25	1.60	50.0	184.0		
Alpha 21264A	0.750	0.25	2.10	90.0	225.0		
IBM Power PC	1.000	0.25	1.80	6.3	9.8		
Pentium III (Coppermine)	1.130	0.18	1.75	38.0	95.0		
Athlon XP	1.733	0.18	1.75	70.0	128.0		
Pentium 4 (Willamette)	2.000	0.18	1.75	72.0	217.0		
Pentium III (Tualatin)	1.400	0.13	1.45	31.0	80.0		
Pentium 4 (Northwood)	2.200	0.13	1.50	55.0	146.0		
ASICs							
Tensilica Xtensa (Base)	0.250	0.25	2.50	0.20	1.0	typical	typical
Tensilica Xtensa (Base)	0.320	0.18	1.80	0.13	0.7	typical	typical
Lexra LX4380	0.266	0.18	1.80		1.8	typical	worst case
Lexra LX4380	0.420	0.13	1.20	0.05	0.8	typical	worst case
ARM1022E	0.325	0.13	1.20	0.23	7.9	worst case	worst case

0.13–0.25µm
Custom ~1GHz+
ASIC <500MHz

More Recently
Custom >2GHz
ASIC <1GHz

Adapted from [Chinnery'02]

Clock Frequency for High-Performance Processors



Adapted from [Chinnery'02]

ASIC (Hard-Macro) vs. Custom Low-Power Processors

Custom ARM Chips

Processor	Technology (um)	Voltage (V)	Frequency (MHz)	MIPS	Power (mW)	MIPS/mW
ARM710	0.60	5.0	40	36	424	0.08
Burd	0.60	1.2	5	6	3	1.85
Burd	0.60	3.8	80	85	476	0.18
ARM810	0.50	3.3	72	86	500	0.17
ARM910T	0.35	3.3	120	133	600	0.22
StrongARM	0.35	1.5	175	210	334	0.63
StrongARM	0.35	2.0	233	360	950	0.38
ARM920T	0.25	2.5	200	220	560	0.39
ARM1020E	0.18	1.5	400	500	400	1.25
XScale	0.18	1.0	400	510	150	3.40
XScale	0.18	1.8	1000	1250	1600	0.78
ARM1020E	0.13	1.1	400	500	240	2.08

StrongARM
0.45x Vdd
1.6x perf
0.55x power

XScale
Similar Vdd
Similar perf
0.40x power

Adapted from [Chinnery'07]

Hard vs. Soft Macro Low-Power Processors

ARM Core	Technology (um)	Frequency (MHz)	Power (mW)	MIPS/mW
ARM7TDMI	0.25	66	51	1.17
ARM7TDMI-S	0.25	60	66	0.83
ARM7TDMI	0.18	100	30	3.00
ARM7TDMI-S	0.18	90	35	2.28
ARM7TDMI	0.13	130	10	11.06
ARM7TDMI-S	0.13	120	13	8.33

ASIC designs from previous slide are “hard macros” meaning they use extra optimization for a 1.3–1.4× improvement over a straight-forward fully synthesized soft macro.

Adapted from [Chinnery’07]

Factors Contributing to the Performance Gap

FACTORS CONTRIBUTING TO SUPERIOR CUSTOM PERFORMANCE	vs. Poor ASIC	vs. Best Practice ASIC	Factor Affects
Microarchitecture: e.g. pipelining	×1.80	×1.30	# of stages, IPC
Timing overhead: clock tree design, registers, slack passing	×1.45	×1.10	$t_{timing\ overhead}$
High speed logic styles: e.g. dynamic logic	×1.40	×1.20	t_{comb}
Logic design	×1.30	×1.00	t_{comb}
Cell design and wire sizing, including transistor sizing	×1.45	×1.10	t_{comb}
Layout: floorplanning, placement, managing wires	×1.40	×1.00	overall performance
Exploiting process variation and accessibility	×2.00	×1.20	overall performance

These factors cannot really just be multiplied together. Poor ASICs are roughly 3–8× slower than custom, but with best practices we can potentially close the gap to be less than 3×.

Adapted from [Chinnery'02]

Factors Contributing to the Power Gap

Contributing Factor	Typical ASIC	Excellent ASIC
microarchitecture	5.1×	1.9×
clock gating	1.6×	1.0×
logic style	2.0×	2.0×
logic design	1.2×	1.0×
technology mapping	1.4×	1.0×
cell and wire sizing	1.6×	1.1×
voltage scaling	4.0×	1.0×
floorplanning and placement	1.5×	1.1×
process technology	1.6×	1.0×
process variation	2.0×	1.3×

These factors cannot really just be multiplied together. Poor ASICs are roughly 3–4× less power efficient than custom, but with best practices we can potentially close the gap to be less than 2.5×.

Adapted from [Chinnery'07]

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

Data, Clock, and Power Gating

Voltage Scaling

Microarchitecture: Pipelining

- What is the opportunity?
  - Pipelining reduces cycle time which can be used to improve performance or reduce energy (through reduced Vdd)
- What is the challenge for ASICs vs. custom?
  - Traditional standard-cell libs have larger timing overhead (setup, delay)
  - Traditional CAD tools had simplistic clock tree synthesis and poor support for latch-based design
- What are possible approaches for closing the gap?
  - Use standard-cell libs with high-quality state-element cells
  - Use better clock tree synthesis or even manual global clock tree design
  - Use modern CAD tools that support latch-based design and can automatically insert useful clock skew

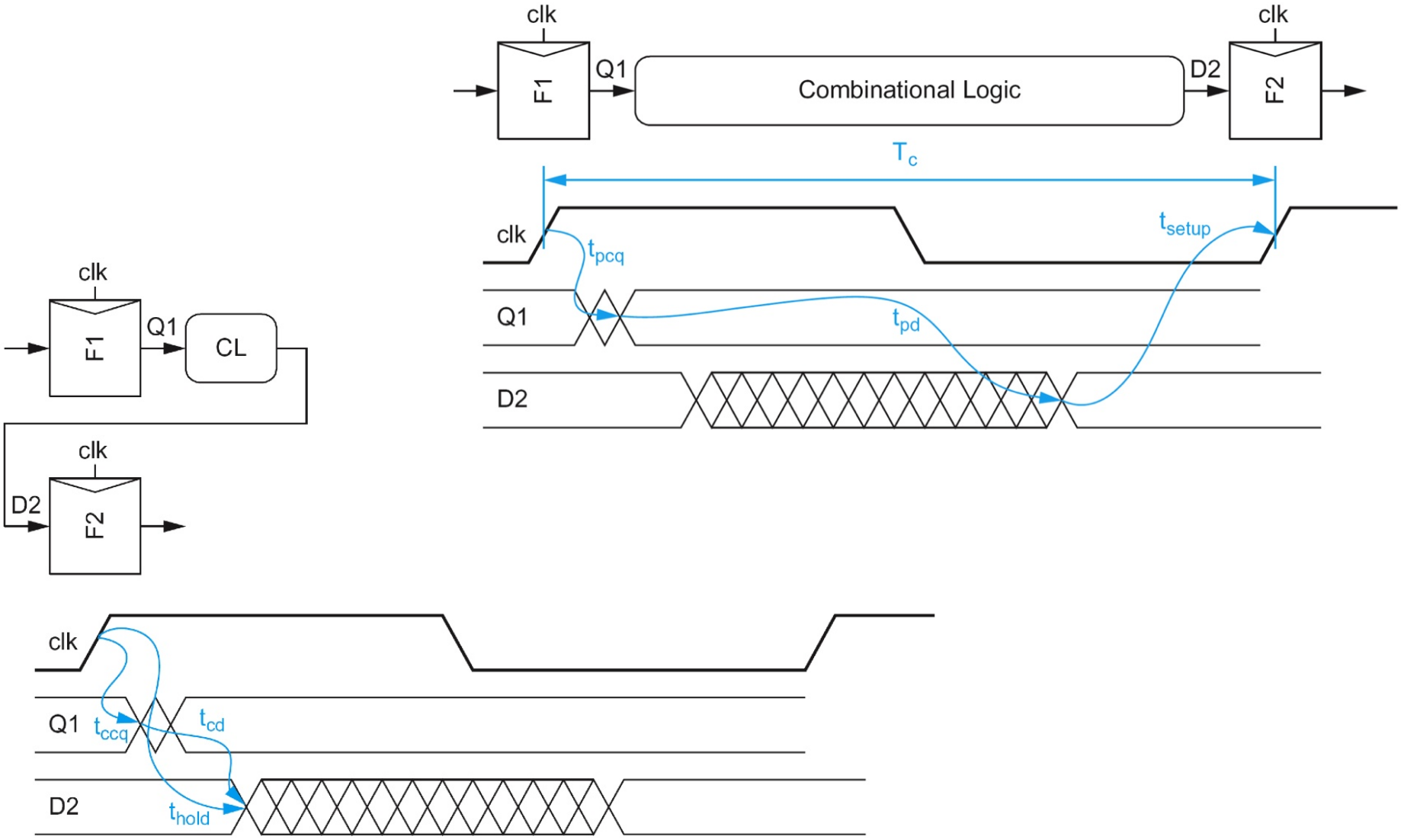
Review of Sequential Element Timing Parameters

t_{pd}	logic propagation delay (delay until new output value stable)
t_{cd}	logic contamination delay (delay until old output value changes)
t_{pcq}	latch/flip-flop clock-to-output propagation delay
t_{ccq}	latch/flip-flop clock-to-output contamination delay
t_{pdq}	latch input-to-output propagation delay
t_{cdq}	latch input-to-output contamination delay
t_{setup}	setup time (delay before sampling edge input must be stable)
t_{hold}	hold time (delay after sampling edge input must be stable)

Max-Delay Constraint: $T_C >= t_{pcq} + t_{pd} + t_{setup}$

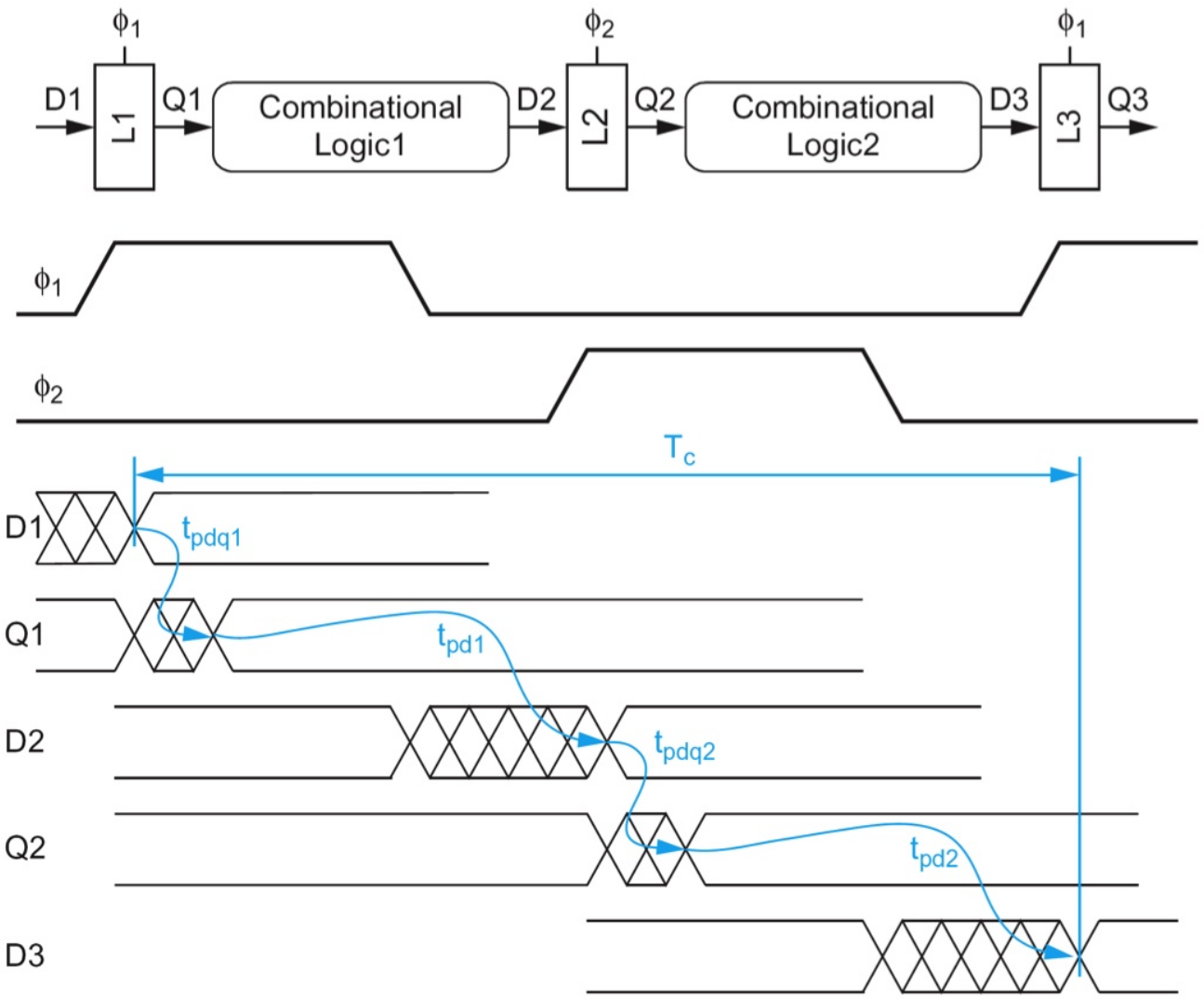
Min-Delay Constraint: $T_{hold} <= t_{ccq} + t_{cd}$

Review Flip-Flop Timing Constraints



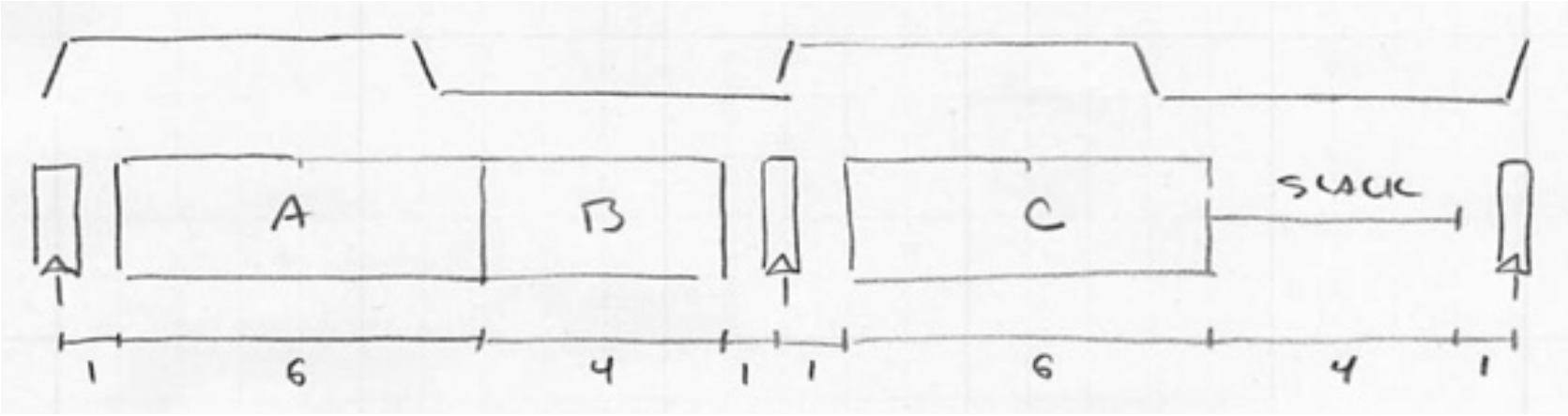
Adapted from [Weste'11]

Review Latch Timing Constraints

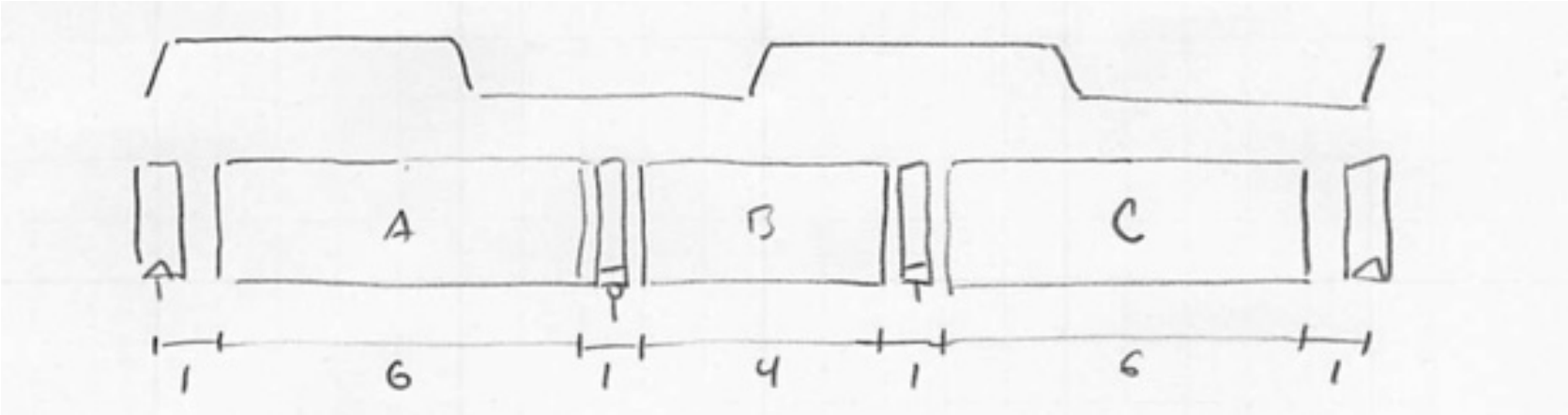


Adapted from [Weste'11]

Balancing Pipeline Stages with Latch-Based Design

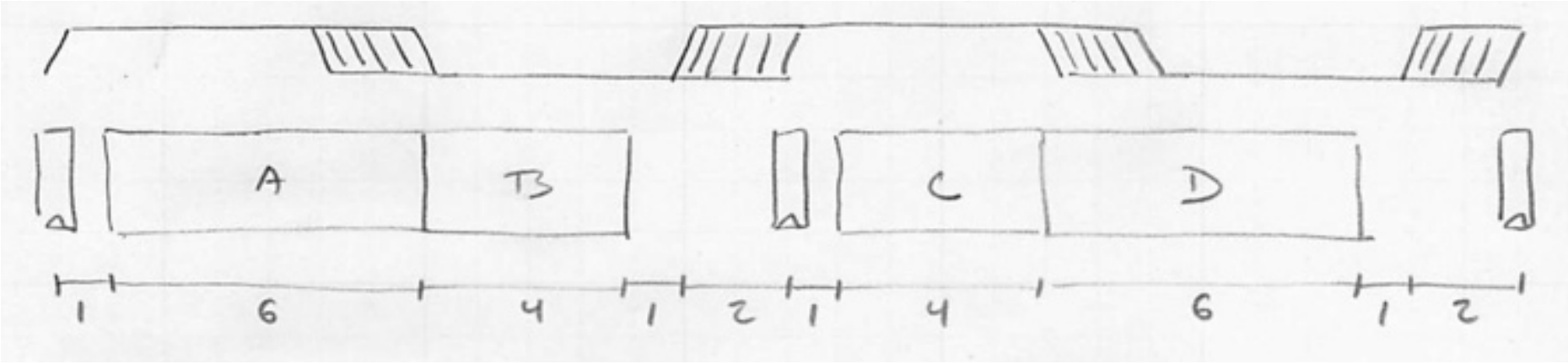


Flip-Flop-Based Approach

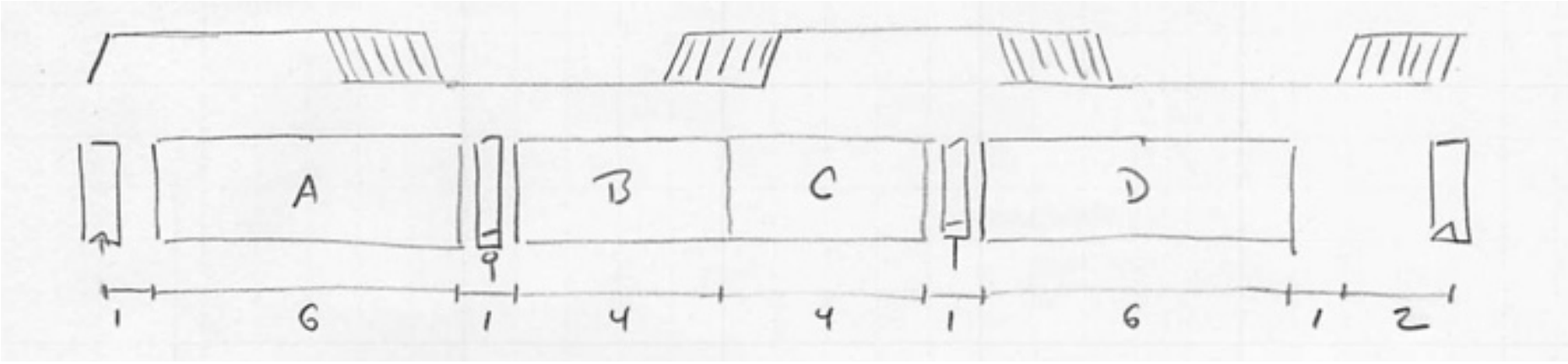


Latch-Based Approach

Timing Uncertainty with Latch-Based Design



Flip-Flop-Based Approach



Latch-Based Approach

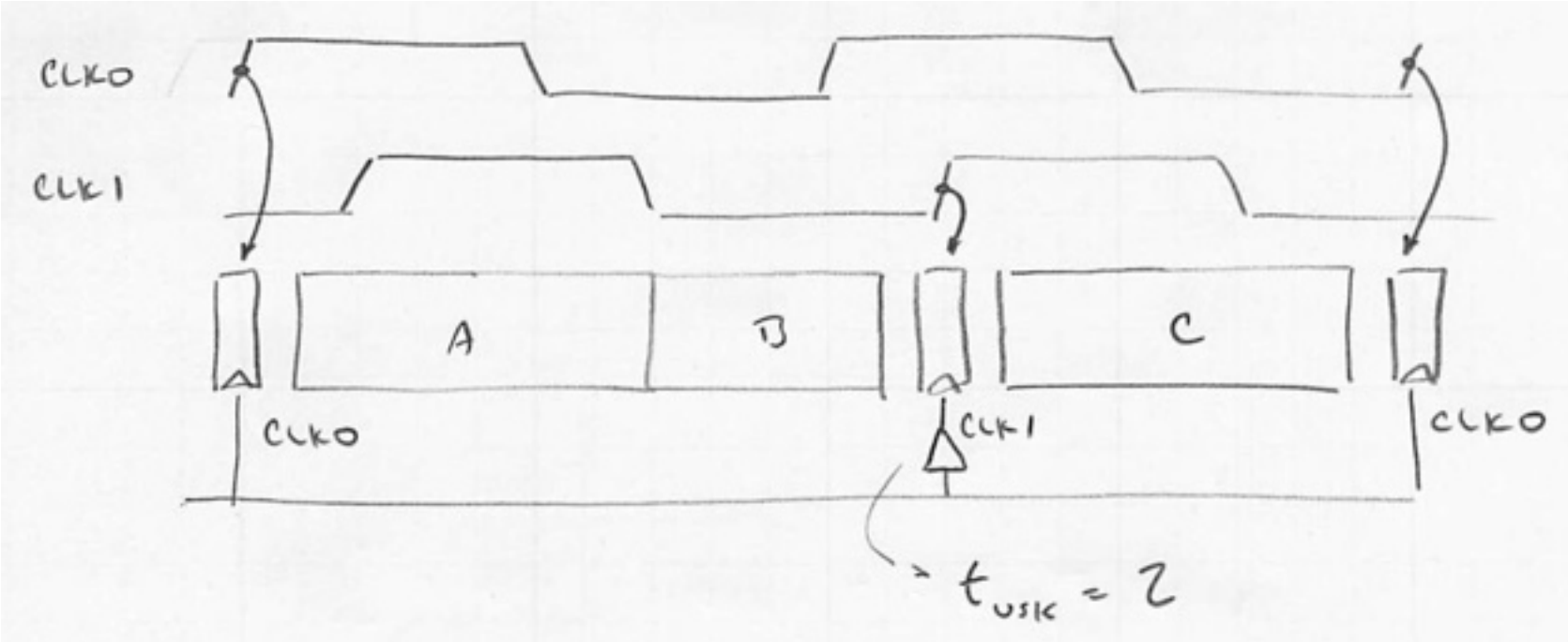
Benefit of Latch-Based Design

Design	Area (mm^2)	% Area Increase	Clock Period (ns)	% Speed Increase
Flip-flop T1030 Base Xtensa	0.367		2.58	
Latch T1030 Base Xtensa	0.379	3.3%	2.34	10.3%
Flip-flop T1040 Base Xtensa	0.382		2.67	
Latch T1040 Base Xtensa	0.402	5.2%	2.40	11.3%
Flip-flop T1040 Base_no_loop_no_pif	0.246		2.60	
Latch T1040 Base_no_loop_no_pif	0.272	10.6%	2.18	19.3%
Flip-flop T1030 MAC16	0.440		2.61	
Latch T1030 MAC16	0.452	2.7%	2.46	6.1%

These results are for a tool which automatically converts flip-flops into latches and then retimes the latches for time borrowing and to hide clock uncertainty

Adapted from [Chinnery'02]

Balancing Pipeline Stages with Useful Clock Skew



Useful Clock Skew

	Flip-Flops	Period (ns)	Slack (ns)	Buffers
Zero-Skew	16,770	5.5	-0.2	1,036
Useful-Skew	16,770	5.5	0.3	1,048

Graphics Processor using Artisan library in 0.15 µm TSMC process

Adapted from [Dai'02]

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

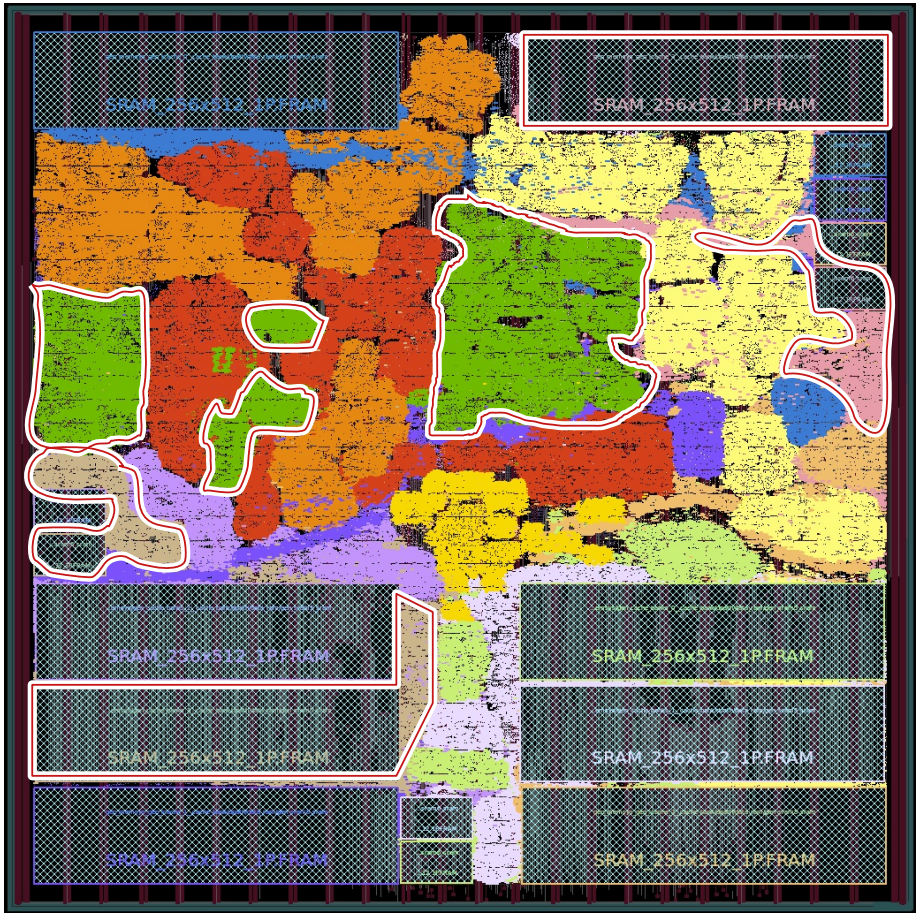
Data, Clock, and Power Gating

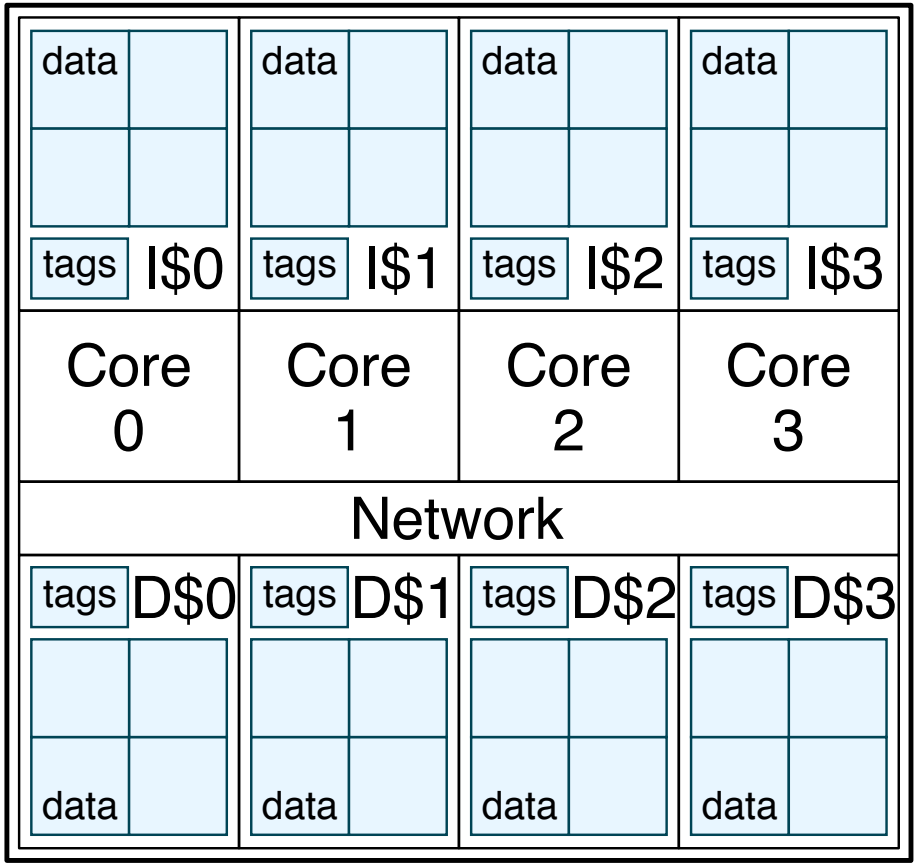
Voltage Scaling

Floorplanning and Placement

- What is the opportunity?
  - Exploiting modularity, hierarchy, and regularity in the physical domain can significantly reduce wire lengths which improves cycle time and reduces energy (less interconnect capacitance) [bonus: reduced tool exec time]
- What is the challenge for ASICs vs. custom?
  - Support for hierarchical design in traditional CAD tools is cumbersome leading designers to stick with a fully flat methodology
  - Wire load models are notoriously inaccurate, so synthesis tools either optimize for wire lengths that are too long or too short
- What are possible approaches for closing the gap?
  - Use modern CAD tools with soft/hard macro-floorplanning and support for hierarchical design with individually implemented macro-blocks
  - Use modern CAD tools with support for physical synthesis
  - Consider fine-grain standard-cell tiling with automatic routing

Macro-Floorplanning

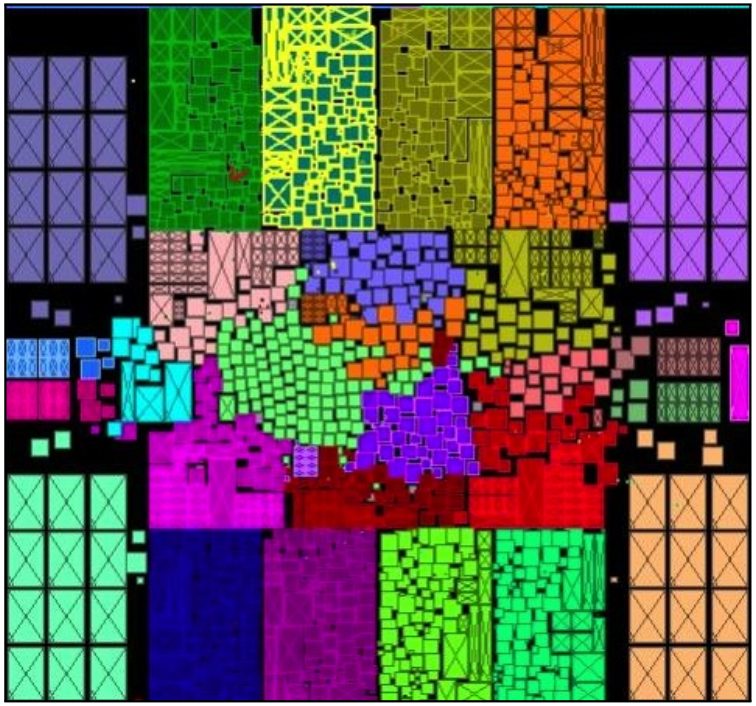




Core 0, I$0, D$0 highlighted

Physical Synthesis

- Physical synthesis tool performs first-iteration mapping to logic gates before performing a preliminary "rough" placement
- Preliminary placement incorporates macro-floorplanning and does not need to produce a "legal" fine-grain placement of each cell



- Preliminary placement is used to create more accurate wire load estimates, and then tool re-synthesizes design
- Physical synthesis tool iterates until meets timing

Fine-Grain Tiling of Standard Cells

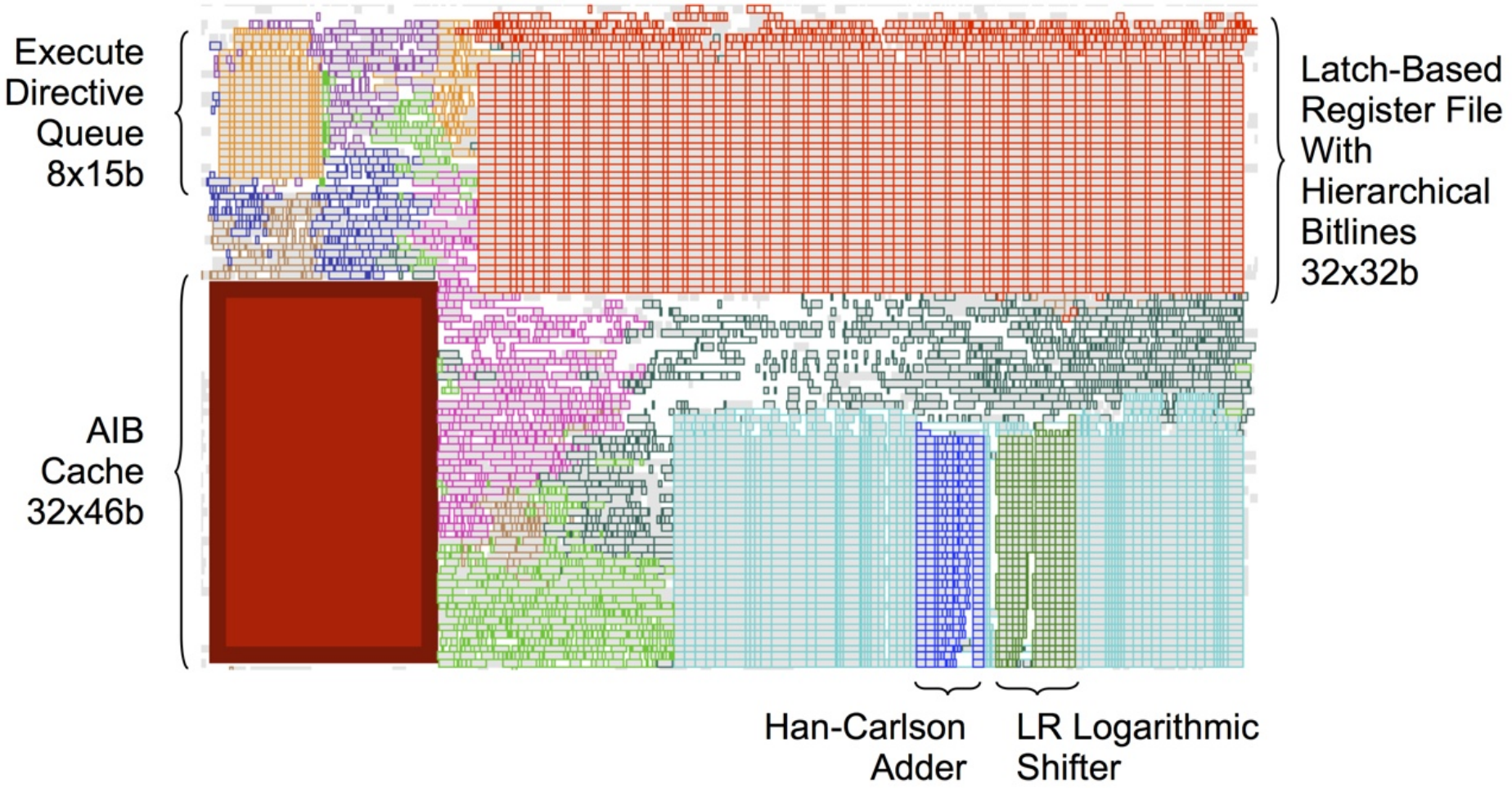
Purely Synthesized Datapath

36% of area
60% of delay

Tiled Standard Cells
w/ Automatic Routing

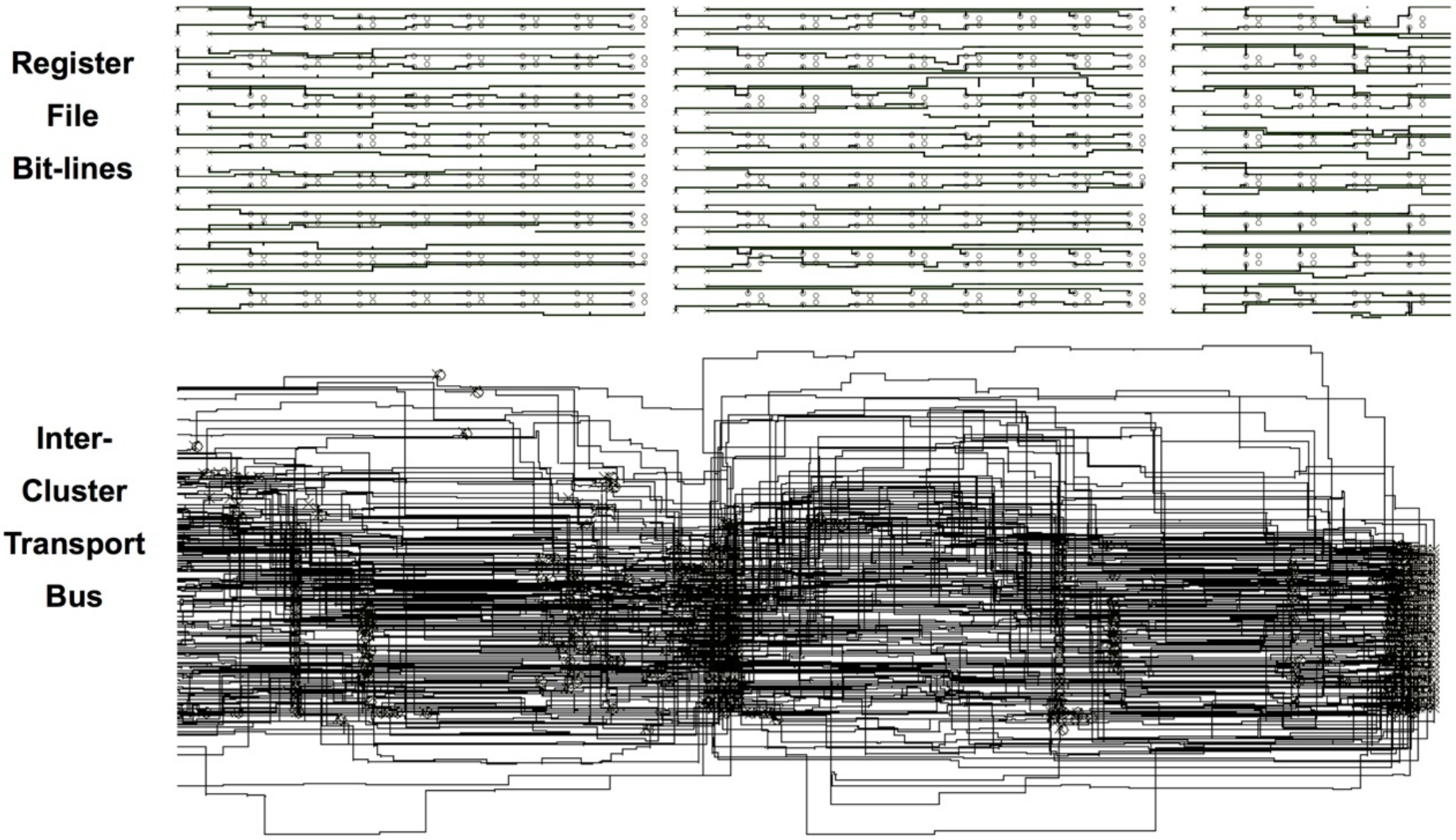
Adapted from [Chang'02]

Scale Processor: Fine-Grain Tiling of Standard Cells



230,000 cells were pre-placed across entire chip
58% of all standard cells

Scale Processor: Automatic Routing for Tiling



Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

Data, Clock, and Power Gating

Voltage Scaling

Cell Sizing

- What is the opportunity?
  - Gates with transistors optimally sized according to their context will improve cycle time and reduce energy
- What is the challenge for ASICs vs. custom?
  - Standard-cell libs have a fixed set of gate sizings
- What are possible approaches for closing the gap?
  - Use standard-cell libs with a diverse range of sizings (both large sizes for high performance, and small sizes for low power)
  - Consider emerging CAD tools with support for regenerating customized standard-cells either post place-and-route or as part of logic synthesis

SAED 90 nm Library – NAND Gate Data Sheet

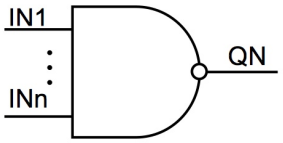


Figure 10.6. Logic Symbol of NAND

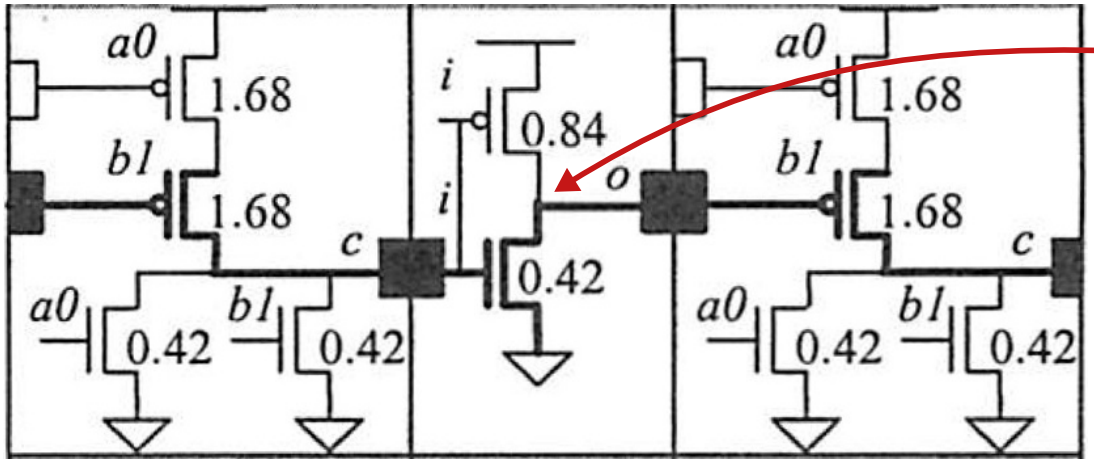
Table 10.11. NAND Truth Table (n=2,3,4)

IN1	IN2	. . .	INn	QN
0	X	. . .	X	1
X	0	. . .	X	1
. . .	. . .	. . .	. . .	1
X	X	. . .	0	1
1	1	1	1	0

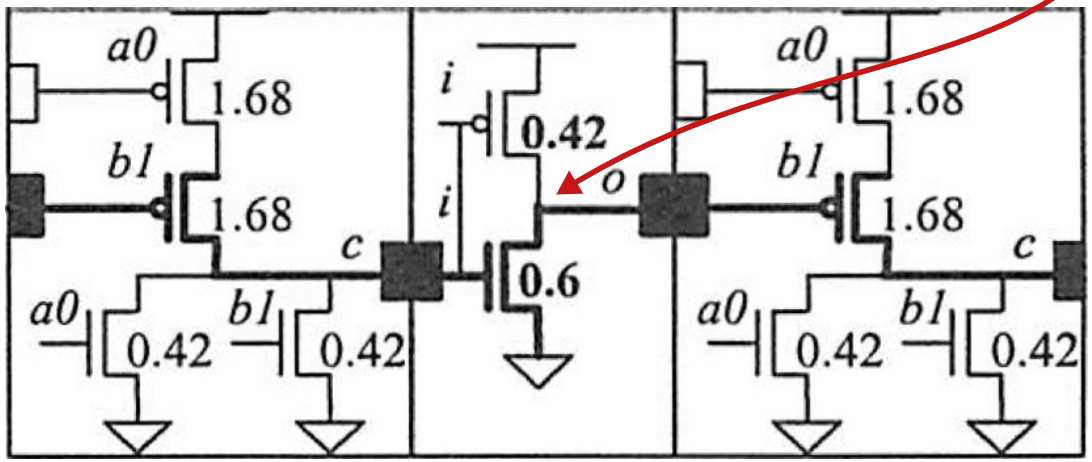
Cell Name	Operating Conditions: VDD=1.2 V DC, Temp=25 Deg.C, Operating Frequency: Freq=300 MHz, Capacitive Standard Load: Csl=13 fF				Area
	Cload	Prop Delay (Avg)	Power		
			Leakage (VDD=1.2 V DC, Temp=25 Dec.C)	Dynamic	
		ps	nW	nW/MHz	(um^2)
NAND2X0	0.5 x Csl	140	38	3583	5.5296
NAND2X1	1 x Csl	132	78	5208	5.5296
NAND2X2	2 x Csl	126	157	9191	9.2160
NAND2X4	4 x Csl	125	314	17902	14.7456
NAND3X0	0.5 x Csl	128	91	5331	7.3728
NAND3X1	1 x Csl	192	102	12200	11.9808
NAND3X2	2 x Csl	212	155	19526	12.9024
NAND3X4	4 x Csl	241	260	44937	15.6672
NAND4X0	0.5 x Csl	147	106	5357	8.2944
NAND4X1	1 x Csl	178	161	15214	12.9024

Adapted from [SAED'11]

Critical Path Optimization with Transistor Sizing



Original Implementation After Place-and-Route



New Implementation After Generating Custom Cell

Critical path goes through falling transition of inverter

Custom inverter cell has been created to optimize falling path

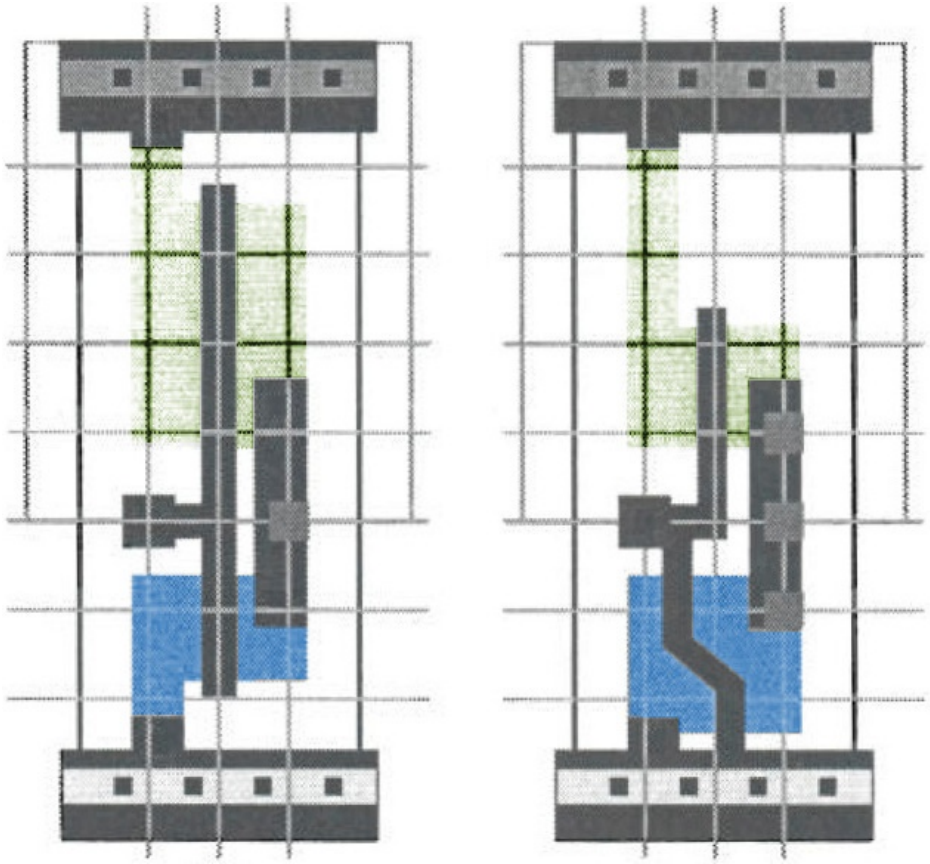
NMOS larger: lower effective R

PMOS smaller: lower parasitic C

Rising transition of inverter is now slower

Adapted from [Côté'02]

Modifying Cell Layout in Same Footprint

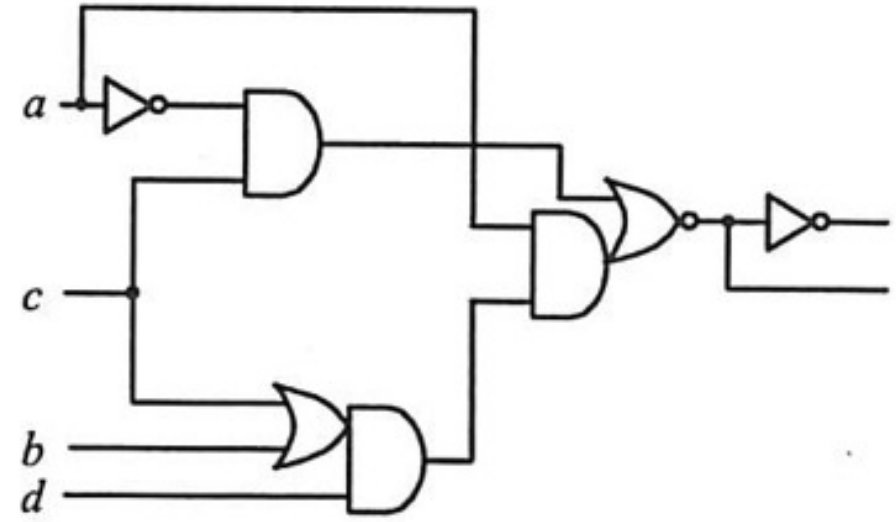


Original Cell Resized Cell

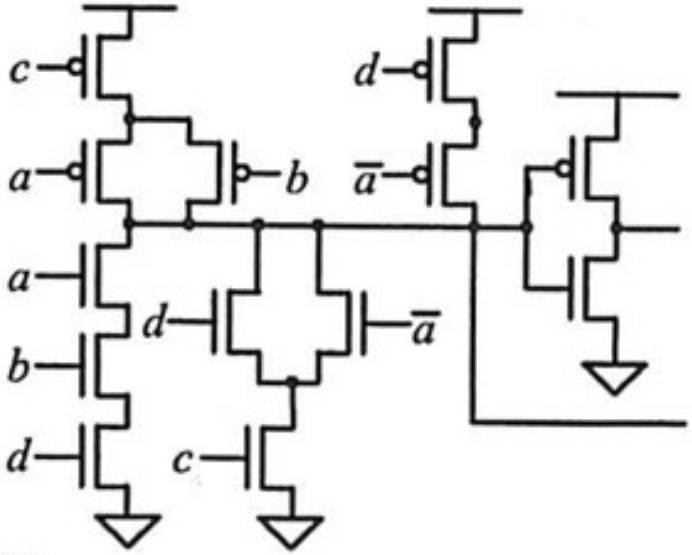
- Cells are resized after place-and-route on critical paths
- Cells keep same footprint including pin locations
- In this example, NMOS size is increased while PMOS size is decreased to accelerate pull-down path
- Example bus controller with 12K gates in 0.3 µm: reduced critical path by 13.5% and reduced power by 18% by generating 300 optimized cells

Adapted from [Côté'02]

Automatically Creating Complex Cells



(a)



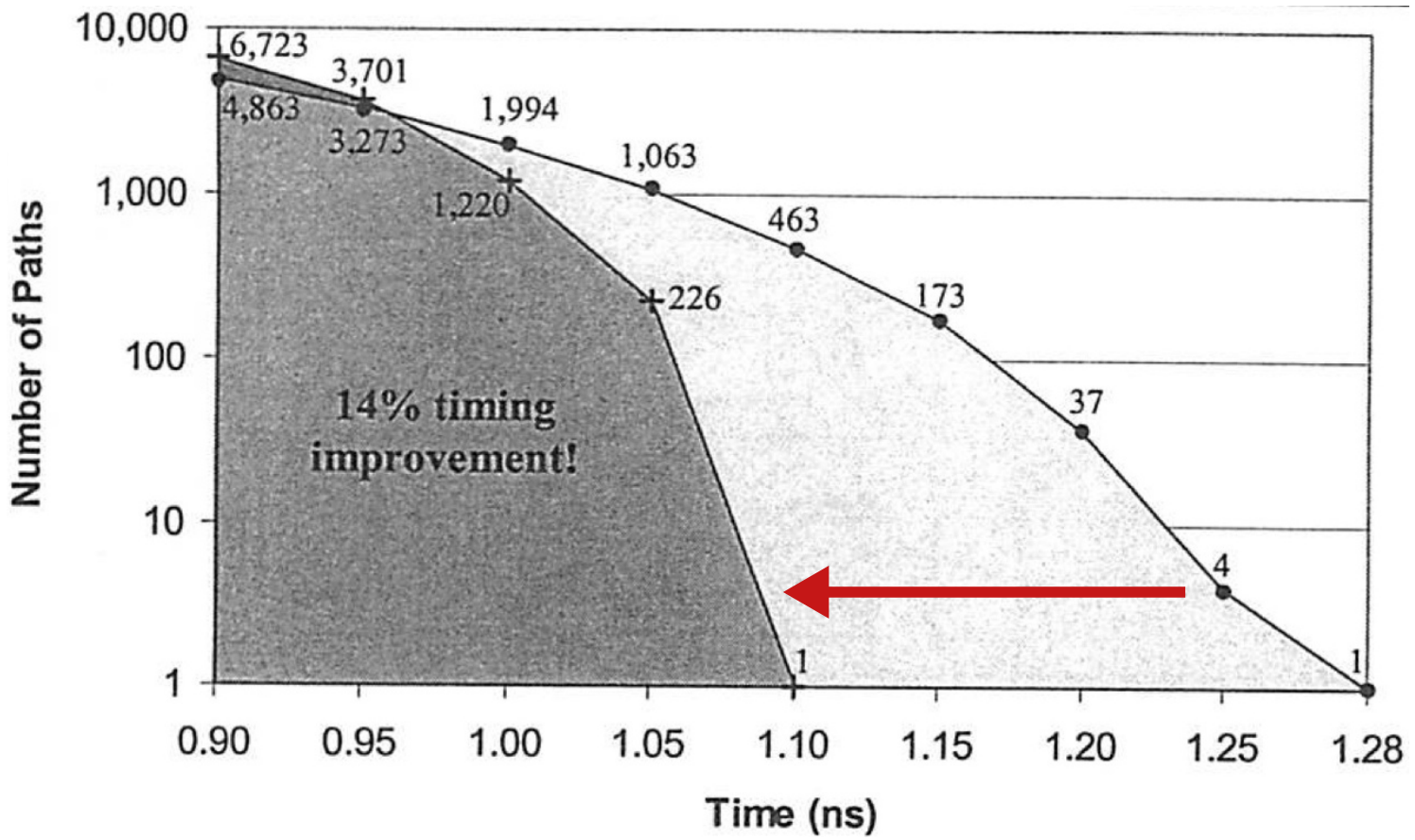
(b)

(c)

Transition on Input #	Original		Optimized	
	Rise Time (ns)	Fall Time (ns)	Rise Time (ns)	Fall Time (ns)
a	0.29	0.34	0.13	0.11
b	0.18	0.30	0.17	0.13
c	0.18	0.31	0.16	0.15
d	0.18	0.27	0.15	0.14

Adapted from [Bhattacharya'02]

Benefit of Automatic Resizing



Improvement for customized standard-cell generation during logic synthesis for an adder design in 0.18 µm

Adapted from [Bhattacharya'02]

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

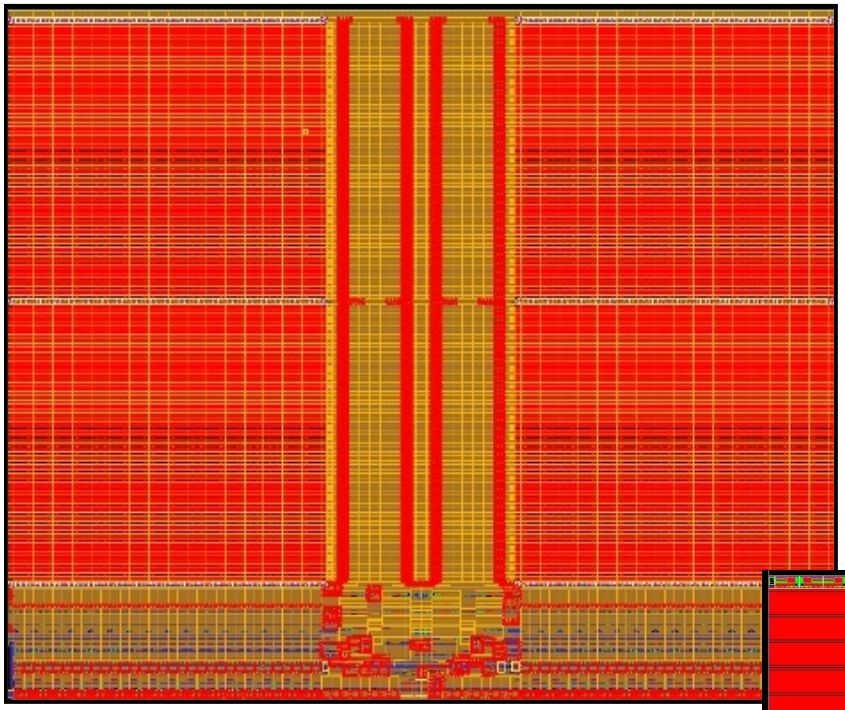
Data, Clock, and Power Gating

Voltage Scaling

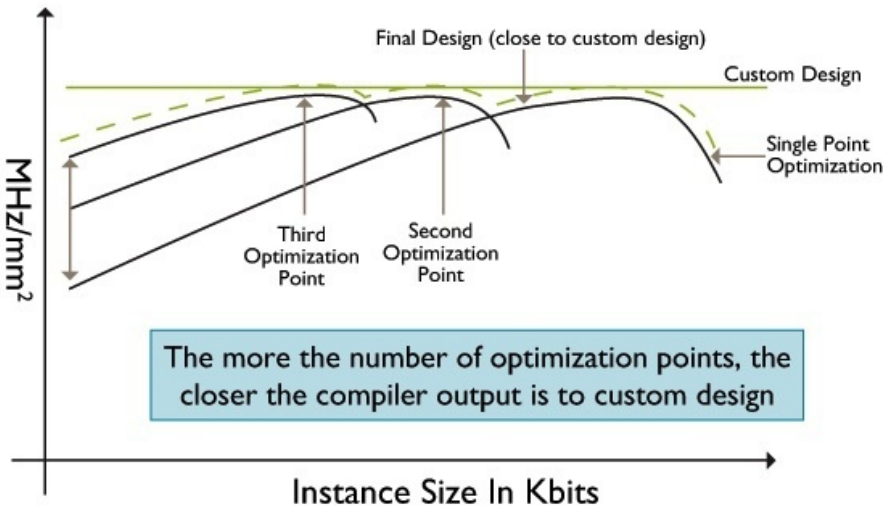
High-Speed Logic Styles

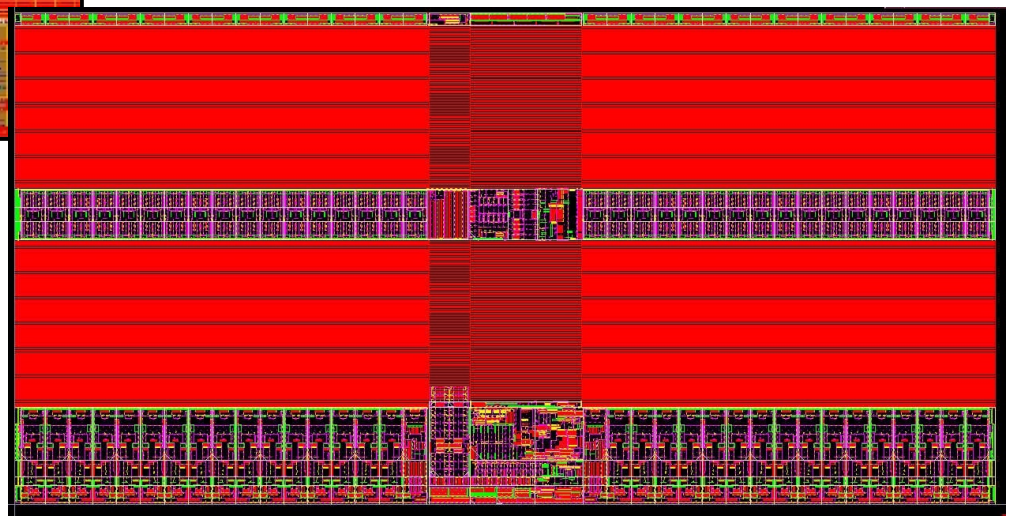
- What is the opportunity?
  - More aggressive logic styles (e.g., DCVSL, pass-transistor logic, dynamic domino logic, dynamic bitlines in memories) can significantly reduce cycle time compared to static CMOS logic
- What is the challenge for ASICs vs. custom?
  - Aggressive logic styles often require carefully designing the context for each cell (glitching, precharge/evaluate, and noise for dynamic logic; cascaded voltage drops and effective resistances for pass-transitor logic)
- What are possible approaches for closing the gap?
  - Use SRAM and register-file generators whenever possible
  - Consider creating crafted cells for critical blocks

SRAM and Register File Memory Compilers



Multi-Ported Generated Register File

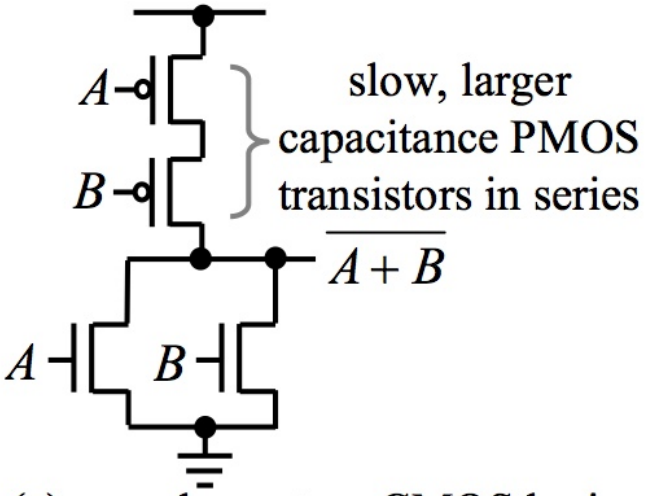




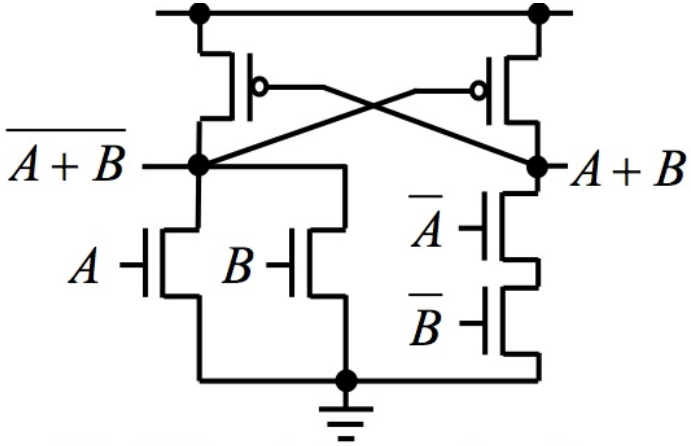
Single-Ported Generated SRAM

Adapted from [Artisan]

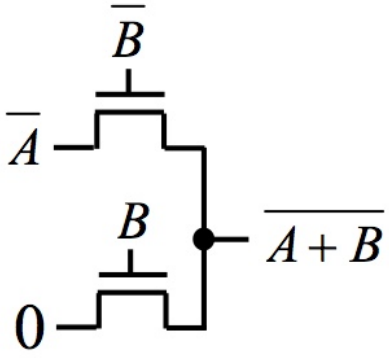
High-Speed Logic Styles



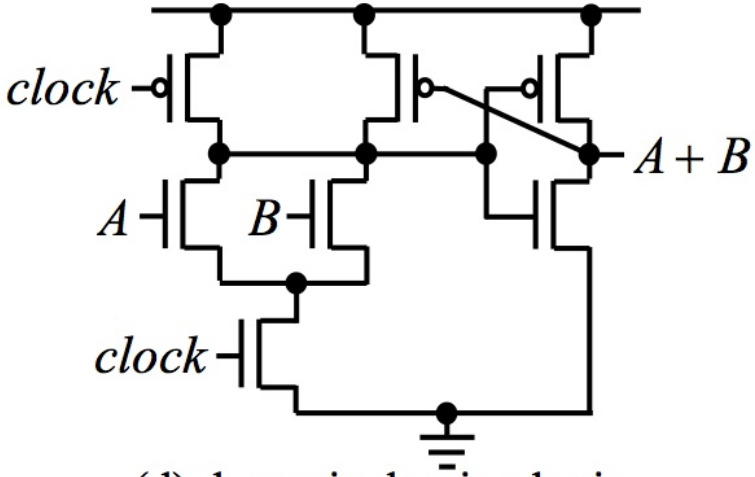
(a) complementary CMOS logic



(b) differential cascode voltage switch logic (DCVSL)



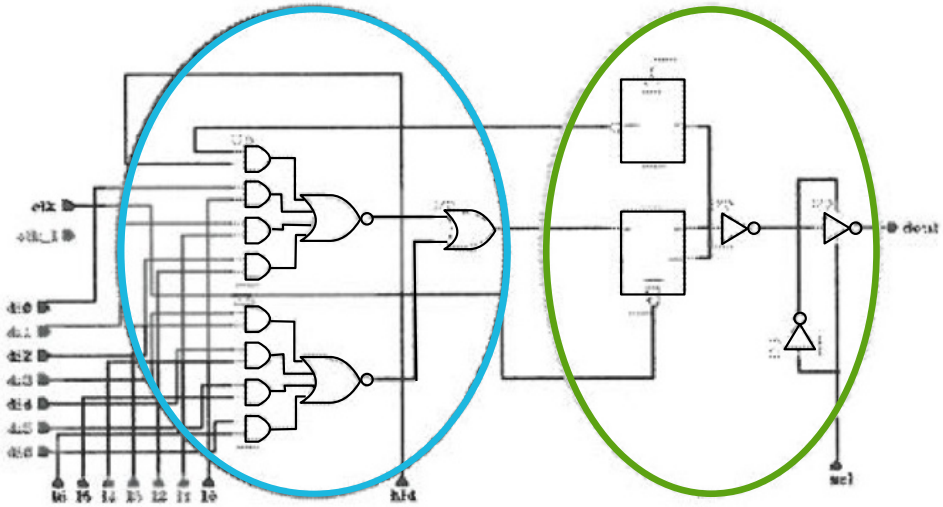
(c) pass transistor logic (PTL)

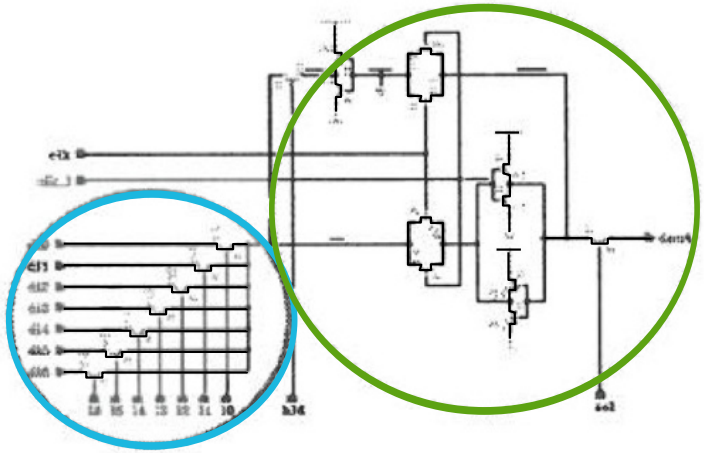


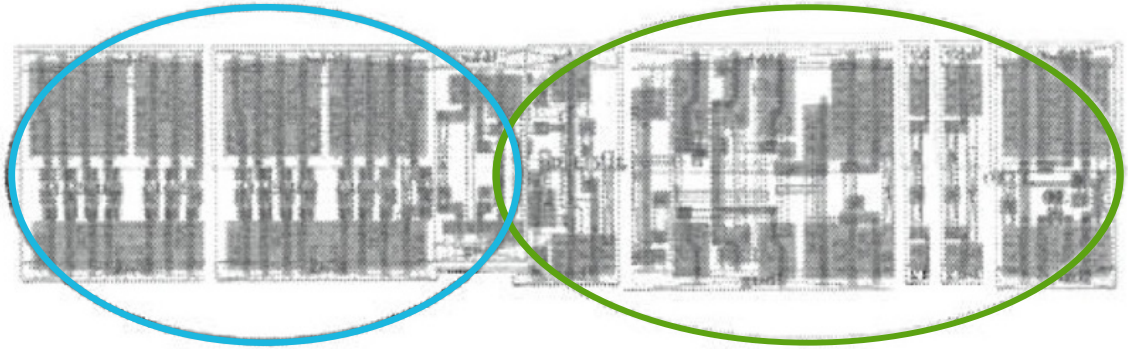
(d) dynamic domino logic

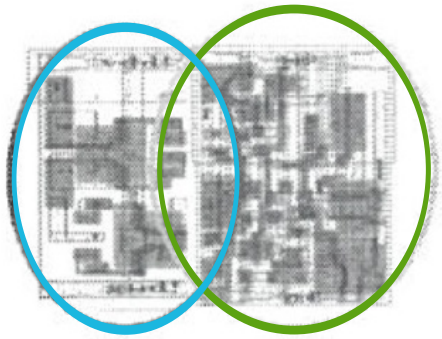
Adapted from [Chinnery'02]

7-Input Multiplexed Latch w/ Tri-State Output







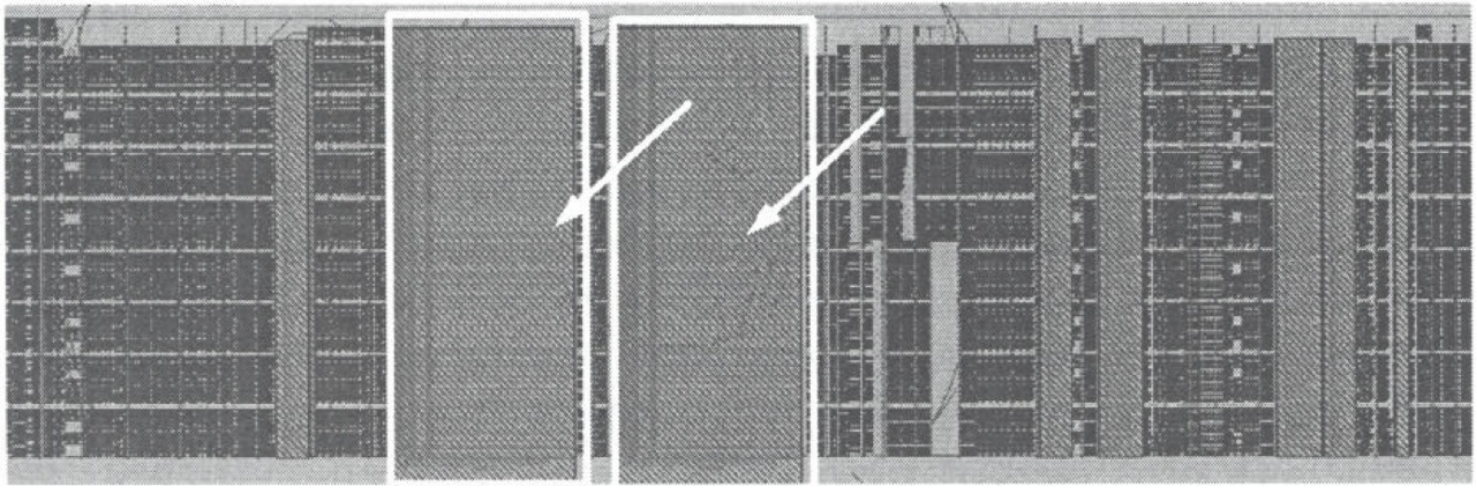


Implemented with Seven Standard Cells
40 transistors
7x larger than full-custom

Implemented with Two Crafted Cells
19 transistors
1.8x larger than full-custom

Adapted from [Chang'02]

64-bit FPU with Domino-Logic Crafted Multiplier



Domino-Logic Crafted 64x64 Multiplier

Tiled Static CMOS Logic

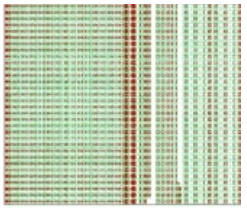
Multiplier	FO4 delays	Area (10^6 χ2)	Relative Area	Transistors	Area Efficiency (χ2/T)
Mitsubishi	24.3	1.53	0.82		
NEC Multiplier	27.0	2.19	1.18	135,318	16.2
MIT/Stanford	29.8	1.86	1.00	124,574	14.9
Fujitsu	32.5	1.63	0.88	82,500	19.8
Mitsubishi	35.2	2.35	1.26	78,800	29.8
Fujitsu Multiplier	41.0	1.65	0.89	60,797	27.1

Adapted from [Chang'02]

Benefit of Crafted Cells

Purely Synthesized Datapath

11% of area
30% of delay



Tiled Standard Cells
w/ Several Crafted Cells
and Automatic Routing

Adapted from [Chang'02]

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

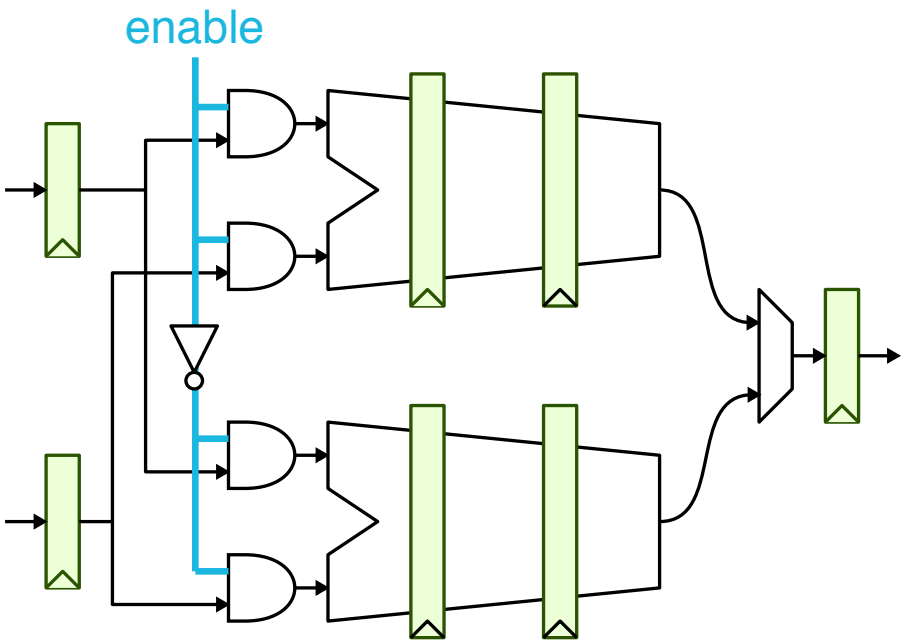
Data, Clock, and Power Gating

Voltage Scaling

Data, Clock, and Power Gating

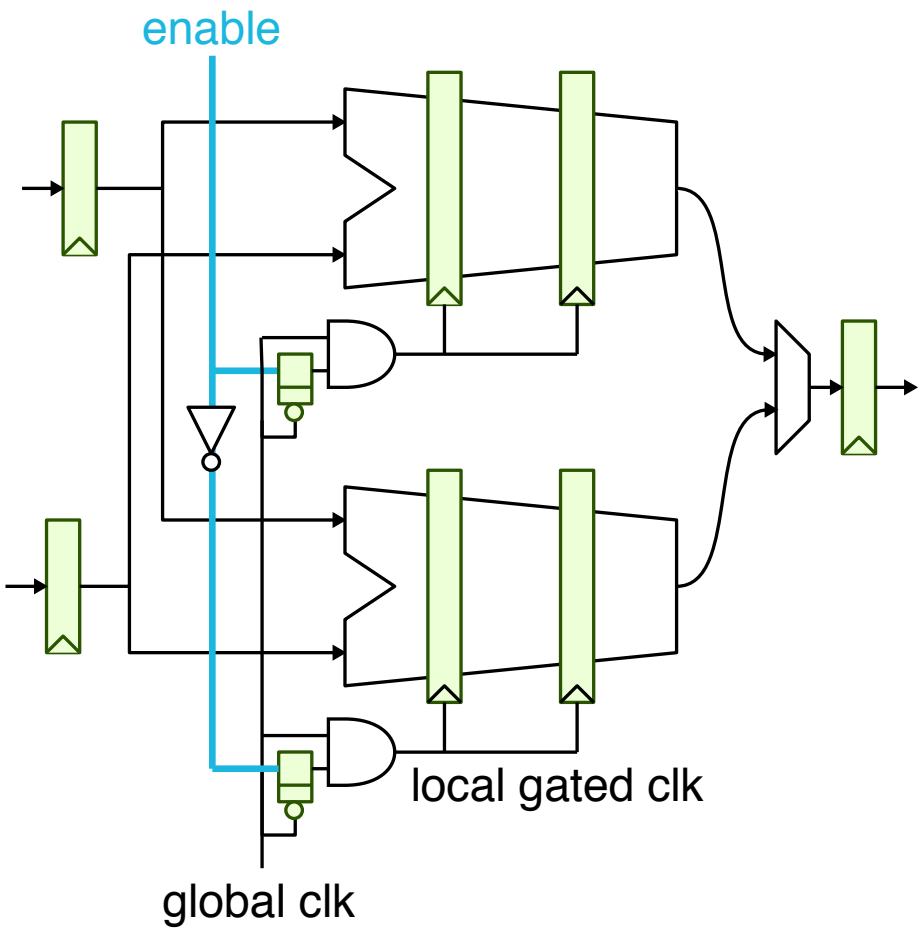
- What is the opportunity?
  - Eliminating activity with data and clock gating reduces dynamic energy and disconnecting power supply with power gating reduces static power
- What is the challenge for ASICs vs. custom?
  - Traditional standard-cell libs may not have clock and power gating cells
  - Traditional CAD tools have poor support for logic in the clock network and for expressing power gating at the register-transfer level
  - Creating data, clock, and power gating enable signals which do not impact timing can be difficult
- What are possible approaches for closing the gap?
  - Use standard-cell libs with diverse clock and power gating cells
  - Use modern CAD tools with support for automatic clock gating
  - Use modern design formats to specify low-power intent (e.g., UPF)

Data Gating



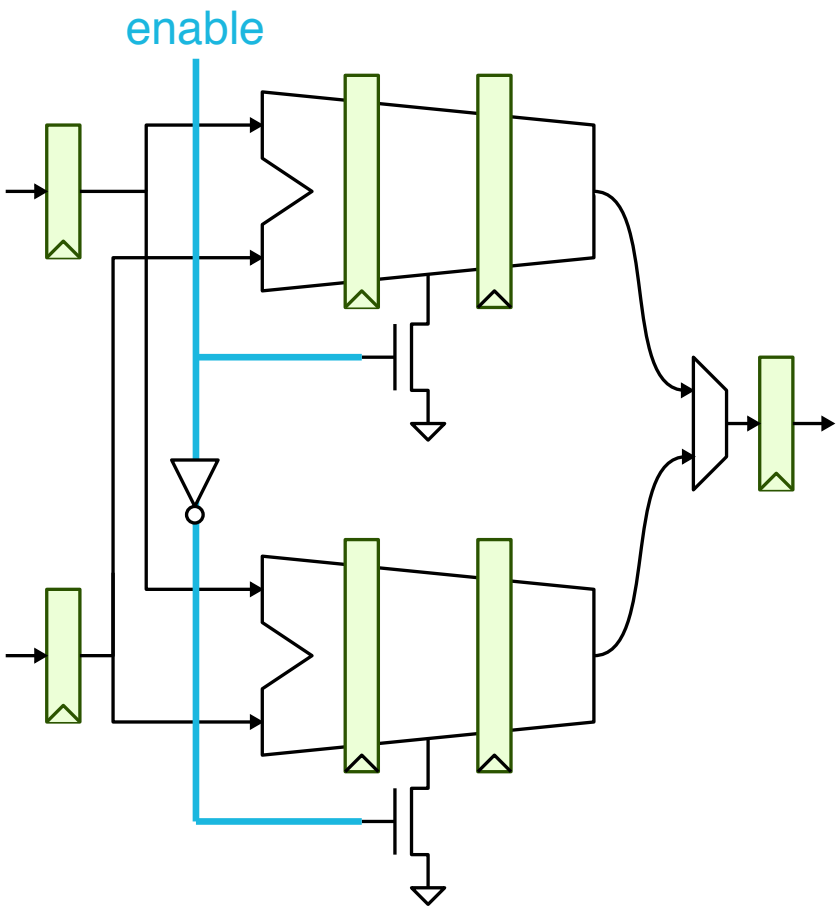
- Use enable signal to prevent data bits from toggling when module is inactive
- Adds delay of AND gate to data path
- Enable signal needs to be ready early in the cycle

Clock Gating



- Use enable signal to prevent local clock tree from toggling when module is inactive
- Fine-grain clock gating: single register with enable signal, can be automatically inferred by CAD tools
- Coarse-grain clock gating: entire module, must be managed explicitly in the RTL
- Clock gating in StrongARM processor reduced power consumption by 33%

Power Gating



- Use enable signal to disconnect module from power or ground
- Reduces leakage, but extra energy/delay to switch sleep transistor and charge up internal cap upon wakeup
- Break-even point can be $>$10K cycles in a large module, so power gating must be used carefully
- Power gating in XScale processor reduced leakage by 5$\times$ while retaining state

Agenda

Exploring the Performance and Power Gap

Microarchitecture: Pipelining

Floorplanning and Placement

Cell Sizing

High-Speed Logic Styles

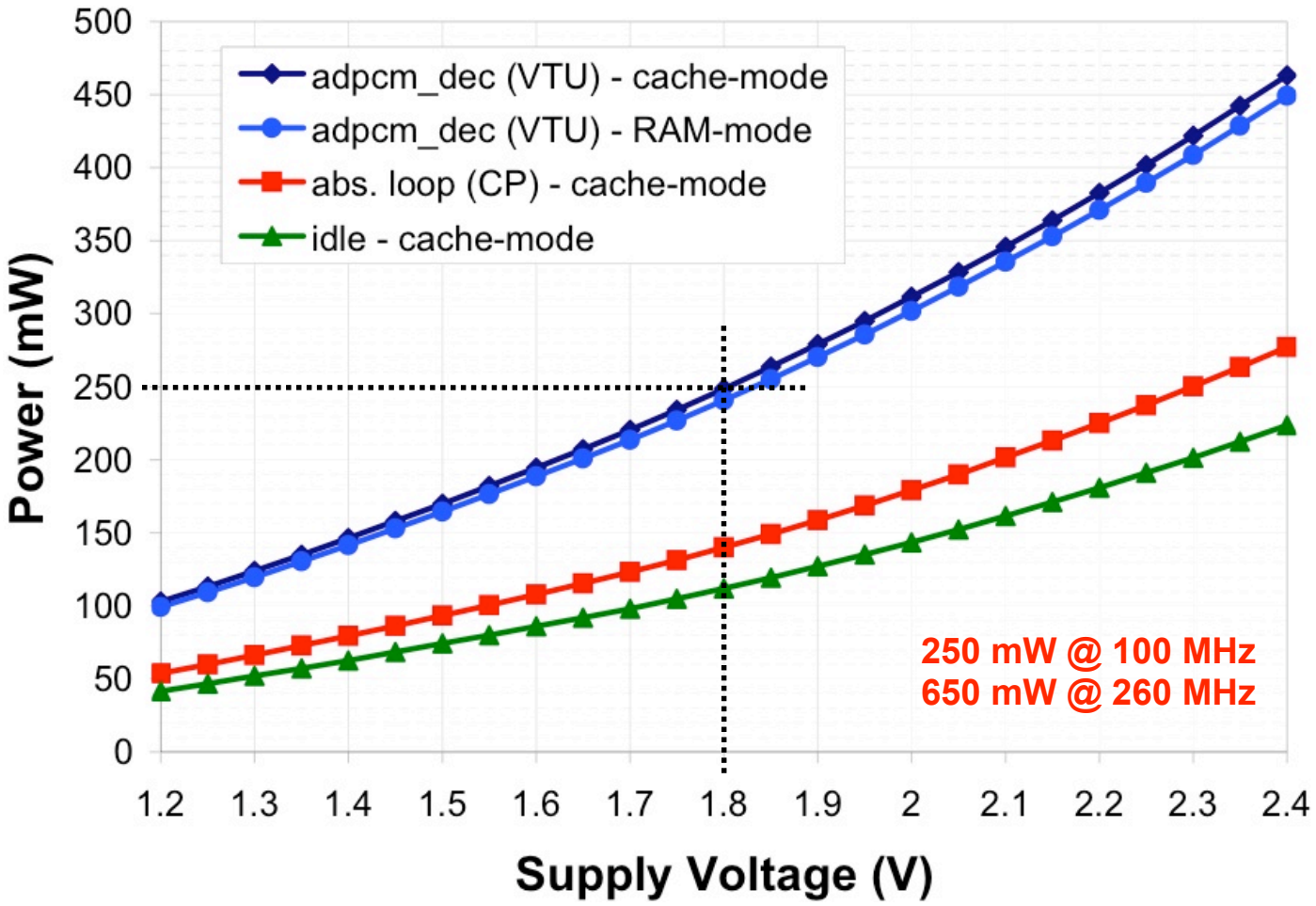
Data, Clock, and Power Gating

Voltage Scaling

Voltage Scaling

- What is the opportunity?
  - Reducing the supply voltage and frequency can result in significant energy and power savings at reduced performance
- What is the challenge for ASICs vs. custom?
  - Traditional standard-cell libs are usually characterized at one supply at each process and temperature corner
  - Dynamic voltage and frequency regulation can require tight mixed-signal integration
- What are possible approaches for closing the gap?
  - Consider emerging CAD tools with support for multi-Vdd and multi-Vt cells
  - Consider limited mixed-signal integration for applications where DVFS can have significant energy/performance benefits

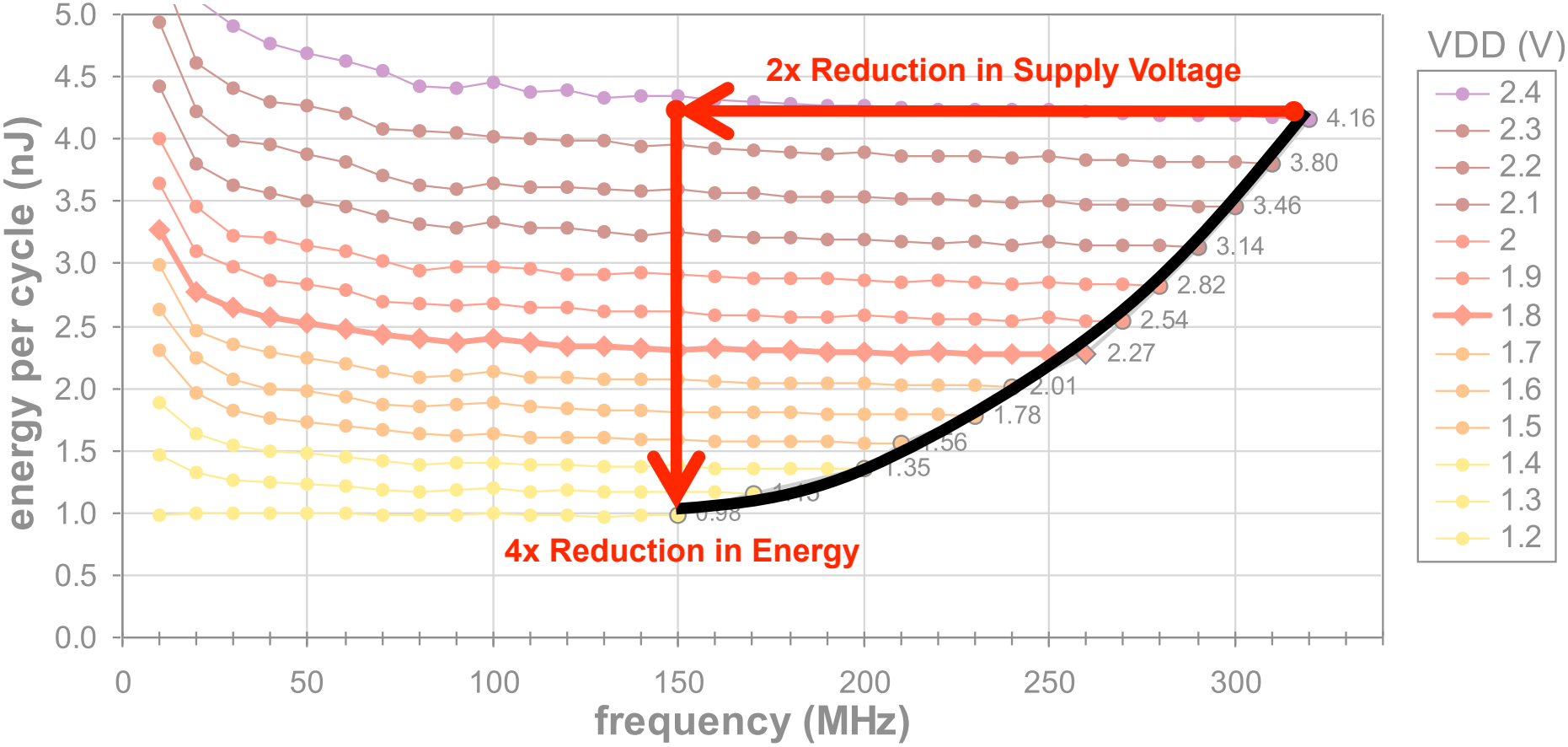
Scale Processor: Power vs Voltage at 100 MHz



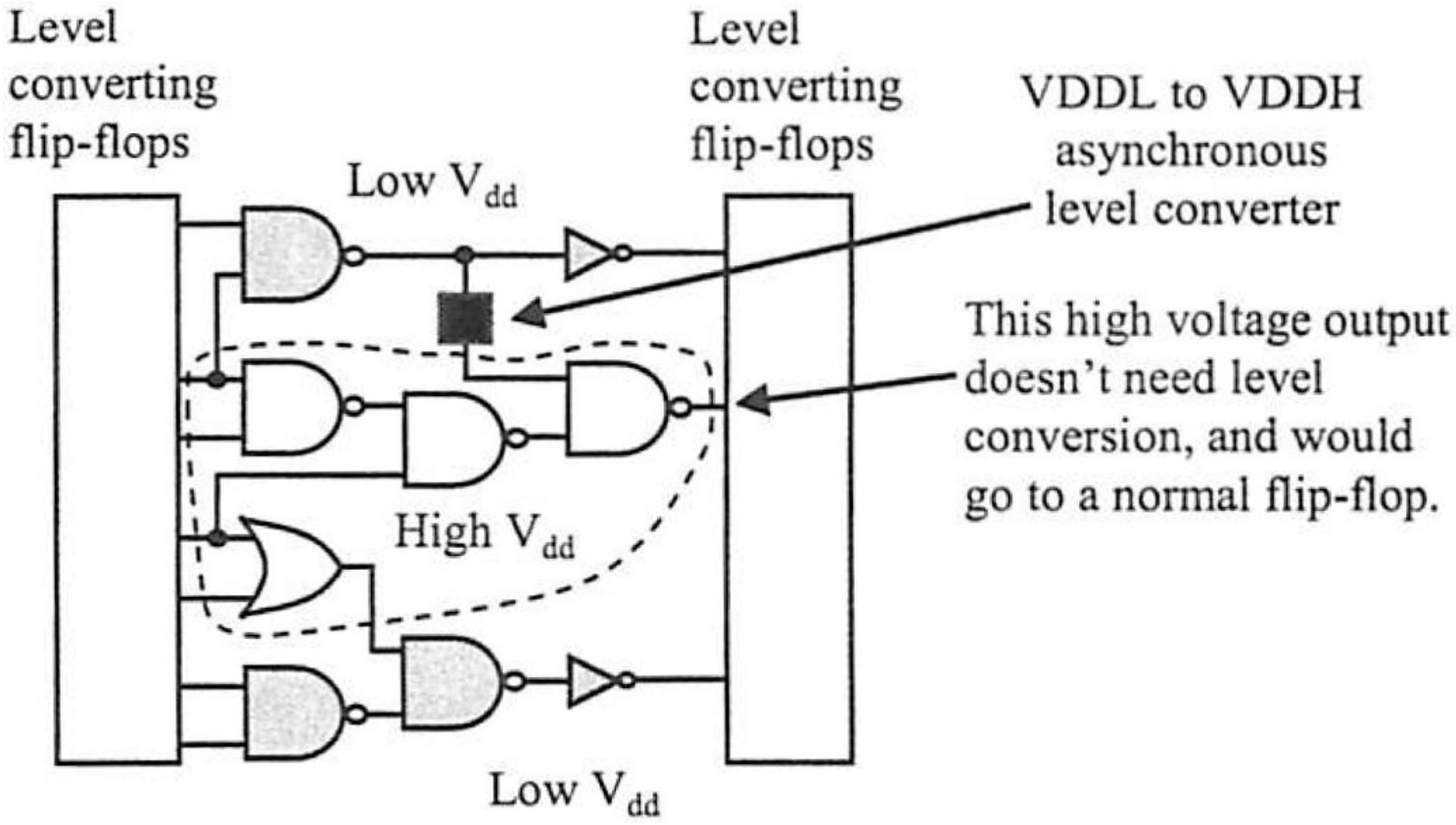
Control Processor Absolute Value Loop

```
  la t9, input_end
forever:
  la t8, input_start
  la t7, output
loop:
  lw t0, 0(t8)
  addu t8, 4
  bgez t0, pos
  subu t0, $0, t0
pos:
  sw t0, 0(t7)
  addu t7, 4
  bne t8, t9, loop
  b forever
```

Scale Processor: Energy vs Frequency and Voltage



Static Fine-Grain Clustered Voltage Scaling



Tools can automatically use low-Vdd and/or low-Vt standard cells for logic gates off the critical path

Adapted from [Kulkarni'07]

Benefit of Static Fine-Grain Clustered Voltage Scaling

	VDDL = 0.6V		VDDL = 0.8V	
Circuit	CVS	GECVS	CVS	GECVS
c432	1.0%	1.5%	0.8%	0.8%
c880	8.2%	10.3%	15.0%	21.3%
c1355	0.0%	0.0%	0.0%	1.0%
c1908	4.3%	7.7%	3.4%	8.4%
c2670	21.1%	25.5%	16.5%	25.0%
c3540	3.2%	8.3%	2.9%	9.7%
c5315	7.6%	19.0%	8.3%	22.0%
c7552	14.9%	20.2%	22.0%	28.8%
Huffman	6.6%	12.7%	6.7%	14.4%
Average	**7.4%**	**11.7%**	**8.4%**	**14.6%**

Nearly 15% reduction in dynamic power just by using low-Vdd (0.8 V) for logic gates off the critical path (high-Vdd is 1.2 V in this 0.13 µm process); notice that using an even lower Vdd (0.6 V) has less power savings due to the overhead of level converters

Adapted from [Kulkarni'07]

Acknowledgments

- [Artisan] "ARM Artisan – Embedded Memory IP," 2013. `http://www.arm.com/products/physical-ip/embedded-memory-ip`
- [Bhattacharya'02] D. Bhattacharya and V. Boppana, "Design Optimization with Automated Flex-Cell Creation," Chapter 10 in [Chinnery'02].
- [Chang'02] A. Chang and W.J. Dally, "Exploiting Structure and Managing Wires to Increase Density and Performance," Chapter 11 in [Chinnery'02].
- [Chinnery'02] D. Chinnery and K. Keutzer, "Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design," Springer, 2002.

Acknowledgments

- [Chinnery'07] D. Chinnery and K. Keutzer, "Closing the Power Gap Between ASIC & Custom: Tools and Techniques for Low-Power ASIC Design," Springer, 2002.
- [Côté'02] M. Côté and P. Hurat, "Fater and Lower Power Cell-Based Designs with Transistor-Level Cell Sizing," Chapter 9 in [Chinnery'02].
- [Kulkarni'07] S. Kulkarni et al., "Power Optimization Using Multiple Supply Voltages," Chapter 8 in [Chinnery'07].
- [Dai'02] W. Dai and D. Staepelaere, "Useful-Skew Clock Synthesis Boosts ASIC Performance," Chapter 8 in [Chinnery'02].
- [Weste'11] N. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 4th ed, Addison Wesley, 2011.