18-643 Lecture 12: Reconfiguration and All That

James C. Hoe
Department of ECE
Carnegie Mellon University

Housekeeping

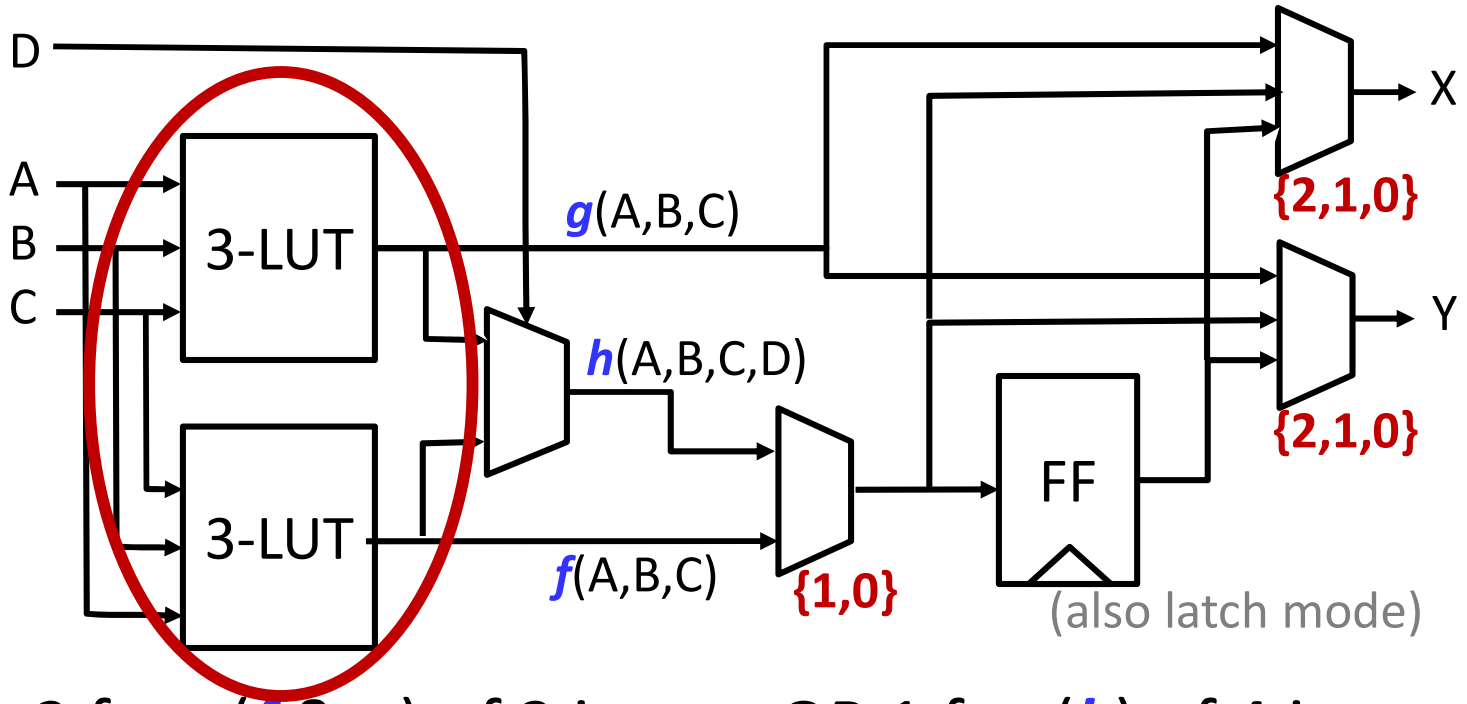
- Your goal today: understand reconfiguration behind-the-scene and the unique considerations when designing "soft logic"
- Notices
  - Handout #6: Lab 3, due noon, 10/30 (or 11/3)
  - Handout #7: Paper Review, sign-up due 10/27
  - Midterm in class, Wed 10/25
  - Project proposal due 10/30!!
- Readings (see lecture schedule online)
  - Kuon and Rose, "Measuring ...," ISFPGA, 2006.
  - Papamichael, et al., "CONNECT ...," ISFPGA, 2012.

Midterm Heads Up

- Covers lectures (L1~L13), labs, assigned readings
- Types of questions
  - freebies: remember the materials
  - **>> probing: understand the materials <<**
  - applied: apply the materials in original interpretation
- ****64 minutes, 56 points****
  - 8 short-answer format questions
  - 8 points per answer, best 7 counted
  - start of class on 10/25, in-person on-paper
  - closed book, individual effort

Configuration and Reconfiguration

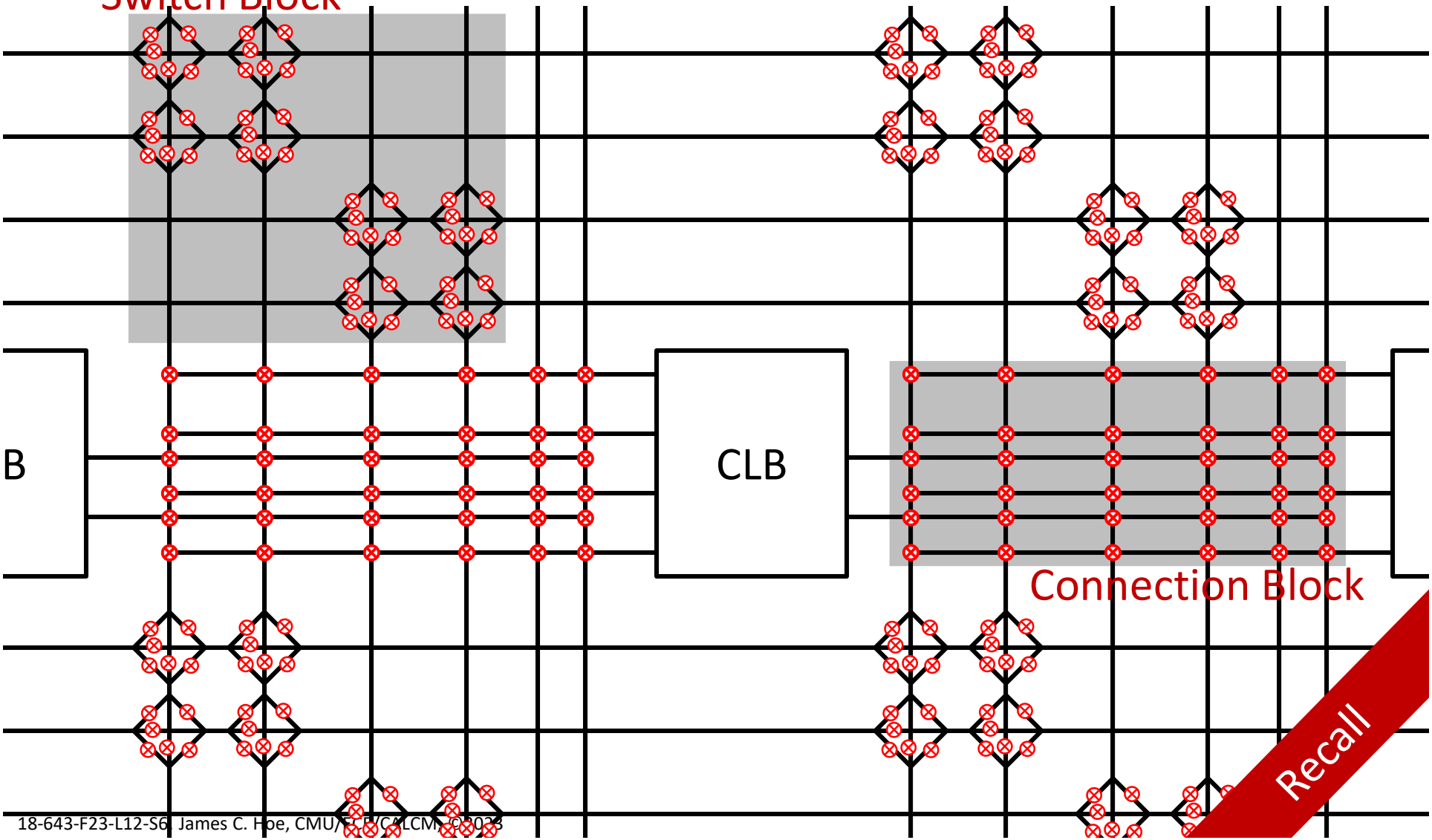
1980's Xilinx LUT-based Configurable Logic Block (in a sketch)



- 2 fxns (*f* & *g*) of 3 inputs OR 1 fxn (*h*) of 4 inputs
- hardwired FFs (too expensive/slow to fake)
- Just 10s of these in the earliest FPGAs

Recall

Configurable Routing (1980s Xilinx simplified)



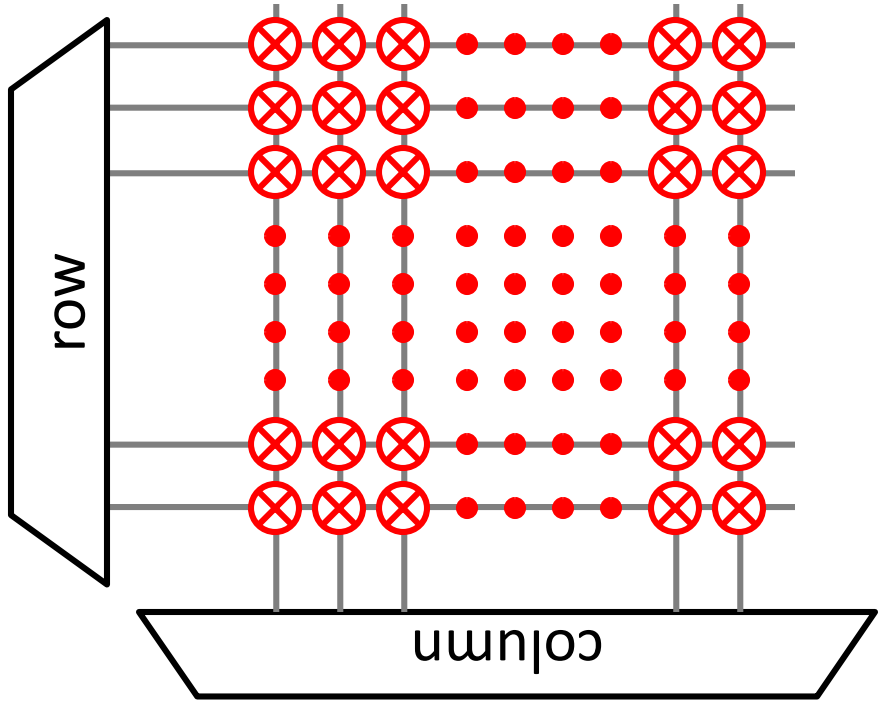
Recall

Bitstream defines the chip

- After power up, SRAM FPGA loads bitstream from somewhere before becoming the "chip"
  - many built-in loading options
  - non-trivial amount of time; must control reset timing and sequence with the rest of the system
  - forget what it does when powered-off
- Bitstream reverse-engineering ameliorated by
  - proprietary knowledge
  - encryption

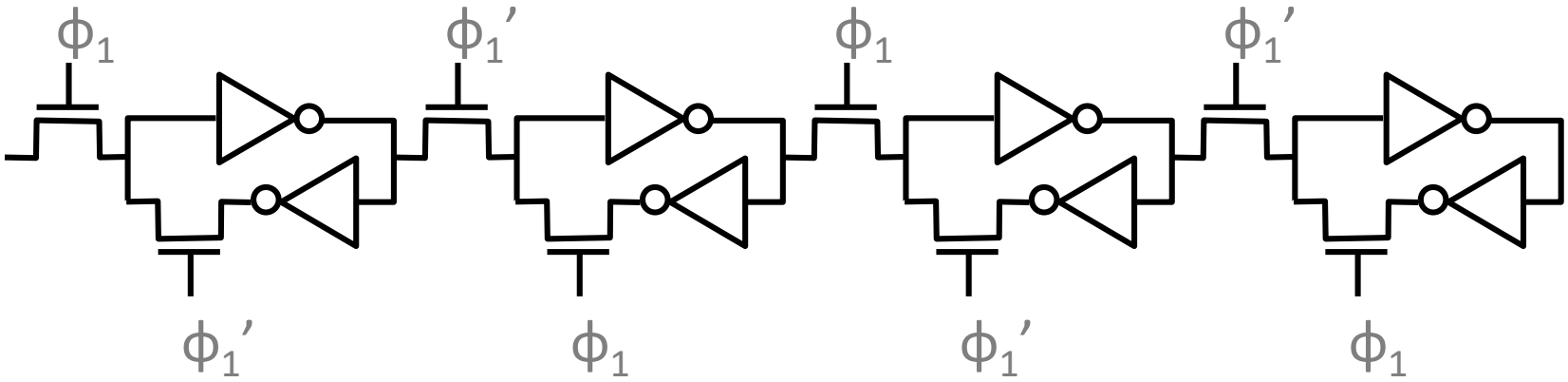
Setting Configuration Bits

- Behind-the-scene infrastructure, if used as ASIC
  - doesn't need to be fast (happens "offline")
  - simpler/cheaper the better
- Could organize bits into addressable SRAM or EPROM array
  - very basic technology
  - serial external interface to save on I/O pins



Serial Scan

- SRAM-based config. bits can be setup as one or many scan-chains on very slow config. clock
  - no addressing overhead
  - all minimum sized devices



- At power-up config manager handshake externally (various options, serial, parallel ROM, PCI-E, . . .)

Full-fledged config. "architecture" in modern devices to support scale and features

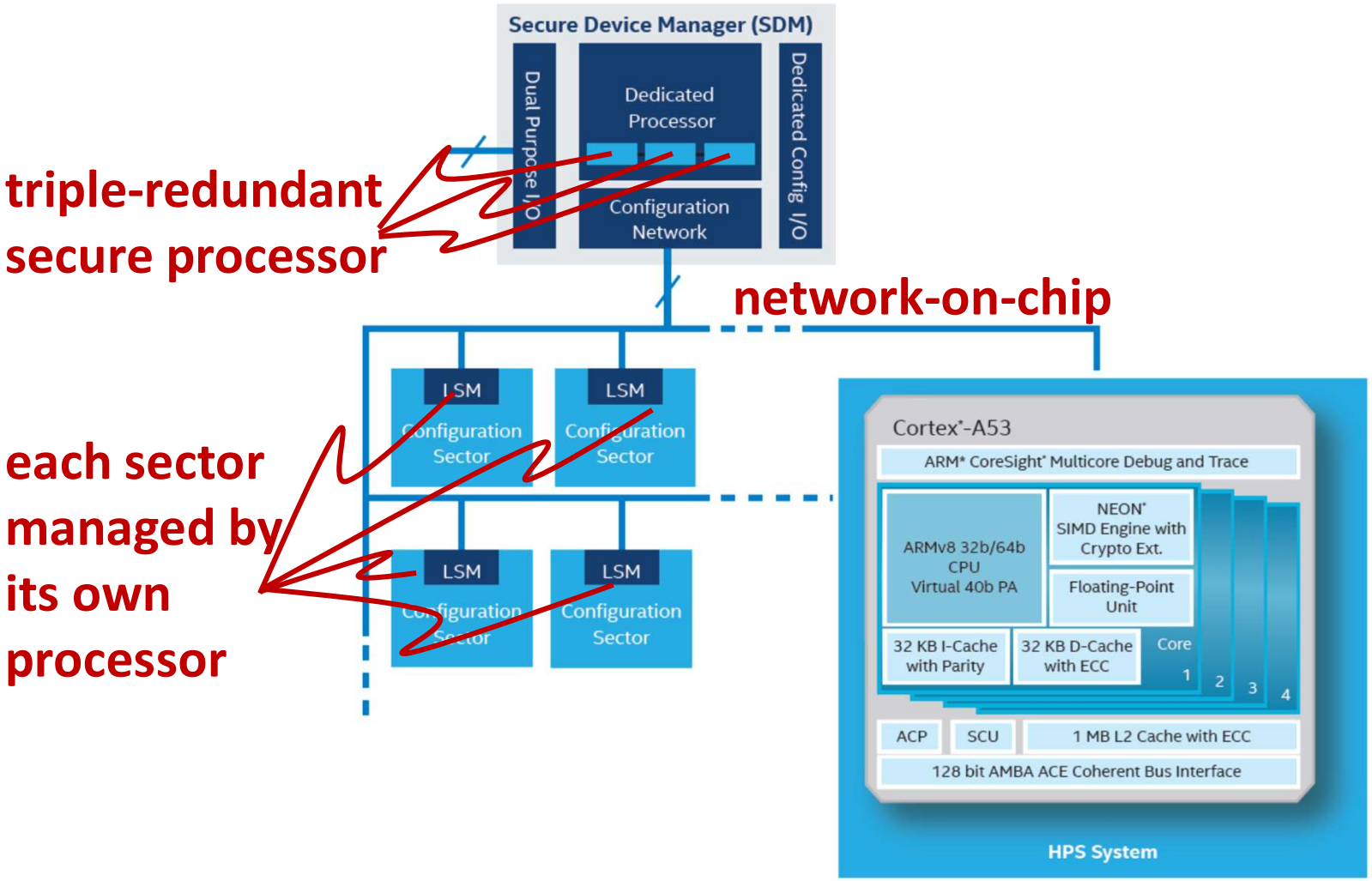
Modern Configuration Architecture
[Intel® Stratix® 10 Configuration User Guide]

Table 1. Intel® Stratix® 10 Configuration Scheme, Data Width, and MSEL

Configuration Scheme		Data Width (bits)	MSEL[2:0]
Passive	Avalon® -ST	32[1]	000
		16[1]	101
		8	110
	JTAG	1	111
	Configuration via Protocol (CvP)	x1, x2, x4, x8, x16 lanes	001 [2]
Active	AS - fast mode	4[1]	001
	AS - normal mode	4[1]	011

Modern Configuration Architecture

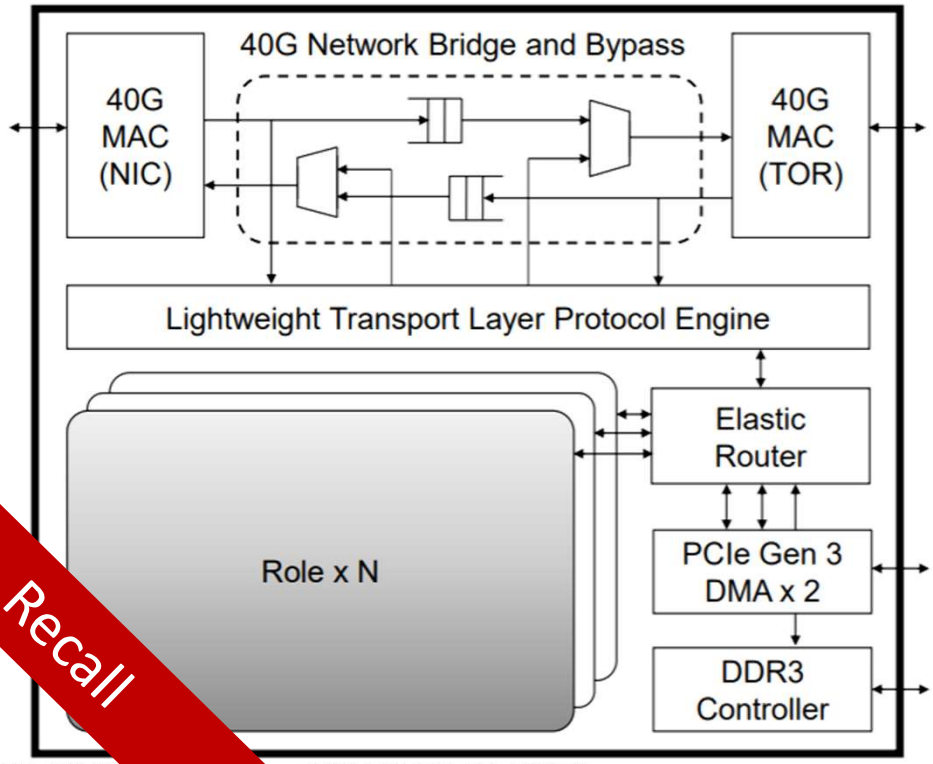
Stratix® 10 Secure Device Manager



[Figure 2: "Intel® Stratix® 10 Secure Device Manager Provides Best-in-Class FPGA and SoC Security"]

Role-and-Shell

- Fixed “shell”: base NIC fxn & infrastructure wrapper
- Reloadable “roles”: network acceleration, local and remote CPU offload, FPGA accelerator plane



Recall

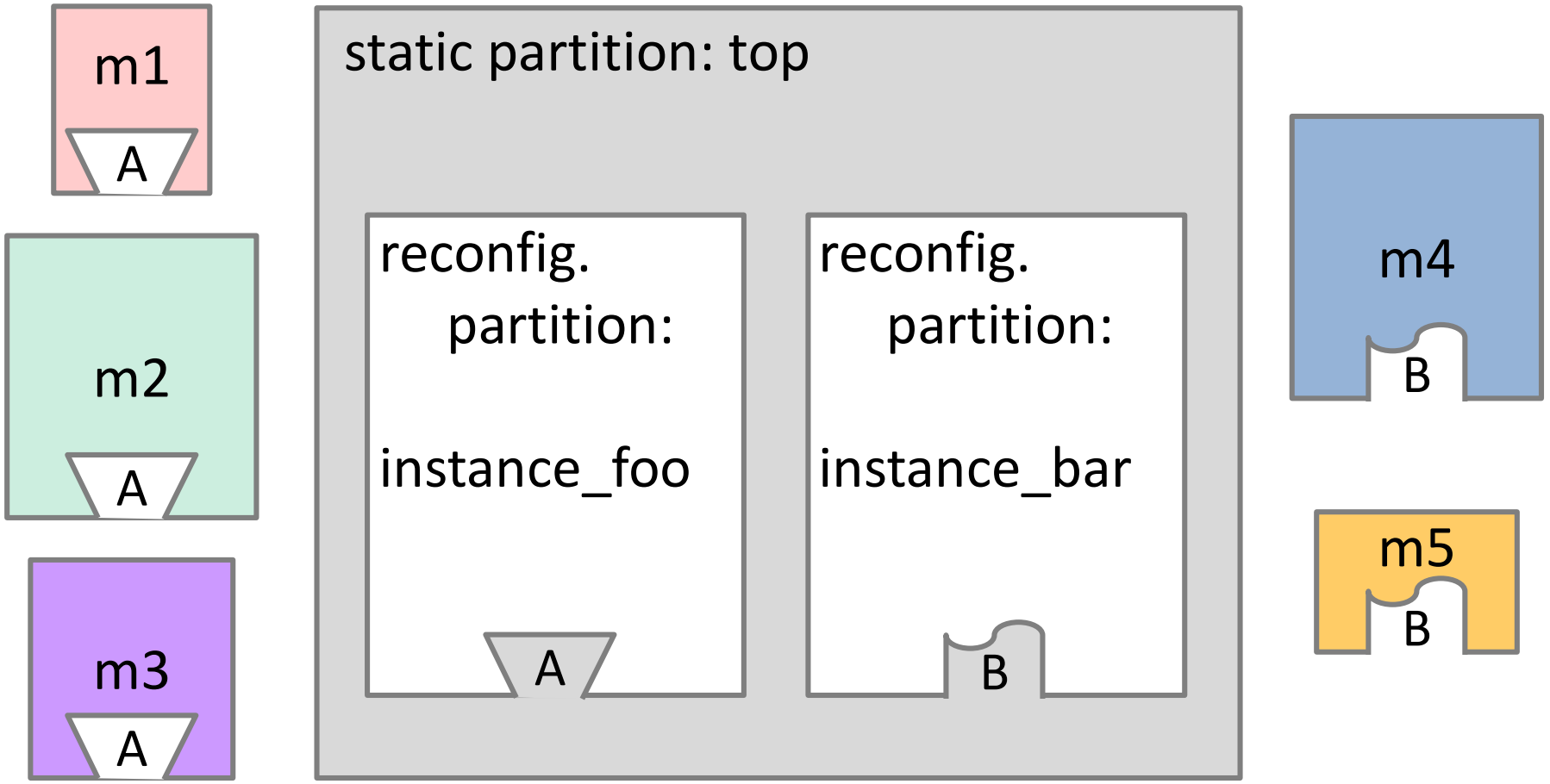
	ALMs	MHz
Role	55340 (32%)	175
40G MAC/PHY (TOR)	9785 (6%)	313
40G MAC/PHY (NIC)	13122 (8%)	313
Network Bridge / Bypass	4685 (3%)	313
DDR3 Memory Controller	13225 (8%)	200
Elastic Router	3449 (2%)	156
LTL Protocol Engine	11839 (7%)	156
LTL Packet Switch	4815 (3%)	-
PCIe DMA Engine	6817 (4%)	250
Other	8273 (5%)	-
Total Area Used	**131350 (76%)**	-
Total Area Available	172600	-

1st-gen Stratix V Catapult

Partial Reconfiguration (PR)

- Some regions of fabric retain their configured "personality" while other are reconfigured
  - e.g., keep the external bus interface from babbling while the functionality behind is changed
- The alive part can even control the reconfig.
  - e.g., load the bitstream through the bus
- Basic technology mature but usage not prevalent under the ASIC model
- Essential to FPGA as a flexible, sharable computing device

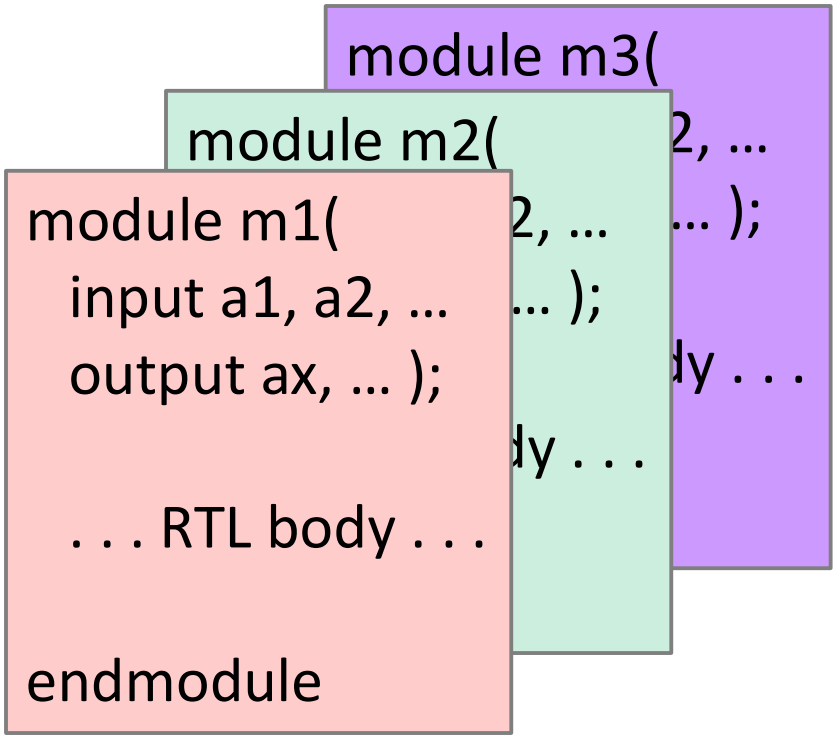
Static and Reconfigurable Partitions



Concrete Syntax (Xilinx's approach)

```
module top();
  . . . .
  foo instance_foo (a1, a2, …);
  bar instance_bar (b1, b2, …);
  . . . .
endmodule
```

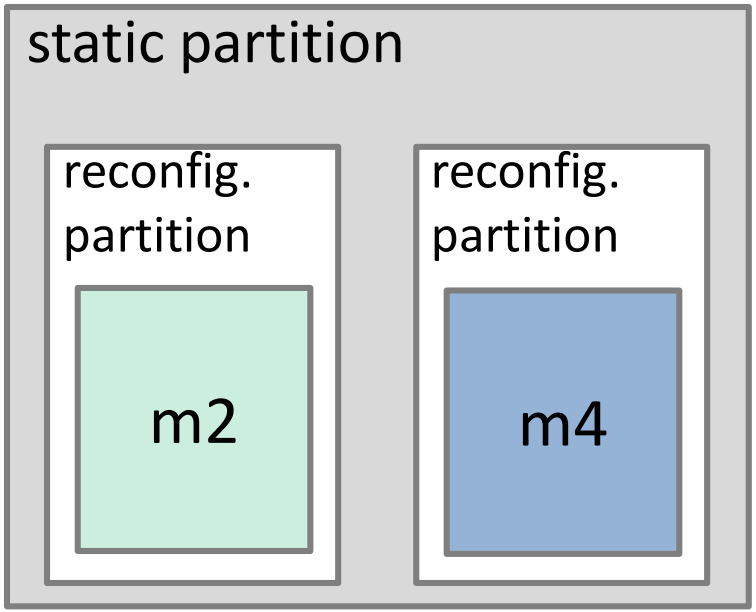
```
module foo(
  input a1, a2, …
  output ax, … );

  // nothing here

endmodule
```

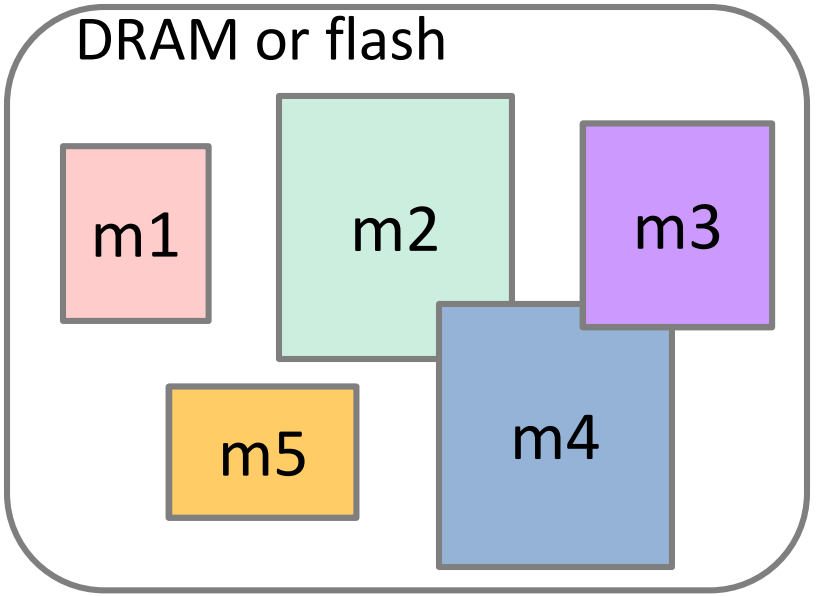


Logic region and interface locations for *foo* and *bar* set by floorplanning

At Run Time

- Power up with full-design bitstream
- Partial bitstreams in DRAM or flash memory
- Configuration API driven by ARM or fabric
  - reconfig. time depend on size, as low as msec
  - handshake signals to pause/start partition interface





Today's Practical Constraints

- Number and size of PR partitions fixed apriori
  - too few/too large: internal fragmentation
  - too many/too small: external fragmentation
- Not all PR partitions are equal—even if same interface and shape
  - a module needs a different bitstream for each partition it goes into
  - build and store upto MxN bitstreams for N partitions and M modules
- PR is not all that fast . . .

To be continued on Wednesday

Cost of Reconfigurability: Hard vs soft logic

[Kuon and Rose, 2006]

- Altera Stratix II FPGA, 90nm
  - Quartus II “balanced”, “standard fit”
  - hard multipliers and memory blocks
- ST Micro 90nm standard cells
  - Synopsys “high-effort”, add scan chain
  - ST Micro memory compiler
  - Cadence place and route
- Basic Results
  - avg 21x/40x in area (w/wo using hard macros)
  - 3~4x critical path
  - ~12x dynamic power

Benchmarking

Table 1: Benchmark Summary

Design	ALUTs	Total 9x9 Multipliers	Memory Bits
booth	68	0	0
rs_encoder	703	0	0
cordic18	2 105	0	0
cordic8	455	0	0
des_area	595	0	0
des_perf	2 604	0	0
fir_restruct	673	0	0
mac1	1 885	0	0
aes192	1 456	0	0
fir3	84	4	0
diffeq	192	24	0
diffeq2	288	24	0
molecular	8 965	128	0
rs_decoder1	706	13	0
rs_decoder2	946	9	0
atm	16 544	0	3 204
aes	809	0	32 768
aes_inv	943	0	34 176
ethernet	2 122	0	9 216
serialproc	680	0	2 880
fir24	1 235	50	96
pipe5proc	837	8	2 304
raytracer	16 346	171	54 758

- Opencores and local designs
  - removed cases where FPGA and ASIC are more than 5% different in FF count (Bias?)
- Metrics evaluated
  - logic density
  - circuit speed
  - power consumption

[Table 1: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]

Area Ratios

Table 2: Area Ratio (FPGA/ASIC)

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	33			
rs_encoder	36			
cordic18	26			
cordic8	29			
des_area	43			
des_perf	23			
fir_restruct	34			
mac1	50			
aes192	49			
fir3	45	20		
diffeq	44	13		
diffeq2	43	15		
molecular	55	45		
rs_decoder1	55	61		
rs_decoder2	48	43		
atm			93	
aes			27	
aes_inv			21	
ethernet			34	
serialproc			42	
fir24				9.8
pipe5proc				25
raytracer				36
Geomean	40	28	37	21

Differences attributed to "overhead" surrounding LUTs and FFs

[Table 2: Kuon and Rose, "Measuring the Gap between FPGAs and ASICs," 2006]

Critical Path Ratios

Table 3: Critical Path Delay Ratio (FPGA/ASIC Fastest Speed Grade

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	4.8			
rs_encoder	3.5			
cordic18	3.6			
cordic8	1.8			
des_area	1.8			
des_perf	2.8			
fir_restruct	3.5			
mac1	3.5			
aes192	4.0			
fir3	3.9	3.4		
diffeq	4.0	4.1		
diffeq2	3.9	4.0		
molecular	4.4	4.5		
rs_decoder1	2.2	2.7		
rs_decoder2	2.0	2.2		
atm			2.7	
aes			3.7	
aes_inv			4.0	
ethernet			1.6	
serialproc			1.0	
fir24				2.5
pipe5proc				2.5
raytracer				1.4
Geomean	3.2	3.4	2.3	2.1

Table 4: Critical Path Delay Ratio (FPGA/ASIC) Slowest Speed Grade

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	6.6			
rs_encoder	4.7			
cordic18	4.9			
cordic8	2.5			
des_area	2.6			
des_perf	3.8			
fir_restruct	5.0			
mac1	4.6			
aes192	5.4			
fir3	5.4	4.6		
diffeq	5.4	5.5		
diffeq2	5.2	5.4		
molecular	6.0	6.1		
rs_decoder1	3.0	3.6		
rs_decoder2	2.7	3.0		
atm			3.6	
aes			4.9	
aes_inv			5.5	
ethernet			2.2	
serialproc			1.4	
fir24				3.3
pipe5proc				3.5
raytracer				2.0
Geomean	4.3	4.5	3.1	2.8

[Table 3&4: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]

Dynamic Power Ratios

Table 5: Dynamic Power Consumption Ratio (FPGA/ASIC)

Name	Method	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	Sim	16			
rs_encoder	Sim	7.2			
cordic18	Const	6.3			
cordic8	Const	6.0			
des_area	Const	26			
des_perf	Const	9.3			
fir_restruct	Const	9.0			
mac1	Const	18			
aes192	Sim	12			
fir3	Const	12	7.4		
diffeq	Const	15	12		
diffeq2	Const	16	12		
molecular	Const	15	15		
rs_decoder1	Const	13	16		
rs_decoder2	Const	11	11		
atm	Const			11	
aes	Sim			4.0	
aes_inv	Sim			3.9	
ethernet	Const			15	
serialproc	Const			24	
fir24	Const				5.2
pipe5proc	Const				12
raytracer	Const				12
Geomean		12	12	9.2	9.0

[Table 5: Kuon and Rose, “Measuring the Gap between FPGAs and ASICs,” 2006]

Actual Mileage Varies

- Comparisons strongly affected by
  - exact design, FPGA/ASIC target, methodology, availability/use of macro's
  - comparing less than "best-effort" designs can bias in either direction—*same RTL not best for both*
  - design is not a point---a full comparison would have to be Pareto-front to Pareto-front
- Either
  - precise in a specific context, or
  - warm-fuzzy rule of thumb (2x<<"~10x"<<100x)

  A moving target with arch and process changes

Design Soft Logic Differently

RTL for FPGA not scaled version of ASIC RTL

1. Different relative cost in logic vs. wires vs. mem
2. Different relative speed in logic vs. wires vs. mem
3. Soft Logic design can change . . .

Design differently for FPGA and use FPGA differently!!

FPGA Logic Peculiarities

imagine an ASIC RTL designer with no FPGA training

- Logic slower than expected (can put much less between clock edges)
- CLB-mapped logic not divisible for pipelining
  - over-pipeline adds cycles without freq. increase
  - "sweetspot" frequency that is easy to reach but hard to exceed
- Sharp aberrations around hard macro use

  e.g., multiply faster than add in Virtex-II
- Design for performance
  - correct and maximal usage of hard macros
  - shallowly pipelined, wide datapath

FPGA Wire Peculiarities

imagine an ASIC RTL designer with no FPGA training

- Wire delay significant—absolute and relative—even short wires
- High true area cost but low apparent cost
  - routing over-provisioned to handle worst case
  - in a "typical" design, wires appear cheaper relative to other resource types

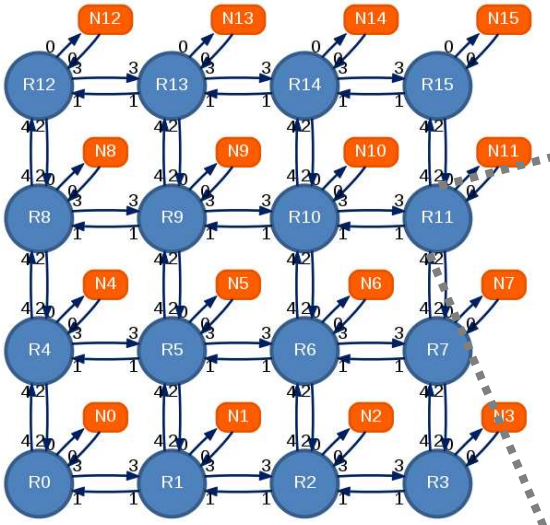
    best case is nearest-neighbor, regular grid
- Counterintuitively, you SHOULD use wider busses
  - consume unused "free" wires
  - compensate for lower frequency

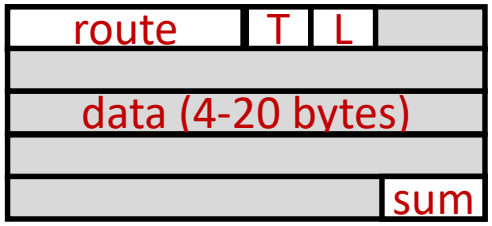
imagine an ASIC RTL designer with no FPGA training

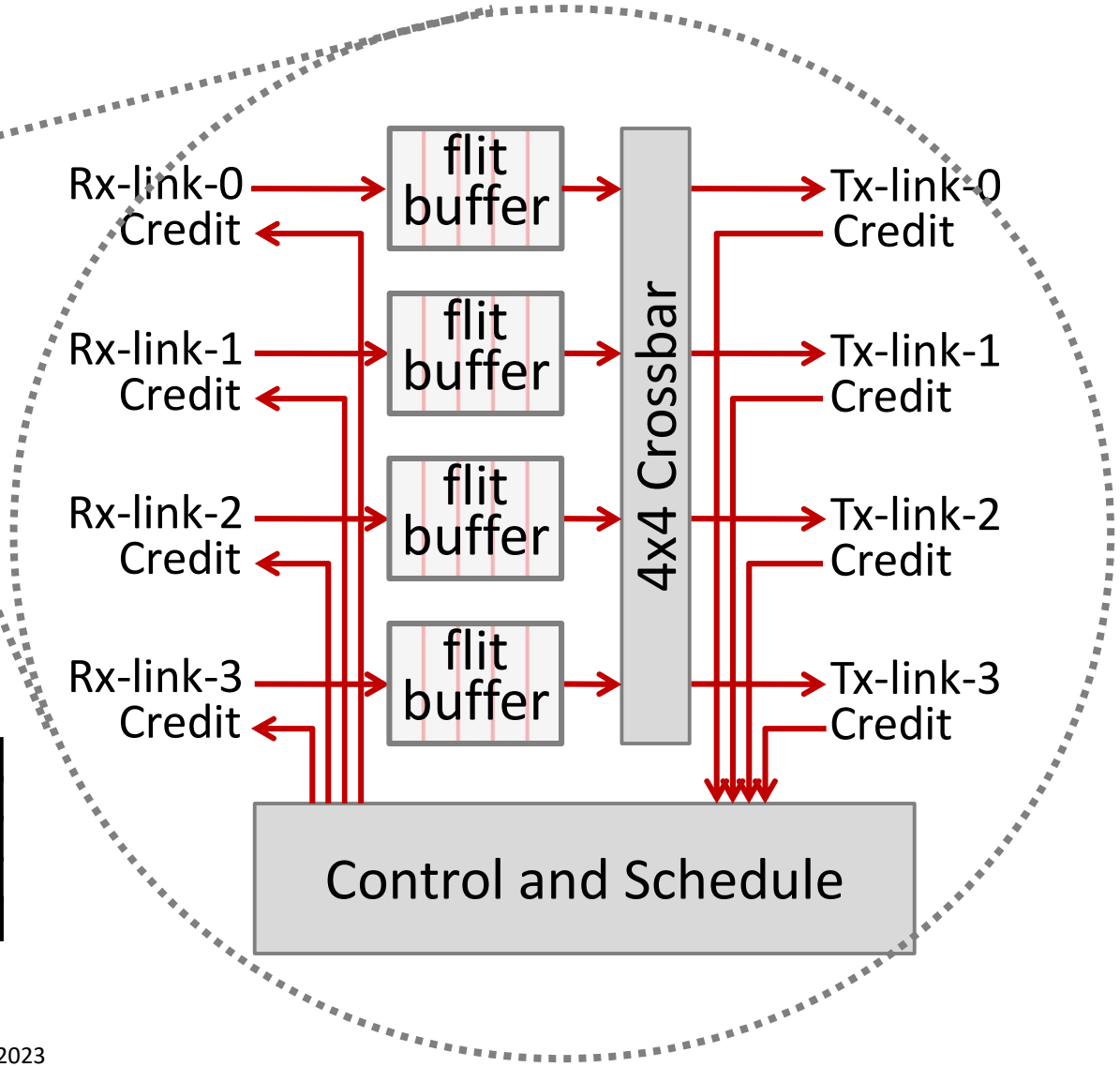
FPGA Memory Peculiarities

- Large memory (BRAM) abnormally fast
- Large memory are “free” until your run-out
- Quantized memory options
  - jumps between FF-based vs. LUT-RAM vs. BRAMs
  - optimal choice depends on many factors: size, aspect ratio, contention with other IPs
- Must manage RAM usage
  - don’t waste BRAM on small buffers
  - tune buffer sizes to natural granularities, e.g., zero incremental cost to go from 2Kb to 4Kb
  - pack buffers to share same physical array

Soft NoC Case Study [Papamichael, 2012]

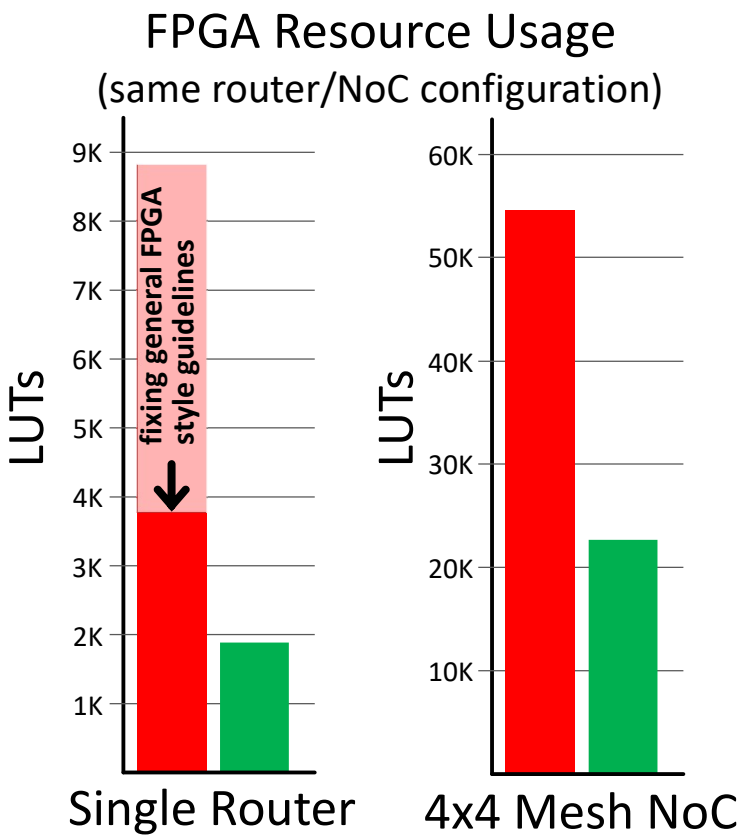


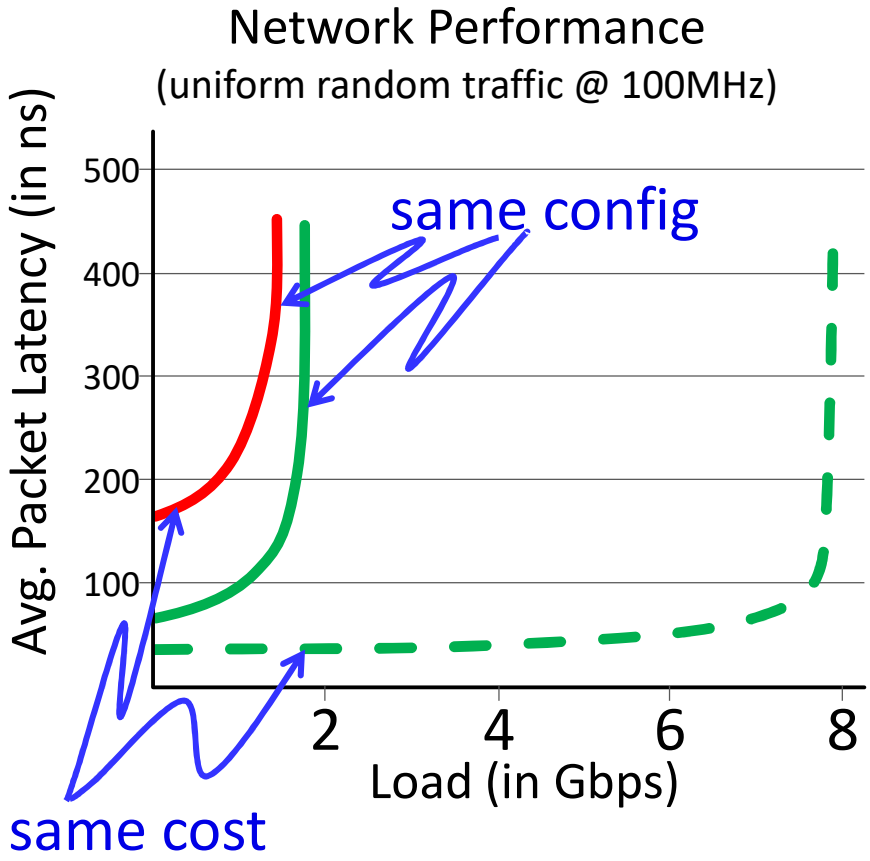




FPGA- vs ASIC-tuned RTL on FPGA

- **ASIC RTL** from nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router
- **FPGA RTL** from www.ece.cmu.edu/calcm/connect/

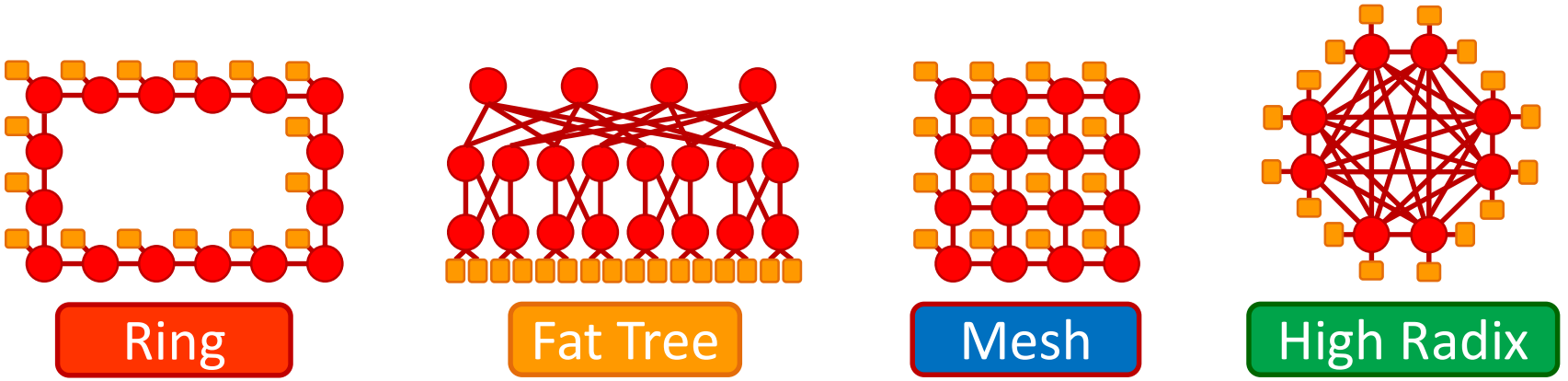


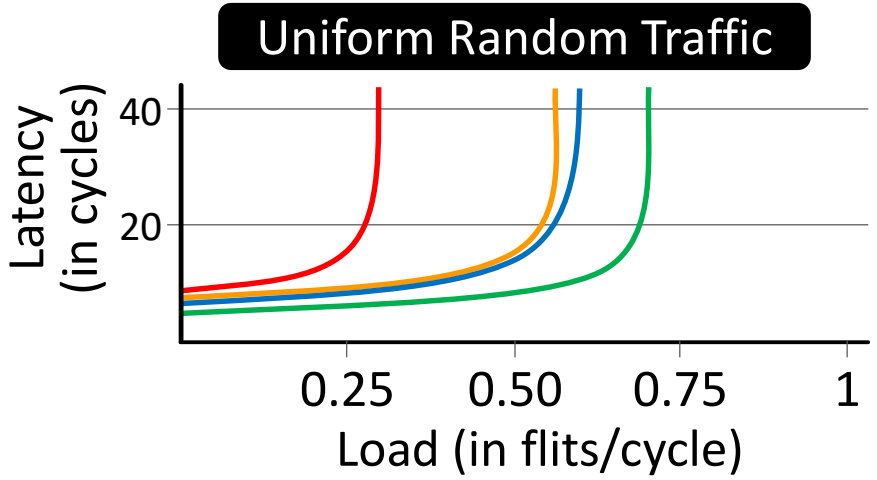


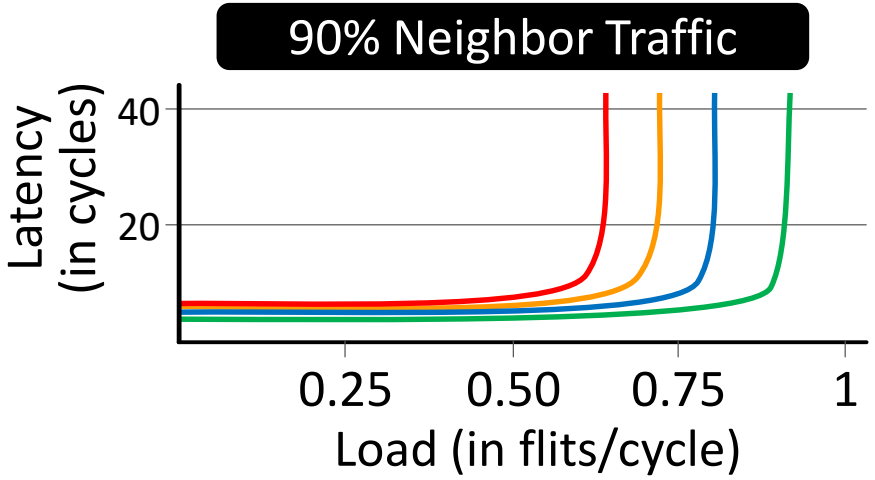
[Papamichael, ISFPGA 2012]

Soft-IPs need not be general purpose

- Reconfigurable fabric provides generality
- Soft-IPs should be maximally specialized to usage





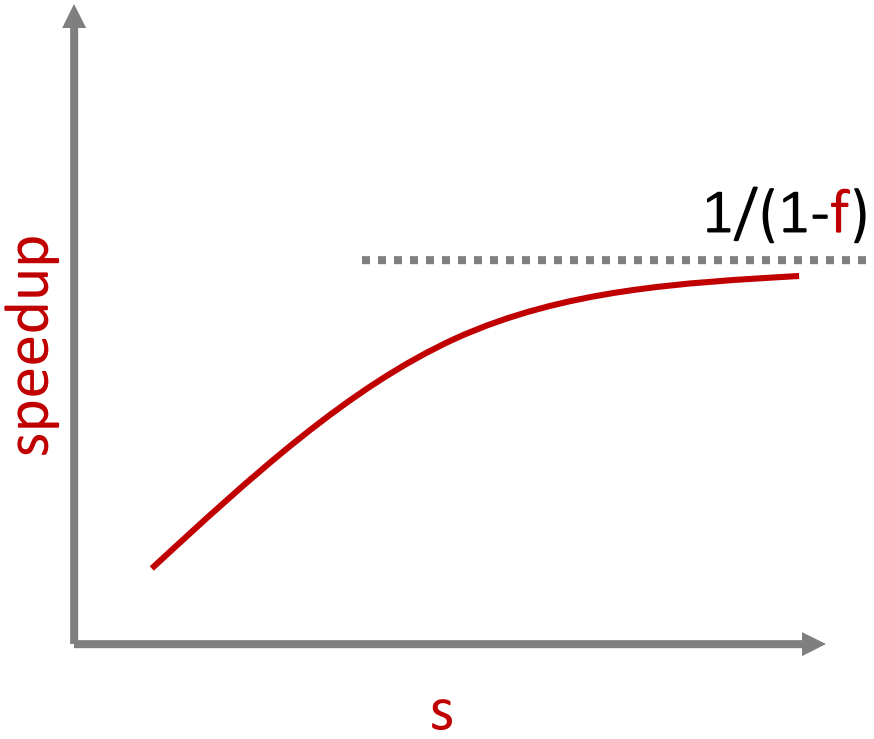


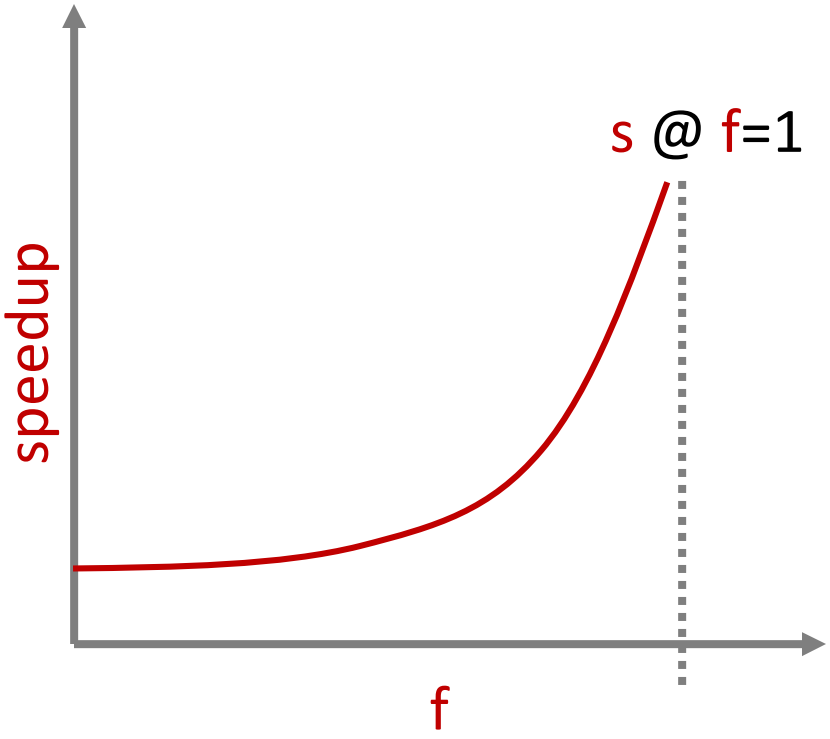
Stop Thinking "Field Programmable"

- "Field Programable" is when we wanted FPGA to be ASIC
  - programmability avoided manufacturing NRE
  - programmability reduces design time/cost (incremental development; at speed testing; field updates, etc.)
  - BUT once programmed at power-on, FPGA is fixed

- Let's use programmability to be more than ASIC
  - repurpose fabric over time, at large and small time scales
  - share fabric by multiple applications concurrently

Amdahl's Law

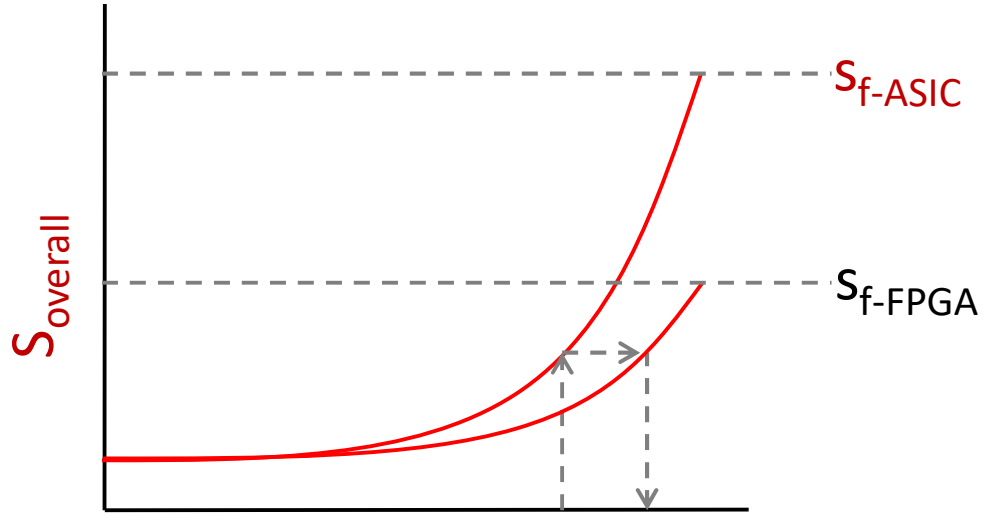




$$\text{speedup} = 1 \,/\, ((1-f) + f/s)$$

Turn Programmability into Performance

- Amdahl's Law: $S_{overall} = 1 / ((1-f) + f/s_f)$
- $s_{f\text{-}ASIC} > s_{f\text{-}FPGA}$ but $f_{ASIC} \neq f_{FPGA}$
- $f_{FPGA} > f_{ASIC}$ (when not perfectly app-specific)
  - more flexible design to cover a greater fraction
  - reprogram FPGA to cover different applications



[based on Joel Emer's original comment about programmable accelerators in general]

Parting Thoughts

- FPGAs pay an overhead for reconfigurability
  - significant but reducing
  - power and BW bottleneck can compress differences
- FPGAs differ from ASICs in more than then reconfiguration overhead---require distinct architecture and tuning
- FPGA is more than ASIC (more on this next lecture)