18-643 Lecture 13: The Rangos Ballroom

Shashank Obla

PhD Student, Department of ECE

Carnegie Mellon University

You are designing an Event Hall

What is the first thing you do?

You are designing an Event Hall

Specification

Host lectures/orientation events with audience seating

Hall Design

E.g., McConomy Auditorium

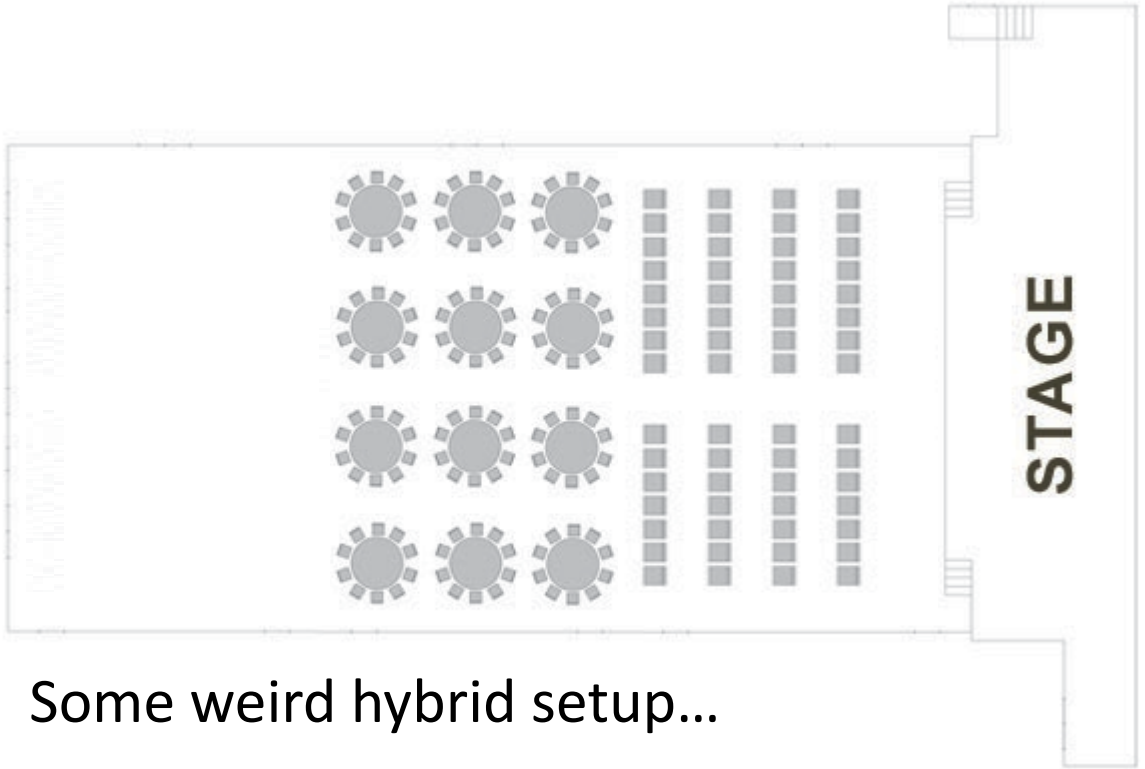
You are designing an Event Hall

Specification

- Host lectures/orientation events with seating
- Conduct Banquets
- Poster Sessions

And switch between different events quickly

Hall Design



Some weird hybrid setup…

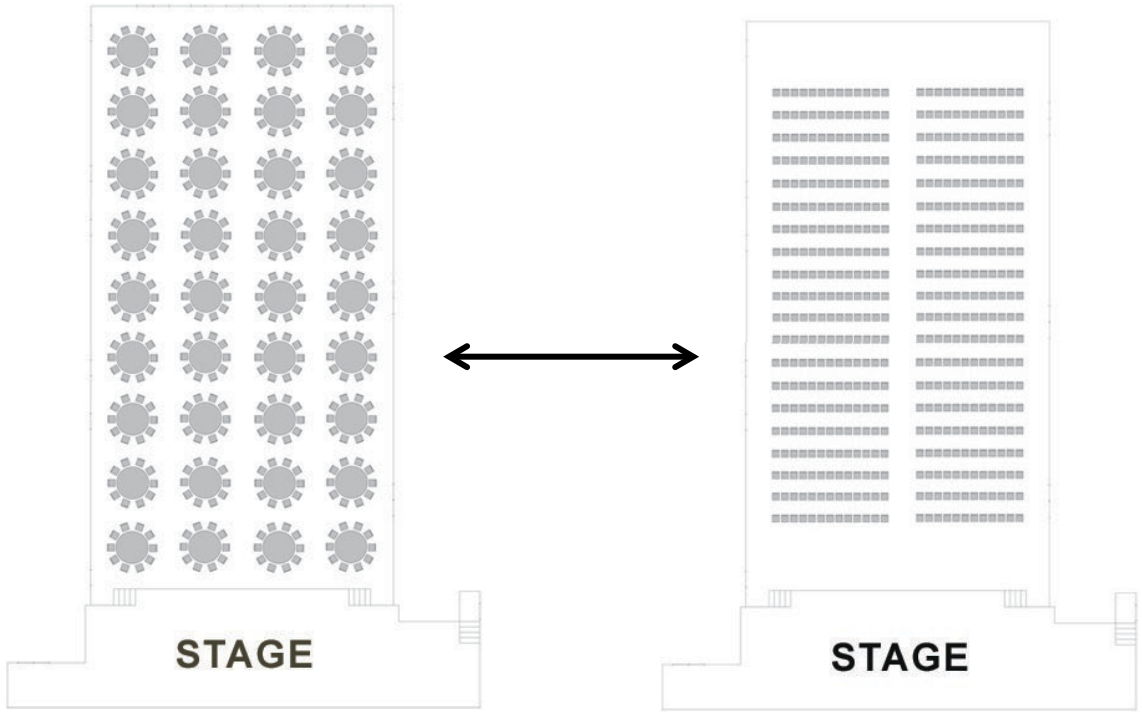
You are designing an Event Hall

Specification

- Host lectures/orientation events with seating
- Conduct Banquets
- Poster Sessions

Can take time to switch between but want better arrangement…

Hall Design



A reconfigurable room, e.g., Rangos Ballroom!

All pictures of Rangos from : https://www.cmu.edu/cohon-university-center/center-facilities/Rangos/index.html

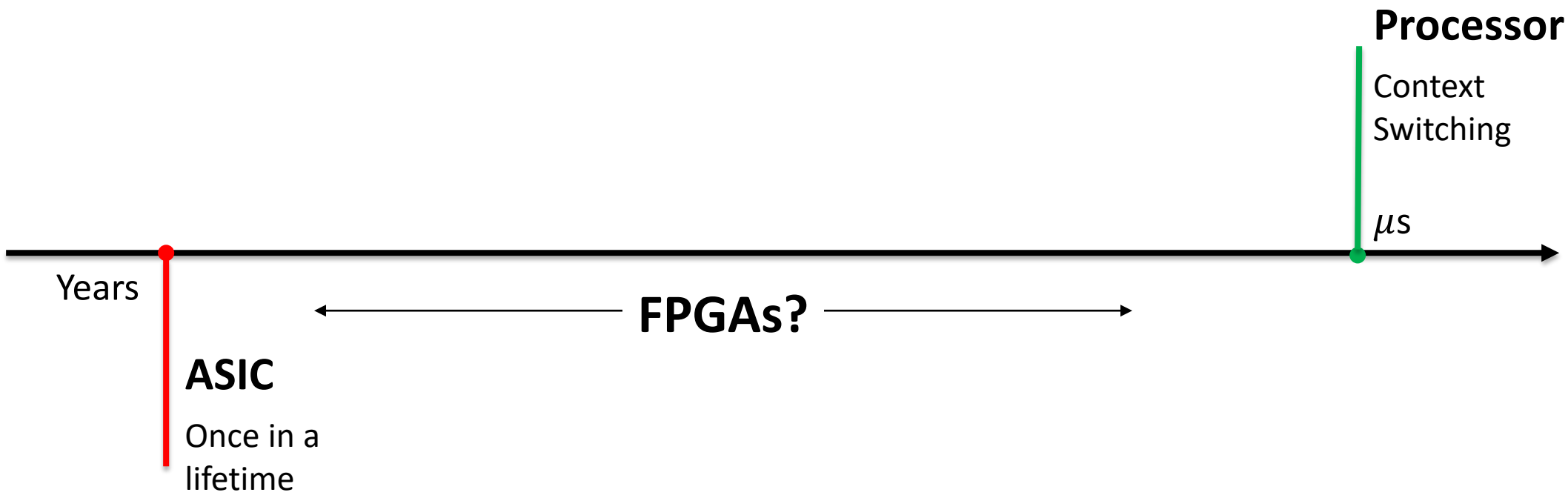
18-643 Lecture 13: FPGA Programmability

Shashank Obla
PhD Student, Department of ECE
Carnegie Mellon University

Housekeeping

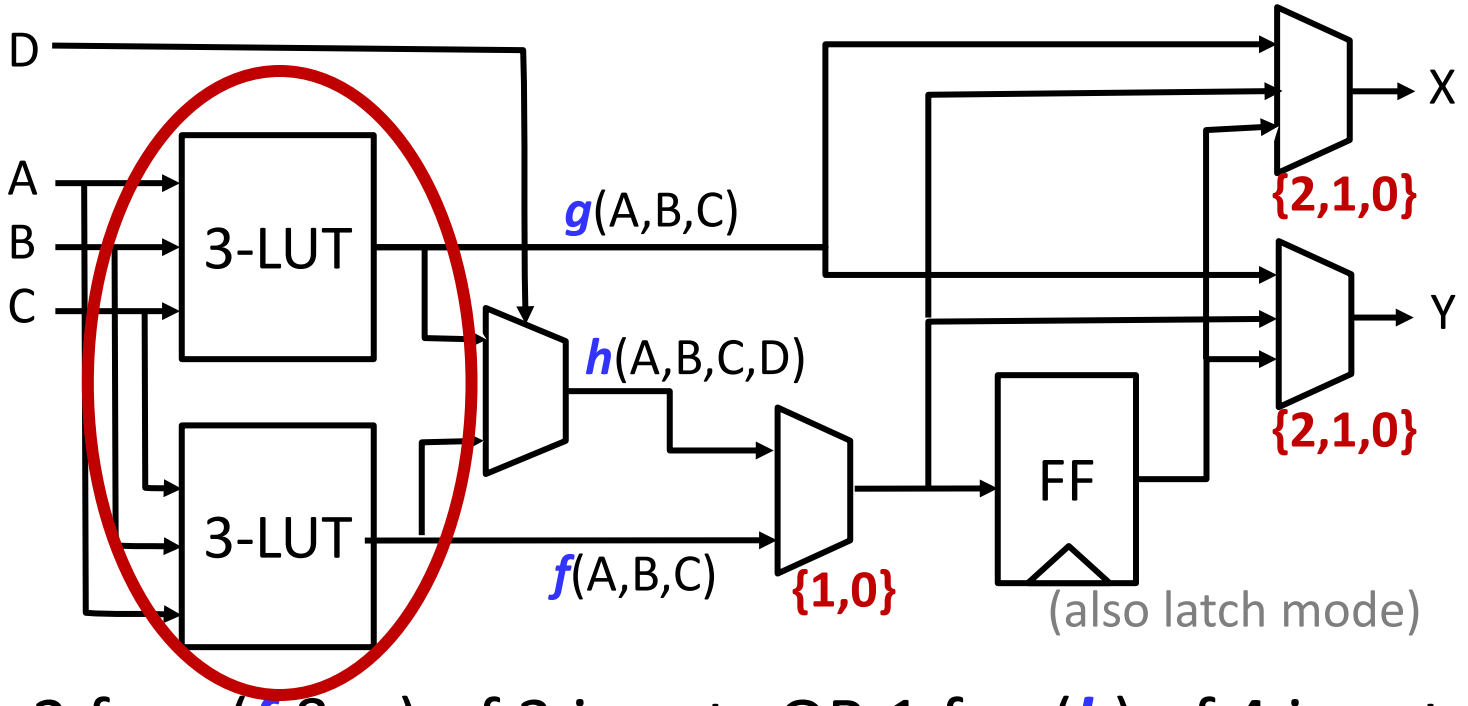
- Your goal today: appreciate FPGAs as truly programmable and dynamic devices
- Notices
  - Handout #6: lab 3, due noon, 10/31
  - Project Proposal also due on 10/31!
  - Midterm in class on Wed 10/26
  - Recitation tomorrow on the finer details
  - Ask questions and be interactive!
- Readings (see lecture schedule online)
  - For a textbook treatment: Ch 4, Reconfigurable Computing

The Programmability Timescale



Note: For this lecture, programmability refers to "radically" changing the function of the device

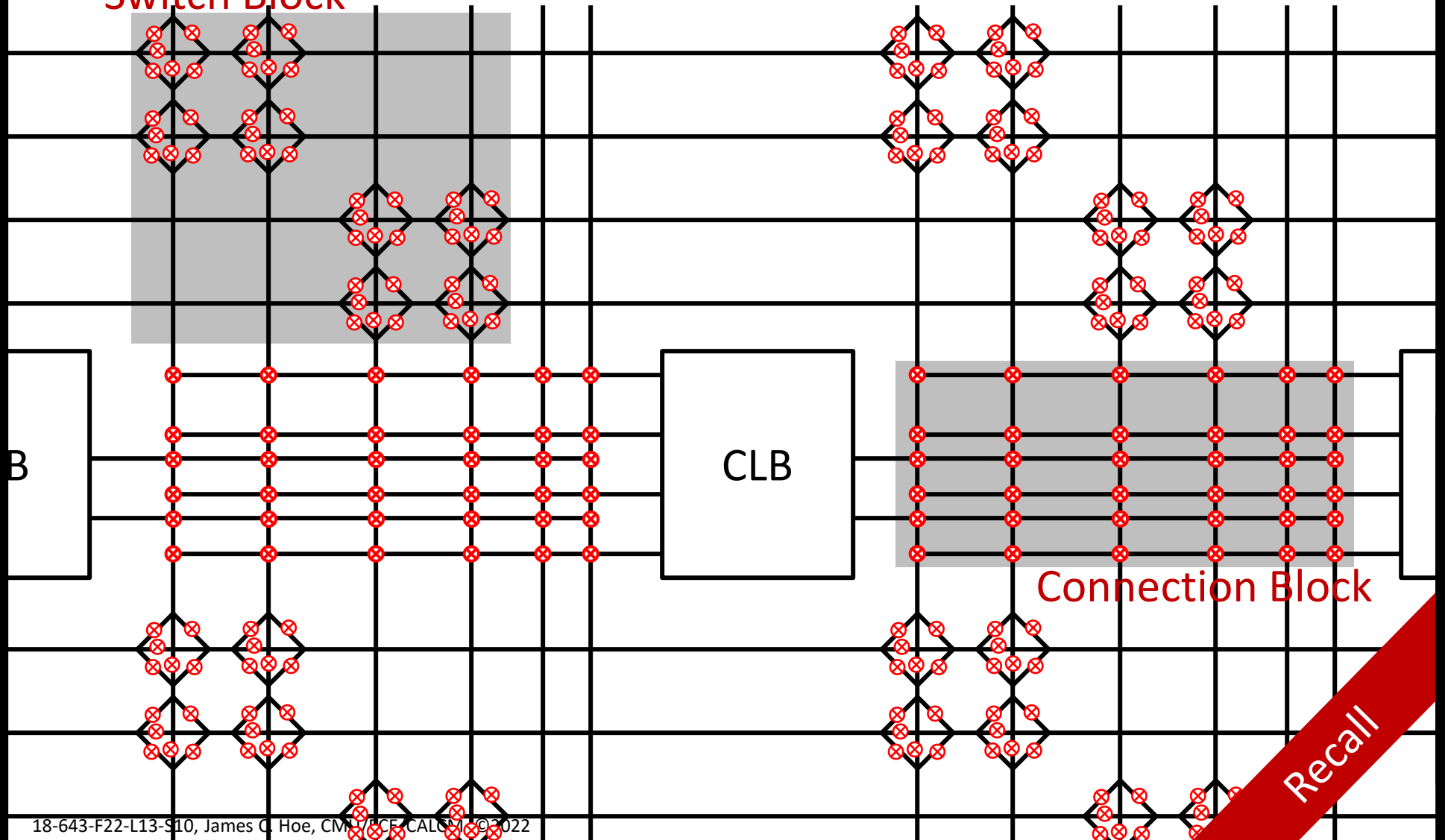
1980's Xilinx LUT-based Configurable Logic Block (in a sketch)



- 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)

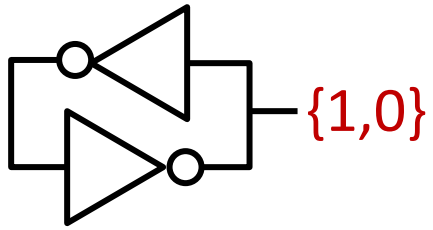


How about no mask, no fab? i.e., "field programmable"

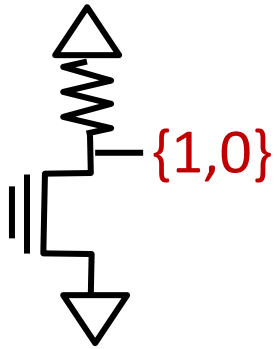
- Again, mass produce identical devices but this time fully-finalized
- Then what can be changed?

bits

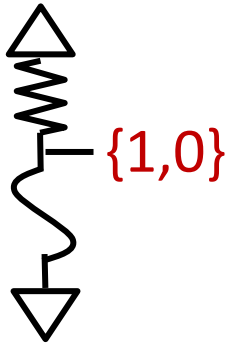
– SRAM



EPROM

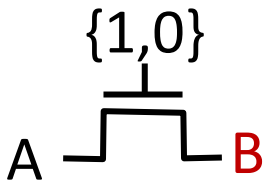


(anti)fuse

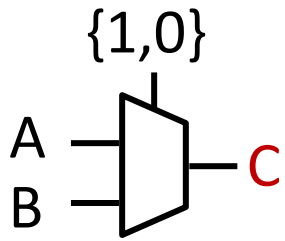


connections

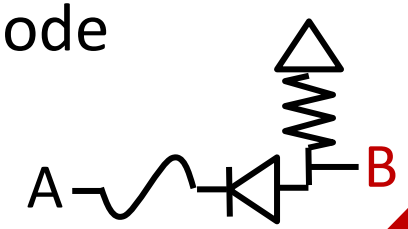
– pass gate



mux



diode



Recall

programmable vs reprogrammable

Bitstream defines the chip

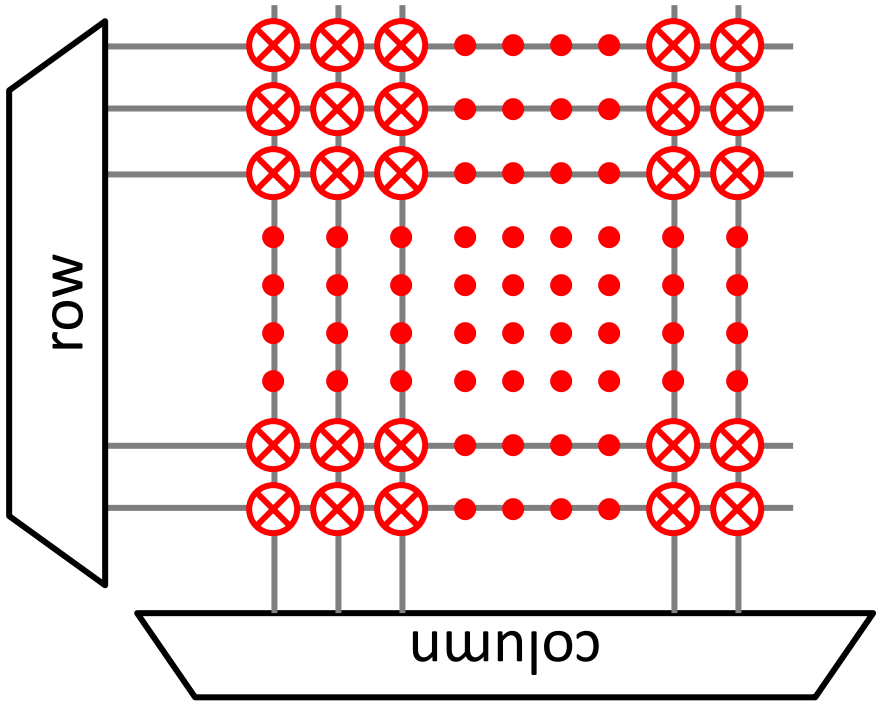
- After power up, SRAM FPGA loads bitstream from somewhere before becoming the "chip"

  a bonus "feature" for sensitive devices that need to forget what it does

- Many built-in loading options
- Non-trivial amount of time; must control reset timing and sequence with the rest of the system
- Reverse-engineering concerns ameliorated by
  - encryption
  - proprietary knowledge

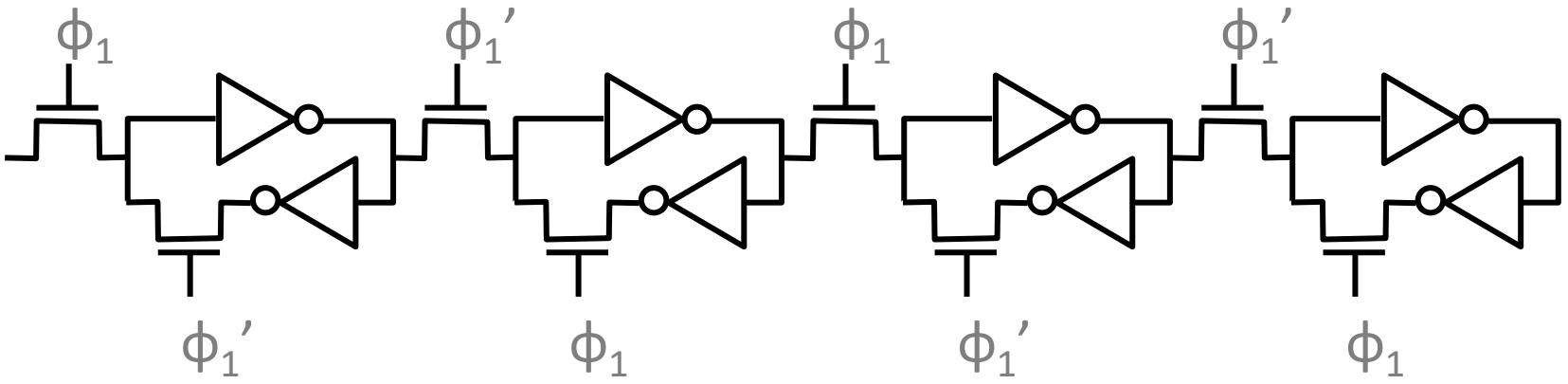
Setting Configuration Bits

- Behind-the-scene infrastructure
  - doesn't need to be fast (usually offline)
  - simpler/cheaper the better (at least used to be)
- 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

Reconfiguration is a big ordeal

Event Planning

Can take time to switch between but want good arrangement…

Might be okay… Seems like a big process, can't do much even if it was open.

Hall Design

But we made it more efficient…

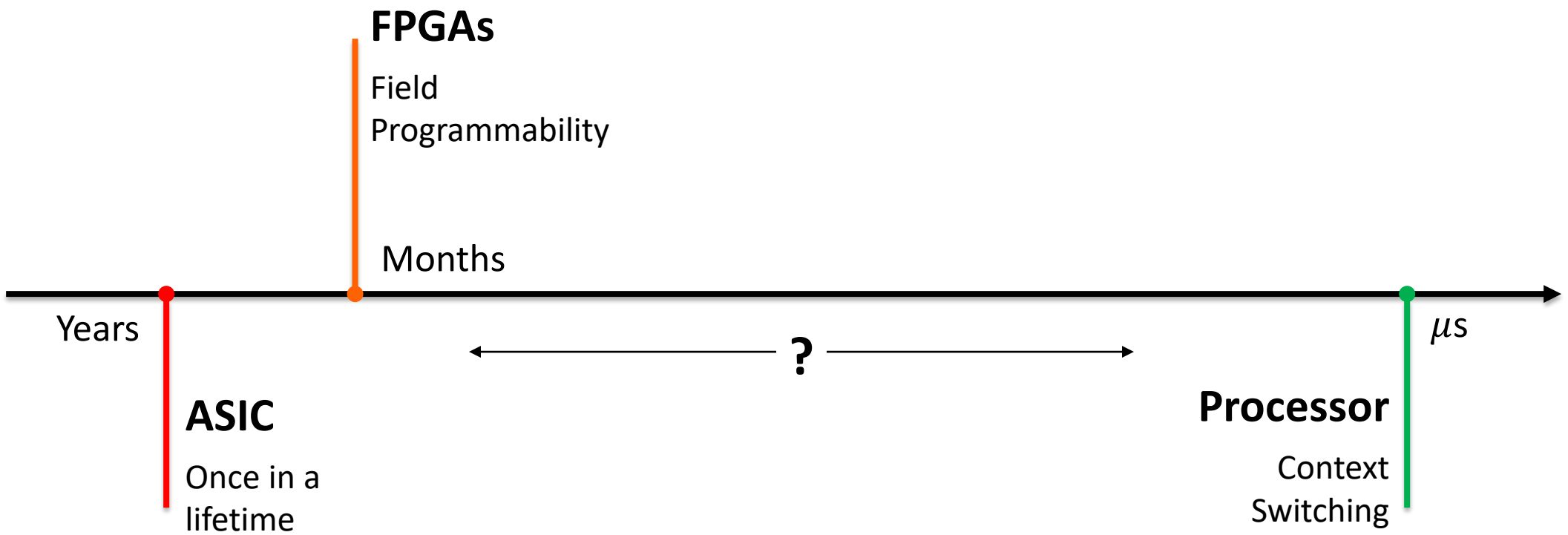
Event Planning

Umm… I could have multiple events in a day, downtime is too much

- Could bring in food for the event
- Set up the stage if required
- Scout the room beforehand
- Make sure rules are followed

Hall Design

Back to the Programmability Timescale



Or as Xilinx brands it, Dynamic Function eXchange (DFX)

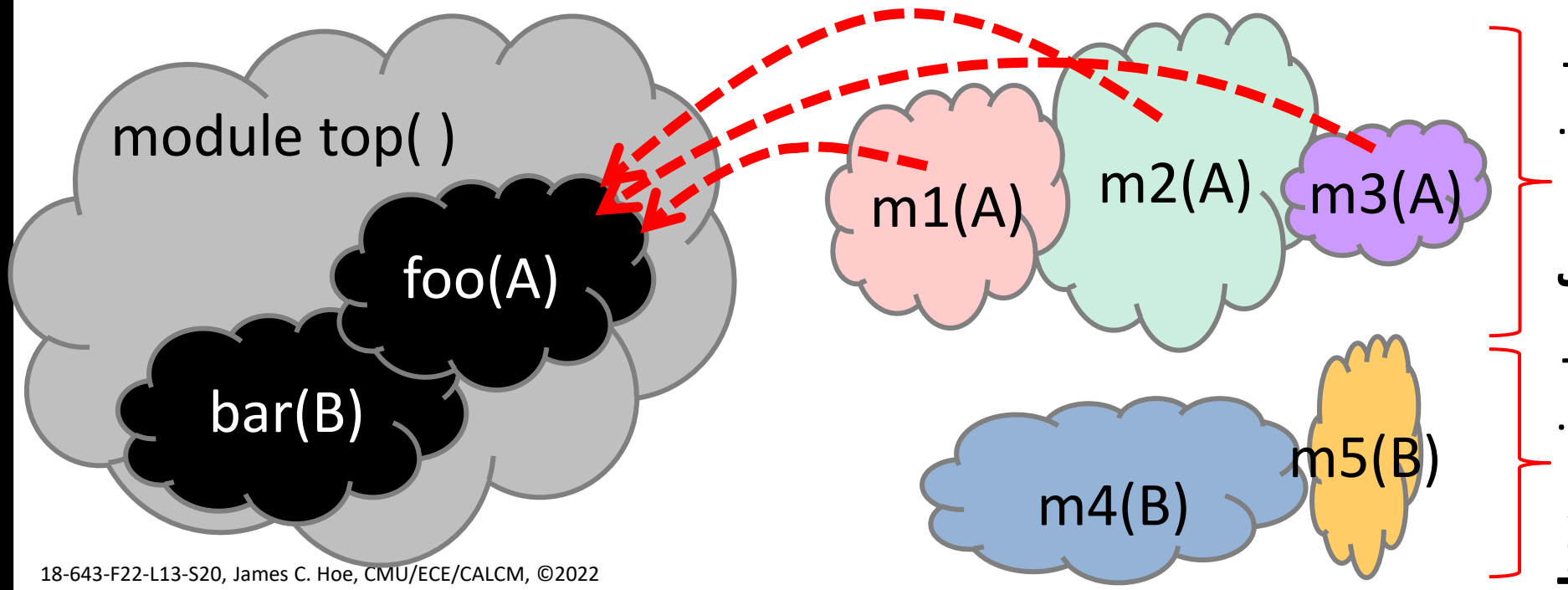
DYNAMIC PARTIAL RECONFIGURATION

Partial Reconfiguration (PR)

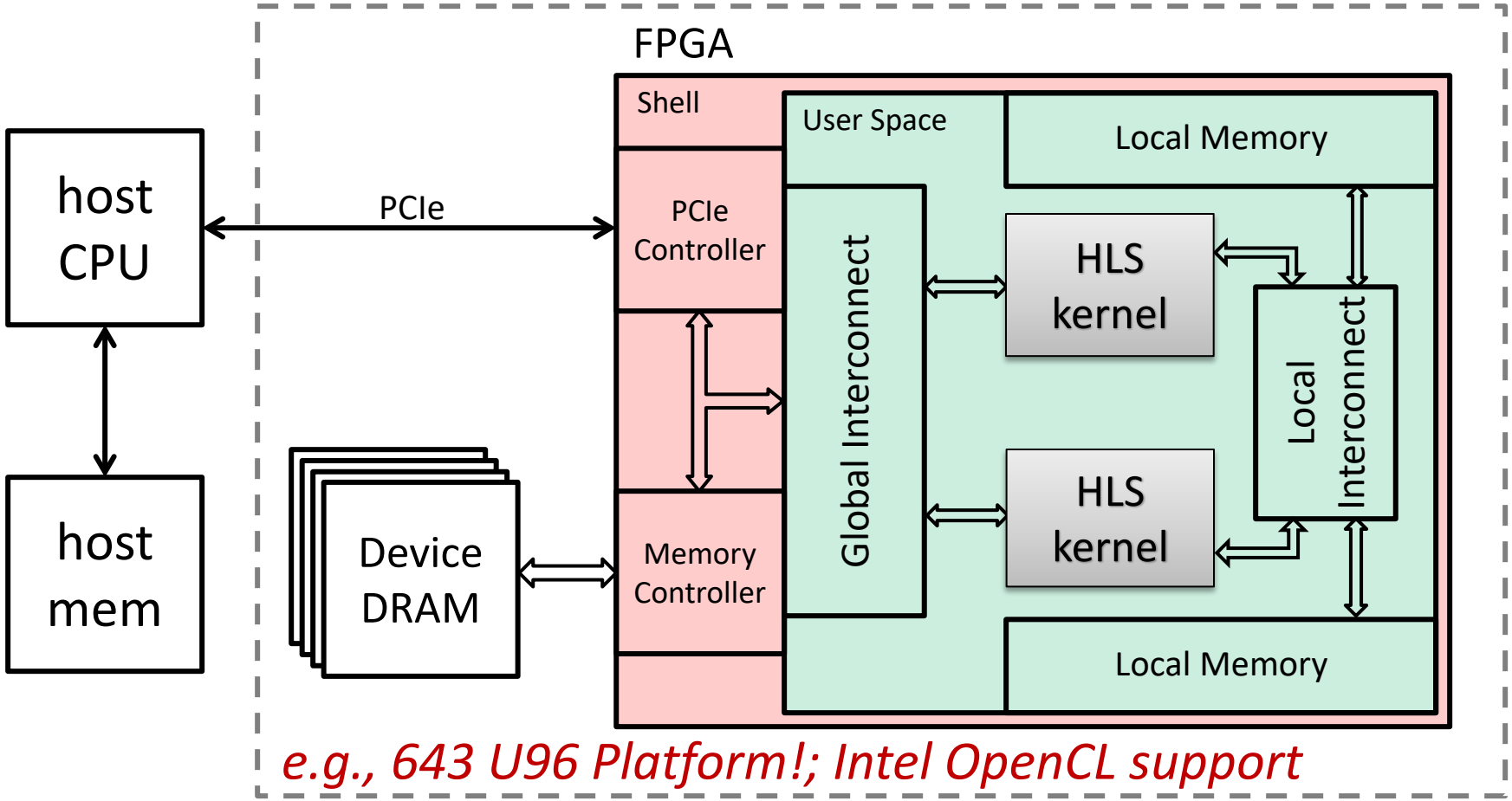
- Shrinks the granularity of "field programmability" to within the fabric
- Some parts of fabric retain their configured "personality" while other parts are reconfigured
  - e.g., keep the external bus interface from babbling while the functionality behind is changed
- The alive part can even control the reconfig.
  - e.g., load the bitstream through the bus
- Implemented with the ability to mask which configuration bits are written to at runtime

PR Conceptually

- Module **top**() instantiate submodules **foo**(**A**) and **bar**(**B**) with interface **A** and **B** respectively
  - **foo**(**A**) and **bar**(**B**) are "blackboxes", i.e., interface only, no internals
  - **m1**()~**m5**() have matching interfaces, **A** or **B**

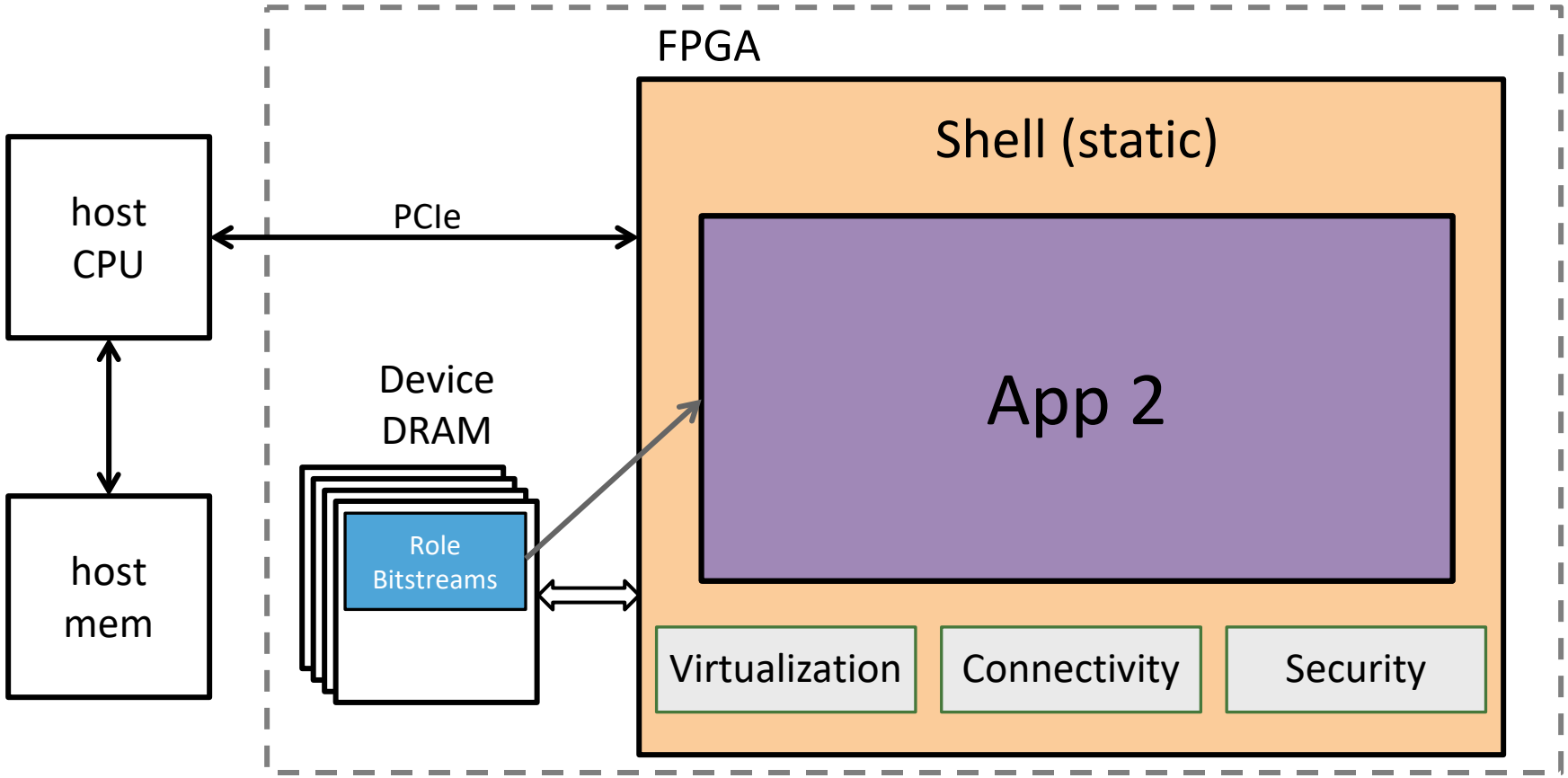


Role-and-Shell PR Usage



- virtualize and simplify surrounding
- enforce security and QoS
- keep system alive

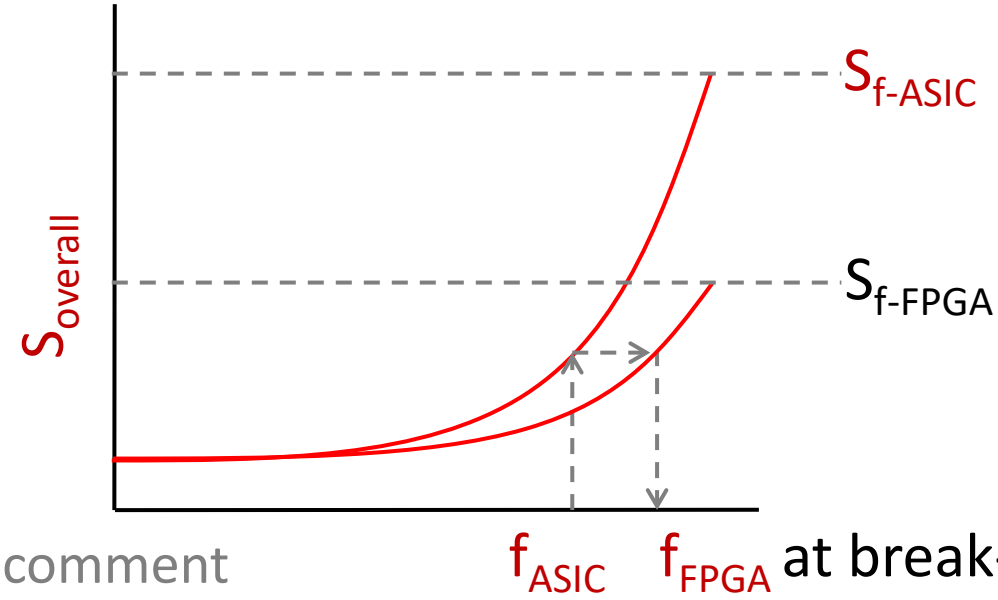
Time-Sharing with PR



FPGA as an accelerator!

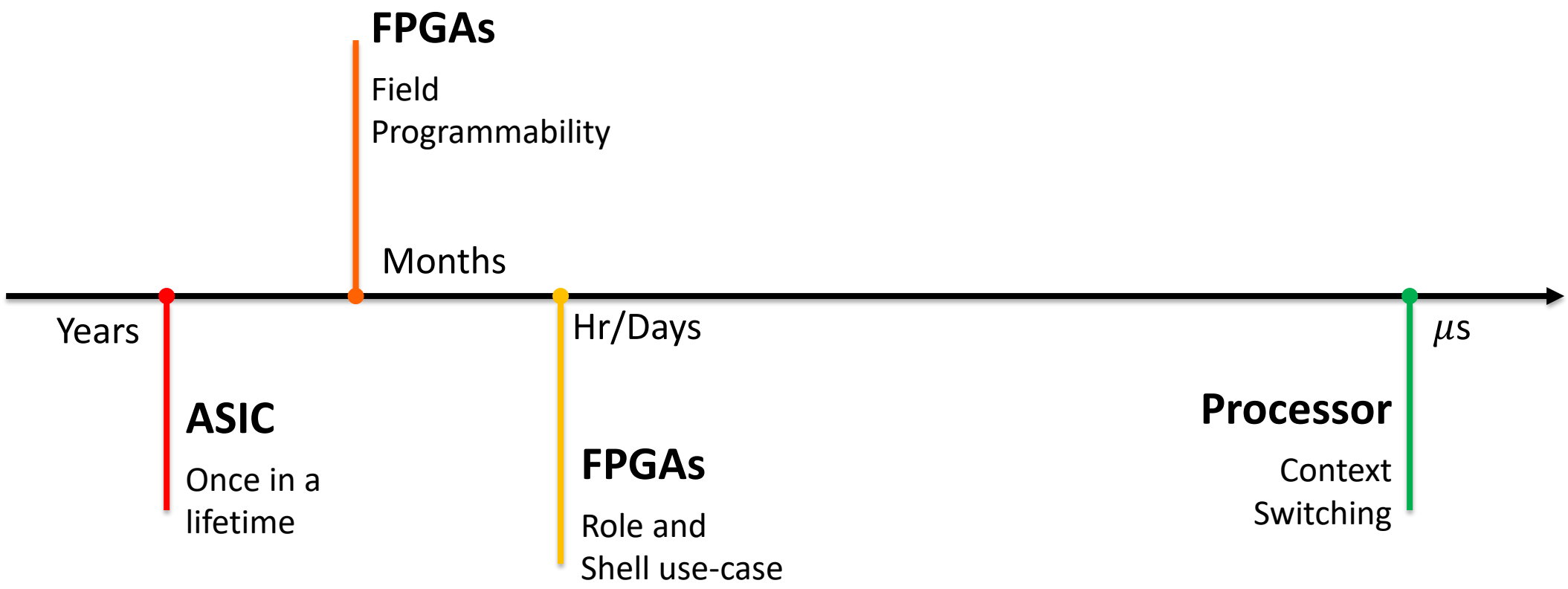
Turn Programmability into Performance

- 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]

Back to the Programmability Timescale



Renting out an Event Venue

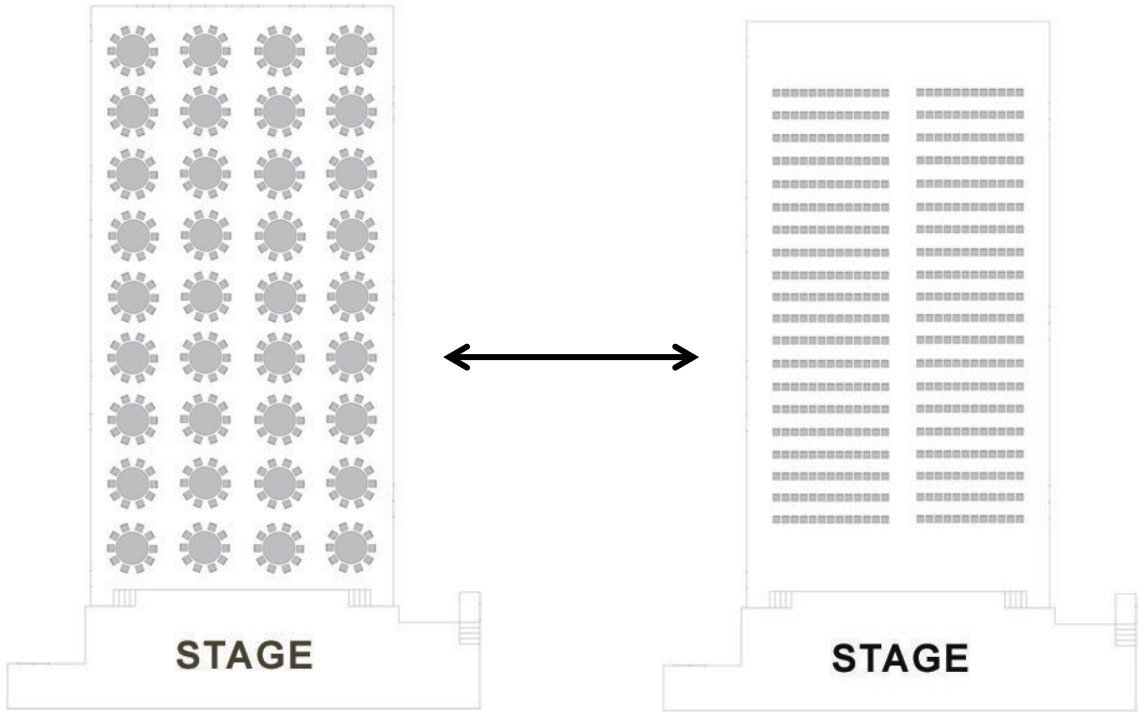
Event Planning

Want to maximize the time spent doing events…

Well… maybe we reconfigure only once a day and hold multiple events in the same day

May not be an ideal setup for all events in the same day

Specification



Reconfiguration takes a couple of hours…

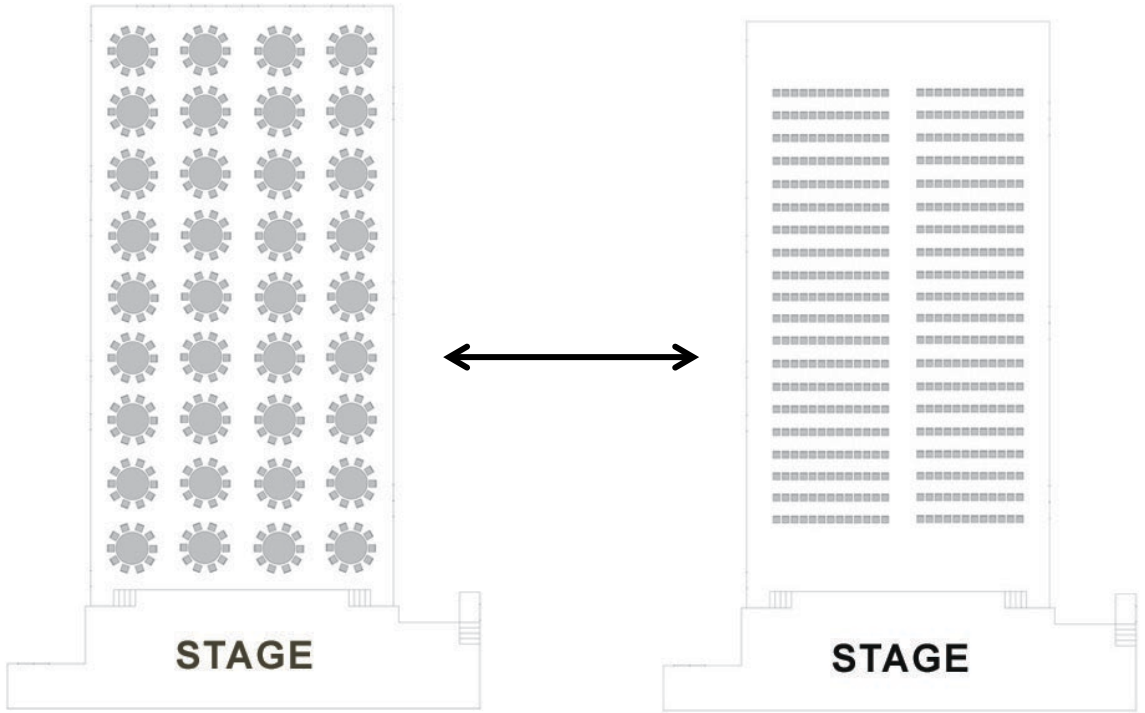
Renting out an Event Venue

Event Planning

Want to maximize the time spent doing events…

We can reconfigure between every event! The setup can now be as perfect as we can make it for that event! – We had some *slack* in efficiency which we made up

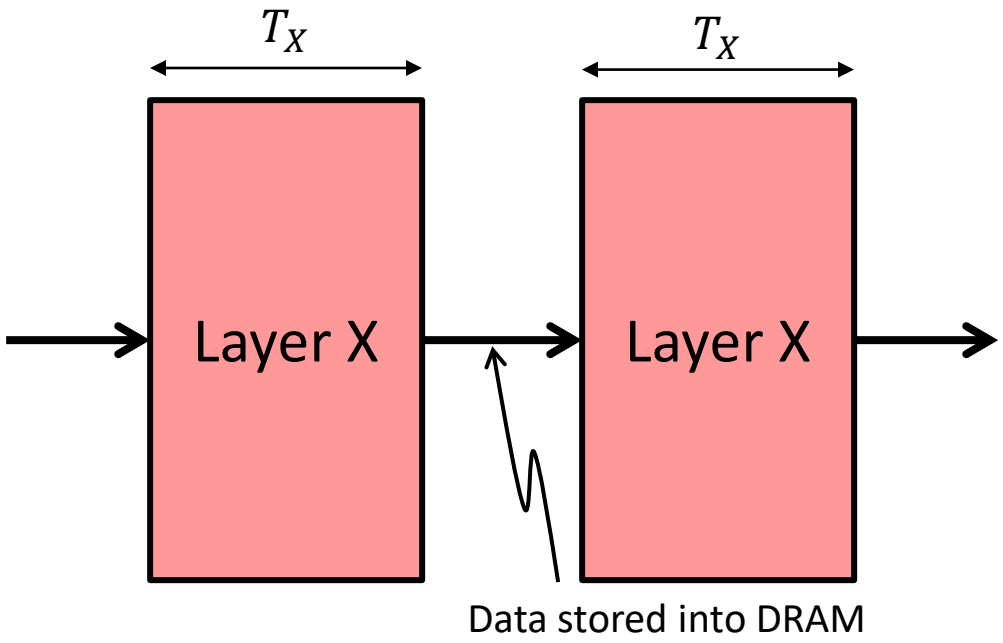
Specification



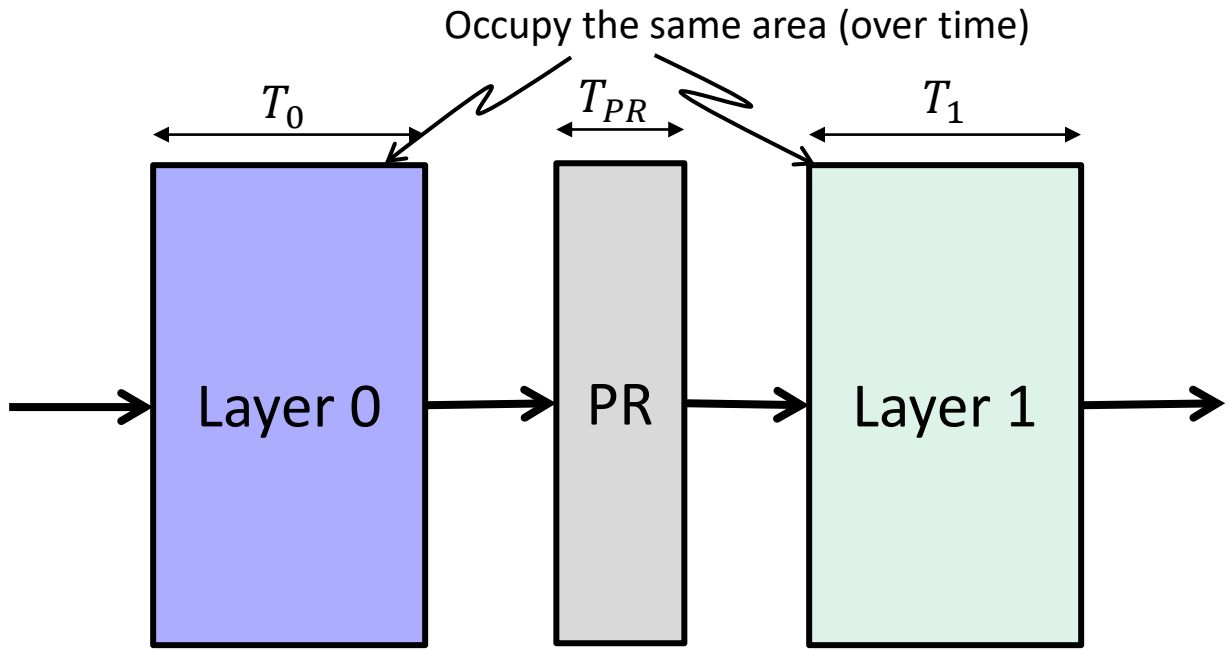
Reconfiguration takes 15 minutes…

Lab 3 – Analytically Speaking

// #define ENABLE_DFX



$$Throughput = \frac{1}{2T_X}$$



$$Throughput = \frac{1}{T_0 + T_1 + T_{PR}}$$

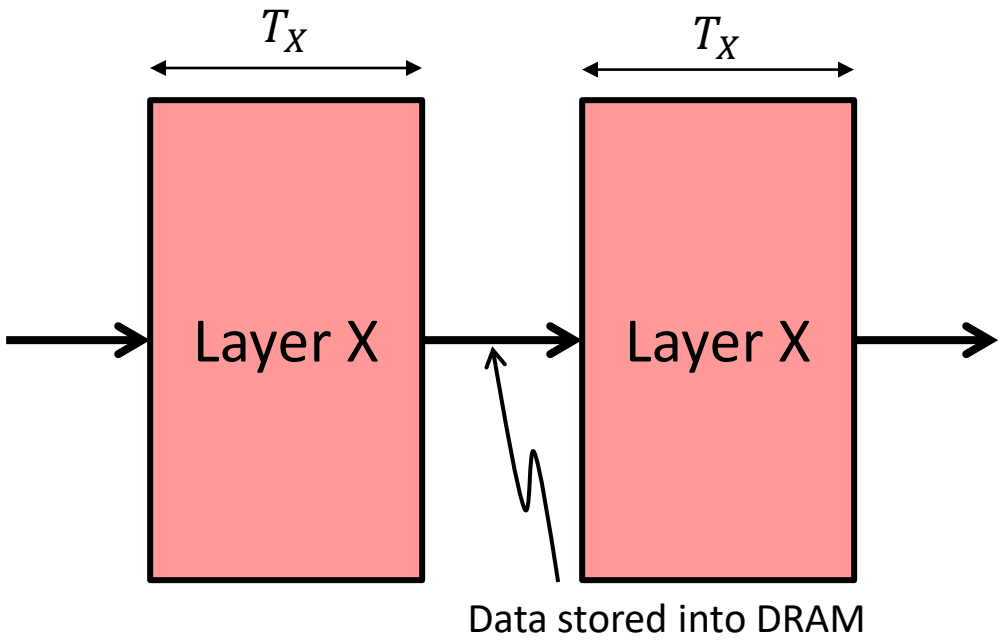
#define ENABLE_DFX

$$2T_X >^{?} T_0 + T_1 + T_{PR}$$

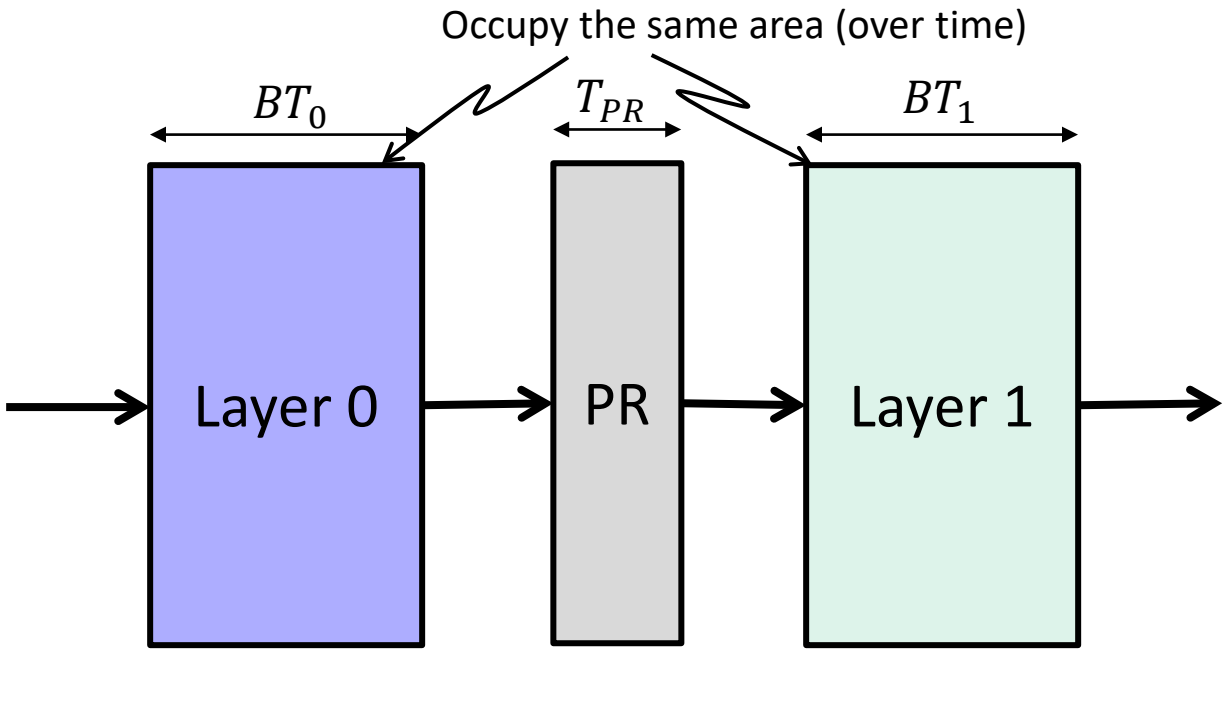
PR is not free…

Analytically Speaking (Batching)

// #define ENABLE_DFX



$$Throughput = \frac{\cancel{B}}{2\cancel{B}T_X}$$



$$Throughput = \frac{1}{T_0 + T_1 + \frac{T_{PR}}{B}}$$

#define ENABLE_DFX

$$2T_X >^{?} T_0 + T_1 + \frac{T_{PR}}{B}$$

More detailed analytical models:
https://users.ece.cmu.edu/~jhoe/distribution/2020/fpl2020dpr.pdf

Batching is not free…

Is designing two kernels vs one harder?

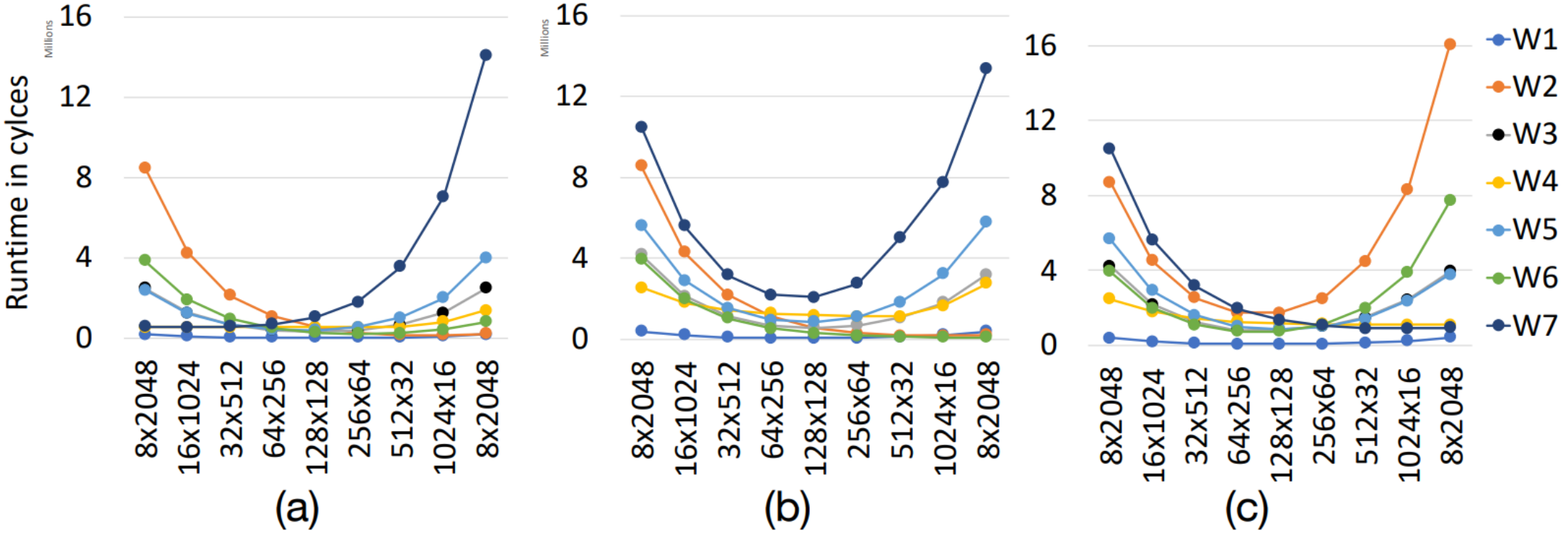


Chart showing the variation of runtime for workloads run with different shapes of systolic array with fixed number of PEs (=16384) for three dataflows (a) Output Stationary, (b) Weight Stationary, (c) Input Stationary

Compromise is hard… With DFX you only need to understand one layer at a time!

Samajdar, A., Zhu, Y., Whatmough, P., Mattina, M., & Krishna, T. (2018). Scale-sim: Systolic cnn accelerator simulator. *arXiv preprint arXiv:1811.02883*.

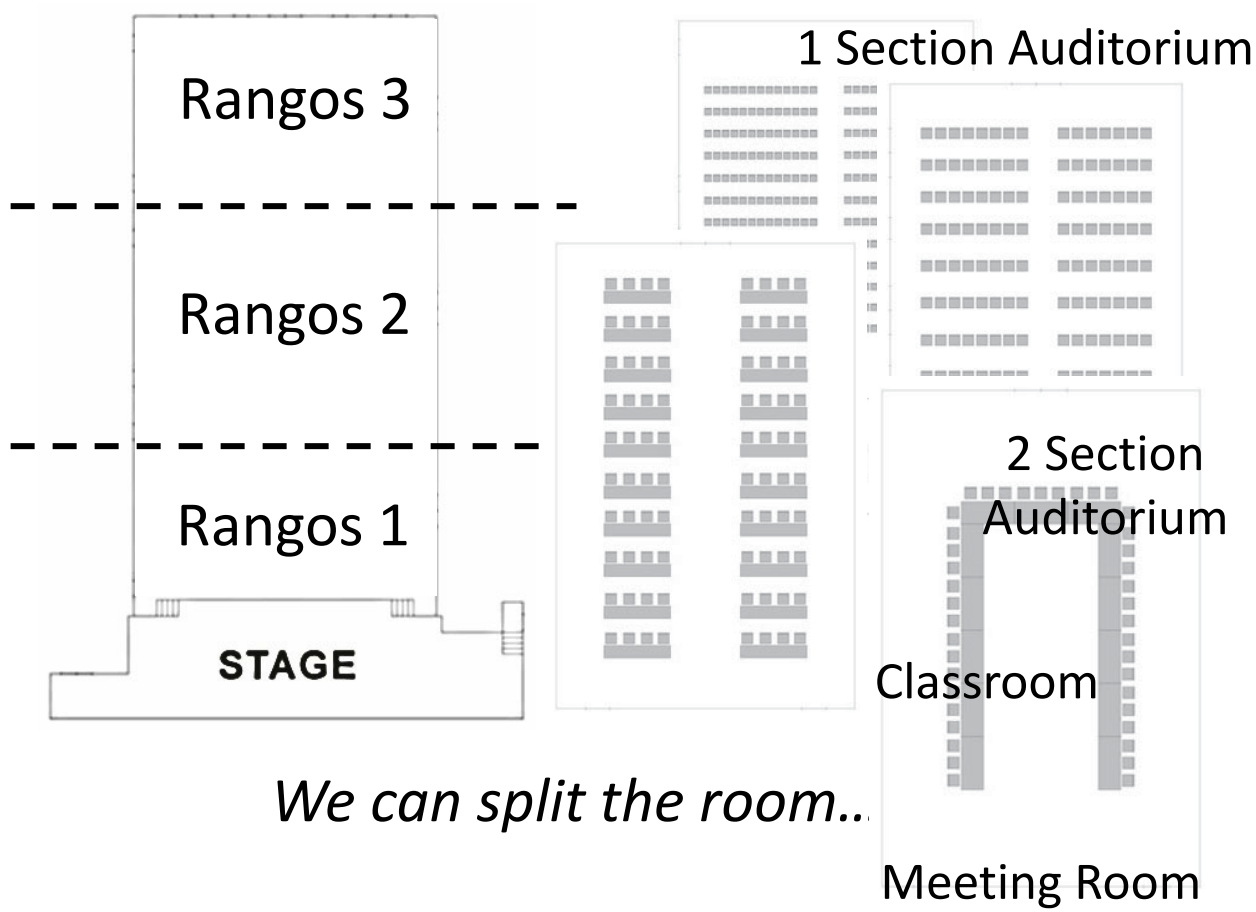
Renting out an Event Venue

Event Planning

What do we need to make use of such a setup?

- Need multiple events
- Room should be isolatable and shareable
- Pay based the number of sections used and the event duration

Specification



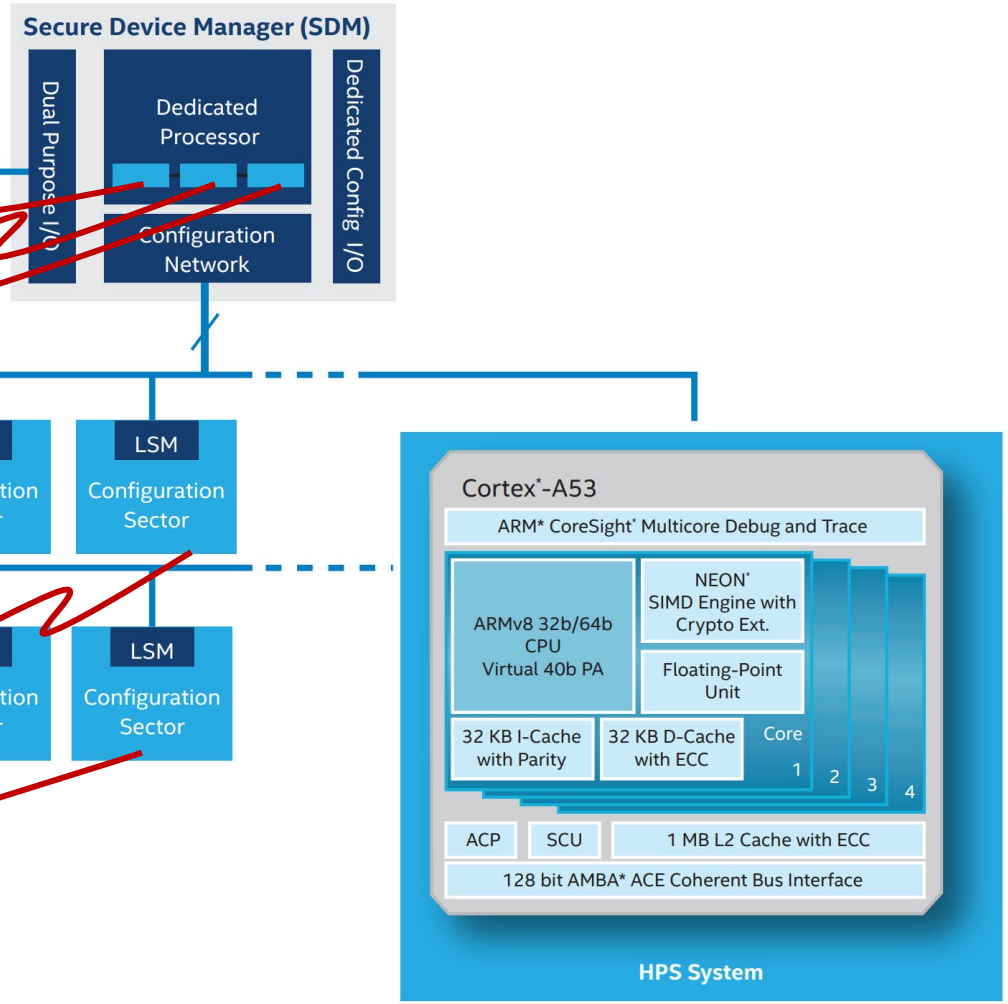
We can split the room..

Modern Configuration Architecture

e.g., Stratix® 10 Secure Device Manager

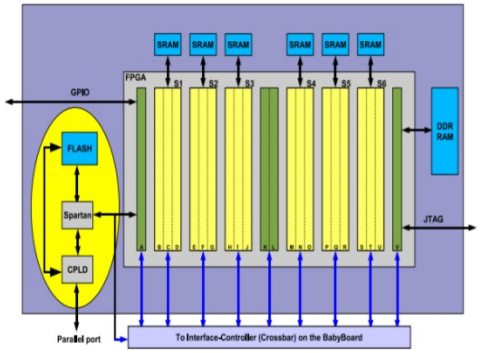
triple-redundant secure processor

each sector managed by its own processor



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

PR been around for a long time



Bobda et al. [FCCM 2005]

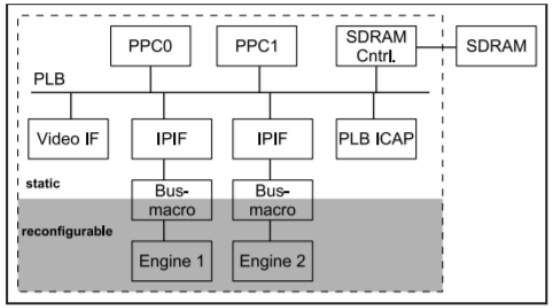
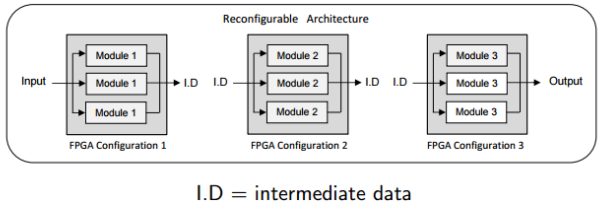


Figure 1 Block diagram of the Autovision system.

Claus et al. [2007]



Arram et al. [FPGA 2015]

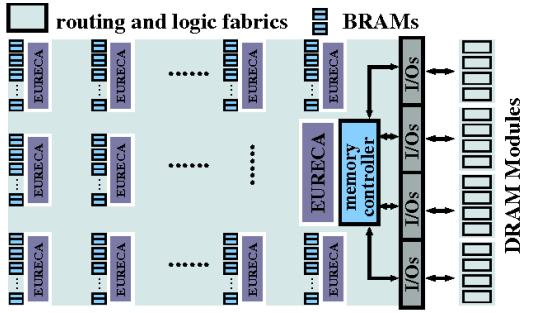
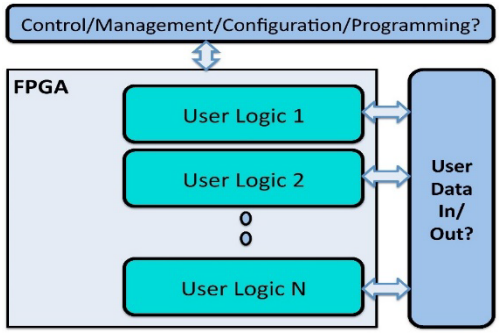


Figure 3: EURECA memory architecture overview.

Niu et al. [FPGA 2015]



Stuart et al. [FPGA 2015]

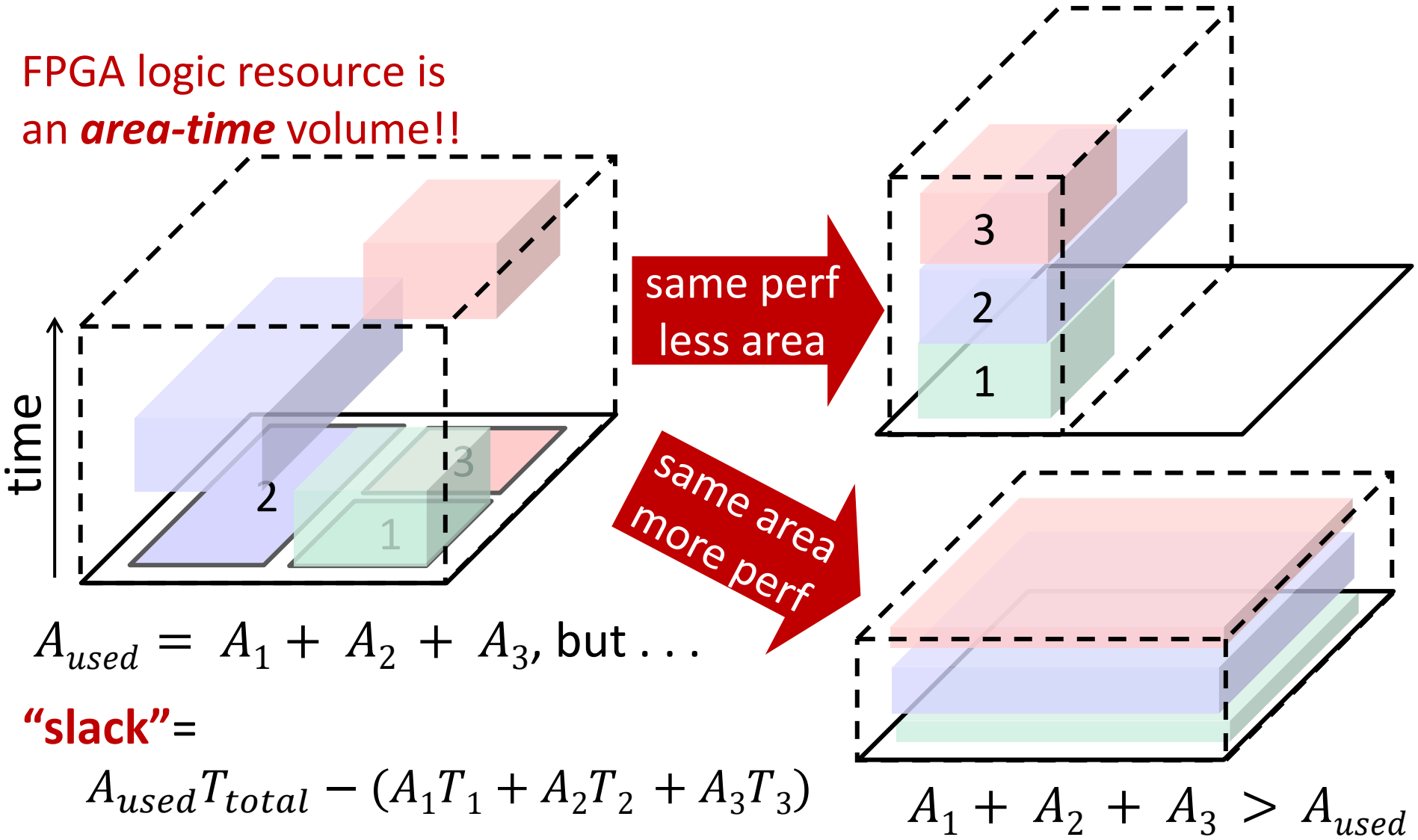
Why isn't it used?

Field Programmable to Programmable

- Programmability is a very costly feature
- When we wanted FPGA to be an ASIC
  - programmability avoided manufacturing NRE
  - programmability reduces time to solution (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

PR FPGA Designs Different from ASIC

FPGA logic resource is
an ***area-time*** volume!!



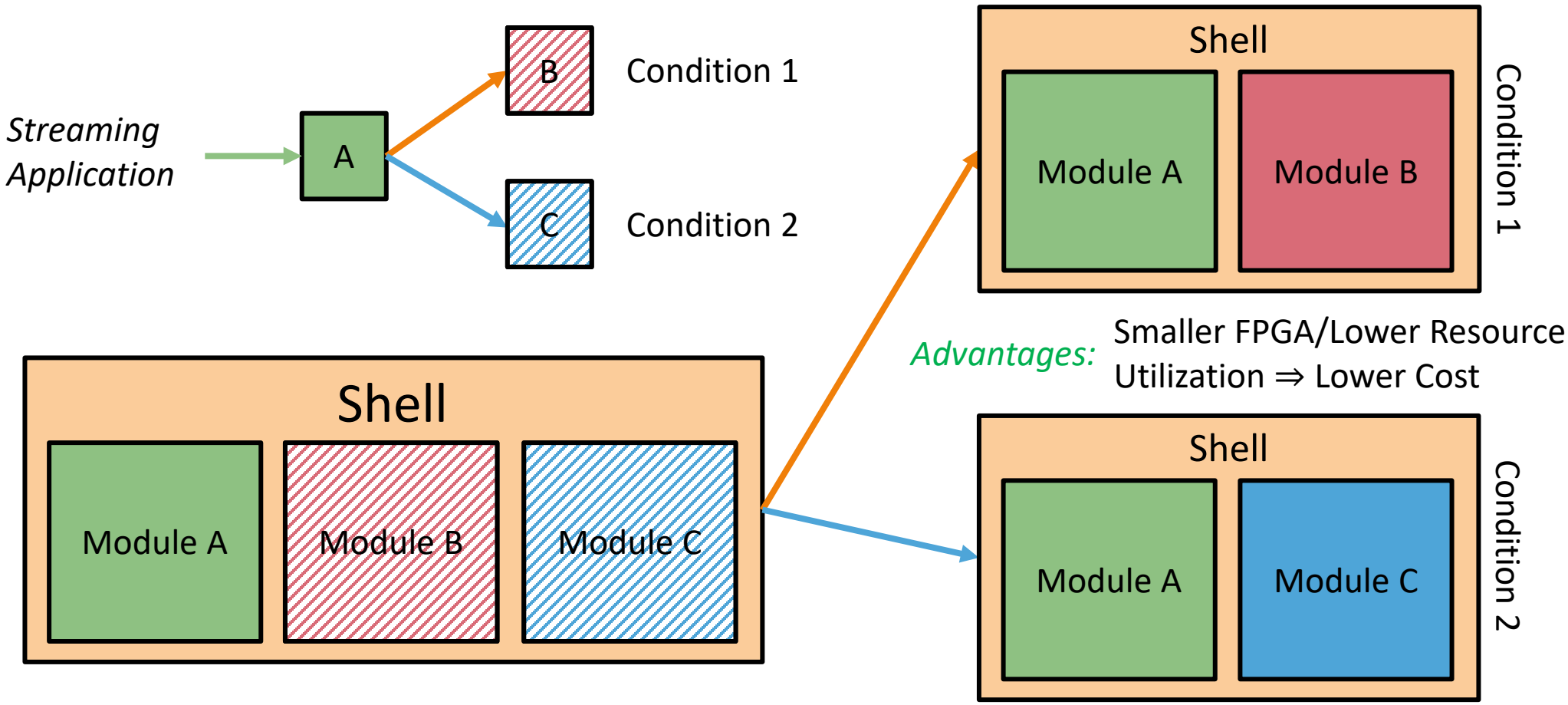
$A_{used} = A_1 + A_2 + A_3$, but . . .

"slack"=

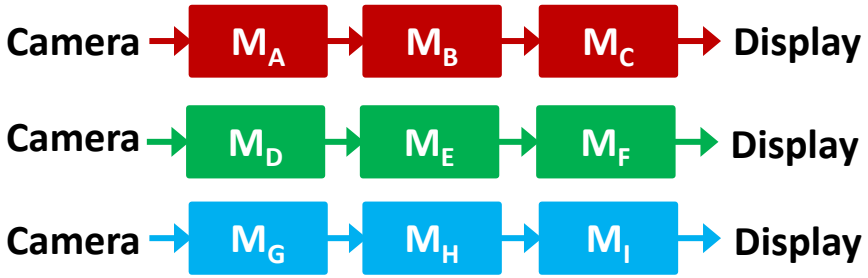
$$A_{used}T_{total} - (A_1T_1 + A_2T_2 + A_3T_3)$$

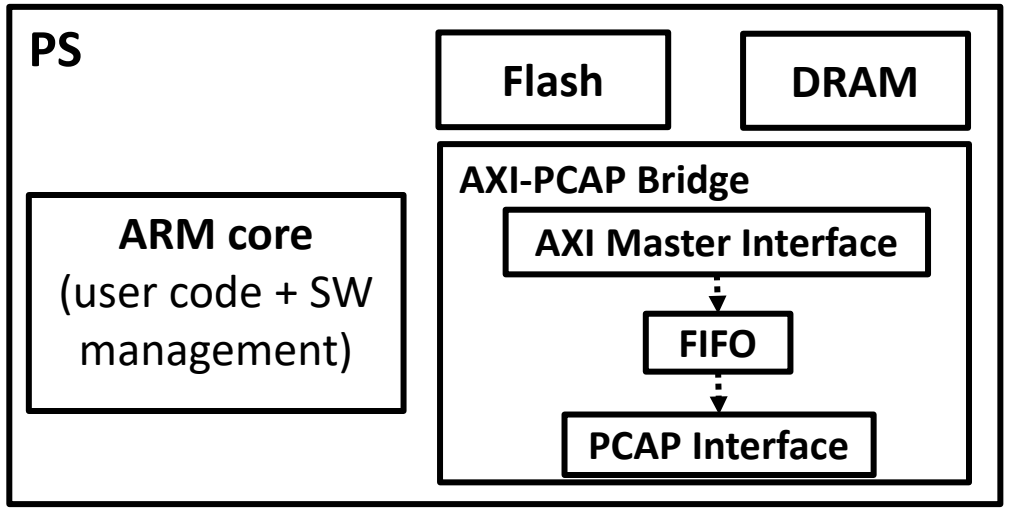
$$A_1 + A_2 + A_3 > A_{used}$$

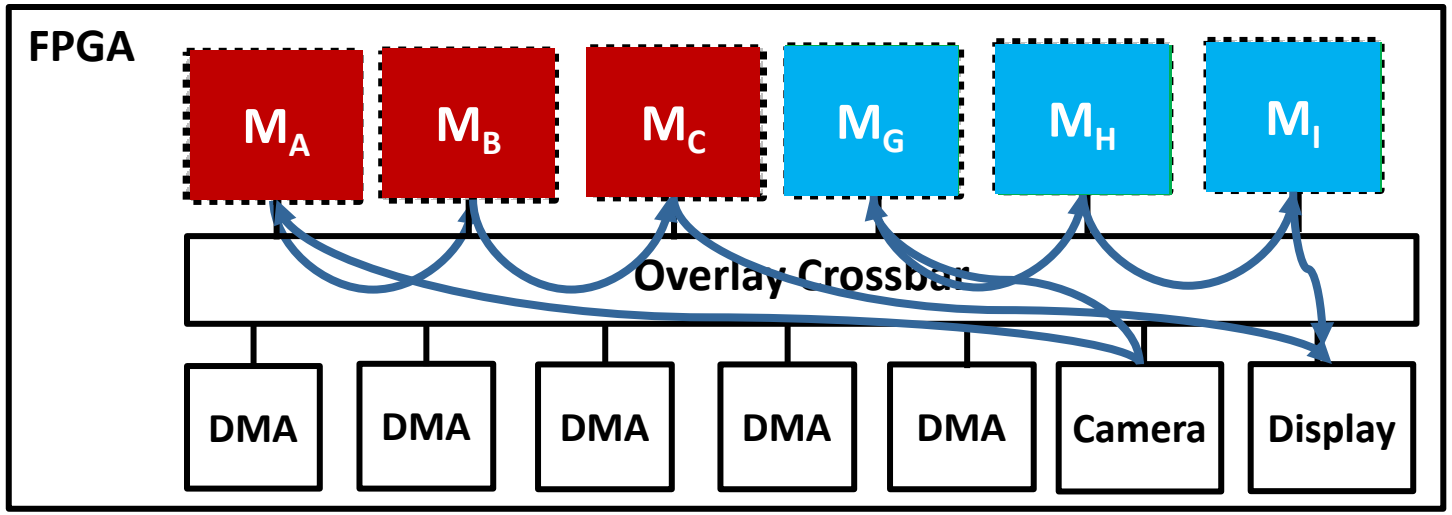
Exploiting Design Slack at Runtime



Spatial and Temporal Multitenancy

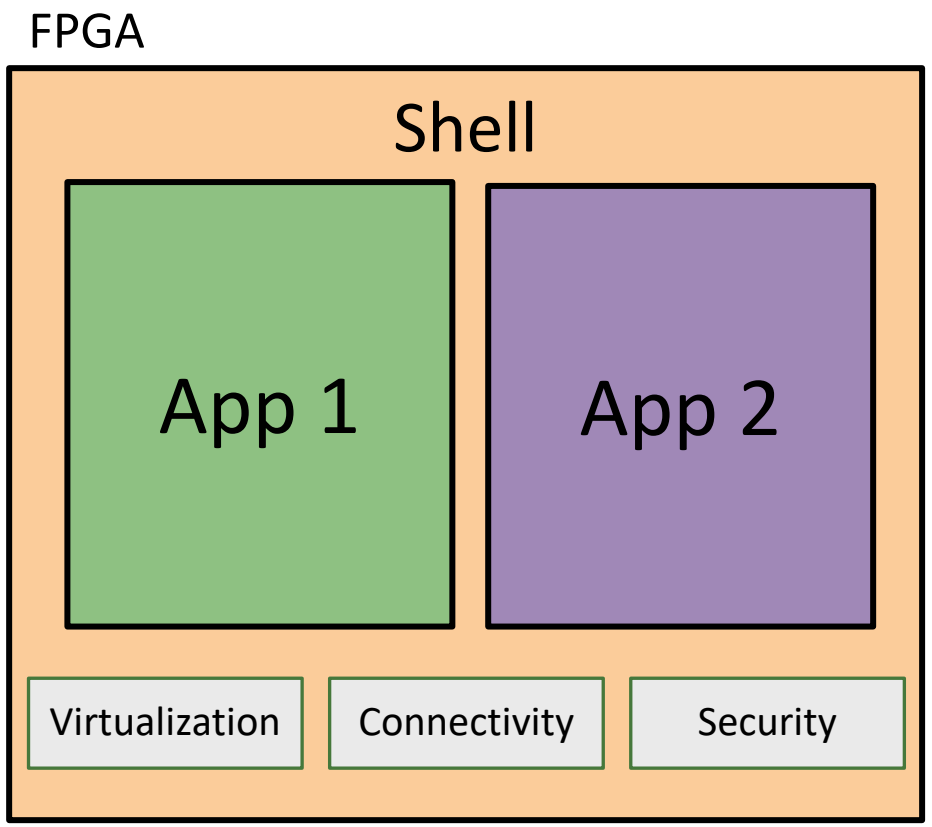






https://users.ece.cmu.edu/~jhoe/distribution/2018/fpl2018.pdf

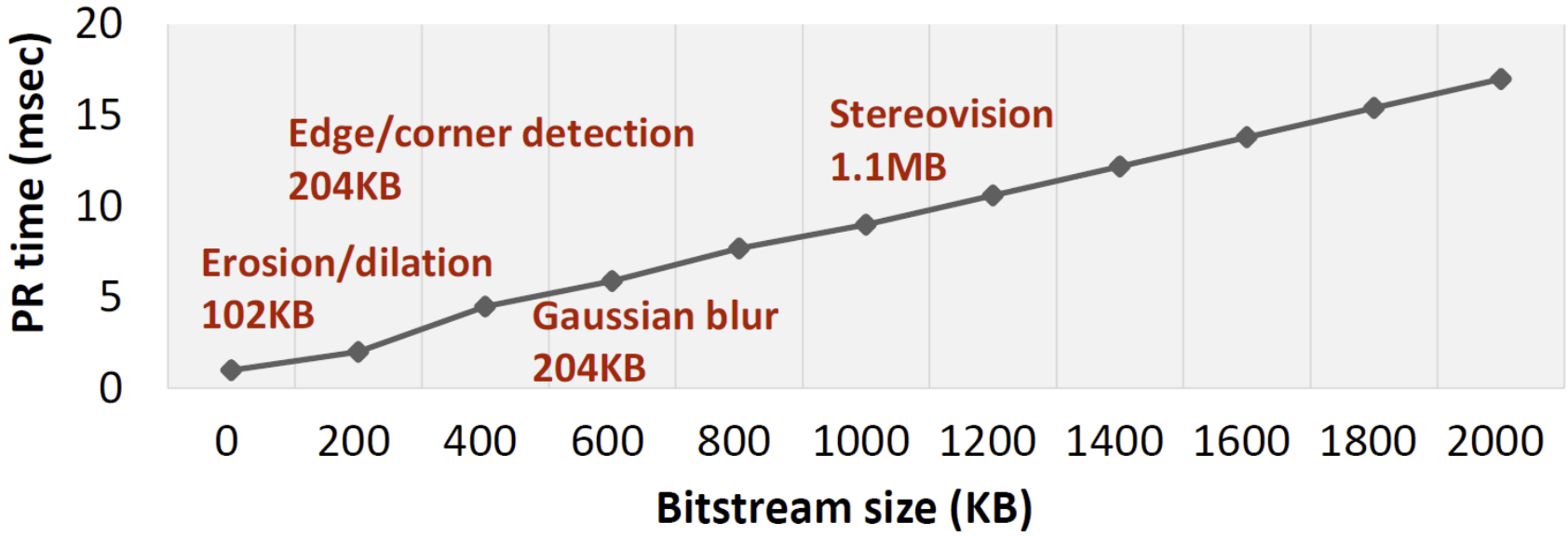
Today's Practical Constraints



- # 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

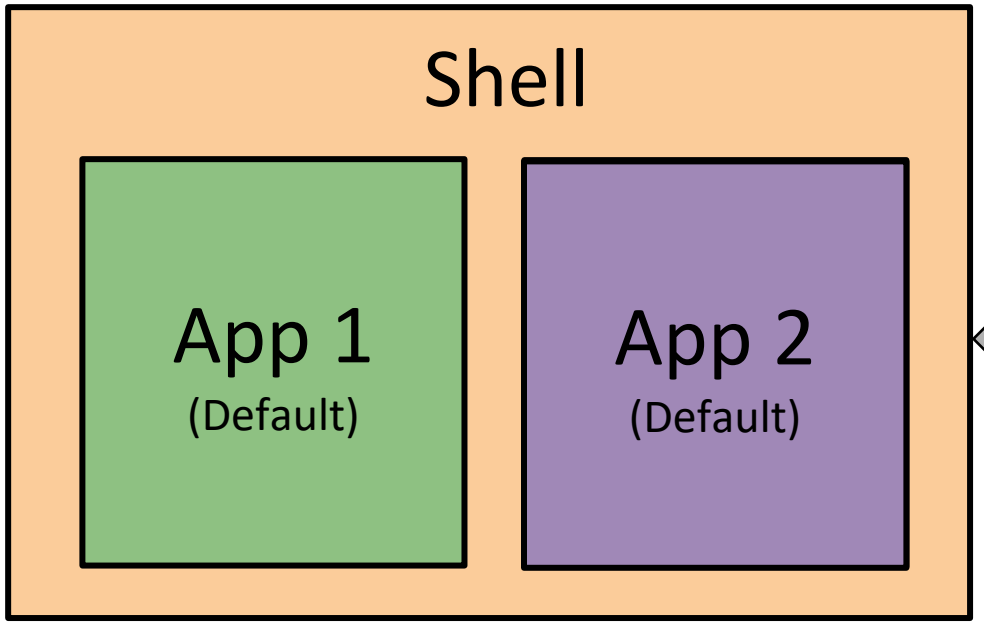
Today's PR Overhead

- Reconfigurations take on the order of msec
- Time to reconfig grows with PR partition size (~128MB/s with Xilinx PCAP)
- Only one PR with PCAP at a time



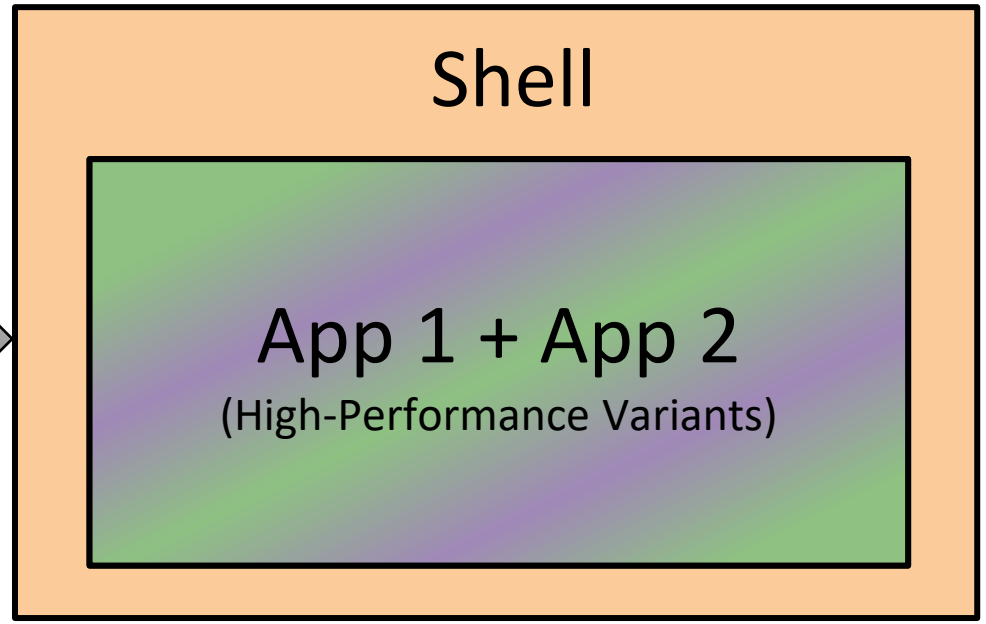
Many innovative ways to get around: Hierarchical PR

Spatial and Time Multiplexing



Multiple simultaneous applications

Simple Role and Shell design



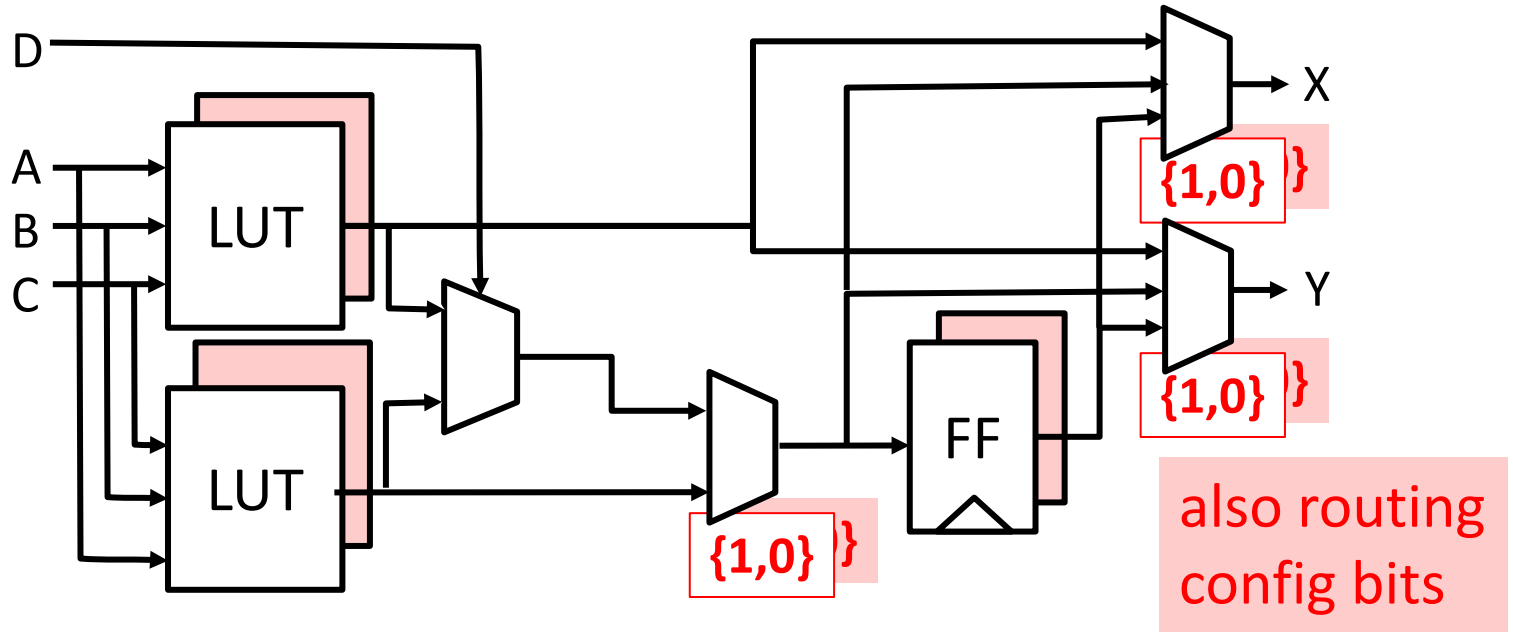
Avoids disadvantages of slot-based approach

But did we really get out of the "ASIC-style" of thinking?

Khawaja, A., Landgraf, J., Prakash, R., Wei, M., Schkufza, E., & Rossbach, C. J. (2018). Sharing, protection, and compatibility for reconfigurable fabric with amorphos. In 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18) (pp. 107-127).

ASIDE: Multi-Context Configuration?

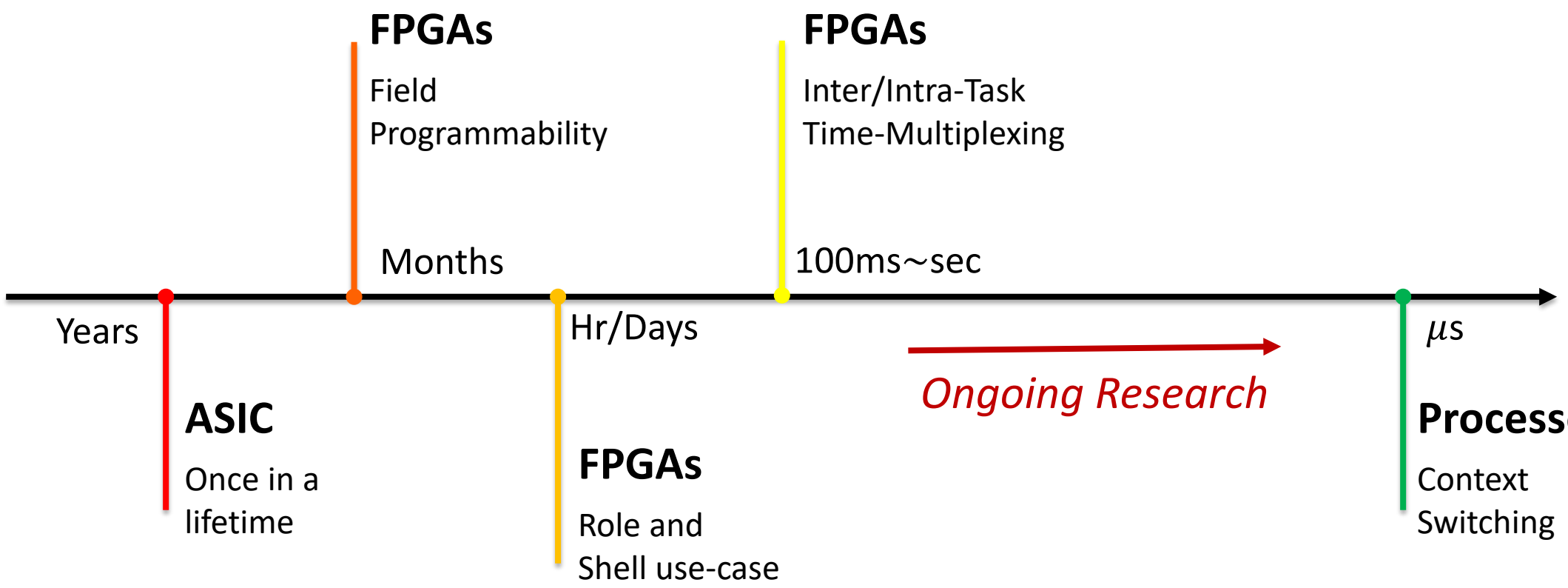
- If bitstream defines the chip, load multiple contexts and switch between them, may be even cycle by cycle – 3D FPGA with time as a dimension



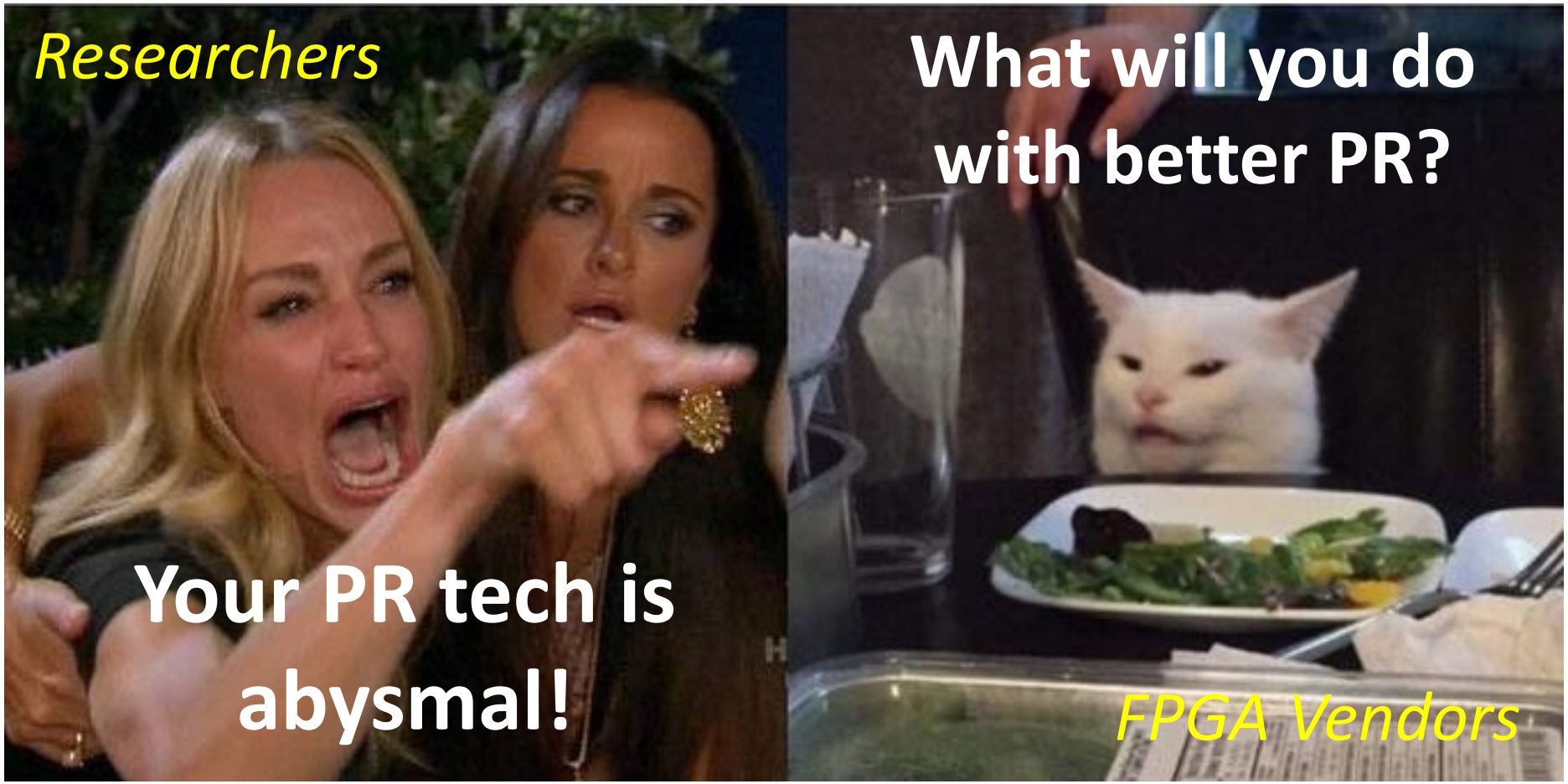
  - what is and isn't amortized across contexts?
  - studied in research and attempted commercially

(see Tabula - Spacetime)

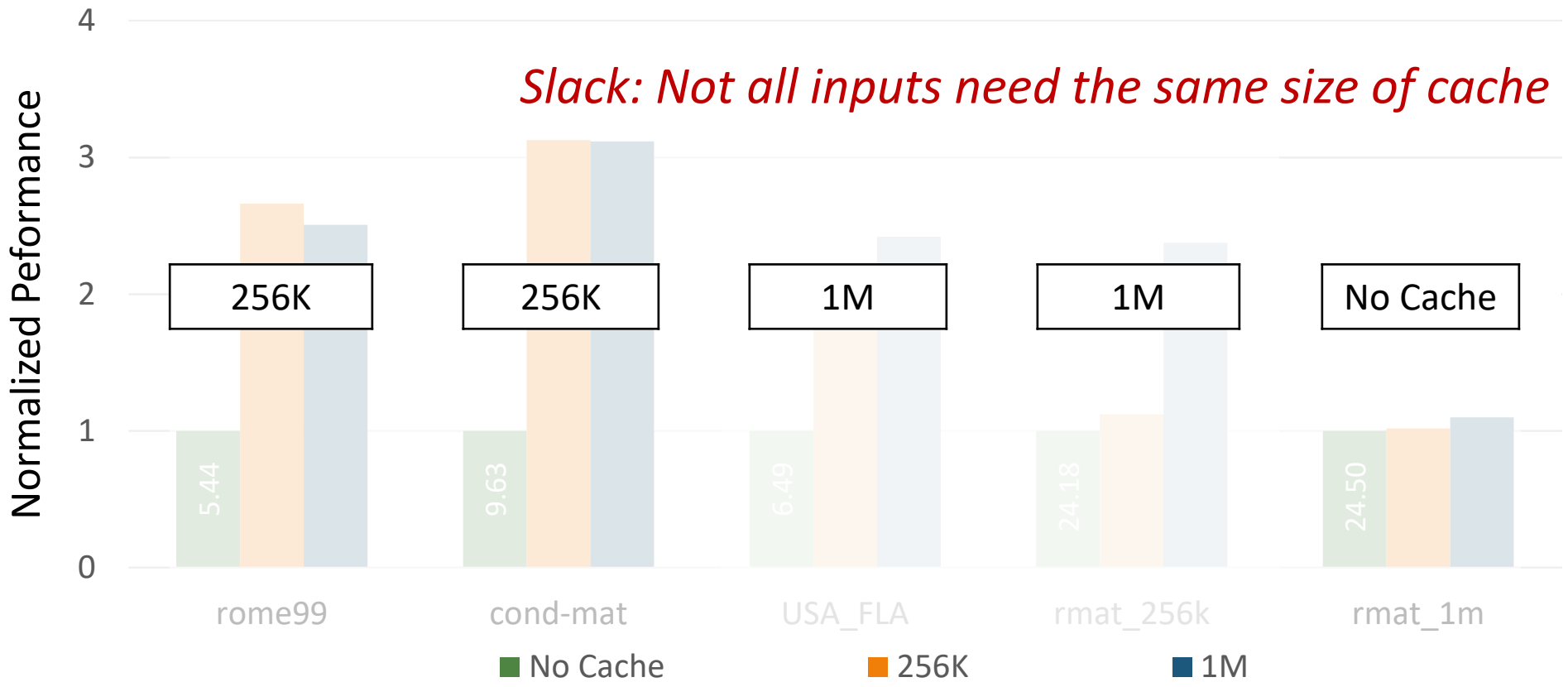
This is where we're at technologically…



We want to push further but: chicken or egg?

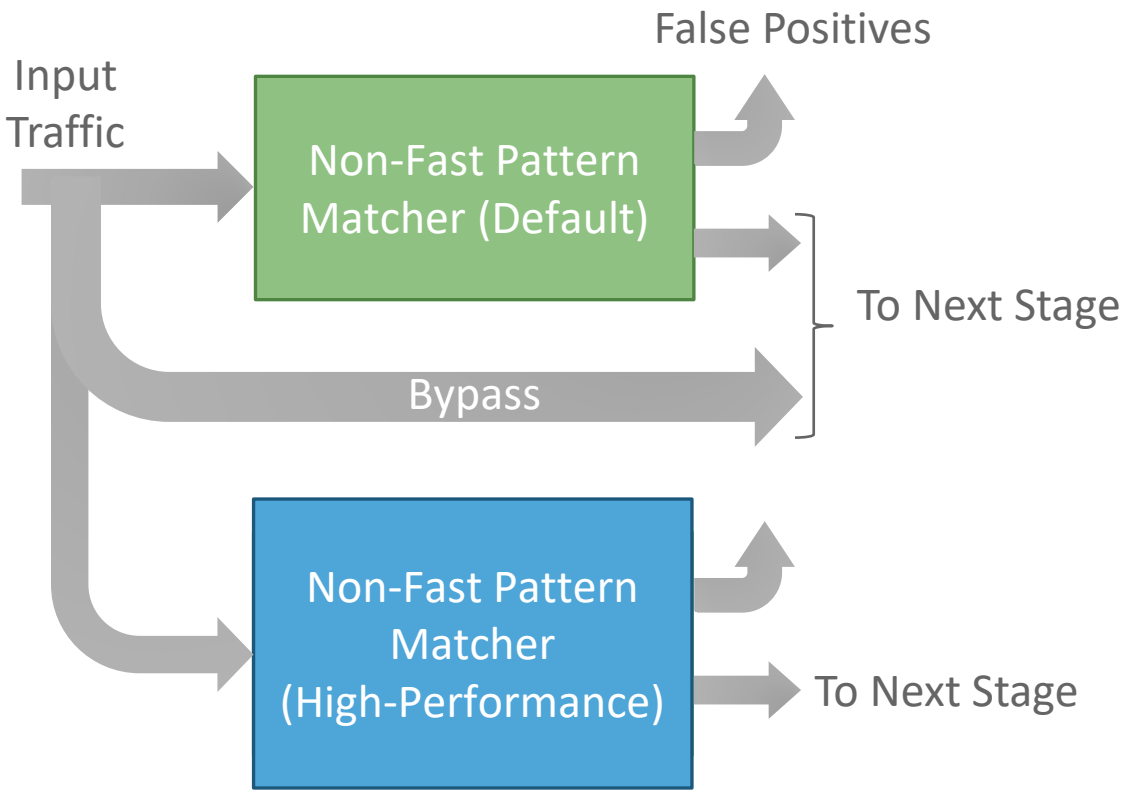


Effect of Caching on BFS Performance (MTEPS)



Reacting to bursty traffic patterns

- *QoS Requirement*: Zero Packet Loss in Intrusion Detection/Prevention System
- Single NFPM sufficient for usual case of average 15Gbps stage traffic
- Bypass can handle occasional overflow cases – but what if not occasional?
- *Slack: We don't need the units all the time* – Dynamically request more parallel processing units or faster units with PR



Does this make sense on current FPGA architectures/deployments?

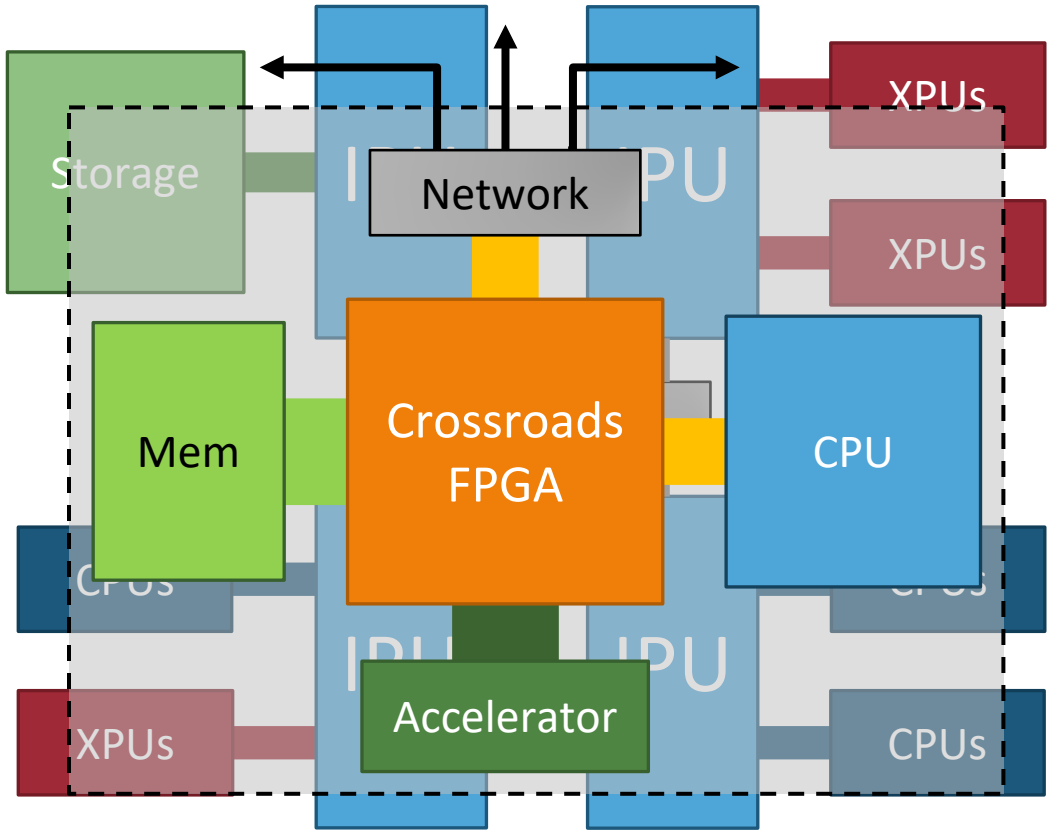
One Deployment, One Design

- **Compile time** choices
  - re-tune for large vs small FPGA part choices
  - re-tune for same FPGA part for different deployments
- **Runtime** choices
  - adjust datapath to changing operating conditions
  - add/change FPGA usage to changing conditions
- **Specializing** (compile time and/or runtime) by substituting or inserting custom nodes

FPGA is more than ASIC; don't use it as less

FPGAs in Datacenters (More in Week 11)

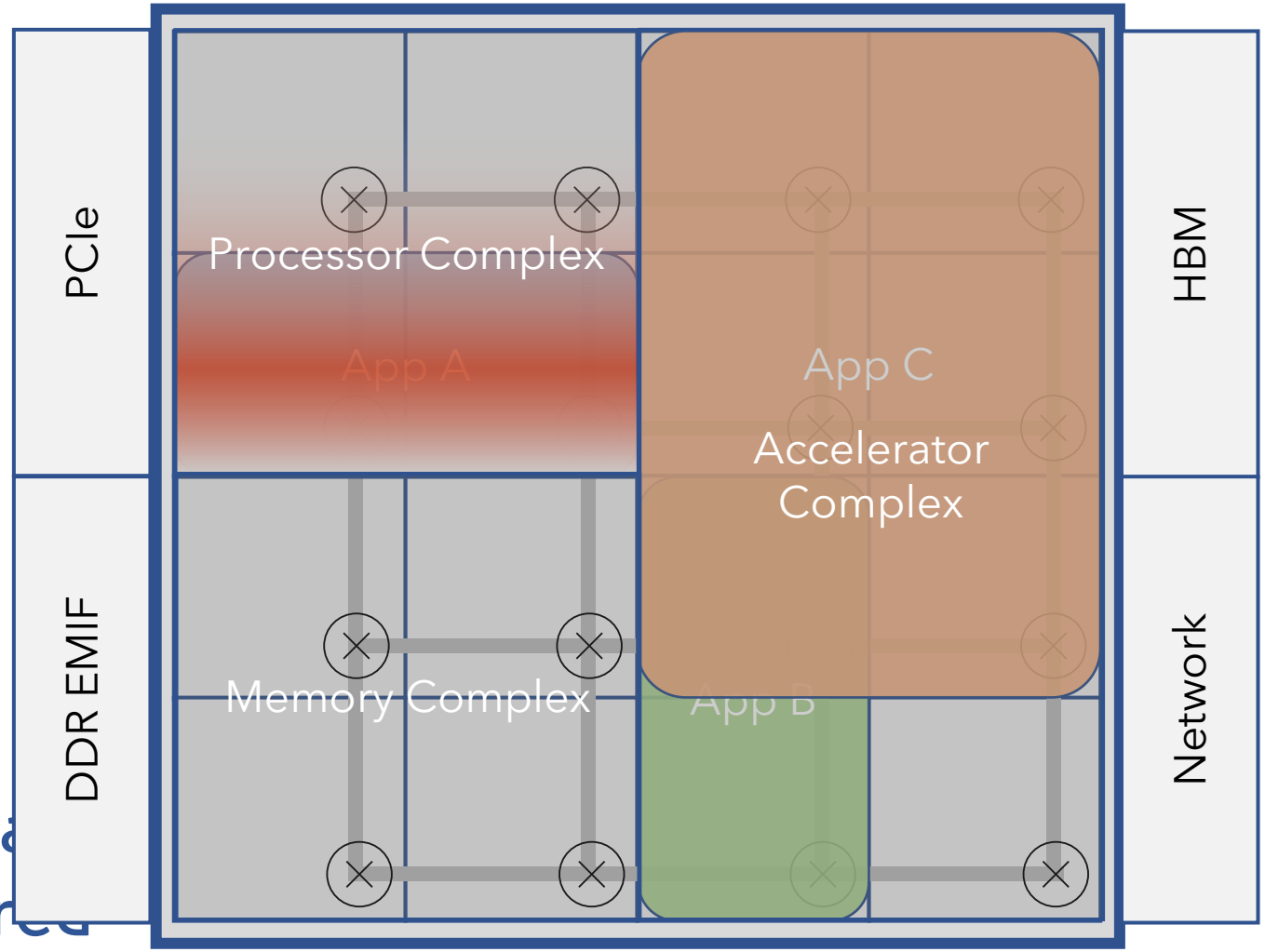
- FPGA as a data hub allows computing near-data and on data flows
- Immense demand to be near-data – PR is essential to cater to this demand and support multi-tenancy
- Traditional ASIC-style approaches under-utilize the programmability of FPGAs
  - Need better PR hardware
  - Applications designed to exploit slack



Look up Research Vector 5 at www.crossroadsfpga.org/seminars

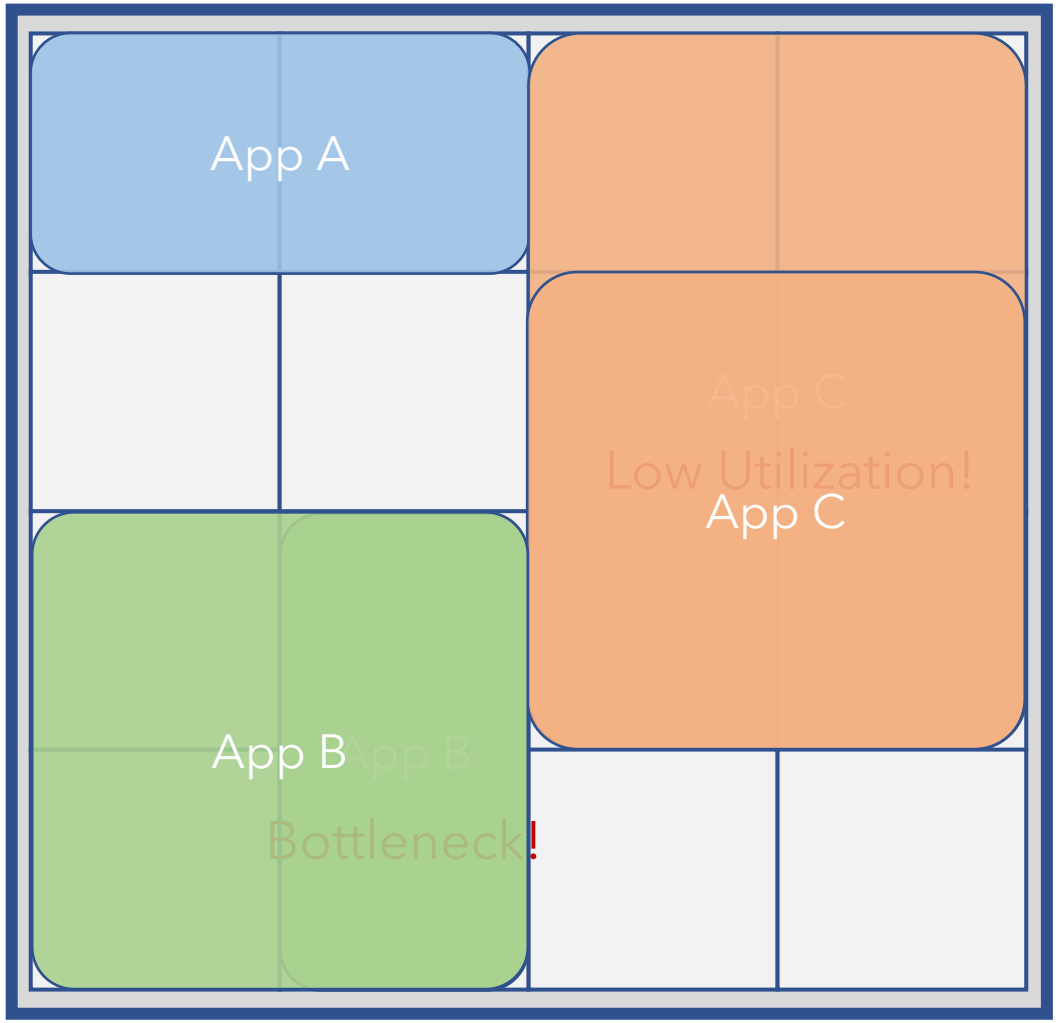
IPU: https://www.intel.com/content/www/us/en/newsroom/news/infrastructure-processing-unit-data-center.html

RV5: PR is essential to Crossroads



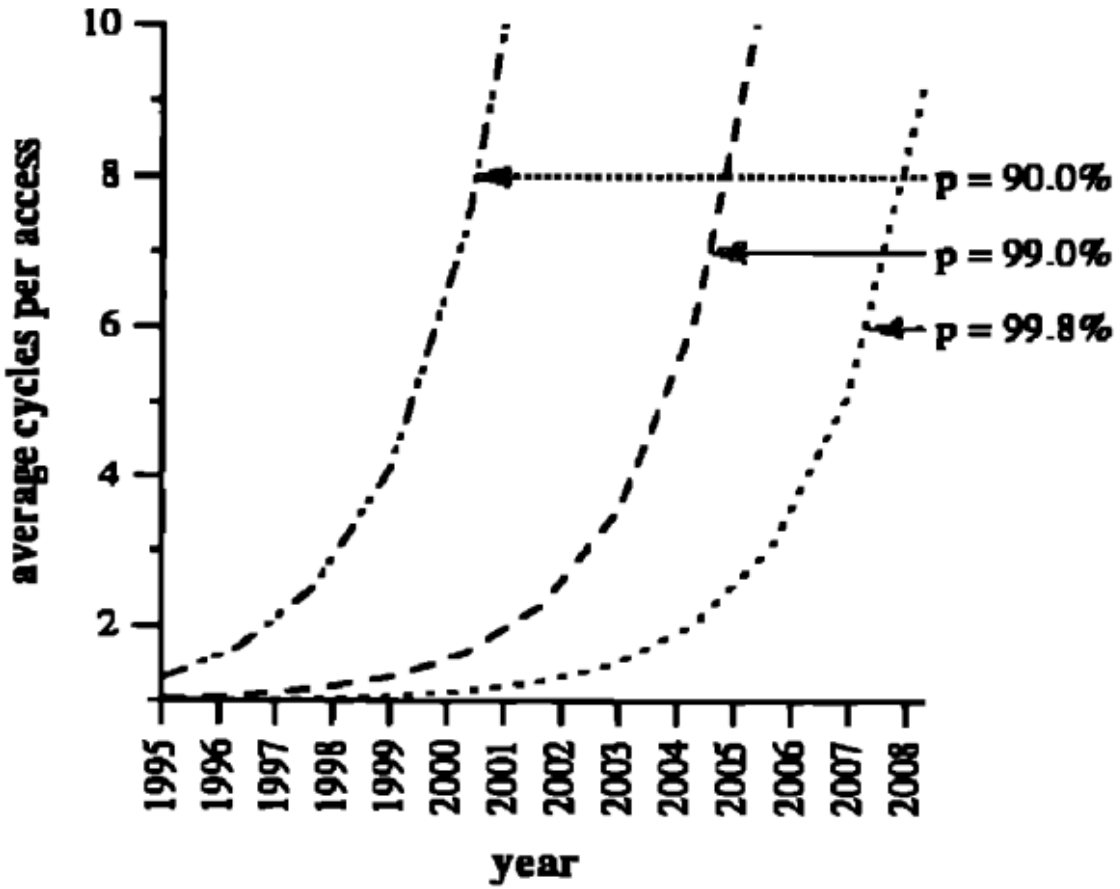
The Crossroads FPGA is virtualized!

RV5: PR can be used for more!

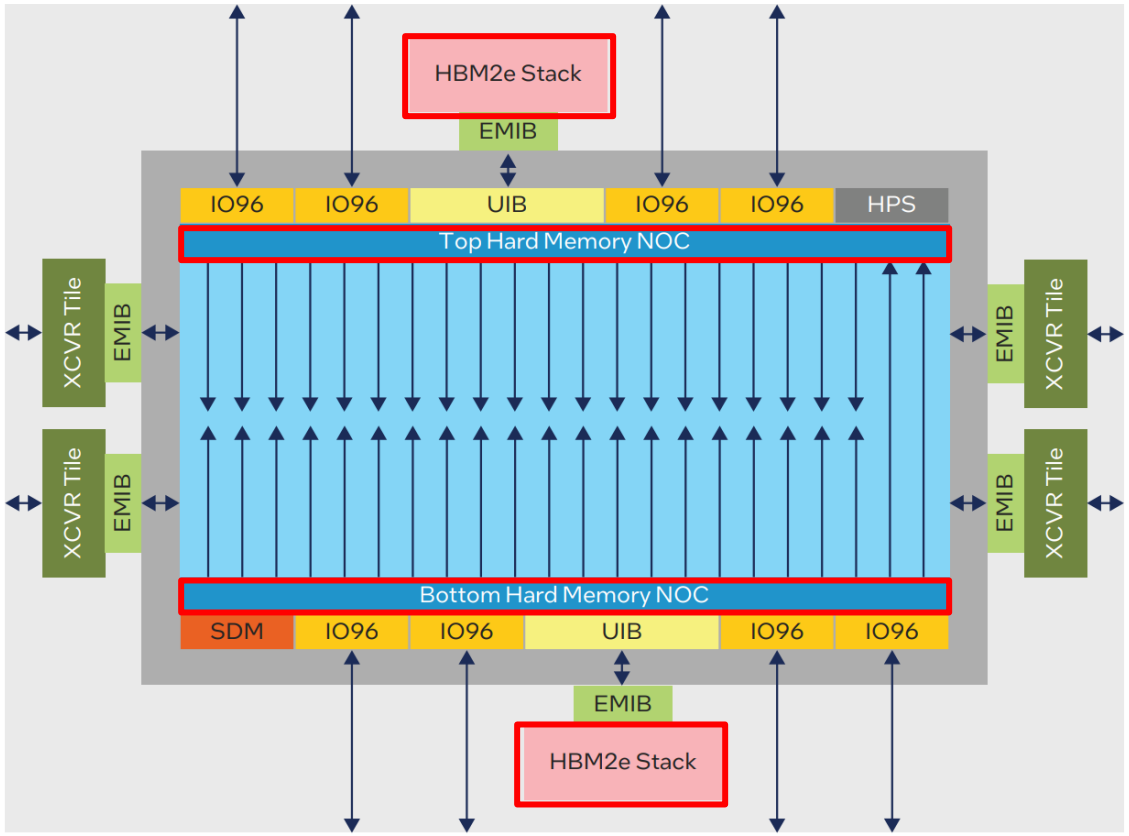


How do we recognize these situations and respond to them?

Memory Bandwidth – A Balancing Act

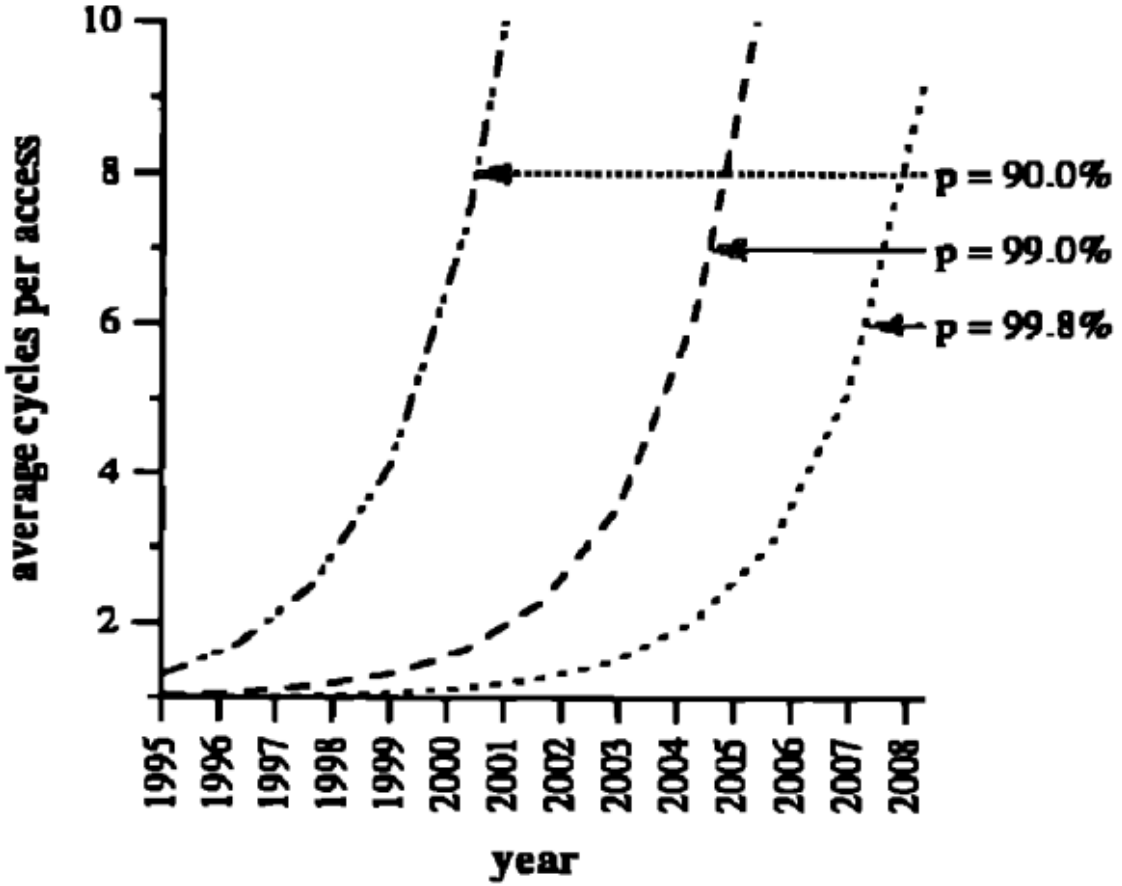


Wm. A. Wulf and Sally A. McKee. 1995. Hitting the memory wall: implications of the obvious. SIGARCH Comput. Archit. News 23, 1 (March 1995), 20–24

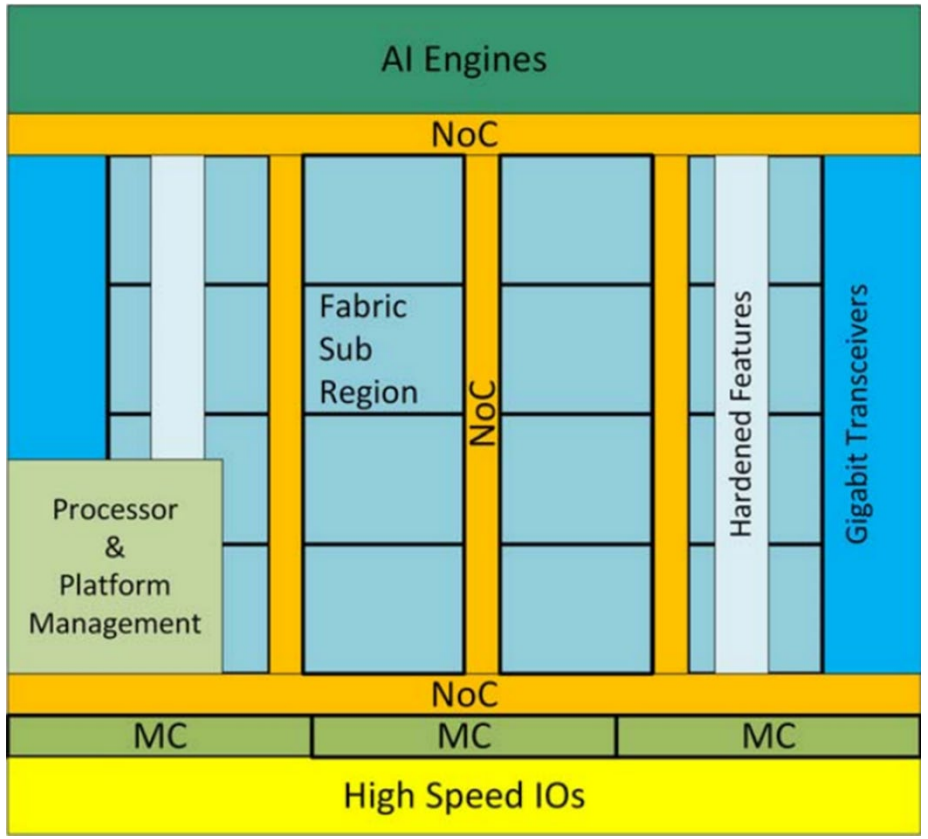


Intel Agilex M: https://www.intel.com/content/www/us/en/products/docs/programmable/agilex-m-series-memory-white-paper.html

Memory Bandwidth – A Balancing Act

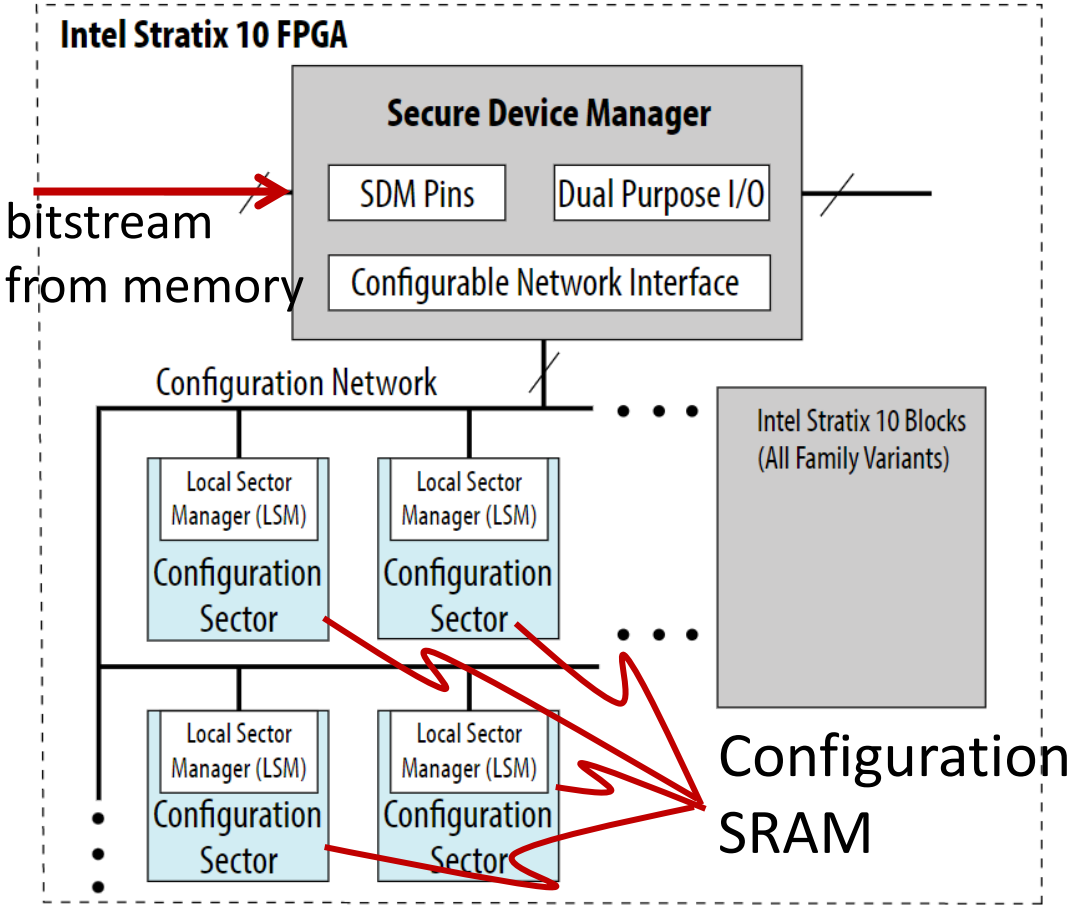


Wm. A. Wulf and Sally A. McKee. 1995. Hitting the memory wall: implications of the obvious. SIGARCH Comput. Archit. News 23, 1 (March 1995), 20–24



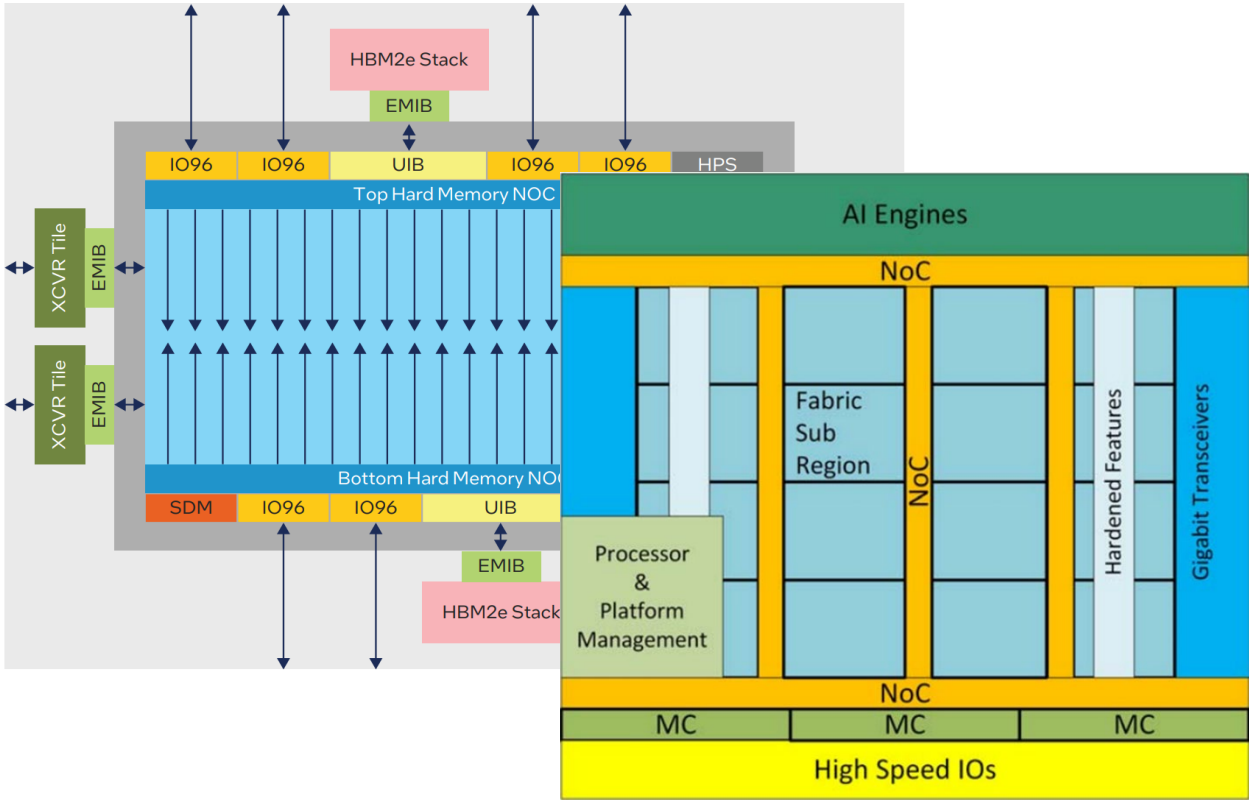
Xilinx Versal: I. Swarbrick *et al*., "Versal Network-on-Chip (NoC)," *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*, 2019, pp. 13-17, doi: 10.1109/HOTI.2019.00016.

PR has a lot to contend with…



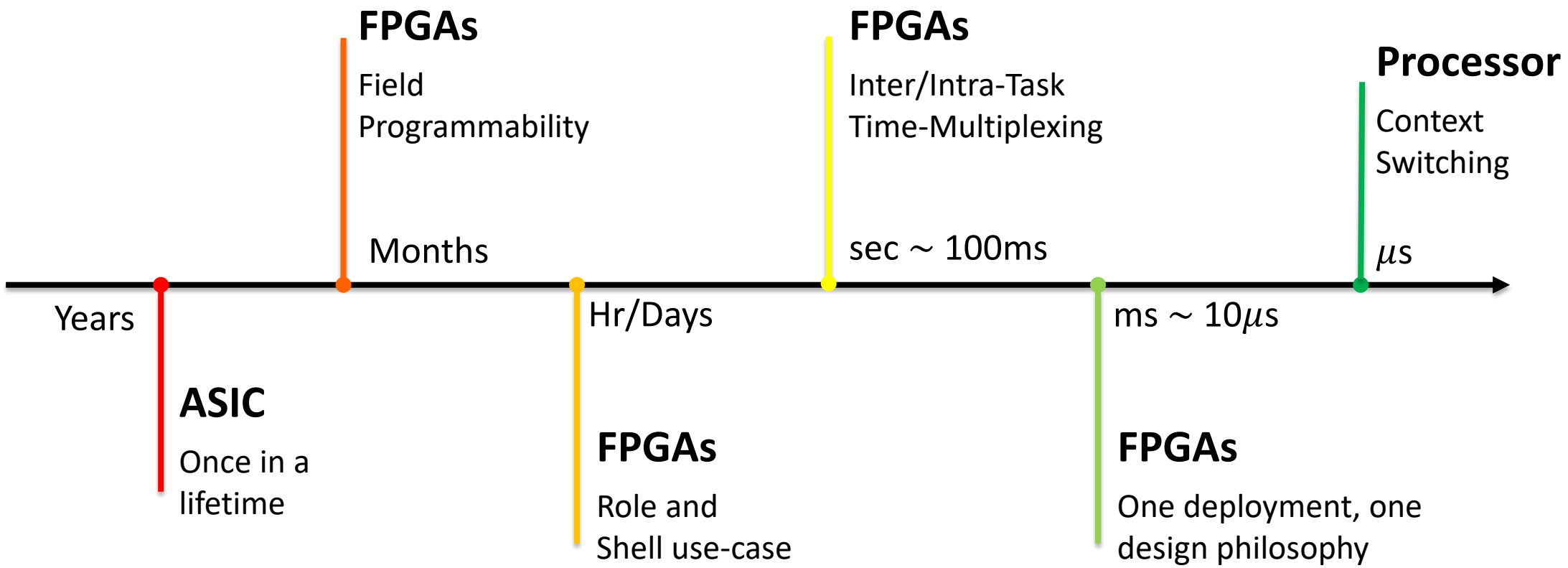
https://www.intel.com/content/www/us/en/docs/programmable/683717/current/device-configuration-and-secure-device.html

Intel Agilex M: https://www.intel.com/content/www/us/en/products/docs/programmable/agilex-m-series-memory-white-paper.html



Xilinx Versal: I. Swarbrick *et al.*, "Versal Network-on-Chip (NoC)," *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*, 2019, pp. 13-17

It's always been about reclaiming "Slack"



We filled the gap! But it gets harder the finer we go…

Break further from the ASIC mentality

- **Dynamism** — actually use the programmability
  - support more functionality on same parts cost
  - achieve better performance by specializing
- **Shareability** — multitenancy to consume "slack"
  - too much resource: partition fabric spatially
  - too much throughput: repurpose fabric temporally
- **Manageability** — bring FPGA under OS purview
  - part of compute resource pool (CPU cycles, DRAM)
  - seamless interface, virtualization and isolation (security and QoS)

Parting Thoughts

- FPGA's win over processor is speed and efficiency; FPGA's win over ASIC is flexibility
- For computing, don't use FPGA like an ASIC; but don't think about it like a processor either
- Partial reconfiguration really does work!!
  - put it to good use in Lab 3
  - could still be much better . . . .
  - have to find strong new uses and use modality and properly integrate and support it

Go from applying FPGAs to computing

⇒ making better FPGAs for computing