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 for >200

Hall Design

E.g., McConomy Auditorium

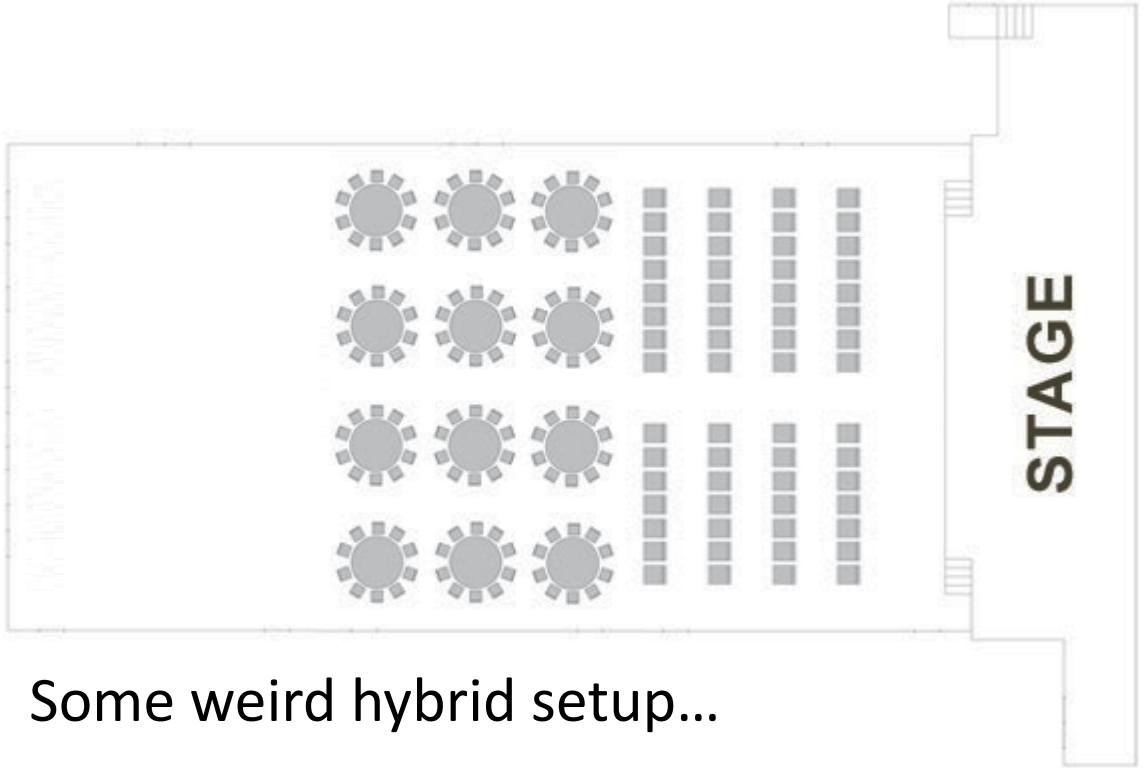
You are designing an Event Hall

Specification

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

And switch between different events quickly – no downtime

Hall Design



Some weird hybrid setup…

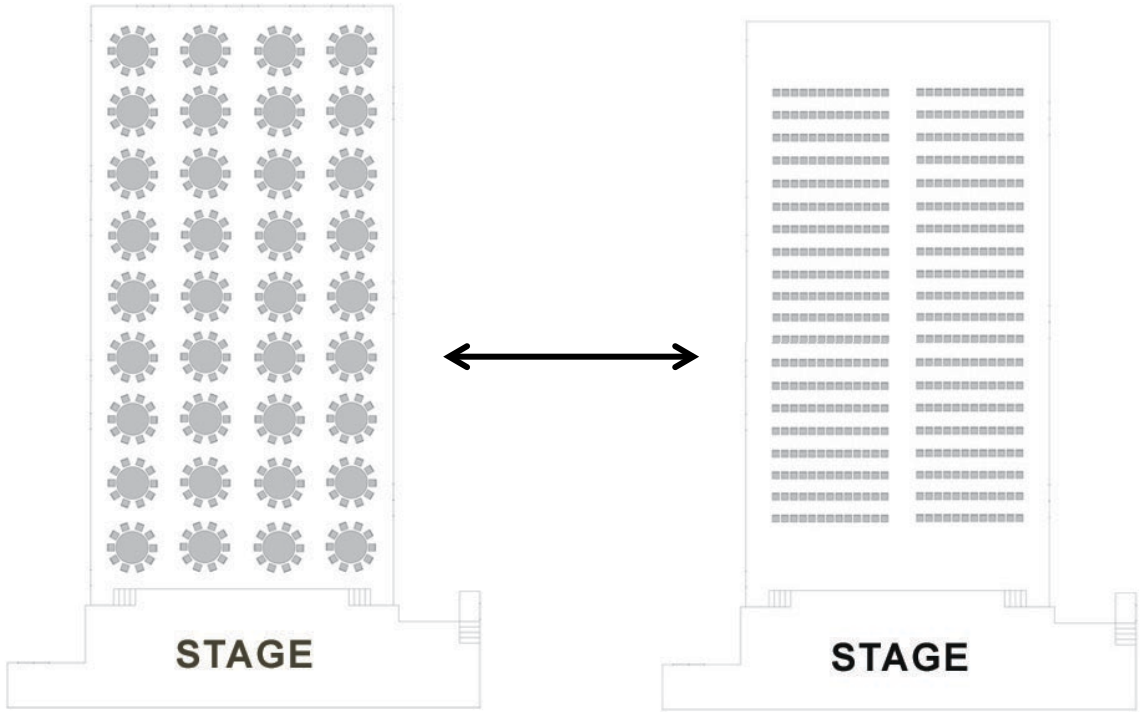
You are designing an Event Hall

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:
Why Partial Reconfiguration?

Shashank Obla

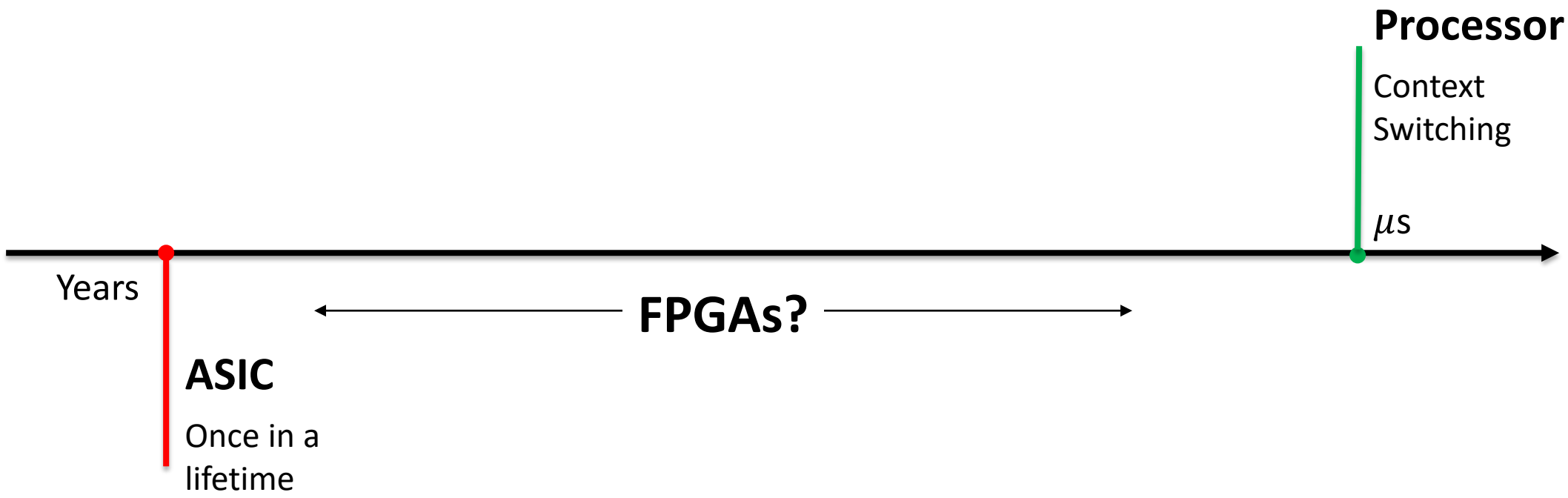
PhD Student, Department of ECE

Carnegie Mellon University

Housekeeping

- Your goal today: appreciate FPGAs as truly programmable and dynamic devices; also, why you should use DFX in Lab 3
- 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)
  - 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

Bitstream defines the chip

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

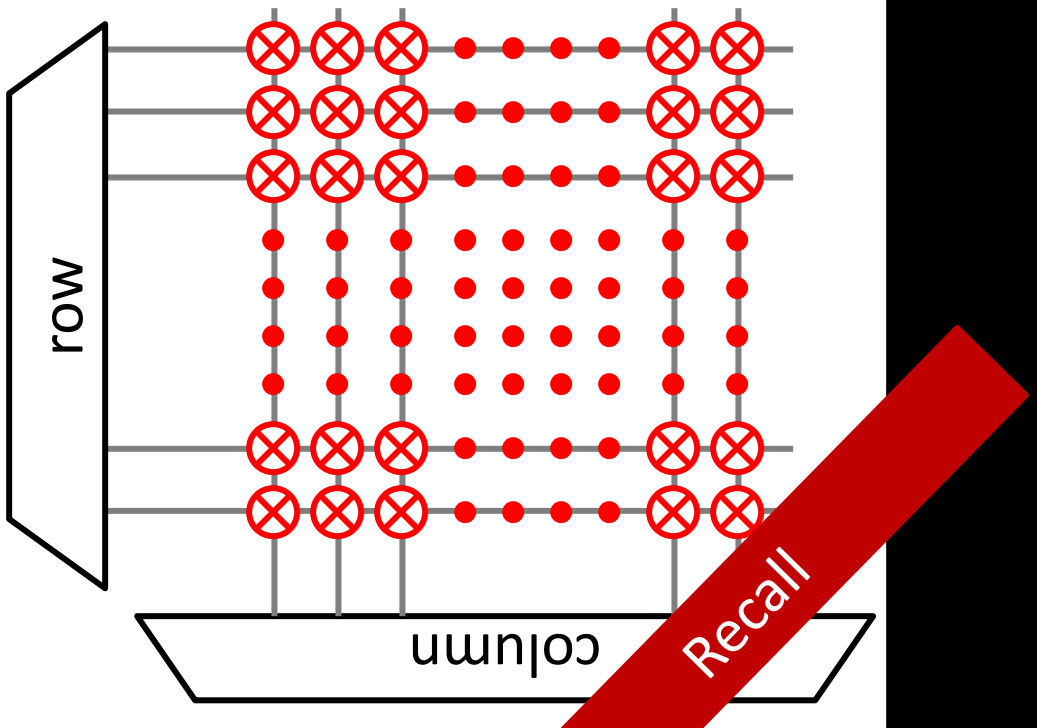
  a bonus "feature" for sensitive devices that need to forget what it does

- 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

Recall

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



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…

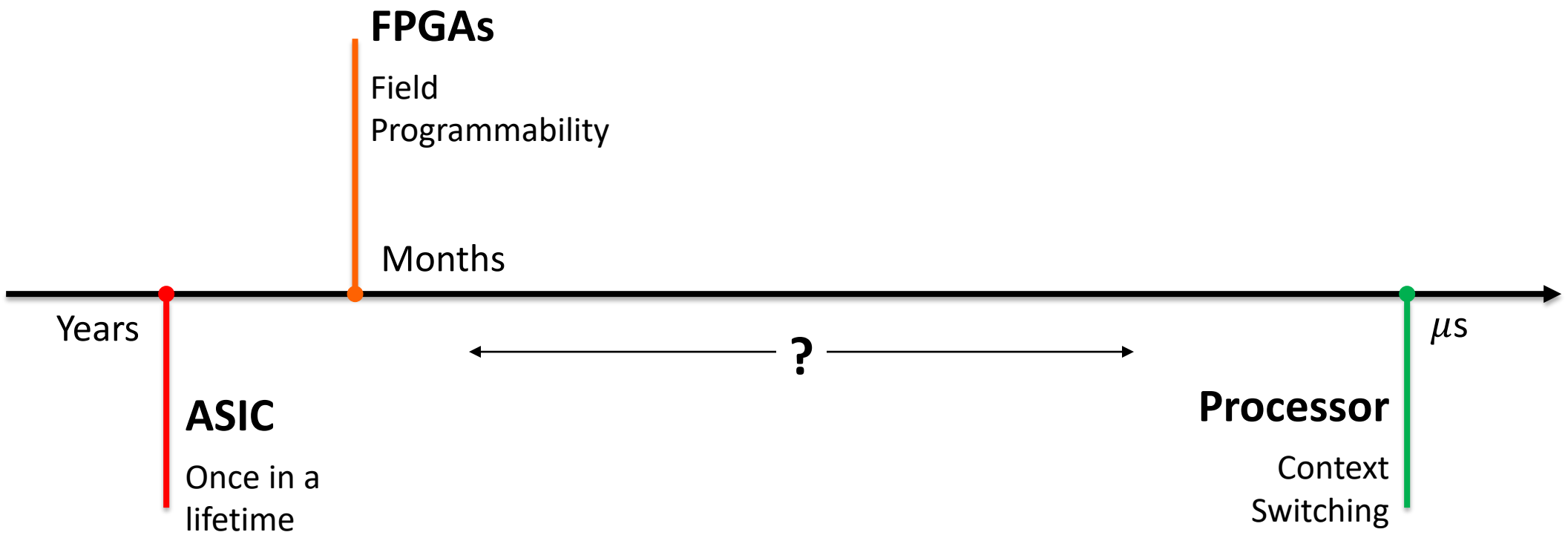
Event Planning

We could have multiple events in a day, but the downtime is now unacceptable

- Need to bring in food for the event
- Set up the stage if required
- Scout the room beforehand
- Does the room even exist?

Hall Design

Back to the Programmability Timescale



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

DYNAMIC PARTIAL RECONFIGURATION

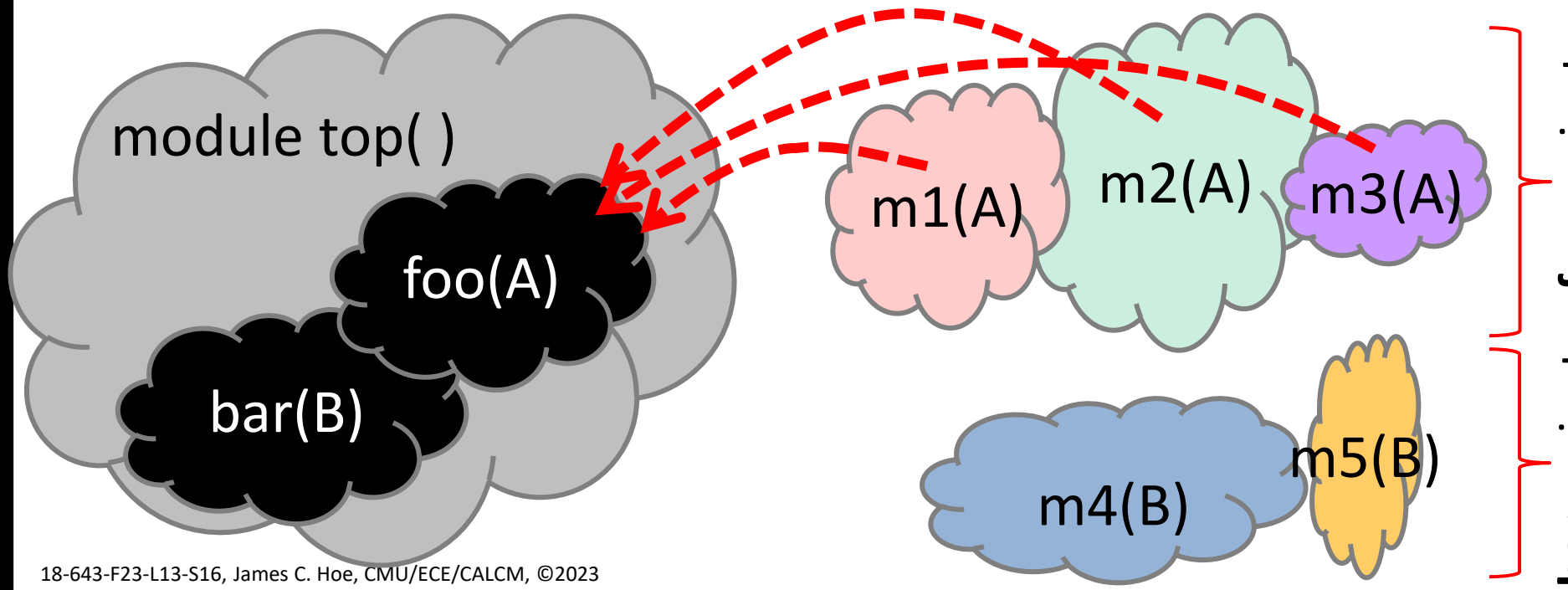
Partial Reconfiguration (PR)

- 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

Recall

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**



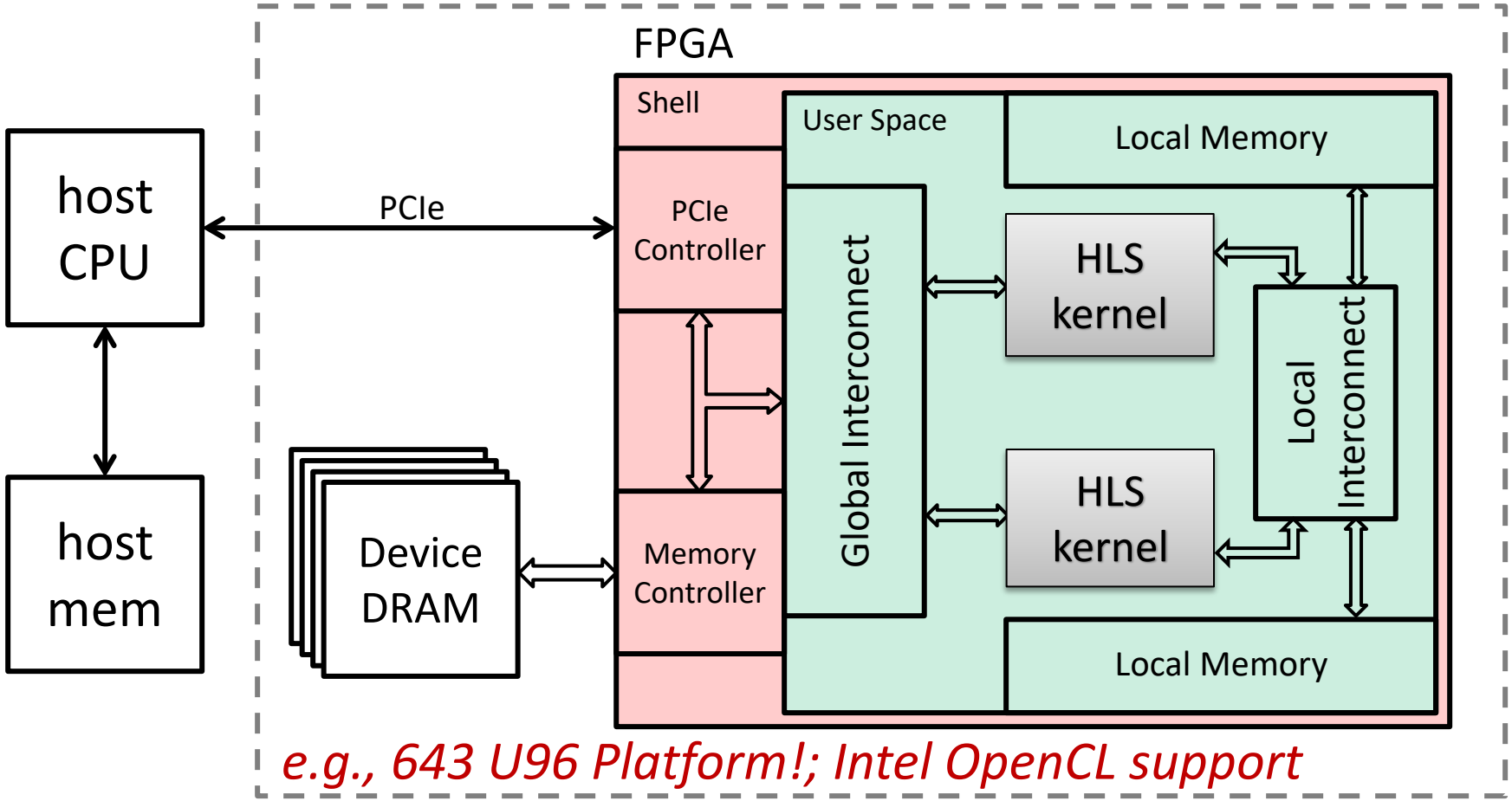
Have you used PR before?

LET'S CONNECT THESE CONCEPTS BACK TO REALITY – WHERE HAVE WE SEEN THIS BEFORE?

Imagine a world where 643 Labs didn't use Partial Reconfiguration (PR)

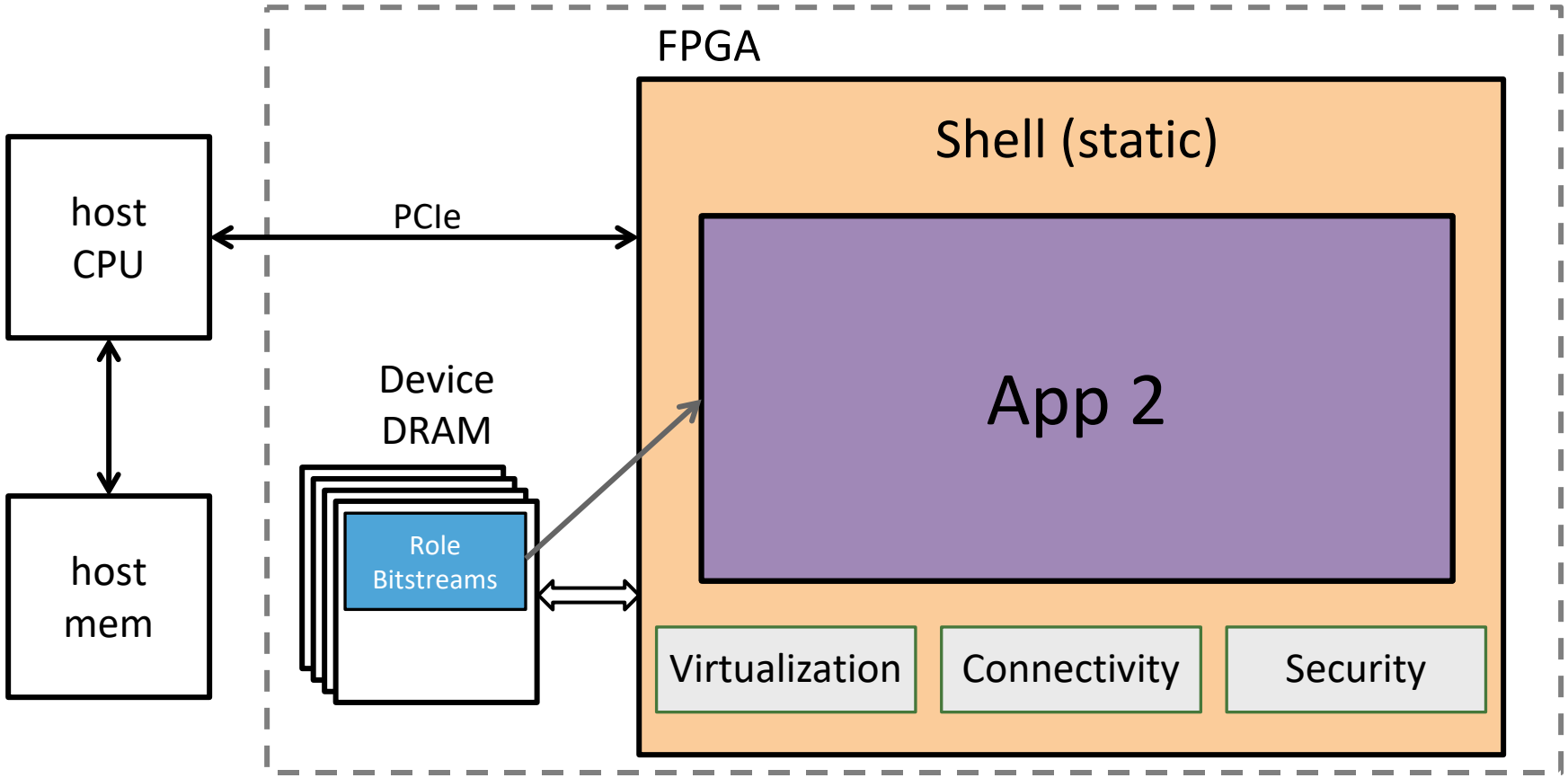
- You used PR in Lab 1 already!
- Until F2021 we used the FPGA like an ASIC (u96v2_sbc_full)
- Loading a new FPGA bitstream was an ordeal
  - Copy new bitstream into the SD Card
  - *Reboot the Ultra96* to load the bitstream
  - Sometimes that didn't work and had to reflash the SD Card ☹
- ARM system and the DRAM cannot handle the FPGA undergo a full change while they were active – need to protect the system

Role-and-Shell PR Usage



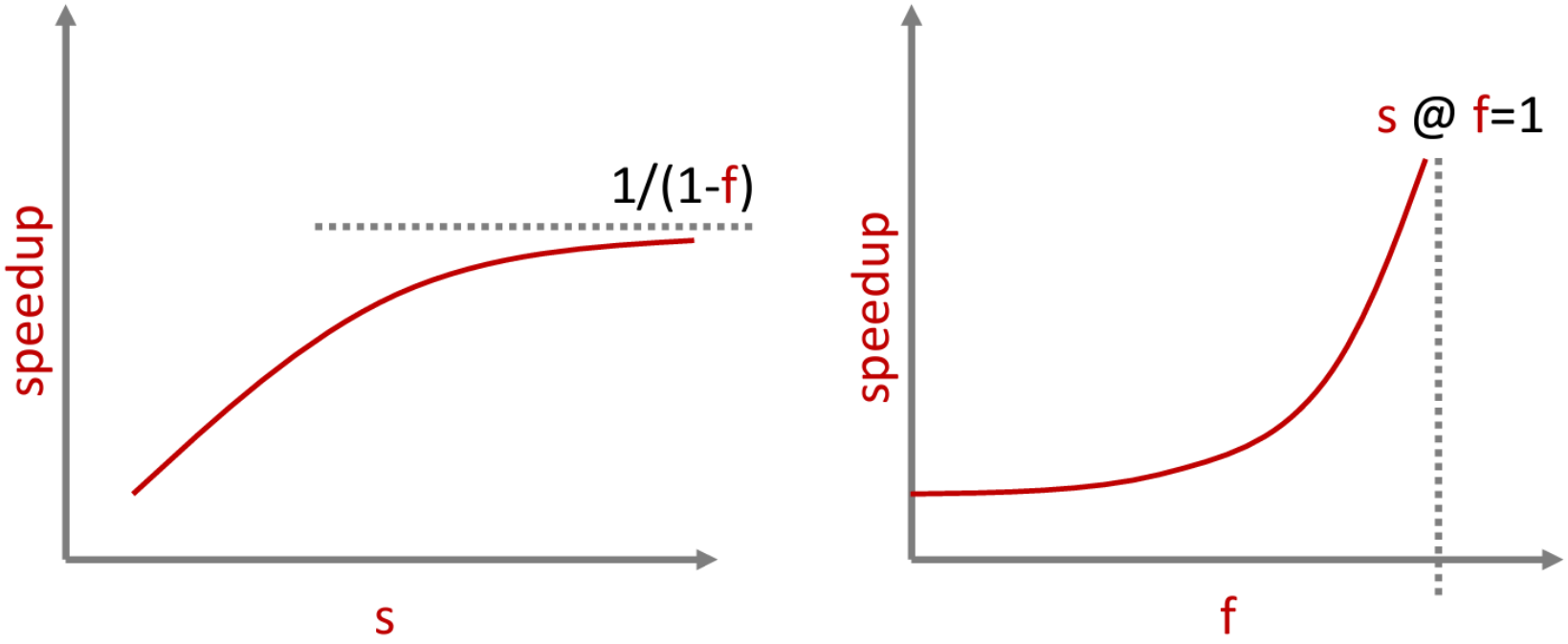
- virtualize and simplify surrounding
- enforce security and QoS
- keep system alive

Time-Sharing with PR



FPGA as an accelerator!

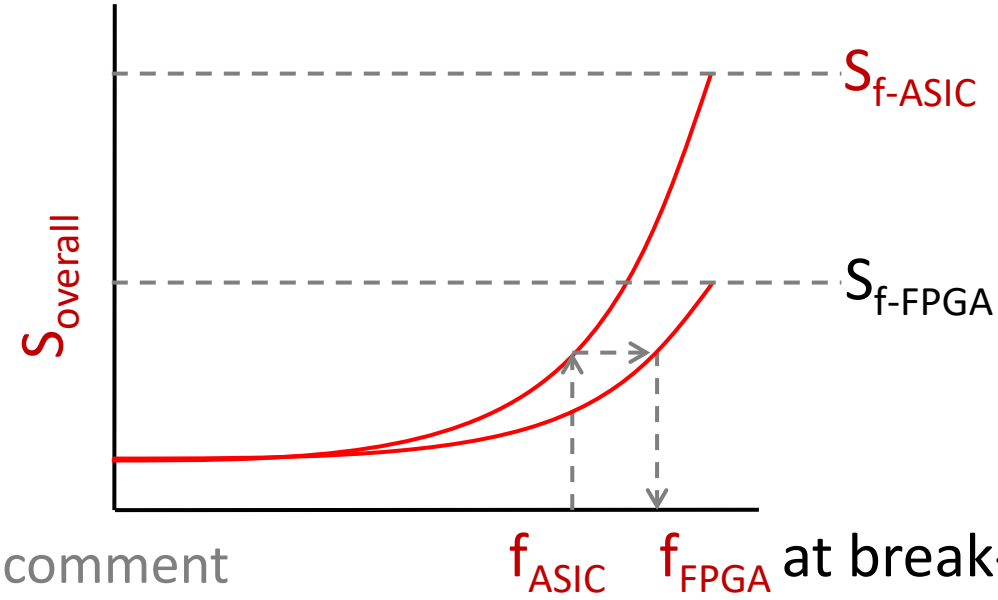
Amdahl's Law



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

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]

Lab 3 Out-of-the-Box is better with PR!

```
void krnl_cnn_layerX(const cnndata_t* input, const cnndata_t* weights,
        cnndata_t* output, uint64_t batch_size, uint64_t r_ofm, uint64_t c_ofm,
        uint64_t m_ofm, uint64_t n_ifm);
```

VS

```
void krnl_cnn_layer0(const cnndata_t* input, const cnndata_t* weights,
        cnndata_t* output, uint64_t batch_size);
void krnl_cnn_layer1(const cnndata_t* input, const cnndata_t* weights,
        cnndata_t* output, uint64_t batch_size);
```

All have the same kernel tiling sizes, but the hardcoded layer dimensions make the difference – HLS can optimize loop bounds for each layer – overcomes the overhead of PR Time

Is designing two kernels vs one harder?

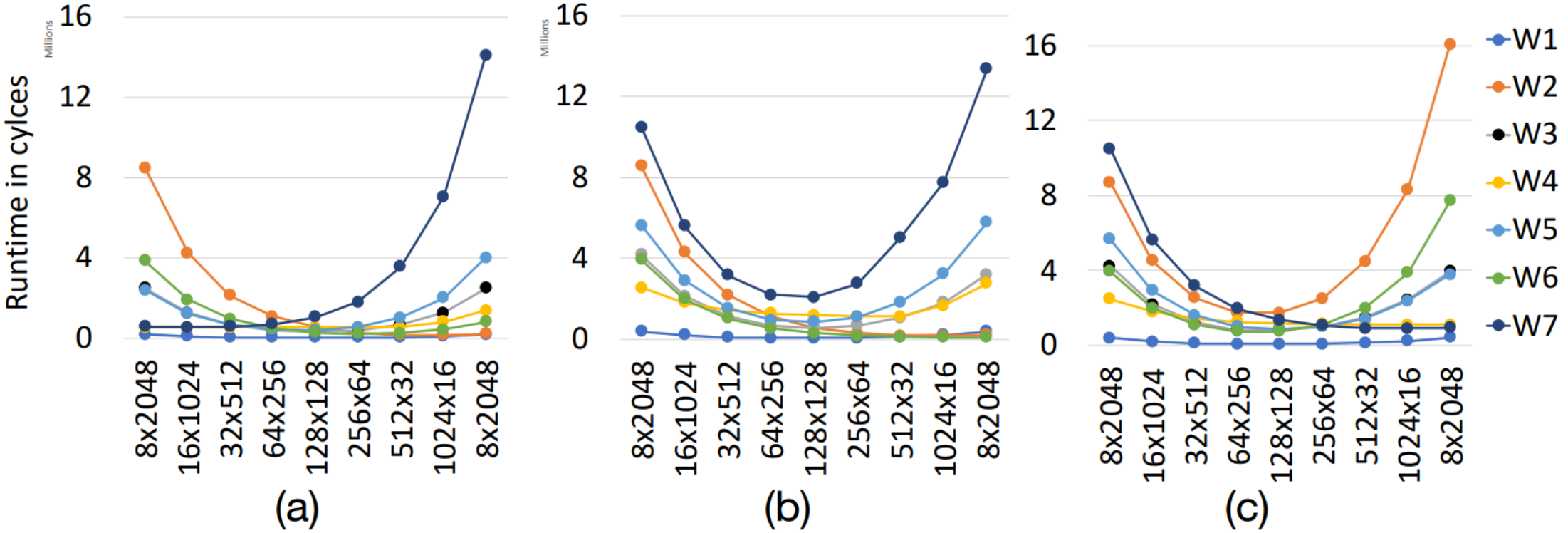
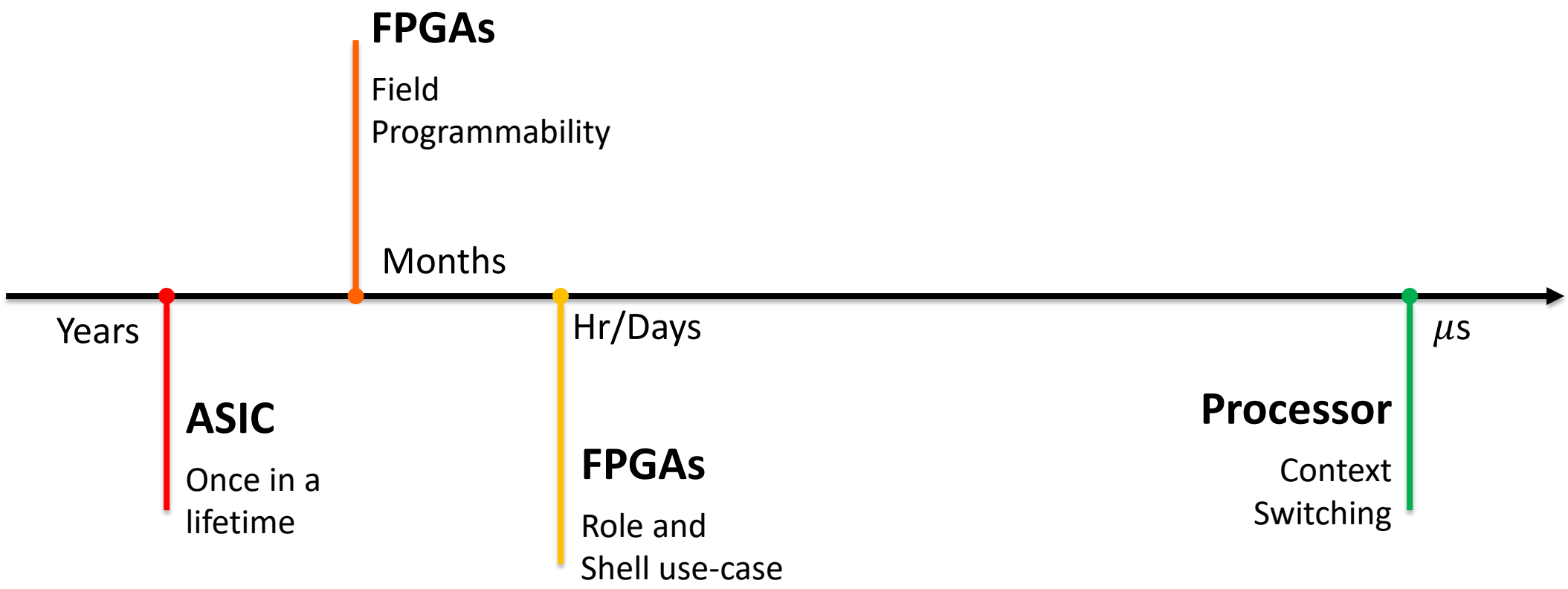


Chart showing the variation of runtime for workloads run with different shapes of systolic array with fixed number of PEs (=16384) for three dataflows (a) Output Stationary, (b) Weight Stationary, (c) Input Stationary

Tradeoff 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*.

Back to the Programmability Timescale



How often do we change the setting?

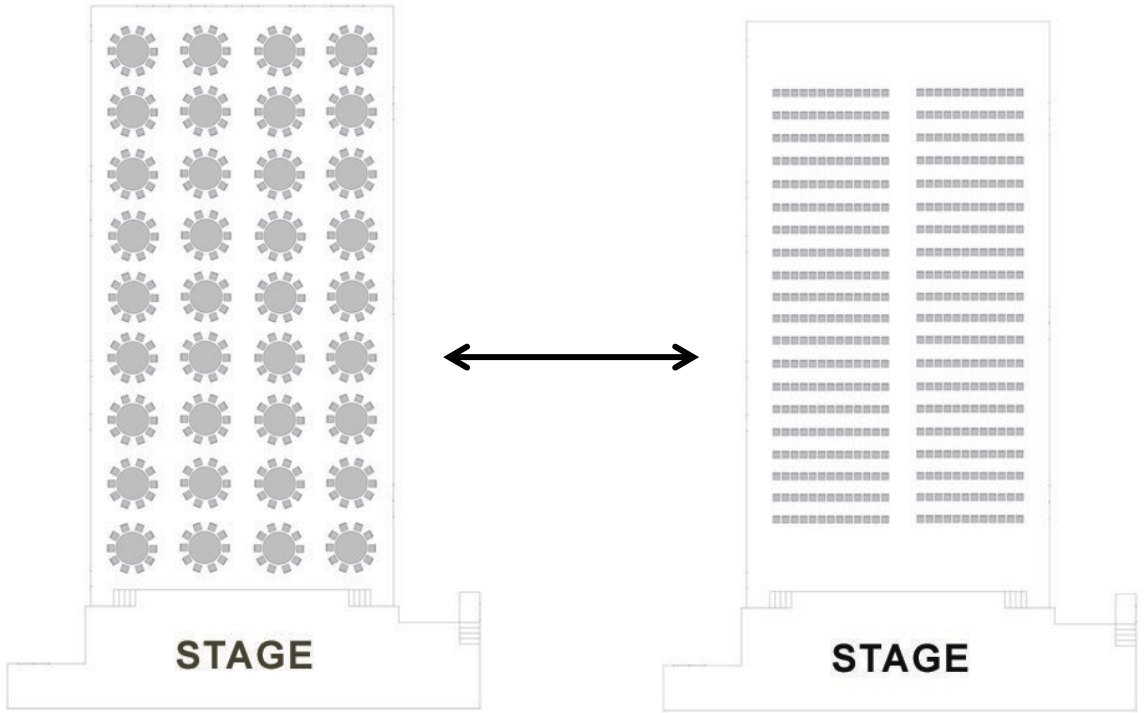
Event Planning

Specification

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, unless you have the same type of event in a day



Reconfiguration takes a couple of hours…

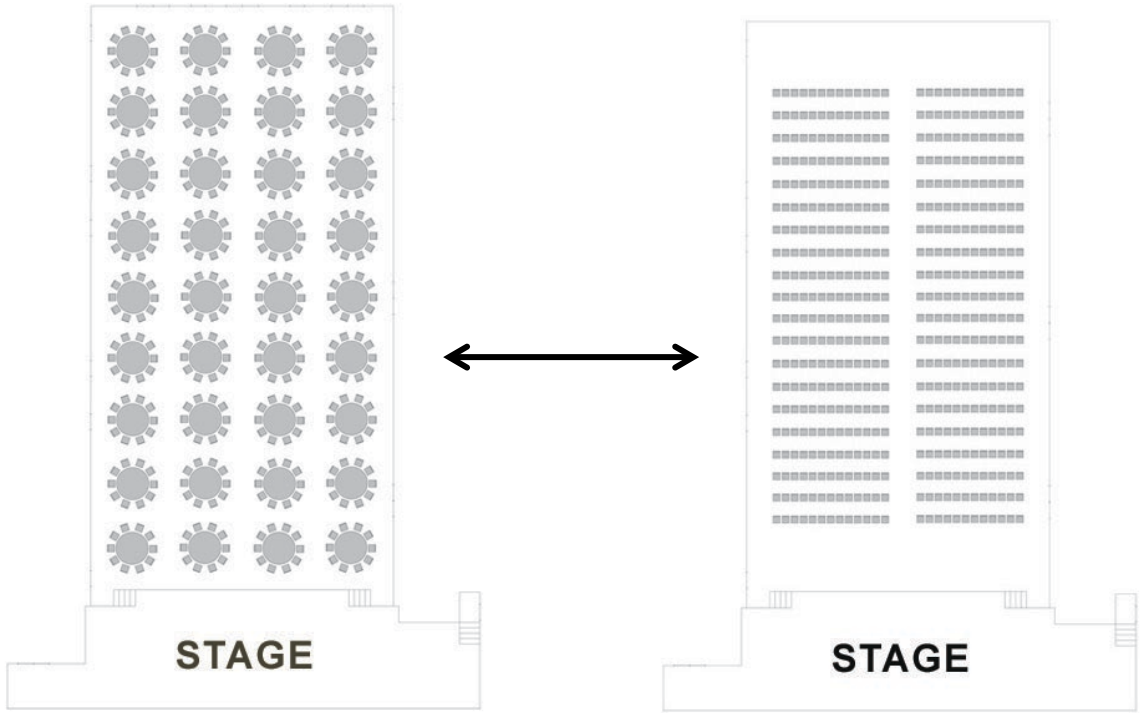
Relative Time Taken Matters!

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

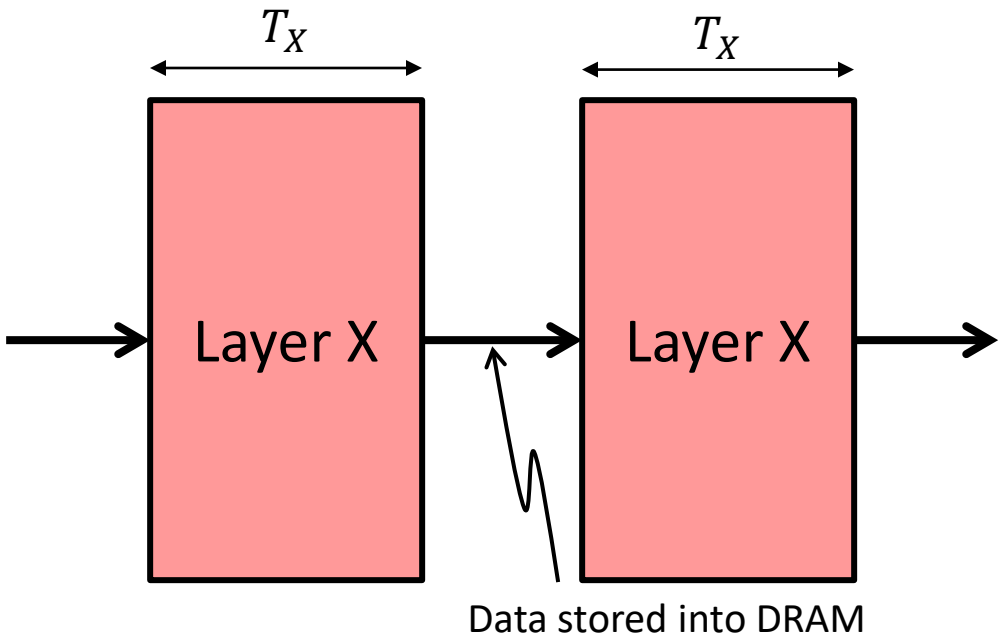
Specification



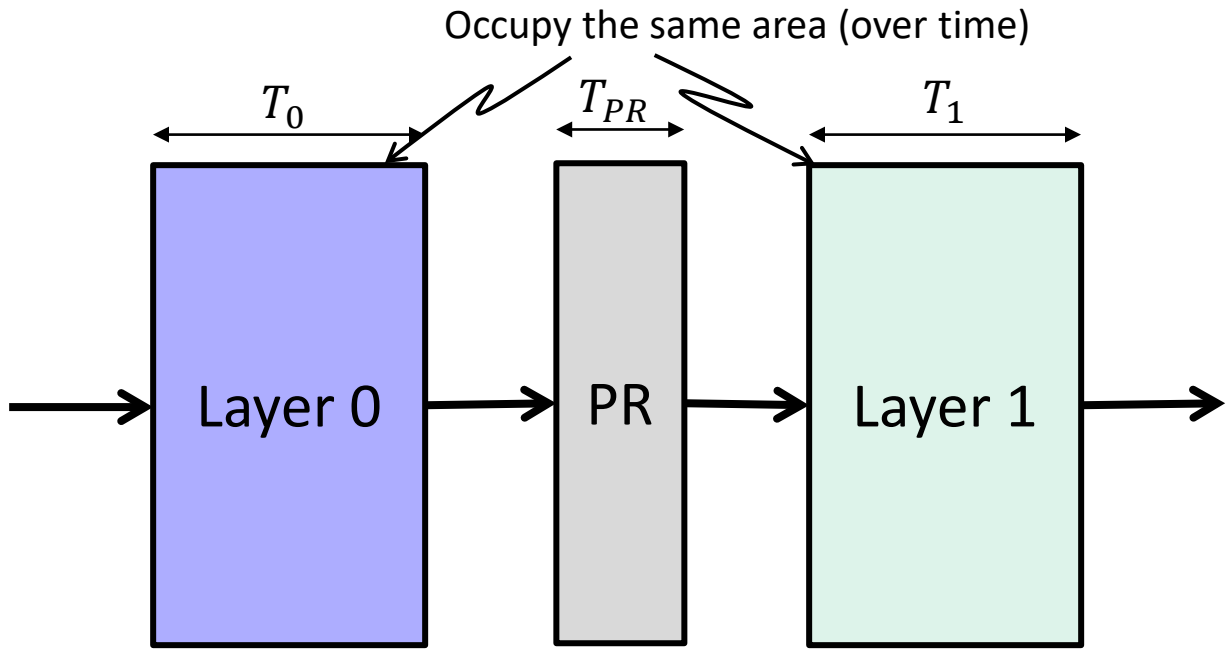
Reconfiguration takes 15 minutes…

Lab 3 – Analytically Speaking

// #define ENABLE_DFX



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



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

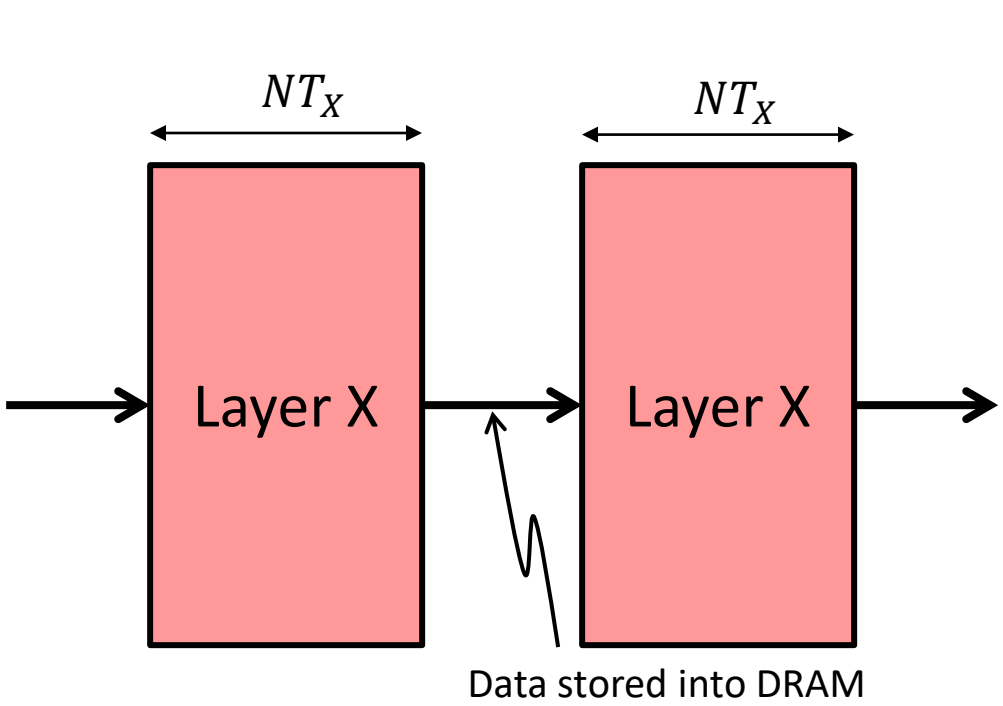
#define ENABLE_DFX

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

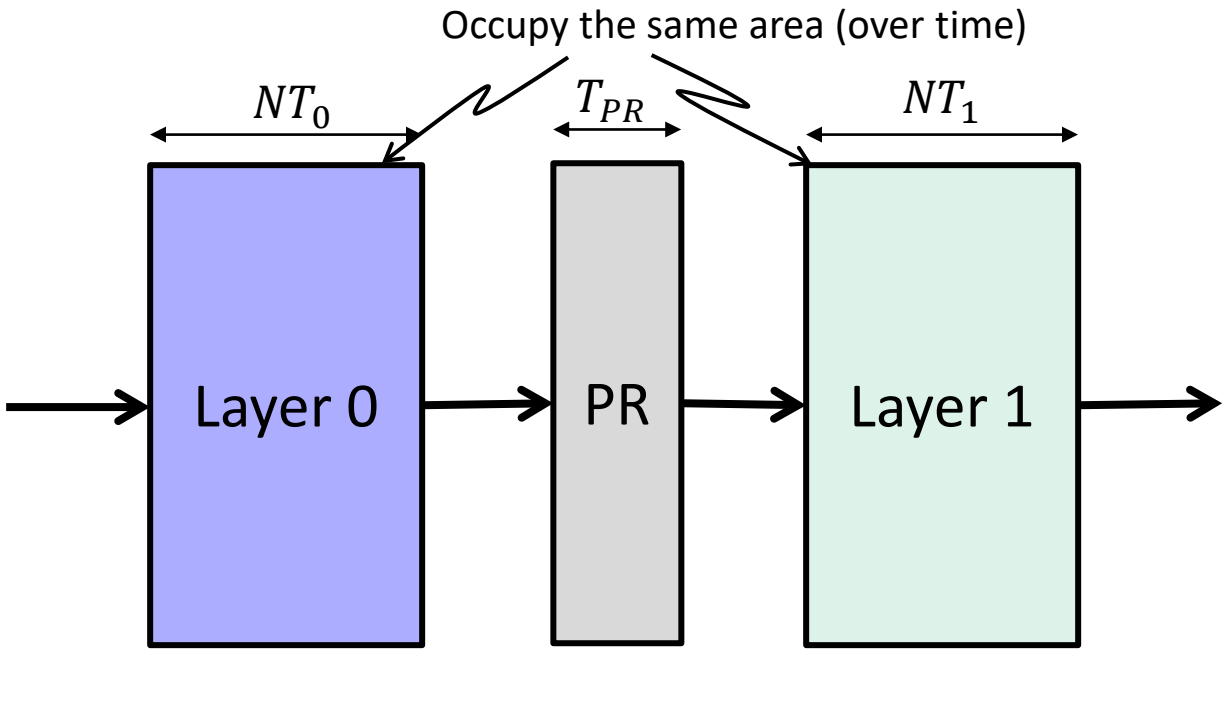
PR is not free…

Analytically Speaking (Batching)

// #define ENABLE_DFX



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



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

#define ENABLE_DFX

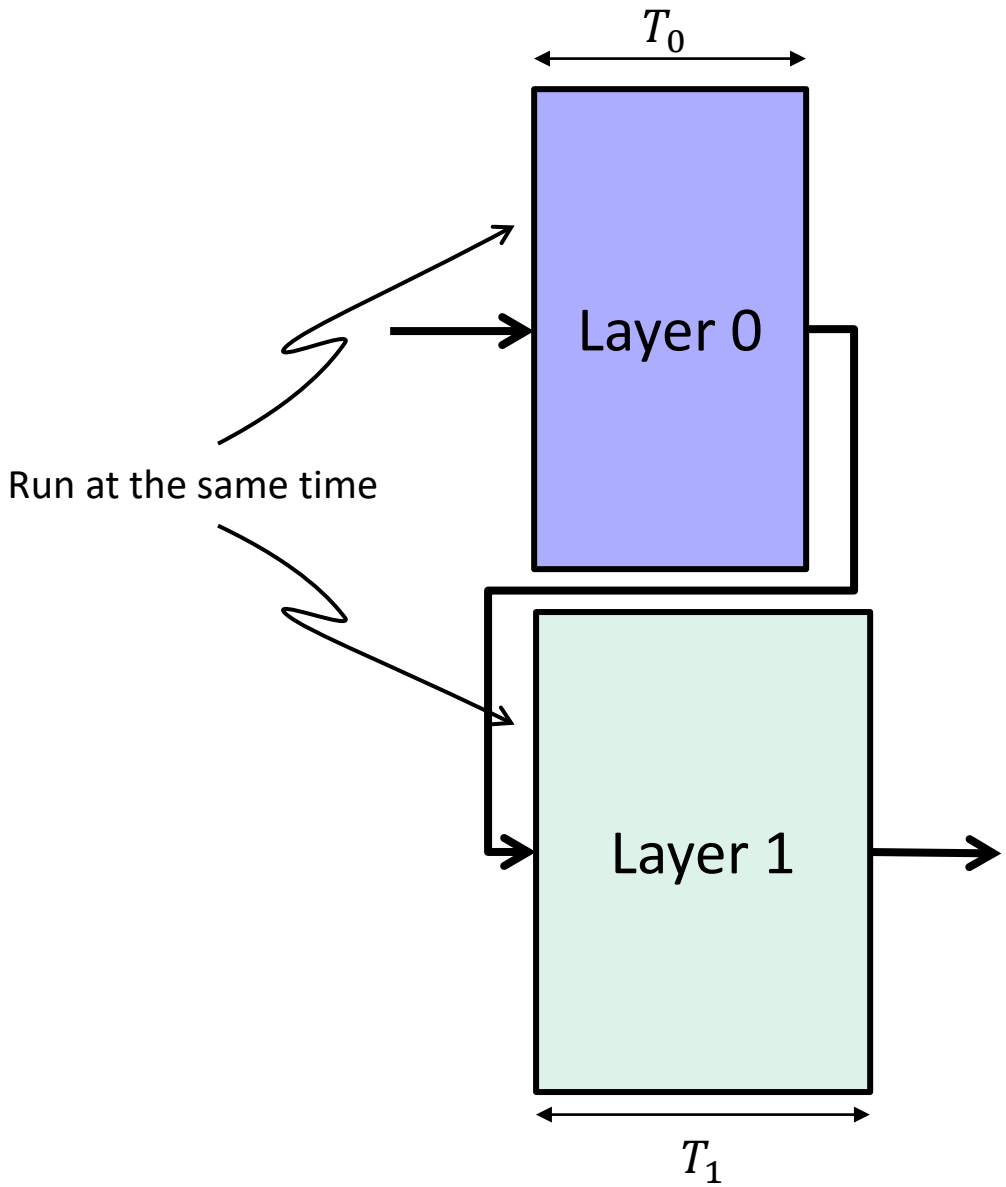
$$2T_X >^{?} T_0 + T_1 + \frac{T_{PR}}{N}$$

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

Batching is not free…

Beyond Lab 3: Can we have layers side-by-side?

- Custom kernels gain the same benefit as when we used PR
- Can save memory bandwidth by designing the kernels to pipeline
- They must take the same amount of time to allow perfect overlap
- Ultra96 is too small to have two meaningful kernels side by side
- Doesn't scale if we have many more layers and you don't use PR



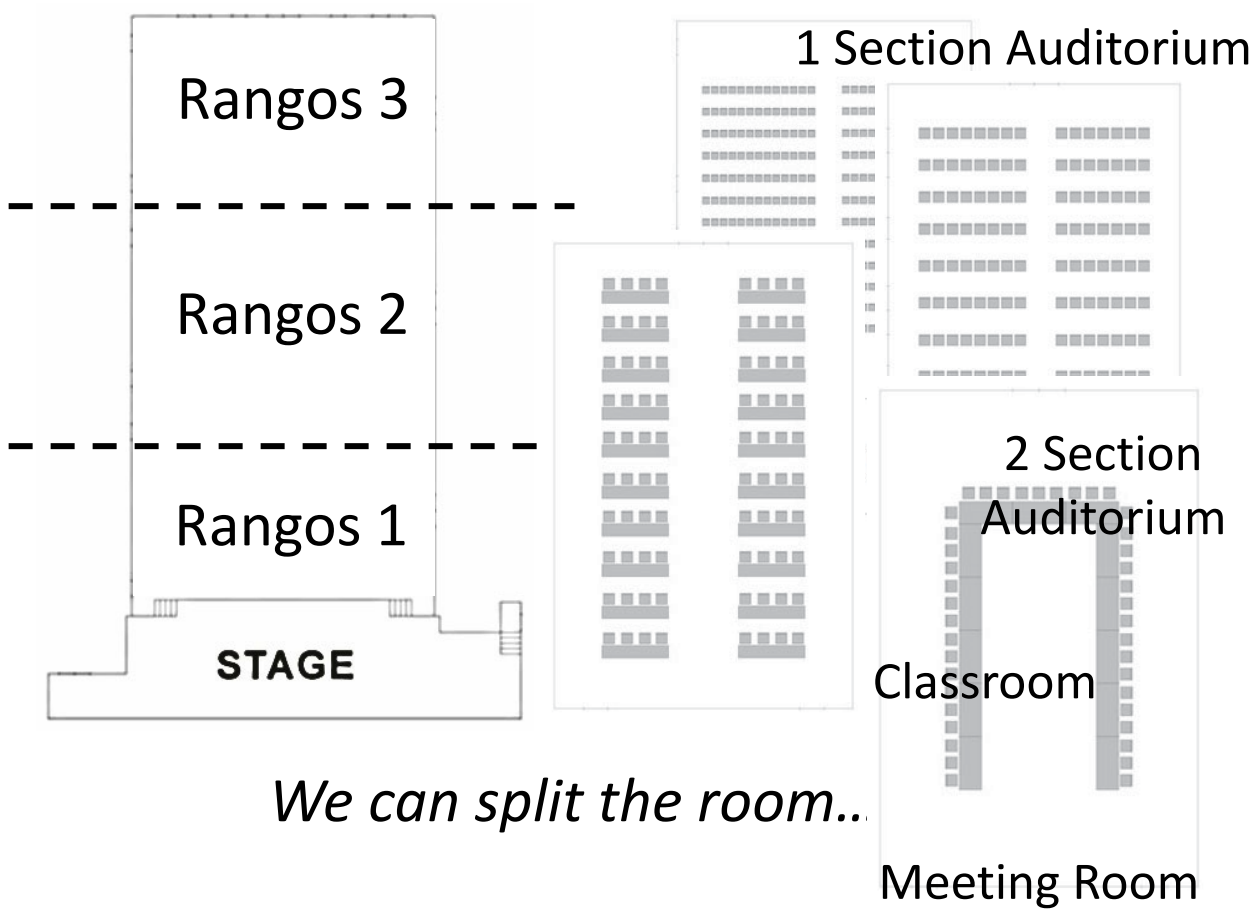
Renting out an Event Venue

Event Planning

What if the venue is very big, too big for certain events?

- Smaller events like a department PhD orientation or a small lecture waste a lot of space
- Cannot prepare the rest of the venue for the next event

Specification



We can split the room..

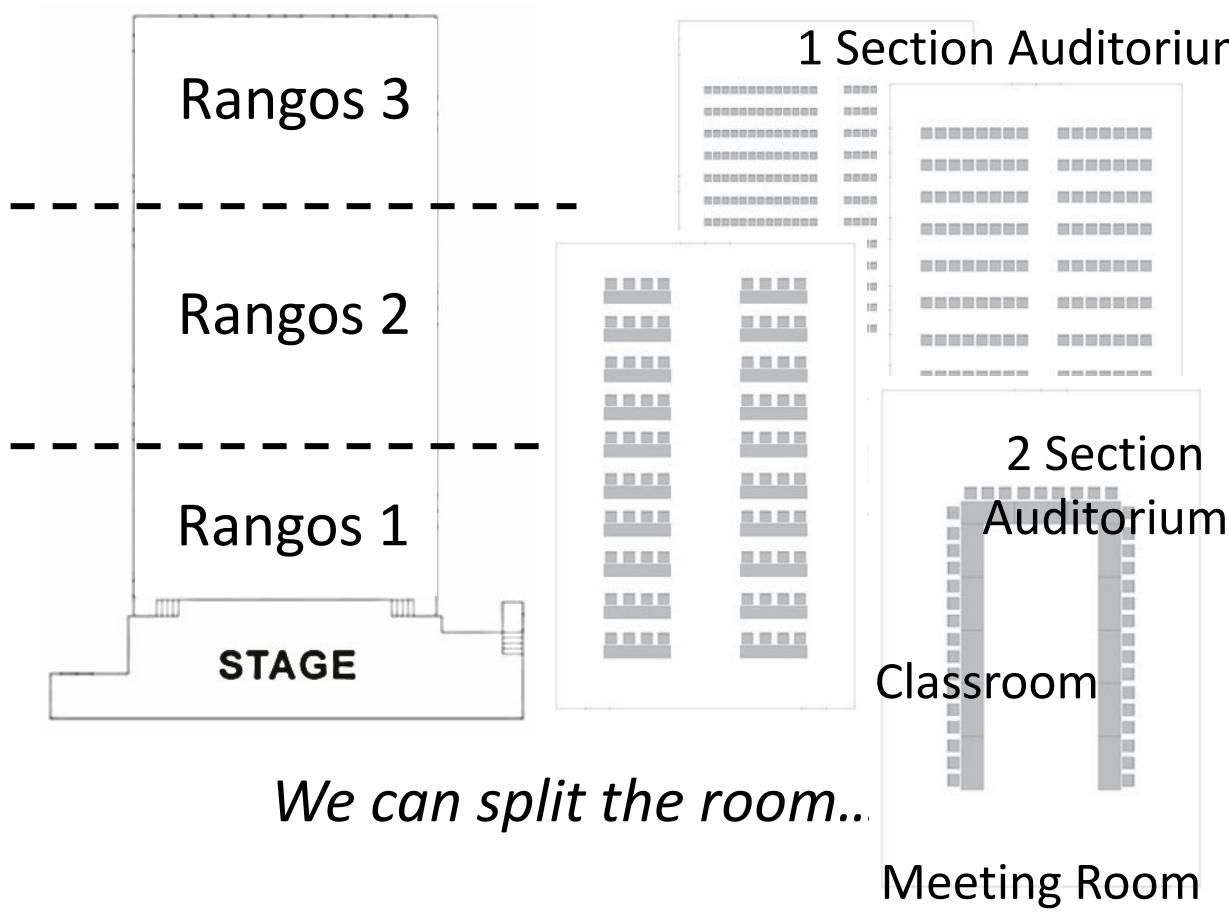
Renting out an Event Venue

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

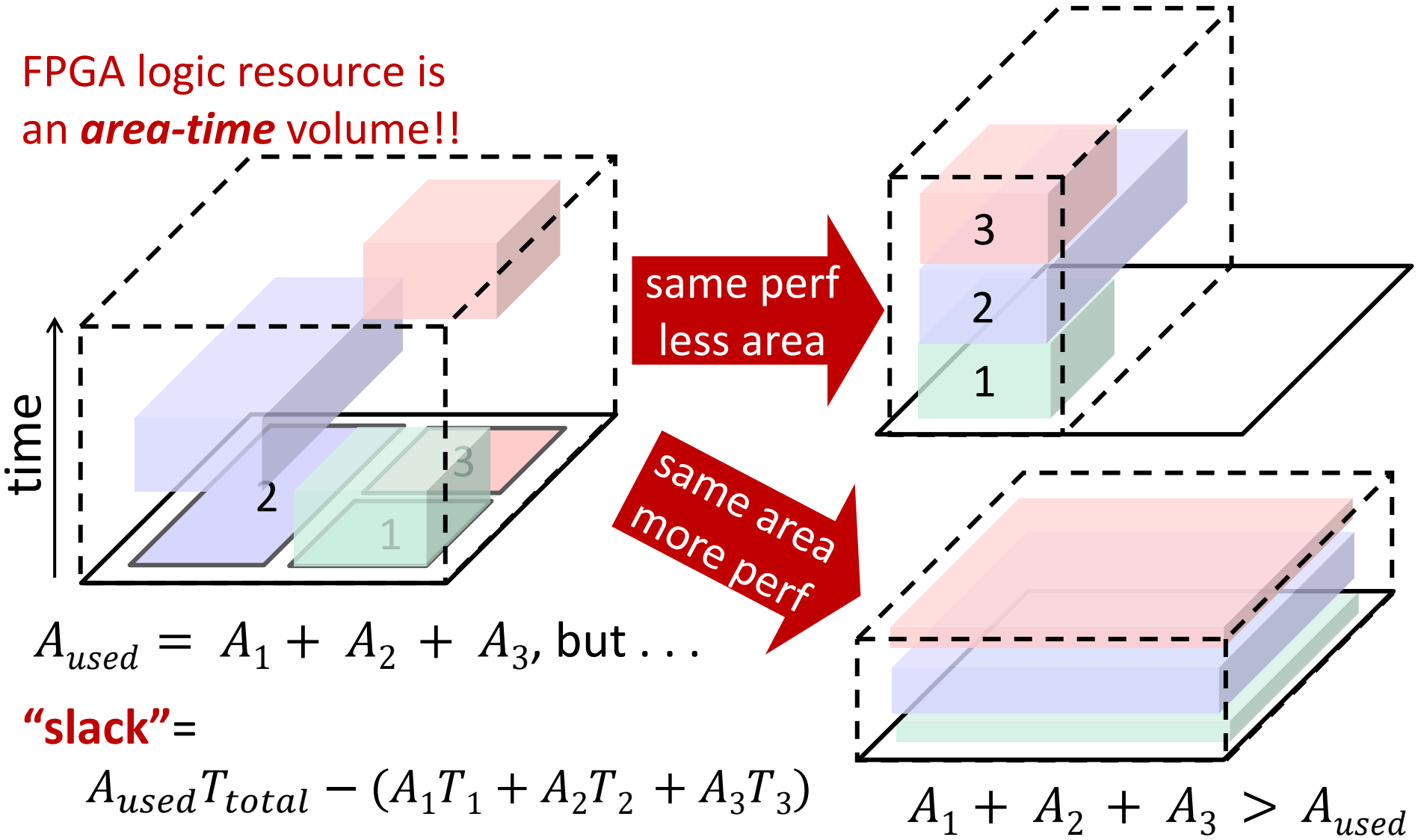
Specification



We can split the room..

PR FPGA Designs Different from ASIC

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



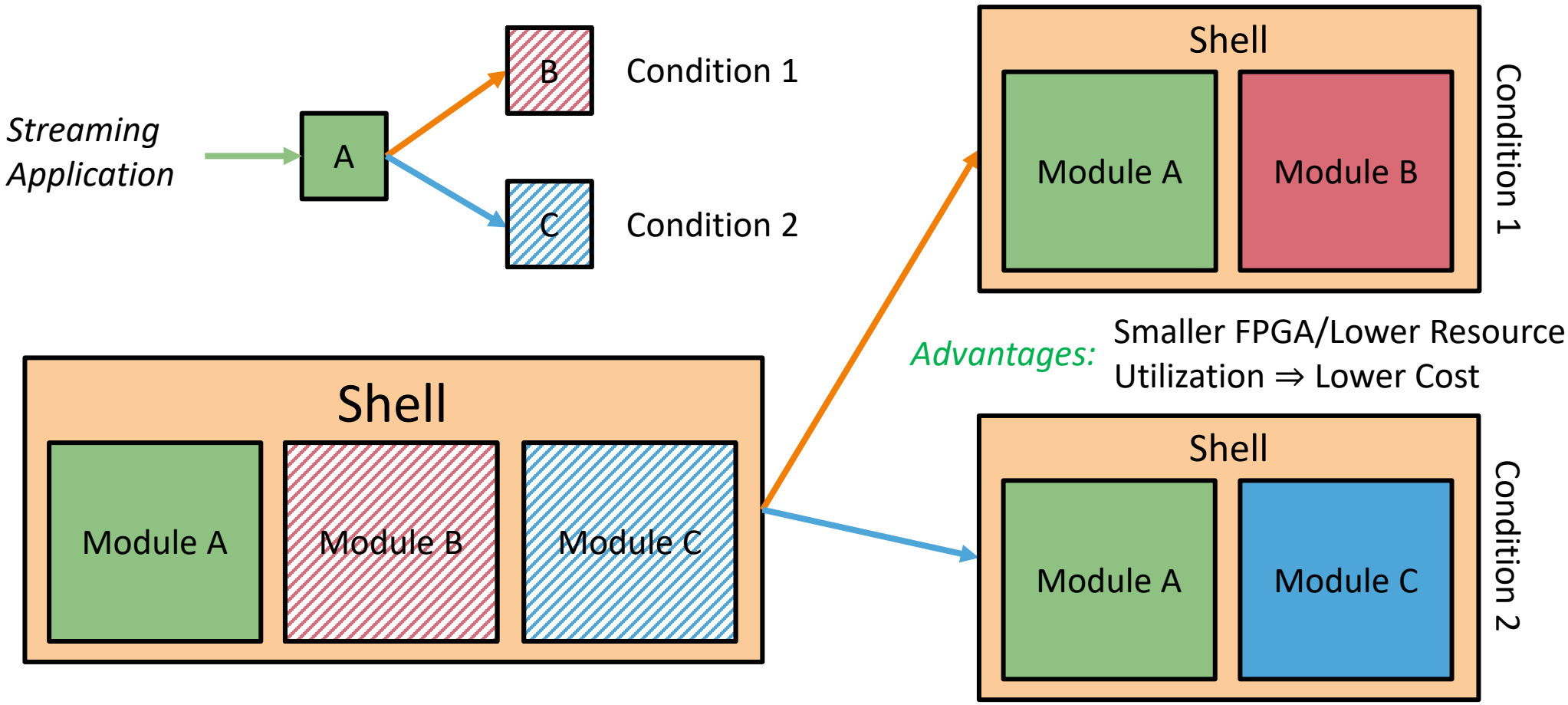
$A_{used} = A_1 + A_2 + A_3$, but . . .

"slack"=

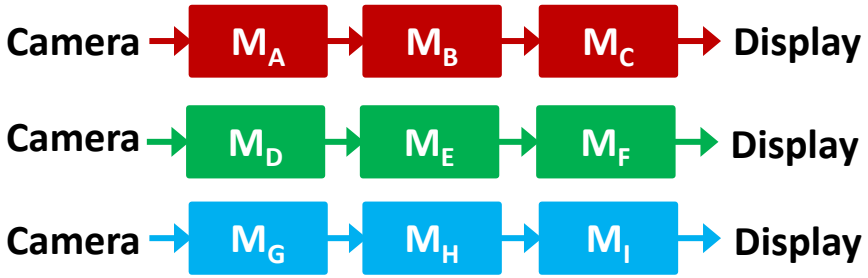
$$A_{used}T_{total} - (A_1T_1 + A_2T_2 + A_3T_3)$$

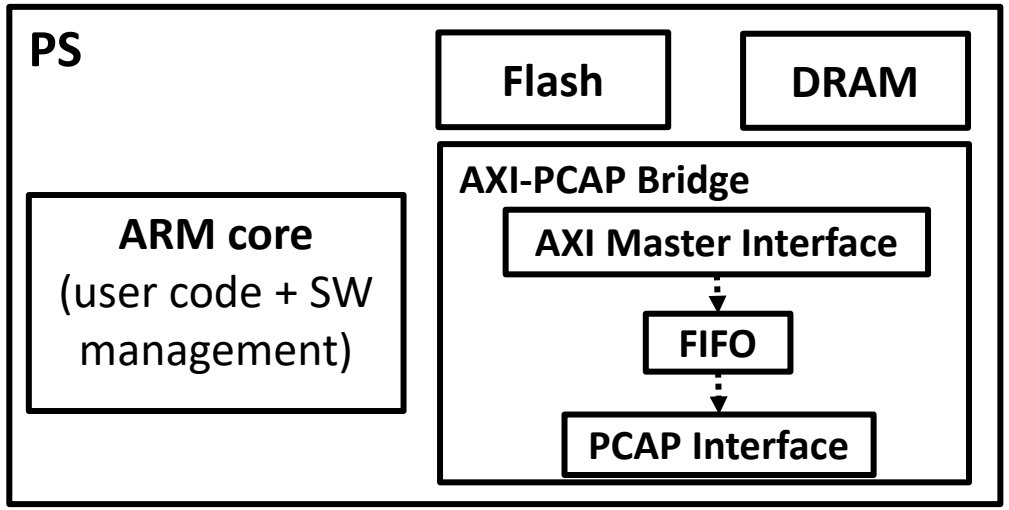
$$A_1 + A_2 + A_3 > A_{used}$$

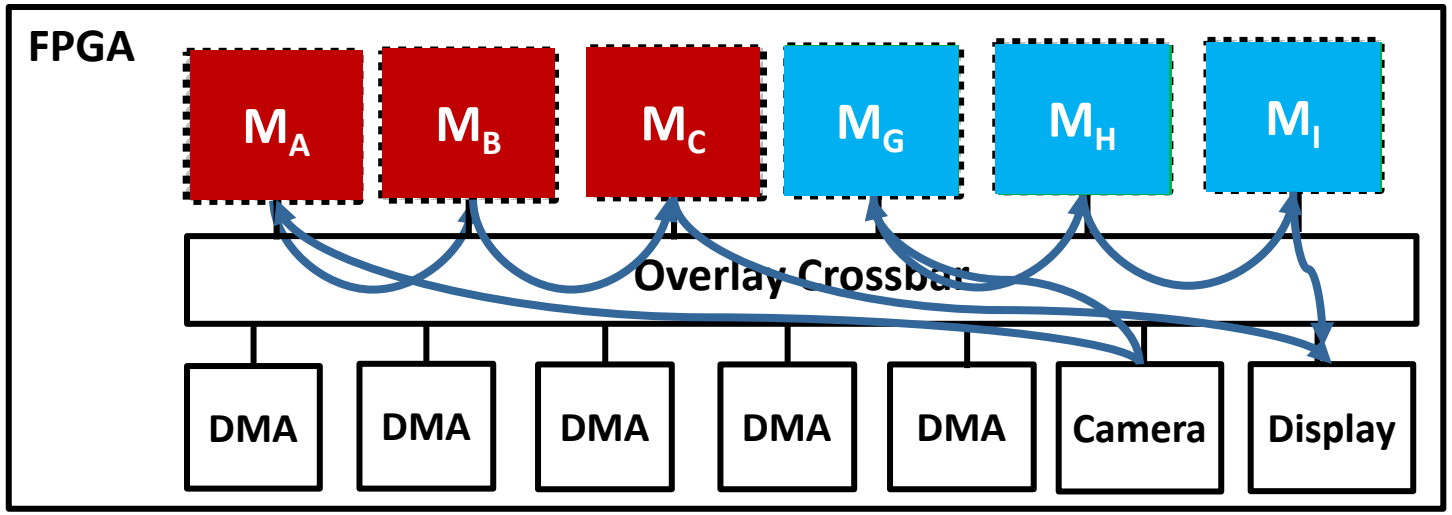
Exploiting Design Slack at Runtime



Spatial and Temporal Multitenancy







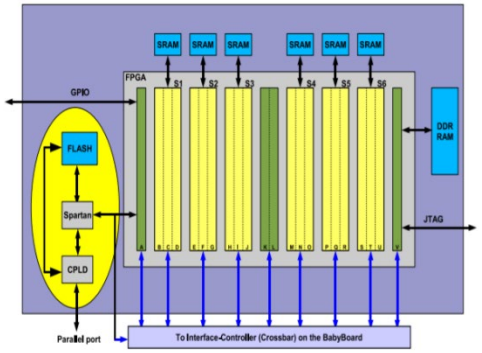
https://users.ece.cmu.edu/~jhoe/distribution/2018/fpl2018.pdf

SEEMS OBVIOUS TO USE PR THIS WAY…

Field Programmable to Programmable

- 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
- Programmability is a very costly feature
- 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 been around for a long time



Bobda et al. [FCCM 2005]

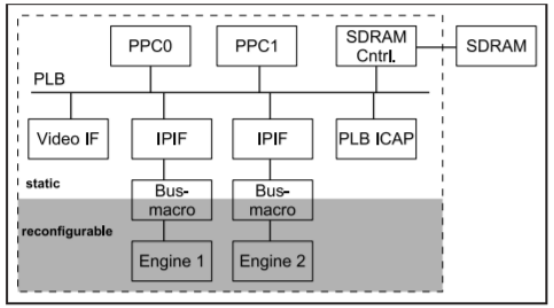
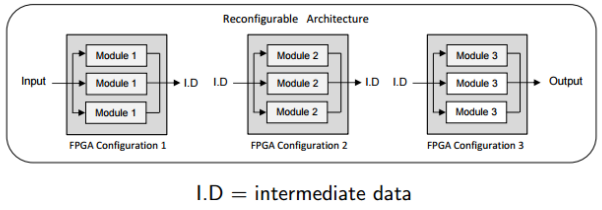


Figure 1 Block diagram of the Autovision system.

Claus et al. [2007]



Arram et al. [FPGA 2015]

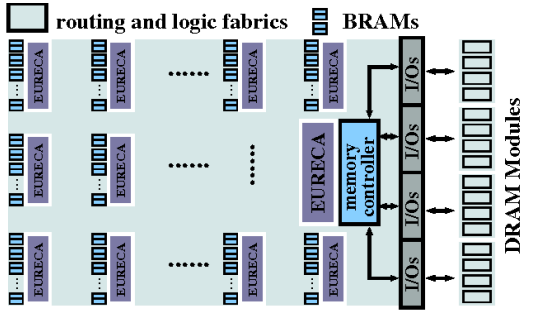
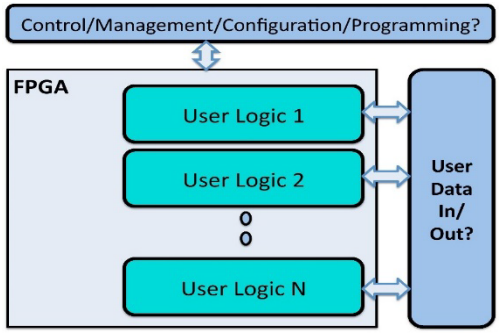


Figure 3: EURECA memory architecture overview.

Niu et al. [FPGA 2015]

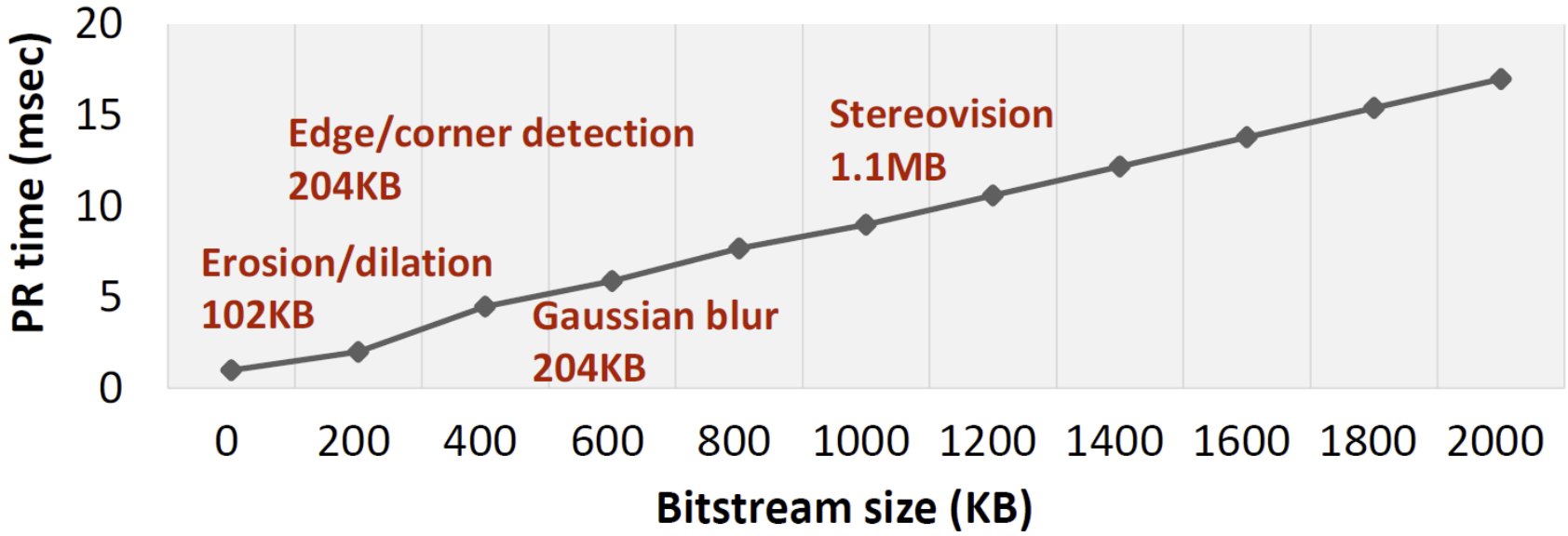


Stuart et al. [FPGA 2015]

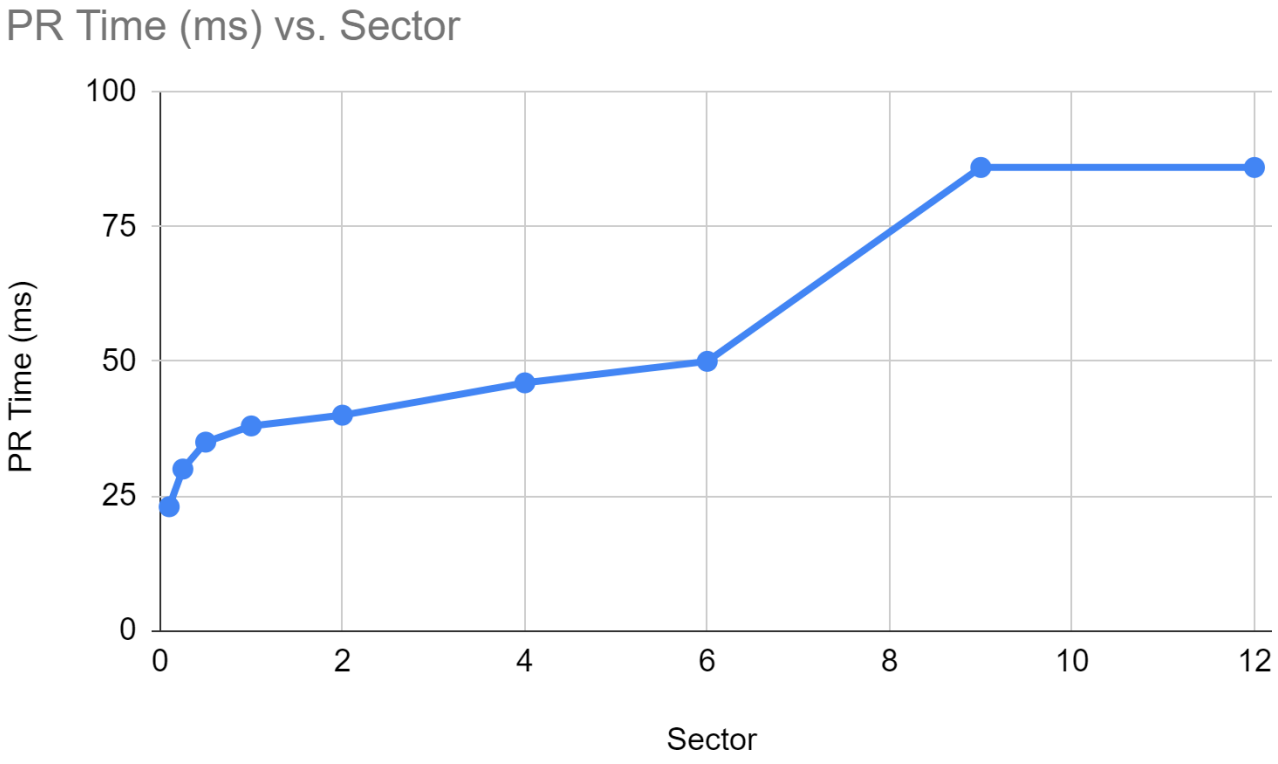
Why isn't it used?

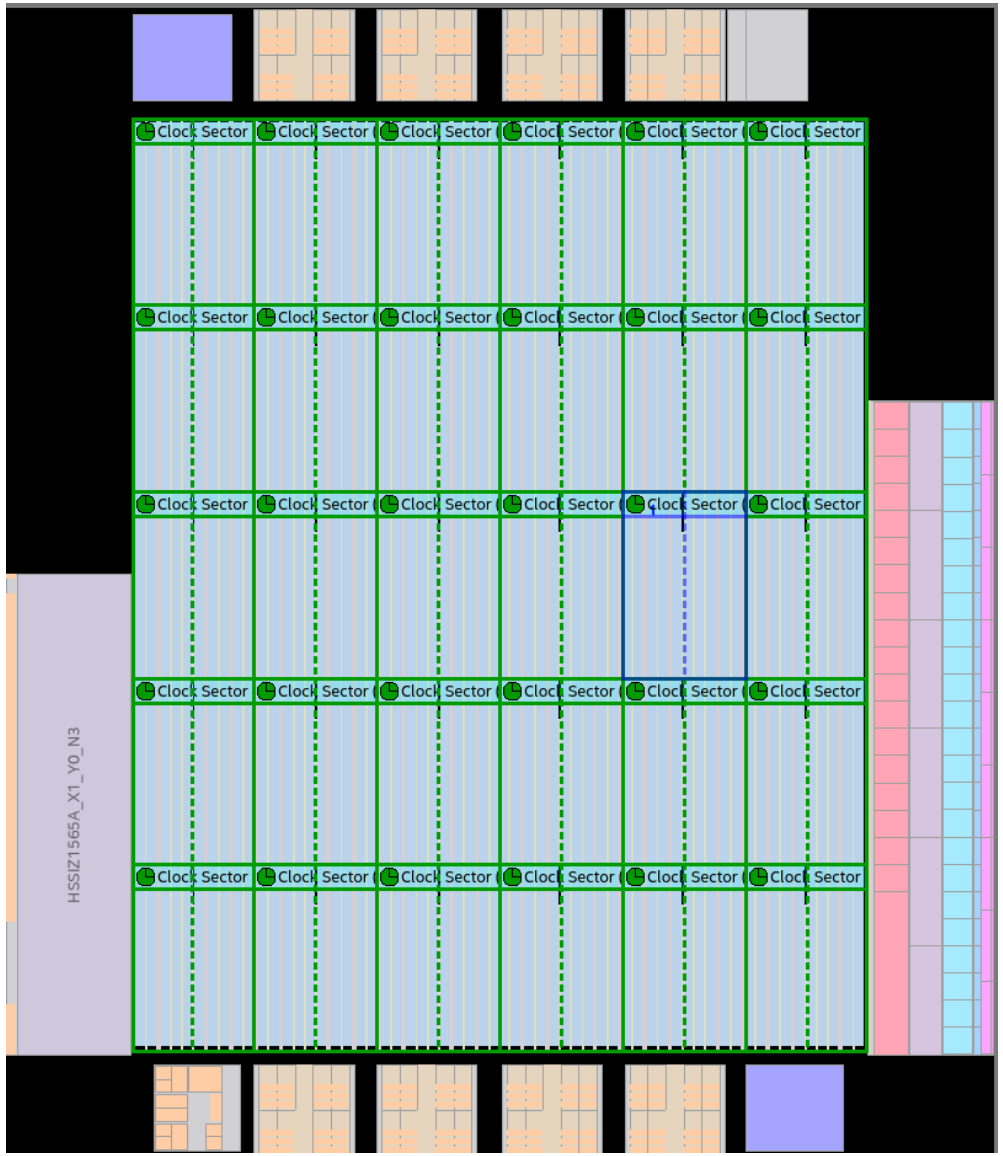
Today's (?) PR Overhead

- 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



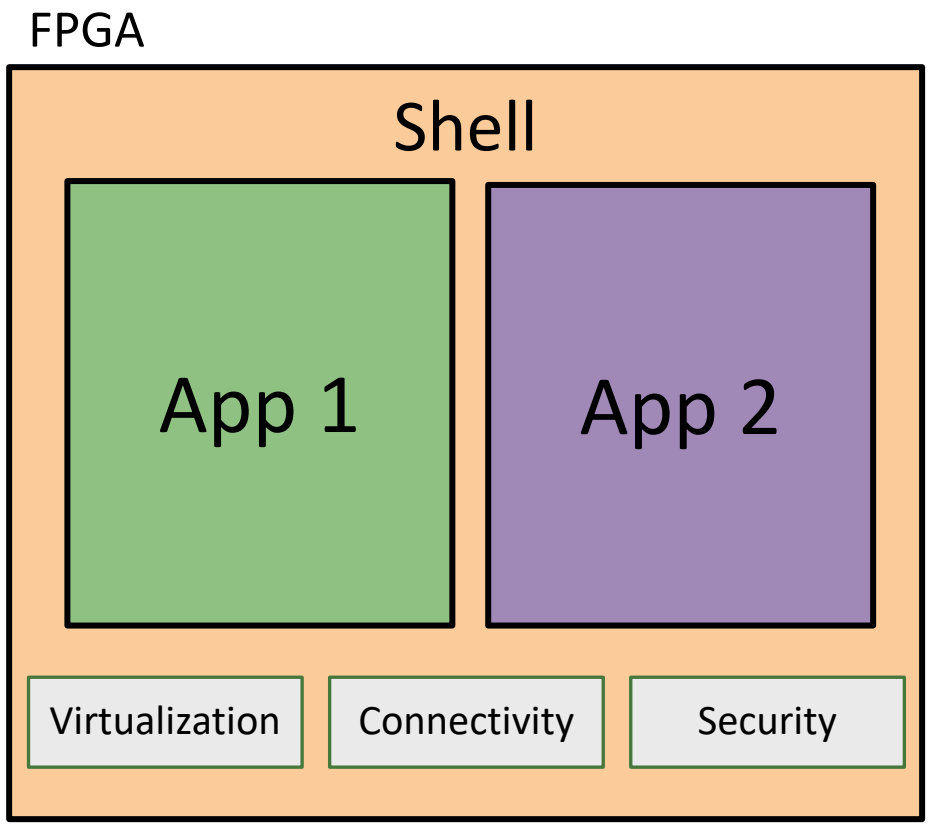
Today's PR Overhead (Intel Agilex 7)





For reference, a sector has a 25% higher DSP FLOP/cycle capability than the entire Ultra96v2

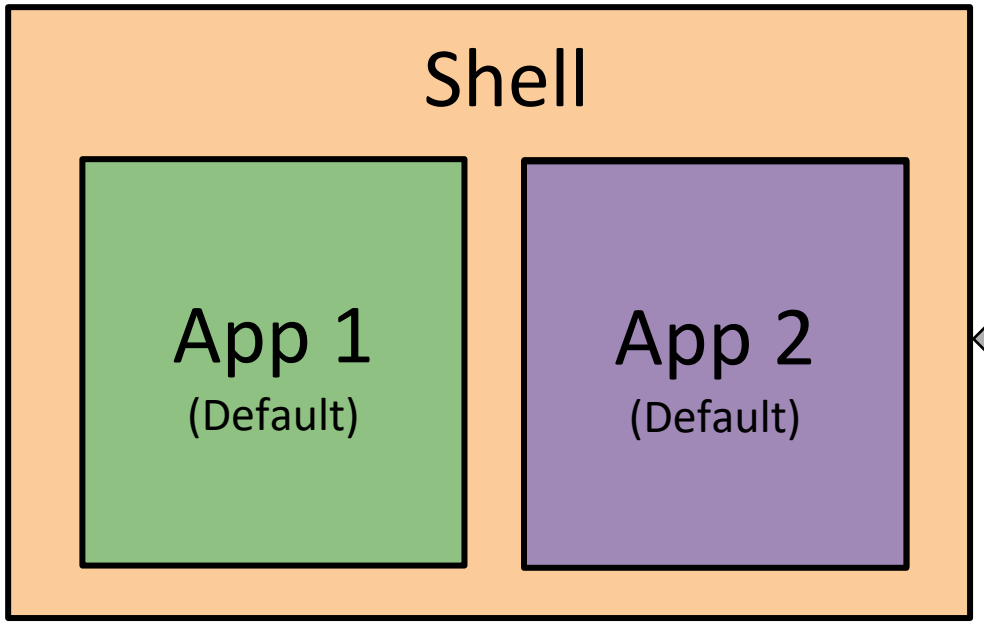
Today's Practical Constraints



- # 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

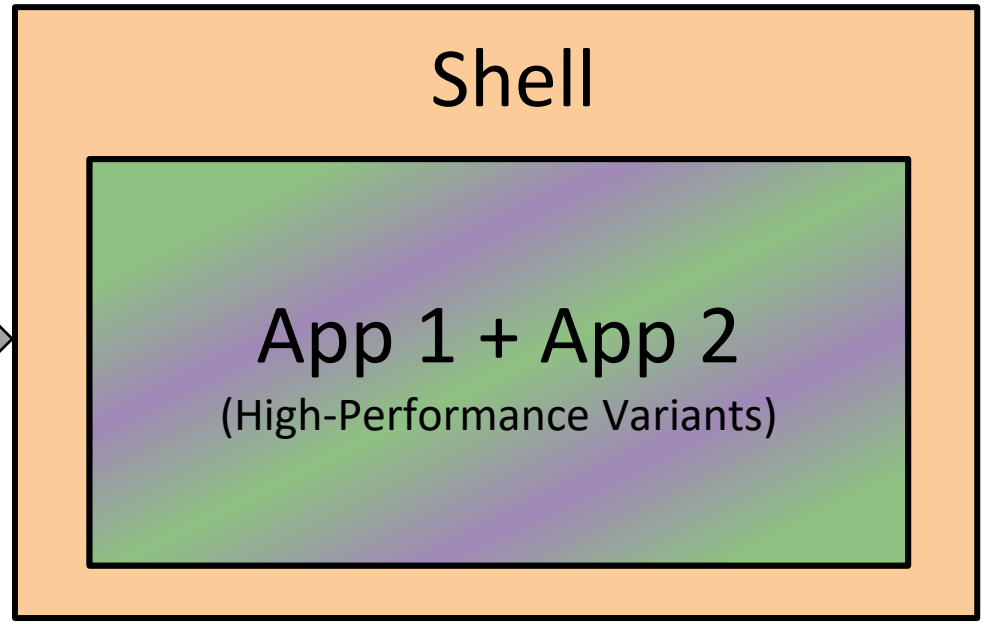
Many innovative ways to get around: Hierarchical PR

Spatial and Time Multiplexing



Multiple simultaneous applications

Simple Role and Shell design

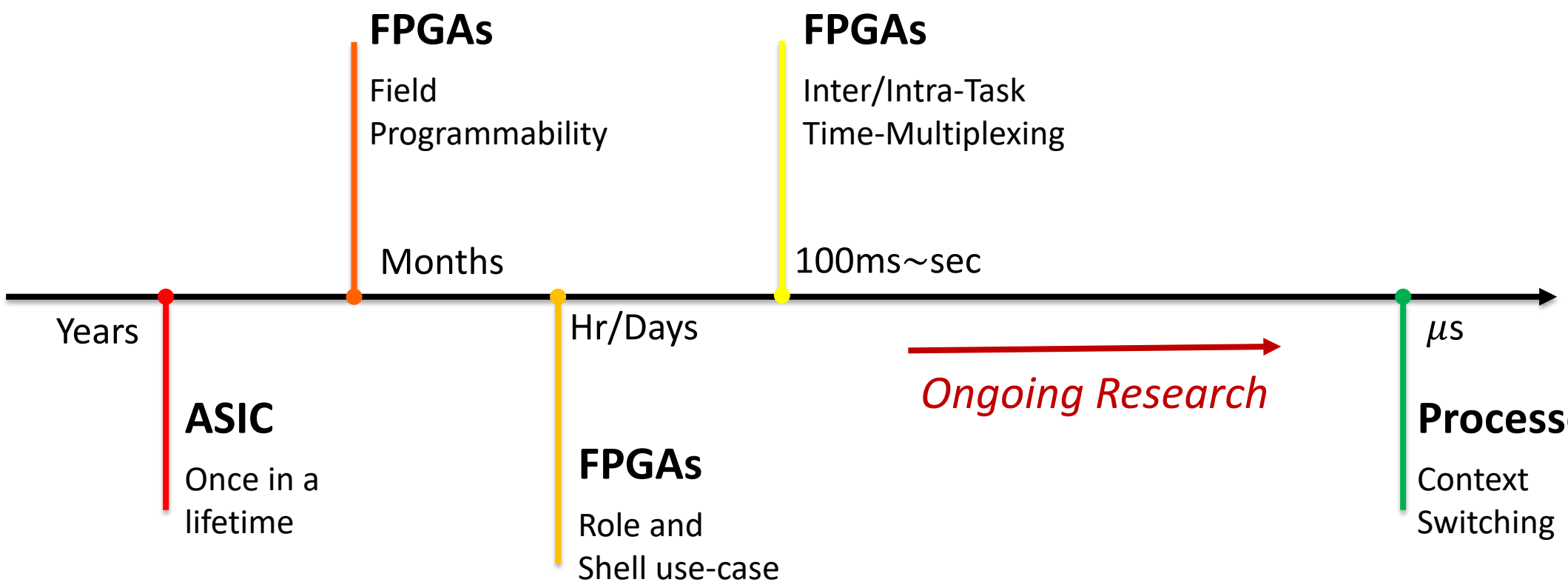


Avoids disadvantages of slot-based approach

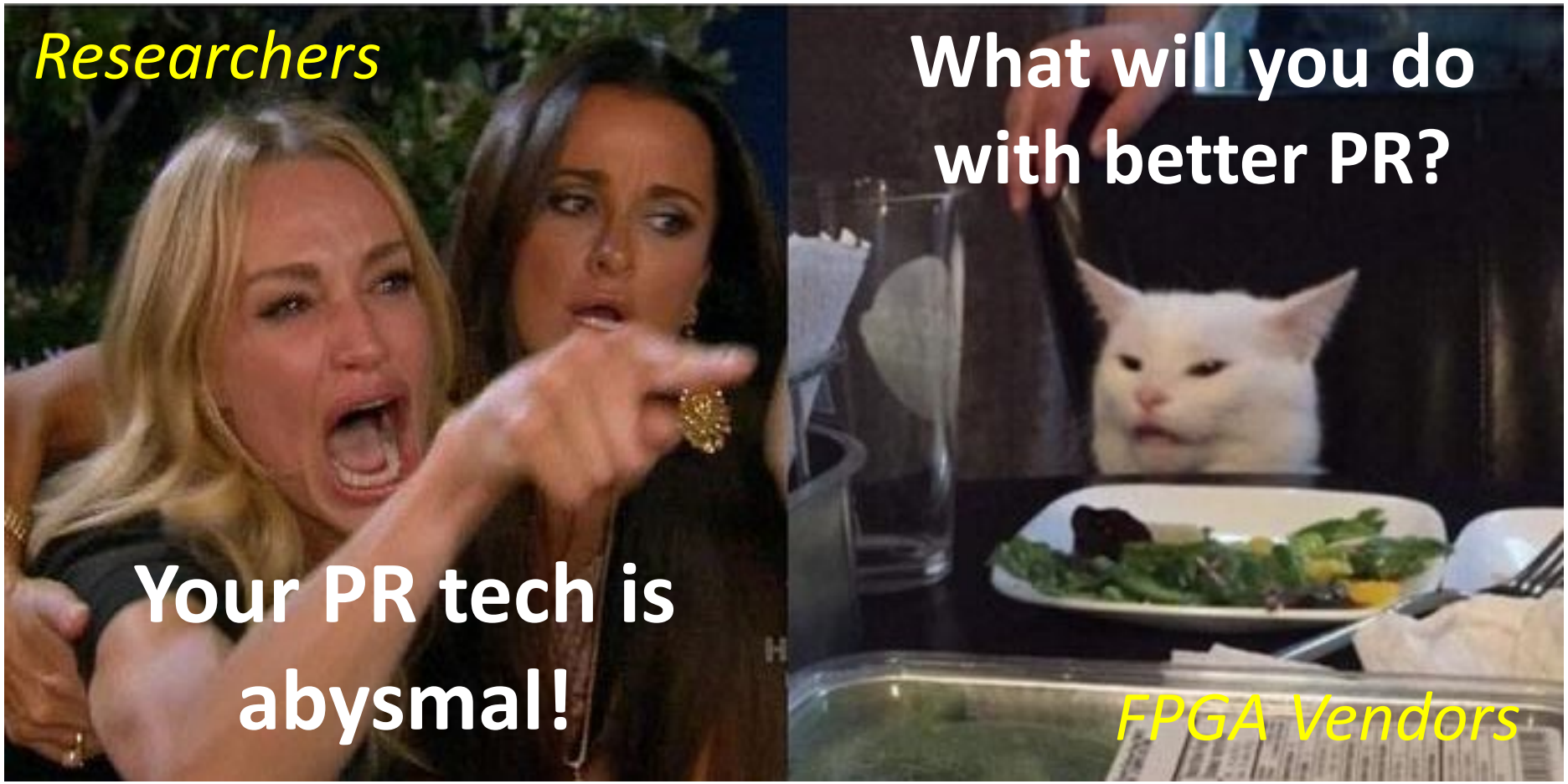
But did we really get out of the "ASIC-style" of thinking?

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).

This is where we're at technologically…



We want to push further but: chicken or egg?

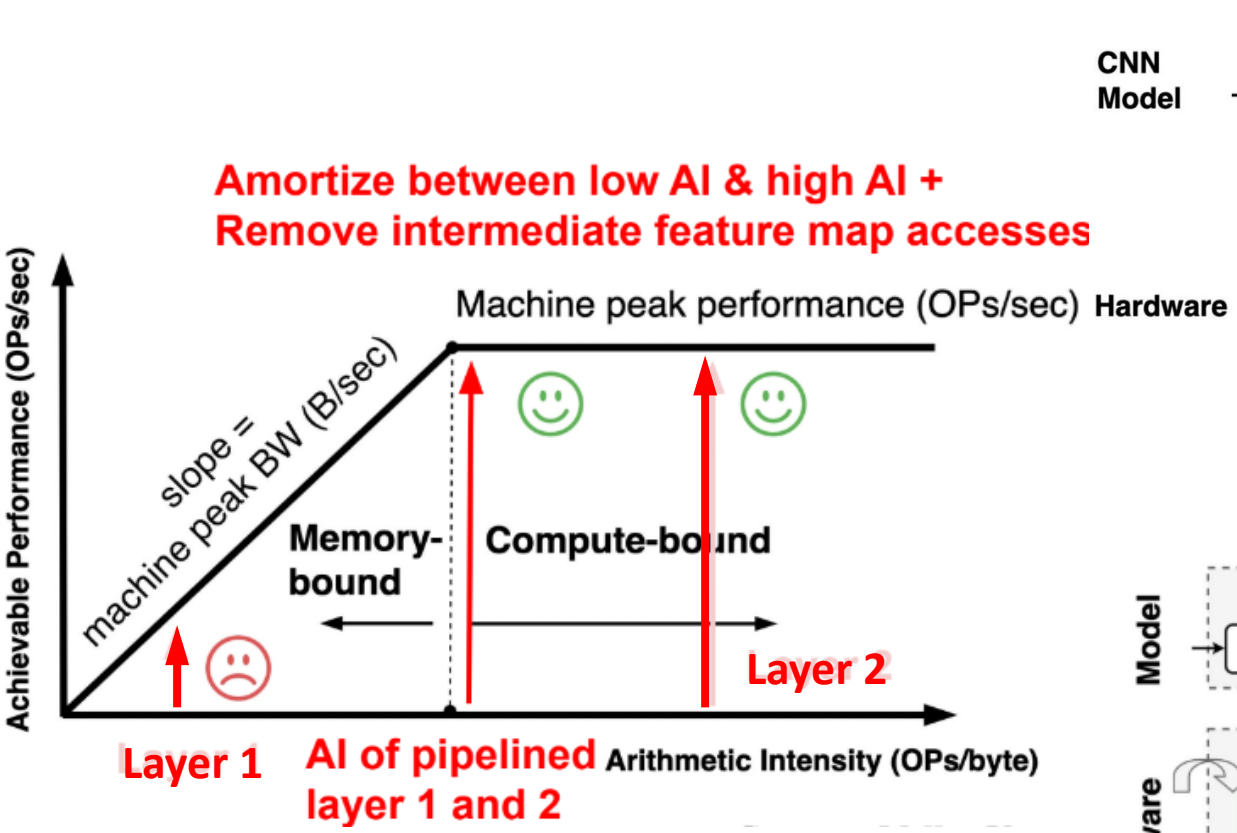


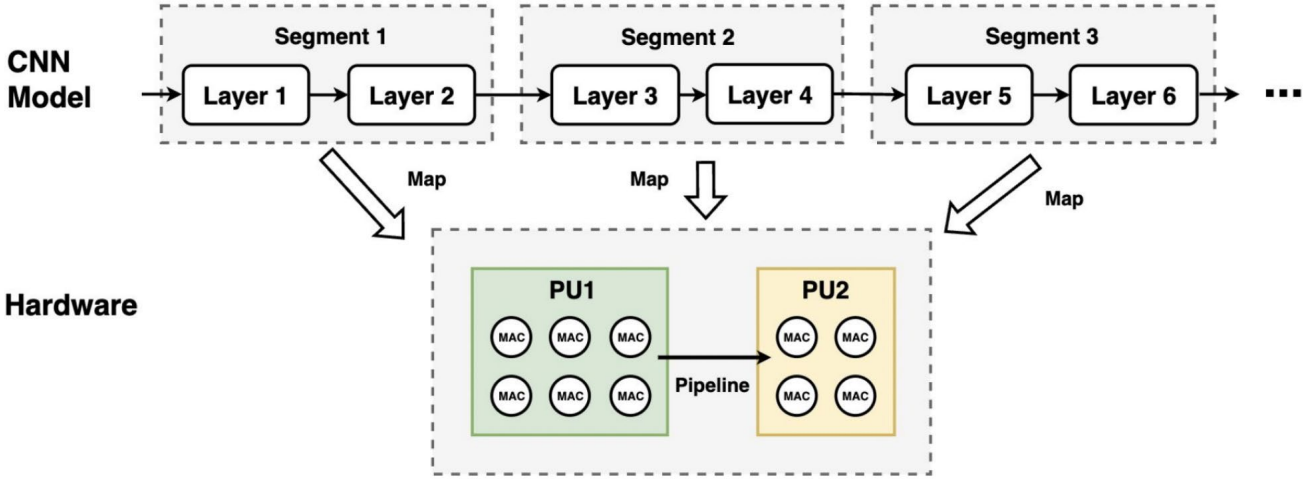
One Deployment, One Design

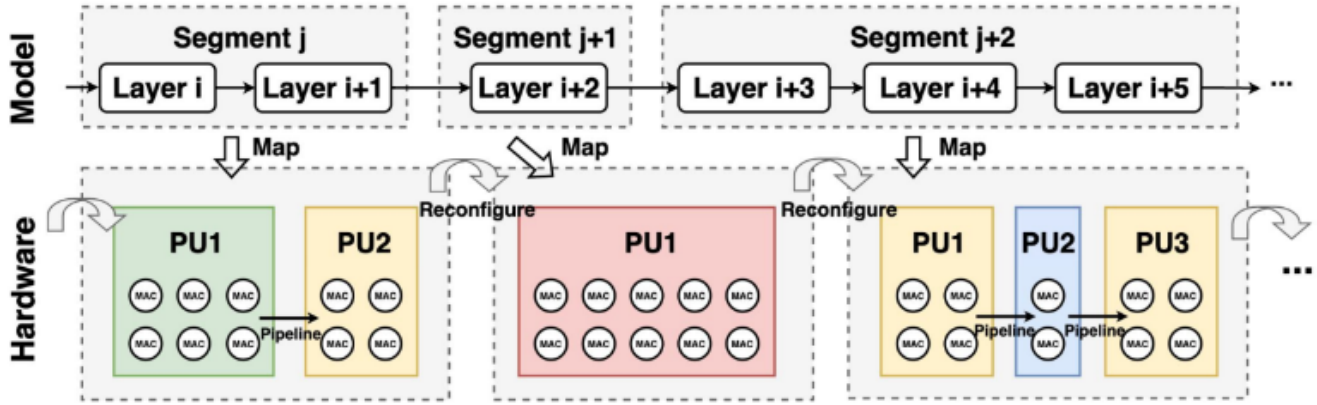
- **Compile time** choices
  - re-tune for large vs small FPGA part choices
  - re-tune for same FPGA part for different deployments
- **Runtime** choices
  - adjust datapath to changing operating conditions
  - add/change FPGA usage to changing conditions
- **Specializing** (compile time and/or runtime)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

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

Extending the CNN example: Reconfigurable Intra-Segment Pipeline Architecture

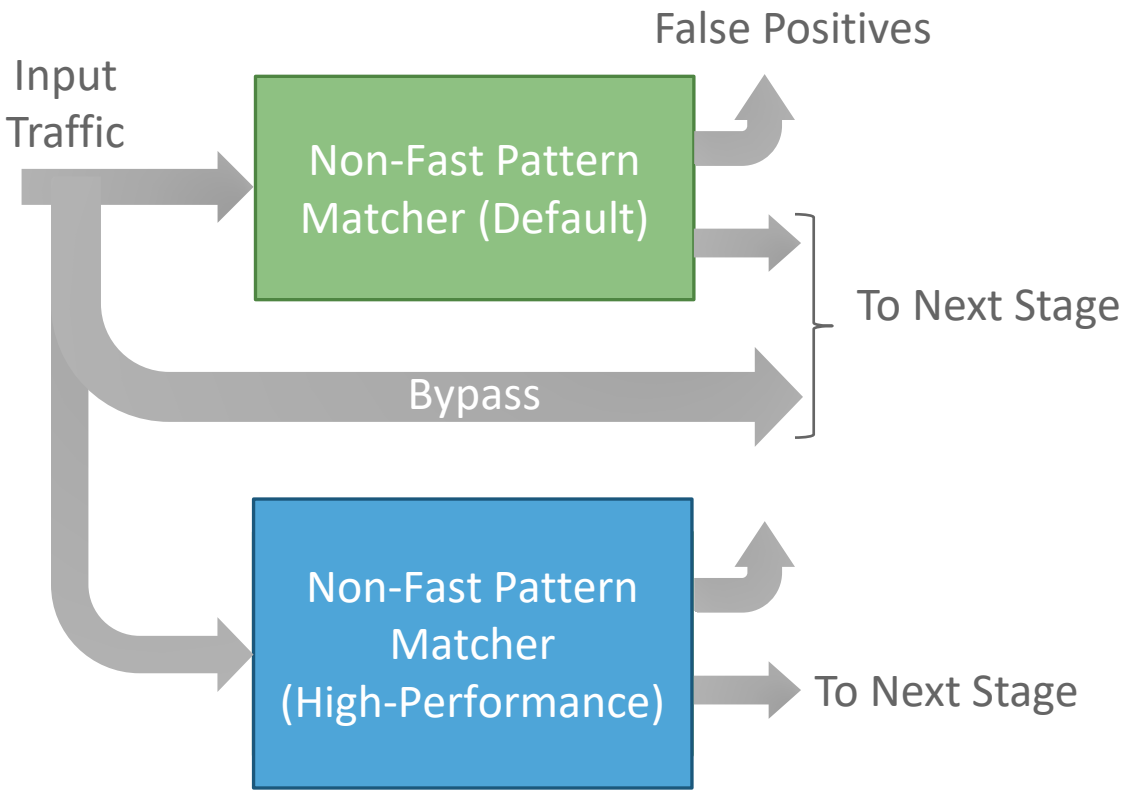






Reacting to bursty traffic patterns

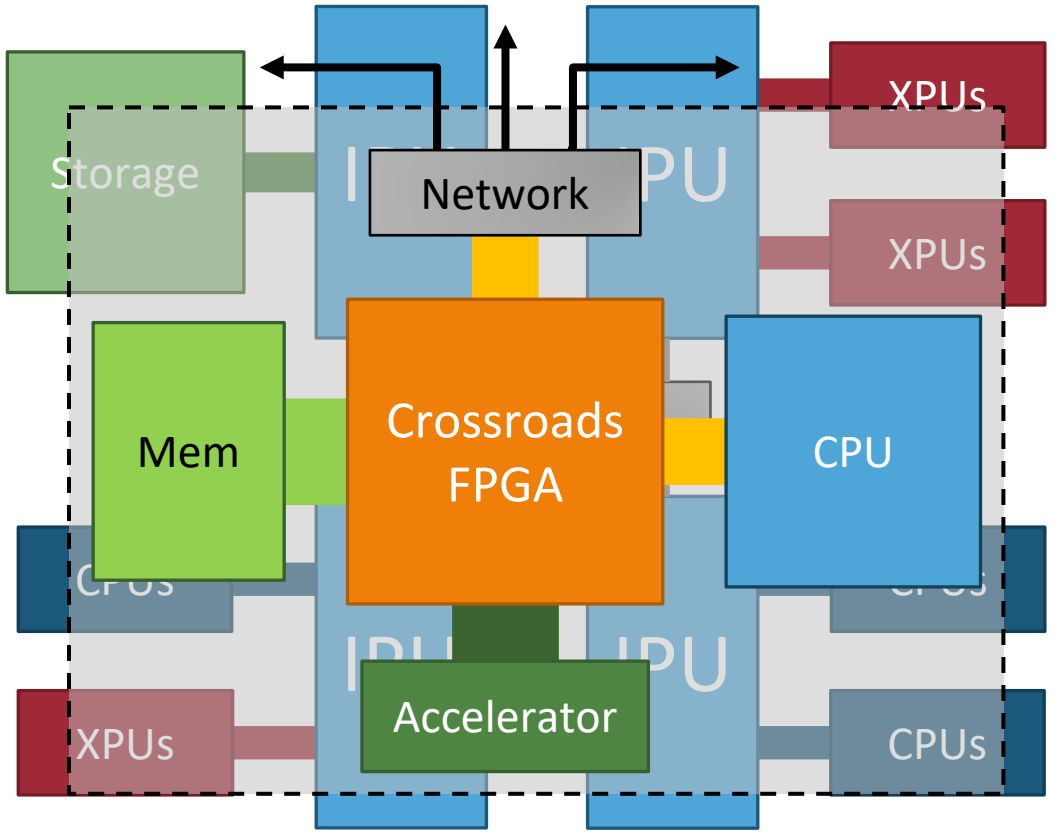
- *QoS Requirement*: Zero Packet Loss in Intrusion Detection/Prevention System
- Single NFPM sufficient for usual case of average 15Gbps stage traffic
- Bypass can handle occasional overflow cases – but what if not occasional?
- *Slack: We don't need the units all the time* – Dynamically request more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



Does this make sense on current FPGA architectures/deployments?

FPGAs in Datacenters (More in Week 11)

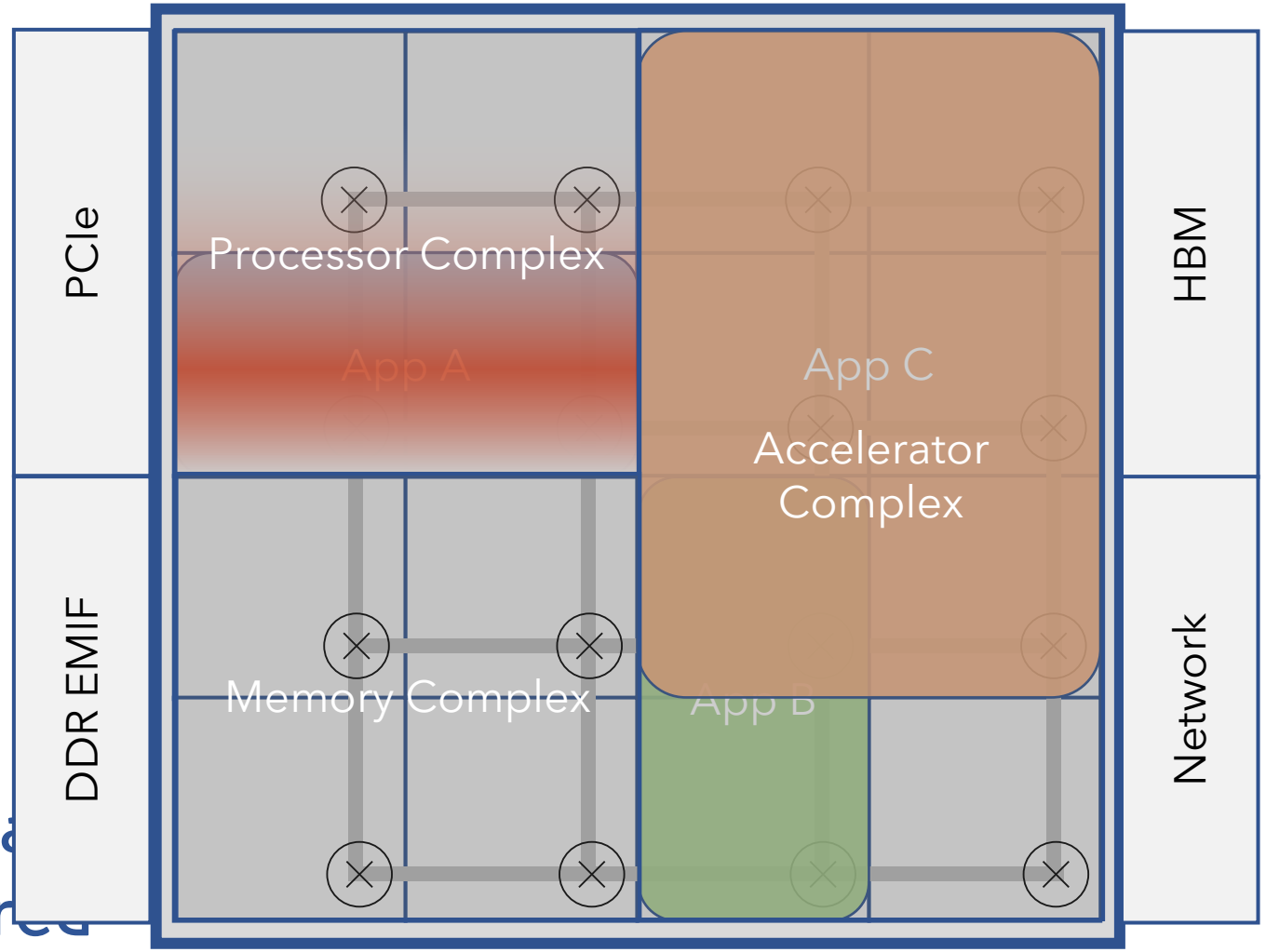
- FPGA as a data hub allows computing near-data and on data flows
- Immense demand to be near-data – PR is essential to cater to this demand and support multi-tenancy
- Traditional ASIC-style approaches under-utilize the programmability of FPGAs
  - Need better PR hardware
  - Applications designed to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



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

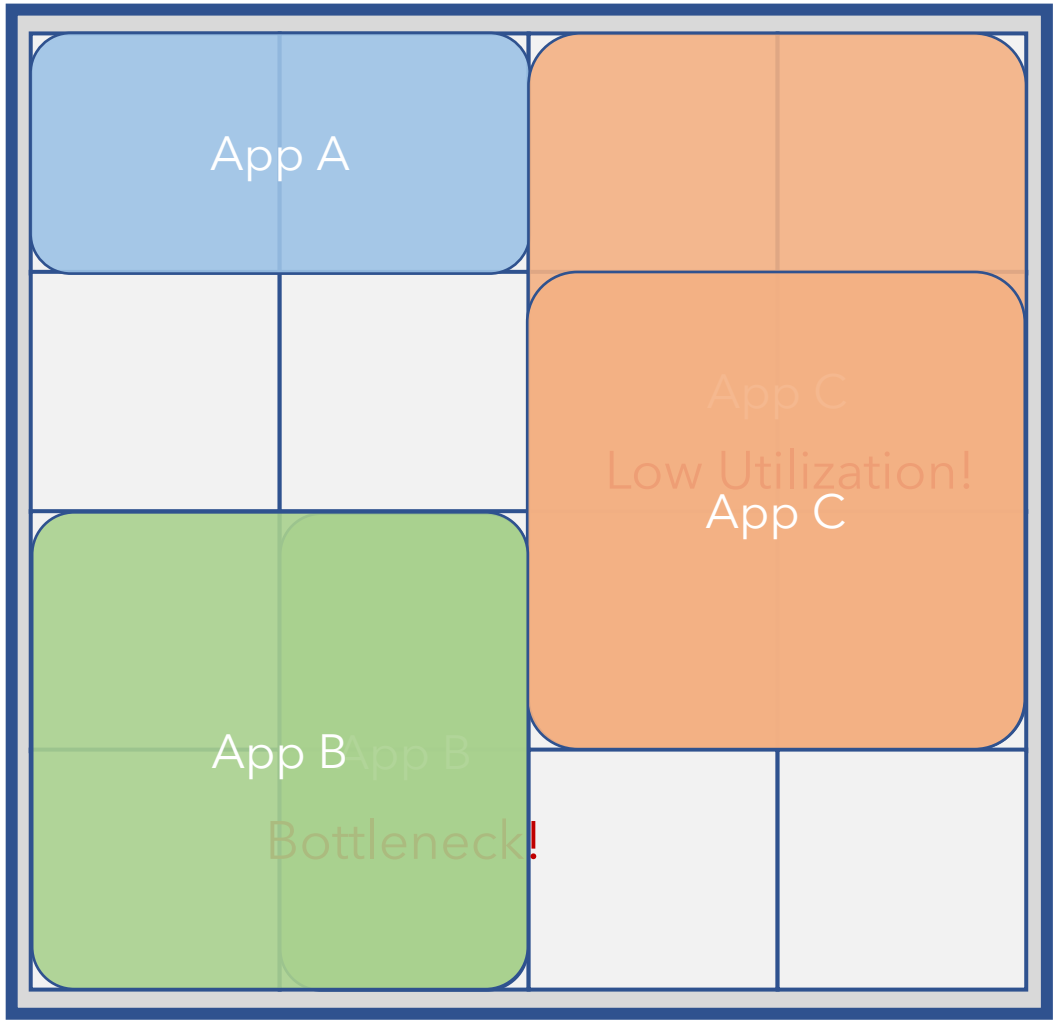
IPU: https://www.intel.com/content/www/us/en/newsroom/news/infrastructure-processing-unit-data-center.html

RV5: PR is essential to Crossroads



The Crossroads FPGA is virtualized!

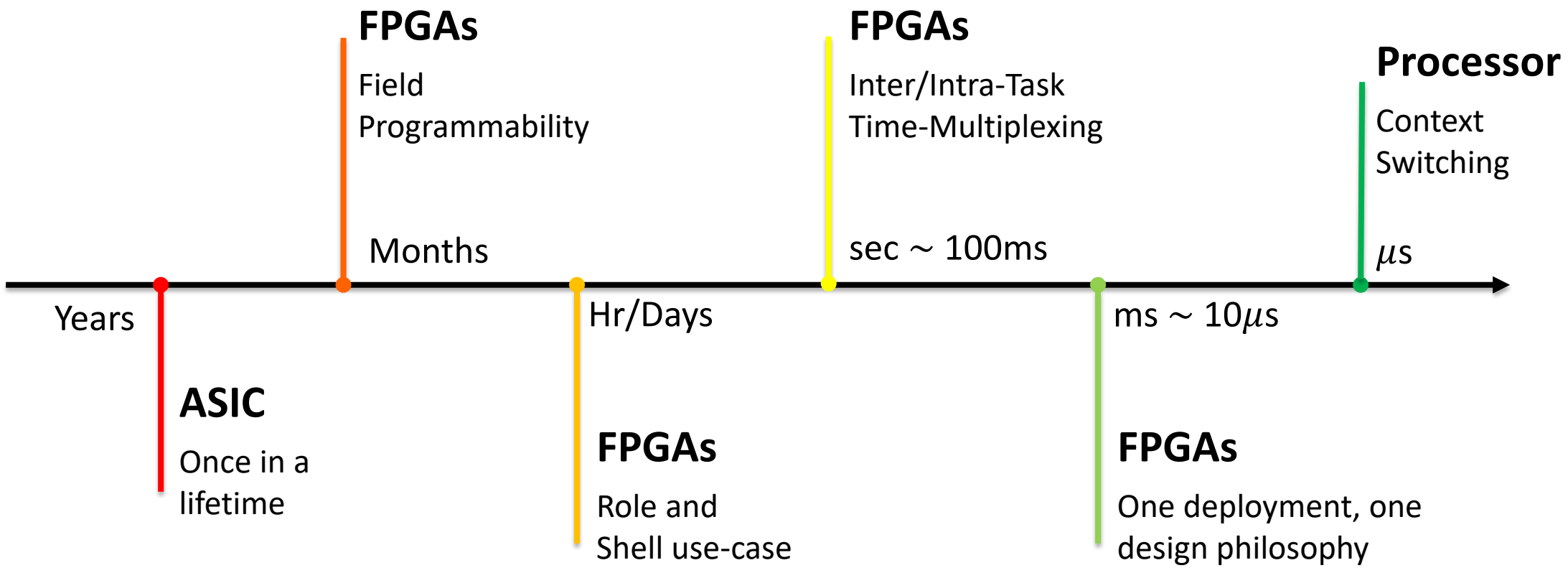
RV5: PR can be used for more!



Scale-up as Performance Requirements Increase

How do we recognize these situations and respond to them?

It's always been about reclaiming "Slack"



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

Discussion: What needs to change in the computing stack to support such use of FPGAs?

	CPUs	FPGAs
Application Scaling	Processes/Threads allow applications to scale as parallelism changes	
Programming Model	Inherently sequential model with consistency defined for parallelism	
Global Communication	RPCs, MPI (distributed computing frameworks)	
System Orchestration	Load balancing allow tasks scale with workload demands	
Operating System	Scheduling processes/threads with priority using context switching	

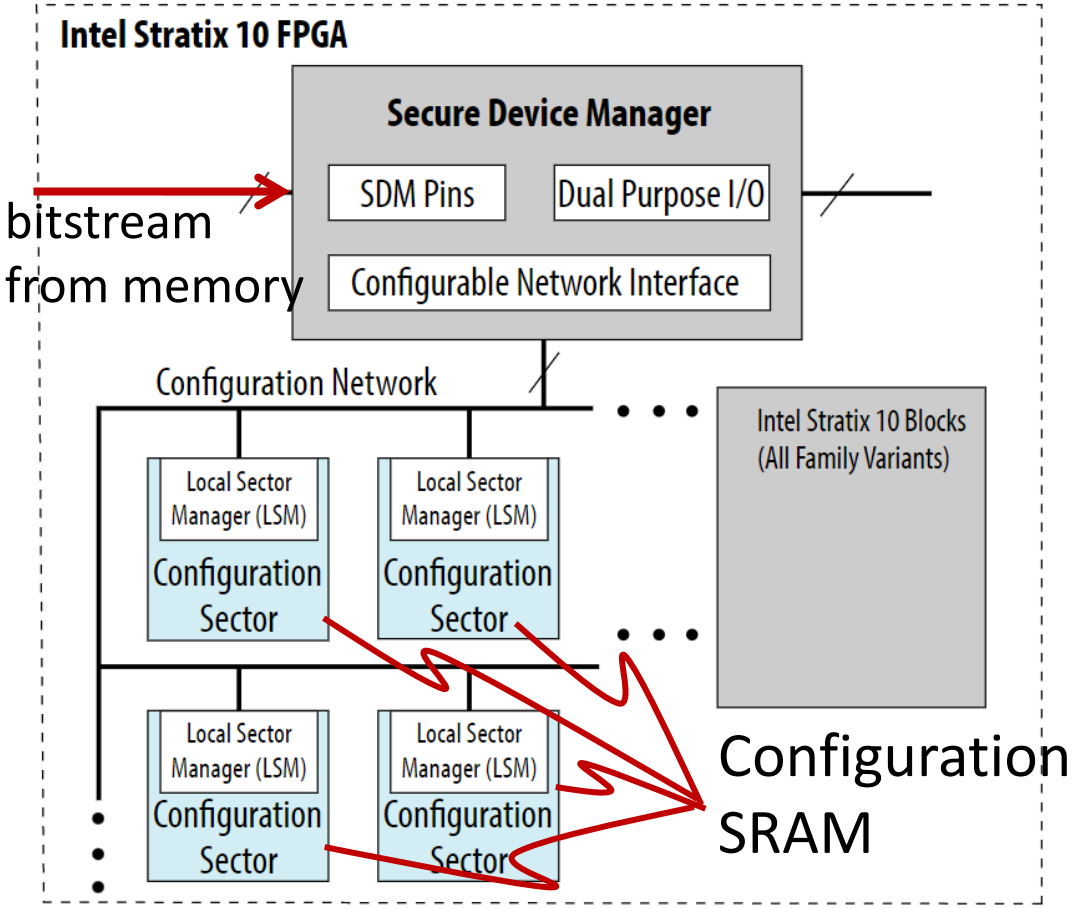
Discussion: What needs to change in the computing stack to support such use of FPGAs?

	CPUs	FPGAs
Application Scaling	Processes/Threads allow applications to scale as parallelism changes	Need generatable designs that can traverse the performance-cost design space
Programming Model	Inherently sequential model with consistency defined for parallelism	Data forwarding optimization complicates module duplication and scaling
Global Communication	RPCs, MPI (distributed computing frameworks)	If the FPGA is truly virtualized, this should not have to change
System Orchestration	Load balancing allow tasks scale with workload demands	FPGAs will be part of a heterogenous local system – each component with its own niche
Operating System	Scheduling processes/threads with priority using context switching	Handle spacial resource? What needs to scale? What's a context?

Break further from the ASIC mentality

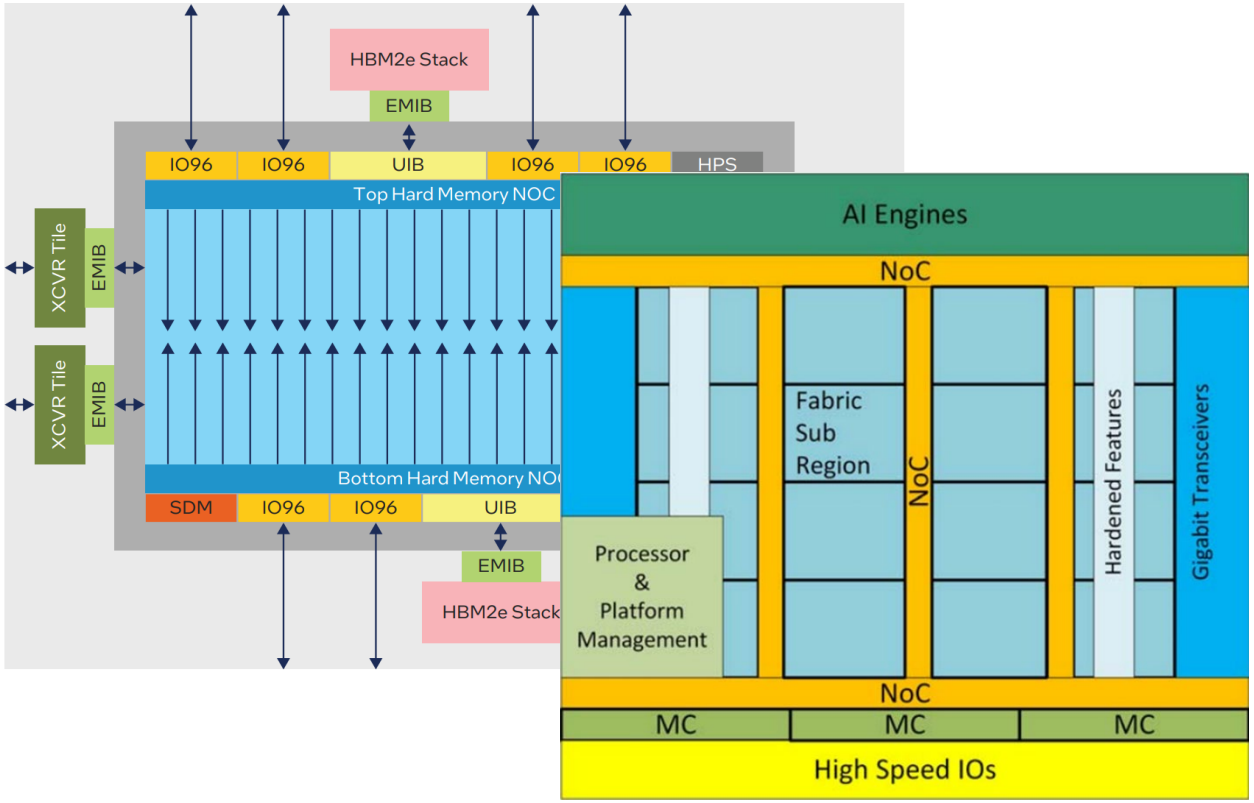
- **Dynamism** — actually use the programmability
  - support more functionality on same parts cost
  - achieve better performance by specializing
- **Shareability** — multitenancy to consume "slack"
  - too much resource: partition fabric spatially
  - too much throughput: repurpose fabric temporally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

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

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

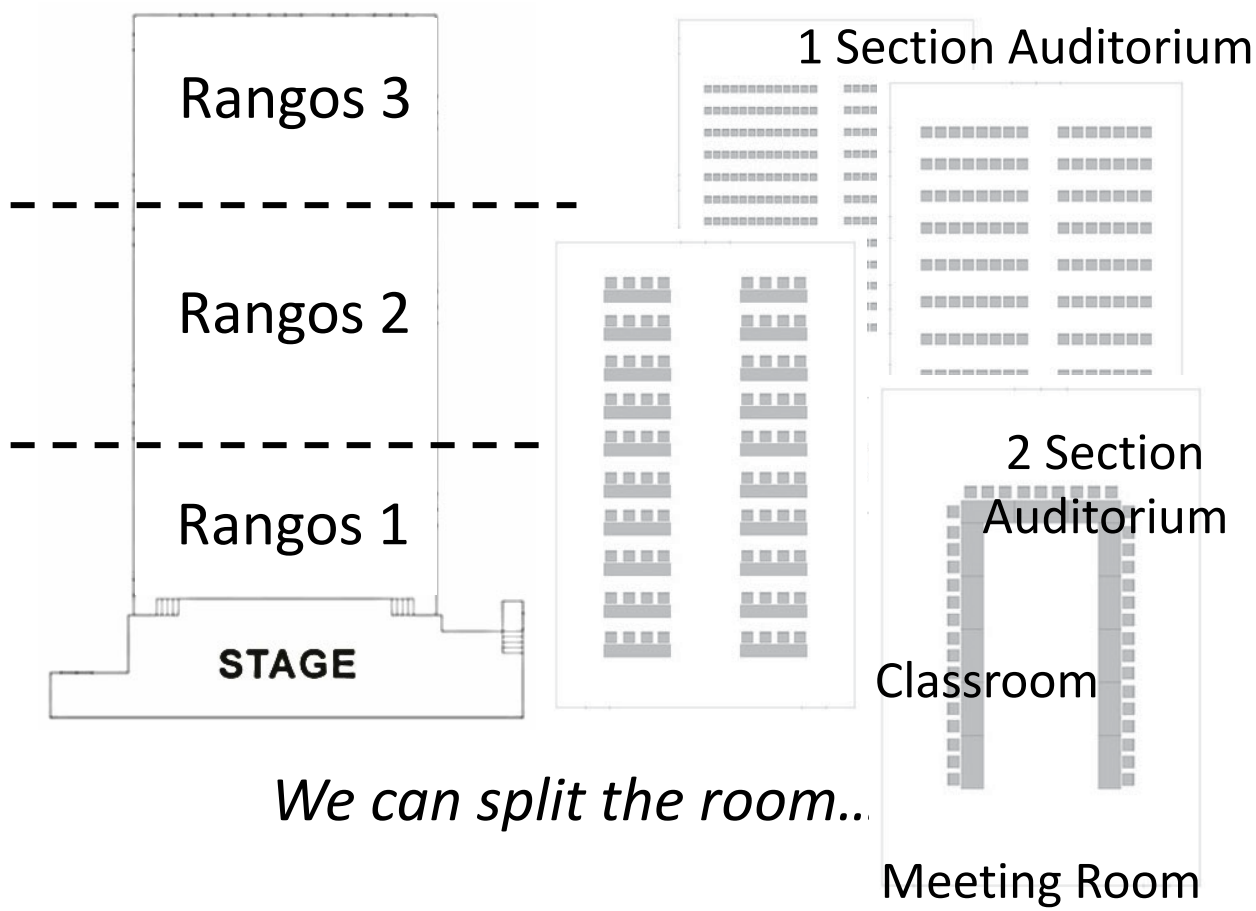
Renting out an Event Venue

Event Planning

When would I ever need Rangos to be a classroom?

- My class is not a regular (CMU should make more classrooms if a regular class cannot fit)
- My class is not small (cannot wing it without a classroom)
- Students need to move to another event nearby on a tight schedule

Specification



We can split the room..