Performance Modeling with A-Ports

Michael Pellauer[†]
Murali Vijayaraghavan[†]
Michael Adler[‡]
Joel Emer[†‡]

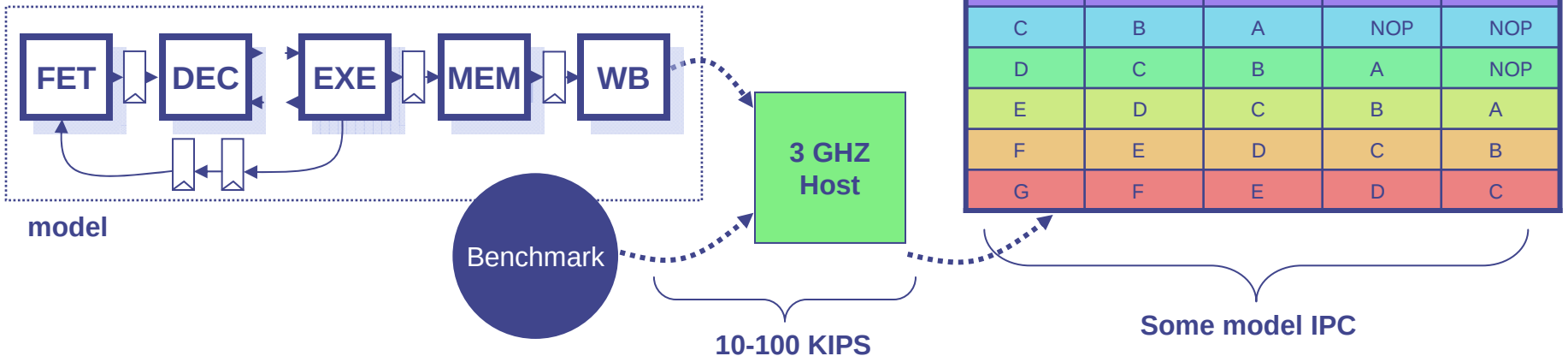
[†]MIT Computer Science and AI Lab
Computation Structures Group
{pellauer, vmurali, emer}
@csail.mit.edu

[‡]Intel
VSSAD Group
{michael.adler, joel.emer}
@intel.com

Performance Modeling

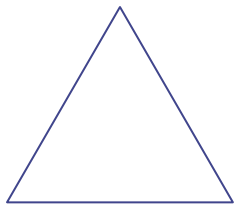
= clock cycle

- Performed early in the design process
  - Guide architectural decisions
  - Perform feasibility analysis



FET	DEC	EXE	MEM	WB
A	NOP	NOP	NOP	NOP
B	A	NOP	NOP	NOP
C	B	A	NOP	NOP
D	C	B	A	NOP
E	D	C	B	A
F	E	D	C	B
G	F	E	D	C

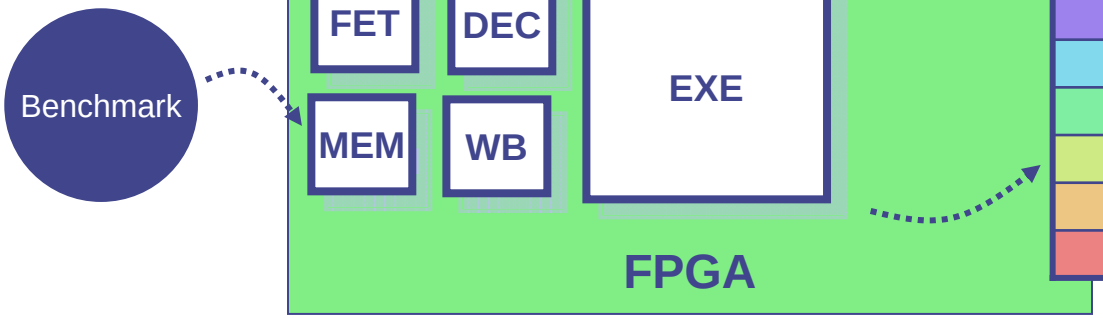
- Simulator architect begins to tradeoff between:
  - Simulation speed
  - Model development time
  - Accuracy/Fidelity



- Can we improve this by using FPGAs?

Performance Models on FPGAs

- FPGAs run at ~100 MHz
- How can we achieve a better result than software?
- Parallelism



FET	DEC	EXE	MEM	WB
A	NOP	NOP	NOP	NOP
B	A	NOP	NOP	NOP
C	B	A	NOP	NOP
D	C	B	A	NOP
E	D	C	B	A
F	E	D	C	B
G	F	E	D	C

On one FPGA clock tick we will be simulating many different modules

- But...
  - Clock period might be bad
  - ASIC structures might not correspond to FPGA structures
  - It either fits or it doesn't

Simulate a Clock using the FPGA

- Solution: “virtualize” the clock
  - One FPGA cycle does not have to correspond to one model cycle
  - Result of simulation does not have to be FPGA waveform

FPGA CC	FET	DEC	EXE	MEM	WB
0					
1					
2	A	NOP	NOP	NOP	NOP
3					
4					
5	B	A	NOP	NOP	NOP
6					
				NOP	NOP
				A	NOP

This is not RTL prototyping!

- Can take advantage of FPGA structures (BRAMS)
- Can use high-level synthesis (Bluespec)
- Don’t need complete RTL for the entire design

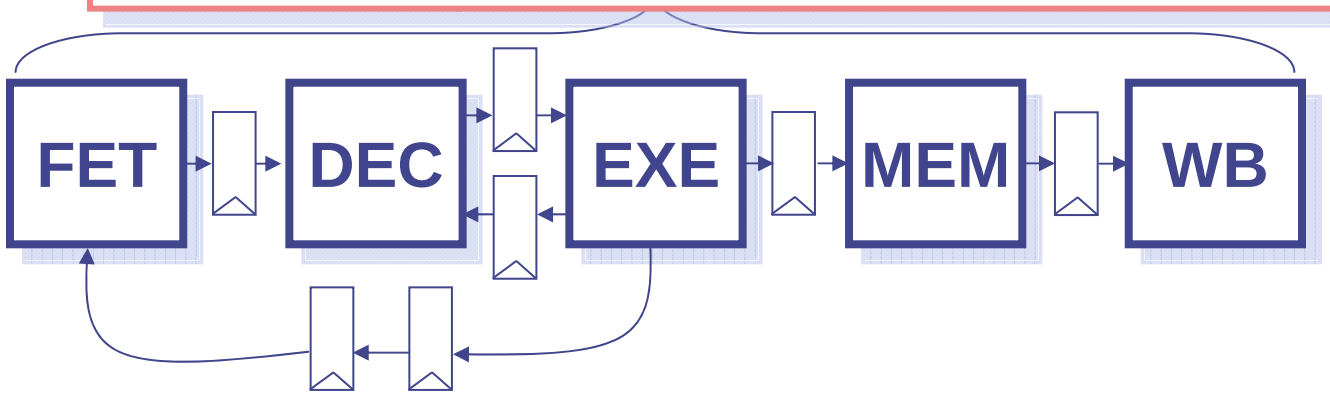
Back of th

- Each module takes 10 FPGA cycles to simulate one model cycle
- FPGA at 100 MHz = 10 MIPS

Unit-Delay Simulation on FPGAs

- All mod
  - Used

- But…
  - What if you have some uncommon case which requires more than *n* cycles?
  - What if you can't bound *n* to start with?

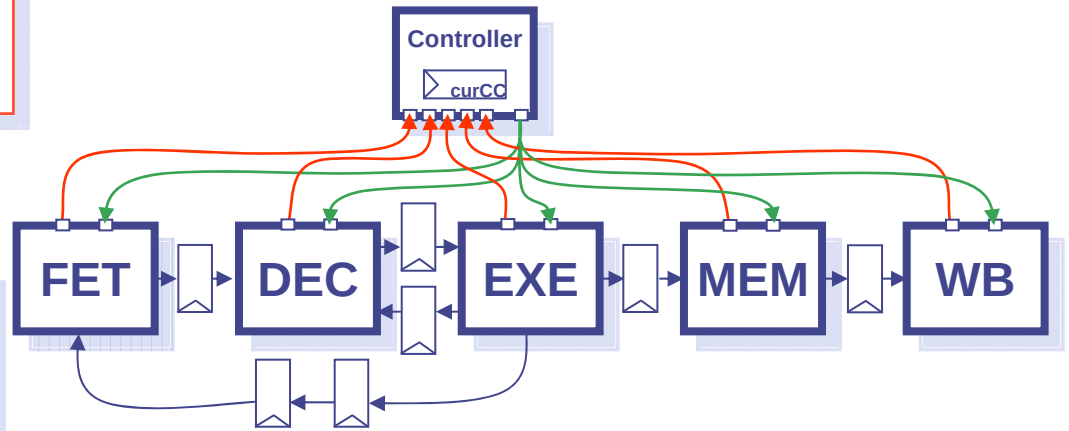


- Advantages:
  - Nice FIFO abstraction
    - Each module follows the same "read, work, write" loop
  - Easy to reason about
    - $\text{Frequency}_{\text{simulator}} = \text{Frequency}_{\text{FPGA}} / n$

Barrier Synchronization on FPGAs

Maintains FIFO abstraction

Dynamic worst case



- But…
  - Modules may need “shadow state”
  - Harder to reason about performance
  - When should you take more CCs vs worsening clock rate?
  - If you trade too much space for time performance will suffer
- We can use metrics to aid the simulator architect
  - Make judicious tradeoffs

Metrics for FPGA Performance Models

- FPGA cycle to Model cycle Ratio (FMR):

$$FMR = \frac{Cycles_{FPGA}}{Cycles_{Model}}$$

- Simulator Frequency:

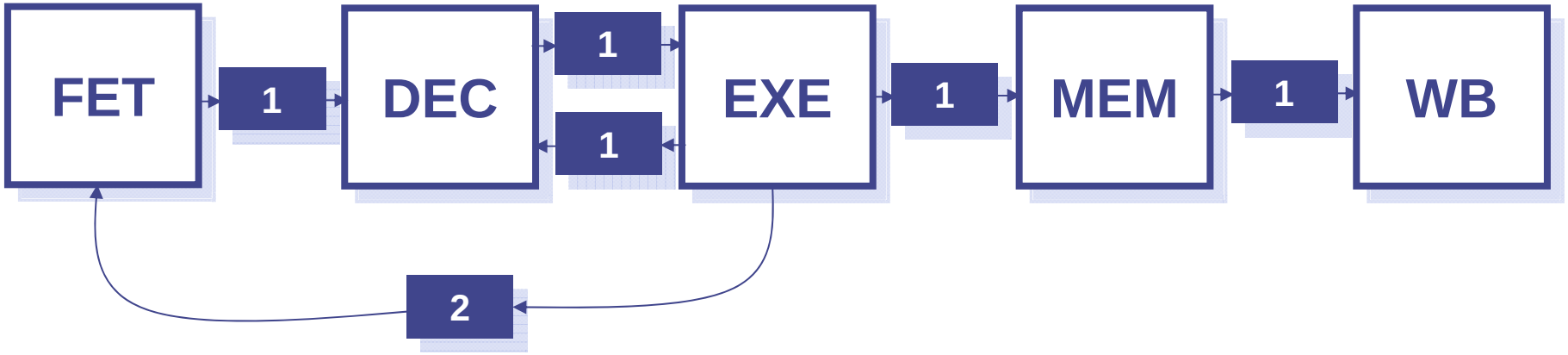
$$Frequency_{simulator} = \frac{Frequency_{FPGA}}{FMR}$$

- Good for comparing two simulators simulating the same machine on the same input data
- Particularly useful for considering simulator refinement optimizations

Applying the Metrics

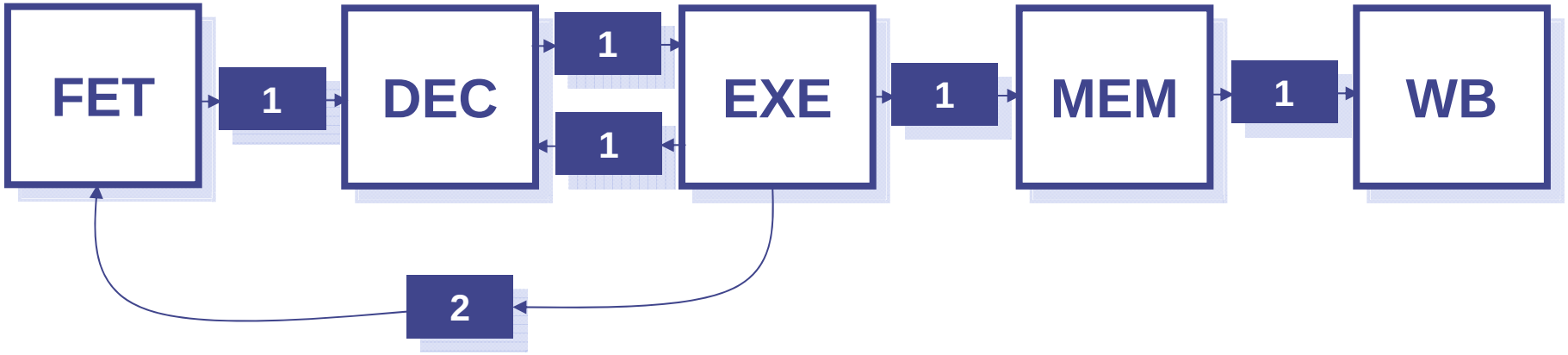
- $FMR_{unitdelay} = n$
- $FMR_{barrier}$ = sum of worst of all model cycles divided by number of cycles simulated
- But what about $Frequency_{FPGA}$?
  - Unit Delay should not be the critical path
  - But we would expect the Controller to scale poorly
  - Experiment: 25 -> 100 modules == 120 MHz -> 75 MHz
- Summary:
  - Unit-Delay: Only applies if you can bound a small *n*
  - Barrier Synchronization: Dynamic worst-case can improve FMR, but $Frequency_{FPGA}$ does not scale
- Can we have the best of both worlds?

Asim Ports in Software



- Used in software Asim performance models
  - All communication goes through Ports
  - Ports have a model time latency
  - Beginning of a model cycle a module reads all Ports
  - End of a model cycle write all Ports
- Related: RAMP RDL channels, UT Fast Connectors

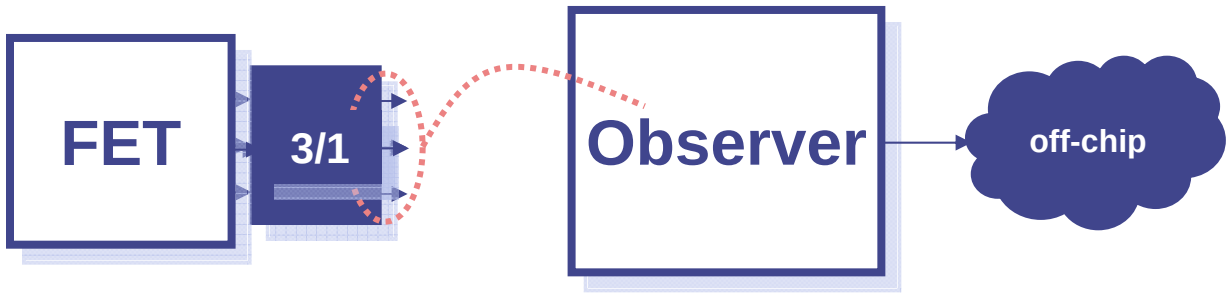
A-Ports: Asim Ports on an FPGA



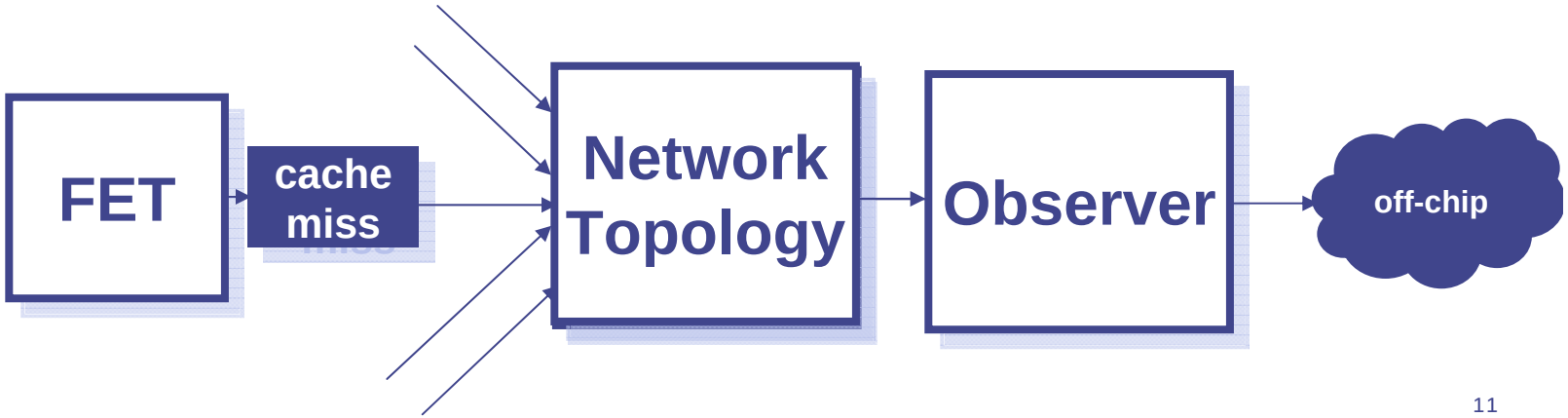
- Distribute the control, no combinational paths, no local counters
- Implemented using balanced/heavy/light protocol
- Finite buffering
  - Only read when not empty, only write when not full
  - Use Bluespec to manage via implicit conditions
- Maintains the FIFO abstraction
- Allows adjacent modules to “slip” in model time
  - Simulate different model CCs on the same FPGA cycle

Observing the Result of Simulation

- Simulation Results are only observed via A-Ports
  - Similar to “clock boundaries” and metastability



- Events: a convenient abstraction



Baseline A-Ports Assessment

- $Frequency_{FPGA}$
  - As good as Unit-Delay
  - No combinational paths between modules
- FMR
  - As good as Barrier Synchronization
    - Module makes local decision to proceed when input ready
    - Barrier Synchronization is bounded by dynamic worst case
  - But for certain topologies A-Ports can do better
    - Consumers can run ahead to fill buffers
    - Long-running ops on different model cycles can overlap

Example: FMR Decoupled A-Ports

= model cycle

FPGA CC	FET	DEC	EXE	MEM	WB
0	A	NOP	NOP	NOP	NOP
1	B	A	NOP	NOP	NOP
2	C	B	A	NOP	NOP
3	D	B		A	NOP
4	E (full)	B		A	
5		B		A	
6		B		A	
7		C	B	A	
8	F	D	C	B	A
9	G (full)	D		C	B
10		D			C
11		D			
12		D			
13		E	D		
14		F	E	D	

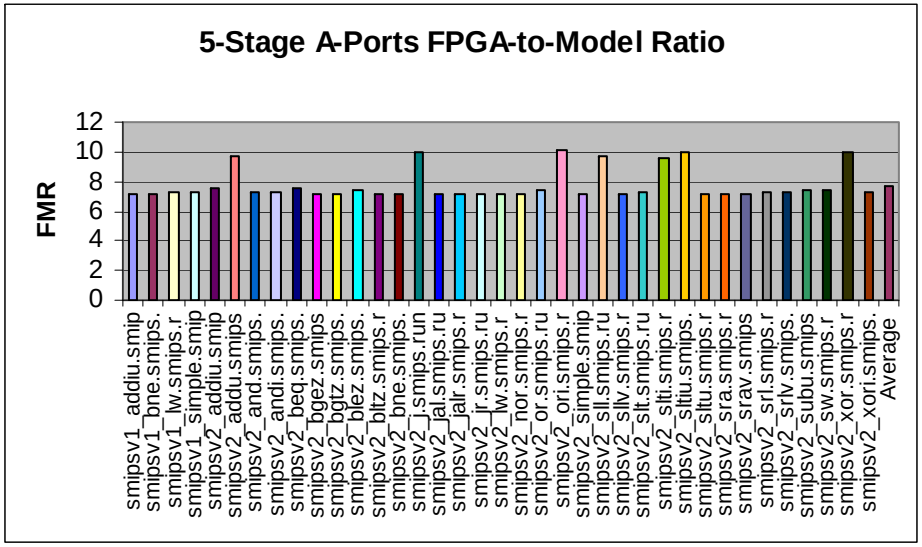
run-ahead in time until buffering fills

long-running ops can overlap

Observed results of simulation do not change!

Experimental Results

- In-Order Model (5-stage Pipeline)
  - Average FPGA-to-Model Ratio (FMR) using A-Ports: 7.74



Synthesis Results Virtex II Pro 30:

Slices:	8794/13696	64%
Slice Flip Flops:	5470/27392	19%
4 input LUTs:	16665/27392	60%
BRAMs:	25/136	18%

- Out-of-Order Model (R10k-like 4-way superscalar)
  - Average FMR for Barrier: 19.54
  - Average FMR for A-ports: 16.91

Takeaways

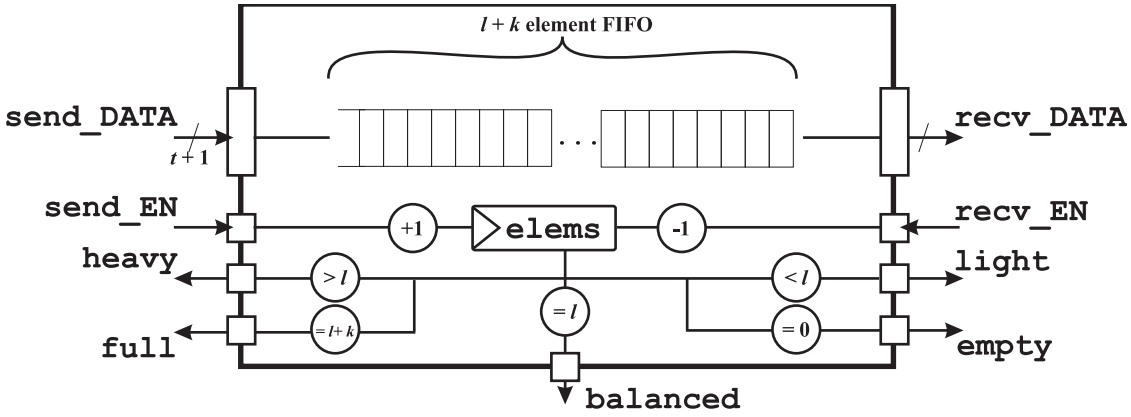
- FMR and other metrics aid in reasoning about FPGA performance models
- Barrier synchronization and unit-delay are useful on a limited class of applications
  - Barrier: small scale, Unit-Delay: small bound
- A-Ports combine the benefits of the two approaches
  - Local routing and control ($Frequency_{FPGA}$)
  - Better performance than barrier (FMR)
- Changing an FPGA functional simulator to a cycle-accurate simulator can be done easily and cheaply
- Open Questions
  - Can A-Ports FMR be expressed equationally?
  - Given a specific topology and work distribution, how much buffering would maximize A-Ports FMR?

Questions?

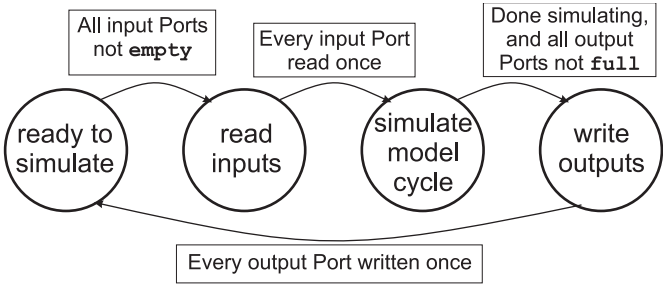
pellauer@csail.mit.edu

Extra Slides

A-Port implementation



- Implemented as a FIFO of at least *l* elements



- Protocol stages are conceptual
  - can all be done in one FPGA cycle

Adjacent models are now decoupled
Can slip ahead, behind in time
Consumer can only go ahead l cycles
Producer can only go ahead *k* cycles (extra buffering)

Metrics for FPGA Performance Models

- In software PM:

$$IPS_{simulator} = \frac{Frequency_{simulator}}{CPI_{model}}$$

- FPGA version:

$$IPS_{simulator} = \frac{Frequency_{FPGA}}{CPI_{model} \times FMR}$$

- The metric of “real-time” interactivity
- New Iron Law of FPGA Performance Models?