Compiling a Parallel DSL to GPU

Ramesh Narayanaswamy
Badri Gopalan
Synopsys Inc.

Agenda

- Overview of Verilog Simulation
- Parallel Verilog Simulation Algorithms
- Parallel Simulation Tradeoffs on GPU
- Challenges

Verilog Hardware Description Language

Objectives

Model Hardware
- Modern chips run into 100s of Millions of Gates
- Gates in the real world can be thought as parallel blocks in simulation

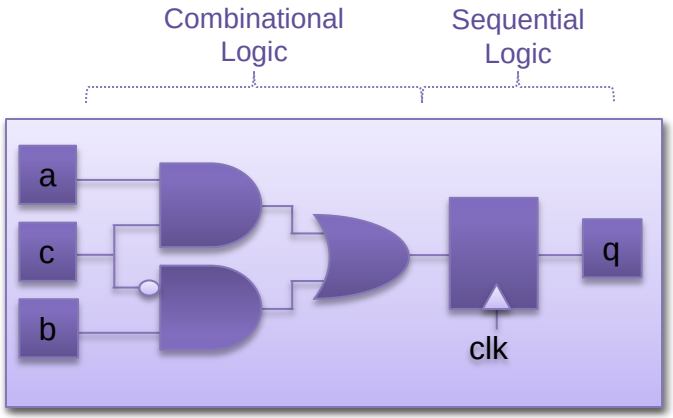
Model Testbench to test Hardware

- Hardware Models
  - Multiple Implementation Levels – Gate, Register Transfer
  - Input Output Behavior – Behavioral

- Testbench
  - Generate Input Data, Expected Results
  - Drive and Compare
  - Input Output Behavior Level
  - SystemVerilog Testbench Extensions

Verilog – Models

- Gate Level
  - Interconnect
    - Combinational logic
    - State holding gates: Latches, Flip Flops
    - Largely Bit level
- Register Transfer Level (RTL)
  - Interconnect Guarded Processes
  - Word Level
  - Process has Multi line behavior
  - Some Process advance clock by one
  - Clocks, Sequences are explicit
- Behavioral Level
  - Process encompasses many clock cycles
  - Process may be longer



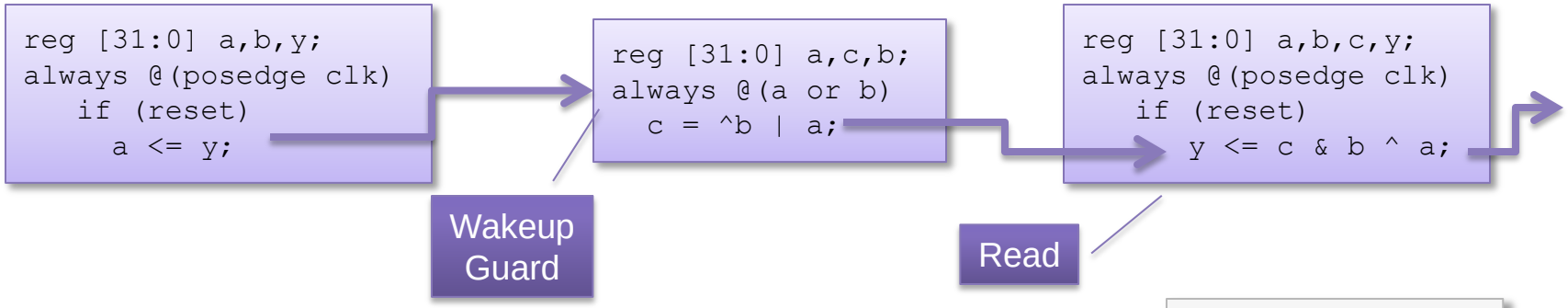

```
reg [31:0] q, a, b; reg c, clk;
always @(posedge clk)
  if (c)
    q <= a;
  else
    q <= b;
```

Guarded Process

```
reg [31:0] q, a, b; reg c, clk;
always @(posedge clk)
  if (c)
    wait @(posedge clk);q <= c;
  else
    q <= b;
```

Behavioral statements

How Verilog Models Parallelism



- Guarded Processes
  - `@(A or B)` change on A or change on B
  - `@(posedge C)` *0* to *{1,x,z}* or *{x,z}* to *1* change on C
  - `#10;` 10 time units have elapsed
- Process body - Sequential code with Assignments
- Global State
  - Simulation Time
  - Values of Variables
  - Process State

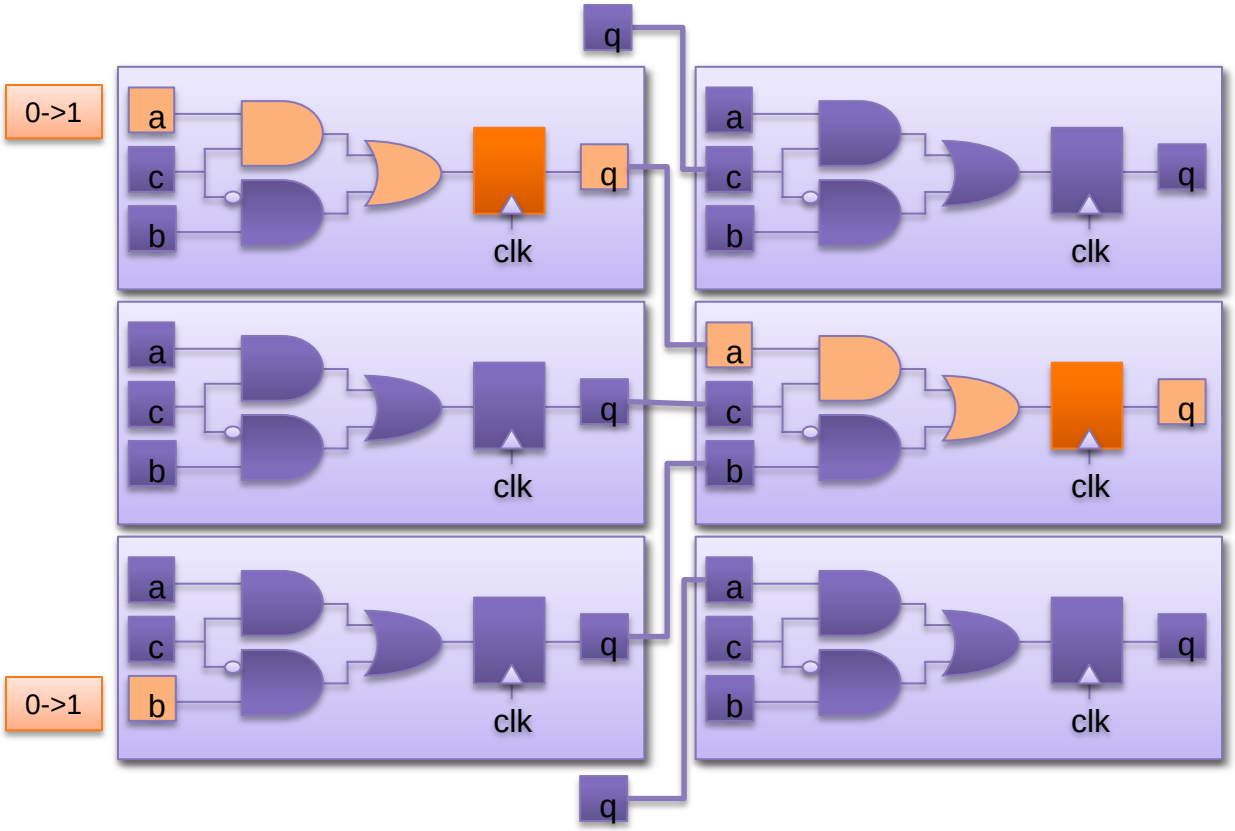
Global State
- Simulation Time
- Variable Values
- Process State
  - Where to continue ?
  - Blocked on a guard ?
  - Executing ?

Collection of Executing Processes: Parallel Workload

HDL Simulators can implement Parallel Workload with:
- Serial Simulation Algorithms
- Parallel Simulation Algorithms

Serial Simulation Algorithms

Event Driven Simulation with Dynamic Scheduling



Each block guarded by a change-check

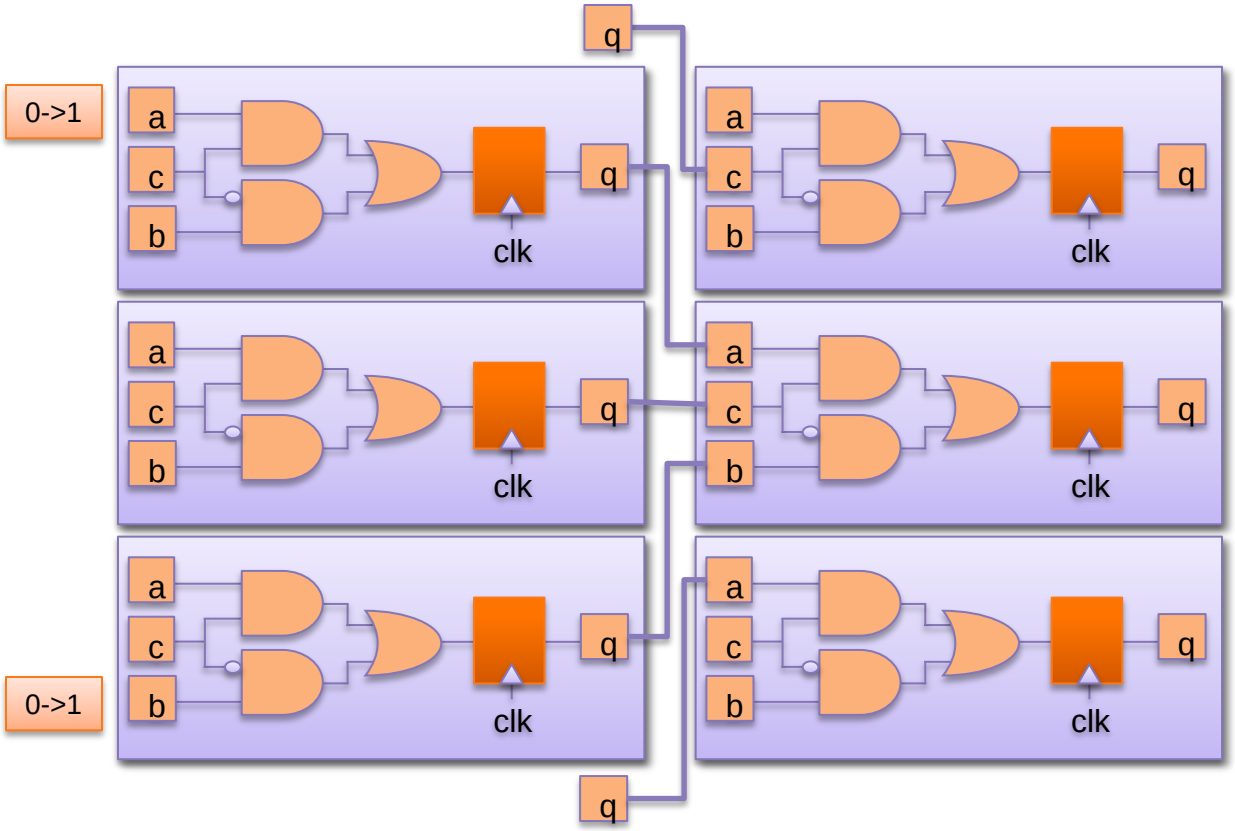
Functional evaluation dynamically scheduled, executes only what is needed

Many optimizations for serial simulation, works in normal scenarios of low event activity

But: not easy to expose parallelism

Serial Simulation Algorithms

Oblivious simulation



All parallel blocks evaluated on any change

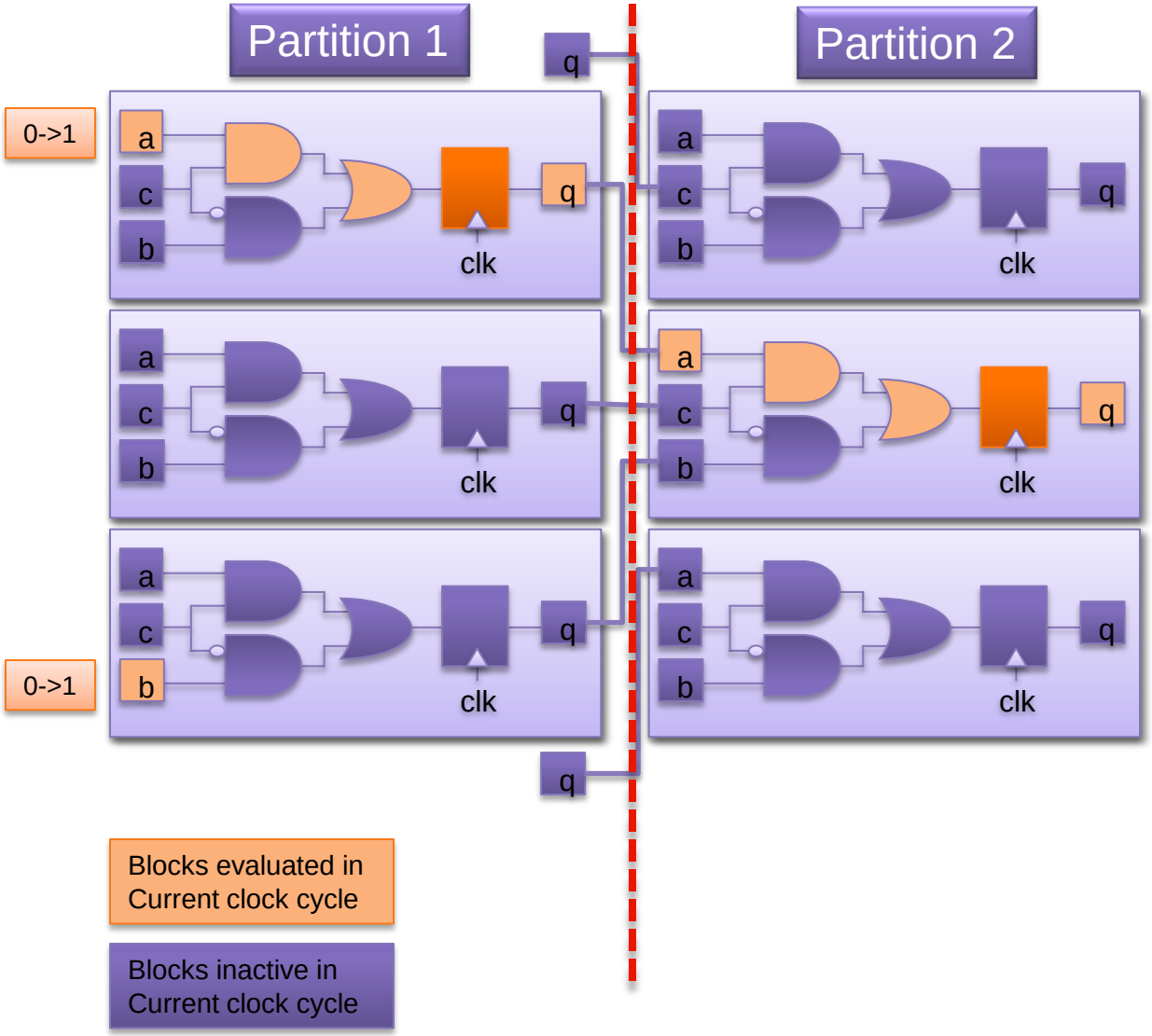
No event guards required: simplifies implementation

On the surface, *looks* suitable for *fine grained parallel* implementation

But: high overhead in the case of low event activity %

Parallel Simulation Algorithms

Task Based Parallelism



Design (and / or testbench) split into partitions

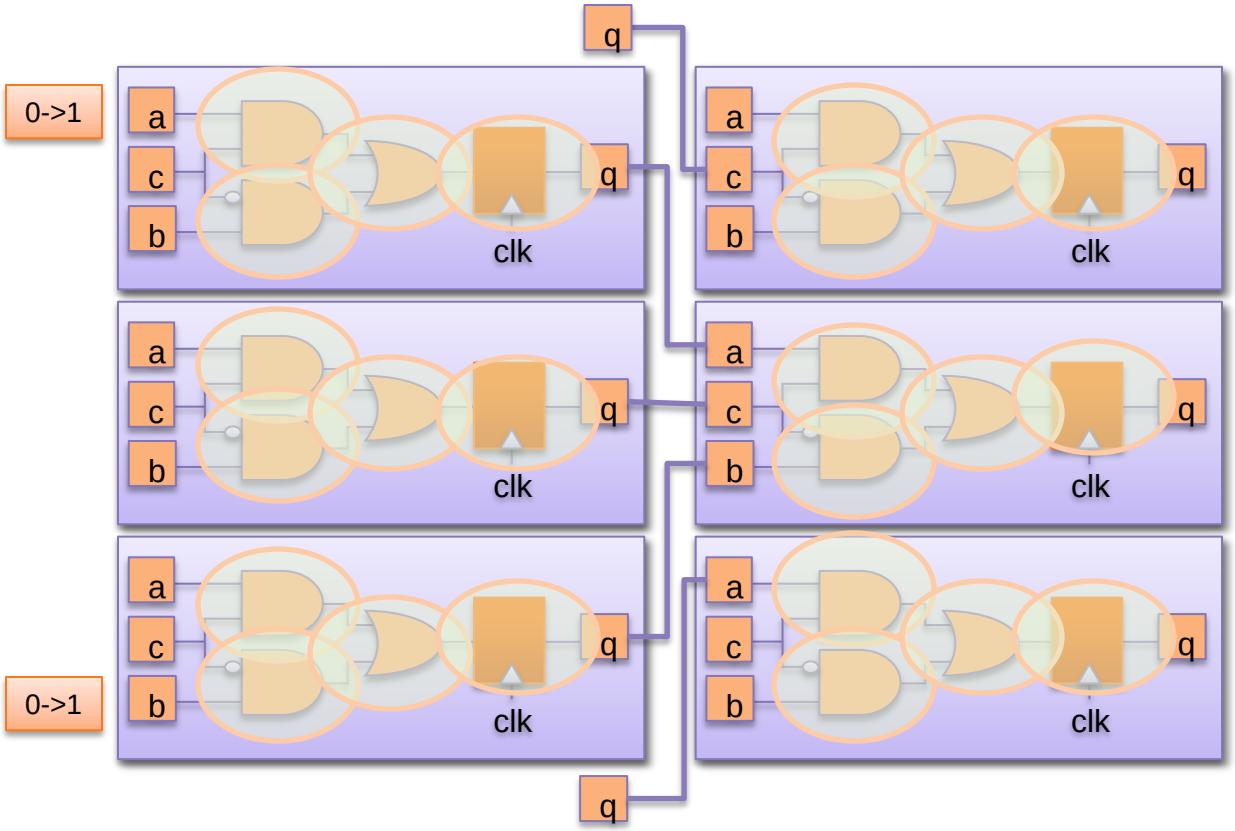
Each partition mapped into an execution unit

More suitable for bigger cores

Speedup depends on relative and balanced activity in partitions

Parallel Simulation Algorithms

Fine Grain Parallelism + Oblivious simulation



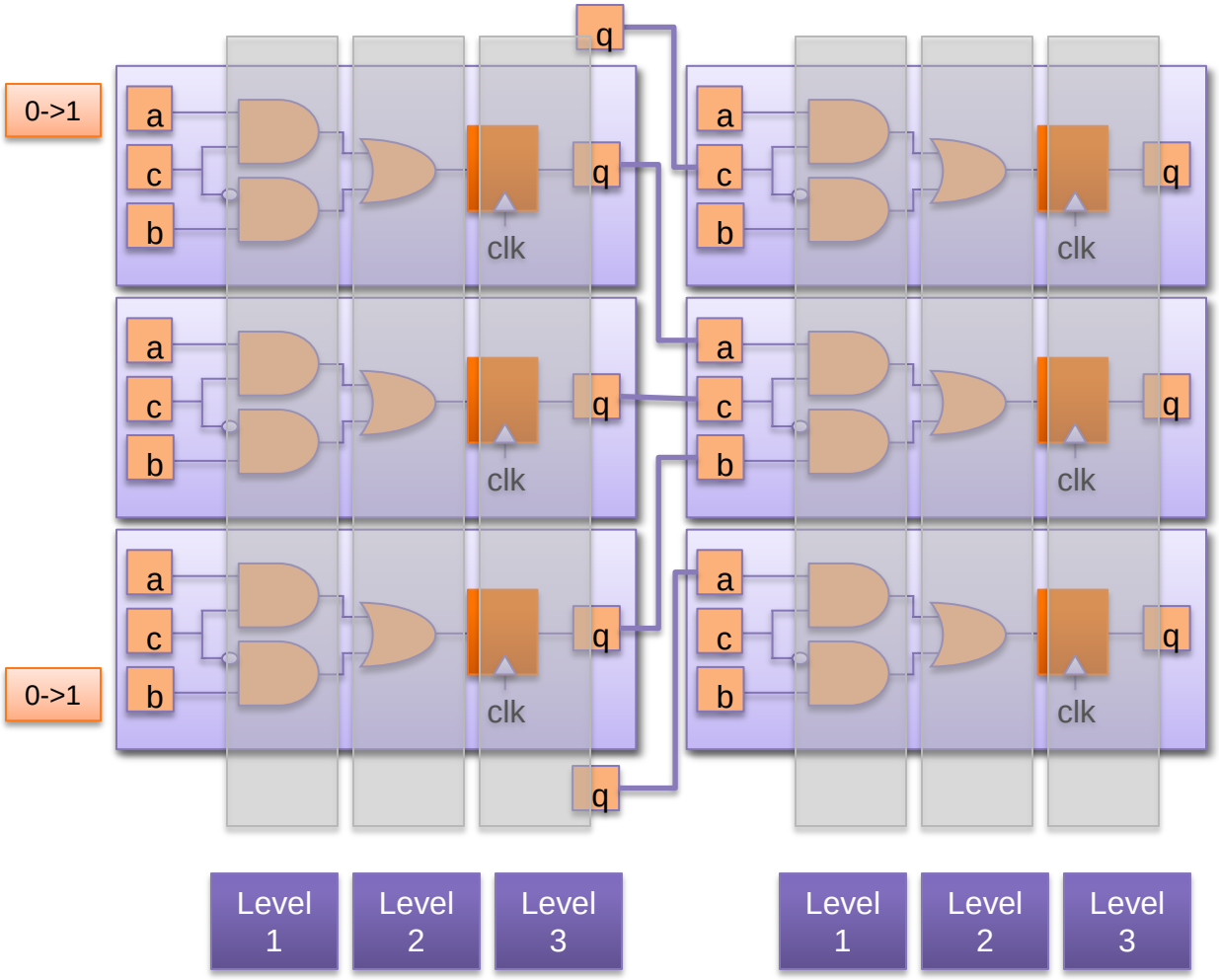
All parallel blocks evaluated on any change

No event guards required: simplifies implementation

Lots of Parallel Blocks

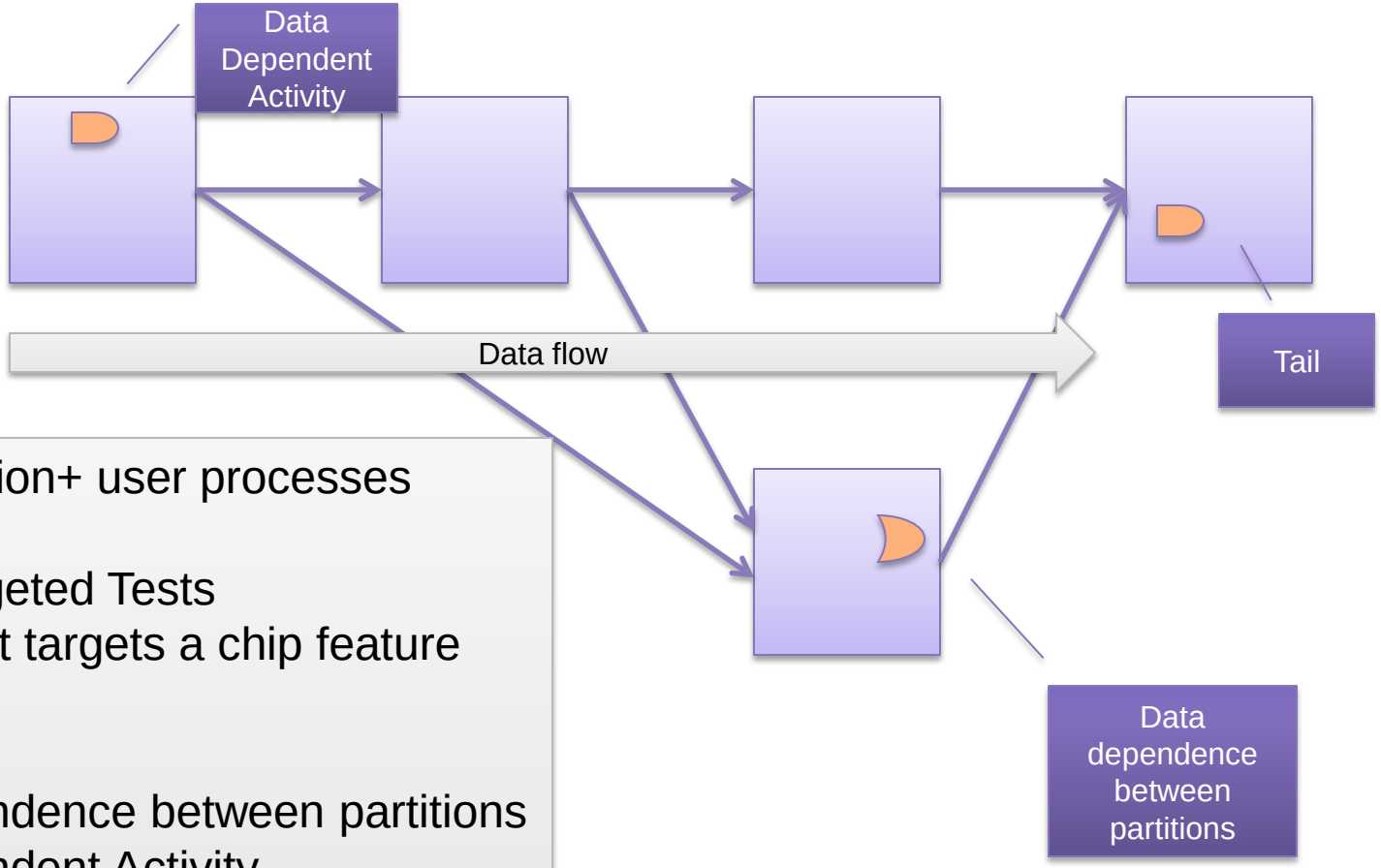
Parallel Simulation Algorithms

Mapping Oblivious simulation to GPU



Parallel Simulation Complexity

Hypothetical Medium Sized Chip RTL



Model: 1 Million+ user processes
Workload:
- 10K+Targeted Tests
- Each test targets a chip feature

Simulation
- Data Dependence between partitions
- Data Dependent Activity
  - 0.5 – 3% Activity per Test Phase
- Low Effective Parallelism
  - Activity
  - Tail Region

Summary

Challenges / Benefits in using GPU for RTL Sim

- What Works ?
  - A lot of potential parallelism in the model
    - Fermi / CUDA thread scheduling
  - A lot of memory accesses per clock cycle
    - Fermi provides 144 GBps
  - CUDA software ecosystem is robust and improving

- Wishlist
  - Global Barrier
  - Latency Optimized Core
  - Lower launch overhead
  - CUDA profiler for large datasets

References

- Mary L. Bailey, Jack V. Briner, Jr., and Roger D. Chamberlain. 1994. Parallel logic simulation of VLSI systems. *ACM Comput. Surv.* 26, 3 (September 1994), 255-294
- Keckler, S.W.; Dally, W.J.; Khailany, B.; Garland, M.; Glasco, D.; , "GPUs and the Future of Parallel Computing," *Micro, IEEE* , vol.31, no.5, pp.7-17, Sept.-Oct. 2011

Questions ?