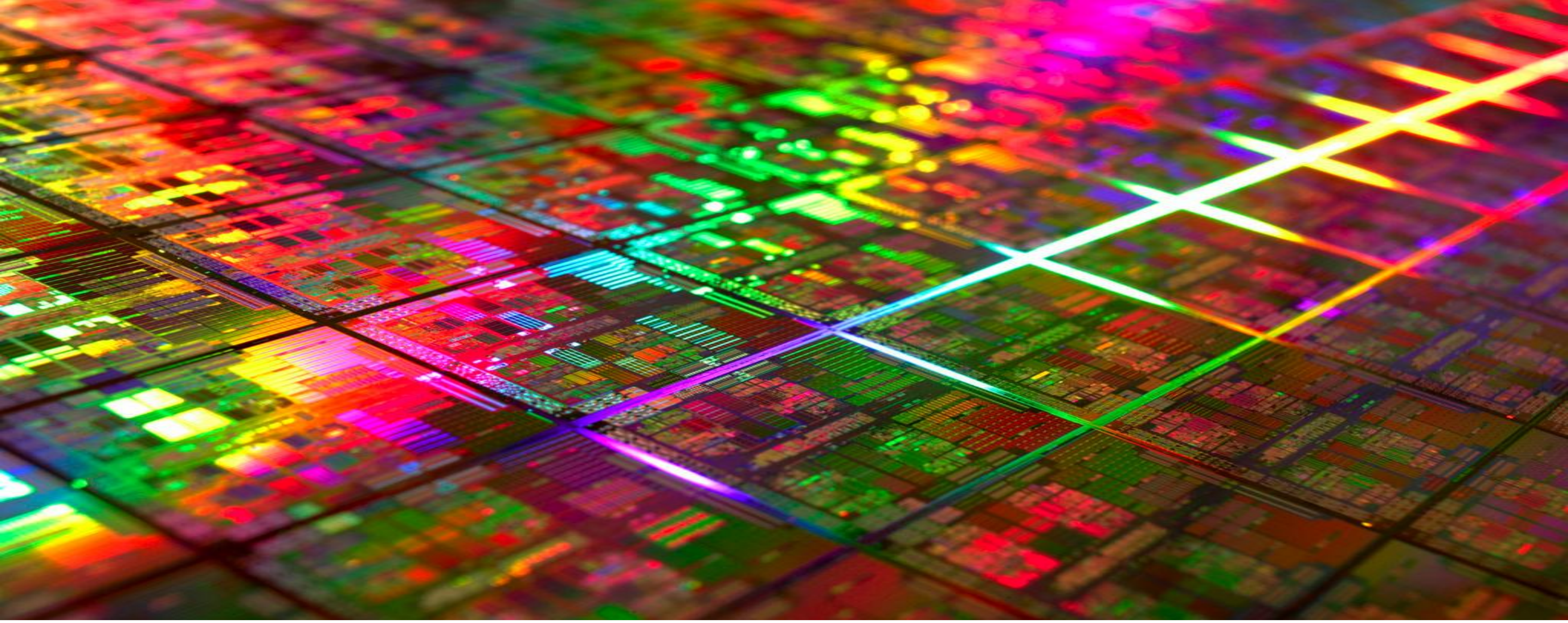
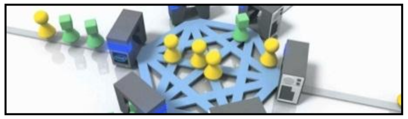
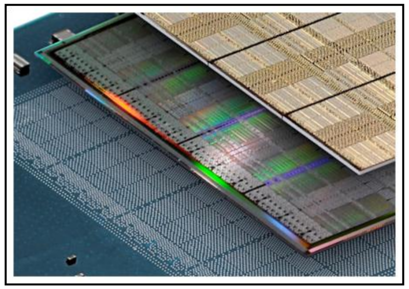
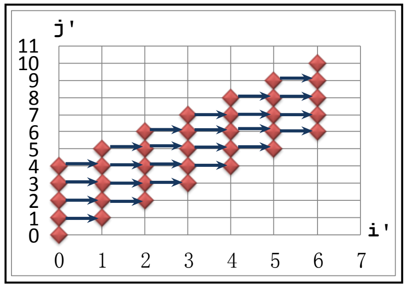


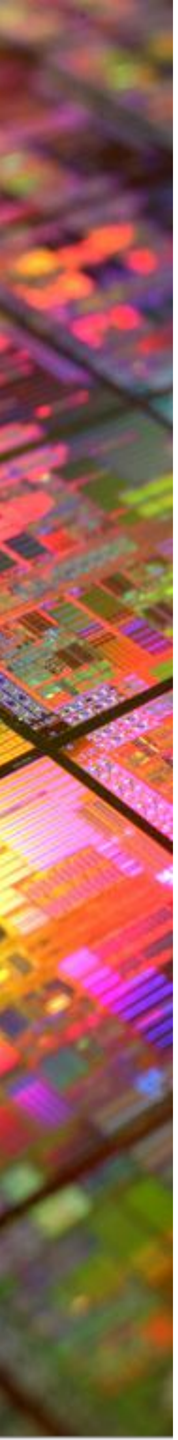
GPU Accelerated High Performance Logic Simulation of Integrated Circuits

Yangdong Steve Deng
dengyd@tsinghua.edu.cn
Institute of Microelectronics
Tsinghua University

Research Overview

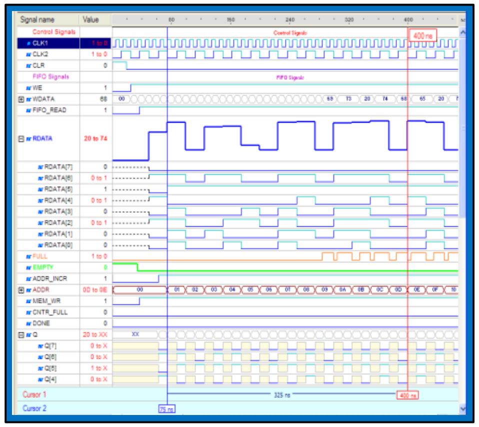
- Parallel algorithms/applications
  - Massively parallel simulation for System-on-Chips
  - Internet routing processing
- GPU architecture
  - Heterogeneous integration for routing processing
  - Supporting task-pipelined parallelism on GPUs
- Compiler
  - Polyhedral based automatic parallelization for GPUs
  - Automatic porting of ARM code to x86
- Sponsored by China National Key Projects, CUDA Center of Excellence, Tsinghua-Intel Center of Advanced Mobile Computing Technology, Intel University Program, NVidia Professor Partnership Award

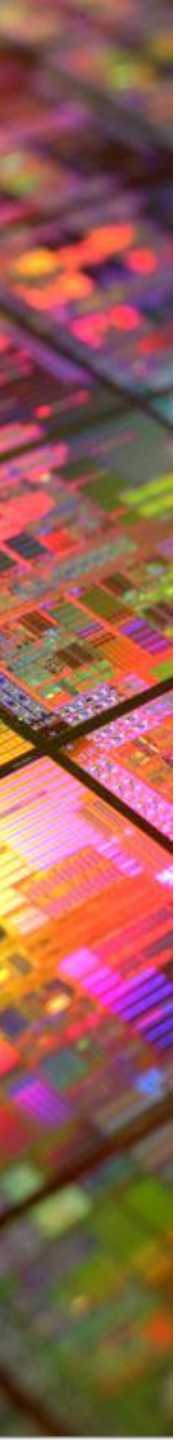


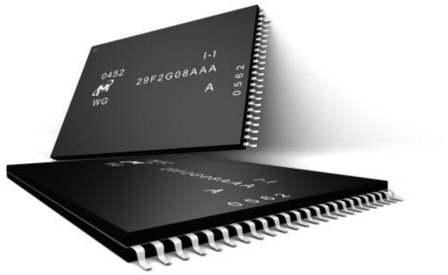
Outline

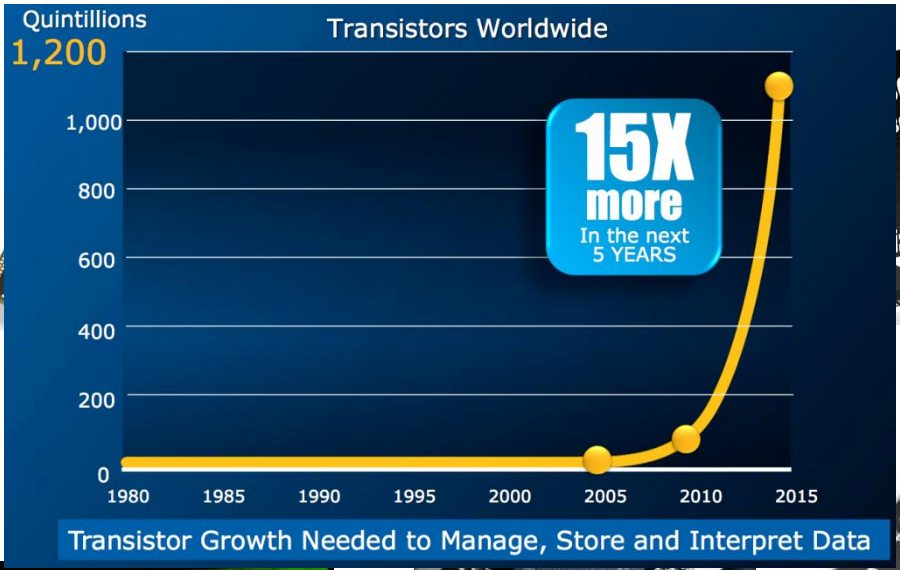
- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled HDL simulation
- Ongoing work

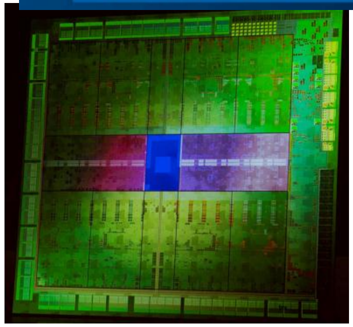


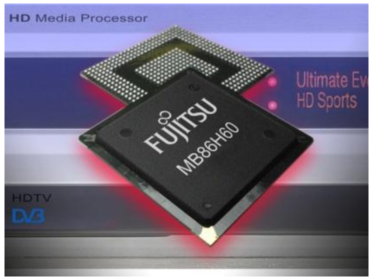
Integrated Circuits

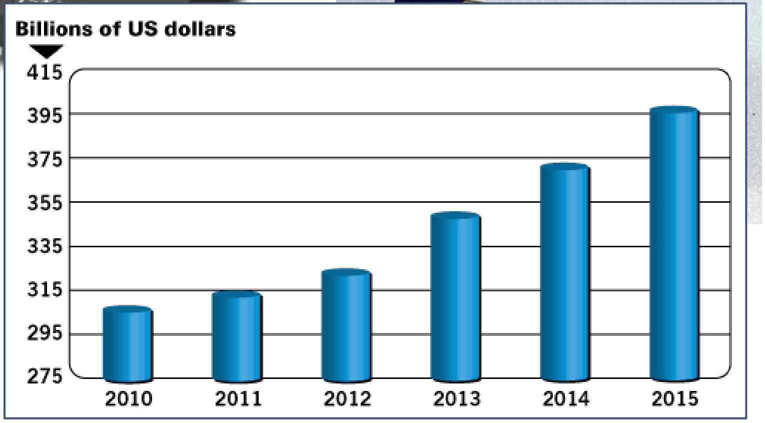


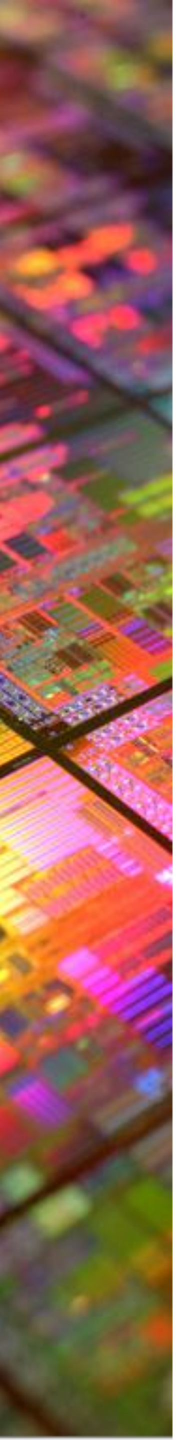




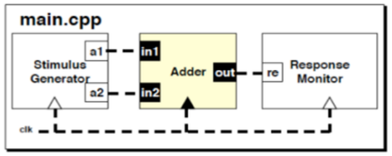


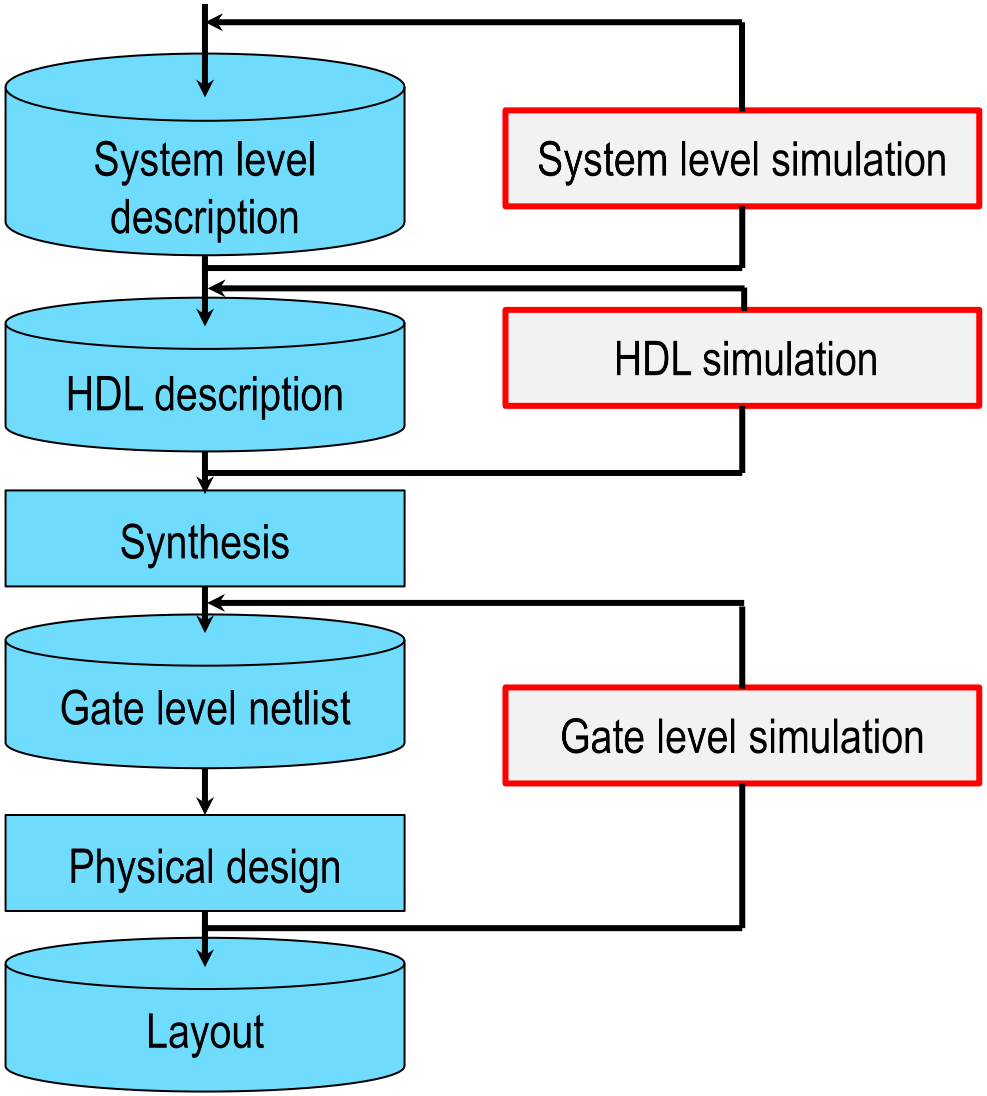


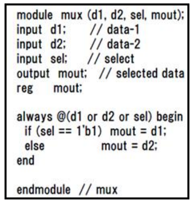




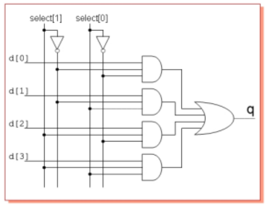
Simplified IC Design Flow

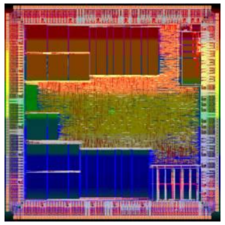


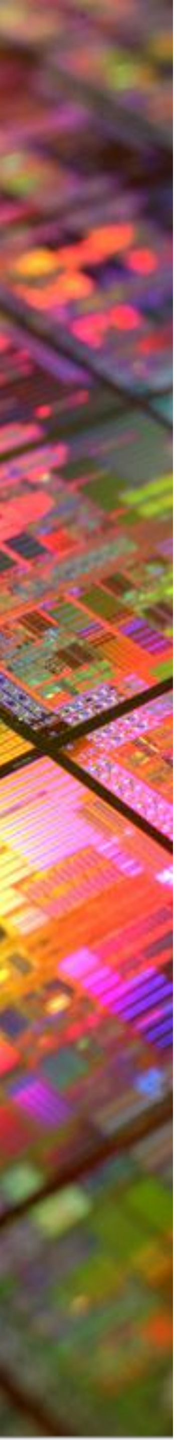




```
module  mux (d1, d2, sel, mout);
input  d1;      // data-1
input  d2;      // data-2
input  sel;     // select
output  mout;   // selected data
reg     mout;

always @(d1 or d2 or sel) begin
  if (sel == 1'b1)  mout = d1;
  else              mout = d2;
end

endmodule  // mux
```

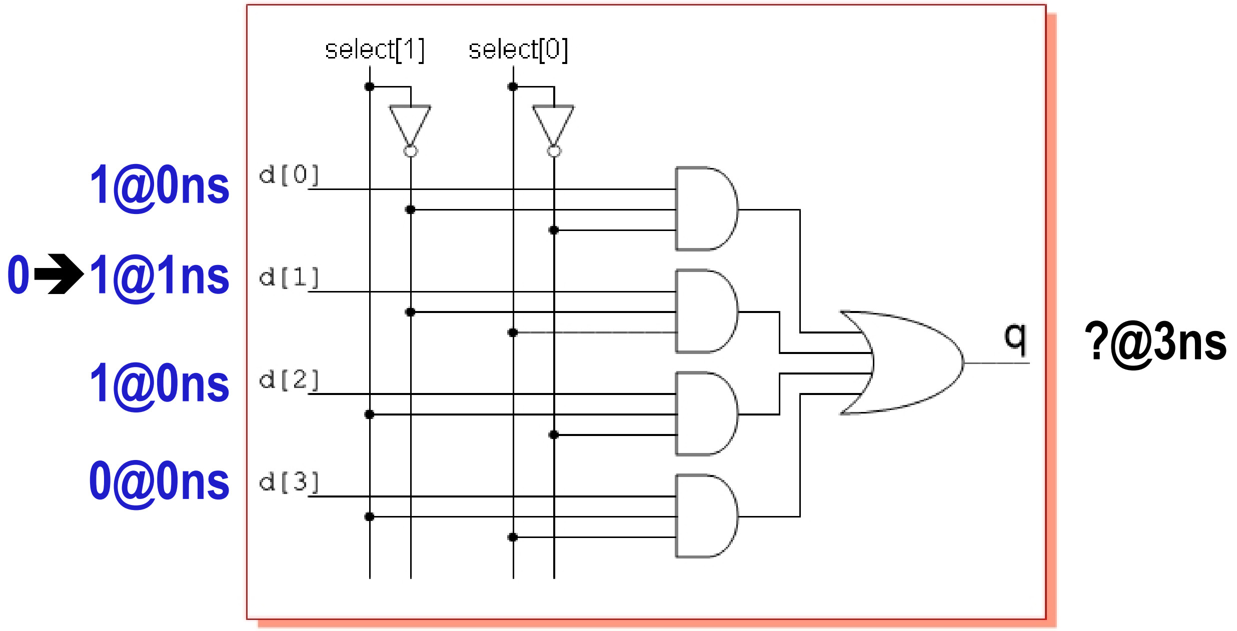


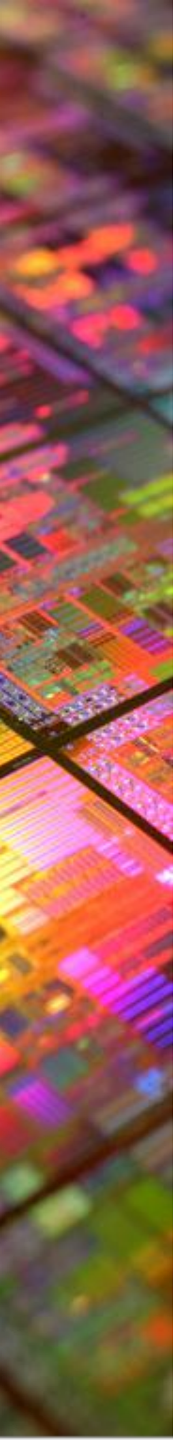




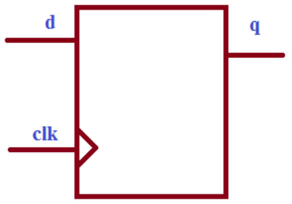
Logic Simulation

- Major method for IC design verification
  - Performed on a proper model of the design
  - Apply input patterns as stimuli
  - Observe output signals: satisfy design requirements?
  - The simulator evaluates the operation of the model





Complexity of Logic Simulation



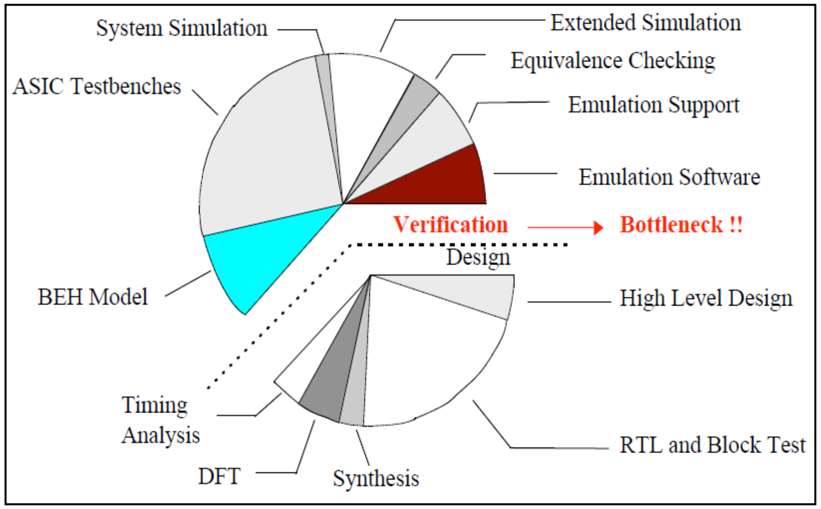
- For a 1-bit register with 1 input
  - 2 possible states and 2 possible inputs
  - Number of test patterns required = $2^1 x 2^1 = 4$
- For a Z80 microprocessor (~5K gates)
  - Has 208 register bits and 13 primary inputs
  - Possible state transitions = $2^{bits+inputs} = 2^{221}$
  - It takes 1 year to simulate all transitions at 1000MIPS
- For a chip with 500M gates
  - ????? years?

d	q	clk	q Next
0	0	↑	0
0	1	↑	0
1	0	↑	1
1	1	↑	1

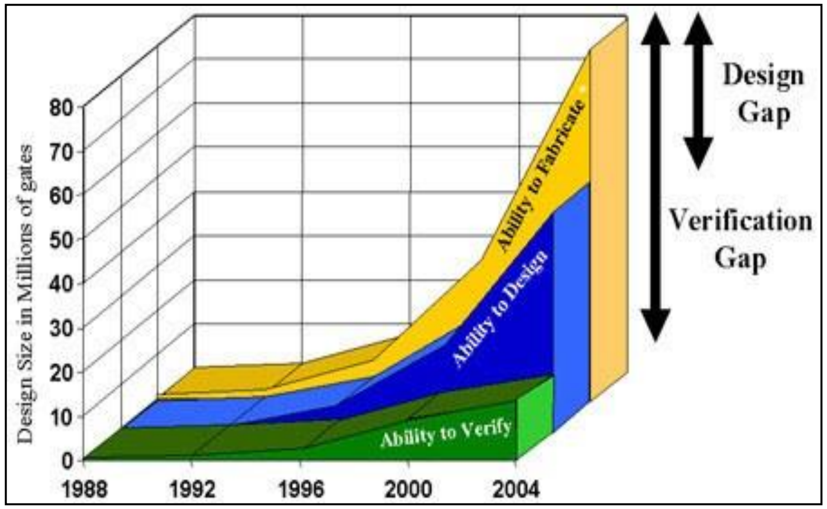
- No exhaustive simulation any more
- But has to meet a given coverage ratio

Verification Gap

- Functional verification has been the bottleneck!
  - Now consumes >60% of design turn-around time
  - Significantly lag behind fabrication capacity



Distribution of IC design effort

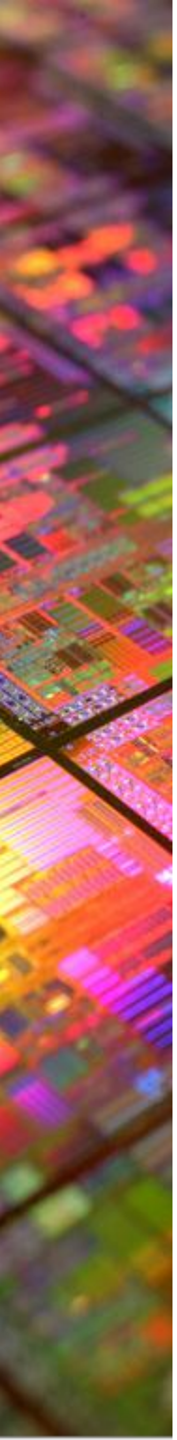


Widening design & verification gap

Problem

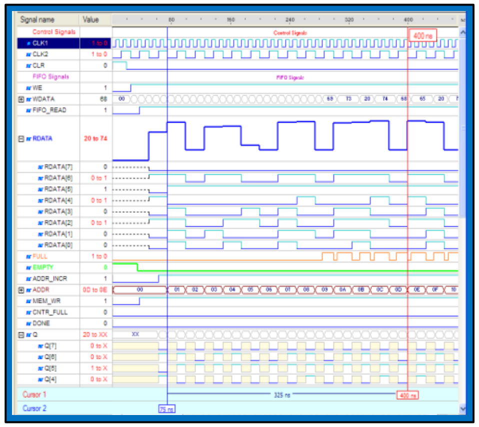
- Can we unleashing the power of GPUs for logic simulation?

$\xrightarrow{?}$



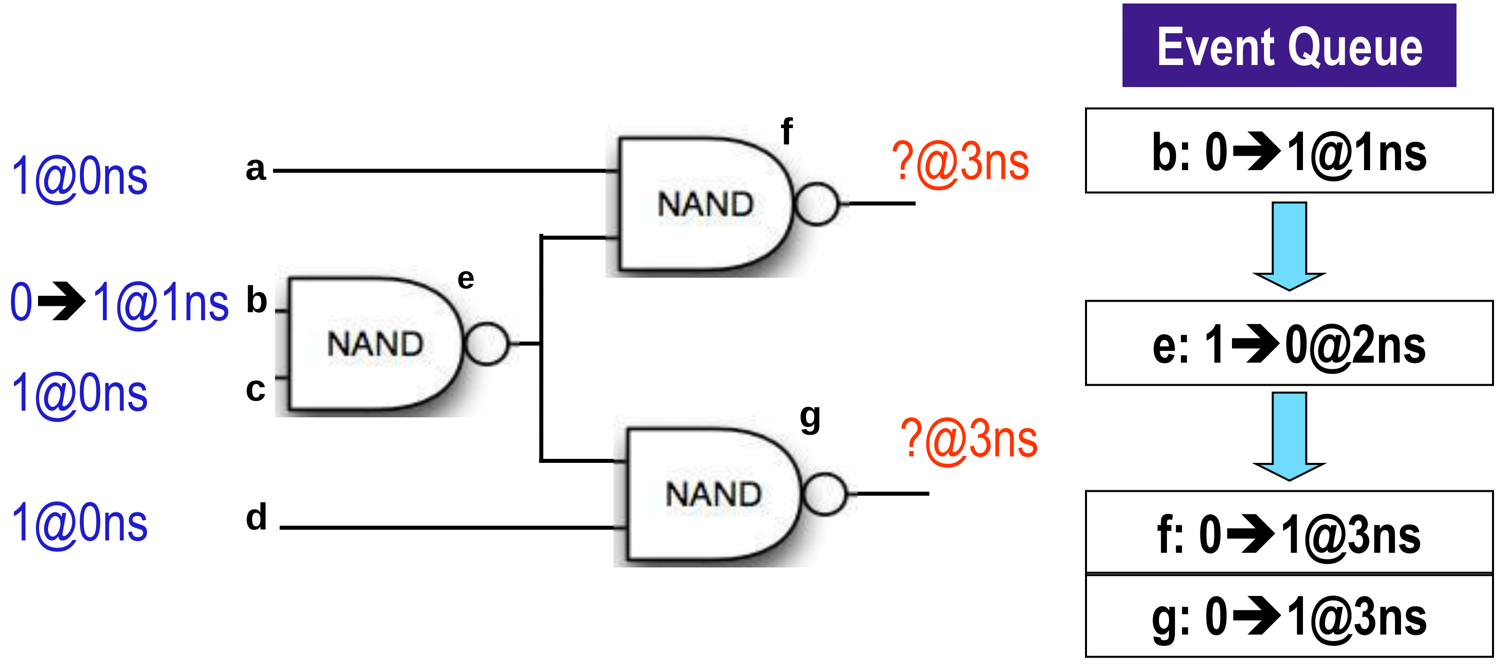
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled HDL simulation
- Ongoing work



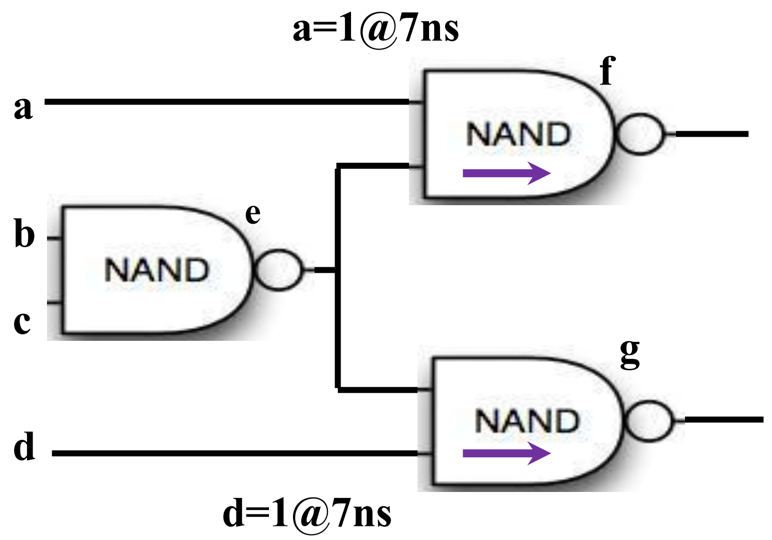
Event Driven Simulation

- Most used algorithm for discrete event systems
  - Event: state transition + time stamp
  - Always evaluate an event with the earliest stamp



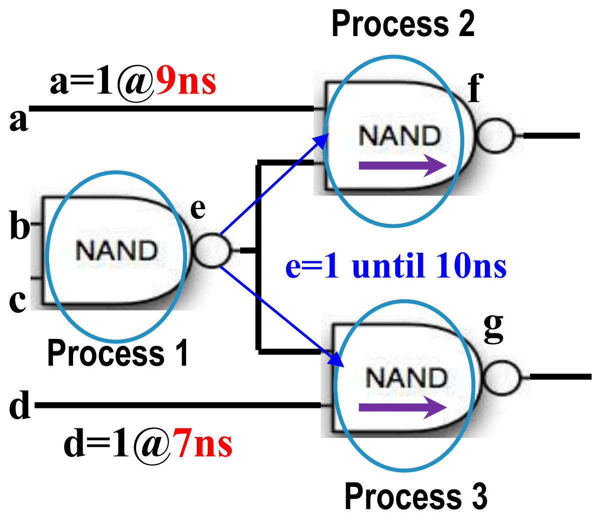
Synchronous Parallel Logic Simulation

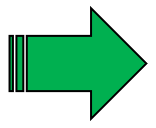
- Simultaneously process events with identical time-stamps
  - Insufficient parallelism
  - Generally <1% of a circuit has activities
  - Especially for timed simulation



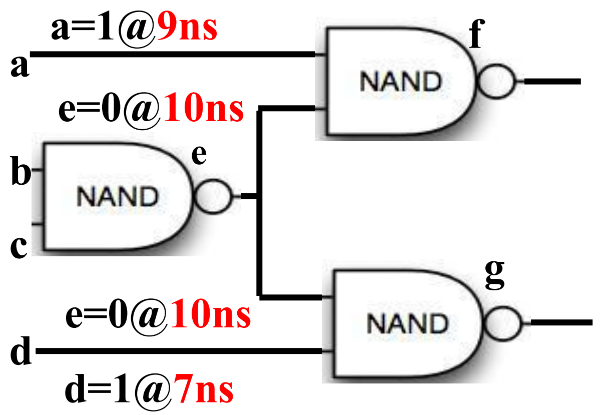
Asynchronous Parallel Logic Simulation

- Conservative simulation
  - Chandy-Misra-Bryant (CMB) algorithm
  - Send events as messages





a=1@9ns
e=0@10ns

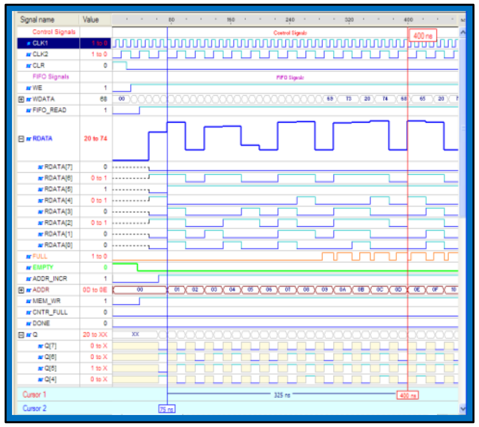


d=1@7ns
e=0@10ns

- Optimistic simulation
  - Look-ahead simulation

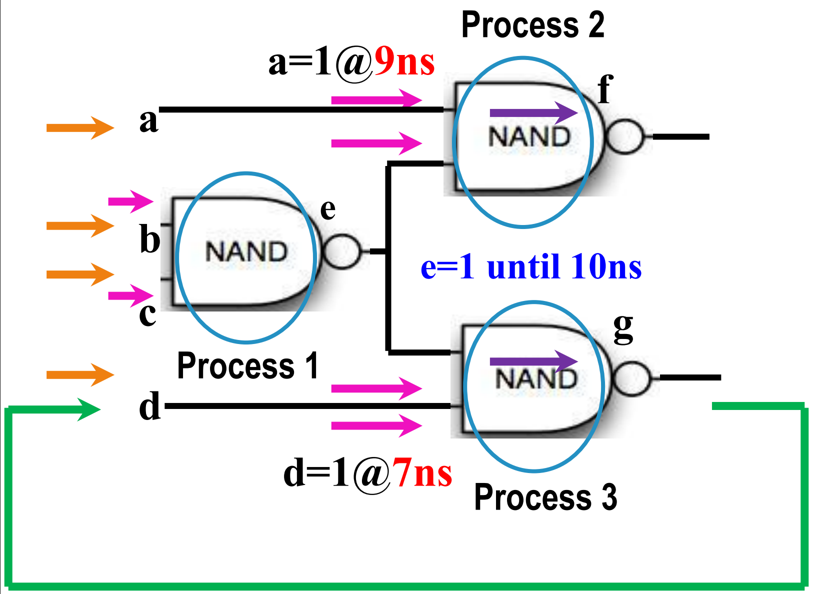
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled HDL simulation
- Ongoing work



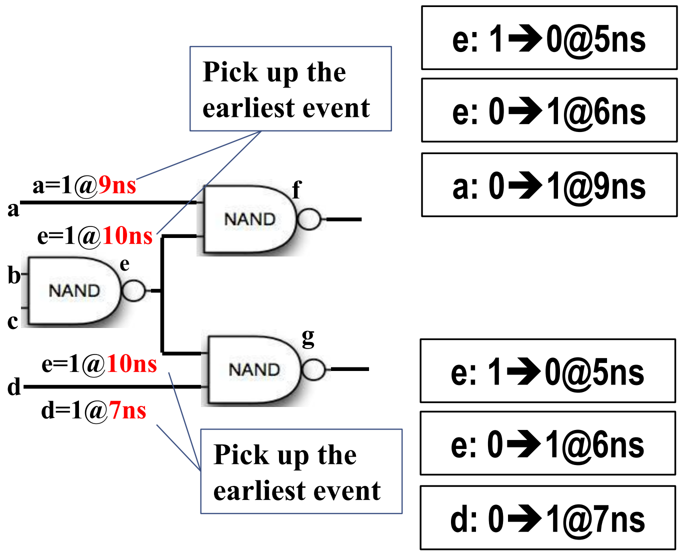
GPU Based CMB Parallel Simulation Flow

```
while not finished
  // kernel 1: primary input update
  for each primary input(PI) do
      extract the first message in the PI queue;
      insert the message into the PI output array;
  end for
  // kernel 2: input pin update
  for each input pin do
     insert messages from output to input pins;
  end for
  // kernel 3: gate evaluation
  for each gate do
      extract the earliest message from its pins;
      evaluate messages and update gate status;
      write the gate output to the output array;
  end for
end while
```



Data Structure: Distributed Event Queues

- Classical CMB allocates an event queue to each gate
  - Need sorting when insertion
  - Hard to maintain on GPU

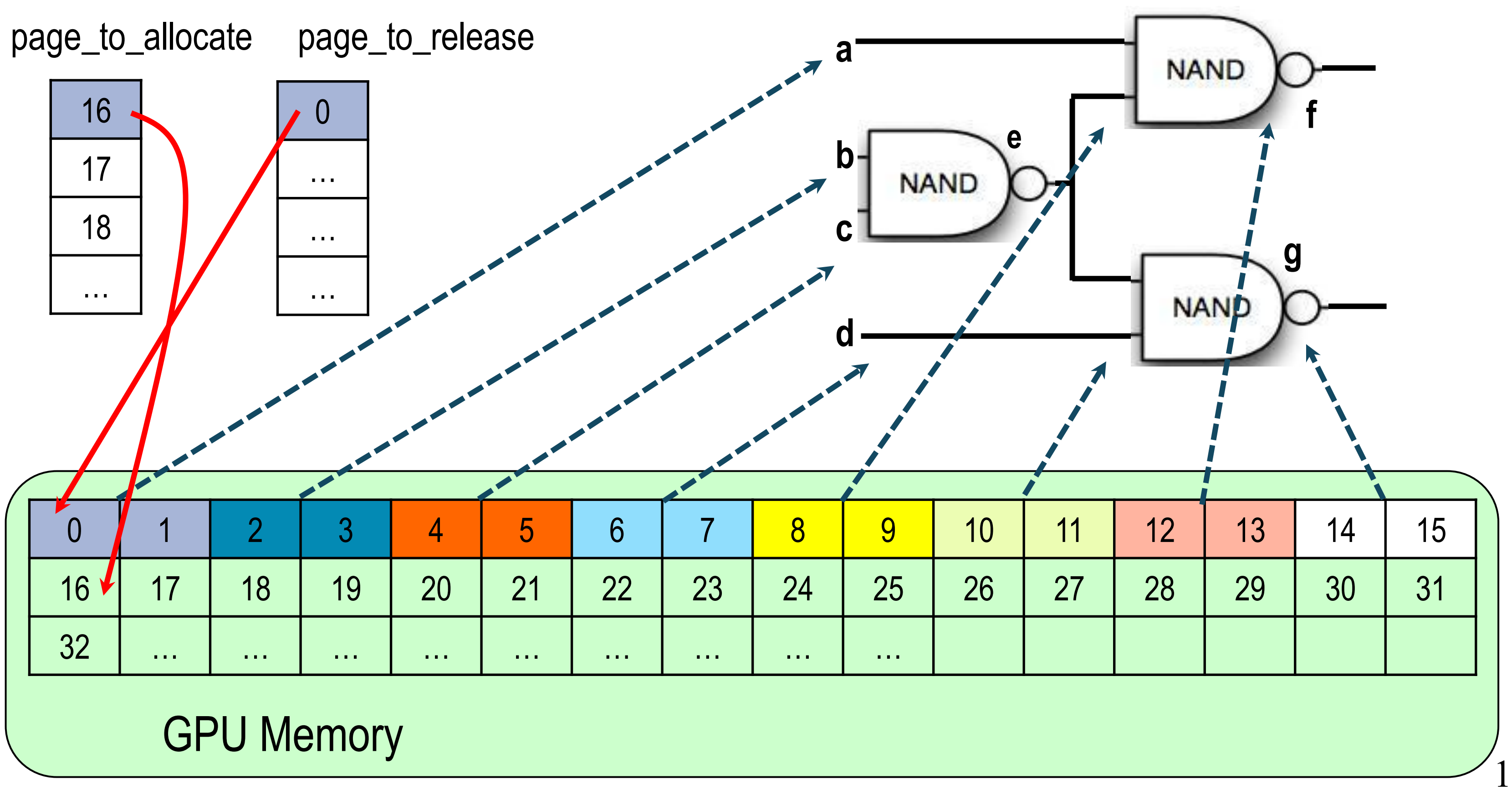


- Our work: every input pin of a gate has its own event queue
  - A queue is naturally ordered
  - Irregular distribution of required queue sizes

Peak # Events on pins	Circuit 1	Circuit 2	Circuit 3
0~99	34444	26106	158086
100~9999	737	1764	40
>=10000	3	3	1

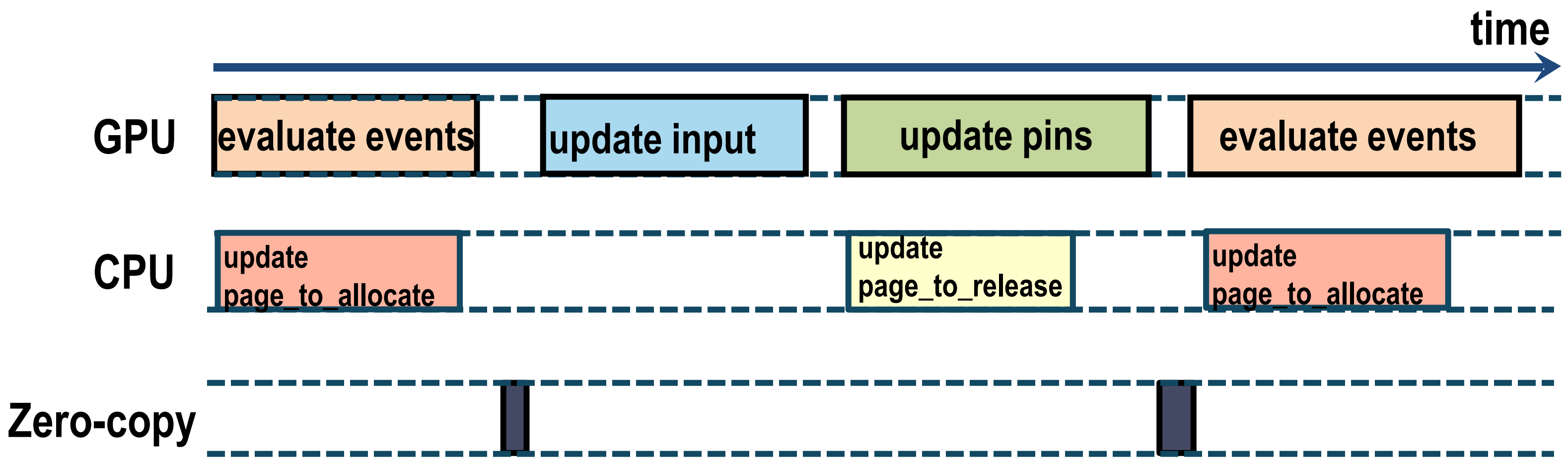
Dynamic GPU Memory Management

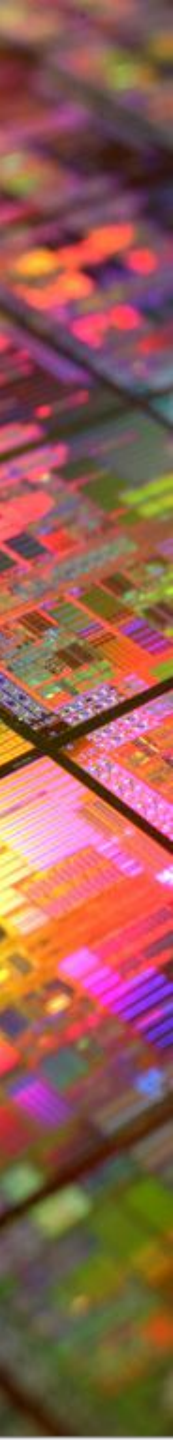
- A dynamic memory allocator for GPU!
  - Proposed earlier than NVIDIA's GPU memory allocator



Dynamic GPU Memory Management

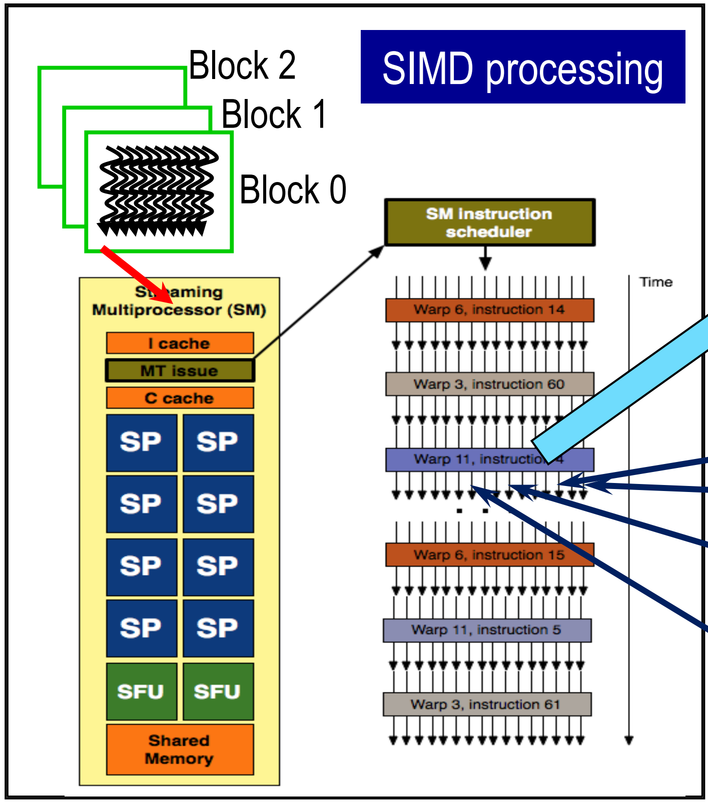
- CPU-GPU collaborated memory management
  - Overlap *update_pin*(GPU) and *update page_to_release*(CPU)
  - Overlap *evaluate_gate*(GPU) and *update page_to_allocate*(CPU)
  - Exploit *zero-copy* to transfer memory maintenance information



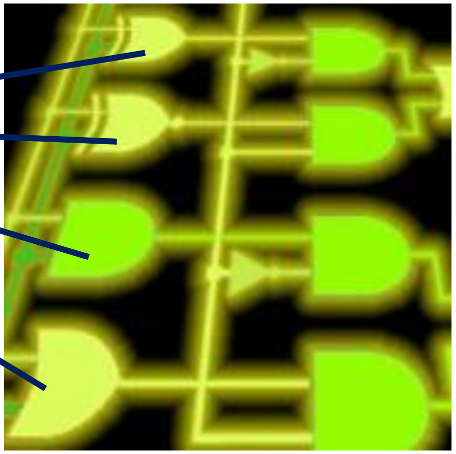


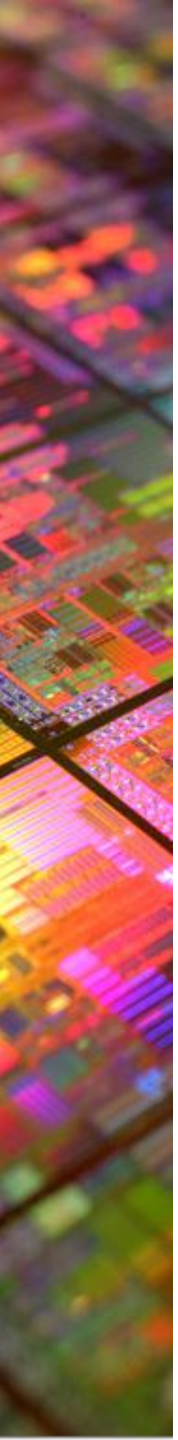
Parallel Organization of Event Evaluation

- A thread works on a gate
- Event evaluation through truth-table lookup
- Try assigning gates of the same type into a single warp



	a	b	y
NAND	0	*	1
	…	…	…
NOR	1	*	0
..	…	…	…



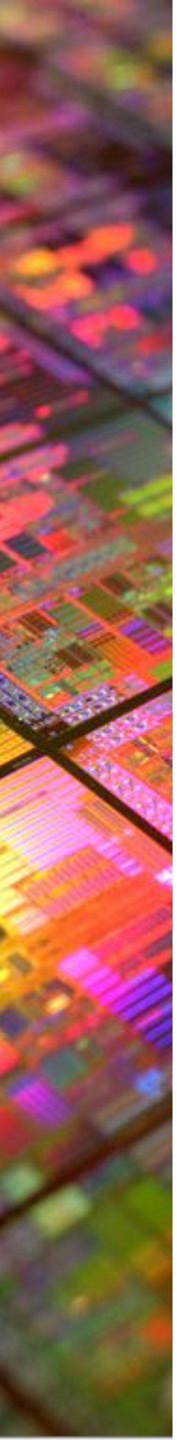


Gate Level Simulation Results

- World's fastest logic simulator on general purpose hardware[1]
- 30X speed-up on average[2] (100X for random patterns)
  - 1 month on CPU vs. 1 day on GPU

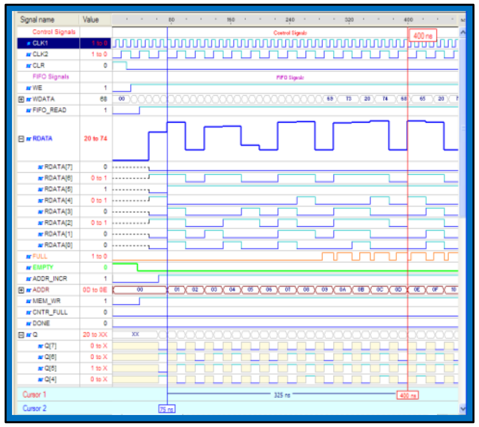
Design	Baseline Simulator[1] (s)	GPU-Based Simulator (s)	Speedup
AES	109.90	4.45	**24.70**
DES	183.11	4.50	**40.66**
SHA	56.66	0.41	**138.20**
R2000	9.20	3.15	**2.92**
JPEG	136.33	43.09	**3.16**
NOC	5389.42	347.95	**15.49**
M1	118.48	22.43	**5.28**

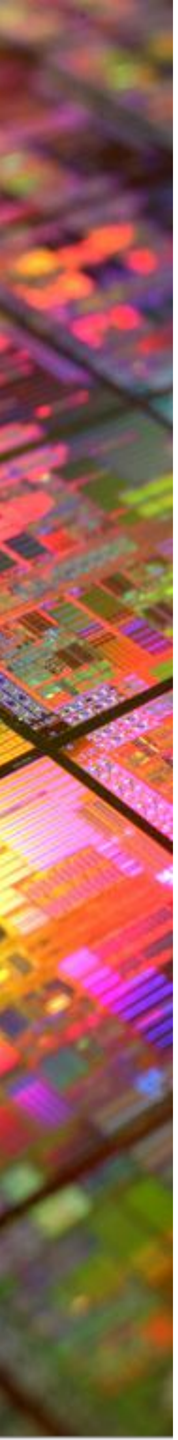
[1]Published on DAC 2010 and ACM Trans. on Design Automation 2011
[2]In-house simulator (~20% faster than Synopsys VCS)



Outline

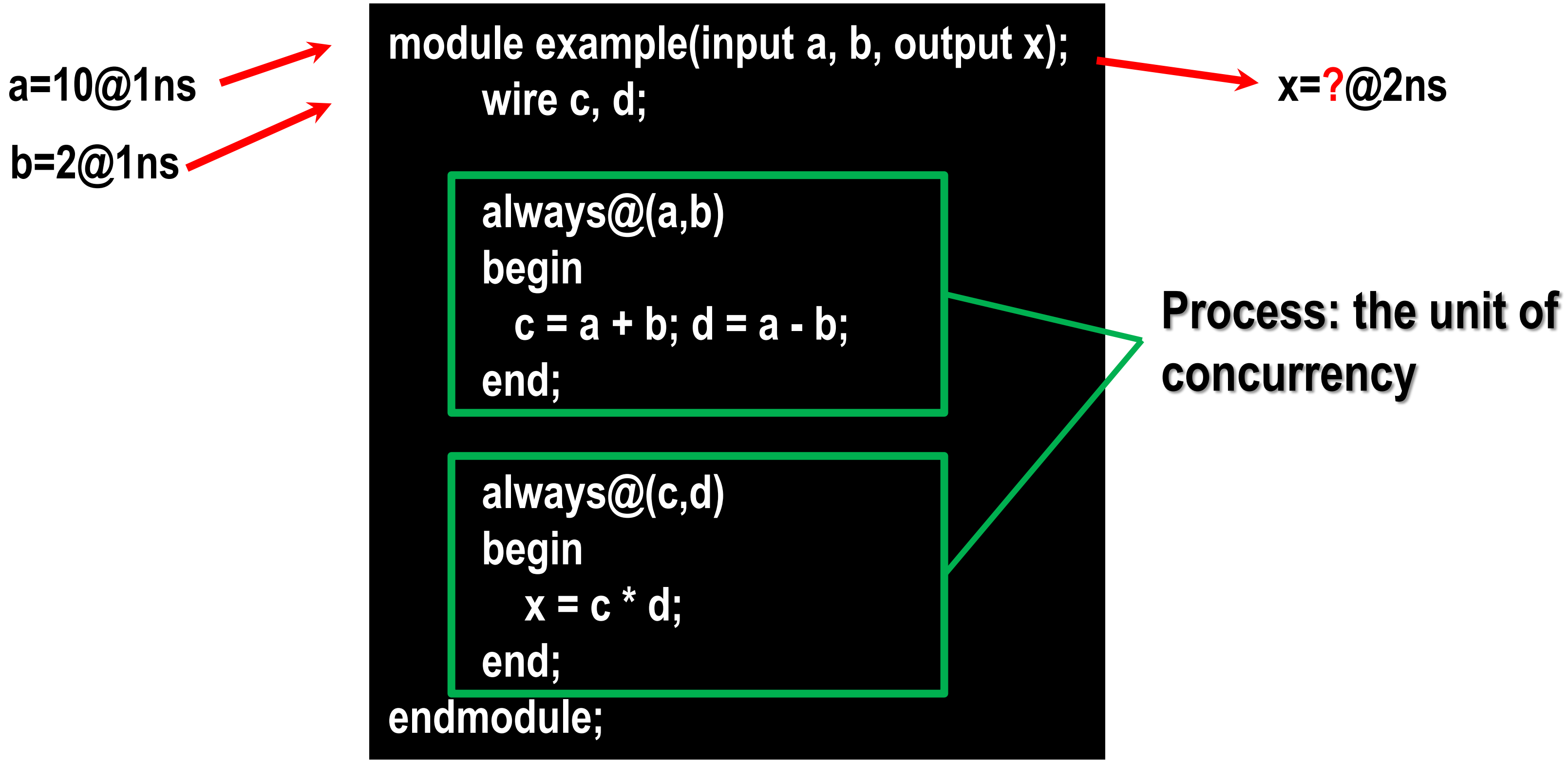
- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled HDL simulation
- Ongoing work





Simulation of Hardware Description

- Hardware Description Language (HDL)
  - Computer languages to describe a circuit’s operation, its design and organization, and tests to verify its operation by means of simulation

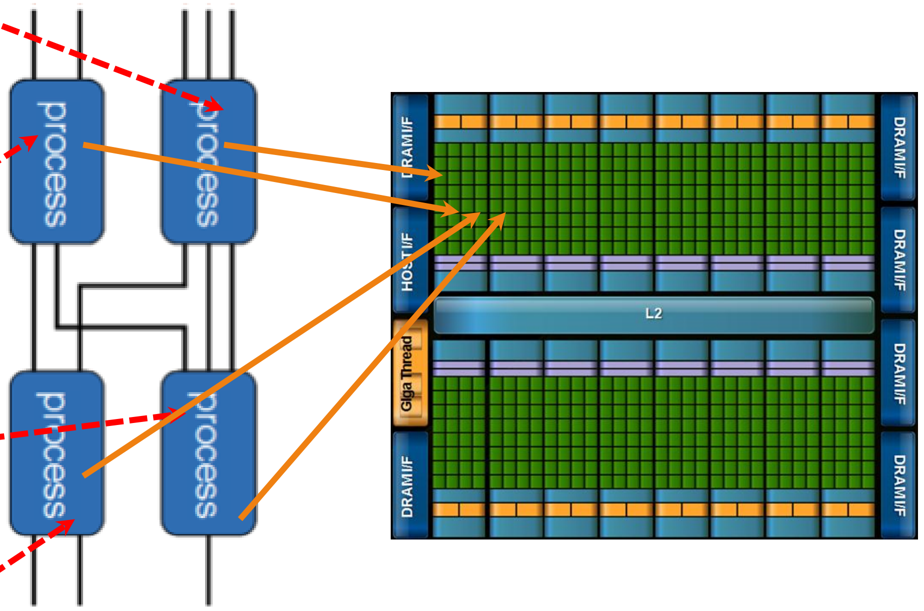


```
module example(input a, b, output x);
        wire c, d;

        always@(a,b)
        begin
           c = a + b; d = a - b;
        end;

        always@(c,d)
        begin
            x = c * d;
        end;
endmodule;
```

Parallel HDL Simulation

```
always@(a,b)
begin
  f = a + b; g = a - b;
end;

always@(c,d,e)
begin
   h = c + d; i = c + e; k = d - e;
end;

always@(f,h)
begin
  x = f * h; y = f + h;
end;

always@(posedge clk)
begin
  z <= x || y;
end;
```

Verilog HDL



Challenges

- Verilog HDL as input
  - Similar to C but has a hardware oriented semantics
- Challenge 1:
  - Arbitrarily complex behavior in a Verilog process
  - Cannot be captured by truth tables
- Challenge 2:
  - Divergent behaviors among different processes and different sensitizations of a single process

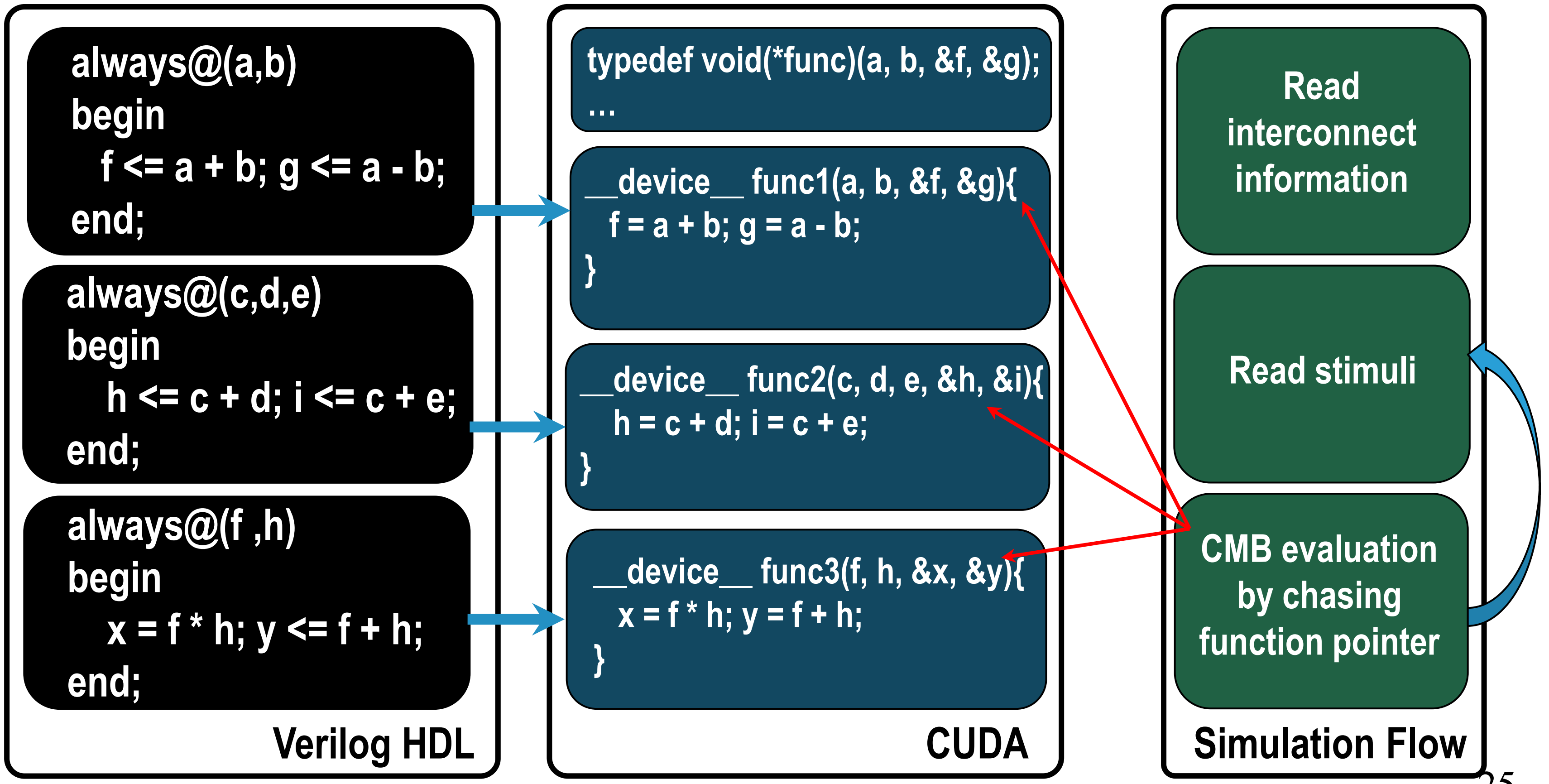
Verilog HDL

```
always@(posedge clk)
begin
    if (en == 1)
       q = d;
   end
end;
```

```
always@(a, b, op)
begin
    if(op == 0)
         sum <= a + b;
    else
         sum <= a - b;
    end
end;
```

Compiled Simulation

- Translating Verilog into equivalent CUDA code
  - Also merge simple processes



Verilog HDL

```
always@(a,b)
begin
   f <= a + b; g <= a - b;
end;
```

```
always@(c,d,e)
begin
   h <= c + d; i <= c + e;
end;
```

```
always@(f ,h)
begin
   x = f * h; y <= f + h;
end;
```

CUDA

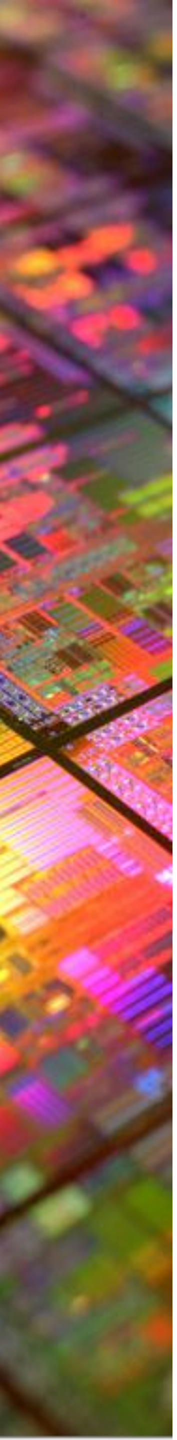
```
typedef void(*func)(a, b, &f, &g);
…
```

```
__device__ func1(a, b, &f, &g){
  f = a + b; g = a - b;
}
```

```
__device__ func2(c, d, e, &h, &i){
  h = c + d; i = c + e;
}
```

```
__device__ func3(f, h, &x, &y){
  x = f * h; y = f + h;
}
```

Simulation Flow
Read interconnect information
Read stimuli
CMB evaluation by chasing function pointer

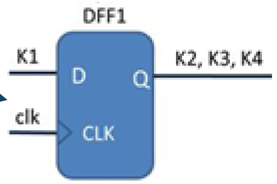


Handling the Semantic Differences

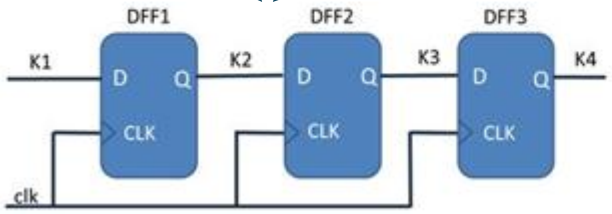
- Careful on the semantic differences between HDL and CUDA!

```
always@(clk)
begin
  K2 <= K1;
  K3 <= K2;
  K4 <= K3;
end;
```

```
If (clk = posedge){
  K2 = K1;
  K3 = K2;
  K4 = K3;
}
```



```
If (clk = posedge){
  K4 = K3;
  K3 = K2;
  K2 = K1;
}
```



Non-blocking to blocking statements

```
always@(clk)
begin
  K2 <= K1;
  K1 <= K2;
end;
```

```
If (clk = posedge){
{
  K3 = K2;
  K2 = K1;
  K1 = K3;
end;
```

Value swapping

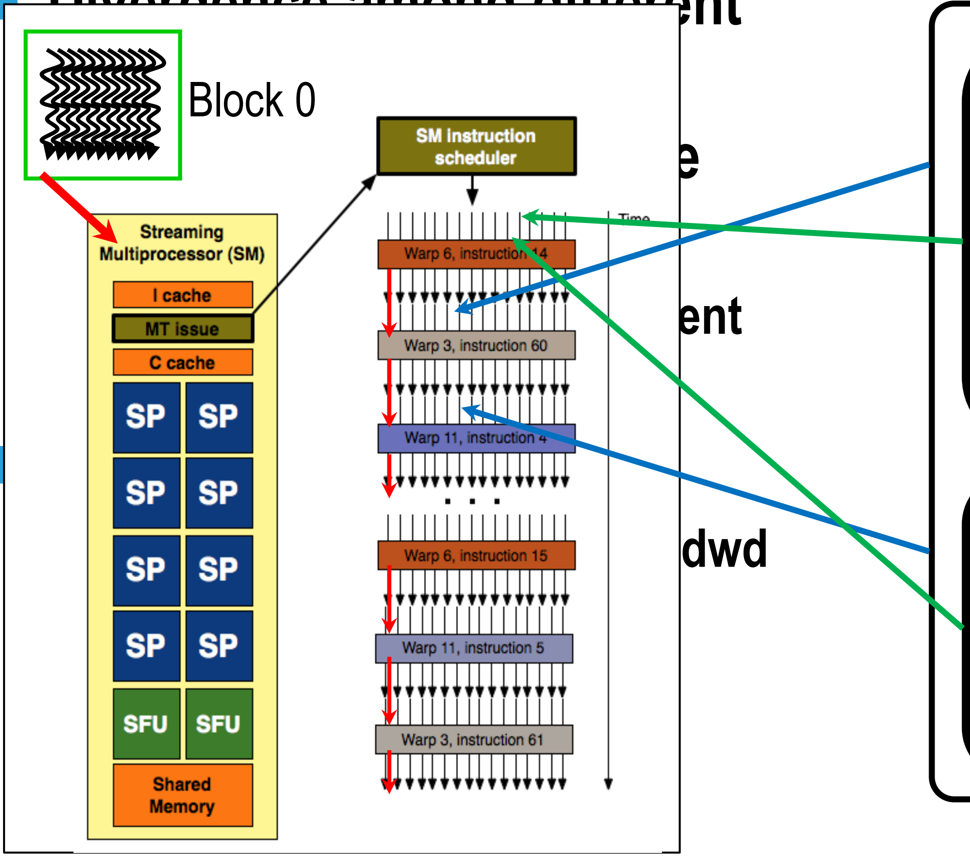
Super-Multithreaded Execution

- Divergence among different

e

ent

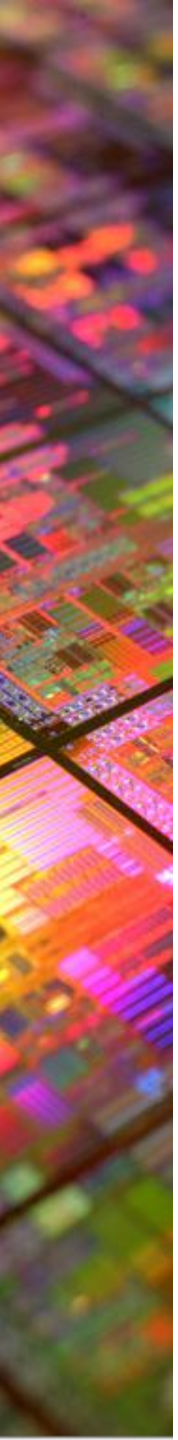
dwd



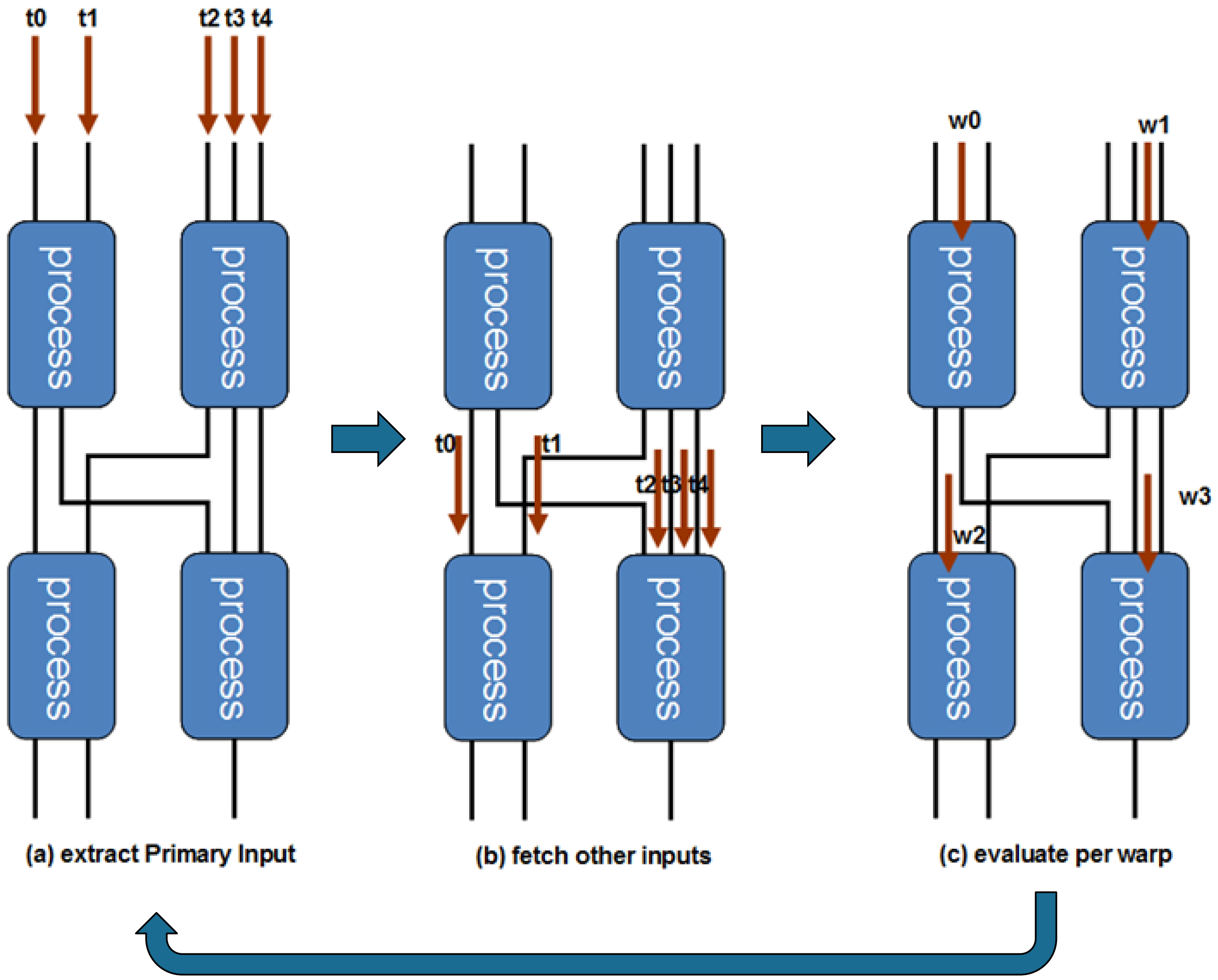
Verilog HDL

```
always@(posedge clk)
begin
   if (en == 1)
      q <= d;
  end
end;
```

```
always@(a, b)
begin
   sum <= a + b;
end;
```



CMB Based HDL Simulation



HDL Simulation Results

- **20-50X speed-up depending on circuit structure**

Design	Mentor Graphics ModelSim (s)	GPU-Based Simulator (s)	Speedup
Mux network for hierarchical bus	83.32	4.019	**20.73X**
Adder network for large numbers	258.47	10.21	**25.32X**
ASIC(des)	178.66	3.54	**50.47X**
ASIC(aes)	104.63	2.67	**39.19X**

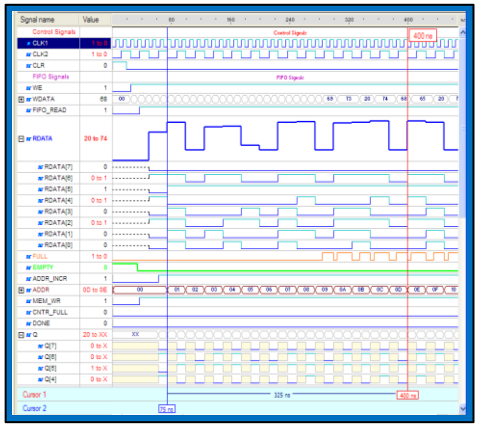
*Published on ICCAD11 and Integration, the VLSI Journal

Related Publications

- H. Qian and Y. Deng, “Accelerating RTL Simulation with GPUs,” IEEE/ACM International Conference on Computer-Aided Design, Nov. 2011.
- Y. Zhu, B. Wang, and Y. Deng, “Massively Parallel Logic Simulation with GPUs, ” ACM Transaction on Design Automation of Electronics Systems, Vol.16, No.3, June, 2011.
- B. Wang, Y. Zhu, and Y. Deng, “Distributed Time, Conservative Parallel Logic Simulation on GPUs,” IEEE/ACM Design Automation Conference, Jun. 2010.

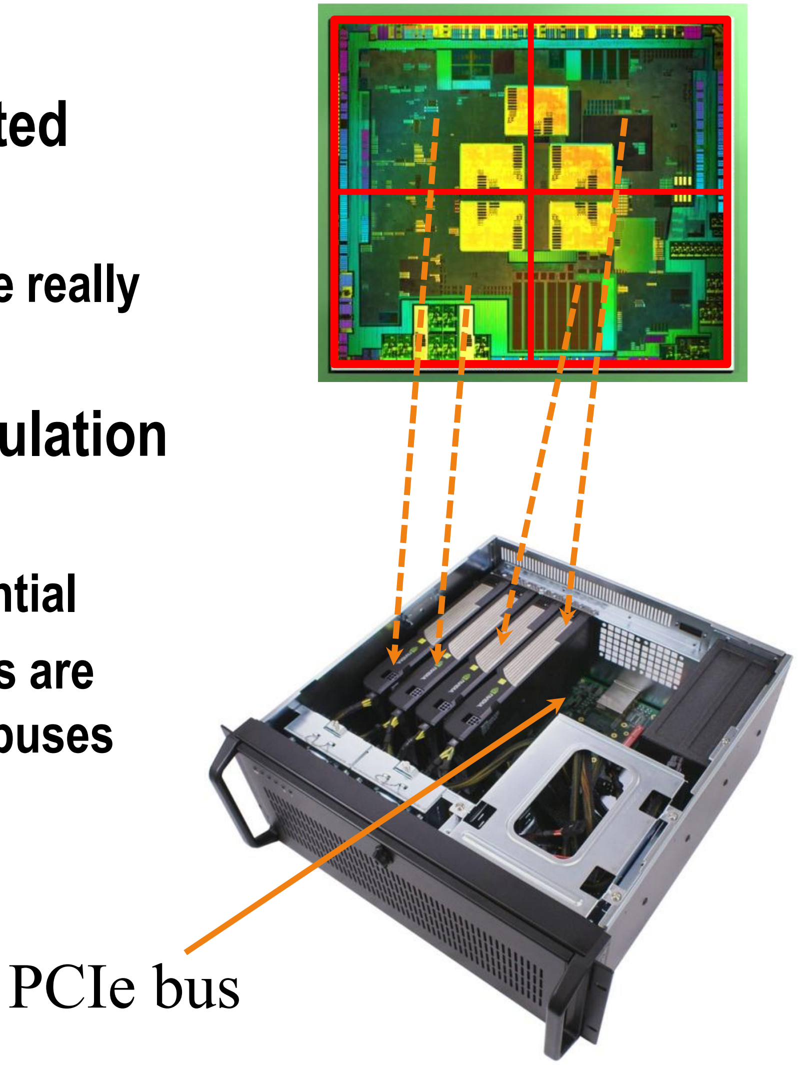
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled HDL simulation
- Ongoing work



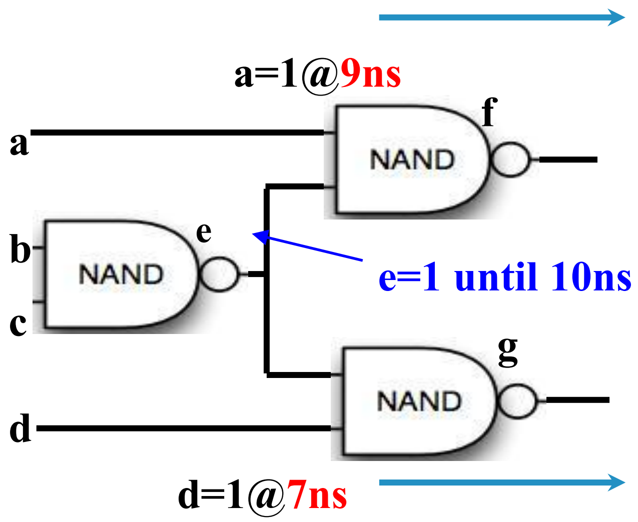
Ongoing Work 1: Scalable Massively Parallel Simulation

- Now we have GPU-accelerated logic simulators
  - But single GPU cannot handle really big designs
- How to scale GPU logic simulation to billion-transistor ICs?
  - Multi-GPU simulation is essential
  - However, fine-grain messages are hard to manage across PCIe buses

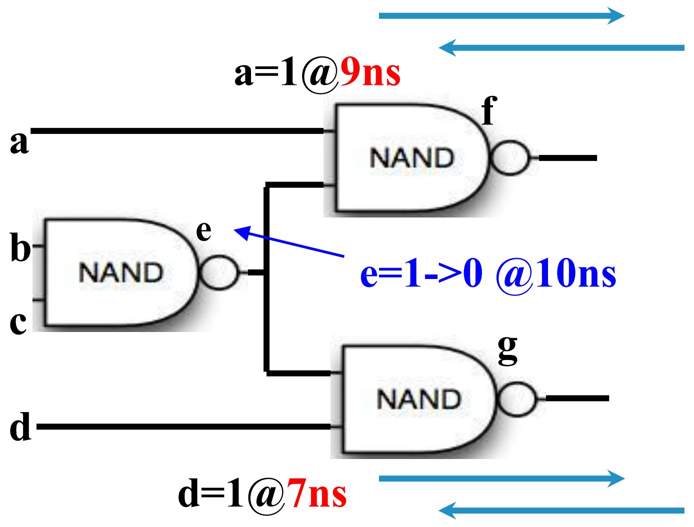


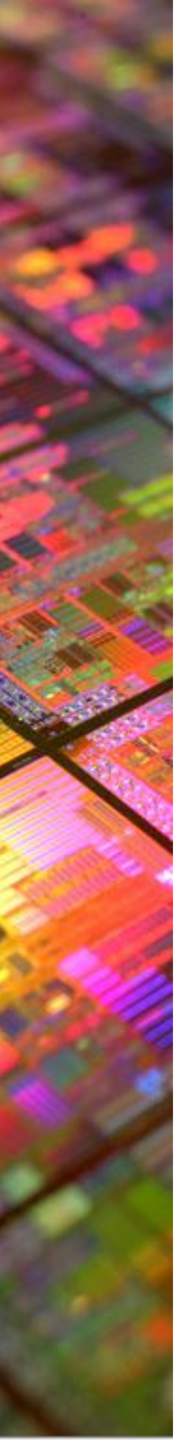
Hybrid Multi-GPU Based Logic Simulation

- Intra-GPU
  - Conservative simulation (i.e. CMB)



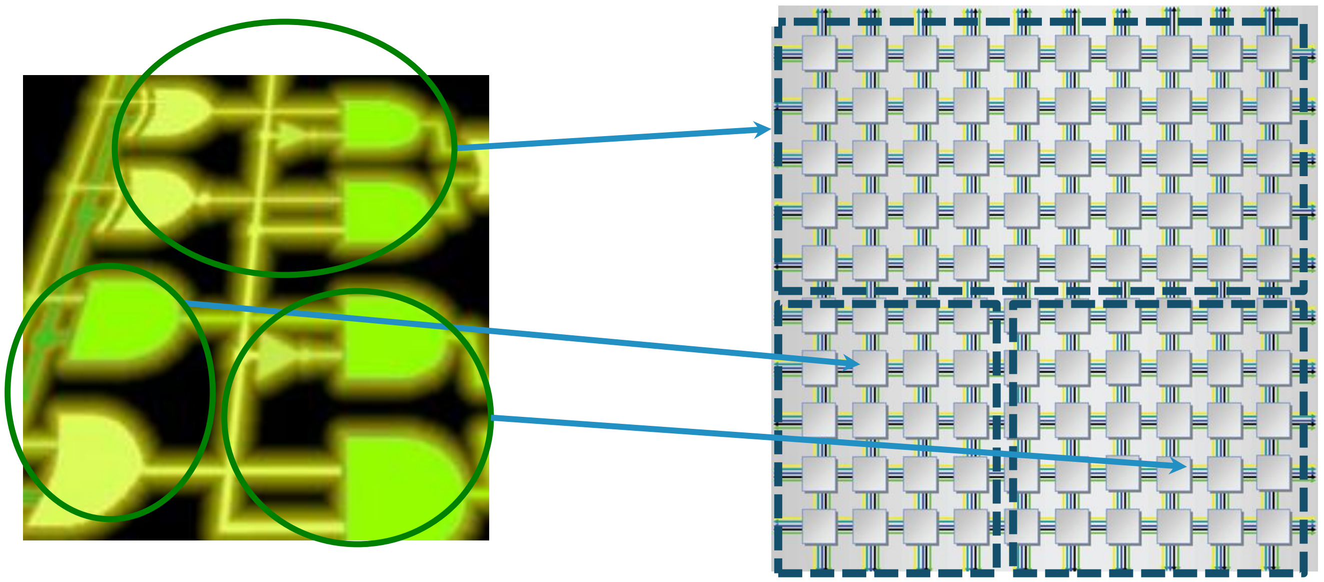
- Inter-GPU
  - Optimistic simulation
  - Less number of messages but needs roll-back



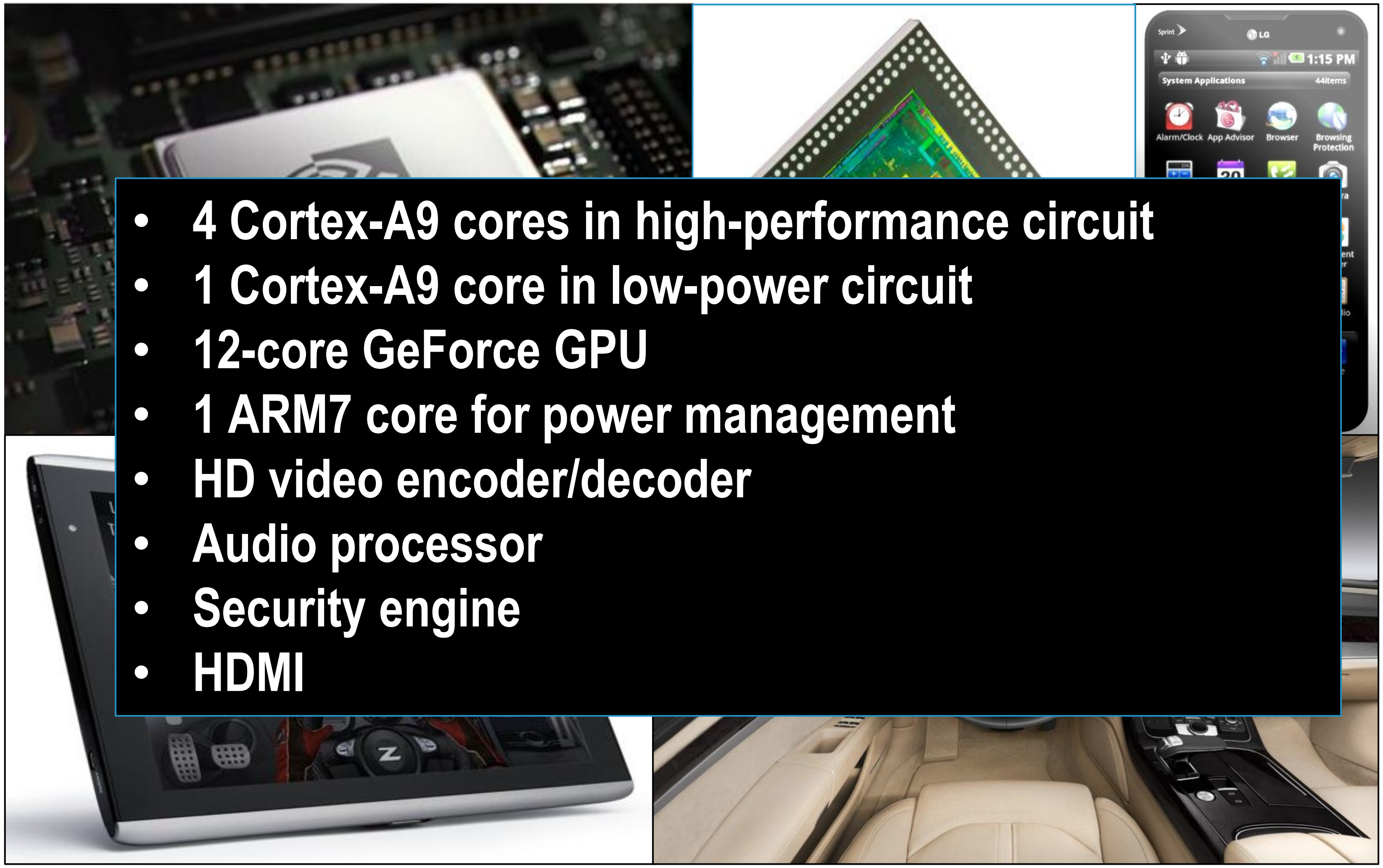


Feasibility Study of Optimistic Simulation

- Using a 64-core Tilera64 as prototyping platforms
  - Automatically translating Verilog HDL into C/C++
  - Automatic mapping “chunks of gates” to processors
  - Encouraging early results
- A hybrid simulation engine is under development

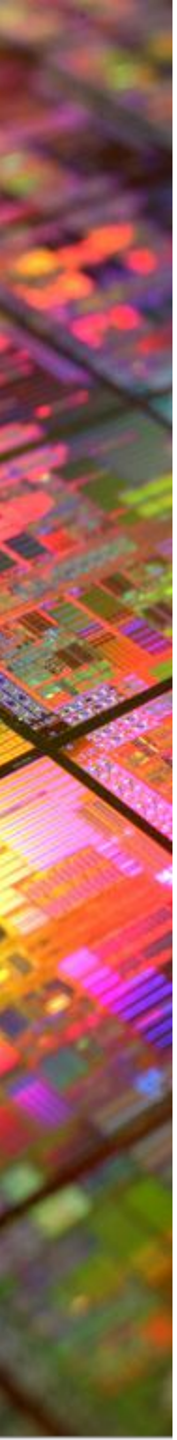


Ongoing Work 2: System Level Simulation for System-on-Chips (SoCs)



- 4 Cortex-A9 cores in high-performance circuit
- 1 Cortex-A9 core in low-power circuit
- 12-core GeForce GPU
- 1 ARM7 core for power management
- HD video encoder/decoder
- Audio processor
- Security engine
- HDMI

SoC Design Flow

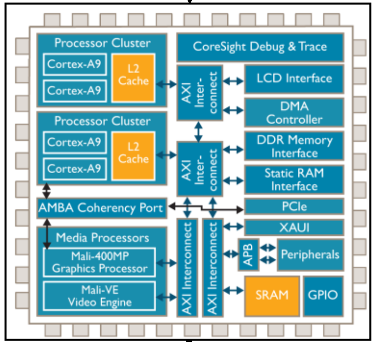


Electronic System Level Design

```
#include "shiftreg.h"

int sc_main(int argc, char* argv[]) {

   sc_signal<bool> reset, din, dout;   // Local signals
   sc_clock clk("clk",10,SC_NS);       // Create a 10ns period clock signal

   shiftreg DUT("shiftreg");           // Instantiate Device Under Test
   DUT.clk(clk);                       // Connect ports
   DUT.reset(reset);
   DUT.din(din);
   DUT.dout(dout);
   …
}
```

Input Specification

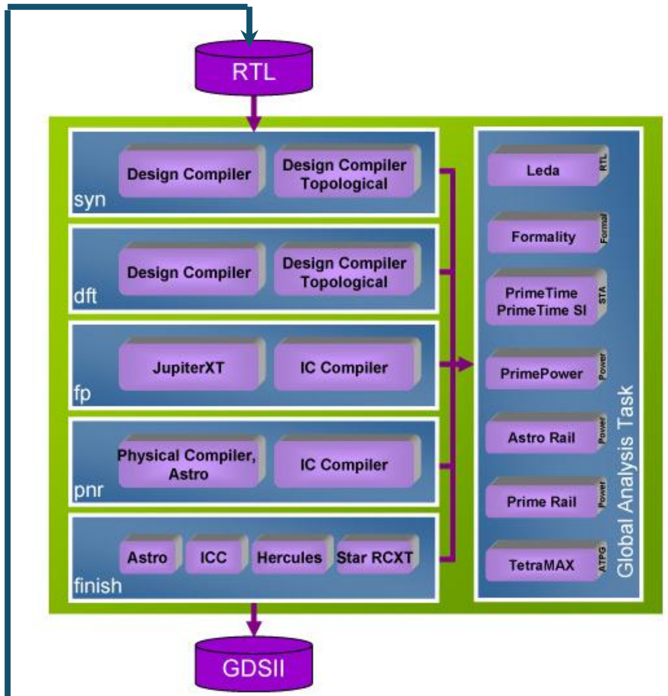
Architecture exploration

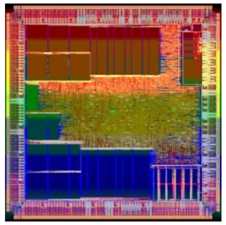


SoC Simulation Model

HW/SW Partitioning

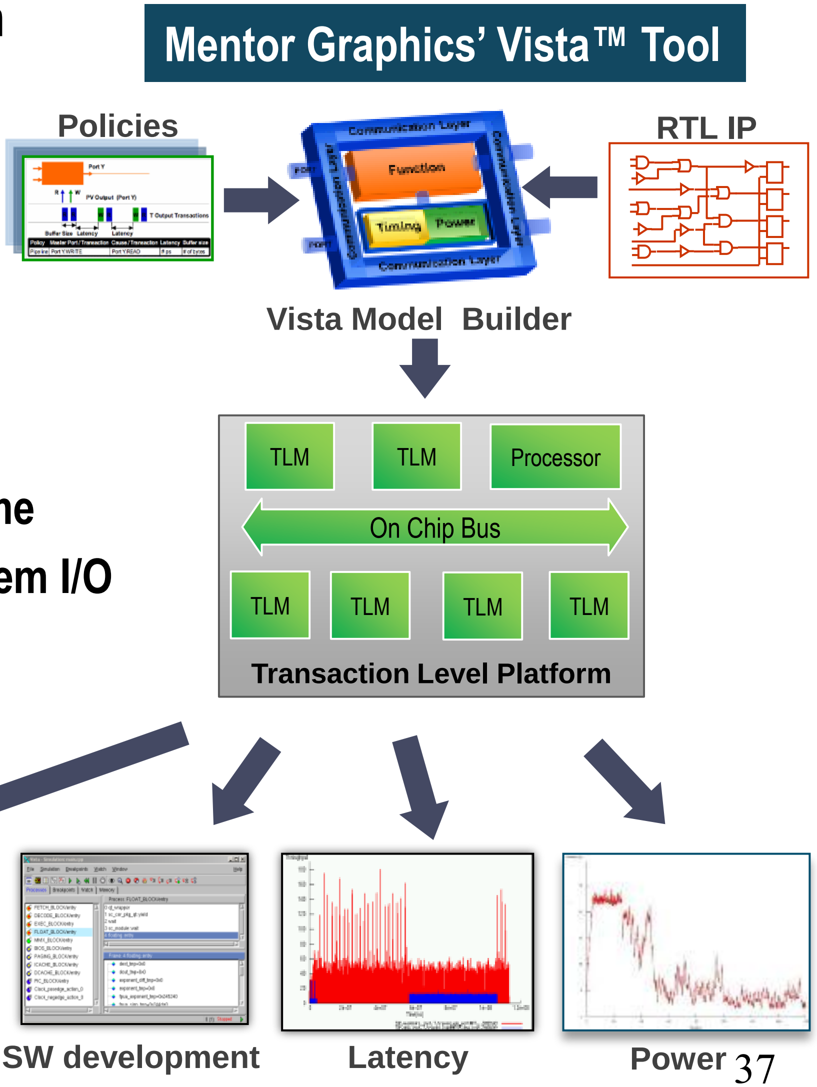
Application Mapping



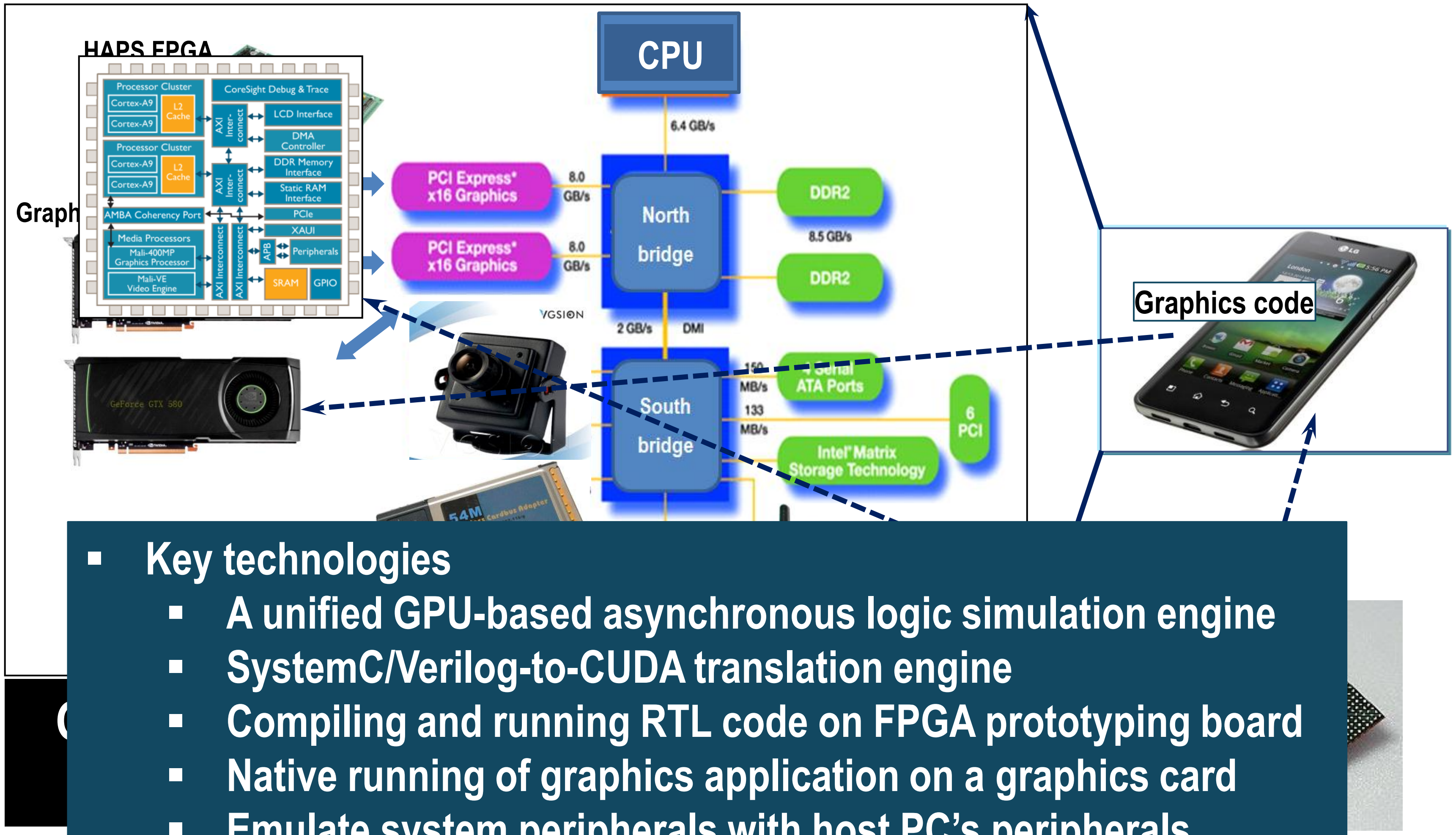


Electronic System Level (ESL) Design

- ESL design of SoC heavily depends on simulation
  - Design verification and validation
  - Architecture exploration
  - Performance analysis
  - Early software development
- Existing problems
  - System level simulation is costly in time
  - Lack of support for capturing full-system I/O behaviors
  - Need multiple abstraction levels



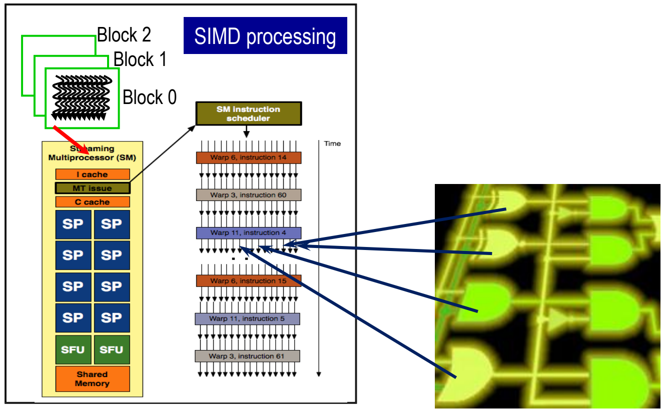
A Hardware Accelerated SoC Simulation Platform

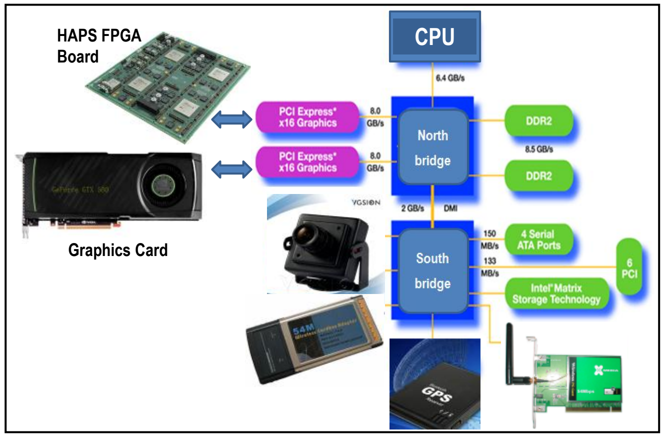


- Key technologies
  - A unified GPU-based asynchronous logic simulation engine
  - SystemC/Verilog-to-CUDA translation engine
  - Compiling and running RTL code on FPGA prototyping board
  - Native running of graphics application on a graphics card
  - Emulate system peripherals with host PC's peripherals

Conclusion

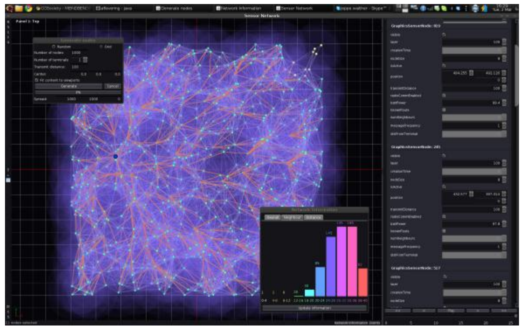
- Logic simulation is the essential means of IC verification
- World's 1st successful asynchronous logic simulation engine on manycore
  - Already done
    - Gate level simulation (30X speedup)
    - Verilog simulation (40X speedup)
  - Ongoing
    - Multi-GPU simulation
    - Full-system behavior simulation supporting mixed abstraction levels

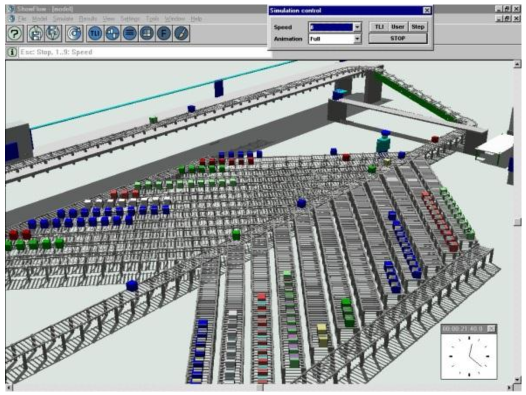


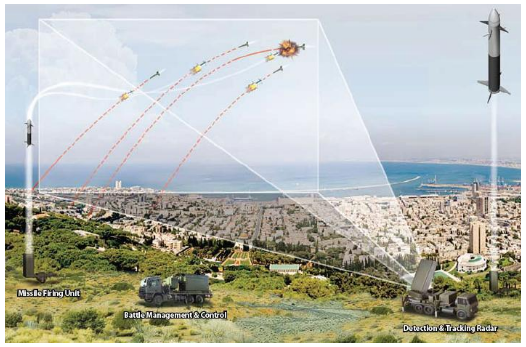


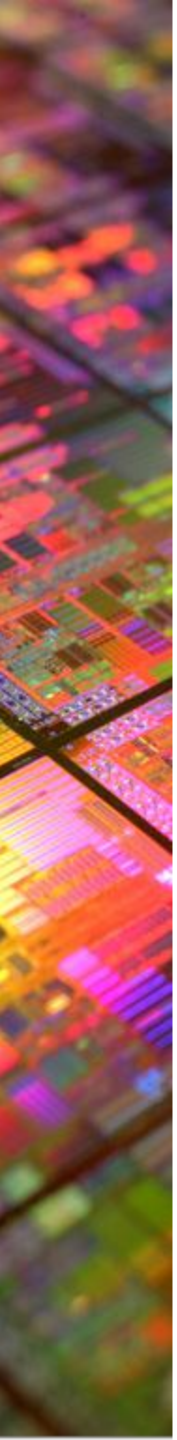
Future Work

- Potential applications
  - Network simulation
  - Factory simulation
  - Military simulation
  - Business process simulation
  - …
- Conservative and optimistic simulations are actually parallel scheduling mechanisms
  - Potentially better load balance than other approaches (e.g., work stealing)









Acknowledgement

- Contributions from David Wang (Stanford University), Yuhao Zhu (University of Texas at Austin), Hao Qian (AMD), and Lingfeng Wang (Tsinghua University)
- Insightful discussions with Dr. Gilbert Chen (Sandvine), Prof. Radu Marculescu (Carnegie Mellon University), Dr. Li Shang (Intel), Xin Zhou (Intel), and Jen-Hsun Huang (NVIDIA)