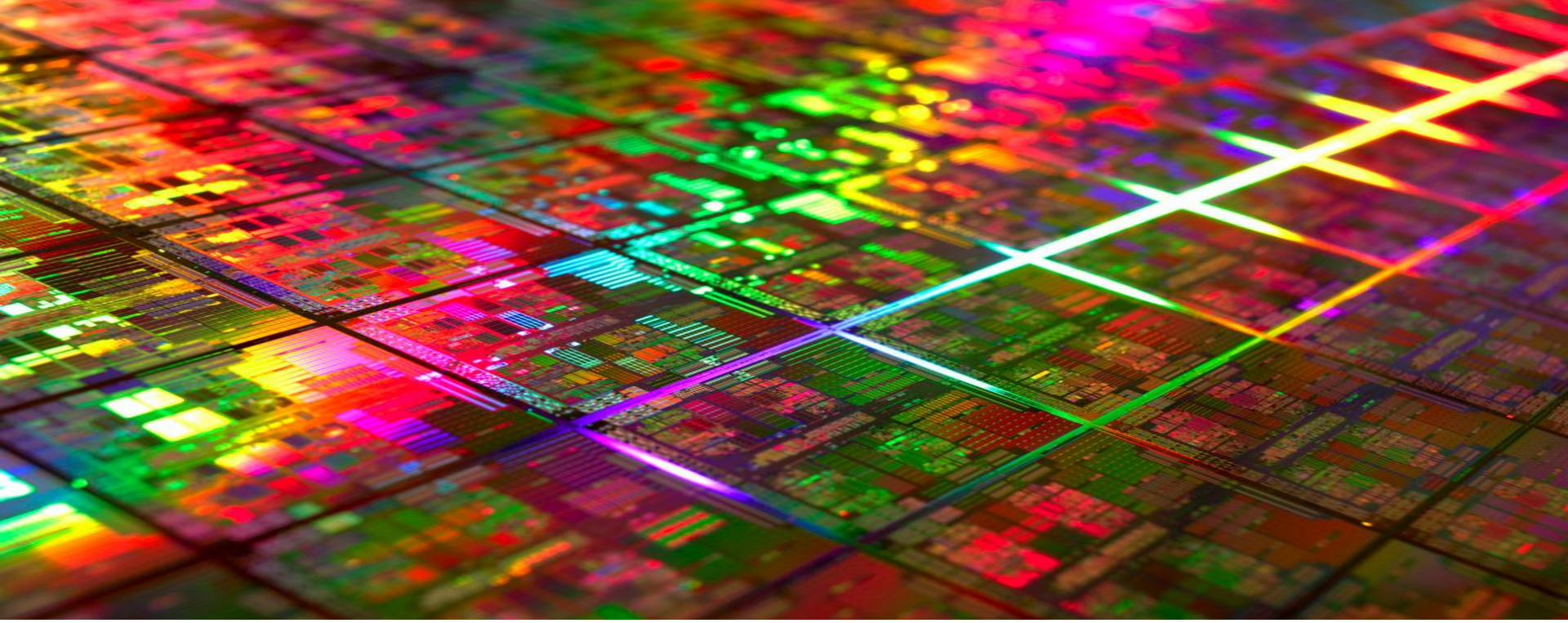
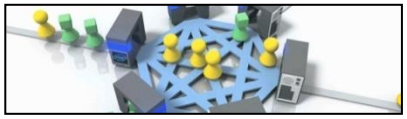


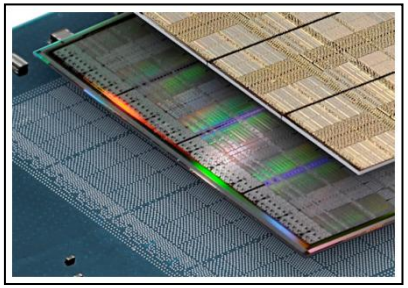
Massively Parallel Logic Simulation

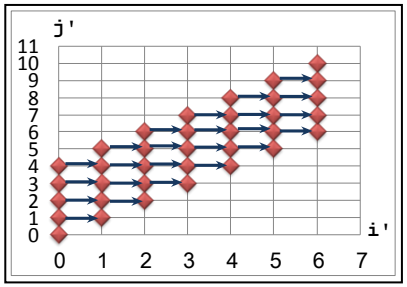
Yangdong Steve Deng 邓仰东
dengyd@tsinghua.edu.cn
Department of Micro-/Nano-electronics
Tsinghua University

Research Overview

- Ongoing work
  - Parallel algorithms/applications
    - Software routers
    - Logic simulation for VLSI design
    - Radar signal processing
  - GPU architecture
    - Heterogeneous integration
    - Supporting task-pipelined parallelism on GPUs
  - Automatic parallelization
    - Polyhedral compilation for GPUs
  - Sponsored by National Key Projects, CUDA Center of Excellence, Tsinghua-Intel Center of Advanced Mobile Computing Technology, Intel University Program, NVidia Professor Partnership Award, NSF China

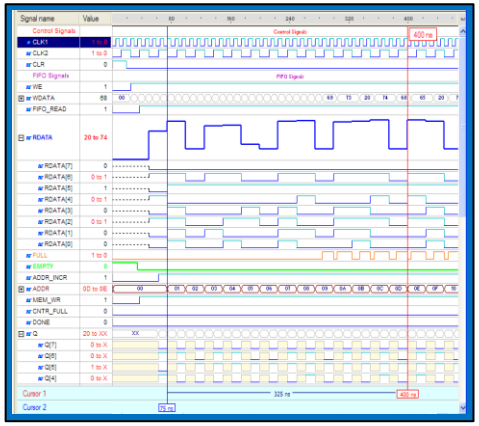






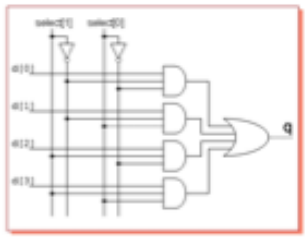
Outline

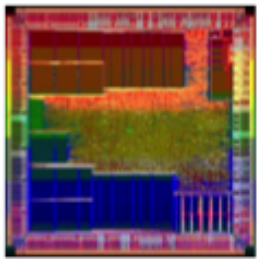
- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled RTL simulation
- Conclusion and future work

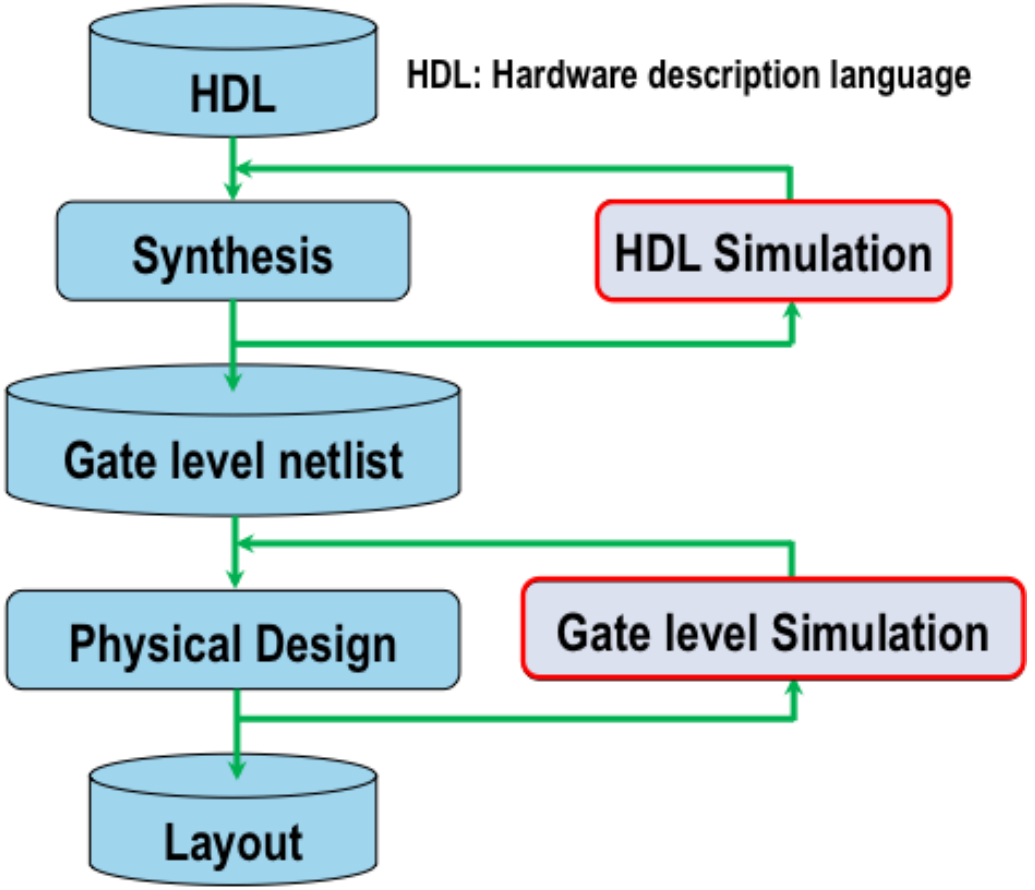


Simplified IC Design Flow

```
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
```

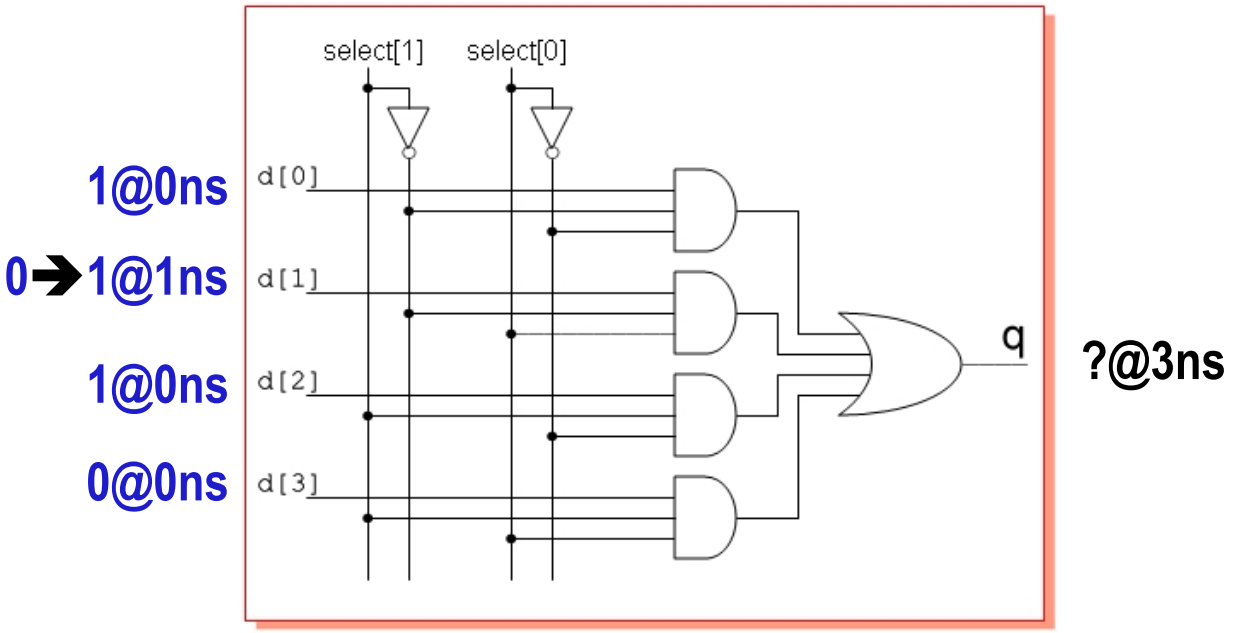






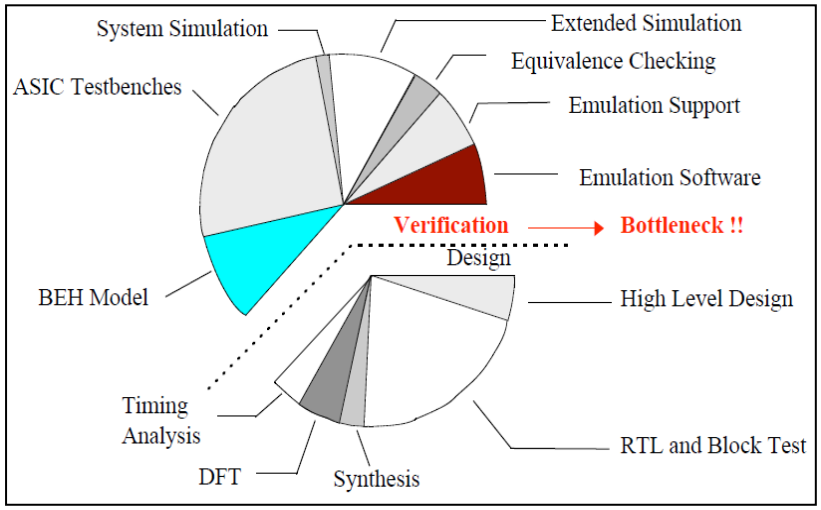
Logic Simulation

- Major method for IC design verification
  - Performed on a proper model of the design
  - Apply input stimuli
  - Observe output signals

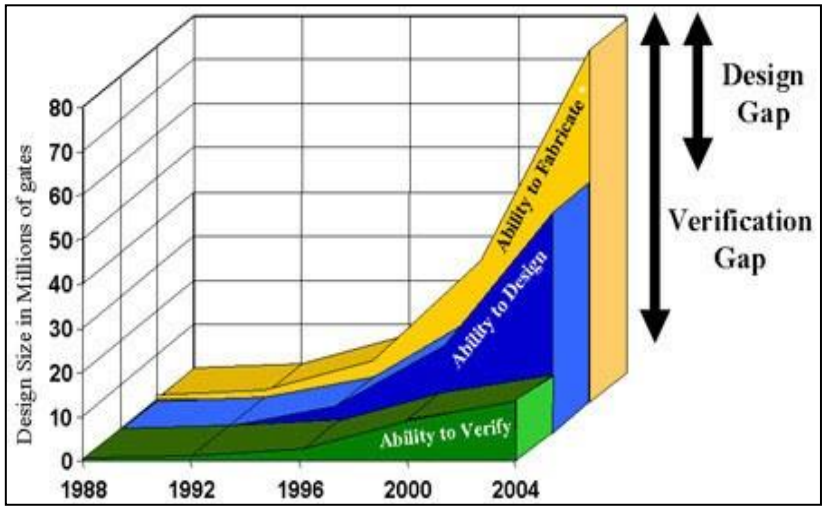


Verification Gap

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



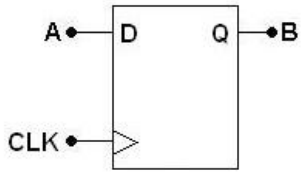
Distribution of IC design effort



Widening design & verification gap

Simulation Complexity

- For a 1-bit register and 1 data input
  - 2 possible input values for each register state
  - Number of test patterns required = $2^{1+1} = 4$
- 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 GPU chip with 1 billion gates
  - ????? years?



Q	A	CLK	B
0	0	⌐	0
0	1	⌐	1
1	0	⌐	0
1	1	⌐	1

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

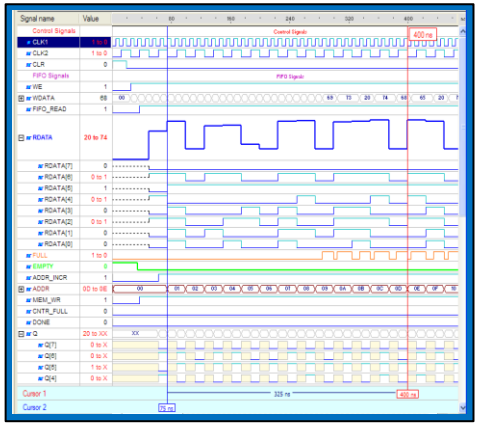
Problem

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

?

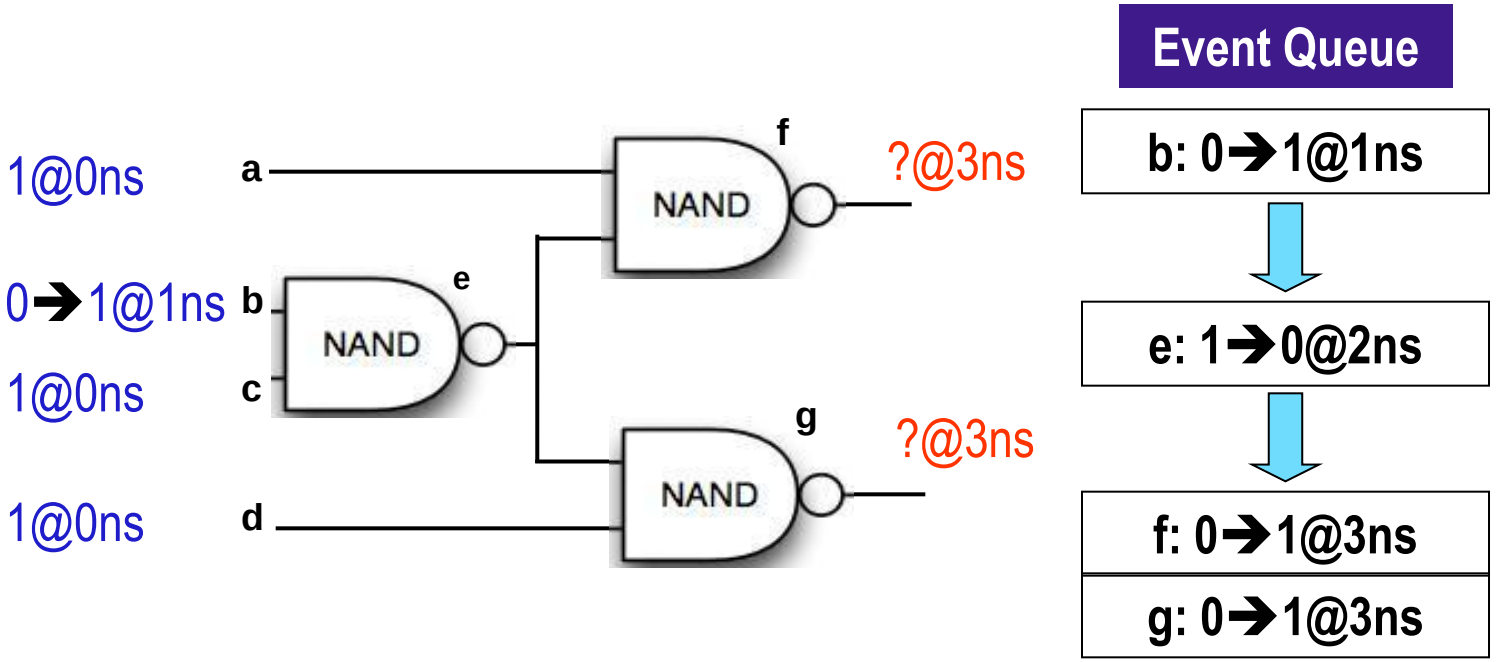
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled RTL simulation
- Conclusion and future work



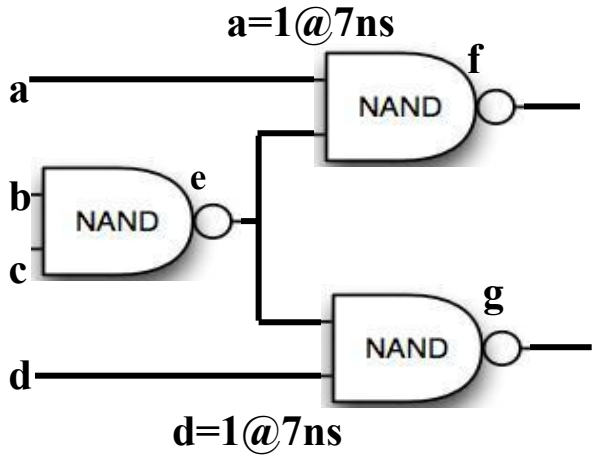
Event Driven Simulation

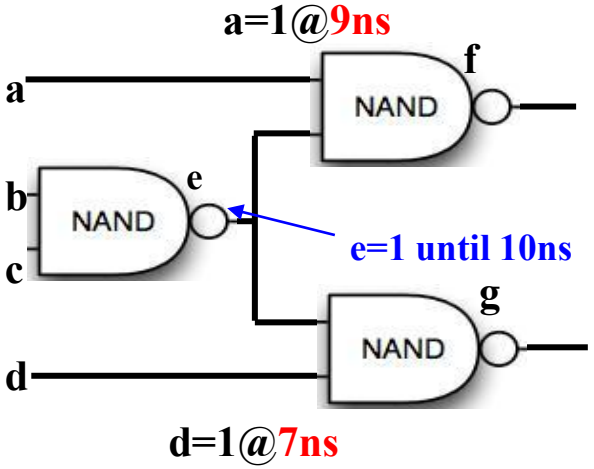
- Most used algorithm for discrete event systems
  - Event: logic transition + time stamp
  - Always evaluate an event with the earliest stamp



Parallel Logic Simulation

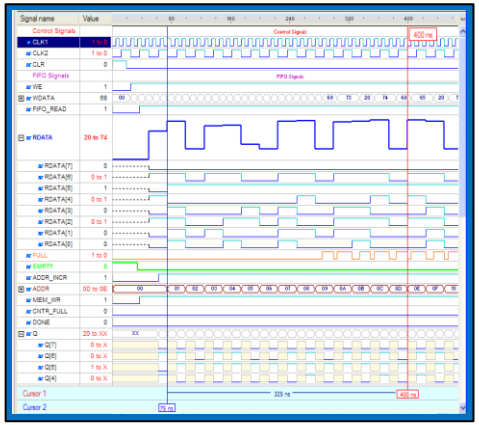
- Synchronous approaches
  - Simultaneously simulate events with the same time-stamp
    - Insufficient parallelism
- Asynchronous approaches
  - Conservative simulation
    - Chandy-Misra-Bryant (CMB) Algorithm
  - Optimistic simulation





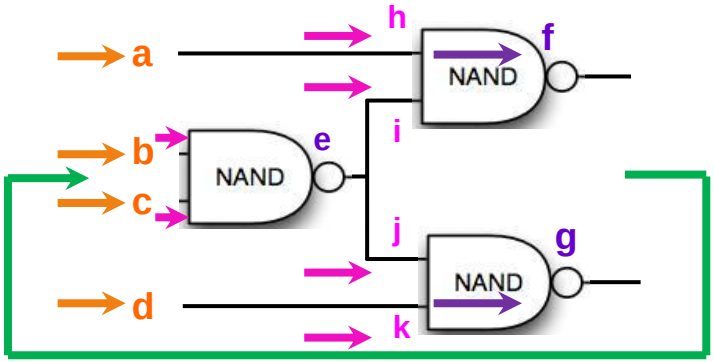
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled RTL simulation
- Conclusion and future work



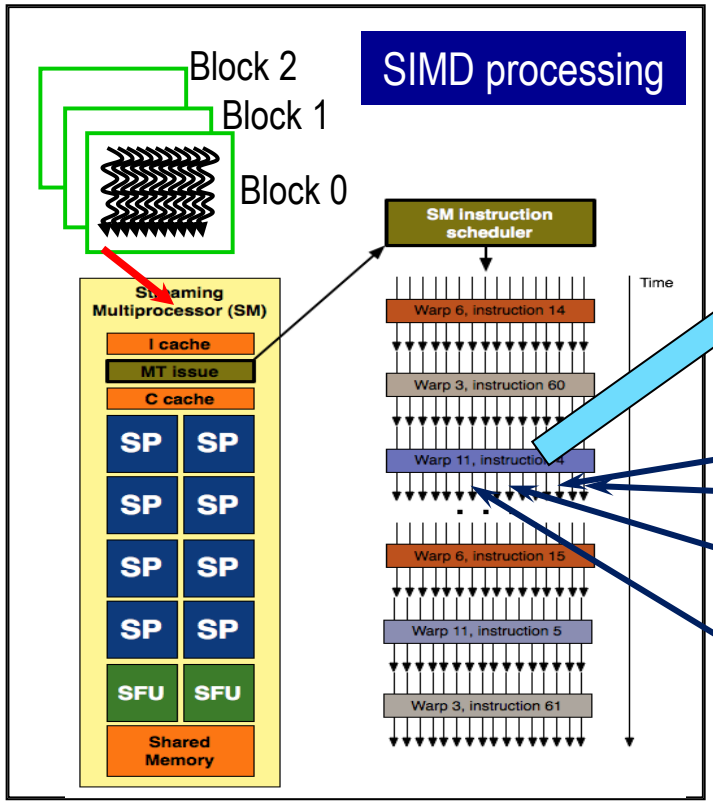
Basic Simulation Flow

```
while not finish
  // 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
```

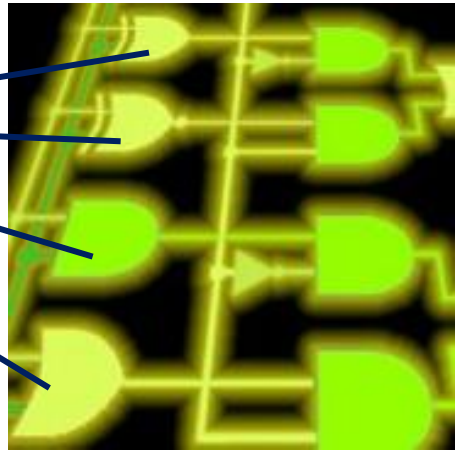


SIMD Evaluation

- Gate evaluation through truth-table lookup
- Try assigning gates of the same type into a single warp



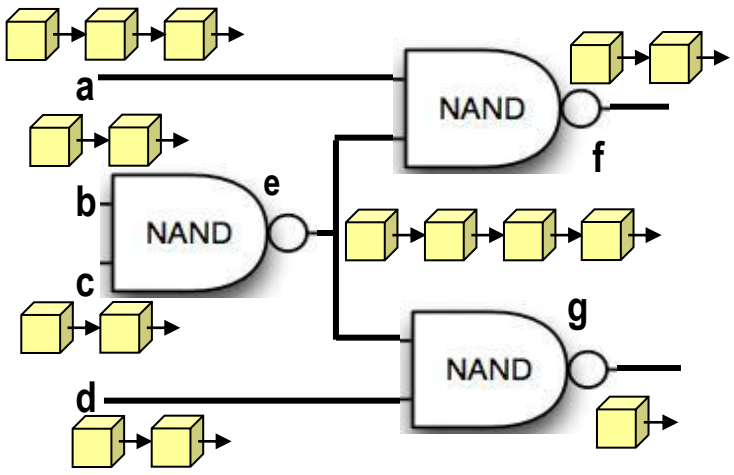
	a	b	y
NAND	0	*	1
	…	…	…
NOR	1	*	0
..	…	…	…



Distributed Event Queues

- Removing the global event queue
- Each gate input has its own events queues
  - Irregular distribution of required event queue sizes

Local Event Queues



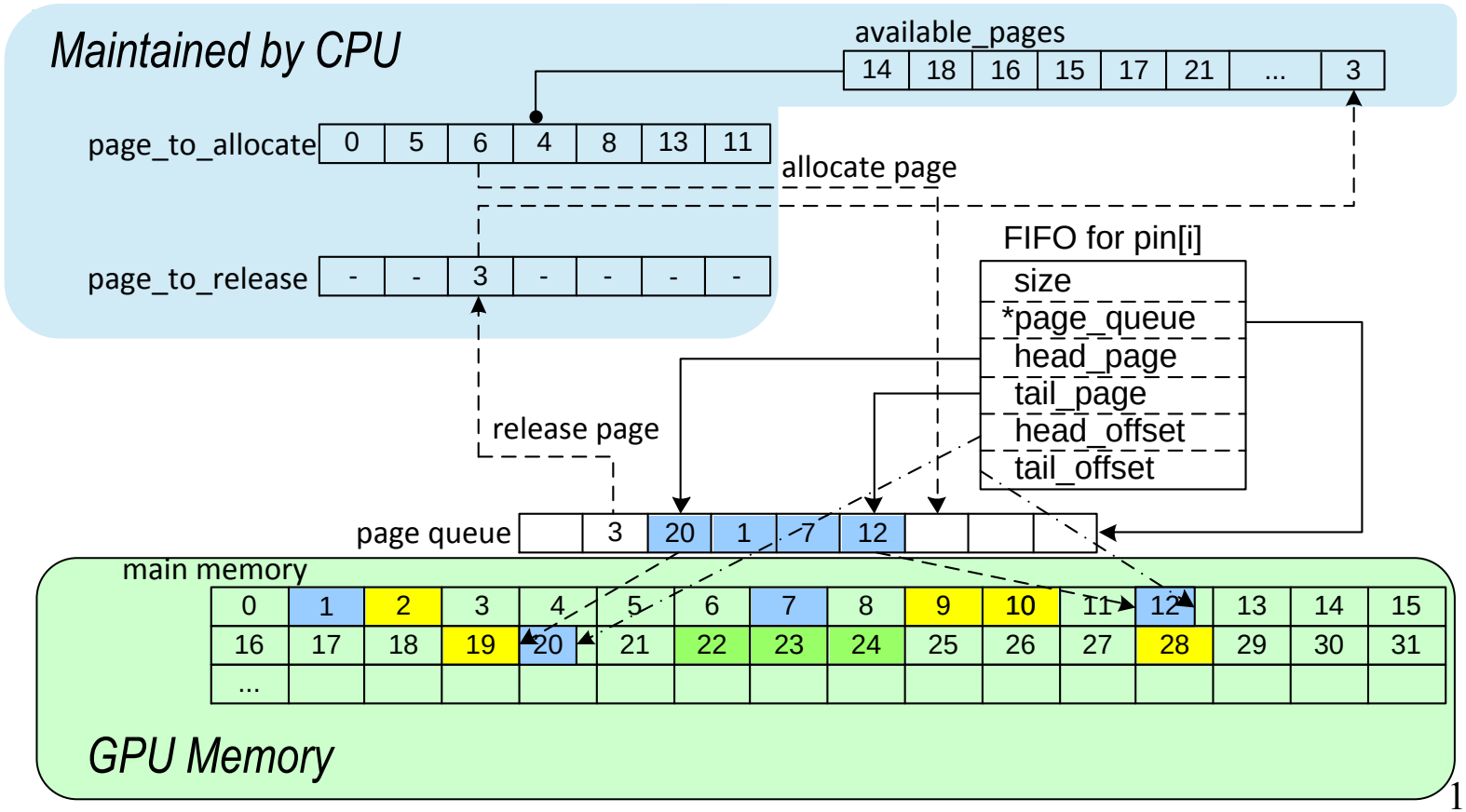
Global Event Queue

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

g: 0➔1@3ns

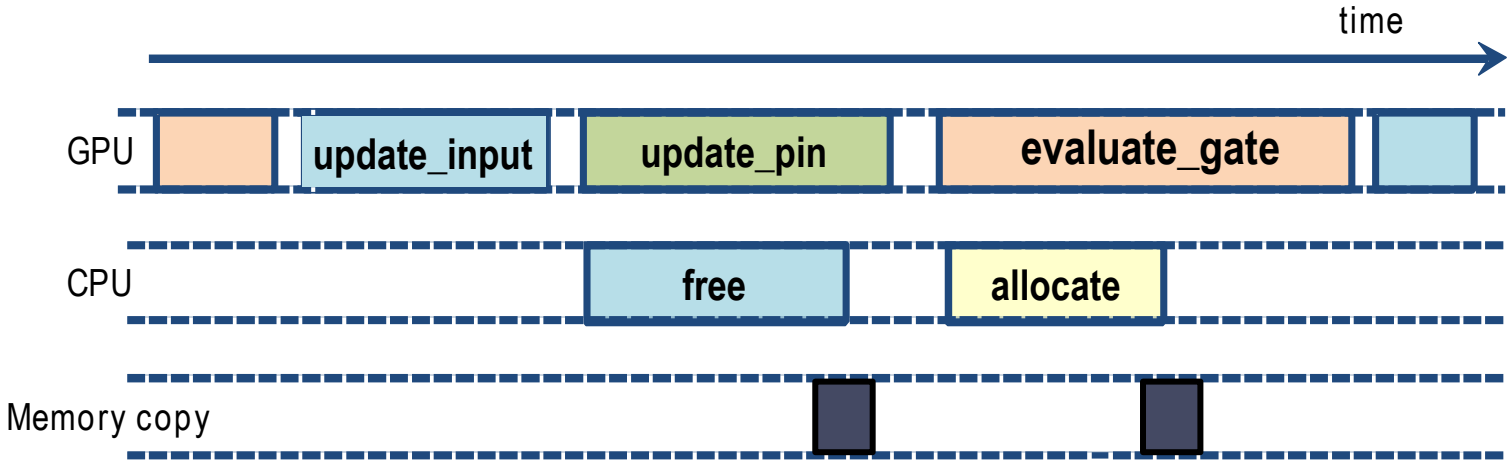
Dynamic GPU Memory Management

- 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



Gate Level Simulation Results

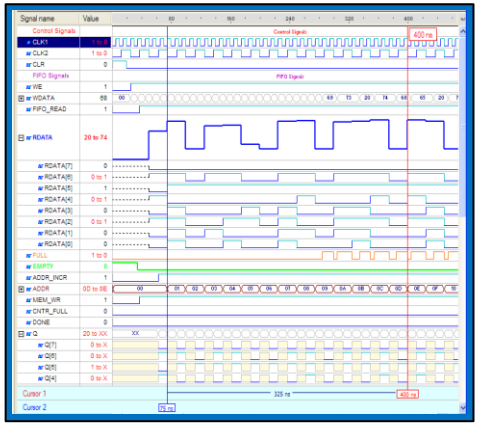
- World's fastest logic simulator on general purpose hardware[1]
- 30X speed-up on average[2] (100X for random patterns)
  - 1 month on CPU vs. 1 day on GPU

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 RTL simulation
- Conclusion and future work



HDL Based RTL Simulation

- RTL (Register transfer level) to GSDII flow dominates
  - Verilog hardware description language (HDL) as input
  - Similar to C but has a hardware context

Description example of multiplexer (Verilog_HDL):

```
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
```

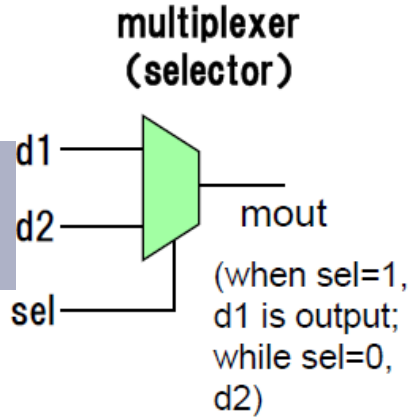
module definition

terminals

Process: the unit of concurrency

functions

end of module definition



Challenge 1: Irregular Behavior

- Verilog HDL as input
- Arbitrarily complex behavior in a Verilog process
  - Cannot be captured by truth tables
- Solution
  - Translating Verilog into CUDA code
  - Running the resultant CUDA code on GPU to evaluate the behaviors

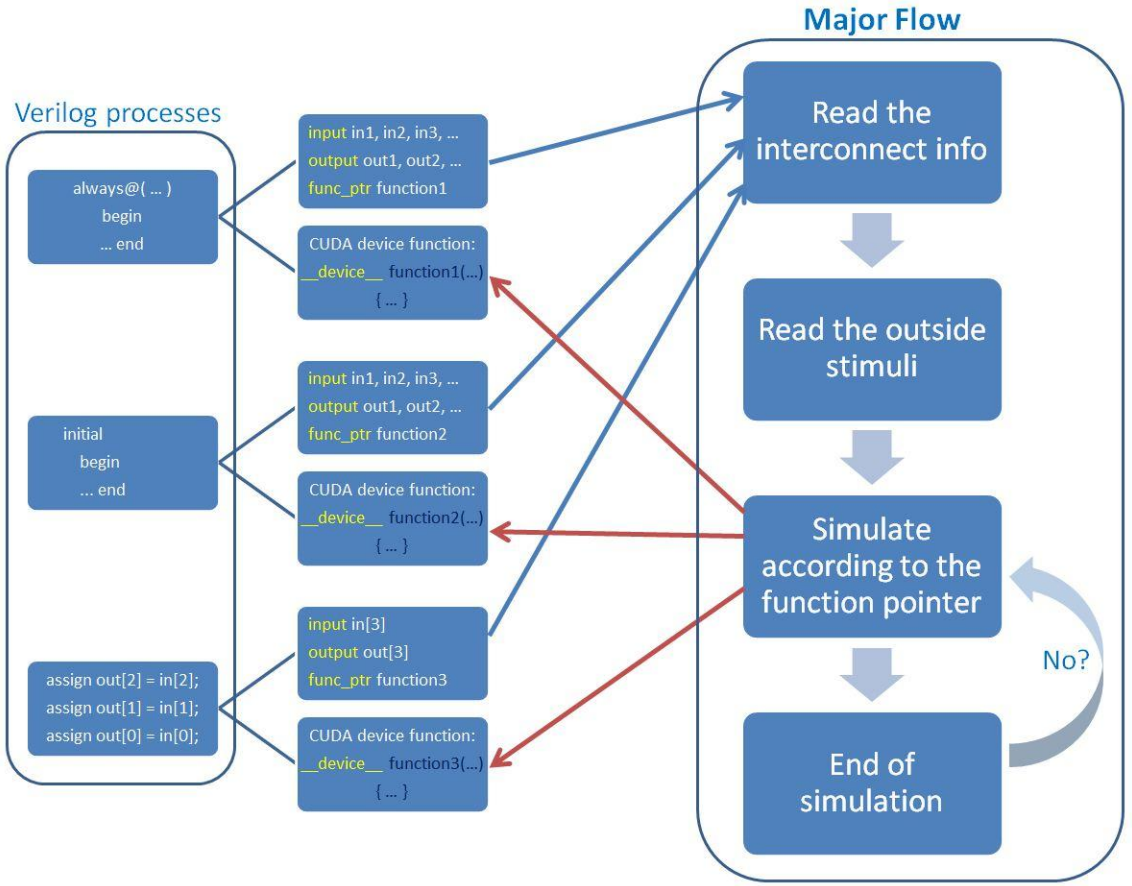
Verilog HDL

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

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

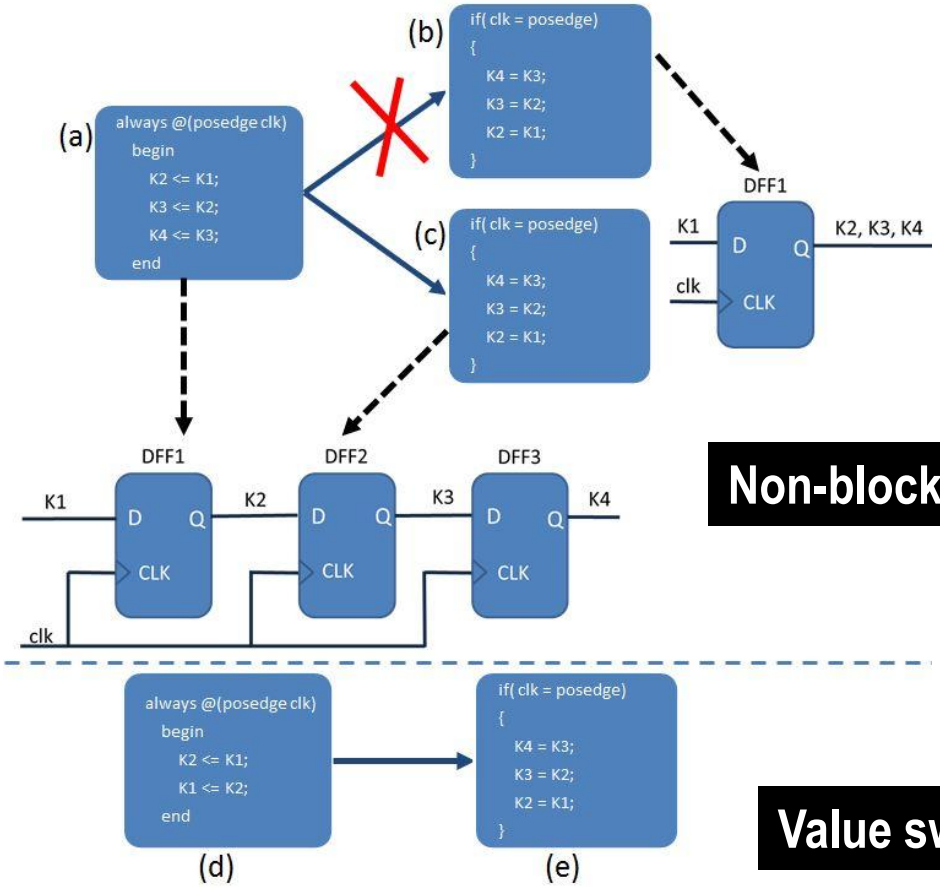
Compiled Simulation

- Translating Verilog into equivalent CUDA code



Handling the Semantic Differences

- Has to take care the semantic difference between HDL and CUDA!

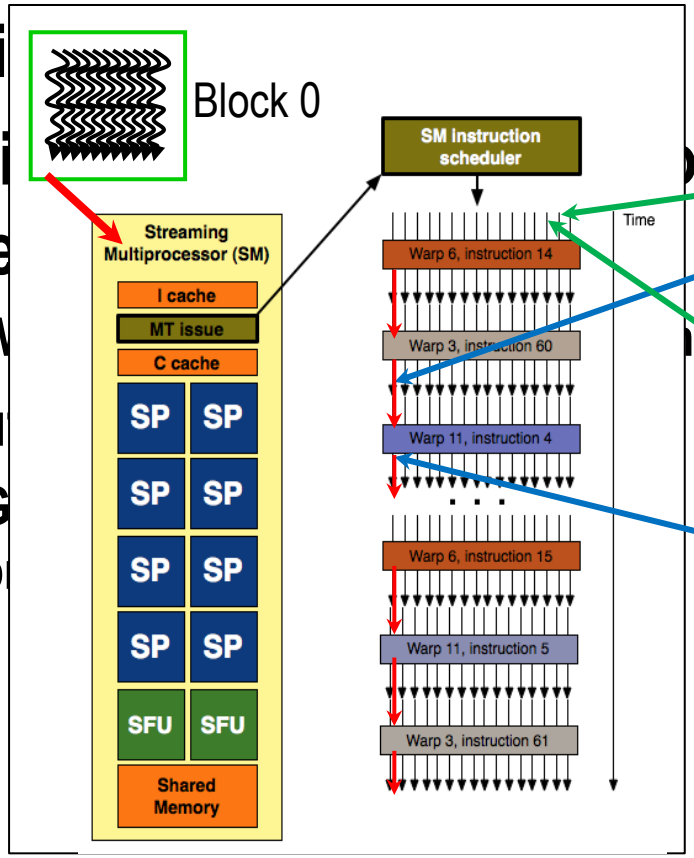


Non-blocking to blocking statements

Value swapping

Challenge 2: Branch Divergence

- Veri
- Arbi … or in a Ve …
  - W … ches
- Solu …
  - G … p…

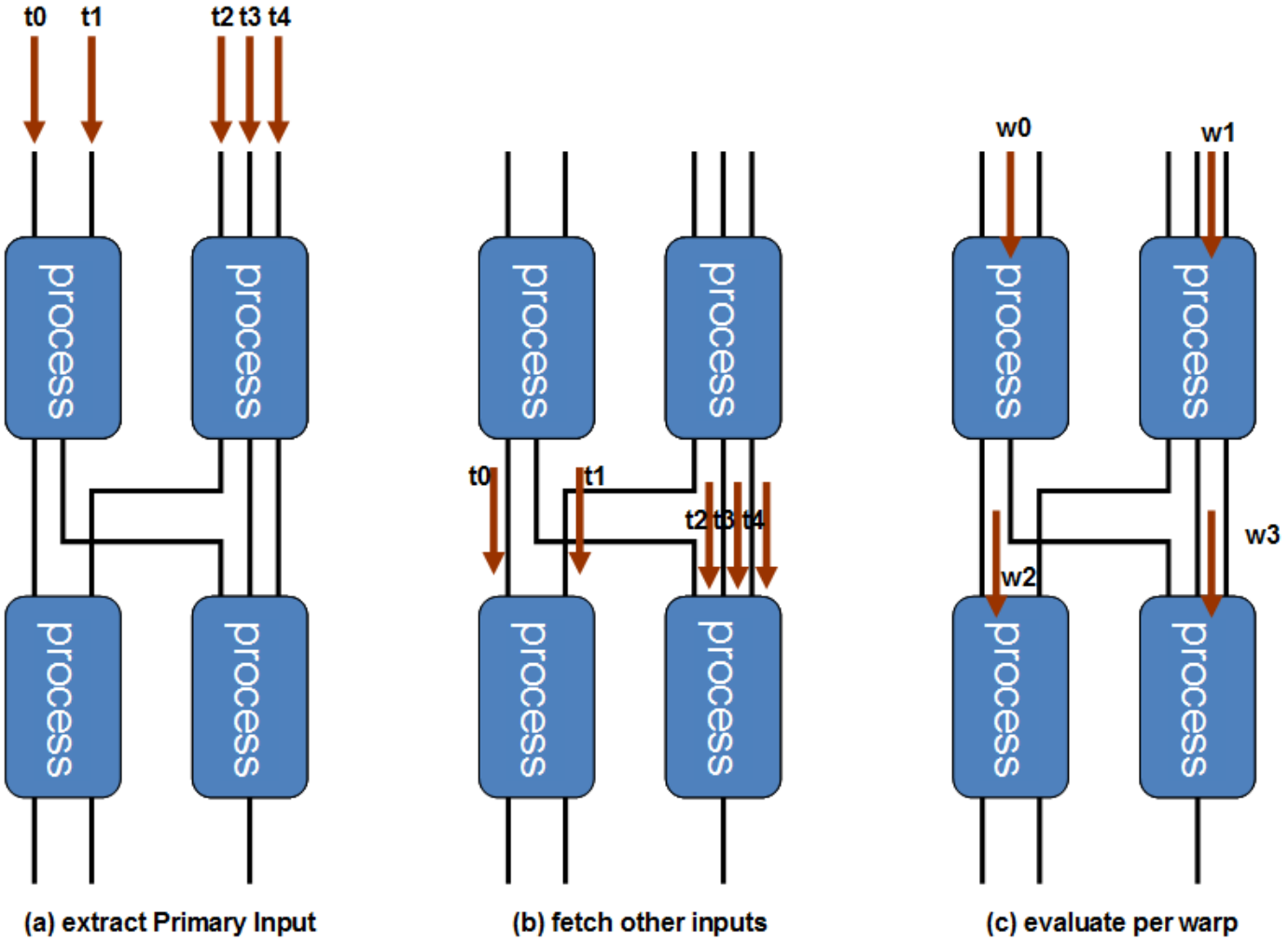


Verilog HDL

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

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

Parallel Simulation



HDL Simulation

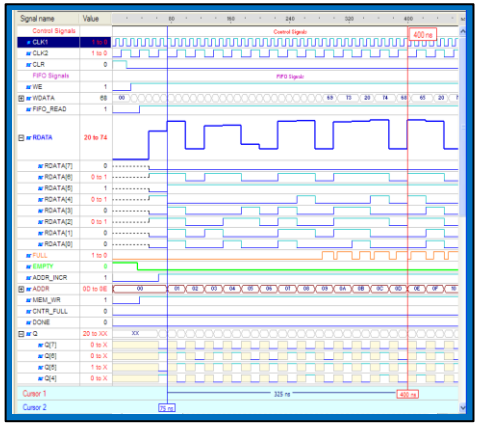
- **20-50X speed-up depending on circuit structure**

Design	Mentor Graphics ModelSim (s)	GPU-Based Simulator (s)	Speedup
Mux	83.32	4.019	**20.73X**
Adder	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

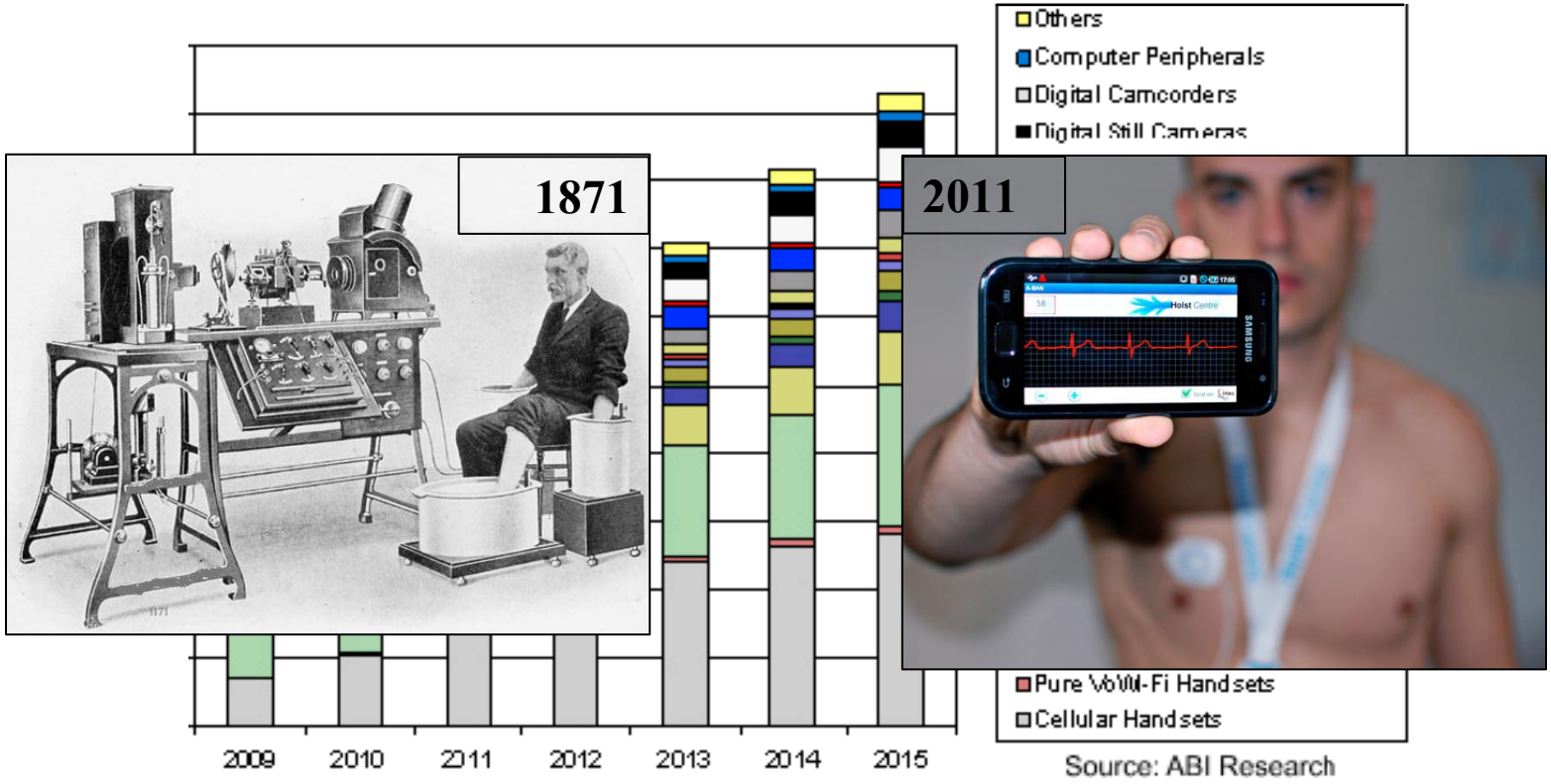
Outline

- Motivation
- Simulation algorithms
- Massively parallel logic simulation
  - Gate Level simulation
  - Compiled RTL simulation
- Conclusion and future work

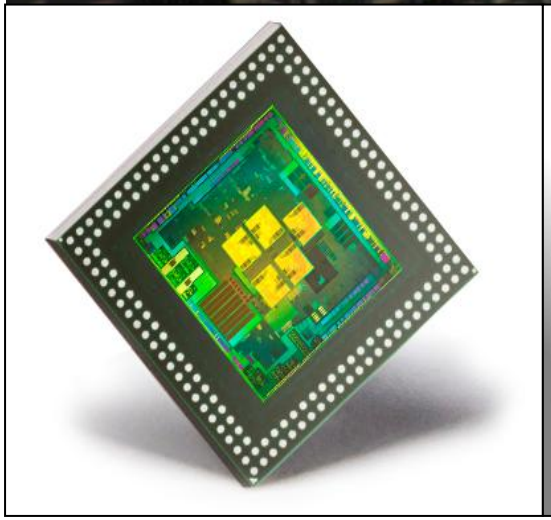


Future Work

- Embedded systems will be a big deal!
  - By 2015, cell phone market = $341B, but PC = ~$200B



SoC - Driver of Embedded Devices



SoC = System-on-Chip (片上系统或系统芯片)

Future Work - SoC Design Flow

Electronic System Level Design

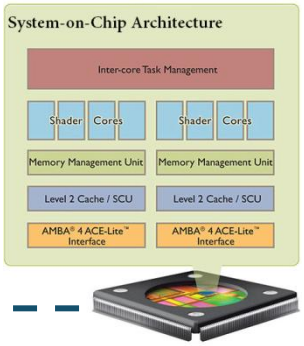
```
#include "shiftreg.h"

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);
   …
}
```

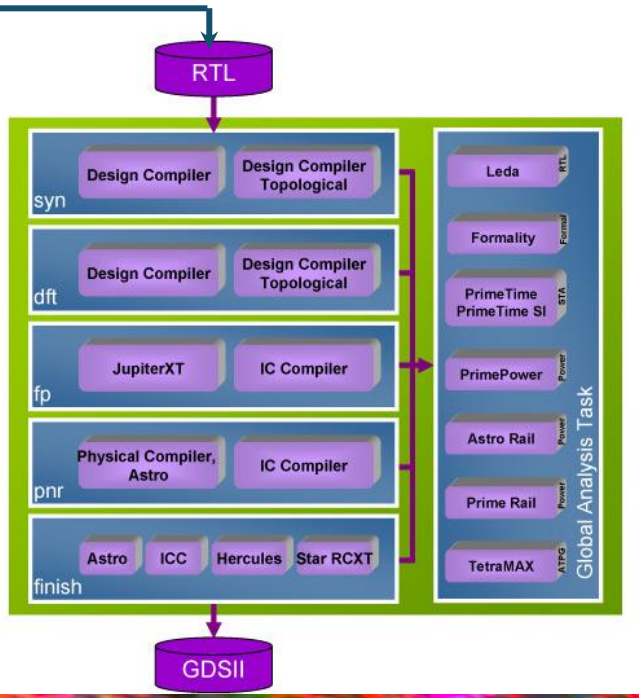
Input Specification

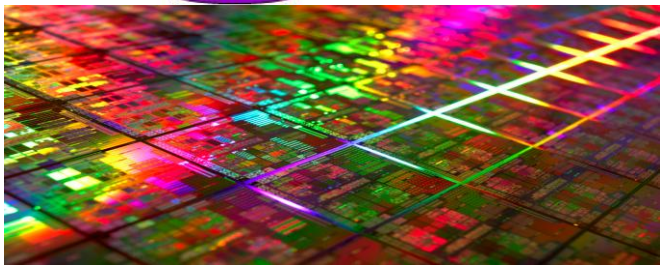
HW/SW Partitioning

Application Mapping

Analysis

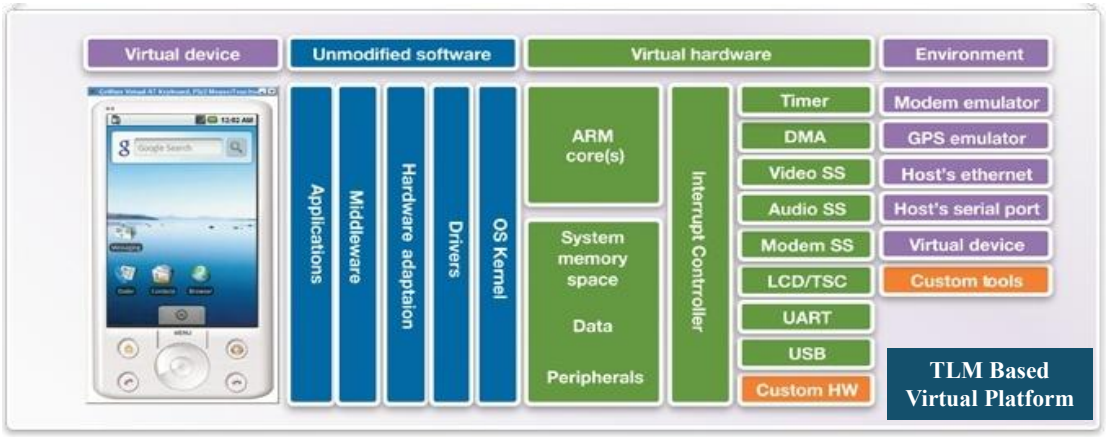


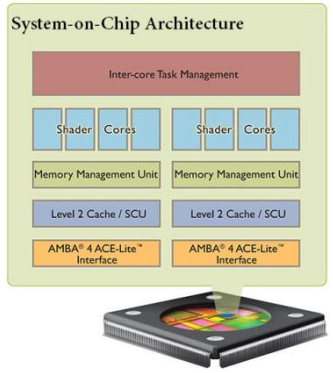




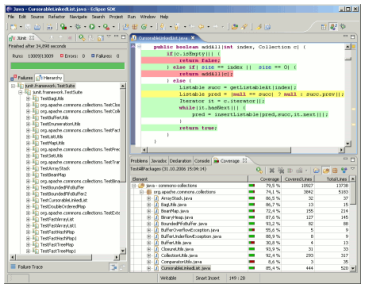
Future Work

- Virtual Platform based SoC design
  - Bottleneck is still simulation speed
  - Hard to capture system I/O behaviors
  - Need to handle HW/SW models at multiple abstraction levels

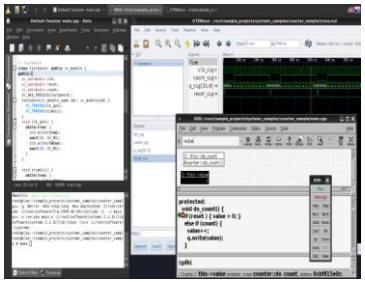




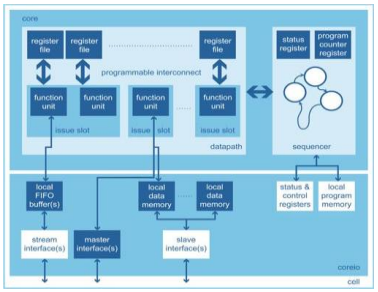
Software Development



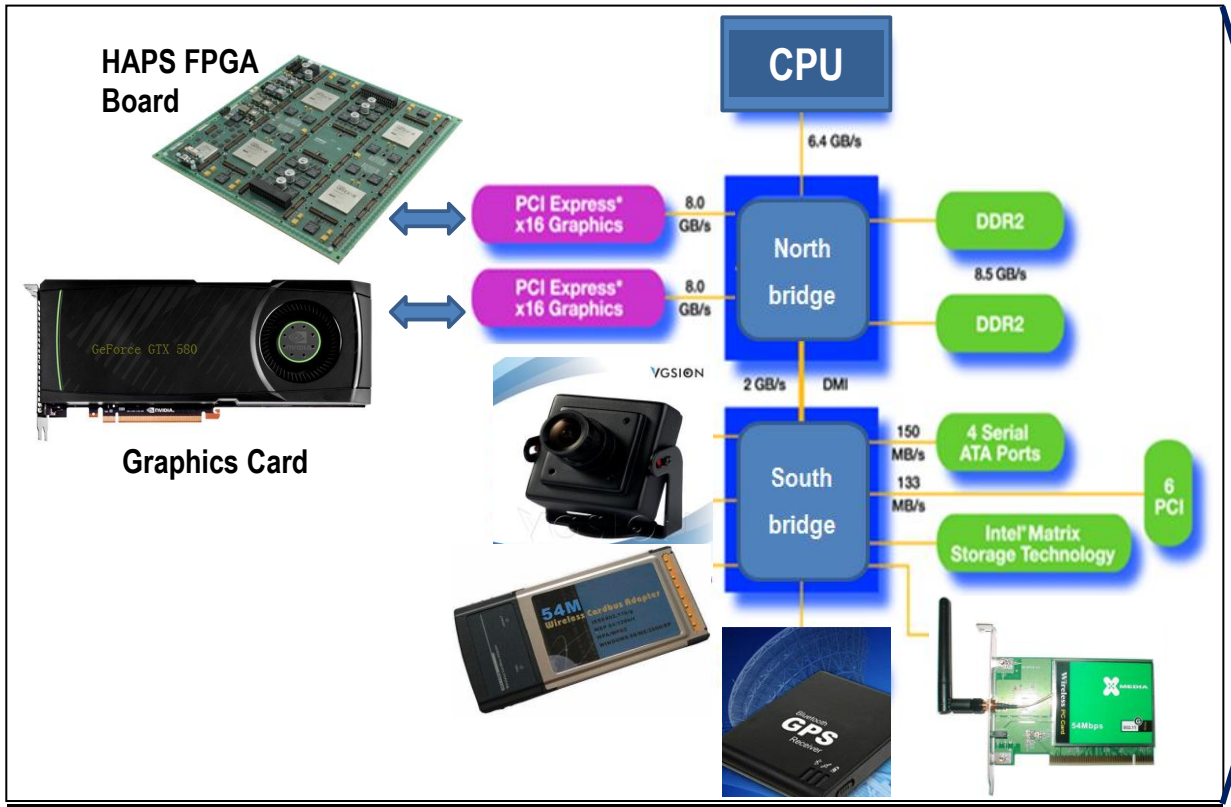
Validation

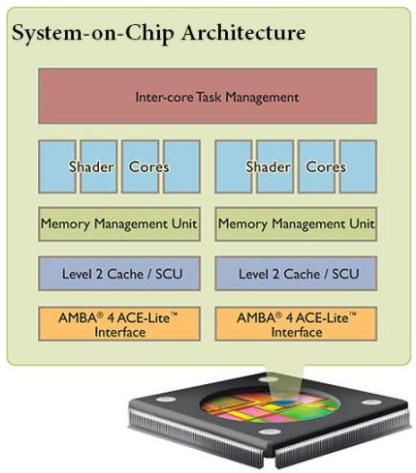


Architecture Exploration



A GPU Based SoC Simulation Framework





GPU-accelerated full-system simulation for System-on-Chips

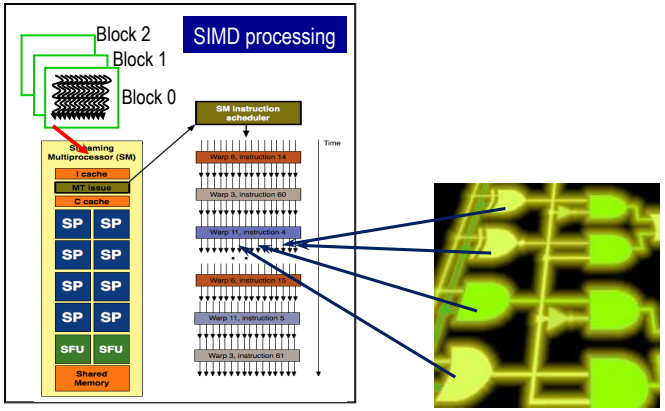
Key Technologies

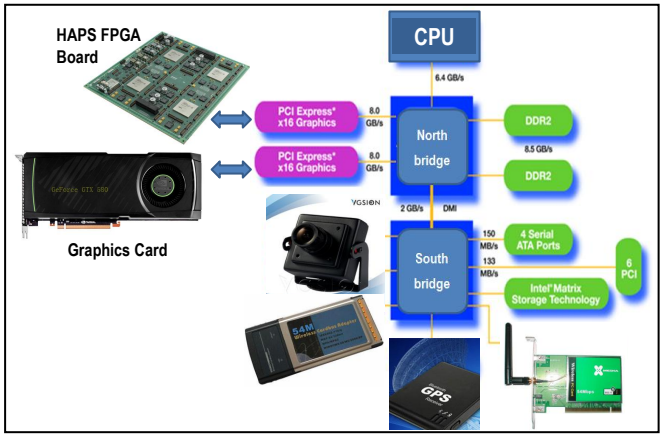
- SystemC/Verilog-to-CUDA translation engine
- A unified GPU-accelerated asynchronous logic simulation engine
- Native running of graphics application on a graphics card
- Simulate system peripherals with host PC's peripherals

SoC Simulation Framework

Conclusion

- Logic simulation is the essential means of IC verification
- We propose a GPU accelerate simulation framework
  - Already done
    - Gate level simulation (30X speedup)
    - Verilog simulation (40X speedup)
  - Ongoing
    - Full-system behavior simulation supporting mixed abstraction levels





Conclusion

- Potential applications
  - Network simulation
  - Factory simulation
  - Multi-agent simulation
  - Business process simulation
  - …
- How about a GPU cloud offering simulation services?