# Distributed Time, Conservative Parallel Logic Simulation on GPUs

Bo Wang[1], Yuhao Zhu[2], Yangdong Deng[1]
[1] Tsinghua University, [2] Beihang University

# Outline

- Motivation
- Background
- Parallel Logic Simulator
- Experiments
- Conclusion

# Motivation

- **Simulation has become a bottleneck for circuit design**
  - **60~80 % of design effort is now dedicated to verification[1]**
  - **Example: the logic simulation of a billion-transistor design could take over one month to finish[2]**

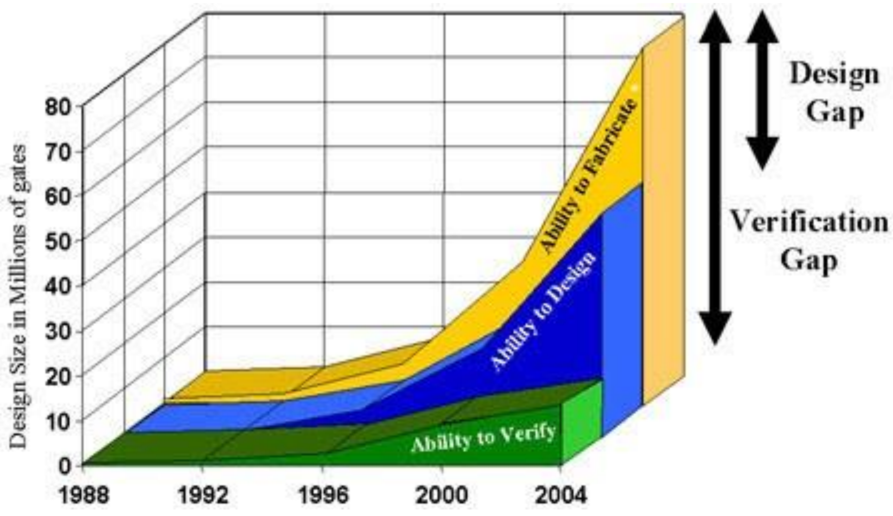



**Figure 1. Verification Gap [3]**

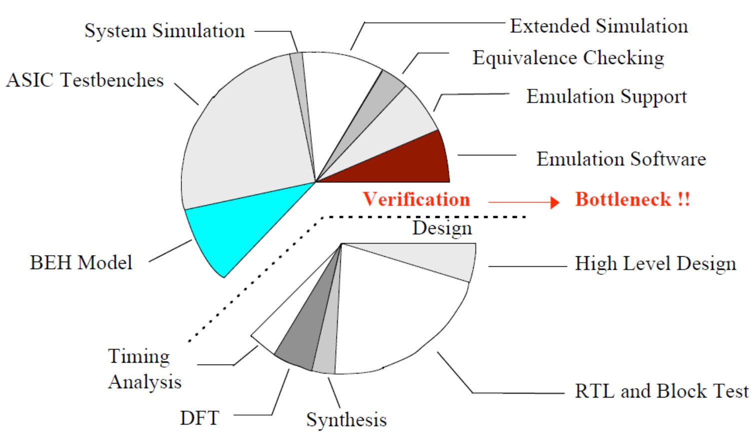



**Figure 2. Breakdown of Effort [4]**


[1] Chien-Nan Liu, SoC Verification Methodology, Oct 2003

[2] ESNUG Industry Discussion, Dec 2003, http://www.deepchip.com/items/0421-01.html

[3] Brian Bailey, A new vision of 'scalable' verification, http://www.eetimes.com/news/design/features/showArticle.jhtml?articleID=18400907

[4] "Functional Verification on Large ASICs" by Adrian Evans, etc., 35th DAC, June 1998.

# Types of simulations

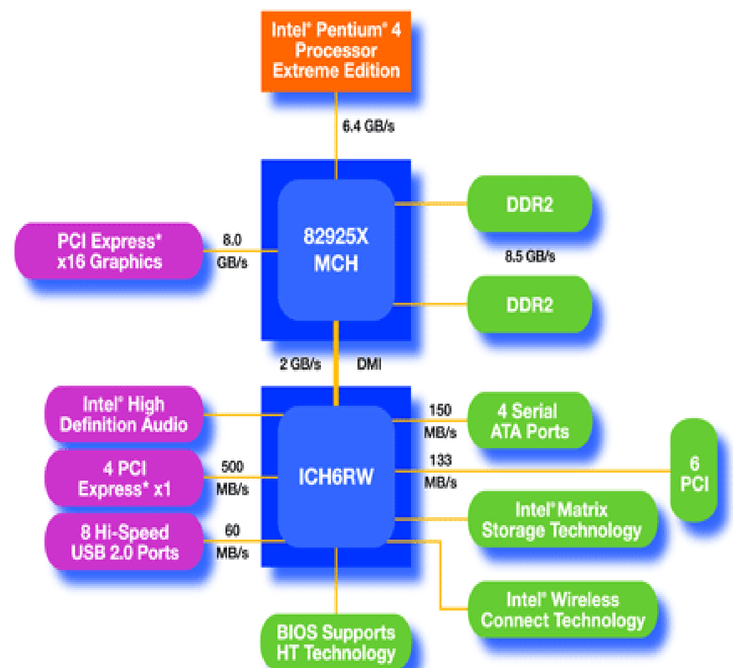
- Behavioral level simulation
- Register transfer level simulation
- Gate level simulation
- Transistor level simulation
- …

# Outline

- Motivation
- Background
- Parallel Logic Simulator
- Experiments
- Conclusion

# New platform

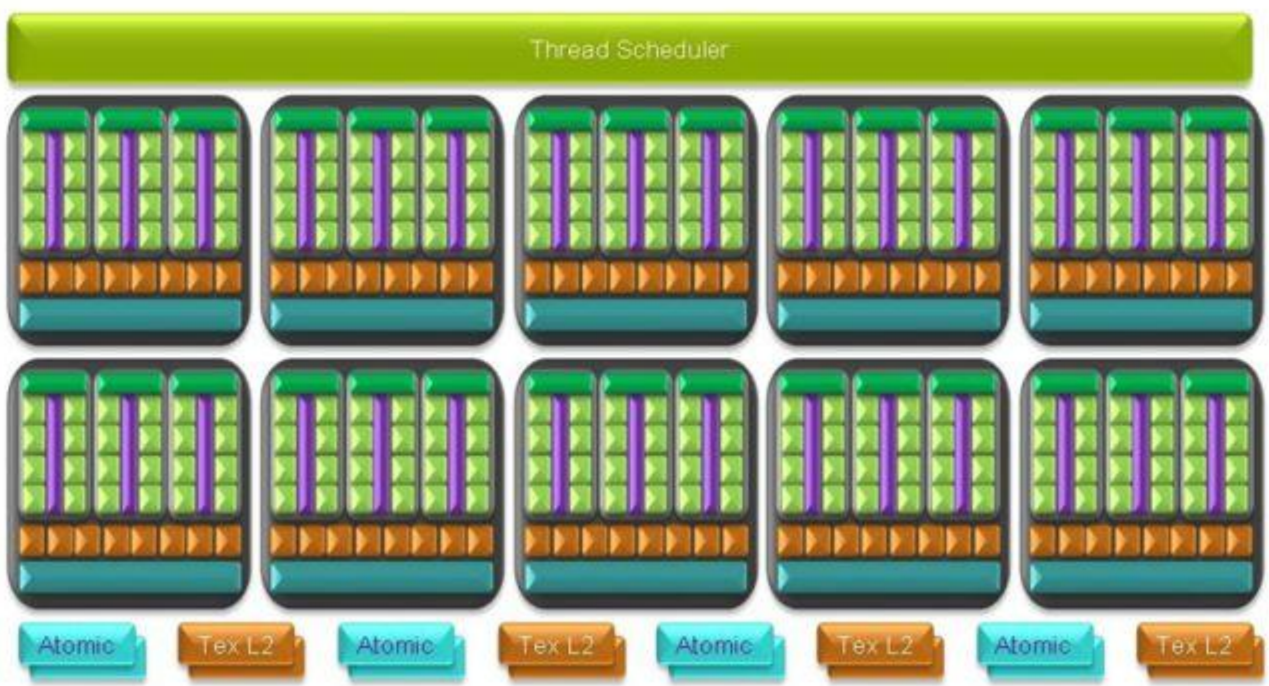
- General-Purpose Graphic Processing Unit (GPGPU)



# GPU architecture (NVIDIA GTX280)

- 30 multi-processors, 240 streaming processors
- 1024 MB GDDR3, 141.7GB/s
- 933 GFlops

GeForce GTX 280 Parallel Computing Architecture

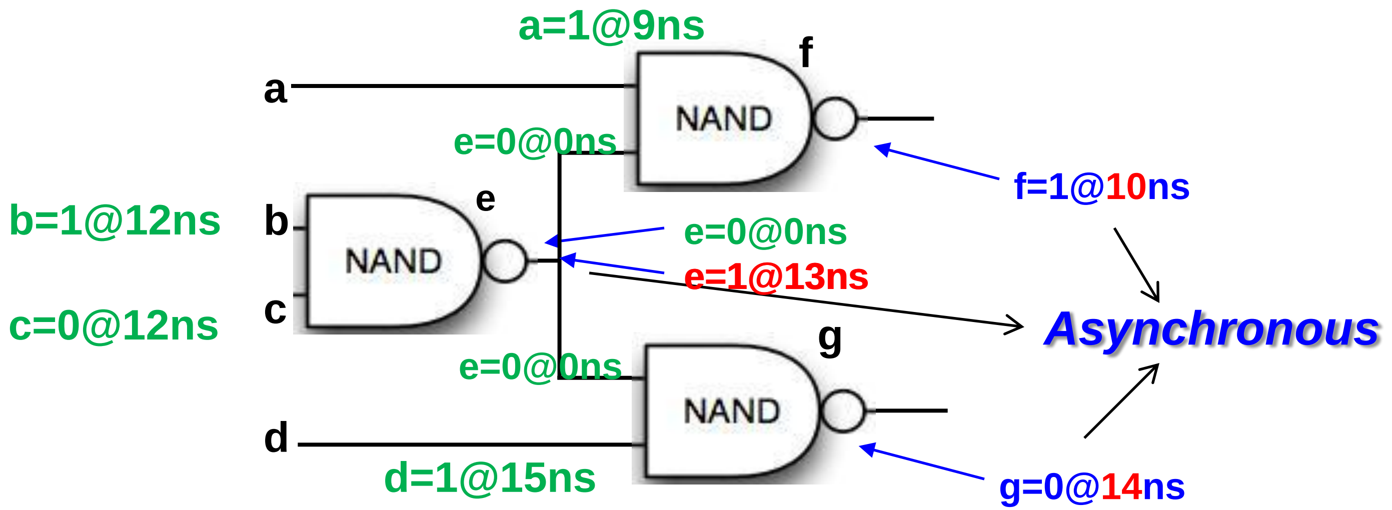


# Simulation algorithms

- Oblivious algorithm
  - All gates are evaluated at each cycle
  - Simple, efficient static gate scheduling
  - Inefficient due to redundant evaluation
- Event-driven algorithm
  - A gate is simulated only if its input value changes
  - Synchronous
    - Events simulated simultaneously have the same simulation time.
  - Asynchronous
    - Events having different simulation time can be simulated simultaneously
    - *Chandy-Misra-Bryant algorithm*

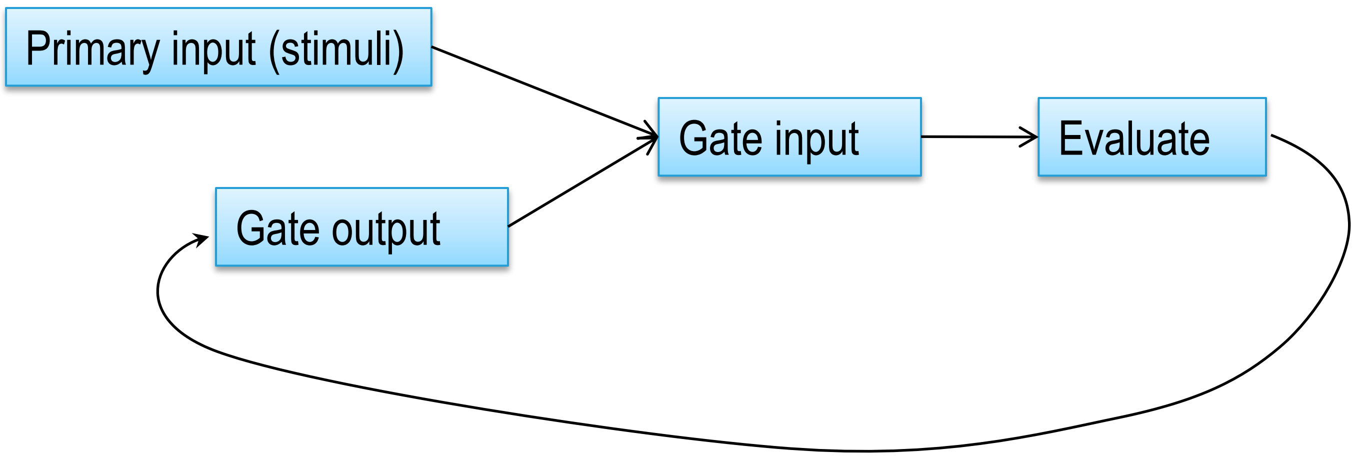
# Simulation algorithms (cnt.)

- Chandy-Misra-Bryant algorithm
  - Event-driven
  - Asynchronous
  - Conservative
  - Parallel and distributed




Assume the gate delay of NANDs is 1ns

# Algorithm revisited



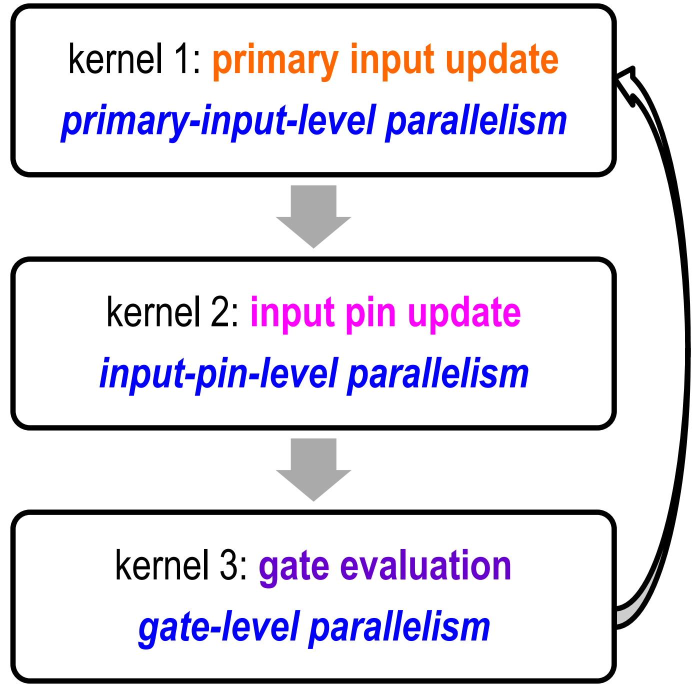

- Each round
  - Stimuli are fetched from primary inputs to the gate inputs
  - Gate outputs are sent to the gate inputs
  - Events arriving at a gate are stored in a priority queue w.r.t. timestamp
  - Each gate evaluate the event with the smallest timestamp if possible

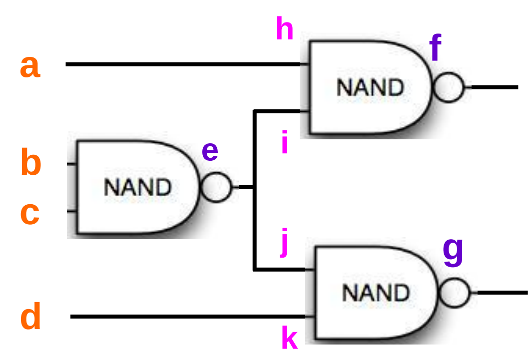
# Outline

- Motivation
- Background
- Parallel Logic Simulator
- Experiments
- Conclusion

# Basic simulation flow

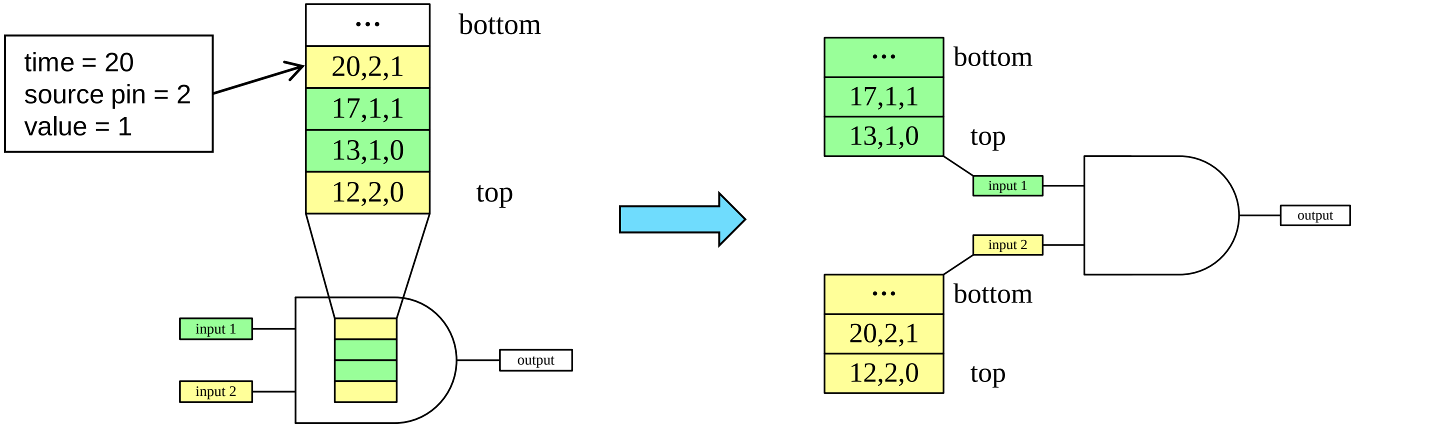
```
while not finish
    // kernel 1: primary input update
    for each primary input(PI) do
        extract the first message in the PI queue;
        insert the message into the PI output array;
    end for each

    // kernel 2: input pin update
    for each input pin do
        insert messages from output array to input pin;
    end for each

    // kernel 3: gate evaluation
    for each gate do
        extract the earliest message from its pins;
        evaluate the message and update gate status;
        write the gate output to the output array;
end while
```



# Priority queue transformation

- **Problem : Maintenance of priority queue on GPU is inefficient**
  - In the original CMB algorithm, each gate stores the events arriving at all its inputs in a centralized priority queue w.r.t. the timestamp.
  - Maintenance of priority queue introduces many branches, which are inefficient for SIMD-like GPU model.
- **Solution**
  - Divide the priority queue of a gate into multiple FIFOs w.r.t. its input pins

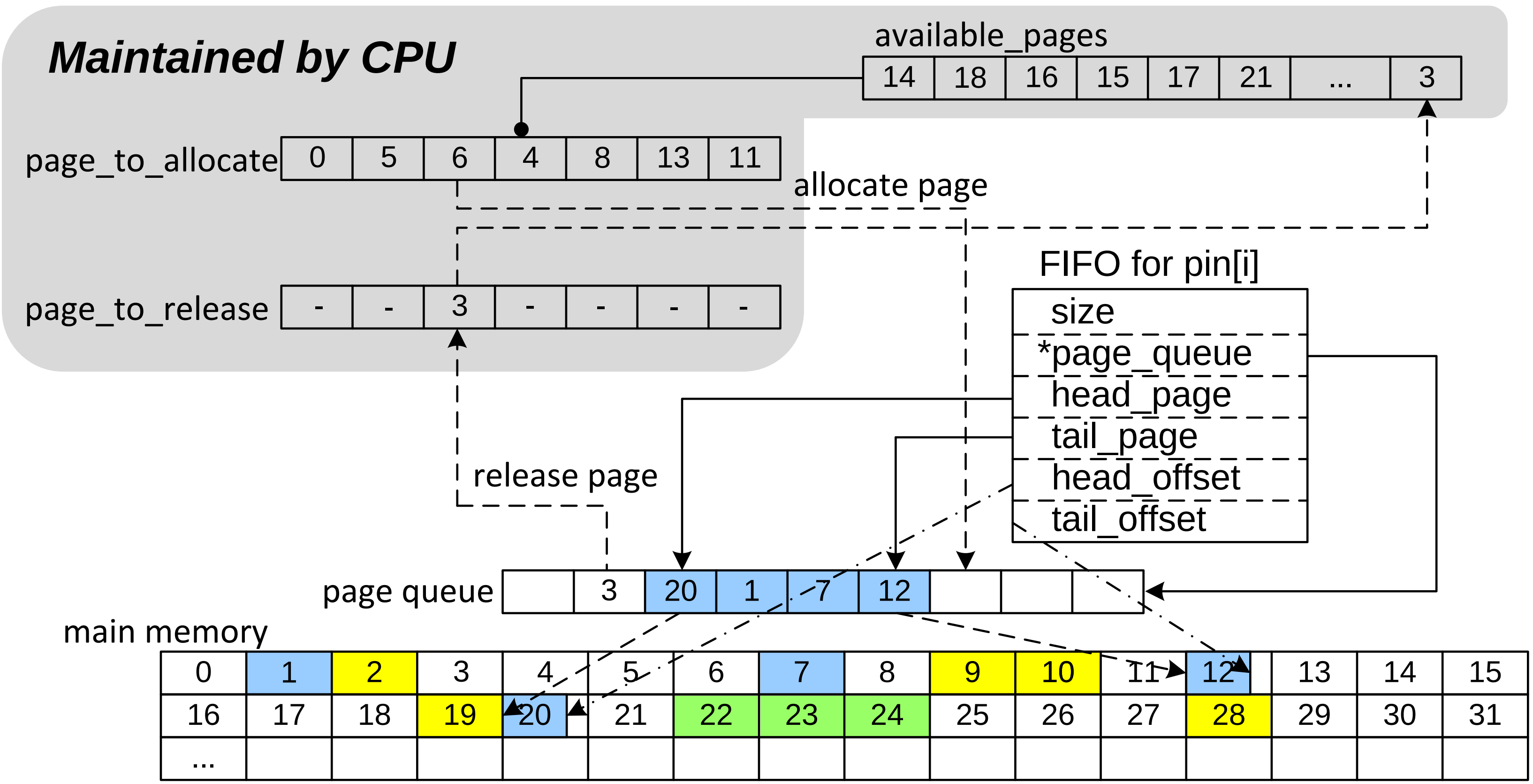


# Dynamic memory management

- Problem :
  Memory demands of each pin_FIFOs are very different
  - #messages on each gate vary drastically
  - #messages on the same gate varies from time to time
  - Static memory pre-allocation is inefficient

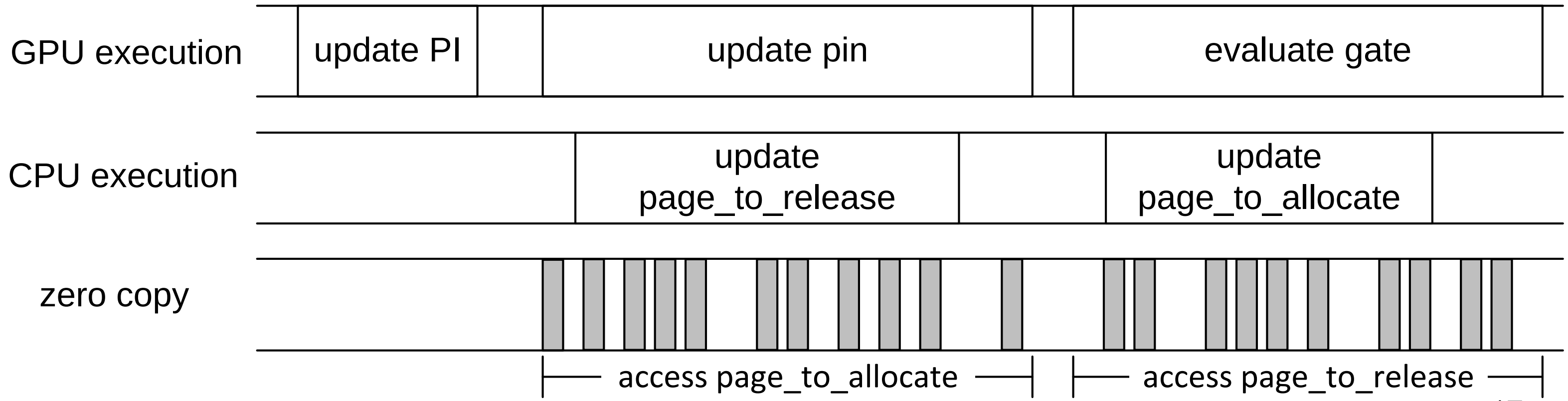
- Solution : Memory paging on GPUs
  - A memory paging mechanism is introduced for the management.
  - GPU-friendly allocate and release methods are provided.

# Dynamic memory management



# GPU/CPU co-processing

- **Problem: memory management needs CPU assistance**
  - **Sequential execution**
  - **Overhead of memory copy**
- **Solution: GPU/CPU Co-processing**
  - **Overlap *update_pin*(GPU) with *update page_to_release*(CPU)**
  - **Overlap *evaluate_gate*(GPU) with *update page_to_allocate*(CPU)**
  - **Adopt *zero-copy* in CUDA**



# Performance optimization

- Memory optimization
  - Coalesced access
    - AOS(Array-of-Structure) → SOA(Structure-of-Array)
  - Hierarchical memory : locality
    - Texture memory : circuit topological information
    - Constant memory: truth table
- Gate reordering
  - Reducing branches
    - Gates sharing the same inputs are closer to each other
    - Gates of the same type are closer to each other

# Outline

- Motivation
- Background
- Parallel Logic Simulator
- Experiments
- Conclusion

# Experiments

- Platform
  - Intel Core 2 Duo E6750 2.66 GHz
  - Memory : DDR2 4GB
  - NVIDIA GTX 280 (DDR3 1GB)
- Baseline
  - A *synchronous* event-driven simulator on single core
- Test cases
  - ITC99
  - OpenCores

| DESIGN | #GATES | #PINS | DESCRIPTION |
|---|---|---|---|
| AES | 14511 | 35184 | AES encryption core |
| DES | 61203 | 138419 | DES3 ENCRYPTION CORE |
| M1 | 17700 | 42139 | 3-stage pipelined ARM core |
| SHA1 | 6212 | 13913 | Secure Hashing algorithm core |
| R2000 | 10451 | 27927 | MIPS 2000 CPU core |
| JPEG | 117701 | 299663 | JPEG image encoder |
| B18 | 78051 | 158127 | 2 Viper processors and 6 80386 processors |
| NOC | 71333 | 181793 | Network-on-Chip simulator |

# Performance

| Design | Simulated cycles | CPU simulation time (s) | GPU simulation time (s) | Speedup |
| --- | --- | --- | --- | --- |
| AES | 42,935,000 | 109.90 | 4.45 | **24.7** |
| DES3 | 30,730,000 | 183.11 | 4.50 | **40.7** |
| SHA1 | 2,275,000 | 56.66 | 0.41 | **138.2** |
| R2000 | 28,678,308 | 9.20 | 3.15 | **2.9** |
| JPEG | 26,132,000 | 136.33 | 43.09 | **3.2** |
| NOC | 1,000,000 | 5389.42 | 347.95 | **15.5** |
| M1 | 99,998,019 | 118.48 | 22.43 | **5.3** |
| b18 | 19,125,000 | 37.30 | 11.49 | **3.3** |

***Speedup is closed related to the stimuli density!***

# Irregular distribution of events

- Irregularity
  - some pins are very hot, some are very cold
- Testcase
  - 50,000 simulation cycles with random stimuli

| Peak number of messages | DES3 | R2000 | M1 | JPEG | NOC |
|---|---|---|---|---|---|
| 0-9 | 68170 | 15747 | 24788 | 178728 | 157891 |
| 10-99 | 63895 | 11567 | 16506 | 117820 | 23297 |
| 100-999 | 3960 | 53 | 663 | 2913 | 590 |
| 1000-9999 | 2253 | 2 | 3 | 202 | 0 |
| 10000-50000 | 85 | 0 | 4 | 0 | 15 |

# Outline

- Motivation
- Background
- Parallel Logic Simulator
- Experiments
- Conclusion

# Conclusion

- Parallel Logic Simulator on GPUs
  - Developed a GPU-friendly CMB algorithm
  - Designed efficient dynamic memory management
  - Utilized GPU/CPU co-processing to hide overhead
  - Achieved high performance
- Future works
  - Study the scalability on industry-strength circuits
  - Apply the techniques to system and RTL simulations

# THANK YOU!