



# Accelerating Logic Simulation on GPUs

February 2010

## Introduction

Graphic processing units (GPUs) are specialized processors used to offload intensive image processing computations from the central processing unit (CPU) of a computer. They are characterized by a high degree of parallelism, ability to support complex image processing sequences, very high bandwidth to local memory caches, and support for floating point operations. GPUs enable fancy animations and high quality video playback on personal computers, but the same computational power can also be put to use in accelerating many scientific computations.

GPUs from Nvidia and AMD (formerly ATI) lead currently in terms of capabilities. The upcoming Larrabee processor from Intel uses the well known x86 architecture for the core - potentially making them easier to program. Unlike a multi-core CPU, a GPU can have several tens, maybe even hundreds, of processing cores. The graph below shows a comparison between processing power of GPUs versus CPUs in the recent past.

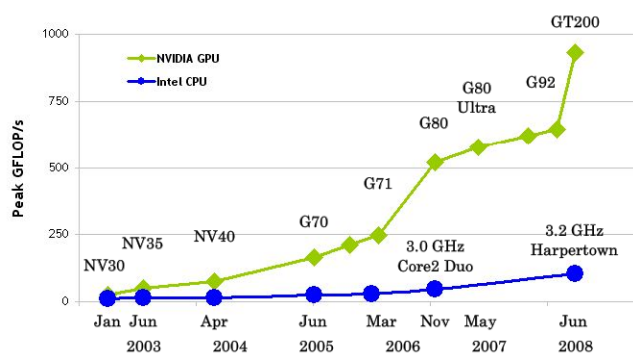


Figure 1: Source - Nvidia

## GPUs and Event Driven Simulation

Applications that benefit most from GPUs are those with innate parallelism. Interesting research problems arise when we have constraints that restrict the natural parallelism. In such cases, ingenuity is needed to cast the problem in a way that is well suited to a GPU platform.

Event driven simulation is used to verify the functional behaviour of logic circuitry. A hardware description language such as Verilog is typically used for the circuit description, and a test bench is used to apply inputs and verify outputs of the behavioural model. This is a crucial step in the process of designing large digital systems, and can be very time consuming in large systems such as microprocessors and systems-on-a-chip (SoC). Ways of speeding up such simulations are therefore very much in demand.

Although the simulated circuit has natural parallelism (many gates), the activity of the gates follows a chain of dependence from inputs to outputs. Typically, about 10-20% of gates in a digital circuit switch values during a given clock cycle. A simulator that only responds to such events, rather

than simulating every gate in every cycle, would be more efficient. However, a cycle-based simulator that goes through every gate in every cycle has fewer overheads, and can be competitive too.

## GPUSIM

GPUSim was developed to speed up gate level simulations of digital systems. We use a hybrid of the event-driven and cycle-based techniques - all gates are considered for evaluation in every cycle, but the computation is restricted to gates that see input activity. Fig. 2 shows the memory organization and the approximate bandwidths - clearly it is beneficial to keep host interactions to a minimum, as the bandwidth to host is much lower than within the device.

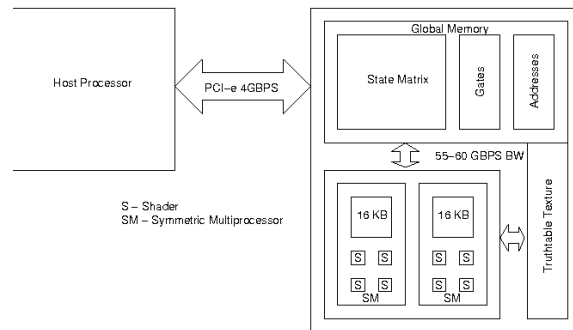


Figure 2: Memory organization

The simulation is divided up into groups of gates known as *ranks*, and many cycles of execution are grouped together to make efficient use of the GPU memory. The planning of the memory layout impacts whether the memory reads and writes can be *coalesced*, yielding the high bandwidths required.

The system was evaluated on several known circuit benchmarks, as well as real examples such as an error correcting code. One observation is that this approach works much better for large circuits than small ones, due to the overheads involved in setting up a simulation. Specific comparisons to other simulators are prone to error because the underlying processing cores are different, but serve to get a feel for the behaviour. A speedup of about 3 was observed on large sequential circuits, while over 30x was observed for large combinational circuits, where even more optimizations are possible. Table 1 shows a comparison between GPUSim and ModelSim (an established commercial simulator). For other circuits, the structure (number of gates and depth of logic) play an important part in the actual speedup obtained.

Circuit	# gates	Speedup
Adder	4232	0.13
Multiplier	34418	3.36
LDPC encoder (combinational)	52941	38.12

Table 1: Speedup comparisons with Modelsim