EDA Design Flow Acceleration using GPGPUs*

Lalith Suresh P.[1], Navaneeth Rameshan[1], Ashwin Narayan[1],
Mark Zwolinski[2], M.S.Gaur[1], V. Laxmi[1], Virendra Singh[3]

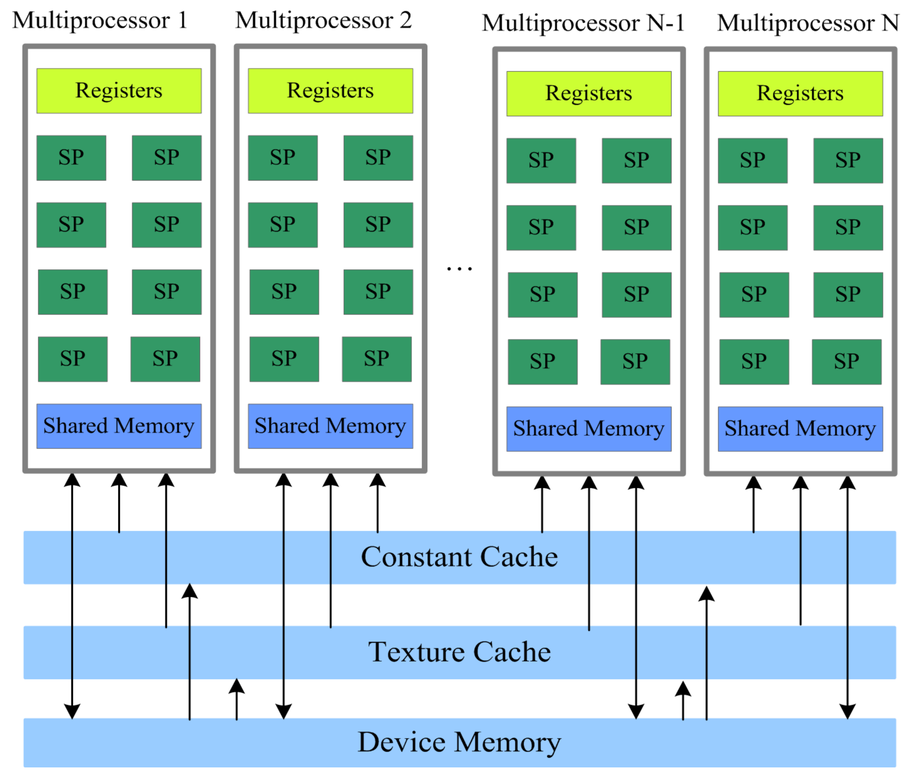
MNIT, Jaipur, India

INTRODUCTION

Various phases of design automation for today's complex hardware are computationally intensive. Existing EDA tools are not powerful enough to meet these computational demands within the time-to-market constraints. There is a growing need to adapt these tools to exploit parallel architectures which are becoming increasingly affordable. we propose two methods to speed up functional verification, using General Purpose Graphics Processing Units (GPGPUs). Functional Verification is the process of validating the functionality of a logic design and verifying that the logic design conforms to the specification.

CUDA PROGRAMMING MODEL

- CUDA is a parallel programming model and supports multi-threading.
- Leverages the computational capacity of the GPUs for non-graphics applications.
- Parallel sections of code, known as *kernels* are executed on the GPU.
- Architecture constitutes multiprocessors, each of which can execute a *block* of upto 512 concurrent *threads,* each running same kernel code.
- Multiprocessors consist of stream processors (SP) each of which handles a thread.
- CUDA threads are lightweight, having little creation overhead and provides faster switching compared to CPU threads.
- Only one kernel can be executed at a time, by launching a *grid* of thread blocks which scales across different multiprocessors.
- Shared memory allows co-operation of threads within a block while threads in different block can't co-operate.

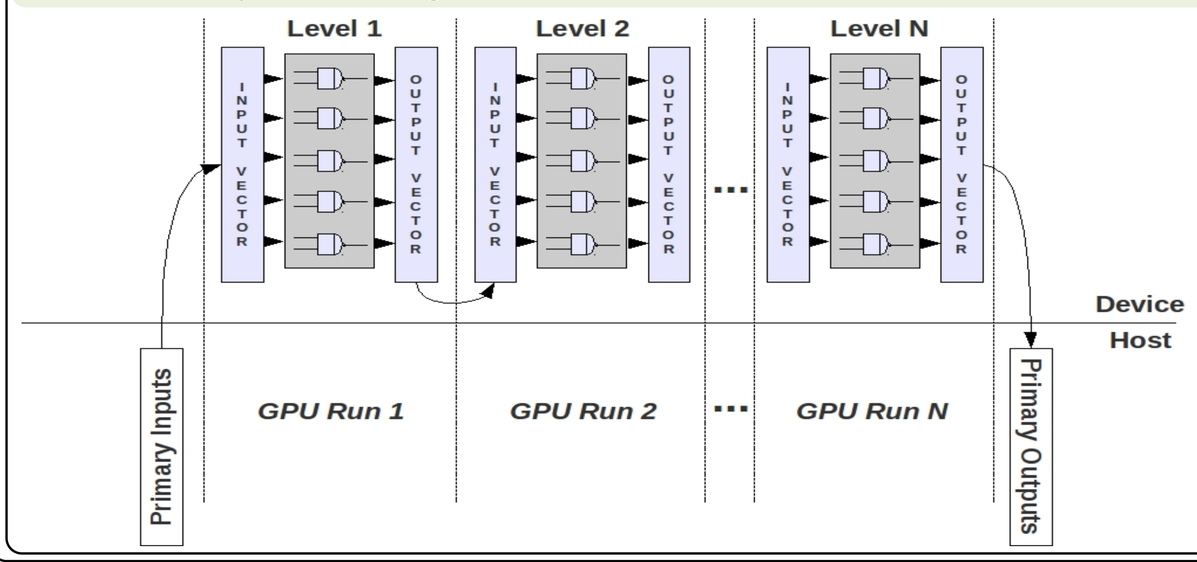


PROPOSED METHOD

- We first parse the ISCAS '85 benchmark netlist to generate per-gate information required for the CUDA based simulation.
- The next step involves vertical levelisation of the circuit. Gates in a level are not dependent on each other.

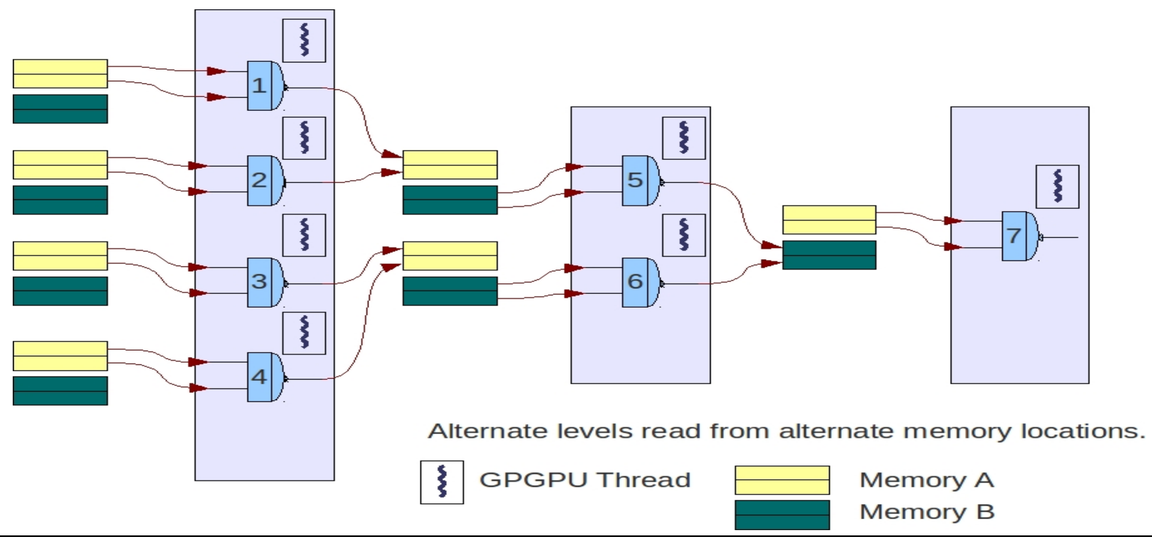
DATA PARALLEL APPROACH

- In the first run, we take the first level of the circuit and the primary inputs (PIs) and evaluate the level for different set of inputs.
- The output of the gate is written to the output vector in accordance with the fanouts of the gate. This output vector forms the input vector for the next level as shown in figure.
- This procedure occurs using threads within the same CUDA block.
- By launching multiple blocks, we can generate outputs for multiple input vectors in parallel.
- The parallelism attainable within a level is limited to the number of threads that can be launched within a block which is 512 and can be overcome by evaluating a level in chunks.



PIPELINE APPROACH

- There are 2 memory locations for any input to a gate to avoid overlap during a write operation.
- In the first run, only the first level is evaluated and the output is propagated to the first memory location in accordance with the fan outs of the gate.
- In the next run, the first level is evaluated for the next set of input-vectors and propagated to the second memory location, while at the same time the second level is evaluated on the inputs obtained from previous run.
- Alternate levels read from alternate memory locations with the input vectors forming a pipeline as shown in figure.



RESULTS

CIRCUIT	SEQUENTIAL	DATA PARALLEL	PIPELINING	SPEED-UP DATA PARALLEL	SPEED-UP PIPELINING
C1355	2.4 SECS	0.91 SECS	2.38 SECS	2.64	1.01
C1908	6.01 SECS	4.08 SECS	5.49 SECS	1.47	1.09
C2670	5.57 SECS	3.23 SECS	9.34 SECS	1.72	0.6

CIRCUIT	SEQUENTIAL	DATA PARALLEL	PIPELINING	SPEED-UP DATA PARALLEL	SPEED-UP PIPELINING
C3540	7.6 SECS	5.21 SECS	7.02 SECS	1.46	1.08
C5315	19.28 SECS	10.37 SECS	17.73 SECS	1.86	1.09
C6288	20.05 SECS	8.19 SECS	17.21 SECS	2.45	1.17

[1]Malaviya National Institute of Technology, Jaipur, India
email: { gaurms, vlaxmi }@mnit.ac.in

[2]University of Southampton, UK
email: { mz }@ecs.soton.ac.uk

[3]Indian Institute of science, Bangalore, India
email: { viren }@serc.iisc.ernet.in

*This research work is supported by INDO-UK research grant UKIERI-2008-SA043

DATE 2010, March 8-12, 2010, Dresden, Germany (Friday workshop on **Designing for Embedded Parallel Computing Platforms: Architectures, Design tools and Applications**)