Preface

In recent times, serial software applications have no longer enjoyed significant gains in performance with process scaling, since microprocessor performance gains have been hampered due to increases in power and manufacturability issues, which accompany scaling. With the continuous growth of IC design complexities, this problem is particularly significant for EDA applications. In this research monograph, we evaluate the feasibility of hardware platforms such as custom ICs, FPGAs, and graphics processors, for accelerating EDA algorithms. We choose applications which contribute significantly to the total runtime of the VLSI design flow and which have varied degrees of inherent parallelism in them. We study the acceleration of such algorithms on these alternative platforms. We also present an automated approach to accelerate certain specific types of uniprocessor subroutines on the GPU.

This research monograph consists of four parts. The alternative hardware platforms, along with the details of the programming model used for interfacing with the graphics processing units, are discussed in the first part of this monograph. The second part of this monograph studies the acceleration of an algorithm in the *control-dominated* category, namely Boolean satisfiability (SAT). The third part studies the acceleration of some algorithms in the *control plus data parallel* category, namely Monte Carlo based statistical static timing analysis, circuit simulation, fault simulation and fault table generation. In the fourth part of the monograph, we present the automated approach to generate GPU code to accelerate certain software subroutines.

Book Outline

This research monograph is organized into four parts. In Part I of this research monograph, we discuss alternative hardware platforms. We also provide details of the programming model used for interfacing with the graphics processor. In Chapter 2, we compare and contrast the hardware platforms that are considered in this monograph. In particular, we discuss custom-designed ICs, reconfigurable architectures such as FPGAs, and streaming processors such as graphics processing units

(GPUs). This comparison is performed over various criteria such as architecture, expected performance, programming model and environment, scalability, time to market, security, and cost of hardware. In Chapter 3, we describe the programming environment used for interfacing with the GPUs.

In Part II of this monograph we present hardware implementations of a control-dominated EDA problem, namely Boolean satisfiability (SAT). We present approaches to accelerate SAT using each of the three hardware platforms under consideration. In Chapter 4, we present a custom IC-based hardware approach to accelerate SAT. In this approach, the traversal of the implication graph and conflict clause generation are performed in hardware, in parallel. Further, we propose a hardware approach to extract the minimum unsatisfiable core for any unsatisfiable formula. In Chapter 5, we discuss an FPGA-based hardware approach to accelerate SAT. In this approach, we store the clauses in the FPGA slices. In order to solve large SAT instances, we partition the instance into 'bins,' each of which can fit in the FPGA. The solution of SAT clauses of any bin is performed in parallel. Our approach also handles (in hardware) the fact that the original SAT instance is partitioned into bins. In Chapter 6, we present a SAT approach which employs a new GPU-enhanced variable ordering heuristic. In this approach, we augment a CPU-based complete procedure (MiniSAT), with a GPU-based approximate procedure (survey propagation). In this manner, the complete procedure benefits from the high parallelism of the GPU.

In Part III of this book, we study the acceleration of several EDA problems, with varying amounts of control and data parallelism, on a GPU. In Chapter 7, we exploit the parallelism in Monte Carlo based statistical static timing analysis and accelerate it on a graphics processor. In this approach, we map the Monte Carlo based SSTA computations to the large number of threads that can be computed in parallel on a GPU. Our approach performs multiple delay simulations of a single gate in parallel and further benefits from a parallel implementation of the Mersenne Twister pseudo-random number generator on the GPU, followed by Box–Muller transformations (also implemented on the GPU). In Chapter 8, we study the acceleration of fault simulation on a GPU. Fault simulation is inherently parallelizable and requires a large number of gate evaluations to be performed for each gate in a design. The large number of threads that can be computed in parallel on a GPU can be employed to perform a large number of these gate evaluations in parallel. We implement a pattern and fault parallel fault simulator, which fault-simulates a circuit in a levelized fashion. We ensure that all threads of the GPU compute identical instructions, but on different data. We study the generation of a fault table using a GPU in Chapter 9. We employ a pattern parallel approach, which utilizes both bit parallelism and thread-level parallelism. In Chapter 10, we explore the GPU-based acceleration of the model card evaluation of a circuit simulator. Our resulting code is integrated into a commercial fast SPICE tool, and the overall speedup obtained is measured. With careful engineering, we maximally harness the GPU's immense memory bandwidth and high computational power.

In Part IV of this book, we present an automated approach to accelerate uniprocessor subroutines which are required to be executed multiple times within an

application, on independent data sets. The target hardware platform is a general-purpose graphics platform. The key idea here is to partition the subroutine into kernels in an automated fashion, such that multiple instances of these kernels, when executed in parallel on the GPU, can maximally benefit from the GPU's hardware resources. This approach is detailed in Chapter 11.

The approaches presented in this monograph collectively aim to contribute toward enabling the VLSI CAD community to accelerate EDA algorithms on different hardware platforms.

College Station, TX Kanupriya Gulati
College Station, TX Sunil P. Khatri
October 2009