Chapter 1

Introduction

In the past decade, the semiconductor industry has experienced a challenging evolution in the complexity of digital integrated circuit (IC) designs: increasing integration density and die size has made it possible to design chips with hundreds of millions of transistors. At the same time, the growing importance of getting products to market quickly has increased the pressure on design teams to deliver new products and new technologies in a short time span: typical development times are less than two years. In this fast evolving landscape, ensuring that the digital ICs are functionally correct is crucial: an error in the design's functionality can delay product deployment by months. Moreover, ICs are embedded in many safety critical applications, where a design flaw can lead to the loss of life.

Due to the importance of design correctness, a significant fraction of engineering development time and resources are devoted to it. Design verification involves checking that the initial functional design of a circuit is correct against the specifications. It consists of a whole set of activities aimed at acquiring a reasonable certainty level that a circuit will function correctly, under the assumption that no manufacturing fault is present. Validating the functionality of digital circuits and systems is an increasingly difficult task. Multiple chip design projects are reporting that approximately 70% of their design time is spent in verification. This is due to the growing complexity of the designs that has not been accompanied by improvements in functional verification techniques.

Part of this high resource allocation is due to the fact that verification methodologies are still very experimental, there is almost no standard approach or methodology in any area pertaining to verification, and the whole process is still largely manual. The cause of this high investment cost can be attributed to the high complexity of the task at hand, but also to the lack of support from the design automation industry. While on the design synthesis front, they have made available tools that can, at least partially, support the complexity and the challenges of such highly integrated designs, on the verification front there has been almost a complete lack of support. The only widely deployed tools for verification are logic simulators. Such simulators are a key tool for the verification team to gain insight in the actual functionality of the design under test; nonetheless, they cannot be used to guarantee the general correctness of any aspect of a design, and thus, their usefulness as push-button verification tools is still limited.

1.1 Functional validation

Designers normally try to ensure correctness by developing multiple set of tests to stimulate the digital design and by inspecting the results of the simulations. These techniques are the only ones available today that can cope with the complexity and scale of current digital ICs; at the same time they have significant limitations.

Today, logic simulation is the mainstream approach for the validation of large synchronous systems because of its scalability : CPU time is proportional to the design size and test length. Simulation is also flexible: practical cycle-based simulators allow for circuits with multiple clocks and the ability to mix cycle-based and event-based simulation. Unfortunately, the fraction of the design space which can be explored by simulation is miniscule, especially for large designs. Only one state and one input combination of the design under test are visited during each simulation cycle. Moreover the test stimuli must be hand crafted by the designer to cover those areas of the design that she wishes to validate. For a large, complex system, it is impossible to test or simulate all possible inputs or sequences of inputs. One measure of the quality of verification for a design that is commonly used in industry is state coverage. State coverage counts how many different configurations of a system have been visited, and thus verified, by the simulation. When the size of the design space, or total number of reachable configurations, is known, the state coverage can be expressed as a fraction of this size. Furthermore, simulation inputs are usually based on the design specification and thus are only aimed at verifying that the design performs all the primary activities indicated in the specification document. However, it is often the case that complex systems manifest unforeseen behavior for corner case situations that were not planned in the specification. Most often, designers are unaware of behavior that results as a by-product of the interactions among different modules and that was unaccounted for in the specification document. Thus these cases do not get checked, while they may as well have negative consequences on the overall behavior of the system.

Overall, designers are discovering that their simulation-based verification approaches are inadequate as ICs become more complex through increased size, more aggressive pipelining, and greater use of concurrency.

1.2 Formal verification

In its broadest meaning formal verification consists of proving formally that the implementation of a digital IC is compatible with its specification. In a formal verification approach, the desired functionality of the system needs to be completely specified, then a formal model of the system needs to be constructed – the implementation – and finally, formal reasoning is used to show that this formal model satisfies the specification. Formal verification techniques have the potential of providing more general results than traditional validation methods: it is possible for instance to guarantee that a specific property holds for a design under all possible input stimuli. Due to the complexity of constructing a complete specification and of formally proving the compatibility between implementation and specification, this approach is infeasible for state-of-the-art hardware designs.

Traditionally formal methods have been mainly explored in academic research settings and only applied to problems of very limited size. However, the recent “verification crisis” – that is the inability of current validation techniques to provide sufficient confidence in the correctness of the design – has spurred increased interest for this approach to verification which has led to new algorithmic solutions and to new approaches that compromise on the completeness of the verification in order to reduce its complexity. In particular, the past ten years have seen efforts in developing commercial formal verification tools; however, so far these tools have not shown the required robustness to be included in the industry mainstream verification methodology, they have only been applied to experimental projects. One of the main limitations shown by these first attempts are in the complexity of the algorithms involved in formal verification: usually their demand for computing resources far exceeds the resources available at most design sites. Another limitation has been the amount of engineering effort that is required for providing a complete formal specification of the design.

As a consequence, formal techniques have only be applied to very simple designs that do not represent the complexity of the digital ICs developed in industry.

1.2.1 Symbolic simulation

Symbolic simulation is a promising approach to formal verification. The key idea is to simulate the design using Boolean symbolic variables instead of constant binary values at the combinational inputs of the circuit's model. During simulation, the approach derives Boolean expressions based on the initial symbolic variables and the functionality of each of the circuit components. At the end of each simulation step, we obtain a set of Boolean expression representing implicitly all configurations – or set of states – that are reachable by the circuits in one clock cycle with an appropriate set of inputs. Thus, this approach allows the complete behavior of a design in a specific state to be verified with a single simulation step, under all possible inputs simultaneously. Thus, it has the potential of 1) verifying many configurations of the design in parallel and providing much better coverage than traditional logic simulation. 2) providing the ability to prove time-bound properties of the design.

The problem with this approach is that it requires extensive manipulation of Boolean expressions, which in turn, often exhaust the memory resources of the host computer even on designs of limited complexity.

1.3 Contributions of the thesis

This thesis addresses the robustness and scalability limitations of symbolic simulation and presents two algorithms that dramatically reduce the memory requirements compared to current techniques.

The first technique, called Cycle-Based Symbolic Simulation, simplifies the Boolean expressions involved in symbolic simulation while trying to maximize the range that they span. The resulting simulator maximizes the level of parallelism achievable with a limited amount of memory. This technique performs very well from a scalability standpoint, and achieves a high level of parallelism in terms of test vectors run through the simulator, while maintaining a low memory profile. We found, however, that a better parameterization technique was needed in order to support the complete state exploration that is typical of symbolic simulation. To this end, we introduced an efficient algorithm that exposes the disjunctive support decomposition properties of a Boolean function.

The disjoint support decomposition of a scalar function $F : \mathcal{B}^m \to \mathcal{B}$, consists of finding other, simpler functions G and H such that: $F(x_1, \cdots, x_m) = G(H(x_1, \cdots, x_h), x_{h+1}, \ldots, x_m)$. An exact solution to this problem that had exponential complexity in the number of variables of the function, was proposed in the late '50s. The algorithm we present in this thesis takes as input a binary decision diagram (BDD) representation of a Boolean function and restructures it in its disjoint decomposition components. The worst case complexity of the algorithm we propose is only quadratic in the size of this representation, making it well suited for use with complex functions.

This algorithm is applied to transform and simplify the Boolean expressions involved in symbolic simulation so that the simulation requires fewer memory resources while producing the same original quality of results. Experimental results show that these solutions provide often more than 10 orders of magnitude better performance (in test vectors per second) than a logic simulator, while, at the same time, improve the scale of symbolic simulation by handling up to thousands more symbolic variables.

1.4 Organization of the thesis

In order to provide the context for our work, we present a quick overview of the steps involved in the design cycle of a digital IC in Chapter 2. The chapter also presents the models of digital systems used in verification and the main algorithms for both traditional simulation and formal verification.

We then present the first technique mentioned in the previous section, *Cycle-Based Symbolic Simulation* in Chapter 3. The chapter provides a formal presentation of the algorithm and simulation results that compare the performance of CBSS to that of a logic simulator.

In the following two chapters, we move away from the main topic of verification to focus on a core topic of Boolean algebra, the theory of Disjoint Support Decompositions (DSD). This theory is the central idea behind our second symbolic simulation technique. We decided to introduce it only at this point, so that the reader has a chance of seeing the type of approach that we take at verification with our first technique, before diving into core theoretical material. The first of these two chapters introduces the main results of the *Disjunctive Support Decomposition theory* of Boolean functions, the second presents our novel algorithm for decomposing the BDD of a function in its disjoint support components. Chapter 5 discusses the algorithm in detail and shows results on the decomposability of many Boolean functions involved in indutrial benchmarks.

This theory and novel algorithm are then deployed in Chapter 6 to present a new symbolic simulation algorithm based on the DSD properties of the state vector functions of simulation. This approach is called *Disjoint Support Decomposition based Symbolic Simulation*. The results compare this approach to a plain symbolic simulator. We conclude the thesis with Chapter 7, where we provide a discussion of the methods presented in the thesis and some directions for future research.

Each of the chapters starts with a presentation of our objectives for the chapter and a review of the previous research developed on its specific topic. The central part covers a formal presentation of the material. When a chapter presents a new algorithm, we conclude presenting simulation results obtained by implementing the algorithm and testing it on indutrial testbenches.