Chapter 1

Introduction

In an ideal world, a field-programmable gate array (FPGA) vendor would use hundreds or thousands of benchmark circuits in determining the architecture of a next generation device, and in developing the associated automatic placement and routing software for it. In this way, the architectural design space would be adequately explored and the best software algorithms would be used and well tested. Similarly, a commercial developer of general computer-aided design tools would require quality benchmark circuits to evaluate the effectiveness and efficiency of various new algorithmic techniques. The use of benchmarks is crucial for all facets of computer-aided design because the vast majority of interesting and practical problems are NP-hard and can only be solved by heuristic or approximate techniques. Similarly, FPGA design is inherently inexact, so architectural questions must also be answered empirically.

A fundamental problem exists for CAD and FPGA research: Because the device and its tools are new, there are few large (or correctly sized) designs available to perform these kinds of exploration and evaluation. In the case of FPGAs, some circuits will always exist by purchasing benchmarks from customers migrating from larger gate-arrays or synthesis from high-level design languages but these rarely suffice. Cutting-edge CAD vendors have no such luxury, and are forced to expend considerable effort creating benchmarks internally.

The proprietary nature of benchmarks, and the rarity of commonly accepted benchmark standards means that it is often very difficult to compare competing heuristic solutions for a given problem. In an effort to address this issue, researchers at the Microelectronics Centre of the University of North Carolina (MCNC) [74] have collected approximately 200 public

benchmarks and have made them freely available by anonymous ftp. These circuits are very popular for empirical validation in academic research, but largely spurned by industry as too small (about half are 100 nodes or fewer).

A related effort by the PREP Corporation [56] has defined a number of small representative benchmarks, with the goal of evaluating the logic capacity and speed of FPGAs. The metric is "how many" of the individual (but joined) circuits can be packed in a given device, and how fast the resulting compound circuit will run. Most researchers believe that this method does not address how logic characteristics change with size, especially with respect to interconnect usage, nor does it yield interesting test-cases for CAD software.

Random graphs are another possibility, particularly attractive because there is an infinite supply. Random graphs have often been used for the evaluation of partitioning algorithms for large circuits (where there are no available benchmarks). In particular, a number of classic partitioning papers [45, 46, 48] have done empirical validation with random graphs. One of the contributions of this thesis is to show that arbitrary random graphs are *not* realistic proxies for real circuits, and exhibit increasingly bad behaviour as the problem size increases.

A traditional graph-theoretic approach to NP-hard problems is to restrict the input domain, then identify an efficient deterministic algorithm for a subclass of graphs. For example, it is NP-hard to exactly determine the minimum number of colours $\chi(G)$ required to form a "proper colouring" of an arbitrary input graph G [29]. But, if G is known to be P_4-free—for any path $xyzw$ in G it is always the case that one of the edges xz, xw or yw also exists in G—it has been shown [14] that a straightforward greedy algorithm exists to determine $\chi(G)$ exactly in linear time.

One could claim that domain restriction is not directly applicable to practical CAD problems because a boolean network really is just an arbitrary graph: "for any G, an orientation of its edges and labeling of its nodes with primitive boolean functions (e.g. $\wedge$, $\vee$, $\neg$) provides a boolean network computing *some* function." However, our fundamental belief, as we will discuss further, is that such an arbitrary labelling of a general graph does not result in a *practical* or *realistic* boolean network as would be produced by a human designer or an automated synthesis tool. Without necessarily ruling out certain types of graphs as *possible* inputs to a software tool, we can perform data analysis to identify the *expected* structure of realistic inputs, and tune our tools to the distribution of expected

inputs.

It is a well known fact that relatively simple heuristics can often perform well—in practice the difficulty associated with random or arbitrary graphs does not occur, because real circuits exhibit much more structure than would be found randomly. For example, channel routing is known to be NP-hard [33, 50], but the search for more complicated algorithms or a guaranteed approximation scheme for the basic algorithmic problem is no longer "interesting" because existing heuristic algorithms work well and quickly [50] for all known data. This situation is analogous to the conclusions of Shew [63] who studied the application of graph colouring to scheduling with a conflict graph. He found that, even though arbitrary conflict graphs are always possible, real-life input *tends* to have P_4-free or nearly P_4-free structure: the heuristic algorithm was working well in practice because it was optimal for large subgraphs of the input it was given.

In the design and evaluation of good inexact architectures and heuristic algorithms it is crucial to understand the type of data that the FPGA or algorithm will be required to handle and thus to trust the test data that are used in its creation. The goals of this research are to provide a greater understanding of the graph-theoretic structure of real-life digital circuits and to apply this knowledge to the generation of high quality benchmark circuits.

In this thesis, we present a careful methodology for dealing with the benchmarking problem. We define a number of new graph-theoretic properties of combinational and sequential circuits. These properties are based on well known and important features of digital logic such as combinational delay, fanout, and reconvergent fanout. We also propose metrics that capture the inherent local structure of circuits not seen in random graphs. Given this better understanding of the combinatorial structure of circuits, we define the new problem of "parameterized circuit generation" and solve this problem by proposing and fully implementing a new algorithm. Since both of these efforts contain a large body of empirical and heuristic work, the final proof is in the resulting circuits themselves. We give conclusive evidence that the circuits we produce are realistic benchmarks by contrasting them both to existing benchmarks and to random graphs. As a byproduct of this *validation* step, we show the non-viability of purely random graphs as benchmarks.

The software tools CIRC and GEN arising from this work are freely available, and themselves form an important contribution to the community. CIRC is a tool for performing

analysis on an input circuit, and producing statistical and structural information about it. GEN takes a list of parameters (discussed in Chapters 3 and 4) and produces a circuit which satisfies the user's specification.

CIRC and GEN have been downloaded under an academic license by more than 30 persons representing more than 20 companies and academic institutions, and have been installed by the author for use at Xilinx, Altera, Actel, and Hewlett Packard Corporations.

1.1 Overview of the Thesis.

The research described in this thesis has three distinct aspects: *characterization* of digital circuits, *generation* of parameterized random benchmarks, and *validation* of circuit quality.

In Chapter 2, we provide further context and motivation for this work, and discuss previous work on circuit characterization and wireability, and circuit generation.

Chapters 3 and 4 address the characterization issue, asking the question "What is a circuit?" Chapter 3 deals with combinational circuits, introducing new characteristics of circuits based on combinational delay, and proposing a new theoretical characterization of reconvergent fanout and metrics for capturing the inherent local structure in combinational circuits. In Chapter 4, we investigate the more complex sequential circuit. We give an abstract model of a sequential circuit, defined in terms of combinational building blocks, and add a number of new characteristics specific to sequential circuits.

In Chapter 5 we formally define the parameterized circuit generation problem for combinational and sequential circuits, and give an algorithm to solve it. The algorithm has been fully implemented in the tool GEN, and we discuss a number of implementation details from this experience.

Chapter 6 deals with the final research topic, empirical validation. Using GEN to "clone" existing benchmarks from their parameterization, we can compare post-place and global route metrics of wireability between real circuits, their GEN-clones, and random graphs of the same size. We use this method to give strong empirical evidence both that our algorithm and tool provide good benchmarks, and that standard models and methods for random graphs do not.

We conclude and describe areas for future work in Chapter 7.

The historic development of the research differs from how it will be presented herein.

The combinational characterization of circuits from Chapter 3, and a predecessor of the algorithm of Section 5.2 for combinational generation, though without locality characteristics, (GEN 1.0) first appeared in the 1996 Design Automation Conference [41]. This work since been submitted for journal publication [42]. The model of Chapter 4 for sequential circuits, excluding sequential reconvergence (Section 4.3) and the updated algorithm for sequential generation (GEN 3.0) was presented at the 1997 ACM Symposium on Field-Programmable Gate Arrays [39]. A journal version is in preparation. Sequential reconvergence (Section 4.3), the work on locality analysis (Section 3.5) and its effects on the generation algorithm have not yet been published outside of the thesis.