APPLICATIONS OF CLONE CIRCUITS TO ISSUES IN PHYSICAL-DESIGN

Michael D. Hutton

Altera Corporation
101 Innovation Drive.
San Jose CA 95134
mhutton@altera.com

Jonathan Rose

University of Toronto
Department of Electrical and Computer Engineering
10 King's College Rd.
Toronto, Ontario Canada M5S 3G4
jayar@eecg.toronto.edu

Abstract

In a companion paper of this session [1] we formally defined the notion of equivalence classes of circuits which are physical clones of an existing benchmark seed circuit created by the GEN tool.

Here we use well-known partitioning and placement tools to study the behaviour of clone circuits. We outline a simple methodology for using equivalence classes of clone circuits for benchmarking. We then apply the methodology to the example of netlist partitioning by generating a large set of clones of a starting benchmark set in order to statistically distinguish two netlist partitioning algorithms.

The primary purpose of this paper is to outline the acceptable and valid uses of clone circuits for benchmarking. In addition to specific examples we will also discuss practical experience with circuits produced by the current version of GEN, so that the user is aware of for what purposes GEN is currently useful, and for what purposes user aid or further research is required.

1. INTRODUCTION

There are two primary motivations for the automatic generation of benchmark circuits. The first is to generate circuits with specific features (most notably size when circuits of a given size are unavailable) and to perform more statistically valid experiments. It is the latter issue that we focus upon in this paper.

In previous work [2,3] we reported on the tools CIRC which extracts a signature from a sequential netlist and GEN which generates a new clone circuit subject to a parameterization derived from this signature. Here we will analyze the behaviour of GEN clone circuits in a given equivalence class [1] in order to understand the bias and variance we can expect within equivalence classes. We will then apply the methodology of generating clone circuits to do a simple test of quality between two competing partitioning algorithms. We will close with a discussion on the practical applications of GEN clone circuits.

2. EQUIVALENCE CLASS VARIANCE

Given that we can generate clone circuits for a given seed circuit, we want to see to what extent the clones exhibit biased behaviour with respect to the original circuit and to what extent they exhibit variance with respect to each other. For this purpose we will use the problem of placing the netlists on the unit grid.

The experiments performed here use benchmark designs from Altera Corporation. These designs vary in size from 627 to 1284 4-input LUTs (lookup-tables), the size of a small programmable logic device. A given design is extracted and synthesized using Altera's MAX+PLUS2 software and then translated into the BLIF netlist format [4]. This translation involves some sanitization to remove combinational cycles, secondary signals and multiple clocks which are generally not handled by academic research tools. The result is a BLIF format sequential netlist of 4-LUTs and DFFs.

From the BLIF netlist we run CIRC to create a signature in a format which is used as input to GEN. Using GEN we create 100 clones of the seed circuit. We then place and route both the original circuit and the clones using VPR [5,6]. For use of the VPR tool we thank Vaughn Betz from the University of Toronto.

VPR does placement and routing of the circuit in a simple grid-like programmable-logic architecture and reports statistics such as the maximum number of tracks used in any row or column and the total wirelength (in grid-segments) used by the circuit. Though more complex architectural features are supported by VPR we will keep to the simple version of an n by n grid.

We have reported previously on the fact that an individual clone can be somewhat biased from the original circuit in ease-of-placement. In [2] it was shown that for a sample of 20 seed-clone pairs in the 800-1000 LUT range there was an average wirelength bias of about 17% for purely combinational seed circuits and in [3] that there was an average bias of 35% for sequential circuits. The same research showed that for random graphs with the same number of nodes and edges the bias was 119% and 151%, respectively, for combinational and sequential circuits. Though the circuits produced by GEN are clearly biased away from the placement metrics of the seed circuits they are dramatically more comparable than seed-random pairings on the same number of nodes and edges. Experience has shown us that the degree of bias for both GEN clones and random circuits increases with the size of the base circuit.

An issue not covered in this previous work was the variation between different clones in the same equivalence class. This is interesting because it gives us an idea of the completeness of the signature characterization as we have defined it: A high variance in the metric measured would imply that the characterization of the equivalence class is fundamentally flawed – i.e. we are generating random graphs. A low variance with a constant bias would more likely imply an incomplete specification – i.e. missing parameter(s). Obviously the best case is for both low bias and low variance (but positive variance so that we do get different circuits).

For each of 8 seed circuits, we generated 100 clones as outlined above. A typical case, as represented by 100 clone circuits generated from the seed circuit ALTR03 (Figure 1), is that almost all circuits are within 5% from the average wirelength of the class. However, in line with the previously reported bias, the average wirelength of the class differed from the seed by 57%. (Note that we're using larger circuits than previously.)

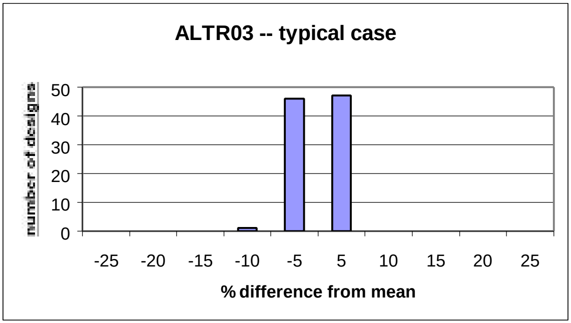

Figure 1. Typical distribution of wirelength around the mean for 100 circuits in an equivalence class (ALTR03).

The greatest-variance case for the 8 circuits studied is shown in Figure 2. Approximately 20% of circuits were between 5% and 15% from the average.

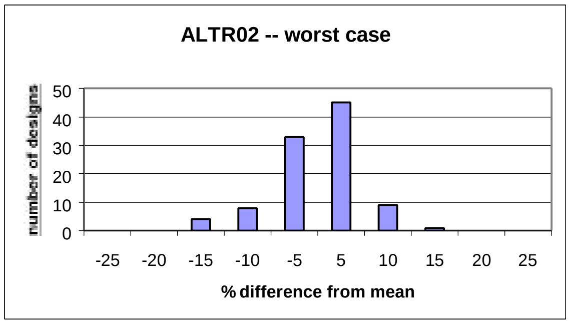

Figure 2. Worst case (most variation over 8 benchmarks) distribution of wirelength around the mean for 100 circuits in an equivalence class (ALTR02).

To contrast the distribution of wirelength for circuits in the same equivalence class with a distribution of circuits not in the same equivalence class we generated 100 circuits using GEN. These circuits were forced to have 100 PI, 50 PO and 1000 LUTs, but were otherwise unconstrained (except for the default distributions of the software as discussed in [2,3]). Note these are not random graphs; they are GEN circuits whose signatures are drawn from default distributions rather than from a common seed circuit. The corresponding distribution of wirelength around the mean is shown in Figure 3 and we see that the distribution is dramatically more varied.

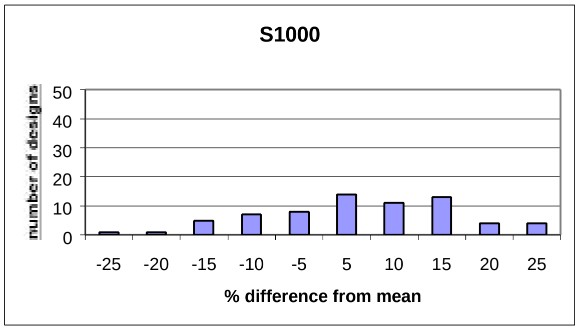

Figure 3. Distribution of wirelength around the mean for 100 circuits generated to have 100 PI, 50 PO, 1000 LUTs, and otherwise using GEN defaults.

Table 1 shows the wirelength (w) reported for the original circuit, and the average (wavg) and standard deviations (wsd) observed for the individual sets of 100 clones in the equivalence class of each seed circuit, and the percentage difference (wdiff) of the mean from the seed circuit. Also shown are the corresponding statistics for maximum track-count (t, tavg, tsd and tdiff) as reported by VPR.

	W	WAVG	WSD	WDIFF	T	TAVG	TSD	TDIFF
altr01	12219	21579	901	77	7	12.4	0.69	77
altr02	4989	7859	354	58	6	9.6	0.66	60
altr03	8560	16215	353	89	7	10.8	0.43	55
altr04	8197	12053	255	47	8	9.4	0.49	17
altr05	5711	7478	176	31	9	7.7	0.48	-15
altr06	6246	9093	253	46	6	9.5	0.50	58
altr07	8983	14716	515	64	6	11.4	0.61	91
altr08	7494	12670	222	69	7	11.0	0.36	57
			avg:	60			avg:	50
s1000		9611	2156			8.3	1.30	

Table 1. Wirelength and track statistics for 8 Altera circuits and their 100 corresponding clones, and for 100 1000-LUT random circuits.

Conclusions: With respect to currently generated GEN circuits can make several points. The "quality" of the circuits, as measured by bias, is significantly better than random circuits but not as close as we would like to the seed circuit. The seed circuit is almost always requires less overall wirelength than the clones we generate. However, the variance is well in line with both expectations and our desire – we want varied circuits and +/- 5 percent is basically the amount of variance we would want to generate for reasonable experiments. For randomly generated benchmarks the variance is much more significant – larger than would be desired.

3. BENCHMARKING METHODOLOGY

Now we will address an important issue in CAD benchmarking: given that algorithm A has out-performed algorithm B by 10%

on two different test circuits, what conclusion can we make? Is it reasonable to conclude that A is better, or is this simply noise because both the algorithms are heuristic? We have two fundamental problems. The first is that the two circuits may not be representative of the typical input to the program. For this we currently have no solution. The second problem is that we are observing noise in the behaviour of the algorithms for these circuits because the algorithms are inherently heuristic: essentially we have a result which has no statistical significance. It is here that the use of clone circuits can play a role in our ability to benchmark.

The following simple methodology follows naturally from the definition of clone circuits and equivalence classes: Given a small set of initial benchmark circuits, use the process outlined in Section 2 to generate a large number of clone circuits equivalent to each seed circuit. Apply each of the clone benchmarks to the problem under consideration, and measure the appropriate statistical metric(s) to distinguish the multiple approaches. Then, in addition to the original circuits, consider the behaviour of the class as a whole to the problem solution.

For example, if our goal is to analyze the effectiveness of two placement algorithms we could apply each to 100 clone circuits of each seed circuit and then compare the distribution of results between the two algorithms. If our goal is to determine whether an experimental programmable logic architecture requires 80 wires per row or if 60 is sufficient we perform place-and-route on the two different parts and analyze the number of fits and no-fits which result. In both cases we can gain more finely grained information from the large number of circuits than would be seen by looking only at the small number of initial benchmarks.

We point out that simply generating large numbers of circuits does not, in itself, allow us to make more accurate experiments. In order to apply this methodology, we are relying on the fact that the circuits being generated by GEN do have similar properties, as exhibited by their low variance, and that relative comparisons are thus justified.

4. EXAMPLE: NETLIST PARTITIONING

In order to illustrate the use of GEN clones, we will apply the above methodology to distinguish two well-known partitioning algorithms.

For the first algorithm we obtained an implementation of the Fiduccia-Mattheyses partitioning algorithm [7] from Charles Alpert's website [8]. This code was originally attributed to Shantanu Dutt and Wenyong Deng at the University of Minnesota Electrical Engineering Department and modified by Alpert for various netlist formats.

The second partitioning algorithm is the original implementation of hMetis by Karypis *et .al.* [9,10] from the University of Minnesota website. This more recent algorithm is expected, *a priori*, to have better results than the FM algorithm.

We performed bipartitioning on 100 clones each of 8 Altera benchmark circuits (800 in total) and the original circuits and recorded the cut-size reported by each tool. We then calculated the mean and standard deviation and calculated the 68% (mean +/- 1 std. dev.) and 95% confidence intervals (mean +/- 2 std. dev.) (*c.f.* [11]) for each of the two algorithms on each of the 8 equivalence classes.

The results of the experiment are displayed pictorially in Figure 4. For a given circuit, we have 4 lines: from top to bottom, the 68% and 95% confidence intervals for hMetis and then the 68% and 95% confidence intervals for FM (each calculated over the 100 circuits in the equivalence class). We observe (as expected) that the hMetis algorithm outperforms basic FM significantly: for half the circuit classes the 68% confidence intervals do not even overlap.

Given the large sample size, we can also get reasonable confidence intervals for the difference in mean cut-size between

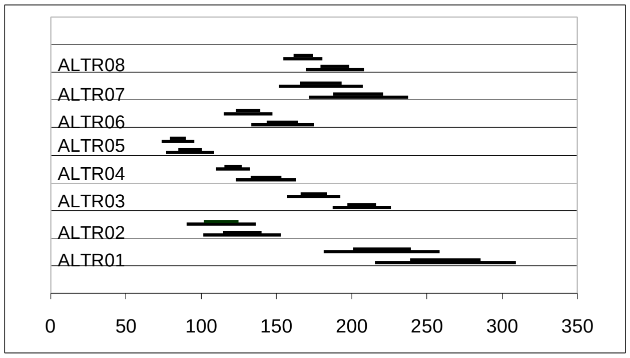

Figure 4. 95% and 65% confidence intervals for cut-size as reported by hMetis (top) and FM (bottom), taken over equivalence classes for 8 seed circuits.

the two algorithms. Table 2 shows this data. The average of the eight 95% confidence intervals for the difference in sample means is (11.3, 15.0), so we are 95% confident that hMetis will outperform FM by between 11% and 15%.

	LOW	HIGH	LOW%	HIGH%
altr01	36.5	48.3	13.9	18.4
altr02	10.5	17.3	8.3	13.6
altr03	29.2	34.4	14.1	16.6
altr04	19.7	24.1	13.7	16.9
altr05	6.3	10.1	6.8	10.9
altr06	20.2	25.4	13.1	16.5
altr07	20.8	29.2	10.2	14.3
altr08	18.9	23.5	10.0	12.4
			11.3	15.0

Table 2. 95% confidence intervals for difference in mean cut-size between hMetis and FM.

Though not the point of this paper, it is clear that hMetis is a superior algorithm to the basic FM implementation.

5. DISCUSSION

Our goal is to provide greater confidence in our ability to do benchmarking on algorithms and architectures. This paper has addressed two specific issues.

First, we can see that equivalence classes of clones generated from seed circuits using GEN exhibit relatively good variance properties but are biased. From this we conclude that we can use these circuits to perform relative comparisons between different tools or different architectures but not to extract absolute parameters about the problem (e.g. expected cut-size of a partitioning algorithm or or correct channel width in a programmable logic device).

The most effective and reasonable applications of clone circuits would be to benchmark the relative performance and run-time of physical design algorithms (partitioning, placement, routing), using the methods outlined in this paper. Some authors (e.g. [12,13]) have already used GEN circuits for this purpose.

A secondary use of clone circuits is as a source of sanitized designs. Production-quality software tools and prototype architectures must deal with a myriad of difficult special cases (e.g. secondary signals, multiple clock-domains, combinational cycles). Since GEN currently outputs synchronous circuits in a single clock-domain, the tool can concentrate on the core algorithmic issues without having to deal with special-case situations. This is, of course, also a drawback when the user later wants to exercise those aspects of the program.

Applications for which GEN cicruits are not currently viable would be in determining, for example, the correct number of lines to put in a channel of a programmable logic device. This is because of the bias we observe in the clone circuits relative to the seed (real) benchmark. We believe that the primary reason that this occurs is that GEN does not generate a true design-hierarchy as would be present in real-life designs, and thus diverges from natural hierarchical structure as the size of the circuit grows. We hope to address this issue in the future.

6. FUTURE WORK

As just mentioned, the primary deficiency in GEN today is the lack of hierarchy in circuits it generates. This results in circuits which do not scale properly as the size increases.

The most useful improvement to GEN would be to have it generate sub-circuits according to a partitioning hierarchy, combining our approach with that of Darnauer and Dai [14]. Though there is currently some hierarchy in GEN, both in the way that sequential circuits are generated and available to the user for hand-specification, it is limited to a single level.

A general hierarchical stitching mechanism would also allow us to introduce parameterized datapath elements (e.g. library modules such as multipliers, adders, multiplexors, etc.) and other library modules into our circuits, which would allow us to test specific features of architectures and software tools.

Other specific features which would improve the practicality of the circuits produced by GEN would be the facility for multiple clock domains, memory, bidirectional pins, and lookup-table programming (allowing synthesis algorithms to be exercised).

References

[1] M. D. Hutton and J. Rose, "Equivalence classes of clone circuits for physical design benchmarking," in *Proc. 1999 Int. Symp. Circuits and Systems (ISCAS'99)*, Orlando, Florida, 1999.

[2] M. D. Hutton, J. P. Grossman, J. Rose, and D. G. Corneil, "Characterization and parameterized generation of synthetic combinational benchmark circuits," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 17, no. 10, Oct, 1998, pp. 985-996.

[3] M. D. Hutton, J. Rose, and D. G. Corneil, "Generation of synthetic sequential benchmark circuits." *in Proc. 5th ACM/SIGDA Int. Symp. FPGAs, (FPGA'97)*, Feb. 1997, pp 149-155.

[4] S. Yang, "Logic synthesis and optimization benchmarks," v. 3.0 Tech. Rep., Microelectronics Center of North Carolina, Research Triangle Park, NC. Available on the MCNC website at http://www.cbl.ncsu.edu.

[5] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. 7th Int. Conf. Field-Programmable Logic*, Aug. 1997. pp 213-222. See also http://www.eecg.toronto.edu/~jayar/.

[6] V. Betz, J. Rose and A. Marquardt, "Architecture and CAD for deep-submicron FPGAs," Kluwer Academic Publishers, 1999. ISBN 0-7923-8460-1.

[7] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in Proc. 19th IEEE Design Automation Conference (DAC)," 1982, pp. 175-181.

[8] C. Alpert, Personal website of partitioning benchmarks and partitioning code at http://vlsicad.cs.ucla.edu/~cheese.

[9] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel hypergraph partitioning: application in the VLSI domain," in *Proc. 34th IEEE Design Automation Conference (DAC)*, 1997, pp. 526-529. (To appear, *IEEE Trans. on VLSI Systems*."

[10] G. Karypis and V. Kumar, "hMetis: a hypergraph partitioning package, V1.5.3," University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN 55455. Nov. 1998.

[11] M. R. Spiegel, "Theory and problems of probability and statistics – SI (metric) ed.", mcGraw-Hill, Singapore, 1980.

[12] Y. Sankar and J. Rose, "Trading quality for compile time: ultra-fast placement for FPGAs," To appear, *Proc 7th ACM/SIGDA Int. Symp. FPGAs (FPGA'99*), Feb, 1999.

[13] J. Swartz, V. Betz, and J. Rose, "A fast routability-driven router for FPGAs," In *Proc 6th ACM/SIGDA Int. Symp. FPGAs (FPGA'98)*, Feb, 1998. pp. 140-149.

[14] J. Darnauer and W. Dai, "A method for generating random circuits and its application to routability measurement," in *Proc. 4th ACM/SIGDA Int. Symp. FPGAs (FPGA'96)*, Feb 1996, pp 66-72.