Structural Analysis and Generation of Synthetic Digital Circuits with Memory

Steven J.E. Wilton
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
(604) 822-1263
stevew@ece.ubc.ca

Jonathan Rose
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada M5S 3G4
(416) 978-6992
jayar@eecg.toronto.edu

Zvonko Vranesic
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada M5S 3G4
(416) 978-5032
zvonko@eecg.toronto.edu

Abstract

One of the most difficult aspects of experimental reconfigurable architecture or CAD tool research is obtaining sufficiently large benchmark circuits. One approach to obtaining such circuits is to generate them stochastically. Current circuit generators construct combinational and sequential logic circuits. Many of today's devices, however, are being used to implement entire systems, and often these systems contain on-chip storage. This paper describes a circuit generator that constructs circuits containing significant amounts of memory. To ensure the circuits are realistic, we have performed a detailed structural analysis of such circuits; this analysis is also described in this paper.

Keywords

Structural Circuit Analysis, Stochastic Circuit Generation, FPGAs

I. Introduction

Until recently, Field-Programmable Gate Arrays (FPGA's) have been used to implement small logic circuits, often the "glue-logic" portions of larger systems. Today, however, they are being used to implement entire systems; often these large systems require on-chip memory. In order to evaluate architectures and tools supporting on-chip memory, benchmark circuits containing both logic and memory are required.

Obtaining enough sufficiently large benchmark circuits, however, is difficult. One approach is to use stochastic circuit generators which generate realistic "circuits". Such generators have been developed before [1], [2], [3], [4], [5], [6]. These generators, however, all produce circuits containing only logic. In this paper, we describe a generator that produces realistic circuits containing both logic and memory.

Circuits containing both logic and memory are significantly different than those containing only logic. Most circuits with memory contain a handful of large arrays to implement that storage; this is very different than the large number of small logic elements that typically make up a logic circuit. In addition, the connections between memory arrays is very different than the connections between logic elements. Each memory array typically has many parallel connections, while a typical logic element only has a few inputs and outputs. Thus, a circuit generator that explicitly targets circuits containing memory is required.

A key issue in the use of stochastically generated circuits is to ensure that the circuits are realistic. In order to understand the nature of digital circuits with memory, we performed a

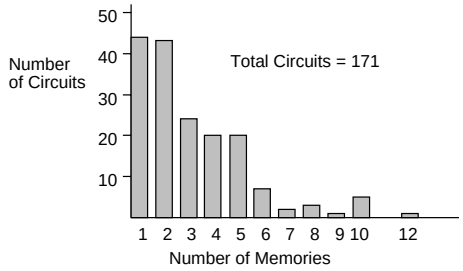


Fig. 1. Distribution of number of memories required.

detailed structural analysis of 171 real circuits, each with both memory and logic components. Although we did not have complete netlists for these circuits, we were able to gather statistics regarding the number, sizes, shapes, and other key characteristics of these memories. In addition, we were able to identify common patterns for the interconnect between memory and logic. This data is summarized in Section II. These statistics and patterns were then incorporated into the circuit generator; the generator is described in Section III.

II. Structural Analysis

The analysis is based on 171 circuits, each containing between 1 and 16 memories. Data regarding these circuits was obtained from several sources: recent conference and journal articles, industrial designers, and a study obtained from Altera Corporation. We were not able to obtain netlists for these circuits, but we did obtain characteristics of how the memories were used and connected. Most circuits were initially designed to be implemented on a gate array or custom chip.

A. Memory Configurations

Figure 1 shows the distribution of the number of memories in each circuit. As the graph shows, circuits with one or two memories are common while the demand for higher numbers of memories decreases rapidly.

Figure 2 shows the distribution of widths and depths of the memories in the analyzed circuits. Since we did not have memory width/depth information for all circuits, the results only show the distribution for 268 of the 533 memories. As the graphs show, the proportion of memories with depths in each power-of-two-interval between 8 and 2048 is roughly constant, while the memory width distribution peaks at about 8, and falls off below 8 and above 16.

Finally, from our analysis, we saw that 69% of the widths and 74% of the depths were powers-

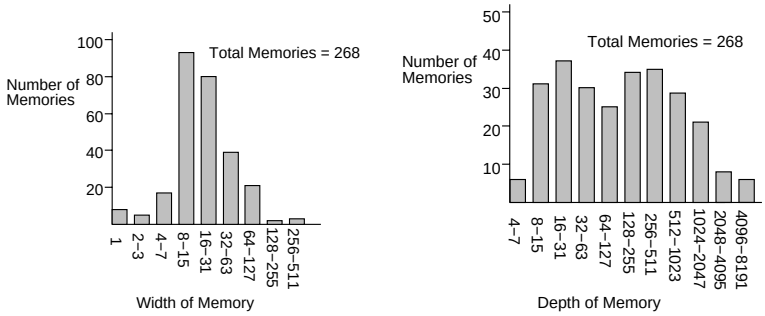


Fig. 2. Distributions of memory widths and depths.

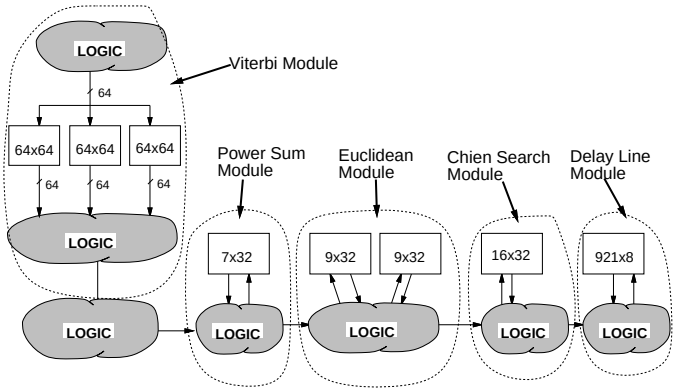


Fig. 3. Forward error correction decoder datapath from [7].

of-two. In addition, 16% of the memories were used as ROMs, and 13% were multi-ported.

B. Memory Clustering

Memories connect to logic through their address pins, data-in pins, data-out pins, and other control pins (write enable and perhaps a clock). Each of these sets of pins is driven by (or drives) one or more logic subcircuits. We will refer to subcircuits driving the data-in pins as *data-in subcircuits*, subcircuits driving the address pins as *address subcircuits*, and subcircuits driven by the data-out pins as *data-out subcircuits*.

Memories in our example circuits typically appear in "tightly-connected" groups. We refer to each of these groups as a *cluster*. More precisely, a cluster is defined to contain one or more memories in which all data-in ports are connected to a common logic subcircuit (or set of logic subcircuits) *or* in which all data-out ports are connected to a common logic subcircuit (or set of logic subcircuits). As an example, the circuit in Figure 3 contains five clusters.

To quantify common sizes and numbers of clusters in circuits, we examined 31 of our example circuits, and counted the number of clusters and the number of memories in each cluster. Figure 4

summarizes the results. We also observed that, of those clusters containing more than one memory, 95% consist of memories with the same width and 75% consist of memories with the same depth.

Finally, Figure 5 shows the distribution of the number of data-in and data-out subcircuits connected to the memories in each cluster in the 31 circuits. These measurements are approximate, since in some circuits it is difficult to deduce how a piece of logic can best be represented by a set of subcircuits. As the graphs show, the memories in most clusters are connected to only a single data-in subcircuit and a single data-out subcircuit.

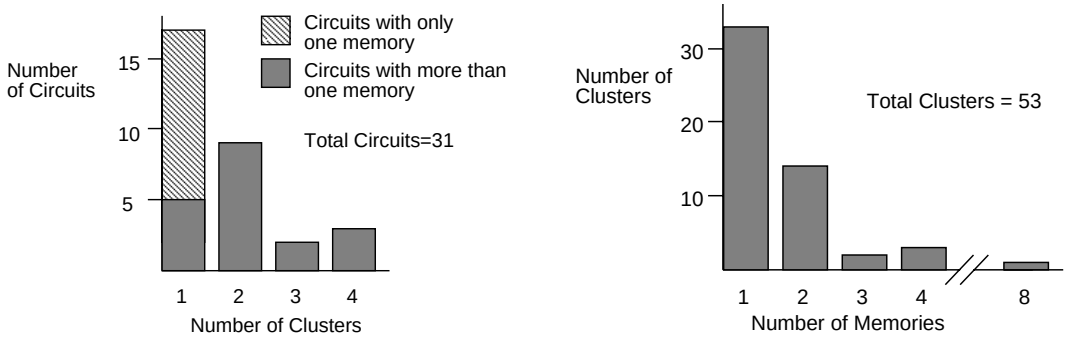
C. Interconnect Patterns

We then considered the manner in which memories and logic subcircuits are interconnected within a cluster. We have identified three common patterns, as shown in Figure 6.

We refer to the first pattern (shown in Figure 6(a)) as *point-to-point*. In this pattern, each logic subcircuit pin connects to exactly one data pin in one memory block Example circuits that employ this pattern can be found in [8], [9], [10], [11], [7], [8], [12], [13], [14], [15], [16], [17], [18], [19], [20].

We refer to the pattern shown in Figure 6(b) as *shared-connection pattern*. A typical use of such a pattern is described in [21], in which the data-in ports of a scratch-pad memory and first-in first-out buffer are connected to a bus. Other examples can be found in [15], [22], [23], [24], [25], [26].

The third pattern occurs only when there are an equal number of subcircuits as memories within a cluster. Although this is actually a special case of the other two patterns, it occurs



a) number of clusters distribution b) number of memories per cluster

Fig. 4. Cluster statistics.

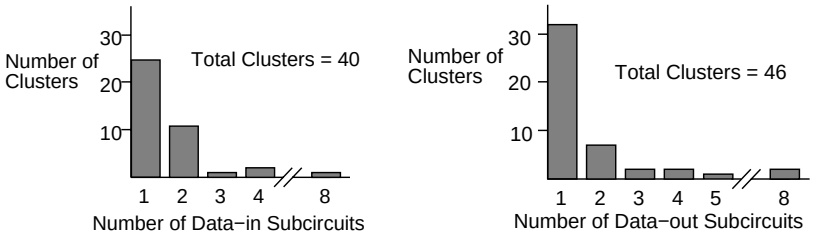


Fig. 5. Logic subcircuit statistics (per cluster).

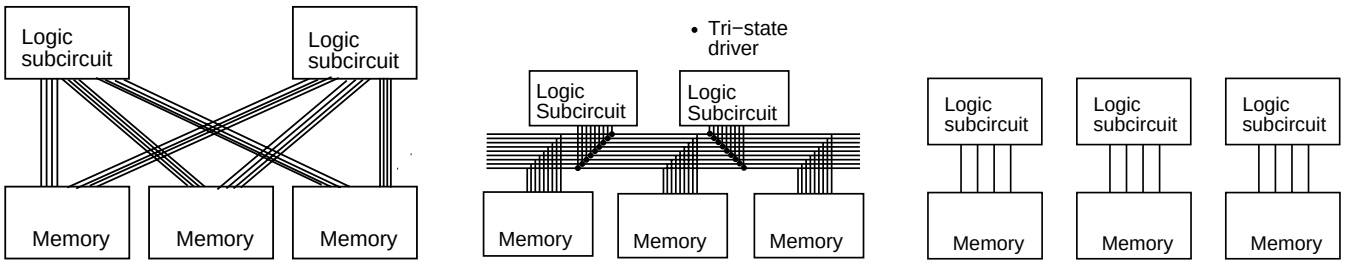
so often, we consider it separately. We refer to this pattern, which is shown in Figure 6(c), as *point-to-point pattern with no shuffling*. An example of this is in [20], in which the data-in ports of two memories are driven by separate serial-to-parallel converters, while the data-out port of each memory drives separate inputs of a large multiplier. Another example can be found in [19].

We analyzed each of the data-in and data-out networks in each cluster in the 31 circuits for which we had block diagrams, and classified each into one of the three categories described above. We found that, for the connection between logic and the data-in ports of the memories, all three patterns were approximately equally common. For the connection between the logic and the data-out ports of the memories, the first two patterns were approximately equally common, while the third pattern was rare.

III. Circuit Generation

A circuit generator was developed using the data from Section II. The generator stochastically generates realistic circuits as follows:

1. A memory configuration is chosen stochastically. The data in Figure 4(b) is used to guide the selection of the number of clusters, and the data in Figure 4(a) is used to guide the selection of the number of memories within each cluster. The width and depth of each



a) point-to-point b) shared-connection c) no shuffling

Fig. 6. Common Interconnect Patterns.

memory is then chosen using the data in Figure 2.

2. Interconnect patterns for the data-out and data-in networks within each cluster are chosen from the patterns in Figure 6. If a multiplexor is required, it is constructed using lookup tables (multiplexors are used rather than tri-state drivers since tri-state drivers are not plentiful in current FPGAs).
3. The logic subcircuits are chosen from a collection of 38 logic circuits obtained from the Microelectronics Center of North Carolina (MCNC) [27]. These circuits are all combinational, and contain between 24 and 184 five-input lookup tables. If the chosen circuits do not have enough inputs or outputs to source or sink the memory inputs or outputs, the subcircuits are replicated. Note that an alternative to randomly choosing pre-constructed circuits would be to use an existing synthetic circuit generator (such as the one in [6]).
4. The memories and logic subcircuits are glued together, and the entire circuit is written in either VHDL or BLIF formats.

The tool can generate a circuit in less than a second on a modern workstation. It has been successfully used in several architectural studies involving FPGAs with on-chip memory. Details can be found in [28], [29]. These studies would not have been possible without such a circuit generator.

IV. Concluding Remarks

In this paper, we have presented a detailed structural analysis of circuits with memory, and described a circuit generator that stochastically generates such circuits. The analysis focused on the number, sizes, shapes, and other key characteristics of memories within circuits, and also the way in which the memories are connected to logic. Statistics gathered during the analysis were then used to calibrate the circuit generator.

Acknowledgments

Financial support was provided by the British Columbia Advanced Systems Institute and the Natural Sciences and Engineering Research Council of Canada.

References

[1] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical Journal*, vol. 46, pp. 291–307, Feb 1970.

[2] J. Varghesse, M. Butts, and J. Batcheller, "An efficient logic emulation system," *IEEE Transactions on VLSI Systems*, vol. 1, pp. 171–174, 1970.

[3] K. Iwama and K. Hino, "Random generation of test instances for logic optimizers," in *Proceedings ACM/IEEE Design Automation Conference*, pp. 430–434, June 1994.

[4] J. Darnauer and W. W. Dai, "A method for generating random circuits and its application to routability measurement," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 66–72, Feb. 1996.

[5] M. Hutton, J. Grossman, J. Rose, and D. Corneil, "Characterization and parameterized random generation of digital circuits," in *Proceedings of ACM/IEEE Design Automation Conference*, pp. 94–99, June 1996.

[6] M. Hutton, J. Rose, and D. Corneil, "Generation of synthetic sequential benchmark circuits," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 149–155, Feb. 1997.

[7] D. A. Luthi, A. Mogre, N. Ben-Efraim, and A. Gupta, "A single-chip concatenated FEC decoder," in *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference*, pp. 13.2.1–13.2.4, May 1995.

[8] M. Toyokura, M. Saishi, S. Kurohmaru, K. Yamauchi, H. Imanishi, T. Ougi, A. Watabe, Y. Matsumoto, T. Morishige, H. Kodama, E. Miyagoshi, K. Okamoto, M. Gion, T. Minemaru, A. Ohtani, T. Araki, K. Aono, H. Takeno, T. Akiyama, and B. Wilson, "A video DSP with a macroblock-level-pipeline and and a SIMD type vector-pipeline architecture for MPEG2 CODEC," in *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pp. 74–75, Feb. 1994.

[9] S. Ti and C. Stearns, "A 200 MFlop CMOS transformation processor," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 6.1.1–6.1.4, May 1992.

[10] C. Benz, M. Gowan, and K. Springer, "An error-correcting encoder and decoder for a 1 Gbit/s fiber optic link," in *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, pp. 7.1.1–7.1.4, May 1991.

[11] P. Tong, "A 40-Mhz encoder-decoder chip generated by a reed-solomon code compiler," in *Proceedings of the IEEE 1990 Custom Integrated Circuits Conference*, pp. 13.5.1–13.5.4, May 1990.

[12] R. DeMara and D. Moldovan, "The SNAP-1 parallel AI prototype," in *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pp. 2–11, May 1991.

[13] A. Curiger, H. Bonnenberg, R. Zimmerman, N. Felber, H. Kaeslin, and W. Fichtner, "Vinci: VLSI implementation of the new secret-key block cipher idea," in *Proceedings of the IEEE 1993 Custom Integrated Circuits Conference*, pp. 15.5.1–15.5.4, May 1993.

[14] K. Fisher, P. Bednarz, J. Kouloheris, B. Fowler, J. Cioffi, and A. ElGamal, "A 54 Mhz BiCMOS digital equalizer for magnetic disk drives," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 19.3.1–19.3.4, May 1992.

[15] S. J. E. Wilton and Z. G. Vranesic, "Architectural Support for Block Transfers in a Shared Memory Multiprocessor," in *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, pp. 51–54, Dec. 1993.

[16] D. C. Chen and J. M. Rabaey, "A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1895–1904, December 1992.

[17] L. K. Tan and H. Samueli, "A 200-Mhz quadrature digital synthesizer/mixer in 0.8um CMOS," in *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, pp. 4.4.1–4.4.4, May 1994.

[18] I. Agi, P. J. Hurst, and K. W. Current, "A 450 MOPS image backprojector and histogrammer," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 6.2.1–6.2.4, May 1992.

[19] M. Matsui, H. Hara, K. Seta, Y. Uetani, L.-S. Kim, T. Nagamatsu, T. Shimazawa, S. Mita, G. Otomo, T. Oto, Y. Watanabe, F. Sano, A. Chiba, K. Matsuda, and T. Sakurai, "200 MHz video compression macrocells using low-swing differential logic," in *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pp. 76–77, Feb. 1994.

[20] S. Molloy, B. Schoner, A. Madisetti, and R. Jain, "An 80k-transistor configurable 25MPixels/s video-compression processor unit," in *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pp. 78–79, Feb. 1994.

[21] T. Karema, T. Husu, T. Saramaki, and H. Tenhunen, "A filter processor for interpolation and decimation," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 19.2.1–19.2.4, May 1992.

[22] M. Kuczynski, W. Lao, A. Dong, B. Wong, H. Nicholas, B. Itri, and H. Samueli, "A 1Mb/s digital subscriber line transceiver signal processor," in *Proceedings of the 1993 IEEE International Solid-State Circuits Conference*, pp. 26–27, Feb. 1993.

[23] N. Sollenberger, "An experimental TDMA modulation/demodulation CMOS VLSI chip-set," in *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, pp. 7.5.1–7.5.4, May 1991.

[24] E. Vanzieleghem, L. Dartios, J. Wenin, A. Vanwelsenaers, and D. Rabaey, "A single-chip GSM vocoder," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 10.3.1–10.3.4, May 1992.

[25] Y. Kondo, Y. Koshiba, Y. Arima, M. Murasaki, T. Yamada, H. Amishiro, H. Shinohara, and H. Mori, "A 1.2GLOPS neural network chip exhibiting fast convergence," in *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pp. 218–219, Feb. 1994.

[26] K. Ueda, T. Sugimura, M. Okamoto, S. Marui, T. Ishikawa, and M. Sakakihara, "A 16b lower-power-consumption digital signal processor," in *Proceedings of the 1993 IEEE International Solid-State Circuits Conference*, pp. 28–29, Feb. 1993.

[27] S. Yang, "Logic synthesis and optimization benchmarks," tech. rep., Microelectronics Center of North Carolina, 1991.

[28] S. J. E. Wilton, J. Rose, and Z. G. Vranesic, "Memory/logic interconnect flexibility in FPGAs with large embedded memory arrays," in *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, pp. 144–147, May 1996.

[29] S. J. E. Wilton, J. Rose, and Z. G. Vranesic, "Memory-to-memory connection structures in FPGAs with embedded memory arrays," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 10–16, February 1997.