Memory/Logic Interconnect Flexibility in FPGAs with Large Embedded Memory Arrays

Steven J.E. Wilton, Jonathan Rose, and Zvonko G. Vranesic
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada, M5S 1A4
{wilton,jayar,zvonko}@eecg.toronto.edu *

Abstract

As the capacities of field-programmable gate arrays (FPGAs) grow, it becomes desirable to create FPGAs with embedded memory arrays. This paper examines the flexibility of the interconnect structure that joins memory and logic. For architectures with only a few memory arrays, we find that both the routability and the delay of circuits are insensitive to the memory/logic interconnect flexibility, which implies that this interconnection can be made very inflexible. This is in contrast to the logic connection block flexibility, which has been shown to require high flexibility [1]. For architectures with more arrays, the memory/logic interconnect flexibility requirements increase and approach those of logic interconnect.

1 Introduction

As FPGAs grow to encompass entire systems, it is becoming clear that an efficient way to implement large blocks of memory is needed. FPGAs with embedded memory blocks are now available from several vendors [2, 3], in contrast to the fine-grain LUT RAM available earlier [4, 5]. Both types of memory provide higher integration, faster memory access and reduce the demand for FPGA I/O pins compared to off-chip memory implementations.

In our previous work [6, 7], we examined FPGA memory architectures in isolation and showed how to provide the ability to implement many different numbers and shapes of user memories. This paper explores the structure of the interconnection between memory and logic. We are interested in finding the minimum memory/logic interconnect flexibility (the cost of the interconnect) that provides both routability and high speed. To do this we employ an experimental approach in which benchmark circuits are implemented on many different interconnect architectures. The following section describes our base architectural assumptions. Section 3 describes how we created enough circuits to test the architecture, and the implementation procedure. The subsequent sections give results and conclusions.

*This work was supported by the Natural Sciences and Engineering Research Council of Canada, MICRONET, and the Walter C. Sumner Foundation.

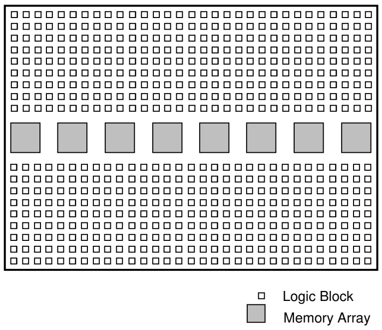


Figure 1: FPGA with Embedded Memory

2 Architectural Framework

We assume that an FPGA with embedded memory consists of distinct logic and memory resources, as illustrated in Figure 1. The memory resources consist of a set of identical arrays that can be combined to implement the required user memory configuration, similar to [6]. The number of bits in each array is fixed, but the aspect ratio can be configured by the user. In the results presented in Section 4, we have assumed that each array has two kilobits, and has a configurable data width of 1, 2, 4, or 8 (similar to [2] and [7]). Unlike [6], here we assume that each memory array has separate input and output data ports (as well as an address port).

The logic resources of the FPGA are assumed to consist of five-input lookup tables, interconnected using symmetric horizontal and vertical channels like those of [1] and [4]. At the intersection of each horizontal and vertical channel is a switch block. The flexibility of each switch block, F_s, is defined as the total number of possible connections offered to each incoming wire [1]. In this paper, we assume $F_s = 3$, which is consistent with previous work. All segments are of length 1; that is, segments only connect neighbouring switch blocks. Each pin of each lookup table can be connected to two channels; within each channel, each pin can be connected to

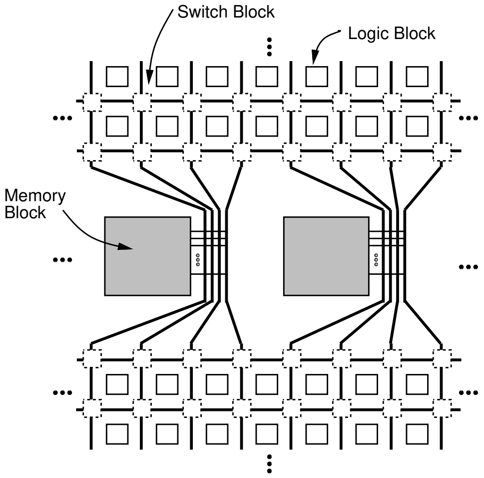


Figure 2: Memory/Logic Routing Structures

F_c tracks. In Section 4, we vary F_c between W and $\frac{W}{2}$, where W is the number of routing tracks per channel.

The memory arrays are assumed to be positioned in a single row in the middle of the chip, as shown in Figure 1, with logic blocks above and below the memory arrays. This is similar to the Altera FLEX10K architecture [2] and the Actel 3200DX architecture [3].

Figure 2 shows the assumed interconnect structure between the logic and memory portions of the FPGA. In the figure, each solid line is a channel of W parallel routing tracks. The vertical tracks in the top half of the logic are connected to those in the bottom half, and the pins on each memory array are connected to one or more of these vertical tracks. Note that in Figure 2, the vertical tracks are shown to "bend" around the memory blocks; in multi-metal layer implementations, the vertical tracks might simply cross on top of the arrays.

We define the flexibility of the memory/logic interconnect, F_{lm}, as the number of vertical tracks to which each memory pin can connect. If there are r logic blocks per memory block in the horizontal dimension, the maximum possible F_{lm} is rW (in Figure 2, $r = 4$). In Section 4, we will vary F_{lm} from 1 to its maximum value and examine the effects on routability and delay.

3 Architecture Evaluation Method

This section describes how we investigate the effect of varying the amount of flexibility in the memory/logic interconnect structure, F_{lm}. The general approach is to map, place and route benchmark circuits on FPGAs with different architectural parameters. We measure the effect on routing demand (the number of tracks needed to route, W) and the speed of the resulting circuits. The following subsections describe the development of appropriate benchmark circuits, and the implementation procedure.

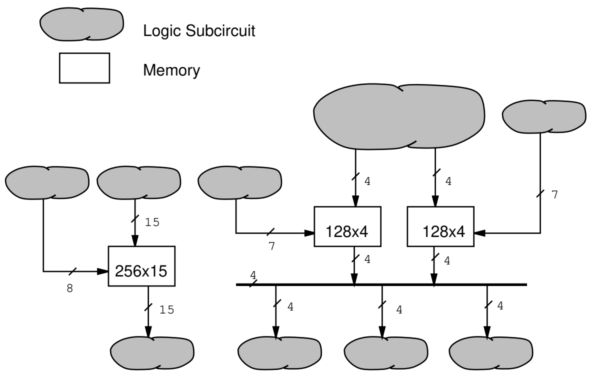


Figure 3: Example Generated Circuit

3.1 Benchmark Circuit Generation

It is difficult to apply the traditional method of placing and routing 10 to 20 benchmark circuits [1, 8] in this analysis because circuits typically have only a few memories each, and so hundreds of such examples may be required to properly exercise the memory-logic connection architecture. Because it isn't feasible to gather that many benchmark circuits, our approach is to study the types of memory configurations found in systems, and then to develop a stochastic memory configuration generator based on that study.

A memory configuration is the set of all memories required by a circuit, and consists of the number of memory *clusters* (clusters are groups of memories which share data input or data output subcircuits), the number of memories within each cluster, and the width and depth of each memory.

The generator chooses the memory configuration parameters independently from probability distributions based on statistics gathered from a set of 171 custom ASIC designs. (Although we were able to obtain data on the memories in these circuits, the detailed circuits were not available).

Next, the logic sub-circuits needed to supply address and data as well as consume the data are generated and attached to the memories. The memory/logic interconnect pattern is chosen from a set of commonly occurring patterns in our example ASIC designs using measured distributions. The sub-circuits are randomly chosen from a collection of 38 MCNC circuits, which have been optimized using SIS [9] and technology-mapped to 5-input lookup tables using FlowMap [10]. The actual construction of the circuits is non-trivial and beyond the scope of this paper. Full details of the memory circuit study and construction of the generator are described in [11].

Figure 3 illustrates an example generated circuit. It has three memories, two of which receive data from a single logic source and drive a common data bus. Interconnections between the various logic sub-circuits and external input/output pins are not shown in this figure.

The circuits are generated without accounting for the

Arch	5-LUTs Used		Arrays Used		Total Terminals		Memory Terminals		r		Aspect Ratio	
	Avg	St.Dev	Avg	St.Dev	Avg	St.Dev	Avg	St.Dev	Avg	St.Dev	Avg	St.Dev
2	382	192	1.63	0.48	2094	1033	28.7	11.7	10.4	3.43	1.00	0
4	502	307	2.61	1.13	2761	1649	47.4	23.3	5.98	2.28	1.01	0.05
8	574	340	3.93	2.27	3164	1838	69.3	41.6	3.50	0.85	1.53	0.96
16	638	360	5.46	3.97	3543	1968	97.8	73.8	3.05	0.23	4.64	3.44

Table 1: Circuit and Implementation Statistics

number of memory bits or number of independent arrays in the FPGA architecture. Once generated, the circuits that do not fit in the target architecture (in terms of total number of bits or number of independent arrays) are discarded. In the next section, we present results for architectures with 2, 4, 8, and 16 memory arrays; for each architecture we generate circuits until we have 200 circuits that fit. Table 1 shows statistics from the 200 circuits generated for each architecture (r and Aspect Ratio will be discussed in Section 3.2).

This stochastic generation approach was also used in [6] which studied only the memory portion of an embedded-memory FPGA.

3.2 Circuit Implementation Procedure

Each benchmark circuit is "implemented" in each FPGA using the following procedure: First, each memory in the memory configuration must be implemented using one or more of the physical memory arrays. For example, a 4Kx2 user memory can be implemented using four 2-kilobit arrays each configured as a 2Kx1 memory with appropriate decoding. We call this process the logical-to-physical mapping. Because there can be many ways to perform this mapping, it is an optimization problem. We use an algorithm described in [11].

Next, the mapped circuits are then placed on an appropriately-sized FPGA using a simulated annealing-based placement program, and the detailed routing is performed using a multi-pass maze router (both tools are described in [11]). The routing is repeated for different values of W (the number of tracks per channel) to determine the minimum W that gives a 100% routable solution. The router considers all input pins of a lookup-table equivalent, as well as all address pins of a memory array. Data pins are also considered equivalent with the constraint that a pin assignment for a specific bit in the data-out port fixes the corresponding assignment in the data-in port (and vice versa).

The size of the FPGA used in the place and route step depends on the number of lookup tables in the circuit to be implemented, as well as the number of memory arrays to be included. For a given number of arrays, we place the required number of logic blocks above and below the memory row, creating a roughly square chip. This will result in different values of r (the number of logic blocks per memory block in the horizontal dimension) for different circuits. If, however, the resulting r is less than 3, we force r to be 3, resulting in a non-square aspect ratio. For some circuits, the number of input/outputs will determine the chip area; for these circuits, a square array with the required number of input/output blocks will be used. Table 1 shows the average values of r and the average aspect ratio (ratio of horizontal logic blocks to vertical logic blocks) for circuits implemented on the four architectures.

Finally, a timing analyzer is used to measure the critical path through the circuit. The timing analyzer finds each net delay by constructing an RC-tree. The memory access time is measured from a modified version of CACTI, a detailed cache access time model [12]. The CLB delay is assumed to be constant. A 0.5um CMOS process was assumed throughout.

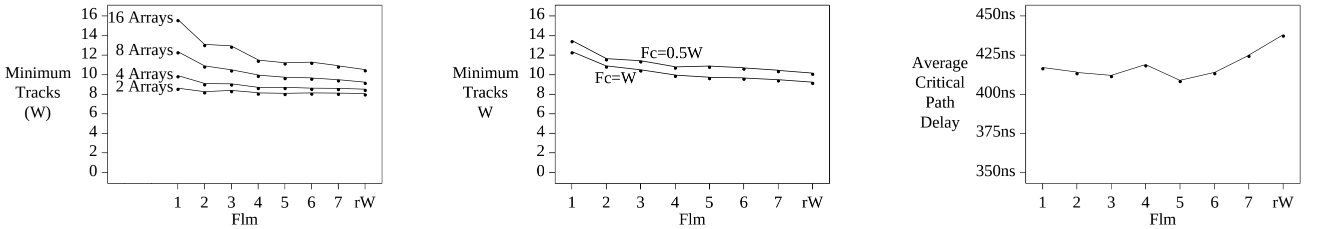
4 Experimental Results

4.1 Effect of F_{lm} on Routability

Figure 4-a shows the effect of changing F_{lm} on the minimum track count required for 100% routability (averaged over 200 stochastically generated circuits). The results are shown for architectures with 2, 4, 8, and 16 two-kilobit blocks. The right-most point on each line corresponds to architectures with $F_{lm} = rW$; as discussed in Section 2, this is the maximum possible F_{lm}. As the graph shows, for architectures with fewer than 8 arrays, the measured minimum W is not sensitive to F_{lm}. This is contrast to [1] which shows that circuit routability for logic is severely hampered by low values of the connection block flexibility, F_c.

As the number of memory arrays increases above 4, the number of tracks needed for complete routing *does* increase significantly for low F_{lm} ($F_{lm} = 4$ is sufficient). There are several reasons for this. First, the circuits generated for these architectures tend to have larger memory requirements, and thus, use more memory pins (this is shown in Table 1). For small values of F_{lm}, the nets connected to each array have fewer routing options than nets connected to only logic elements. The more of these nets there are, the more sensitive the overall circuit routability will be to F_{lm}.

A second effect is from nets that connect to more than one memory array. Memory arrays are connected together when they are combined to form larger user memories, or when independent user memories share a common data bus. The more arrays in a circuit, the more of these memory-to-memory nets there are. The number of routing paths for these nets is very dependent on F_{lm}.



a) $F_c = W$ b) 8 arrays, varied F_c c) delay results for 8-array architecture

Figure 4: Effects of F_{lm} on routability and delay

A final reason may be due to the way in which the memory connections are "spread" among the logic blocks, as determined by the parameter r. For architectures with only a few arrays, the value of r is significantly larger than for architectures with many arrays, as shown in Table 1, column 10. The higher r is, the more "spread out" the memory pins are. It is possible that a concentration of the low-flexibility memory pins in a single channel decreases routability.

4.2 Effect of F_c on choice of F_{lm}

Figure 4-a assumed that the connection block flexibility, $F_c = W$; Figure 4-b shows how the results (for 8 arrays) affected by F_c. Although the track requirements go up as F_c is decreased, the dependence on F_{lm} remains unchanged.

4.3 Circuit Delay

Although the main motivation behind using a small value of F_{lm} was to reduce the area required for the memory/logic interconnect block, removing switches generally increases the speed of a circuit implemented on an FPGA. In this case, however, the memory/logic interconnect block does not represent a major portion of the critical path delay, so we would expect the speed to be roughly independent of F_{lm}. Figure 4-c shows the average critical path delay of the 8 array architecture. As the graph shows, for low flexibilities, the delay actually drops slightly as F_{lm} increases; this is because a low F_{lm} often results in circuitous routes for some nets.

5 Conclusions

In this paper, we have investigated the effects of the memory/logic connection block flexibility on the overall routability and speed of a circuit implemented on an FPGA with embedded memory arrays. We have found that the routability of circuits with memory is not strongly affected by a low memory/logic interconnect flexibility, especially for architectures with fewer than 8 embedded arrays. For such architectures, the logic routing resources are flexible enough to compensate for the low flexibility in the memory/logic interconnect. This becomes less true as the number of arrays (and hence the number of connections to memory) is increased.

The delay of such circuits was also shown to be roughly independent of the flexibility of the memory/logic interconnect. This is primarily because this interconnect block represents only a small portion of the critical path of a circuit.

References

[1] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 277–282, March 1991.

[2] Altera Corporation, *Datasheet: Flex 10K Embedded Programmable Logic Family*, 1995.

[3] Actel Corporation, *Datasheet: 3200DX Field-Programmable Gate Arrays*, 1995.

[4] Xilinx, Inc., *The Programmable Logic Data Book*, 1994.

[5] AT&T Microelectronics, *Product Brief: AT&T Optimized Reconfigurable Cell Array (ORCA) Series Field-Programmable Gate Arrays (FPGAs)*, April 1993.

[6] S. J. E. Wilton, J. Rose, and Z. G. Vranesic, "Architecture of centralized field-configurable memory," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 97–103, 1995.

[7] T. Ngai, J. Rose, and S. J. E. Wilton, "An SRAM-Programmable field-configurable memory," in *IEEE Custom Integrated Circuits Conference*, pp. 499–502, May 1995.

[8] J. L. Kouloheris and A. E. Gamal, "PLA-based FPGA area versus cell granularity," in *1992 Custom Integrated Circuits Conference*, pp. 4.3.1–4.3.4, 1992.

[9] E. Sentovich, "SIS: A system for sequential circuit analysis," tech. rep., Electronics Research Laboratory, University of California, Berkeley, May 1992.

[10] J. Cong and Y. Ding, "An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 48–53, November 1992.

[11] S. J. E. Wilton, *Architecture of Field-Configurable Memory*. PhD thesis, University of Toronto, in preparation.

[12] S. J. E. Wilton and N. P. Jouppi, "CACTI: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 677–688, May 1996.