

Implementing Logic in FPGA Embedded Memory Arrays: Architectural Implications

Steven J.E. Wilton
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
steve@ee.ubc.ca *

Abstract

It has become clear that embedded memory arrays will be essential in future FPGAs. These arrays were originally intended to implement storage, but recent work has shown that they can also be used to implement logic very efficiently. In this paper, we explore how the depth, width, and flexibility of the embedded arrays affect their ability to implement logic. It is shown that each array should contain between 512 and 2048 bits, and should have a word width that can be configured as 1, 2, 4, or 8.

1 Introduction

On-chip storage has become an essential component of high-density FPGAs. The large systems that will be implemented on these FPGAs often require storage; implementing this storage on-chip results in faster clock frequencies and lower system costs. Two implementations of on-chip memory in FPGAs have emerged: fine-grained and coarse-grained. In FPGAs employing fine-grained on-chip storage, such as the Xilinx 4000 FPGAs, each lookup table can be configured as a small RAM, and these RAMs can be combined to implement larger user memories [1]. FPGAs employing the coarse-grained approach, on the other hand, contain large embedded arrays which are used to implement the storage parts of circuits. Examples of such devices are the Altera 10K devices [2], the Actel 3200DX and SPGA parts [3, 4], and the Lattice ispLSI 6192 FPGAs [5].

The coarse-grained approach results in significantly denser memory implementations, since the per-bit overhead is much smaller [6]. Unfortunately, it also requires the FPGA vendor to partition the chip into memory and logic regions when the FPGA is designed. Since circuits have widely-varying memory requirements, this

“average-case” partitioning may result in poor device utilizations for logic-intensive or memory-intensive circuits. In particular, if a circuit does not use all the available memory arrays to implement storage, the chip area devoted to the unused arrays is wasted.

This chip area need not be wasted, however, if the unused memory arrays are used to implement logic. Configuring the arrays as ROMs results in large multi-output lookup-tables that can very efficiently implement some logic circuits. In [7], a new tool, SMAP, was presented that packs as much circuit information as possible into the available memory arrays, and maps the rest of the circuit into four-input lookup-tables. It was shown that this technique results in extremely dense logic implementations for many circuits; not only is the chip area of the unused arrays not wasted, but it is used *more efficiently than if the arrays were replaced by logic blocks*. Thus, even customers that do not require storage can benefit from embedded memory arrays.

The effectiveness of this mapping technique, however, is very dependent on the architecture of the embedded memory arrays. If the arrays are too small, the amount of logic that can be packed into each will be small, while if the arrays are too large, much of each array will be unused. Previous studies have focused on the architecture of these memory resources when implementing storage [8, 9, 10]. Since they are so effective at implementing logic, however, it is important that the design of the embedded memory arrays also consider this. In this paper, we explore the effects of array depth, width, and flexibility of memory arrays when they are used to implement logic.

The architectural space explored in this paper is described in Section 2. Section 3 describes the experimental methodology and reviews the SMAP algorithm. Finally, Section 4 presents experimental results.

2 Embedded Array Architectures

Table 1 summarizes the parameters that define the FPGA embedded memory array architecture, along with

*This work was supported by Cadence Design Systems, the Natural Sciences and Engineering Research Council of Canada, and UBC's Centre for Integrated Computer Systems Research.

Parameter	Meaning	Commercial Devices				Range in this paper
		Altera 10K	Actel SPGA	Actel 3200DX	Lattice isp6192	
N	Number of Arrays	3-16	2-32	8-16	1	1-16
B	Bits per Array	2048	2048	256	4608	256-16384
w_{eff}	Allowable Data Widths	{1,2,4,8}	{8}	{4,8}	{9,18}	several

Table 1: Architectural Parameters

values of these parameters for various commercial devices. The number of embedded memory arrays is denoted by N , the number of bits in each array is denoted by B , and the set of allowable data widths is denoted by w_{eff} . For a fixed B , a wider memory implies fewer memory words in each array. In the Altera FLEX10K, $B = 2048$ bits, and $w_{\text{eff}} = \{1, 2, 4, 8\}$, meaning each array can be configured to be one of 2048×1 , 1024×2 , 512×4 , or 256×8 . In Section 4, we will seek the optimum values for B and w_{eff} for several values of N .

3 Methodology

To compare memory array architectures, we employed an experimental methodology in which we varied B , N , and w_{eff} and mapped a set of 30 benchmark circuits to each architecture. Each circuit contained between 527 and 6598 4-LUTs. Seventeen of the circuits were sequential. The combinational circuits and 9 of the sequential circuits were obtained from the Microelectronics Corporation of North Carolina (MCNC) benchmark suite, while the remaining sequential circuits were obtained from the University of Toronto and were the result of synthesis from VHDL and Verilog. All circuits were optimized using SIS [11] and mapped to four-input lookup-tables using Flowmap and Flowpack [12]. The SMAP algorithm was then used to pack as much circuit information as possible into the available memory arrays. The number of nodes that can be packed to the available arrays is used as a metric to compare memory array architectures.

The results in this paper depend heavily on the SMAP algorithm. The rest of this section briefly reviews SMAP; for more details, see [7].

The SMAP algorithm is based on Flowpack, a post-processing step of Flowmap [12]. Given a seed node, the algorithm finds the *maximum-volume k -feasible cut*, where k is the number of address inputs to each memory array. A k -feasible cut is a set of no more than k nodes in the fanin-network of the seed such that the seed can be expressed entirely as a function of the k nodes; the maximum-volume k -feasible cut is the cut which contains the most nodes between the cut and the seed. The nodes that make up the cut become the memory array inputs. Figure 1(a) shows an example circuit along with the the maximum 8-feasible cut for seed node A.

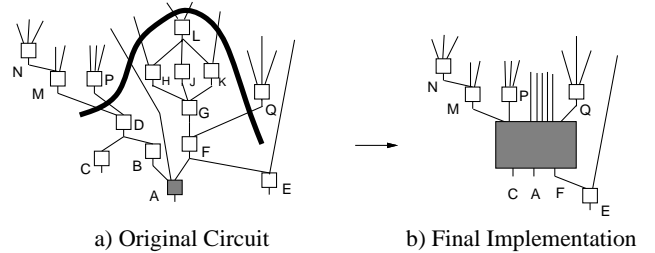


Figure 1: Example Mapping to a 8-Input, 3-Output Memory Block

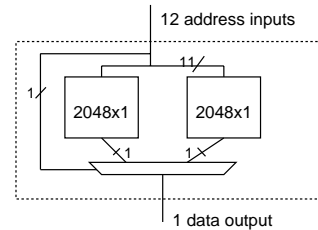


Figure 2: Combining Two Arrays

Given a seed node and a cut, SMAP then selects which nodes will become the memory array outputs. Any node that can be expressed as a function of the cut nodes is a potential memory array output. The selection of the outputs is an optimization problem, since different combination of outputs will lead to different numbers of nodes that can be packed into the arrays. In [7], a heuristic is presented; the outputs with the largest number of nodes in their *maximum fanout-free cone* (maximum cone rooted at the potential output such that no node in the cone drives a node not in the cone) are selected. As shown in [7], those nodes in the maximum fanout-free cones of the outputs can be packed into the array. All other nodes in the network must be implemented using logic blocks. In Figure 1(a), nodes C, A, and F are the selected outputs; Figure 1(b) shows the resulting circuit implementation.

Since the selection of the seed node is so important, we repeat the algorithm for each seed node, and choose the best results.

If there is more than one array available, we group all available arrays into a single *super-array*. For example, two 2Kbit arrays with $w_{\text{eff}} = \{1, 2, 4, 8\}$ can be combined

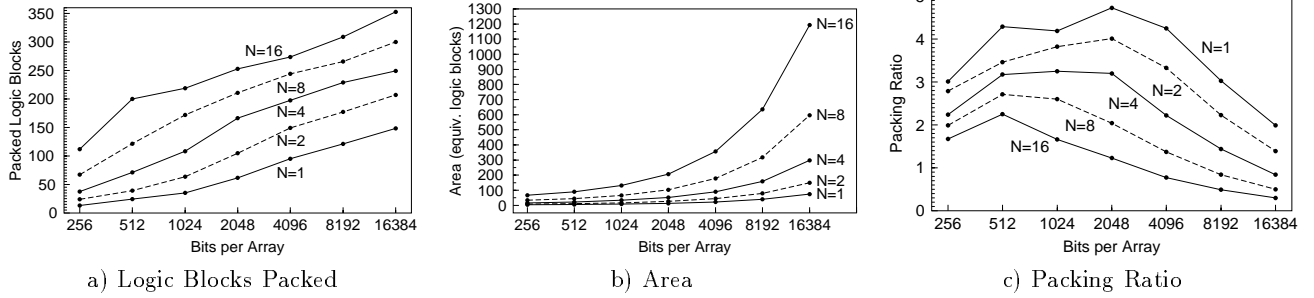


Figure 3: Effects of Memory Array Size

into a super-array of 4Kbits with $w_{\text{eff}} = \{1, 2, 4, 8, 16\}$. The SMAP algorithm is then used to pack logic into this larger super-array. As an example, Figure 2 shows how two 2Kbit arrays can be put in their “x1” mode and combined to implement a 4096x1 bit memory. SMAP can then treat this as a 12-input, 1-output array. A multiplexer is required to multiplex between the two arrays; this multiplexer can be implemented using the logic resources of the FPGA.

4 Results

4.1 Array Size

Figure 3 shows how the effectiveness of each memory block in implementing logic depends on the array size, B . Figure 3(a) shows the number of logic blocks that can be packed into the arrays (averaged over our 30 benchmark circuits) vs. B for several values of N . Figure 3(b) shows the estimated chip area of each memory block as a function of B , again for various values of N . The area estimates were obtained from a detailed area model [13] and are expressed in *logic block equivalents* (LBE). One LBE is the area required to implement one logic block. For $N > 1$, these area estimates include the area due to the multiplexers needed to combine the arrays into a single super-array (these multiplexers are assumed to be implemented using the FPGA logic resources).

Figure 3(c) shows the *packing density* as a function of B for several values of N . The packing density is defined as the ratio of the number of logic blocks that can be packed into the available memory arrays over the area required to implement the memory arrays (in LBEs). A packing density of 1 means that the density of logic implemented in memory arrays is equal to that if the logic was implemented in logic blocks. A packing density greater than 1 means that the density of logic implemented in memory arrays is *greater* than that if logic blocks were used. As Figure 3(c) shows, the packing density is greater than 1 for all but the largest memory arrays. The highest packing density occurs for $B = 2048$ if there are 2 or fewer arrays; if more arrays are available, the optimum value of B is slightly smaller.

4.2 Array Width

As described in Section 2, most FPGA memory architectures allow the user to select the data width of each array from a predetermined set, w_{eff} . Figures 4 and 5 show how the effectiveness of the memory arrays in implementing logic depends on w_{eff} . Figure 4 shows the number of logic blocks packed, the area requirements, and the packing density as a function of the maximum allowable width (highest value in w_{eff}) for each memory array. In all cases, it is assumed that each array contains 2048 bits, and that the user can configure the memory width to be any power-of-two between 1 and the maximum allowable width. Thus, for the left-most point on the graph, only a 2048x1 configuration is available, while for the right-most point, each array can be configured to any configuration between 2048x1 and 64x32. As the figures show, the highest packing density occurs if the maximum available width is 8. Arrays with a larger width require more area (more sense amplifiers and routing) while arrays with a smaller maximum width can not implement as much logic. The optimum choice for the maximum allowable width is independent of N .

Figure 5 shows the results as a function of the minimum allowable width, assuming that $B = 2048$ and that the maximum allowable width is 8. Thus, for the left-most point in the graphs, the user can configure each array to be one of 2048x1, 1024x2, 512x4, or 256x8, while in the architecture corresponding to the right-most point, only the 256x8 configuration is available. As shown in Figure 5(a), more circuit information can be packed into the more flexible arrays. The area results in Figure 5 appear counterintuitive; according to this figure, the more flexible an architecture is, the less area it requires. This is because the area measurements in this graph include the area required to implement the multiplexers needed to combine the N arrays into a super-array. The multiplexers are larger if an array is used in the “x8” configuration than in the “x1” configuration, since more bits must be multiplexed; thus, an architecture in which only a “x8” configuration is allowed ($w_{\text{eff}} = \{8\}$) will require larger multiplexers, on average. This effect is more pro-

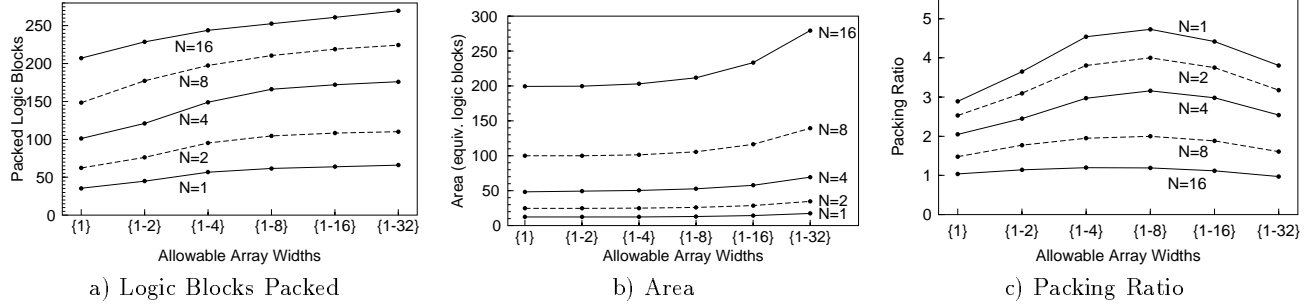


Figure 4: Effects of Maximum Allowable Width

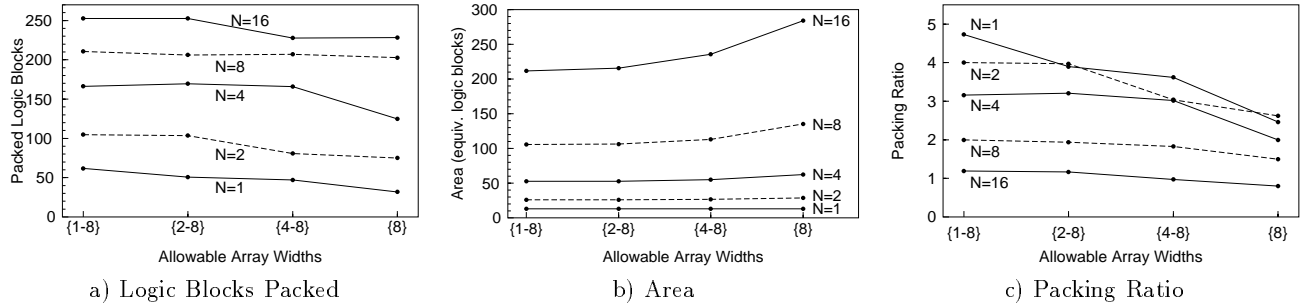


Figure 5: Effects of Minimum Allowable Width

nounced for larger values of N ; this is because the more arrays there are to combine, the more multiplexers are required. The packing ratio is shown in Figure 5(c); in all cases, the more flexible architecture is the best choice.

Overall, the best choice for w_{eff} is $\{1, 2, 4, 8\}$.

5 Conclusions

Although embedded arrays in FPGAs were developed in order to implement on-chip storage, it is clear that these arrays can also be configured as ROMs and used to implement logic. In this paper, we have investigated how the depth, width, and flexibility of the embedded arrays affect the ability of the array to implement logic. Overall, we found that each memory array should contain between 512 and 2048 bits, and should have a word width that can be configured to be 1, 2, 4, or 8.

In [13], similar conclusions were obtained assuming the arrays are used to implement storage. This is an encouraging result, since it means a FPGA vendors can embed arrays in their FPGAs, and these arrays can be efficiently used for either storage or logic, depending on the needs of the application circuit.

References

- [1] Xilinx, Inc., *XC4000 Series (E/L/EX/XL) Field Programmable Gate Arrays v1.04*, September 1996.
- [2] Altera Corporation, *Databook*, June 1996.
- [3] Actel Corporation, *Datasheet: 3200DX Field-Programmable Gate Arrays*, 1995.
- [4] Actel Corporation, *Actel's Reprogrammable SPGAs*, 1996.
- [5] Lattice Semiconductor Corporation, *Datasheet: ispLSI and pLSI 6192 High Density Programmable Logic with Dedicated Memory and Register/Counter Modules*, July 1996.
- [6] T. Ngai, J. Rose, and S. J. E. Wilton, "An SRAM-Programmable field-configurable memory," in *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference*, pp. 499–502, May 1995.
- [7] S. J. E. Wilton, "SMAP: heterogeneous technology mapping for for area reduction in FPGAs with embedded memory arrays," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, February 1998.
- [8] S. J. E. Wilton, J. Rose, and Z. G. Vranesic, "Architecture of centralized field-configurable memory," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 97–103, 1995.
- [9] 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.
- [10] 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, Feb. 1997.
- [11] E. Sentovich, "SIS: A system for sequential circuit analysis," Tech. Rep. UCB/ERL M92/41, Electronics Research Laboratory, University of California, Berkeley, May 1992.
- [12] J. Cong and Y. Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, pp. 1–12, January 1994.
- [13] S. J. E. Wilton, *Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*. PhD thesis, University of Toronto, 1997.