

The Architecture of Dual-Mode FPGA Embedded System Blocks

Ernie Lin and Steven J. E. Wilton

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada V6T 1Z4
{erniel, steview}@ece.ubc.ca

Abstract

Recently, it has been shown that unused on-chip memories can be valuable when they are used to implement logic. This paper explores how different memory architecture parameters affect its ability to implement logic in dual-mode FPGA Embedded System Blocks. It is shown that the optimum memory architecture has a depth of 32 or 64 words, and that each word should contain 16 bits.

1 Introduction

Traditionally, applications of field-programmable gate arrays (FPGAs) have been limited to small glue-logic subcircuits. As the logic capacity and speed of FPGAs increase, however, FPGAs are becoming the technology of choice for implementing large systems. The requirements of larger systems differ from smaller systems, since larger systems typically require access to memory. As a result, most modern FPGAs now contain large embedded RAM arrays [1, 2]. These arrays provide dense implementations of storage, but require the FPGA vendor partition the chip into logic and storage regions when the FPGA is designed. Since circuits have widely varying requirements for memory resources, this "average case" partitioning may result in poor device utilization in logic-intensive applications. More specifically, if memory is not used completely or not at all, the chip area devoted to memory is wasted.

However, this area need not be wasted, if unused memory arrays are used to implement logic. Unused memories can be configured as large ROM multiple-input, multiple-output lookup tables. Wilton's SMAP [3] and Cong and Xu's EMBPack [4] are two existing algorithms for mapping logic to unused memory arrays.

These algorithms work well if there are only a small number of memory arrays, but as the number of arrays increase, the efficiency of the memories when implementing logic decreases [3]. To improve the logic density when there are a large number of arrays, the APEX20k architecture has enhanced memory arrays called Embedded System Blocks (ESB) [5]. Each ESB can be configured in one of two modes: it can act as a conventional memory, which can be used for storage or logic, or it can act as a programmable array logic (PAL) block for implementing functions as a sum of products. An algorithm, pMapster, which maps circuits to this dual-mode architecture, was presented in [6].

While the amount of logic that can be packed into each ESB depends on the quality of the CAD algorithm used to perform the packing [6], the architecture of the ESB can have a very dramatic effect on the achievable packing density. A memory that is too small will result in a small amount of logic being packed into it, while a memory that is too large will result in underutilization. In this paper, we seek to find the optimum width and depth for these ESBs.

2 Comparison to Previous Work

An FPGA with dual-mode embedded system blocks is essentially a type of hybrid FPGA as described in [7] in that it contains both lookup-tables (the logic resources) and product-term blocks (the ESBs configured in product-term mode). In [7], the optimum architecture for such an FPGA is sought. Our focus differs in three ways. First, in [7], it is assumed that there is approximately a 4:1 ratio of logic blocks to PLA blocks; in our architecture, this ratio is significantly larger. In addition, [7] considered PLA blocks which are dedicated PLA blocks; in our case, the optimum architecture for an ESB will be a compromise between the needs of the block used in conventional memory mode and the needs of the block used in product-term mode. Finally, in [7], the number of product-terms is a free variable; in our architecture, the number of product terms per output is fixed at two ([6] shows that this is a good choice). Thus, we would not expect the conclusions of [7] to hold for our architecture.

The architecture of PLA-based FPGAs have been studied elsewhere [8, 9]; in these studies, it was assumed that all logic resources are product-term blocks. In our case, only a small portion of the logic is implemented in product-term blocks.

The architecture of ESBs which can only be configured as a conventional memory was considered in [10]. That work found that a 512 to 2048 bit memory array works well when implementing logic. In this paper, we extend that work to consider architectures in which each ESB can be configured as either a conventional memory or a product-term block (a dual-mode ESB). As described above, we would expect the optimum architecture to be a compromise between the results in [10] and the results assuming the block is used in product-term mode. Thus, we first present results assuming product-term mode, and then present results for the dual-mode ESB.

Architectural Parameter	Meaning	Range Explored in this Paper	Typical Values (Altera APEX20k)
N	Number of memory arrays	{1, 2, 4, 8, 16}	12 - 52
d	Number of memory words (depth)	{16, 32, 64, 128, 256}	64
w	Number of bits per memory word (width)	{8, 16, 32, 64, 128}	32

Table 1: Memory architecture parameters

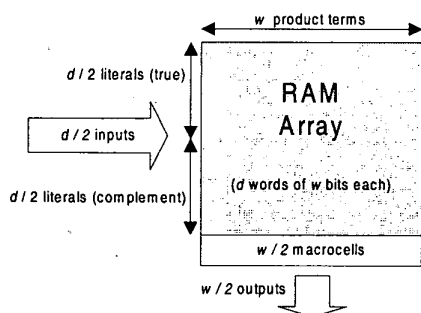


Figure 1: Relationship between memory architecture and product term architecture in the APEX20k

3 Architecture of Product Term Memory Arrays

The heart of each ESB is a memory array. The dimensions of the memory array affect the ability of the ESB to implement logic in either conventional memory mode or product-term mode. Conventional memory mode was considered in [10]; here we focus on product-term mode. In product-term mode, each column in the memory array implements one product-term; these product-terms are combined in the macrocells. Note that unlike conventional PLA blocks, there is exactly two product-terms per macrocell and one macrocell per output; [6] shows that this works well. As shown in Figure 1, an ESB with an array of depth d and width w can implement up to $w/2$ functions of two product-terms each, using at most $d/2$ inputs. Table 1 summarizes the parameters we investigate in this paper.

4 Experimental Methodology

To determine the optimum array width and depth, we used 20 benchmark circuits from the Microelectronics Corporation of North Carolina (MCNC) benchmark suite. Each circuit contains between 97 and 1878 4-input look-up-tables (4-LUTs). All circuits were optimized using SIS [11], and then mapped to 4-LUTs using Flowmap and Flowpack [12].

To evaluate the dual-mode architectures, we then packed logic from each optimized circuit into each ESB in two ways: (a) with the ESB configured as a conventional memory, using the SMAP algorithm, and (b) with the ESB configured in product-term mode, using the pMapster algorithm. For each ESB, we choose the mode which resulted in the best packing density. The logic nodes in the

circuit which could be packed into the ESBs were then implemented in the ESBs, while those that could not be packed were implemented in 4-LUTs.

A higher width and/or depth implies that more logic can be packed into each ESB. On the other hand, a larger ESB requires more chip area. To estimate the area required by each architecture under consideration, we used a detailed area model which contains terms for the memory cells, decoders, drivers, sense amplifiers, macrocells, and control logic.

pMapster Algorithm : Mapping to Product Terms

Since the results of this study depend heavily on the pMapster algorithm, we briefly review the algorithm here. For more details see [6]. The algorithm takes a network of nodes (a node corresponds to a 4-LUT) as input, and determines which nodes are suitable for implementation as product terms; these nodes are then removed from the network. The function implemented by the removed nodes is replaced by a product term memory programmed to implement the same function. This is shown in Figure 2.

Nodes are packed by repeatedly collapsing fanins into a *seed node* until the number of product terms, literals, or outputs in the target architecture is exceeded. For seed nodes with many fanins, the fanin that produces the fewest new product terms while still meeting the constraints given above is collapsed. Fanins that fanout to a node other than the seed are collapsed as well; in this case, the fanin becomes an output of the target memory array.

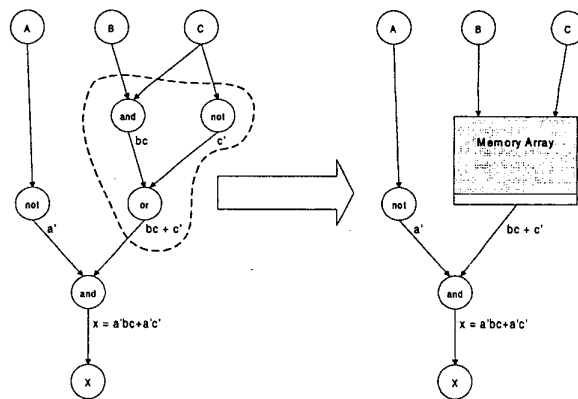


Figure 2: Replacing logic with a memory array

5 Results

The results for an ESB configured as a conventional memory were presented in [10]. In this section, we focus on the results for an ESB in product-term mode. We then combine the results to consider dual-mode arrays.

Effect of Memory Depth

As shown above, a memory with depth d can implement functions of at most $d/2$ literals. Intuitively, the larger d is, the more logic that can be packed into each memory. This is shown in Figure 3(a), which shows the amount of logic packed into a single memory as a function of d , for three different widths w . Figure 3(b) shows the chip area required by a single array, as a function of d ; clearly, as d increases, the area increases. The packing ratio, which is defined as the ratio of the number of logic nodes packed to the chip area is shown in Figure 3(c). As the graph shows, the optimum choice for d is 32 if $w = 16$ or 32, and 64 if $w = 64$ or 128. For arrays deeper than this optimum, the packing tool was unable to effectively use the extra memory space, while for arrays shallower than the optimum, only small logic functions could be implemented by the memory. The dependence of the optimum d on w is

small, and can be explained by noting that a larger w implies more product terms; more product terms work well when there are more literals (and hence, a higher value of d).

Figure 4 shows the packing ratio results assuming there are 16 product-term memory arrays available. In this case, the optimum value of d is the same as if there is only one memory array. This is in contrast to the study in [10], which showed that as the number of arrays increased, smaller arrays were preferred. In that study, a standard memory was targeted. Standard memories work well for subcircuits with few inputs that fan out to a large number of nodes and subsequently are combined into only a few outputs. This sort of subcircuit pattern does occur, however, in any user circuit, there are typically only a few such instances of this pattern. Once this "low-hanging fruit" as been packed, the standard memories are not as effective in implementing logic. A product-term memory array, on the other hand, works well for circuits with a relatively larger number of inputs but fewer product terms. These subcircuit patterns appear commonly in circuits, so larger numbers of memories can be effectively filled. Thus, the optimum memory depth is not as dependent on the number of arrays as it was in [10]. We repeated the experiment assuming 2, 4, 8, and 16 product-term arrays, and the conclusions held.

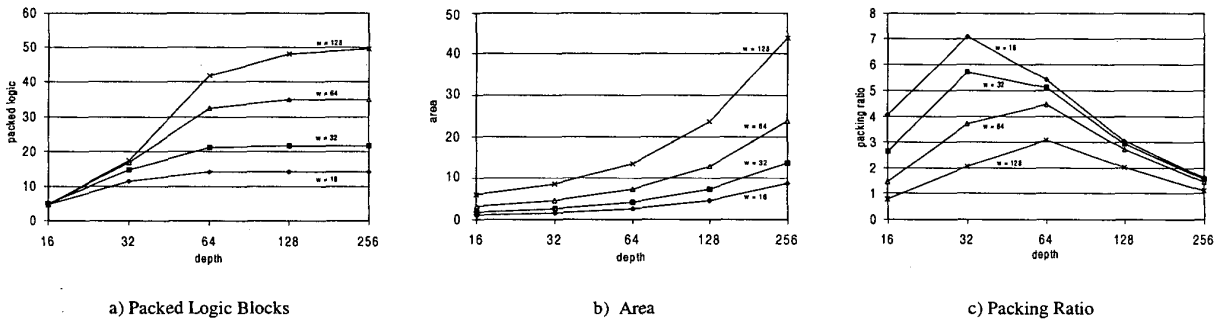


Figure 3: Effect of Memory Depth, $N = 1$

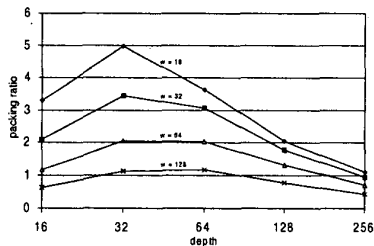


Figure 4: Effect of Memory Depth, $N = 16$

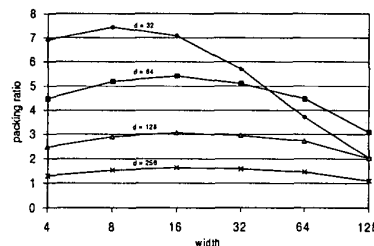


Figure 5: Effect of Memory Width, $N = 1$

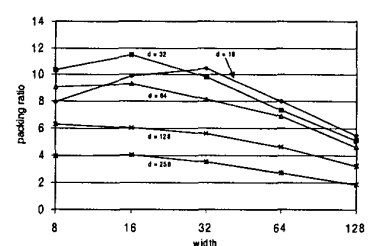


Figure 6: Effect of Memory Depth in a Dual-Mode Memory, $N = 1$

Effect of Memory Width

A memory of width w can implement w product-terms. Intuitively, a wider array can implement more logic, but requires more area. The packing ratio (amount of logic per unit area) as a function of w , for four different values of d , is shown in Figure 5. This figure assumes one memory array, but the results are similar if there is more than one array available. As the graph shows, the optimum width is 16 for a memory with a depth of 64 or more, and 8 for a memory with a depth of 32. Of all the combinations we investigated, the best packing ratios were obtained when $w=16$.

Dual-Mode Memory Arrays

In the Altera APEX20k, each array can be independently configured as either a standard memory or a product-term memory. Our CAD suite takes advantage of this flexibility by mapping logic twice: once assuming a standard memory and once assuming a product-term memory. The better of the two mapping results is chosen, and the array is configured in the appropriate mode.

For dual-mode arrays, we must consider the effect of the memory dimensions on the CAD tool that packs logic to standard memories as well as the CAD tool that packs logic to product-term memories. Intuitively, the optimum array architecture will be a compromise between the optimum architectures for each mode. Figure 6 shows the packing density as a function of width for several depths. As the graph shows, the optimum width is 32 for a memory with a depth of 16 bits, 16 for a memory with a depth of 32 or 64, and 8 for a memory with a depth of 128 or 256, although there is only a very small difference between the packing ratios at 8 bits wide and at 16 bits wide when the memory has a depth of 128 or 256 bits. The optimum depth was found to be 32 bits for various widths.

6 Sensitivity of Results

Clearly, our architectural results are sensitive to the CAD tools employed and the circuits used. We did not have access to another product-term memory mapper (which was flexible enough for our purposes here), however, we were able to consider the effects of pre-mapping optimization on the conclusions. We repeated the experiments using Chortle [13] to technology-map the circuits (rather than Flowmap) and found that the conclusions held. We also attempted to optimize the circuits more heavily (by running SIS scripts twice) before mapping; again, there was no change in the architectural conclusions.

7 Conclusions

It has been previously shown that having on-chip memories in an FPGA is beneficial because memory implementations of logic are very efficient in terms of area. In this paper, we have examined how the width and depth of product term mode memory arrays affect the amount of logic that can be packed into the arrays. We have found that each memory array should have a depth of 32 or 64 bits, and a width of 16 bits.

References

- [1] Xilinx Inc., Virtex-E 1.8V Extended Memory Field-Programmable Gate Arrays Datasheet, ver. 1.4, Apr. 2001.
- [2] Altera Corp., APEX20k Programmable Logic Device Family Datasheet, ver. 4.0, August 2001.
- [3] S. J. E. Wilton, "SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, February 1995, pp. 97-103.
- [4] Cong, J. and Xu, S., "Technology Mapping for FPGAs with Embedded Memory Blocks," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1998, pp. 179-187.
- [5] Heile, F. and Leaver, A., "Hybrid Product Term and LUT Based Architectures Using Embedded Memory Blocks," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1999, pp. 13-16.
- [6] E. Lin, and S. J. E. Wilton, "Macrocell Architectures for Product Term Mode Embedded Memory Arrays," in *Proceedings of the International Conference on Field-Programmable Logic and Applications*, Belfast, UK, August 2001, pp. 48-58.
- [7] A. Kaviani and S. Brown, "The Hybrid Field-Programmable Architecture," *IEEE Design and Test of Computers*, Vol. 16, No. 2, pp. 74-83, Apr-Jun, 1999.
- [8] J.L. Kouloheris, A. El Gamal, "PLA-Based FPGA Area vs. Cell Granularity," in *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 4.3.1-4.3.4, May 1992.
- [9] J.A. Anderson, "Architectures and Algorithms for Laser-Programmed Gate Arrays with Foldable Logic Blocks", M.A.Sc. thesis, University of Toronto, 1997.
- [10] S. J. E. Wilton, "Implementing Logic in FPGA Embedded Memory Arrays: Architectural Implications," in *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pp. 12.4.1-12.4.4, May 1998.
- [11] E. Sentovich, "SIS: A System for Sequential Circuit Analysis," Tech. Rep. UCB/ERL M92/41, Electronics Research Laboratory, University of California at 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, Jan. 1994.
- [13] R. J. Francis, J. Rose, Z. Vranesic, "Chortle: A Technology Mapping Program for Lookup Table Based FPGAs," in *Proceedings of ACM/IEEE Design Automation Conference*, pp. 613-619, June 1990.