IMPLEMENTING ARRAY MULTIPLIERS IN XILINX FPGAS

by Robert W. Canik of National Instruments and

Dr. Earl E. Swartlander, Jr. of the University of Texas at Austin

Abstract

This paper discusses aspects to consider when mapping array multiplier designs to field programmable gate arrays (FPGAs). FPGAs provide configurable logic through an array of configurable logic modules interconnected by programmable routing resources and surrounded by programmable Input/Output blocks. However due to the lack of consistent structure most typical designs do not map well to FPGAs. The structure of array multipliers make them a natural fit for FPGA realization, potentially delivering the performance and utilization originally promised at the introduction of FPGAs. Two design examples are developed and mapped to the Xilinx family of FPGAs. The results of this effort are reported and projections are made as to how the designs performance vary when they are scaled for larger applications and the speed grades of the components are changed.

1: Introduction

The Field Programmable Gate Array (FPGA) was introduced in 1985 with the intention of combining the logic density and versatility of gate arrays with the time-to-market advantages and off-the-shelf availability of user-programmable standard parts. Since that time several companies have introduced a wide range of FPGAs with different gate counts and speed grades. In this paper we focus attention on parts developed by Xilinx. All the parts Xilinx offers utilize the same general architecture which include three major configurable elements: configurable logic blocks (CLBs), input/output blocks (IOBs), and interconnections. An array of Configurable Logic Blocks (CLBs) provides the functional elements from which the user's logic is constructed. The logic blocks are arranged in a matrix in the center of the device. Each logic block has a combinatorial logic section and storage elements. The combinatorial section can realize any two logic functions of the inputs of the CLB which generate two outputs. These outputs may be latched by the storage elements in the CLB. The input/output blocks (IOBs) are user-configurable and provide an interface between the external package pins of the device and the internal logic. The IOBs can be configured to be an input or output, registered or combinatorial, and active-high or active-low. Programmable interconnections provide routing paths to connect IOBs and CLBs into the desired network. All interconnects are composed of metal segments with programmable switching points. Three types of routing resources are available in Xilinx FPGAs: general purpose interconnects, longlines, and direct connects. General purpose interconnects consist of horizontal and vertical groups of signals that separate the CLBs in the array and the configurable switch matrices at their intersections. The switch matrices allow connections to be made between the vertical and horizontal metal segments. Longlines are metal segments that run the entire length of the die vertically and horizontally. These segments bypass the switching matrices thus providing a low skew path for signals that must go across the die. Direct connections are dedicated metal paths between adjacent CLBs that encounter no switching elements thus introduce negligible delay. The interconnection resources have proven to be the most troublesome part of a Xilinx design. Some designs with very random distributions of interconnection requirements may map into the available CLBs of a Xilinx part but will not be routable. When designs can be routed in an FPGA the speed of the overall design is usually limited by the worst routing interconnection. Xilinx has introduced three families of FPGAs since 1985, each utilizing the same basic architecture while varying in the complexity of the CLBs, amount of interconnect resources, gate count (die size), and speed grades. All three families are re-configurable via their utilization of SRAM that must be loaded on power-on with configuration data. [1]

Because of the flexibility of the Xilinx FPGA architecture, designers are tempted to utilize these FPGAs for general purpose applications. However due to the limited logic of each CLB, multiple CLBs must be utilized to realize practical circuits. This may lead to considerable delay as significant routing resources are engaged to interconnect the required CLBs The speed of the design is limited the worst routing path of the design. Since most designs are partitioned into register intensive modules and combinatorial sections, the mix of flip-flops and combinatorial logic at the CLB level results in poor overall utilization of the FPGA resources. Register intensive designs make little use of the combinatorial

1058-6393/95 $4.00 © 1995 IEEE

capabilities of the CLBs, while combinatorial designs make no use of the flip-flops in the CLBs.

The logic capability and input/output structure of CLBs make them natural fits for counter and adder applications. With multiple inputs and two outputs, CLBs are perfect for realizing full adders. For counter or pipelined adder applications the flip-flops in the CLBs are also used, yielding an extremely high utilization of the CLB resources. Since Xilinx has had FPGAs available for almost a decade, designers have realized this potential and the majority of the application notes developed for these FPGAs deal with the topic of designing fast adders and counters.

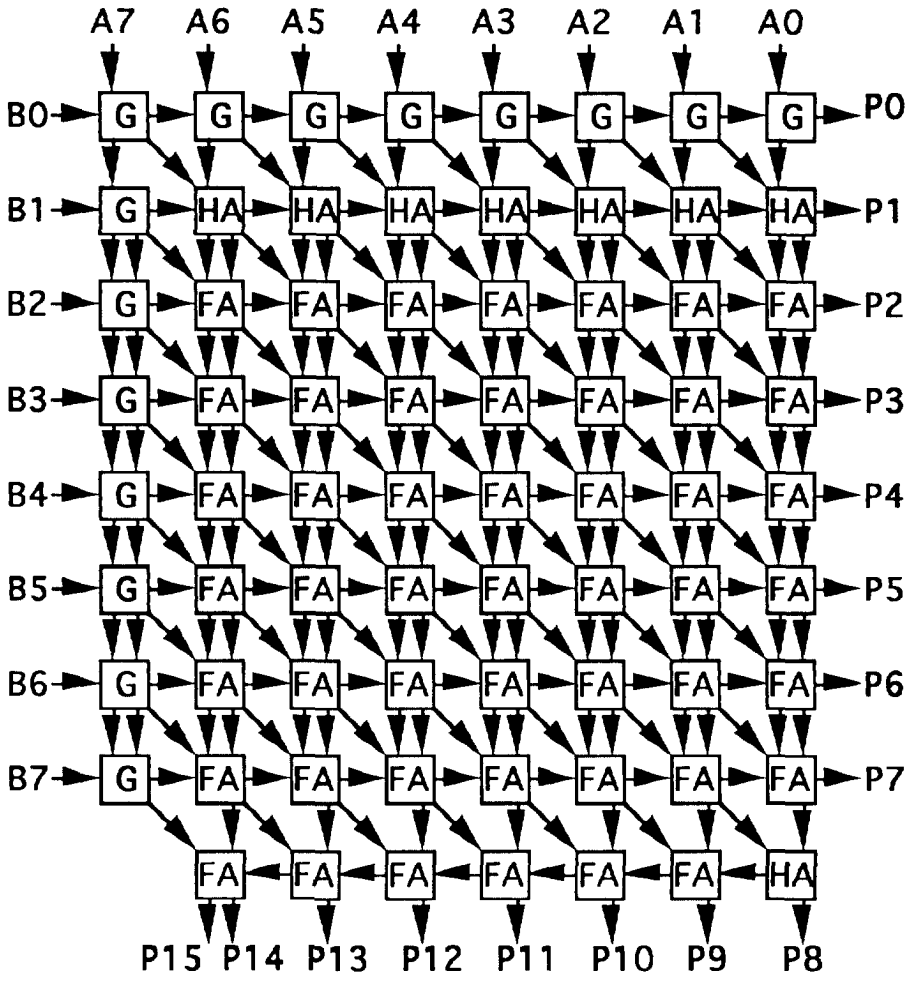
The ideal application for an FPGA must meet three criteria. First, the design must require a regular pattern of small combinatorial modules that will fit into a CLB. Second, these small modules must only require interconnections to adjacent modules to minimize the build up of routing delay. And finally, an application should be able to utilize the flip-flops at the CLB outputs to acheive a higher utilization of it's resources.

The regular distribution of full adders required by array multipliers make them ideal candidates for efficient realization in a Xilinx FPGA. Array multipliers are one of several techniques utilized to multiply two binary numbers. Array multipliers combines the generation of all bit products and their summation with an array of full adders. Figure 1 illustrates the network of AND gates and full and half adders required to implement an 8 x 8 array multiplier. The delay required to generate the product in an N x N array multiplier is (2N - 2) adder delays [2]. An array multiplier can also be used to multiply two 2's complement numbers by inverting the outputs of the AND gates in the left most column and the AND gates embedded in the bottom row of special full adders and converting the half adder in the bottom right hand corner to a full adder with a hardwired one for one of the inputs [3]. By inspection of the network in Figure 1 it is obvious that the array multiplier's full and half adders can map directly into the CLBs of an FPGA. The special full adders required of array multiplier (they are required to AND two signals together to generate one of the bits to be added) must be able to generate two outputs of the four input signals. The input signals are a bit of the multiplicand, a bit of the multiplier, the carry out of the full adder above it, and the sum output of the full adder diagonally above it. The CLBs available in the Xilinx 3000 family exactly meet this requirement being capable of generating any two functions of four inputs. The required interconnect shown in Figure 1 indicate that array multiplier also meets the second requirement of ideal applications for an FPGA, that being the logic modules only require interconnection to adjacent modules. For pipelined array multipliers the two flip-flops in the CLBs can be used to latch the carry and sum outputs thus utilizing the maximum potential of the CLBs resources.

2: Approach

To illustrate how well the Xilinx FPGAs are suited for the realization of array multipliers two design options were investigated. First, a simple, 8 x 8 combinatorial array multiplier was developed and mapped into a Xilinx 3120 FPGA and analyzed. Then a more complex, 8 x 8 fully pipelined array multiplier was developed, mapped into a Xilinx 3190 FPGA, and analyzed. The following sections illustrate the design approach and results.

An 8 x 8 combinatorial array multiplier was developed by directly mapping the 8 x 8 array multipliers logic blocks in Figure 1 into an array of 8 x 8 CLBs in a Xilinx 3120 FPGA. One modification was required since the block diagram in Figure 1 shows a 9 x 8 array of AND gates and adders. The top two rows of the block diagram were condensed into one row of CLBs that each contained two AND gates and a half-adder. To assist the placement and routing process the design was captured using a Viewlogic schematic entry package with a library of Xilinx schematic symbols. The Xilinx library allows you to attach an attribute to a schematic symbol that will dictate which CLB in the FPGA that a particular logic component will get mapped into. However, only a limited number of symbols will allow this attribute to be associated with it. This forced the design to be captured in its most fundamental level of granularity: the CLB primitive symbol. The CLB primitive is essentially a schematic symbol that exactly represents a CLB in the FPGA. Its functionality is determined by typing equations into the schematic symbol itself. Thus the entire design was captured by creating an array of CLBs, attaching the appropriate input/output signals, typing in their internal equations, and attaching a location attribute to dictate their placement within the design. Dictating the location of the CLBs is important because while the Xilinx tools will probably be able to generate a placement and routing scheme to realize the design, the performance of the design will be limited by the worst case routing path. This worst case path can be created by two poorly placed CLBs. The Xilinx placement and routing tools were used to place the IOBs around the perimeter of the array and route the interconnections. Several features in the Xilinx FPGA were utilized to aid the mapping of the design into the part and routing the interconnections. The longlines that run across the entire length of the design horizontally and vertically were used to carry the multiplier and multiplicand bits to all the CLBs in their perspective rows and columns. The hardwired direct connects were used to route the full adder carry outputs to the adjacent CLBs. The sum outputs of the full adders were the only signals required to use the general purpose routing resources to make their connections however these connections only went to the diagonally adjacent CLB. Thus an 8 x 8 array multiplier used all but one of the CLBs in a 8 x 8 FPGA . However none of the internal flip-flops of the CLBs were utilized in the combinatorial



where: G = AND Gate
FA = Full Adder
HA = Half Adder

Figure 1 - Block Diagram of an 8x8 Array Multiplier

design. After the design was captured and routed, it was simulated to ensure proper functionally using Viewlogic's simulator and a delay analyze performed using the Xilinx tool, XDelay, to determine the worst case delay through the part thus indicating its operational performance.

After the completion of the combinatorial 8 x 8 array multiplier design and verification, a completely pipelined version of an 8 x 8 array multiplier was developed. This was very similar to the combinatorial multiplier however every full adder output was latched by the flip-flops in the CLBs and the timing of the entrance and exit of the multiplier, multiplicand, and product bits to the array had to be controlled via shift registers. The multiplicand bits (A[7:0]) must be delayed one clock cycle at each level as they move from the top row of the array to the bottom. Since the full adder CLBs they enter only have two outputs that are already used they cannot be used to delay the multiplicand bits, thus rows of CLBs in the FPGA must be dedicated to building the shift register required to time the arrival of the multiplicand bits. Since CLBs contain two flip-flops only one row of CLBs is required to build the shift register required for every two rows of full adder CLBs. The shift register rows should be every third row in the FPGA to keep the shift register flip-flops close to the full adder CLBs that they drive to reduce routing delays. While the multiplicands bits must be delayed one clock cycle for every row they encounter, the multiplier bits arrive at all CLBs in their perspective rows at the same instance. The arrival of the data to the entire row must be delayed one clock cycle per row starting at the top row (0 shifts) to the bottom row (7 shifts). Thus the least significant bit (LSb) of the multiplier goes straight to the first row of CLBs, bit B1 arrives at the second row one clock later after going through a one bit shift register, and so on until the seventh bit of the multiplier, B7, arrives at the seventh row seven clocks later after going through a seven bit shift register. Since the multiplier bits (or the delayed versions) go to

the entire row of CLBs at the same instance they can still utilize the horizontal longlines for interconnection. Because of the way an array multiplier processes data the LSb of the product, P0, is available after the first clock, the second LSb, P1, arrives one clock later and so on until the two most significant bits, P15 and P14, depart from the last full adders carry and sum outputs fourteen clocks later. Typically the designer would prefer the data come out simultaneously thus requiring all products bits except the two most significant bits to be delayed through shift registers until the two most significant bits are ready. Therefore P0 must be delayed through a fourteen bit shift register, P1 through a thirteen bit shift register, and so on. The inputs to the ripple carry adder at the bottom of the array multiplier must also be delayed through shift registers if the ripple carry adder is to be pipelined. If the ripple carry adder is not pipelined it will easily be the slowest combinatorial path through the design thus limiting the overall performance therefore its inputs were delayed so it could be completely pipelined. With this completely pipelined design the performance will be limited by the worst case combinatorial path which will be the clock-to-output of a CLB plus routing delay to one nearby CLB plus the data setup time of that CLB. No path is required to go through multiple CLBs without encountering a flip-flop. The longest routing delay encountered in the pipelined array multiplier will be the horizontal longlines used to carry the multiplier bits to the CLBs in their perspective rows. It can be proven that an N x N fully pipelined array multiplier can be mapped into a 2N x 2N CLB array FPGA part. The increase in CLB usage over the combinatorial design is required to accommodate the shift registers. Thus the 8 x 8 array multiplier for the design example required a 16 x 16 array of CLBs found in the Xilinx 3190 FPGA. The pipelined design was captured, routed, and analyzed in the same manner as the combinatorial design.

3: Results

As mentioned in the previous section the designs were analyzed with the Xilinx timing analyzer tool, XDelay, and their function verified with the Viewlogic simulator, Viewsim. This section presents the results of the analysis of the two designs.

The non-pipelined array multiplier provided the best utilization of the FPGA resources but the poorest overall performance. The design used all but one of the available CLBs in an 8 x 8 array in the Xilinx 3120 thus using 98.4% of the combinatorial resources in the CLBs but none of the flip-flops available. This utilization will scale well as larger designs are mapped into larger FPGAs; for example, mapping a 32 x 32 array multiplier into a 32 x 32 array of CLBs in the Xilinx 4025 part will yield a CLB utilization of 99.9%. By dictating the placement of each full adder and AND gate, the Xilinx router had no problem routing the required interconnections. The Xilinx timing analyzer was employed to determine the worst case combinatorial delay through the part. The results of the timing analysis show as expected the worst case paths are from the least significant bits of the multiplier, B1 and B0, and ripple down the right side of the array and across the ripple carry adder to generate the most significant bit of the product, P15. The worst case path has a delay of 61.8 nsec yielding an operating frequency of over 16 MHz. The simulation results indicate the individual bits ripple out of the multiplier at different times but all bits typically settle within 47ns after applying a new pattern.

The Xilinx 3000 family offers a wide variety of FPGA components ranging in size from an 8 x 8 array of CLBs up to 22 x 22 (with the 4000 family going up to 32 x 32). Within the 3000 family Xilinx offers a full range of speed grades from the slowest 30XX-50 part to the fastest 31XX-3 components. These two combinations allow the designer to select an array size to accommodate the width of the two numbers to be multiplied and the speed grade required to meet the performance requirements of the overall system design. However the performance is not a constant with respect to the array size since as the multiplier and multiplicand increase in size the number of rows the signals must ripple through and the size of the final ripple carry adder increase. By using the average of the routing and CLB delays from the timing analysis, projections can be made as to the decrease in performance that can be expected as the array size increases. The difference in performance due to the different speed grades used can also be projected by multiplying the delay scaling factor for each speed grade to the expected delay of the fastest component offered, that being the 31XX-3 parts. By using the Xilinx design editor, XACT, it was possible to re-target the design to all the different speed grades offered for the 3120 and 3020 components and re-analyze the performance of the re-targeted designs. The results of this analysis was then used to project the performance of larger array sizes and different speed grades which is shown in Table 1.

	DELAY		
Array Size	**8 x 8**	**12 x 12**	**16 x 16**
Part #	3120/3020	3142/3020	3190/3090
31xx-3	61.8 ns	92.6 ns	123.4 ns
31xx-4	73.7 ns	110.2 ns	143.1 ns
31xx-5	87.6 ns	131.5 ns	175.2 ns
30xx-125	108.0 ns	162.1 ns	216.0 ns
30xx-100	134.2 ns	200.9 ns	267.8 ns
30xx-70	173.5 ns	260.2 ns	346.4 ns
30xx-50	261.9 ns	392.6 ns	523.2 ns

Table 1 - Width & Speed Grade vs Performance

The fully pipelined design was then analyzed in the same manner as the non-pipelined design. The utilization of the CLBs was determined to be 195 of the

256 CLBs (16 x 16) available or 76.2%. However not all the combinatorial resources in some of the CLBs were used. Only the CLBs which generated bit products (AND gates) and full or half adders use the combinatorial resources in the CLBs. Thus yielding only 63 of the 256 CLBS or 24.6% utilization of the available combinatorial resources, nowhere near the utilization of the combinatorial resources of the non-pipelined array multiplier. Likewise some of the shift registers and bit product CLBs only use one of the available flip-flops in a CLB yielding a utilization of 366 of the 512 available flip-flops or 71.5% utilization. The Xilinx timing analyzer was employed to determine the worst case combinatorial delay through the part. Unlike the results of the non-pipelined design the longest combinatorial path does not ripple through multiple CLBs. Instead the longest path is determined by the worst case route of any signal between two CLBs since the clock to output and data setup time is the same for each CLB. From the timing analysis it was no surprise to see that the worst case route is from the multiplier bits (or their delayed versions) to the entire row of CLBs they must drive. These nets are the longest routing paths in the design because all the other nets only have one output and one CLB input that is nearby due to their placement. The worst case path has a delay of 10.0 nsec yielding an operating frequency of 100MHz.

An interesting characteristic of the pipelined design is its scalability. Since its performance is determined by the worst case routing delay between two interconnected CLBs, the throughput performance of the design remains constant as the array size varies. This characteristic allows the designer to ignore the impact on performance that the array size has as he selects the array size. The array size only affects the latency of the multiplier not the throughput. Therefore the only variable that effects the performance on the design is the speed grade of the part. Like the non-pipelined design the Xilinx design editor was used to re-target the design to all the different speed grades offered for the 3090 and 3190 components and the performance of the design was then re-analyzed using the timing analyzer, XDelay. The results of this analysis are shown in Table 2.

Part # - Speed	Worst Delay	Operating Freq.
3190-3	9.7 ns	103.1 MHz
3190-4	11.5 ns	87.0 MHz
3190-5	13.2 ns	75.8 MHz
3090-125	16.9 ns	59.2 Mhz
3090-100	21.1 ns	47.4 MHz
3090-70	26.0 ns	38.5 MHz
3090-50	42.1 ns	23.8 MHz

Table 2 - Speed Grade vs Performance

4: Conclusion

The results of the analysis of the two designs reveal several interesting characteristics. The non-pipelined design, while utilizing a high percentage of the combinatorial resources of the CLBs, used none of the available flip-flops. While it did provide the fastest latency time to calculate the product of two binary numbers (61.8 nsec) this latency did not scale well as the design technique was ported to larger arrays. The non-pipelined design is best suited for scalar applications particularly were cost is of great concern. The pipelined design provided a constant level of performance as the array size varied but suffered from poor utilization of the combinatorial and register resources in the CLBs. The utilization of the pipelined design scaled poorly with respect to array size as an N x N array multiplier requires a 2N x 2N array of CLBs to be realized. The pipelined design is best suited for vector operations or signal processing applications where emphasize is on raw throughput and not latency or cost. Xilinx FPGAs provides both design choices a low cost, off-the-shelf solution with a low time-to-market design cycle.

Future work in the area of utilizing Xilinx FPGAs for computer arithmetic application is as boundless as the flexibility of the FPGAs themselves. One exciting possibility is an array of interconnected FPGAs that would provide a programmable computation engine. Since the Xilinx FPGAs are programmable, this array of components could be dynamically configured to perform different functions on an incoming stream of data in a signal processing application. Each Xilinx could be configured to perform multiplication, addition, or subtraction to incoming data streams before passing the output data to the next FPGA for further processing. The hardware could be reconfigured on the fly as the needs of the application change or the same hardware could be used for different applications.

Increases in performance and better utilization of FPGA resources will become possible in future generations of FPGAs. As Xilinx and other FPGA vendors move their architectures to faster technologies the performance of arithmetic units realized with FPGAs will increase. More dramatic improvements in performance and utilization may be realized as the capabilities of CLBs improve with future generations.

5: References

[1] The Programmable Logic Data Book, XILINX, 1994.

[2] Earl Swartzlander, Jr., "Computer Arithmetic," in C. H. Chem, Ed., Computer Engineering Handbook, New York: McGraw-Hill, 1992, pg 4.12-4.13.

[3] Charles Baugh and Bruce Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," IEEE Trans. Comput. Vol. C-22:pp. 1045-1047, 1973.