

Architecting Wide-bit Multipliers on Programmable Logic Devices¹

J. P. Davis, S. Devarkal, N. Sontineni
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208 USA
jimdavis@cse.sc.edu

Abstract

In this paper, we explore some of the architectural issues associated with creating high-performance wide-bit multiplier units on programmable logic devices. These devices are the “fabric” of a reconfigurable computing platform consisting of multiple commercial FPGAs. Many applications of reconfigurable computing for arithmetic, such as for DSP and Cryptography, as reported in the literature, have been carried out with relatively small word-widths, thereby avoiding some of the more difficult design tradeoffs and scheduling control issues associated with larger operand word widths. In this paper, we explore some of the results we have obtained in exploring the architectural space associated with creating wide-bit multipliers for cryptography applications. Specifically, we explore these multiplier topologies for multipliers with 192 and 256-bit operands, taking into account some of the optimizations available using some of the given FPGA device architectures—thereby exploring the impact of device selection on final multiplier architecture selection and refinement.

1. Introduction

Reconfigurable computing machines (RCM) have become recognized as a viable means for achieving orders of magnitude speedup in compute-intensive applications, through the use of fine-grained parallelism on a customized logic substrate [1, 2]. Algorithms in a wide variety of military and scientific domains--along the architectures necessary for their implementation--are being implemented on a custom-logic "fabric" consisting of one or more commercially-available field programmable gate arrays (FPGAs) [3]. This fabric is generally suited to fine-grained parallelism [1]. Highly repetitive operations, such as pipelined arithmetic calculations on wide data words, and highly parallel SIMD operations--such as those found in image processing and cryptography--are a good match for an RCM application [1].

Our objective is to create multiplier architectures that can be efficiently implemented on programmable logic devices, such that we can realize high-performance wide-bit arithmetic processing for a range of scientific and military applications. In this paper, we examine one aspect of this problem, namely, the analysis of architectures supporting multiplication of operands 192 and 256 bits in width. We examine this activity of selecting appropriate unit widths and architecture renderings of available multiplier algorithms, and then we examine the effect of a programmable device's available capabilities and resources on this selection. The process of finally converging on an effective architecture and implementation requires iterating through the architecture and design process many times before finally converging on an optimal design for the application and using the target device's resources most effectively.

2. Applications and algorithms

We seek to articulate a set of specialized heuristics that guide us to appropriately use different multiplier architectures in varying applications of crypto-arithmetic processing. To that end, our focus of study is outlined as follows:

1. Identify and model different Multiplier circuits. For consideration in this paper, we identify and select appropriate schemes that can be used on unsigned integer arithmetic appropriate for cryptography.
2. Select an appropriate unit size – we need to select a basic unit for constructing large-width multiplier data paths. Since it is impractical to construct monolithic circuits of very large widths, we must model our different multiplier schemes as basic units, from which we will construct larger units.
3. Having modeled the basic multiplier units, we subject these units to benchmarking. As pointed out in Chen and Mead [4], what might be

¹ The work has been supported by the NSA Lucite program.

considered an efficient or optimal algorithm in a mathematical or software sense may not be so from the standpoint of its architecture because of physical cost considerations. Therefore, we must subject each model to cost evaluation. For synthesized circuits, this is generally done in terms of area, speed and power characteristics. Since we are considering our implementation on a reconfigurable parallel computing platform using an array of Xilinx FPGAs as processing elements (PE), we will for the moment ignore the power issue and focus on the issues of area and speed. Area is important, as we need to be able to assess the ability of a multiplier unit to scale size, while also continuing to “fit” on the FPGA device. Otherwise, we incur performance penalties by having to go “off chip” to complete the arithmetic function. Speed is important for obvious reasons, in that minimizing computation time (in the form of worst-case delay through the unit) affects the overall time for computation.

4. Having characterized each of the identified and modeled multiplier units, we then devise scaled models of the computation architecture for the data path widths that are of interest. We are dealing with unsigned integer arithmetic of the following data path widths: 128, 192, 224, 256, 384 and 521 bits in width. (Note that the 521-bit computation is rounded up to 544 bits, as this allows use of a standard unit configuration for this purpose.) We assume at present that the architectures for each of the bit widths may dictate a different—or even more than one—multiplier architecture for use in a given reconfigurable application. Therefore, to appropriately account for this possibility, we are analyzing the larger multiplier architectures using different basic multiplier units, and evaluating the performance of each configuration.

It is impractical to construct wide-bit multipliers of 192, 256 or 384-bits as monolithic circuits. As discussed in [4], a hierarchical design approach allows better management of design complexity by exploiting the regularity of large numbers of identical structures, and also affords a better chance that a usable and efficient circuit can be synthesized. Therefore, we explore a number of different multiplier architectures, identifying a set of basic 32-bit units, using these basic units to construct the larger multiplier circuits with wide data path widths.

3. Architectural treatment

In the figure below, we depict one of two wide-bit architecture schemes we have been exploring. In this “broadcast” scheme, we break up a 192-bit radix into six separate 32-bit multiplier units, whose outputs are fed into 64-bit pipeline registers, according to a “perfect shuffle” permutation scheme [15]. The partial product registers are grouped into 3-units, where each group is treated as a unit for purposes of shifting partial product data. The leftmost 3-unit, labeled as Registers Rb, Rd, Rf form a 192-bit shift register, on which a 32-bit SHL operation occurs after each multiplication cycle.

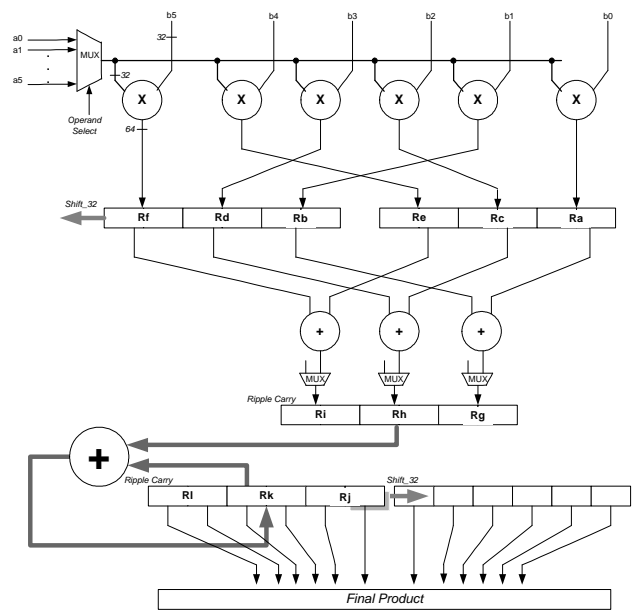


Figure 1. Broadcast-style 192-bit multiplier

The data stored in these registers is then clocked into three 64-bit adder units, organized into an “exchange permutation” scheme [15] whose outputs are fed into a second stage of pipelined registers. The following stage adds the results of these three registers, grouped as a 3-unit, with an “accumulator” register 3-unit consisting of three additional 64-bit pipelined registers. During this final addition, the contents of the permuted partial product addition registers are added to the running value stored in the accumulator. Between each accumulator add, the contents of the 3-unit accumulator are right-shifted 32-bits into an additional 3-unit register group. After six such accumulator-adds and 32-bit shifts, a complete final product is contained within the running 384-bits of 64-bit registers organized into two 3-unit registers. A final product register, consisting of six 64-bit registers, stores the product for latching by another stage of the application.

Based on preliminary assessment reported elsewhere [5], we have settled on evaluating each of the following 32-bit multiplier units as base components to be used in the larger multiplier circuit. We've started with a basic set of multiplier architectures for the Shift-Add [6], the Pencil and Paper method [7], the Booth algorithm [8], and the Montgomery approach found in [9]. Whereas other algorithms exist, each of these techniques has been used for unsigned integer arithmetic, and suits our needs as initial units for study of the problem of scaling to wide data paths, and realizing these circuits in an RCM fabric consisting of multiple FPGAs.

4. Realizing logical architecture on the fabric

There are two different hybrid RCM platforms on which we have been exploring the architecture and design of cryptography applications: the HAL-15® and HC-36m®, both from Star Bridge Systems.

For purposes of this paper, we will concern ourselves with the resources available for use on the Xilinx® devices, and the estimated number of cycles required to pass data off-chip between different components of the wide-bit arithmetic units, and not on specific architectural properties of the Star Bridge platforms themselves.

This issue pertains to the trade-offs associated with selecting multiplier topologies that rely on concurrency through parallelism and/or pipelining, when a given bit-width cannot be accommodated on a single FPGA device. As such, we must consider the cost of moving data off-chip versus the extra cost associated with resource sharing in order to fit a model on the device.

5. Analysis and Experiments

In the paper, we will present the analysis steps, along with the general principles we have codified, for examining multiplier architectures of differing widths for the types of applications we are considering. In addition, we examine the effect that device selection has on the final realization of the multiplier architectures onto the programmable logic fabric. We will present results for multipliers of different configurations, for 192 and 256 bit operands, mapped onto two different classes of Xilinx® XC4000 and Virtex-II device families. Whereas the Virtex-II family of devices has a number of resource capabilities—such as built-in 18x18 multiplier units—the older XC4000 series devices come into play in applications where size and cost are considerations.

6. References

- [1] D.A. Buell, J.M. Arnold, and W.J. Kleinfelder (eds.), *Splash-2: FPGAs in a Custom Computing Machine*, IEEE Computer Society Press, Los Alamitos, CA, 1996.
- [2] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proceedings of the IEEE*, Vol. 86, 1998, pp. 615-639.
- [3] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, Vol. 34, No. 2, June 2002, pp. 171-210.
- [4] Chen, M. C., and C. A. Mead, "Concurrent Algorithms as Space-Time Recursion Equations", in Kung, S. Y., H. J. Whitehouse, and T. Kailath (eds.), *VLSI and Modern Signal Processing*, Prentice-Hall Publishers, 1985.
- [5] D.A. Buell, J.P. Davis, and G. Quan, "Reconfigurable Computing Applied to Problems in Communications Security", in *Proceedings MAPLD-02*, Laurel, MD, 2002.
- [6] Carpinelli, J. D., *Computer Systems Organization and Architecture*, Addison-Wesley Publishing Co., 2001.
- [7] Knuth, D. E., *The Art of Computer Programming, Volume 2, Semi-numerical Algorithms*, 3rd ed., Addison-Wesley Publishing Co., 1997.
- [8] Flynn, M. J., and S. F. Oberman, *Advanced Computer Arithmetic Design*, John Wiley and Sons, Inc., 2001.
- [9] A.F. Tenca, and C.K. Koc, "A Scalable Architecture for Montgomery Multiplication", in C.K. Kok and C. Paar (eds.), *Proceedings CHES '99*, LNCS 1717, Springer-Verlag Publishers, Berlin, Germany, 1999, pp. 94-108.
- [10] S. Bakshi, and D.D. Gajski, "Performance-Constrained Hierarchical Pipelining for Behaviors, Loops, and Operators", *ACM Transactions on Design Automation of Electronic Systems*, Association for Computing Machinery, Vol. 6, No. 1, Jan. 2001, pp. 1-25.
- [11] W.J. Fang, and A.C.H. Wu, "Multiway FPGA Partitioning by Fully Exploiting Design Hierarchy", *ACM Transactions on Design Automation of Electronic Systems*, Association for Computing Machinery, Vol. 5, No. 1, Jan. 2000, pp. 34-50.
- [12] F.R. Boyer, and E.M. Aboulhamid, "Optimal Design of Synchronous Circuits Using Software Pipelining Techniques", *ACM Transactions on Design Automation of Electronic Systems*, Association for Computing Machinery, Vol. 6, No. 4, Oct. 2001, pp. 516-532.
- [13] M. Balakrishnan, and H. Khanna, "Allocation of FIFO Structures in RTL Data Paths", *ACM Transactions on Design Automation of Electronic Systems*, Association for Computing Machinery, Vol. 5, No. 3, Jul. 2000, pp. 294-310.

- [14] A. Darte, R. Schreiber, B.R. Rau, and F. Vivien, "Constructing and Exploiting Linear Schedules with Prescribed Parallelism", *ACM Transactions on Design Automation of Electronic Systems*, Association for Computing Machinery, Vol. 7, No. 1, Jan. 2002, pp. 159-172.
- [15] S.Y. Kung, *VLSI Array Processors*, Prentice Hall Publishers, Englewood Cliffs, NJ, 1988.
- [16] D. Kulkarni, W. A. Najjar, R. Rinker, and F. J. Kurdahi, "Fast area estimation to support compiler optimizations in FPGA-based reconfigurable systems," *Proceedings FCCM-02*, J. Arnold and K. Pocek (eds.), IEEE Press, 2002, pp. 239-247.
- [17] J. Babb, M. Rinard, C. A. Moritz, W. Lee, M. Frank, R. Barua, and S. Amarasinghe, "Parallelizing applications into silicon," *Proceedings FCCM-99*, K. Pocek and J. Arnold (eds.), IEEE Computer Society Press, 1998, pp. 70-80.
- [18] Xilinx, Inc., *Vertex-II 1.5V Field Programmable Gate Arrays: Advance Product Specification*, DS031-1, DS031-2, and DS031-3, September 22, 2002.
- [19] Xilinx, Inc., *XC4000XLA/XV Field Programmable Gate Arrays: Product Specification*, DS015, October 18, 1999.
- [20] Star Bridge Systems, *Description of the Hypercomputer® Hardware*, at <http://www.starbridgesystems.com>.