

ECE 734 Project Proposal

Exploring realizations of large integer multipliers using embedded blocks in modern FPGAs.

Shreesh Srinath

Motivation

Multiplication functions constitute the kernel of many real-life applications. They are used extensively in applications such as digital signal processing, image processing, cryptography and multimedia [1,2,3]. Recent computing oriented FPGAs feature embedded DSP blocks including small embedded multipliers. Achieving efficient realization of multiplication may have significant impact on the specific application in terms of speed, power dissipation and area.

FPGA vendors are now offering hardwired multipliers as one of the resources available to designers. Examples could be that of Xilinx Spartan-3 Family which includes 104 on-chip 18x18 multipliers and Xilinx Virtex-5 & 6 Family which include 25x18 multipliers. Optimized realizations of large multipliers of large integer multipliers using such blocks are studied in [4,5,6]. This project aims to study different approaches to implement large integer multipliers on DSP blocks in an efficient manner in terms of both timing and area.

Related Work

In [4,5], the authors present an efficient design methodology and systematic approach for implementation of multiplication and squaring functions. They propose a general architecture and a set of equations are derived to aid in realization. The method used is that of the “Divide and Conquer Algorithm” [7] with efficient organization of partial products. The symmetric embedded block considered is of size “ $n \times n$ ” and operands of size “ k ” such that $(n \times (m-1)) < k \leq (n \times m)$. The Table 1, below gives the sizes, in bits, of the partial products in the multiplication expression of two operands “ X and Y ”.

Table 1: Sizes, in bits, of the partial products in expression (5)

Partial products	Number of bits
$X_{m-1} \times Y_{m-1}$	$2 \times (k - (m - 1) \times n)$
$X_i \times Y_i$, where $i = m - 2, m - 3, \dots, 1, 0$	$2 \times n$
$X_{m-1} \times Y_i$ or $Y_{m-1} \times X_i$, where $i = m - 2, m - 3, \dots, 1, 0$	$(k - (m - 1) \times n) + n$
$X_i \times Y_j$ or $Y_i \times X_j$, where $i \neq j$ and $i, j = m - 2, m - 3, \dots, 1, 0$	$2 \times n$

The authors then look at timing and area efficient organization of the additions of partial products including the method of deferred parallel carry addition of partial products in which the set of carry bits generated from various levels of partial product additions are combined and processed later.

In [6], the authors note the use of *Karatsuba-Ofman* algorithm [8], and present detailed methodology of splitting the operations by use of the algorithms has been proposed to implement large multiplication using less number of DSP blocks which again target the “ $n \times n$ ” basic embedded multipliers. The authors mention that no reference to Karatsuba-Ofman algorithm for integer multiplication in was found in FPGA literature.

Project Highlight

The prior studies deal with the implementation of large multipliers using symmetric “ $n \times n$ ” embedded blocks. The goal of this project is to study the previous methods of [4,5,6] and develop a set of equations to assist the design and implementation of a large multiplier using asymmetric “ $m \times n$ ” embedded multipliers which are now available in the Xilinx Virtex-5 and 6 families. The approach is different to prior work and is novel as it deals with the asymmetric hardwired multipliers. An example of the approach is as given below. An example implementation based on the proposed approach is to be implemented on an FPGA device.

The DSP48E block in the Xilinx Virtex-5 & 6 contain 25 x 18 multipliers. Consider the multiplication of two numbers X and Y both sized k, which are split into the following chunks .

$$\begin{aligned} X &= [x_2 \ x_1 \ x_0] \text{ and } Y = [y_3 \ y_2 \ y_1 \ y_0]; \\ & \quad n > m; \\ X &= 2^{2n}.x_2 + 2^n.x_1 + x_0; \\ Y &= 2^{3m}.y_3 + 2^{2m}.y_2 + 2^m.y_1 + y_0; \end{aligned}$$

Consider the Multiplication $Z = X \cdot Y$. Substituting X and Y we get,

$$Z = (2^{2n}.x_2 + 2^n.x_1 + x_0) \cdot (2^{3m}.y_3 + 2^{2m}.y_2 + 2^m.y_1 + y_0).$$

$$\begin{aligned} Z &= 2^{2n}.(x_2.y_0) + 2^{2n}.(x_1.y_0) + (x_0.y_0) \\ &+ 2^{(2n+m)}.(x_2.y_1) + 2^{(n+m)}.(x_1.y_1) + 2^m.(x_0.y_1) \\ &+ 2^{(2n+2m)}.(x_2.y_2) + 2^{(n+2m)}.(x_1.y_2) + 2^{(2m)}.(x_0.y_2) \\ &+ 2^{(2n+3m)}.(x_2.y_3) + 2^{(n+3m)}.(x_1.y_3) + 2^{(3m)}.(x_0.y_3) \end{aligned}$$

Replacing the terms with a(1...4), b(1...4), c(1...4) & d(1...4) as below:

$$\begin{aligned} Z &= a_1 + a_2 + a_3 \\ &+ b_1 + b_2 + b_3 \\ &+ c_1 + c_2 + c_3 \\ &+ d_1 + d_2 + d_3. \end{aligned}$$

By observing the sizes of the products we can save the number of additions by grouping terms $(c_1, b_2, a_3) = A$, $(d_1, c_2, b_3) = B$, $(b_1, a_2) = C$, $(d_2, b_3) = D$ and terms $a_1 = E$, $d_3 = F$ remain. We save the additions as the terms grouped are effectively a simple concatenation and not additions. The remaining terms are now indicated by letters A, B, C, D, E, F. The result of the multiplication can now be obtained by clever additions of these terms by clever organization of the additions such that we save area and

optimize for speed. The method of deferred parallel carry addition of partial products can now be applied to achieve this.

Approach

The approach to be followed is as below:

- 1.) The prior work considers the case when the number of hardwired multipliers are available in abundance and hence the multiplication is a highly parallel operation. A paper-and-pencil analysis of FPGA peak floating-point performance [9] clearly shows that DSP blocks are a relatively scarce resource when one wants to use them for accelerating multiplications. Hence when considering the development of the approach the limited number of multipliers would be considered and hence a multi-cycle architecture will be proposed.
- 2.) The set of equations would be developed to aid a designer to use the asymmetric embedded multipliers to implement large multiplication operation on FPGAs using the DSP blocks and the Xilinx Virtex-5 and 6 would be targeted.
- 3.) An example implementation would be demonstrated and proved via FPGA implementation on a specific FPGA device belonging to the Xilinx Virtex-5 family and the analysis of resource usage, speed of operation, area occupied and power consumption would be carried out.

Expected Results

The set of equations which would be developed would be useful for a designer to efficiently use the DSP embedded blocks to implement large multiplications. The systematic design approach is targeted to result in an timing and area efficient implementation.

References

- [1] Walters, E. III, Arnold, M.G., and Schulte, M.J.: "Using truncated multipliers in DCT and IDCT hardware accelerators". Proc. SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, San Diego, California, August 2003, pp. 573–584.
- [2] Sheu, M.-H., and Lin, S.-H.: "Fast compensative design approach of the approximate squaring function", IEEE J. Solid State Circuits, 2002, 37, (1), pp. 95–97.
- [3] Stallings, W.: "Cryptography and network security: principles and practice" (Prentice-Hall, 2003, 3rd edn.), ISBN: 0-13-091429-0.
- [4] S. Gao, N. Chabini, D. Al-Khalili and P. Langlois, "Optimized realizations of large integer multipliers and squarers using embedded blocks", IET Comput. Digit. Tech., 2007, 1, (1), pp. 9–16.
- [5] Gao, S., Chabini, N., Al-Khalili, D., and Langlois, P.: "Optimized multipliers for large unsigned integers". Proc. 23rd IEEE NORCHIP Conf., Oulu, Finland, November 2005, pp. 78–81.
- [6] Florent de Dinechin and Bogdan Pasca. "Large multipliers with less DSP blocks". Technical Report 2009-03, LIP, École Normale Supérieure de Lyon, 2009.
- [7] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms* (MIT Press, 2000).

[8] A. Karatsuba and Yu. Ofman (1962). "Multiplication of Many-Digital Numbers by Automatic Computers". *Proceedings of the USSR Academy of Sciences* 145: 293–294.

[9] D. Strenski. FPGA floating point performance – a pencil and paper evaluation. *HPCWire*, Jan. 2007.