

Design Exploration of 192-bit Elliptic Curve Adder On The StarBridge HC-36 System

Gang Quan, Duncan A. Buell, James P. Davis, Siddaveerasharan Devarkal

Department of CSE

University of South Carolina

Columbia, SC 29208

{gquan, dbuell, jdavis, devarkal}@cse.sc.edu

Abstract

We study the problem of implementing 192-bit elliptic curve adder on StarBridge HC-36 high performance computing system. In this paper, we propose a hierarchical hybrid model for the multipliers, the primary functional units for the elliptic curve adder, to provide finer grain performance/resource usage tradeoffs during the design space exploration. The timing and area cost for the general hybrid multiplier is analytically formulated by exploring the regularity of its internal structure, and then adjusted empirically. Experimental results show that our approach can predict rapidly the timing and area cost for the hierarchical hybrid multiplier with a reasonable accuracy.

1 Introduction

The dramatic increasing of IC technology has enabled multiple million gate FPGAs to be integrated in the same board. The performance for the large and complex custom computations can be greatly improved by implementing them in the reconfigurable hardware process elements, without enduring the huge design cost for an ASIC. However, to fully take the advantage of these hardware platforms and achieve the projected goals, i.e., several order magnitude of speedup over the pure software implementation, these hardware resources must be utilized in an efficient way.

The primary goal for this study is to implement the 192-bit elliptic curve (ECC) point adder [3] on the reconfigurable high performance computing platform. Elliptic curve cryptosystem, emerged as a new generation public key cryptosystem, is particularly appealing for hardware implementation due to its smaller key size and highest strength per bit compared to other public key cryptosystems.

An elliptic curve can be written in homogeneous form as

$$Y^2 Z = X^3 + A X Z^2 + B Z^3$$

for constants A and B. National Institute of Standards and Technology (NIST) has promulgated the 192-, 224-, 256-, 384-, and 521-bit versions of the elliptic curves. The fundamental operation for the elliptic curve cryptography is the multiplication of a point $P = (x,y,z)$ on the curve of a point by a scalar. This multiplication can be done using the doublings and additions analogous to squaring and multiplication method for the computation of exponentiation. The addition of two points $P_1=(x_1,y_1,z_1)$ and $P_2=(x_2,y_2,z_2)$ on

the elliptic curve are graphically shown in Figure 1. A more detailed explanation on can be found elsewhere [3].

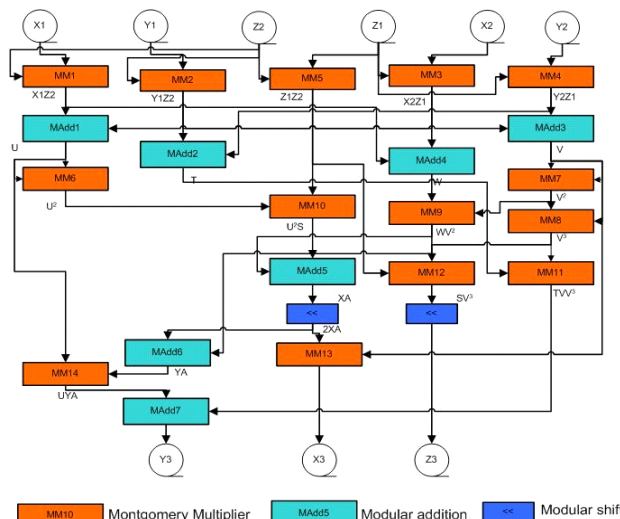


Figure 1 Data flow graph for elliptic curve point addition

The reconfigurable platform under our investigation is the HC-36m Hypercomputer [1] from Star Bridge Systems, Inc. This machine contains an array of seven Xilinx Virtex-II XC2V6000 FPGA devices, four of which can be customized as process elements.

Note that, in Figure 1, one elliptic curve addition requires as many as 14 high bit-width modular multiplications, or 42 high bit-width multi-precision multiplications if the Montgomery modular multiplication technique [4] is applied. As high-bit width multi-precision multiplications have predominated computation cost for adding two points on an elliptic curve, it is necessary to realize these multipliers in the reconfigurable hardware efficiently and effectively to ensure the overall high performance for the system. One major issue of realizing the 192-bit ECC adder on the HC-36m system has to do with making the area and speed tradeoff under the potentially heavy reuse of the FPGA resource. Compromises have to be made with other factors in minds, such as the inter chip communication costs, design primitives and libraries enforced by the manufacturers.

In this paper, we seek to uniformly formulate a large variety of high bit-width multipliers by exploring their internal hierarchical structures. With a uniform formulation of different multiplier architectures, the representations for

the tremendous number of design alternatives can be greatly simplified. Also, such a formulation helps to promote the sharing of the multipliers not only as a monolith devices but also sub computation units at different levels, and therefore, is capable of providing better chances for implementing complicated system under resource and timing constraints.

Moreover, by carefully analyzing the interval regularity of the hybrid multiplier structure, we are able to construct a single analytical formula to estimate the timing and area cost via a combined analytical and empirical approach. Experimental results that our approach can predict the timing/area cost for a give hybrid multiplier structure rapidly and with a reasonable accuracy. Our formulation for the multiplier and the related area/timing prediction can be readily incorporated into the published automatic design space exploration techniques, e.g. [2,5,6,7], and benefit almost every aspect of during the search process such as allocation, partitioning, mapping, and scheduling, which will lead to a much better design results than the traditional ad-hoc designs.

2 High bit-width multiplier implementation

2.1 Hierarchical v.s non-hierarchical

To realize the multipliers in the hardware, there are generally two strategies depending on the internal structure of the multipliers: *hierarchical* or *non-hierarchical*. A hierarchical multiplier recursively derives its product from the low bit-width multiplication results, and a non-hierarchical multiplier constructs the product directly from the inputs monolithically, using the techniques such as shift-add or the look-up table.

The shift-add non-hierarchical multiplier [] has the advantage of low resource requirement, i.e., an ALU, a shift register, and related control circuit, but the computation speed is in the order of $O(n)$, where n is the bit-width of the multiplicands. The speed can be painfully slow for high bit-width multiplications. On the other hand, the look-up-table based multiplier has a much better the computation performance for the low bit-width multiplier. However, the excessive resources requirement makes it impossible or impractical for high bit-width such as 192-, 256- or even 512-bit multipliers.

2.2 Different hierarchical styles

The hierarchical design approach can better manage the design complexity by exploring the internal regularity within the structure. There are different schemes commonly used in the hierarchical multiplier design.

One simple strategy is the naïve divide-and-conquer approach. According to this approach, assume that A and B are n-bit numbers and can be evenly divided into two halves, i.e., A_H and A_L , B_H and B_L , respectively. Let

$$a_0 = A_H \times B_H, a_1 = A_H \times B_L, a_2 = A_L \times B_H, a_3 = A_L \times B_L,$$

Then

$$A \times B = 2^n a_0 + 2^{n/2} (a_1 + a_2) + a_3. \quad (1)$$

Note that, to compute the n-bit multiplication, this approach needs to conduct four sub (n/2-bit) multiplication operations.

A better divide-and-conquer algorithm, i.e., Karatsuba-Ofman Algorithm (KOA) [], reduces the number of sub multipliers by replacing the multiplication operations with several additions. That is, let

$$a_0 = A_H \times B_H, a_1 = (A_H + A_L) \times (B_H + B_L), a_2 = A_L \times B_L.$$

Then, the product can be computed as

$$A \times B = 2^n a_0 + 2^{n/2} (a_2 - a_1 - a_0) + a_2, \quad (2)$$

and only three n/2-bit multiplications are necessary.

To best exploit the parallelism in the hardware implementation of multiplier, the naïve divide-and-conquer requires four sub-multipliers, while KOA requires three. In the case when fewer sub-multipliers can be synthesized due to the limited hardware resource such as the number of the FPGA slices, the performance of the multiplication can be seriously degraded. On the other hand, when abundant hardware resource is available, more sub multipliers cannot increment the computation efficiency of the multiplications. Even though the basic 2-way divide-and-conquer method can, theoretically, be easily extended for m-way partitioning, the practical control overhead will likely outweigh the potential performance benefits. It is desirable that a hierarchical approach that can explore the internal regularity with the available resource in mind. The so called “broadcast multiplier”, proposed by Duncan, *et al* [6], can well serve for this purpose.

The broadcast multiplier uses essentially the m-way partitioning strategy. Different from the m-way divide-and-conquer, it exploits only certain part of the parallelism based on the available resources explained as follows.

Let A and B are n-bit integers. Assume that the available resources can be used to realize k multipliers with each of which can compute the p -bit multiplication, where

$$p = \left\lceil \frac{n}{k} \right\rceil. \quad (3)$$

We can correspondingly partition A and B such that

$$A = \{A_{(k-1)} \wedge A_1 A_0\}, B = \{B_{(k-1)} \wedge B_1 B_0\},$$

and $|A_i| = |B_i| = p$. Then we have

$$A \times B = \sum_{i=0}^{k-1} ((A_{(k-1)} \wedge A_1 A_0) \times B_i \gg (i \times p)). \quad (4)$$

From (4), one can tell that the broadcast multipliers are in fact *block-based* shift-add multipliers. It contains k

iterations of partial product generations and accumulations. The computation efficiency of the broadcast multiplier come from the fact that, with k sub-multipliers, the generation of sub-products, *i.e.*, k sub multiplications, in one iteration can be computed in parallel. When $p=1$, the multiplier reduces to the shift-add multiplier; while when $p=n$, the multiplier is simply a monolith multiplier.

Compared with the divide-and-conquer scheme, broadcast multiplier has the advantage that it is more flexible in terms of determination of the bit-width and numbers of sub-multipliers to be implemented in the hardware, and therefore to achieve better resource usage. However, since it can only make use of partial parallelism in the multiplication, it cannot achieve the ideal computation performance as that by the divide-and-conquer schemes.

As both the divide-and-conquer and broadcast multiplication schemes are both hierarchical. That is, they can construct the final product from sub products. It is reasonable that we can take the advantages of both these techniques by combining them in the implementation of the multiplier at different recursive levels. We call this type of multiplier as the *hybrid multiplier*. By incorporating these two types of strategies differently, we actually provide a finer grain of computation efficiency and resource usage efficiency tradeoffs. This is particular crucial for the design exploration for large system under serious resource consideration as that in our case.

2.3 The hybrid multiplier

A hybrid multiplier is a hierarchical multiplier that can combines different hierarchical strategies at different recursive levels. In our study, as explained before, we allow only two type hierarchical multiplier structure, *i.e.*, KOA and broadcast, in the design, since the variations of combinations of these two techniques can representatively provide a relative large spectrum of multipliers with different area/speed characteristics. Also, to take advantage of the high performance and resource efficiency of the built-in 18x18 hardware multipliers in the Xilinx Virtex II 6000 chips, that is, the PEs in the HC-36 systems, we assume all the multiplications with bit-width less than 18 bit are conducted with these hardware multipliers.

To differentiate the different compositions of the hybrid multipliers, we use an integer list that consists of N elements, with $N+1$ the total number of recursive levels. Specifically, for a n -bit hybrid multiplier Γ_n , we have $\Gamma_n = \{m_1, m_2, \dots, m_N\}$, where $m_i > 0$. m_i represents the implementation strategy at the i th recursive level. Specially, if $m_i = 1$, it denotes that the KOA scheme is applied at the i th level; and if $m_i = k > 1$, the broadcast technique with k multiplication units is used at the i th level. Recall that, for the lowest level, we always use the 18x18 bit hardware multipliers.

For example, an 192-bit hybrid multiplier, $\Gamma_{192} = \{1, 1, 3\}$, has four abstraction levels. At the first level, KOA scheme is adopted which requires three 96-bit multipliers. For each of the 96-bit multipliers (the 2nd level), the KOA

scheme is adopted again requiring three 48-bit multipliers each. For each of the 48-bit multipliers (the 3rd level), the broadcast scheme with three 16-bit multipliers is used. The hardware multipliers built in Virtex II 6000 are used for the 16-bit multiplications

3 Area and timing estimation for the hybrid multipliers

A rapid and accurate area and timing estimation of the hybrid multiplier is critical in the design space exploration since the design space is usually tremendously large. For example, even under very conservative estimation, the search space for our problem can be as high as 2^{120} . Fortunately, due to the regular structure of the hybrid multiplier, the analytic estimation can be reasonably close to the actual results.

The area and timing cost for a hybrid multiplier can be computed recursively by analyzing the basic units, *i.e.*, the KOA and the broadcast multiplier. Specifically, from equation (2) and (4), we can derive the area cost for the hybrid multiplier, $\Gamma_n = \{m_1, m_2, \dots, m_N\}$, at the i th level as follows:

$$S_i(n) = \begin{cases} 3S_{(i+1)}\left[\frac{n}{2}\right] + SO_{KOA}(n), & \text{if } m_i = 1 \\ kS_{(i+1)}\left(k\left[\frac{n}{k}\right]\right) + SO_{BC}(n), & \text{if } m_i = k > 1 \end{cases} \quad (5)$$

In equation (5), $S_i(n)$ represents the area cost for n -bit multiplier, $SO_{KOA}(N)$ represents the area cost for the adders and control overhead for the KOA implementation of an n -bit multiplier, and $SO_{BC}(n)$ is the area cost for the adders and control in a broadcast implementation of n -bit multiplier. It is reasonable to assume that $SO_{KOA}(n)$ and $SO_{BC}(N)$ are linear to n . That is, $SO_{KOA}(n) = \alpha N$, and $SO_{BC}(n) = \beta N$. With several experiments and linear regression, we find that $\alpha = 15$, and $\beta = 11$.

The timing cost estimation for the hybrid multiplier can also be derived in the similar manner. Specifically, we have

$$T_i(n) = \begin{cases} T_{(i+1)}\left[\frac{n}{2}\right] + 4T_{add}(n+2) + TO_{D\&Q}(n), & \text{if } m_i = 1 \\ k\left[T_{(i+1)}\left[\frac{n}{k}\right] + 2T_{add}\left(n + \left[\frac{n}{k}\right]\right) + TC_{BC}(k)\right] + T_{BC}(k), & \text{if } m_i = k > 1 \end{cases} \quad (6)$$

where $T_i(n)$ denotes the cycle numbers for n -bit multiplier, and $T_{add}(n)$ denotes the cycle numbers required conducting n -bit addition. $TO_{KOA}(n)$ is the timing cost for the control in n -bit KOA multiplier. For the broadcast implementation, as shown in (4), we have two types of control overhead: one is the overhead within each iteration, and the other is loop overhead that carries from one iteration to another. In equation (6), they are represented as $TC_{BC}(k)$ and $T_{BC}(k)$, respectively.

Since all our design on the HC-36 systems are carried out within the design environment using the design objects

enforced by the manufacture, we are able to profile the timing parameters for the related hardware modules identify the exact number of the overhead. Through our study, we find that, for $n < 192$, we have $TO_{KOA}(n) = 3$, $TC_{BC}(k) = 2$, and $T_{BC}(k) = k$.

4 Experimental Results

In this section, we use experiments to evaluation the accuracy of our analytical estimation results.

In our experiments, we randomly choose five different compositions of 192-bit hybrid multipliers. We first estimate the slice number as well as the number of cycles for each hybrid multiplier according to equation (5) and (6). Then we design and synthesize these multipliers on the Starbridge HC-36 system to get the actual area and timing cost. The results are list in the following table.

Table 1 Comparison of the analytical estimation results and the practical implementation results for five hybrid multipliers

Γ_{192}	Area (slice numbers)		Time (cycle numbers)	
	Estimated	Actual	Estimated	Actual
{1 6}	6564	6587	43	43
{6 1}	5508	5714	78	78
{1 1 3}	12630	12140	32	32
{1 3 1}	11046	11106	46	46
{3 1 1}	9990	10257	60	60

From Table 1, we can conclude that, due to the high regularity of the hybrid multiplier, our analytical methods can estimate the area/timing cost with a reasonable accuracy. For the area cost, the relative deviations of the estimate results to the actual results are no more than 5%. Table 1 also shows that our analytical method can precisely capture the timing cycles for the five multipliers. This is because, by profiling the design objects in the given manufacture's design library, we are able to capture the exact control overhead in equation (6) and therefore precisely estimate the execution cycles for the multipliers. Overall, our experiments show that our hybrid multiplier model and related analytical area/performance estimations can be effectively used in automatic design exploration for implementing the 192-bit elliptic adder on the HC-36 system.

5 Conclusions and future work

In this paper, we propose a hierarchical hybrid multiplier structure in the design for the 192-bit elliptic curve point adder on the Starbridge high performance platform, i.e, HC-36 system. Through the exploration of the internal regularity of the hybrid multiplier, we present an analytical estimation method which can rapidly predict the area/performance with a reasonable accuracy. Experiments show that our hybrid and related estimation techniques can be effectively used in the

automatic design space exploration to search for high performance design solutions under the given platform and resource constraints.

We are currently incorporating our hybrid multiplier and the evaluation techniques in the automatic design space exploration for implementing the high performance 192-bit elliptic curve adder on the HC-36 system. With this practical example, we would like to identify the inadequacies and limitations of the theoretical design space techniques in the previous researches. Finally, by implementing the application on Starbridge HC-36 system, we intend to investigate the potential and limitation that multi-FPGA system may have for the high performance computing requirements.

References

- [1] Star bridge high performance computers.<http://www.starbridgesystems.com>.
- [2] B.Dave, G.Lakshminarayana, and N.K.Jha. Cosyn:hardware-software co-synthesis of embedded systems. In *Proc. Design Automation Conference*, pages 703–798, 1997.
- [3] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, Cambridge, UK, 1999.
- [4] D. A. Buell, J. Davis, and G. Quan. Reconfigurable computing applied to problems in communications security. *Military and Aerospace Applications of Programmable Logic Devices Conference*, 2002.
- [5] J. Henkel and R. Ernst. A hardware/software partitioner using a dynamically determined granularity. In *IEEE International Conference on Design Automation Conference*, pages 691–696, June 1997.
- [6] D.E. Knuth, *The Art of Computer Programming, Volumn 2: Seminumerical Algorithms*, Addison-Wesley, 1998
- [7] P.L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519-521, 1985
- [8] NIST. Recommended elliptic curves for federal government use. <http://csrc.nist.gov/csrc/fedstandards.html>, July 1999.
- [9] D.A Patterson and J.L.Hennessy, *Computer Organization & Design: The Hardware/Software Interface*, Morgan Kaufmann, 1997
- [10] G. Quan, X. Hu, and G. Greenwood. Performance-driven hierarchical hardware/software partitioning. In *International Conference On Computer Design*, pages 652–657, 1999.
- [11] R.P.Dick and J.K.Jha. Mogac:a multiobjective genetic algorithm for the co-synthesis of hardware-software embedded systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(10):920–935, October 1998.