An Efficient Implementation of Multipliers for ASIC Implementations

[1]K.Sucharitha,[2]P. Rahul Reddy,
[1]*Assistant Professor, Dept of ECE, Ramanandatirtha Engineering College, Nalgonda, India*
[2] *Associate Professor, Dept of ECE, Swami Ramananda Tirtha Institute of Science & Technology, Nalgonda, India.*

Abstract:
Nowadays modular multiplication of long integers is a significant building block for cryptographic algorithms. Even though numerous FPGA accelerators have been proposed for large modular multiplication, earlier systems have been based on basically on O (N^2) algorithms. In this paper, we present a Montgomery multiplier that includes the more effective Karatsuba algorithm which is O ($N^{(\log 3/ \log 2)}$). This system is parameterizable to different bit widths and makes exceptional use of both embedded multipliers and fine-grained logic. The design has expressively lower LUT-delay product and multiplier-delay product related with former designs.

Keywords:*Cryptograph, FPGA, Galois field, Karatsuba Multiplier*

I. INTRODUCTION

Multiplication is the most time consuming process in various signal processing operations like Convolution, Circular Convolution, Cross Correlation, and autocorrelation, Image processing applications such as edge detection, microprocessors arithmetic and logical units etc. The performance of microprocessor is determined by performance of the multiplier. Multiplier operation is depends on the speed of adder. Speed of adder is affected to its path propagation delay. Thus the multiplier is usually slowest and area consuming element in the system. Our designs offer tradeoffs between Computational time area, latency and throughput for performing multiplication. Soto increase the speed of multiplier, it is requiring improving the speed of addition. In this approach it is required to find the longest critical paths in the multi-bit adders and then shortening the path to reduce the total critical path delay. A finite field or Galois field is a field that contains only finitely many elements. The finite fields are classified by size. This classification specifies the order of the field. Notations for the finite fields are GF (pm) or Fpm, where the letters ”GF” stand for “Galois field”. Modern cryptography intersects the disciplines of mathematics, computer science and engineering. In these applications, multiplication is the most common arithmetic. Thus it is desirable to design hardware efficient multiplier for GF (2m) to meet the real time requirement with minimum hardware complexity. There are three popular types of bases over finite fields: polynomial basis (PB), normal basis (NB) and dual basis(DB). Basis is a set of vectors that, in a linear combination, can represent every vector in a given vector space. Polynomialbasis is a mathematical function that is the sum of a number of terms. Normal basis in field theory is a special k ind of basis for Galois extensions of finite degree, characterized as aforming a single orbit for the Galois group. Dual basis is a setof vectors that forms a basis for the dual space of a vector space. One advantage of the normal basis is that the squaring of an element is computed by a cyclic shift of the binary representation. The dual basis multipliers require less chip area than other two types. Modular multiplication consists of two steps: first aclassical multiplication and then a modular reduction. The straight forward multiplier is used to get speed efficient design while a Karatsuba multiplier is used to get an area efficient design. Merits are reduced hardware complexity and high throughput.

II. LITERATURE SURVEY

C.Grabbe, M.Bednara,J.Teich, [1] presented four high performance GF (2233) multipliers for an FPGA realization and analysed the time and area complexities. The finite field elements are represented as polynomial basis and normal basis. In polynomial basis, classical multiplier and Karatsuba multipliers were designed. The advantage of classical multiplier is regular structure and pipelined operation. The disadvantage is high space complexity. In Karatsuba multiplier the advantage is less number of gates are required. The normal basis multipliers are Massey- Omura and Sunar-Koc multiplier. The advantage of Massey-Omura is high flexibility and Sunar-Koc is total number of gates are reduced. P.L.Montgomery, [2] presented Karatsuba Of man algorithm for multiplying two polynomials. Here multiplication of 5-term, 6-term and 7-term polynomials are provided with scalar multiplication of 13, 17 and 22.Using 6-term polynomial only leads to better asymptotic performance than standard karatsuba.

C.Paar, [3] presented a new bit parallel structure for a multiplier with low space complexity in Galois field is introduced. Finite field of GF(2n) is considered and field extension of GF((2n)m). The field elements are represented in the canonical base or in standard basis. Field of the form GF((2n) m) are

referred as composite field. Karatsuba Of man algorithm is used to multiply two polynomials effectively. Advantages are complexity is reduced by introducing the composite field. The main disadvantage is security is less and does not have a regular structure.

C.Rebeirno and D.Mukhopadhyay, [4] presented a hybridtechnique which has a better area delay product. Masking strategies are introduced to prevent power based side channel attacks on the multiplier. SCAs are the biggest threat to modern cryptography systems. In basic recursive KM, the number LUTs required to combine the partial products ismuch lower but it cannot applied directly to ECC. The hybrid KM requires least resources as compared to other KMs used for elliptic curve arithmetic; also it has a unique architecture. Demerits are it is not efficient for FPGA platform as the number of utilized LUTs is 65%.

A.Reyhani-Masoleh and A.Hasan, [5] presented a new bit parallel structure for the polynomial basis multiplication which is applicable to all type of irreducible binary polynomial. The main advantage of this new formulation is that it can be used with any field defining irreducible polynomial. Then a bit parallel hardware architecture generalization is provided. The architecture consists of two parts IP network and Q network. The space and time complexities are analysed as a function of reduction matrix Q. the main advantage is only fewer number of lines are require don the bus.

III. PROPOSED ALGORITHM

The basic to our Montgomery multiplier design is there cursive Karatsuba algorithm. This permits us to calculate modular multiplication with a complexity approaching $O(N$ (log 3/ log 2)). Background on the Karatsuba algorithmis provided in Section A. Our design uses multiple-precision arithmetic techniques so that the critical path delay is independent of the multiplier's bit-width. Except stated otherwise, we assume we are multiplying two $2k$-bit unsigned integers and the limb-widths of all components are w. The number of limbs in a $2k$-bit word is $(2s = [2k/w])$. We use either a coarse-grained carry-save technique or a pipelined multiple-precision technique in all of our adders and subtractor. The critical path of the circuit primarily depends upon the limb-width w.

A. *Karatsuba Algorithm*

The Karatsuba multiplication algorithm was proposed by Karatsuba and Of man in 1962 [13].To illustrate the algorithm, we let X and Y be two 2k-bit unsigned integers and split them both in half.

$X = 2^k X_1 + X_0$ and $Y = 2^k Y_1 + Y_0$

In conventional long multiplication, the product XY is computed with four k-bit multiplications and three additions as shown Equation 1.

$XY = 2^{2k} z2 + 2^k z1 + z_0$

$z_2 = X_1Y_1$, $z_1 = X_0Y_1 + X_1Y_0$, $z_0 = X_0Y_0$…….. (1)

As shown in Equation 2, Karatsuba noticed that the middle term z_1 can be computed reusing the terms z_2 and z_0. Reusing these terms allow us to replace two of the multiplication sand one addition with four additions/subtractions and one multiplication.

$z_1 = X_0Y_1 + X_1Y_0$

$= X_1Y_1 + X_0Y_0 - (X_1 - X_0)(Y_1 - Y_0)$

$= z_2 + z_0 - (X_1 - X_0)(Y_1 - Y_0)$

$XY = T_1 - T_2$

$T_1 = 2^{2k} z_2 + z_0 + 2^k (z_2 + z_0)$

$T_2 = 2^k (X_1 - X_0)(Y_1 - Y_0)$ …….. (2)

B. *High-Level Architecture*

We designed a parameterizable Karatsuba layer and apply it recursively. In our prototype architecture, we achieve high performance by fully parallelizing and pipelining all Karatsubalayers.

We also optimized the original Karatsuba algorithm to suit our hardware implementation. First, instead of computing the product of two 2's complement numbers (T_2) in Equation 2,we compute the absolute value of $(X_1 - X_0)$ and $(Y_1 - Y_0)$.These absolute values are then added to or subtracted from thefirst term (T_1). Second, we take advantage of the fact that thesum (lower half z_2 + upper half z_0) actually appears twice andrearrange T_1as Equation 3. This allows us to remove somelogic required for the addition.

$T_1 = z_{00} + (2^{2k} + 2^k)(z_{01} + z_{20})$

$+ 2^k z_{00} + 2^{2k} z_{20} + 2^{3k} z_{21}$

$z_0 = 2^k z_{01} + z_{00}$ ..... (3)

Figure 1 shows the block diagram of a single level of our recursive Karatsuba multiplier for 2k-bit inputs. It is constructed from three k-bit Karatsuba multipliers (or embedded multipliers if k is equal to or smaller than the native multiplier bit-width).

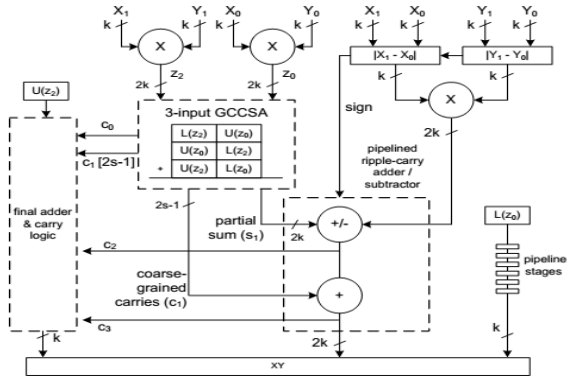

Fig. 1. Block Diagram of Recursive Karatsuba Multiplier.

We use a coarse-grained carry-save adder(CGCSA) for the first three additions in the middle 2k-bits.Pipelined multiple-precision adders are used in the 3-inputadder, the final adder logic and the absolute value units. They are carefully pipelined to minimize latency while achieving high clock frequency.

C. Detailed Implementation

We indicate (d = $L(z_2)$,$U(z_0)$) as the first input to the coarsegrained carry-save adder, (e = $U(z_0)$,$L(z_2)$) as the secondand (f = $U(z_2)$,$L(z_0)$) as the third. Figure 2 shows a block diagram of an example 3-input CGCSA with 4 limbs. In the first cycle, the lower half of inputs d and e are summed limb wise, producing a k-bit partial sum and an s-bit coarse-grained carry. These sums and carries are then duplicated and added to a registered version of input f. The 3-input adder generatesa 2k-bit partial sums (s), coarse-grained carries (c_1[2s− 2...0])and two MSB carries (c_0 and c_1[2s − 1]). Shown in Figure 1,the partial sum and coarse-grained carries are fed into the 3-input multiple-precision adder/subtracter, while the two MSB carries are pipelined and fed into the final adder.

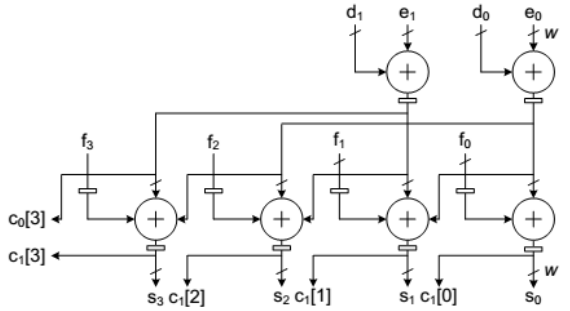

Fig. 2. An Example 3-Input Coarse Grained Carry-Save Adder with 4 Limbs.

As shown in Figure 1, the 3-input pipelined ripple-carry adder / subtracter is used to combine the partial sum and coarse-grained carries from the CGCSA and add or subtract the third partial product (X_1−X_0) (Y_1−Y_0), depending on the signs from the absolute value units. Figure 3 shows an example3-input adder/ subtracter with 4 limbs. g, s, and C1[i] represent the partial product from the third multiplier, and the partialsum and coarse-grained carries from the GCCSA, respectively. The absolute value units and the final adder have a similar construction as the ripple-carry adder. A special circuit is used to reduce the four 1-bit carries to a 2-bits sum before it is fed into a k-bit adder.

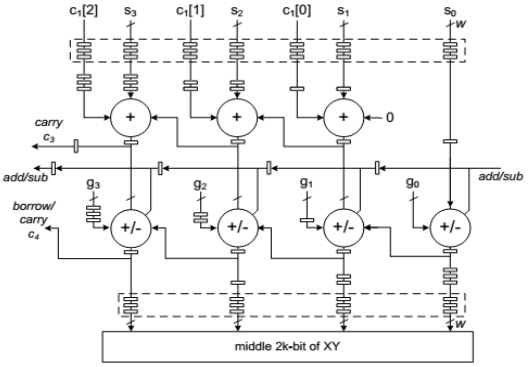

Fig. 3. Multiple-Precision 3-Input Adder / Subtracter.

Although our recursive Karatsuba multiplier provides integer multiplication with low complexity, its input to output latency grows quickly with the number of bits. To solve this problem, we propose a batch-pipelined architecture that can hide the long latency with data level parallelism. A universal data path is constructed using the Karatsuba multiplier, ripple adder / subtracter and shifter to compute step 1, 2, (3 and 4)of Algorithm 1 in three passes.

IV. RESULT AND DISCUSSION

Figure 4 shows the resource utilization of batch-pipelined Montgomery multipliers with different bitwidths. We found the designs' clock frequencies are lower than the theoretical 400 MHz limit. A major contributor to this is likely increased routing delay in large designs as the average separation between components increases. We could reduce this routing delay by inserting additional pipelining registers.

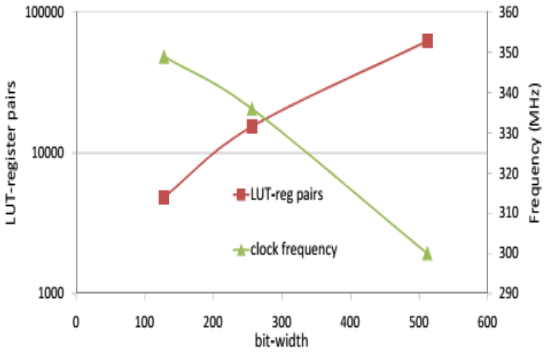

Fig. 4. Result of our Batch-Pipelined Montgomery Multipliers.

V. CONCLUSION

In this paper, we offered a Karatsuba-based Montgomery multiplier for cryptography applications using long integers. The multipliers have expressively lower area-delay products related with earlier designs. They also be responsible for exceptional performance and energy efficiency related with software implementations.

REFERENCES

[1] C.Grabbe,M.Bednara,J.Teich,J.von zur Gathen, and J.Shokrollahi," FPGA designs of parallel high performanceGF(2233) multipliers," in Proc.Int.Symp.Circuits Syst.(ISCAS),May 2003,pp.268-271.

[2] P.L.Montgomery," Five, six, and seven-term Karatsuba like formulae," IEEE Trans.Comput., vol.54, no.3, pp.362-369, Mar.2005.

[3] C.Paar, A new architecture for a parallel finite field multiplier with low complexity based on composite fields,IEEE Trans.Comput.,vol.45,no.7,pp.856-861,1996.

[4] C.Rebeiro and D.Mukhopadhyay," Power attack resistantefficient FPGA architecture for Karatsuba multiplier," in Proc.Int.Conf.VLSI Des.,2008,pp.706-711 N.S.Kim, T.Mudge, and R.Brown, "A 2.3 Gb/s fully integrated and synthesizable AES rinjdael core," in proc. IEEE Custom Integrated Circuits Conf., 2003, pp.193-196.

[5] A.Reyhani-Masoleh and A.Hasan,"Low complexity bit parallel architecture for polynomial basis multiplication over GF (2m)," IEEE Trans.Comput., vol.53, no.8, pp.945- 995, Aug.2004.

[6] A. Daly and W. Marnane, "Efficient architectures for implementing Montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic," in Proc. FPGA. New York, NY, USA: ACM, 2002, pp. 40–49.

[7] D. Narh Amanor, C. Paar, J. Pelzl, V. Bunimov, and M. Schimmler, "Efficient hardware architectures for modular multiplication on FPGAs," in Proc. FPL, Aug. 2005, pp. 539 – 542.

[8] S. Ors, L. Batina, B. Preneel, and J. Vandewalle, "Hardware implementation of a Montgomery modular multiplier in a systolic array," in in Proc. International Parallel and Distributed Processing Symposium, April 2003, p. 8 pp.