International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 01
January 2017

# High-Throughput Finite Field Multipliers Using Redundant Basis for FPGA and ASIC Implementations

MS. L. SUREKHA & Miss. C. NIRMALA

**Abstract**: *Redundant basis (RB) multipliers over Galois Field GF($2^m$) have gained huge popularity in elliptic curve cryptography (ECC) mainly because of their negligible hardware cost for squaring andmodular reduction. In this paper, we have proposed a novel recursive decomposition algorithm for RB multiplication to obtain high-throughput digit-serial implementation. Through efficient projection of signal-flow graph (SFG) of the proposed algorithm, a highly regular processor-space flow-graph (PSFG) is derived.By identifying suitable cut-sets, we have modified the PSFG suitably and performed efficient feed-forward cut-set retiming to derive three novel multipliers which not only involve significantly less time-complexity than the existing ones but also require less area and less power consumption compared with the others. Both theoretical analysis and synthesis results confirm the efficiency of proposed multipliers over the existing ones. The synthesis results for field programmable gate array (FPGA) and application specific integrated circuit (ASIC) realization of the proposed designs and competing existing designs are compared. It is shown that the proposed high-throughput structures are the best among the corresponding designs, for FPGA and ASIC implementation. It is shown that the proposed designs can achieve up to 94% and 60% savings of area-delay-power product (ADPP) on FPGA and ASIC implementation over the best of the existing designs, respectively.*

## 1. INTRODUCTION

FINITE FIELD multiplication over Galois Field (( GF($2^m$ ) is a basic operation frequently encountered inmodern cryptographic systems such as the elliptic curve cryptography (ECC) and error control coding. Moreover, multiplication over a finite field can be used further to perform other field operations, e.g., division, exponentiation, and inversion. Multiplication over GF($2^m$ ) can be implemented on a general purpose machine, but it is expensive to use a general purpose machine to implement cryptographic systems in cost-sensitive consumer products. Besides, a low-end microprocessor cannot meet the real-time requirement of different applications since word-length of these processors is too small compared with the order of typical finite fields used in cryptographic systems. Most of the real-time applications, therefore, need hardware implementation of finite field arithmetic operations for the benefits like low-cost and high-throughput rate.

The choice of basis to represent field elements, namely the polynomial basis, normal basis, triangular basis and redundant basis (RB) has a major impact on the performance of the arithmetic circuits. The multipliers based on RB have gained significant attention in recent years due to their several advantages. Not only do they offer free squaring, as normal basis does, but also involve lower computational complexity and can be implemented in highly regular computing structures.

Several digit-level serial/parallel structures for RB multiplier Over GF($2^m$) have been reported in the last years after its introduction by Wu et al. An efficient serial/parallel multiplier using redundant representation has been presented. A bit-serial word-parallel (BSWP) architecture for RB multiplier has been reported by Namin et al. Several other RB multipliers also have been developed by the same authors for reducing the complexity of implementation and for high-speed realization. We find that the hardware utilization efficiency and throughput of existing structures can be improved by efficient design of algorithm and architecture.

In this paper, we aim at presenting efficient digit-level serial/parallel designs for high-throughput finite field multiplication over GF($2^m$)based on RB. We have proposed an efficient recursive decomposition scheme for digit-level RB multiplication, and based on that we have derived parallel algorithms for high throughput digit-serial multiplication. We have mapped the algorithm to three different high-speed architectures by mapping the parallel algorithm to a regular 2-dimensional signal-flow graph (SFG) array, followed by suitable projection of SFG to 1-dimensional processor-space flow graph (PSFG), and the choice of feed-forward cut-set to enhance the throughput rate. Our proposed digit-serial multipliers involve significantly less area-time-power complexities than the corresponding existing designs. Field programmable gate array (FPGA) has evolved as a mainstream dedicated computing platform. FPGAs however do not have abundant number of registers to be used in the multiplier. Therefore, we have modified the proposed algorithm and architecture for reduction of register-complexity particularly for the implementation of RB multipliers on FPGA platform. Apart from these we also present a low critical-path digit-serial RB multiplier for very high throughput applications.

## 2 LITERATURE SURVEY

### 2.1 BRIEF STUDY: MULTIPLERS

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets

1. High speed,
2. Low power consumption,
3. Regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed,
4. Low power and compact VLSI implementation.

The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand "serial-parallel" multipliers compromise speed to achieve better performance for area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics. AND gates are used to generate the Partial Products (PP). If the multiplicand is N-bits and the Multiplier is M-bits then there is N* M partial product.

#### 2.1.1 HISTORY OF MULTIPLIERS

The early computer systems had what are known as Multiply and Accumulate units to perform multiplication between two binary unsigned numbers. The Multiply and Accumulate unit was the simplest implementation of a multiplier. The basic block diagram of such a system is given below.

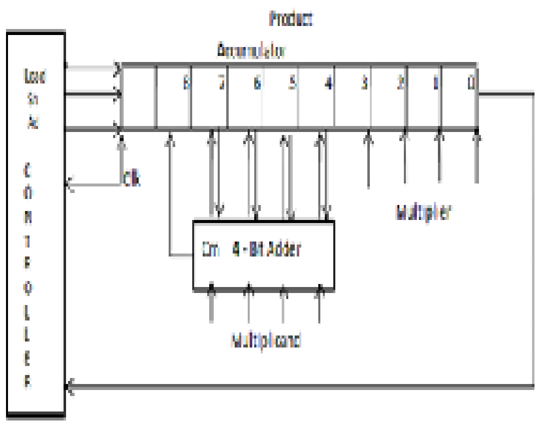

Fig.2.1 Multiplier Block Diagram

### 2.1.2 IMPLEMENTATION

The MAC unit requires a 4-bit multiplicand register, 4-bit multiplier register, a 4-bit full adder and an 8-bit accumulator to hold the product. In the figure above the product register holds the 8-bit result. In a typical binary multiplication, based on the multiplier bit being processed, either zero or the multiplicand is shifted and then added.

Following the same process would require an 8-bit adder. Instead, in the above design the contents of the product register are shifted right by one position and the multiplicand is added 5 to the contents. This multiply and accumulate block is also known by the name serial-parallel multiplier as the multiplier bits are processed serially but the addition takes place in parallel. The second type of multiplier is the parallel array multiplier.

The desire to speed up the rate at which the output is generated resulted in the development of this category of multiplier. In a serial-parallel multiplier discussed above, it takes one clock cycle to process one bit of the data input at any given time. Therefore, when working on an N-bit input it would take at least N clock cycles to generate the final output. In a parallel array multiplier the result is obtained as soon as inputs are presented to the multiplier. This is mainly because of the use of AND array structure to compute the partial product terms. Once the partial product terms are generated the only delay in generating the output is contributed by the adders which sum the partial product terms

column wise to generate the result. The figure below represents a parallel array multiplier with N=8 bit inputs.

In Figure block A stands for an AND gate. Block AHA stands for AND GATE and HALF ADDER structure and AFA stands for AND GATE and FULL ADDER structure. FA stands for full adder. The partial product terms are added along the diagonal (as shown by the arrows along the diagonal) to generate the product bits P. The carry from each block is passed onto to the next column and this is shown by vertical arrows . The gate level representation of an AND gate, HALF ADDER and FULL ADDER is given below.

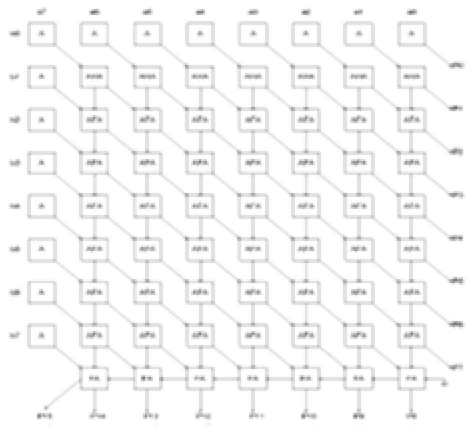

**Fig.2.2 Parallel array multiplier for N=8 bits.**

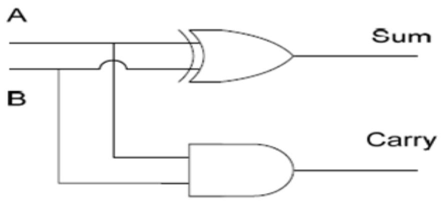

**Fig.2.3 Gate level implementation of a HALF ADDER.**

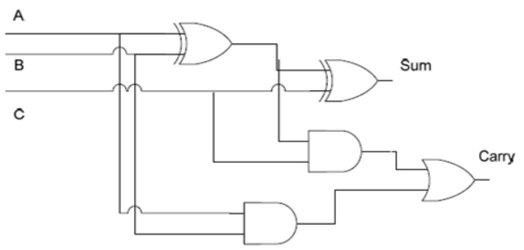

**Fig.2.4 Gate level implementation of a FULL ADDER.**

## 3.AREA-TIME-POWECOMPLEXITIES

### A. Complexities of PS-I, PS-II, and PS-III

PS-I requires P PPGUs, where each of the(P-1) regular PPGUs consists of XOR gates and AND gates. The finite field accumulator requires $Pn$ XOR gates and n bit-registers. The proposed design in total requires $Pn$ XOR gates, $Pn$ AND gates and $(Pn + 2n)$ bit-registers. After a latency of(P+Q) cycles, PS-I gives the desired output word in every Q cycles of duration $(T_A + T_X)$.

PS-I for any value of d consists of (P/d) PPGUs. The complete structure of the multiplier thus requires $Pn$XOR gates, $Pn$ AND gates and $(Pn/d + 2n)$ bit-registers. The latency of the structure amounts to $(P/d + Q)$ cycles, where the duration of minimum cycle period is

$\{T_A + (1 + \lceil \log_2 d \rceil) T_X\}$. PS-II has similar area-time complexities as those of PS-I except that it involves less registers and lower latency than the latter. In total, PS-II requires $(Pn+n)$ registers and yields its first result after a latency of $(\log_2 P + Q)$ cycles. For any value of , PS-II requires $\{Pn/d + n\}$ registers.PS-III requires(P+1) PPGUs, where each of the(P-2) regular PPGUs consists of n XOR gates, n AND gates and 2n bit-registers.The proposed design in totalwould require $Pn$ XOR gates and $Pn$ AND gates. Besides, it needs a total of $(2Pn+2n)$ bit-registers. After a latency of $(P+Q+1)$ cycles, PS-III gives the desired output word in every Q cycles of duration $T_X$ .

## B. Comparison with Existing Digit-Serial RB Multipliers

The area-time complexities of proposed structures and existing structures for RB multiplier are listed in Table I. For simplicity of discussion,we refer PS-I of Fig. 4 and PS-II of Fig. 6 as the case of d=1, respectively. In the authors have shown that their structures outperform the previous structures. Therefore we compare the performance of proposed structures only with those . PS-I and PS-II (for d=1), not only involve less time complexity (shorter ACT), but also have less XOR gates than those of existing designs. PS-I and PS-II (for $1 < d < P$) require less registers, at the cost of a small increase in critical-path. And PS-III has the lowest time-complexity among all the structures listed in Table I.

## C. Comparison with Existing Digit-Serial Multipliers Having a Type I ONB

The complexity of RB multiplier is almost the same as that of type I ONB [10]. The area-time complexities of the proposed multipliers and architectures (for which there exists a type I ONB) are shown in Table II. Note that these complexities are estimated by substituting for m+1, according to the definition.

The authors have shown that their multipliers outperform the previously proposed structures. Therefore we compare our proposed structures only with. PS-I and PS-II not only require less number of logic gates and registers ( number less XOR gates and nearly m number less registers), but also have shorter ACT compared to the structure .

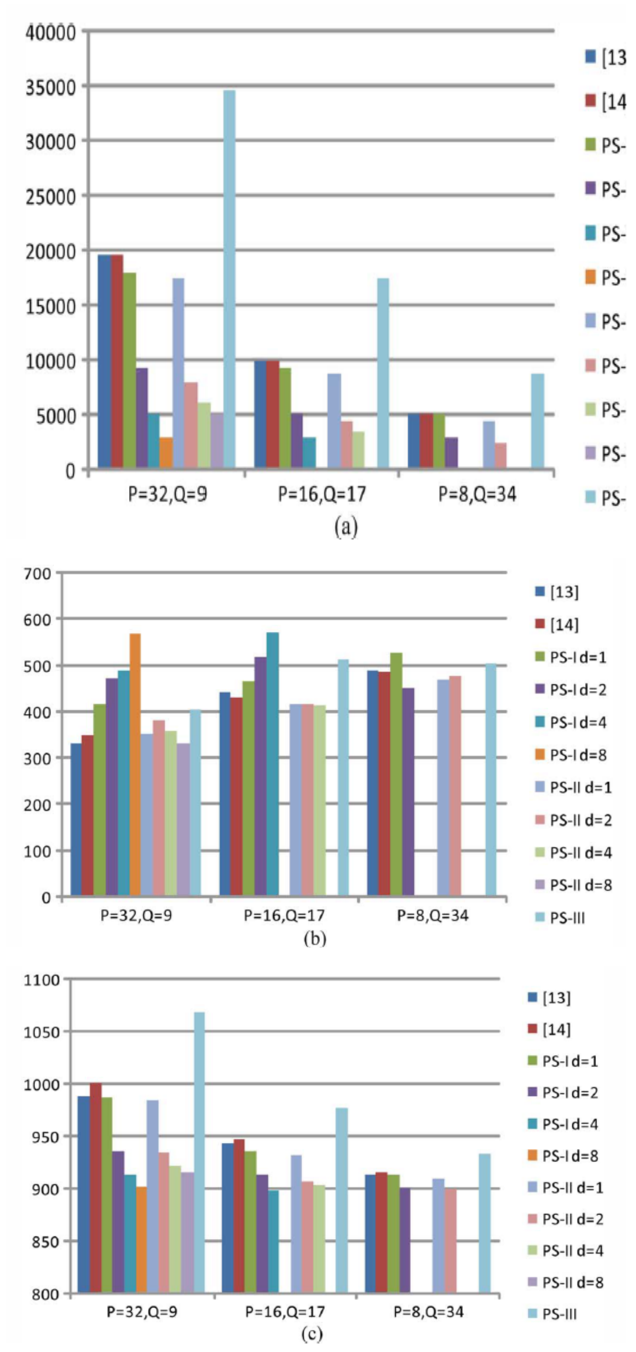

Fig. . Comparisons of keymetrics of various structures for . (a)Comparisons of area-complexity (number of ALUT). (b) Comparisons of maximum frequency (MHz). (c) Comparisons of power consumption (mW).

## D. Comparison of Synthesis Results for FPGA Implementation

We have used Altera Quartus II 12.0 and chosen Arria II GZ EP2AGZ225FF35C3 FPGA device to synthesize the proposed designs as well as the existing competing designs. The key synthesis results are obtained, in terms of area, maximum frequency and power consumption with respect to various , and . The number of adaptive look-up table (ALUT) is taken as the area measure. For fair estimation, we have used the same input data and the same clock frequency (100 MHz) to obtain the synthesis results using Quartus II PowerPlay Power Analyzer.

.

## 4 XILINX

## 4 .1XILINX ISE OVERVIEW

The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows you to take your design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes your design through the following steps in the ISE design flow.

### 4.1.1DESIGN ENTRY

Design entry is the first step in the ISE design flow. During design entry, you create your source files based on your design objectives. You can create your top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. You can use multiple formats for the lower-level source files in your design.

### 4.1.2 SYNTHESIS

After design entry and optional simulation, you run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

### 4.1.3 IMPLEMENTATION

After synthesis, you run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, you can run the implementation process in one step, or you can run each of the implementation processes separately. Implementation processes vary depending on whether you are targeting a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).

### 4.1.4 VERIFICATION

You can verify the functionality of your design at several points in the design flow. You can use simulator software to verify the functionality and timing of your design or a portion of your design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows you to create and verify complex functions in a relatively small amount of time. You can also run in-circuit verification after programming your device.

### 4.1.5 DEVICE CONFIGURATION

After generating a programming file, you configure your device. During configuration, you generate configuration files and download the programming files from a host computer to a Xilinx device.

## 5 Conclusion

We have proposed a novel recursive decomposition algorithm for RB multiplication to derive high-throughput digit-serial multipliers. By suitable projection of SFG of proposed algorithm and identifying suitable cut-sets for feed-forward cut-set retiming, three novel high-throughput digit-serial RB multipliers are derived to achieve significantly less area-time-power complexities than the existing ones. Moreover, efficient structures with low register-count have been derived for area-constrained implementation; and particularly for implementation in FPGA platform where registers are not abundant. The results of synthesis show that proposed structures can achieve saving of up to 94% and 60%, respectively, of ADPP for FPGA and ASIC implementation, respectively, over the best of the existing designs. The proposed structures have different area-time-power trade-off behavior. Therefore, one out of the three proposed structures can be chosen depending on the requirement of the application environments

## 6 REFERENCES

[1] I. Blake, G. Seroussi, andN.P.Smart, Elliptic Curves in Cryptography,ser. London Mathematical Society Lecture Note Series.. Cambridge,U.K.: Cambridge Univ. Press, 1999.

[2] N. R. Murthy and M. N. S. Swamy, "Cryptographic applications of brahmaqupta-bha skara equation," IEEE Trans. Circuits Syst. I, Reg.Papers, vol. 53, no. 7, pp. 1565–1571, 2006.

[3] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," J. VLSI Digit. Process., vol. 19, pp. 149–C166, 1998.

[4] P. K. Meher, "On efficient implementation of accumulation in finite field over GF($2^m$) and its applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 541–550, 2009.

[5] L. Song, K. K. Parhi, I. Kuroda, and T.Nishitani, "Hardware/software codesign of finite field datapath for low-energy Reed-Solomn codecs,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 8, no. 2, pp.160–172, Apr. 2000.

[6] G. Drolet, "A new representation of elements of finite fields GF($2^m$) yielding small complexity arithmetic circuits," IEEE Trans. Comput.,vol. 47, no. 9, pp. 938–946, 1998.

[7] C.-Y. Lee, J.-S. Horng, I.-C. Jou, and E.-H. Lu, "Low-complexity bit-parallel systolic montgomery multipliers for special classes of GF($2^m$)," IEEE Trans. Comput., vol. 54, no. 9, pp. 1061–1070, Sep. 2005.

[8] P. K. Meher, "Systolic and super-systolic multipliers for finite field GF($2^m$) based on irreducible trinomials," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 4, pp. 1031–1040, May 2008.

[9] J. Xie, J. He, and P. K. Meher, "Low latency systolic montgomery multiplier for finite field GF($2^m$) based on pentanomials," IEEE Trans.Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 2, pp. 385–389, Feb.2013.

[10] H.Wu, M. A. Hasan, I. F. Blake, and S. Gao, "Finite field multiplier using redundant representation," IEEE Trans. Comput., vol. 51, no. 11, pp. 1306–1316, Nov. 2002.

[11] A. H. Namin, H. Wu, and M. Ahmadi, "Comb architectures for finite field multiplication in $F_{2^m}$ ," IEEE Trans. Comput., vol. 56, no. 7, pp. 909–916, Jul. 2007.

[12] A. H. Namin, H. Wu, and M. Ahmadi, "A new finite field multiplier using redundat representation," IEEE Trans. Comput., vol. 57, no. 5, pp. 716–720, May 2008.

[13] A. H. Namin, H.Wu, and M. Ahmadi, "A high-speed word level finite field multiplier in $F_{2^m}$ using redundant representation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 10, pp. 1546–1550,Oct. 2009.

[14] A. H. Namin, H. Wu, and M. Ahmadi, "An efficient finite field multiplier using redundant representation," ACMTrans. Embedded Comput. Sys., vol. 11, no. 2, Jul. 2012, Art. 31.

[15] North Carolina State University, 45 nm FreePDK wiki [Online]. Available:http://www.eda.ncsu.edu/wiki /FreePDK45:Manual

[16] P. K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for Reed-Solomon Codec," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 6, pp. 747–C757, Jun. 2009.

[17] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation.
New York: Wiley, 1999.

[18] J. L. Massey and J. K. Omura, "Computational method and apparatus for finite field arithmetic," U.S. patent application, 1984.

[19] S. Gao and S. Vanstone, "On orders of optimal normal basis generators," Math. Comput., vol. 64, no. 2, pp. 1227–1233, 1995.

[20] A. Reyhani-Masoleh and M. A. Hasan, "Efficient digit-serial normal basis multipliers over GF($2^m$) ," IEEE Trans. Comput., vol. 52, no. 4,pp. 428–439, Apr. 2003.

[21] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity word-level sequential normal basis multipliers," IEEE Trans. Comput., vol. 54, no. 2, pp. 98–C110, Feb. 2005.

[22] A. H. Namin, H. Wu, and M. Ahmadi, "A word-level finite field multiplier using normal basis," IEEE Trans. Comput., vol. 60, no. 6, pp.

## Author's Biography

MS. L. SUREKHA received her B.Tech degree from Modugula Kalavathamma Institute of Technology for Women (affiliated by JNTU Anantapuram) Department of ECE. She is pursuing M.Tech in Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP.

Miss. C. NIRMALA is currently working as an associate professor in ECE Department, Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP. She received her M.Tech from Annamacharya Institute of Technology and Science,New Boyanapalli ,Rajampet, AP.