Design Of An Area Efficient Parallel-In Parallel-Out Multiplier Using Redundant Representation

[1] G. MADHUSUDHANA RAO, [2] P. JAYA BABU, [3]T. PRAVALLIKA, [4]KATURI NISSI

[1]*H.O.D & Associate Professor*, [2]*Assistant Professor*, [3]*Assistant Professor*, [4]*M.Tech Scholar*
Dept. Of E.C.E,
N.V.R College of Engineering & Technology, Tenali, A.P

ABSTRACT: *Two digit-level finite field multipliers in F2m utilizing redundant representation are presented. Embedding F2m in cyclotomic field F(n)2 causes a certain amount of redundancy and consequently performing field multiplication using redundant representation would require more hardware resources. Based on a specific feature of redundant representation in a class of finite fields, two modern multiplication algorithms along with their pertaining architectures are proposed to alleviate this problem. Considering area-delay product as a measure of evaluation, it has been shown that both the proposed architectures considerably outperform existing digit-level multipliers utilizing the same basis. It is also shown that for a subset of the fields, the proposed multipliers are of higher performance in terms of area-delay complexities among several recently proposed optimal normal basis multipliers. The main characteristics of the post place & route application specific integrated circuit implementation of the pro- posed multipliers for three practical digit sizes are also reported.*

Index Terms: Digit-level architecture, finite field arithmetic, multiplication algorithm, redundant representation

I.INTRODUCTION

Finite field computation has gained growing attention because of its wide range of applications in coding theory, error control coding, and especially in cryptography, where ElGamal and elliptic curve cryptography (ECC) two out of the three well-known cryptosystems, are depends on finite field arithmetic. Finite field computation is performed by utilizing arithmetic operations in the underlying finite field. Among the basic field operations, multiplication plays a fundamental role as more complicated operations, namely, field exponentiation and field inversion can be carried out with consecutive use of field multiplication.

Similar to linear algebra, the concept of representation bases is also used in finite field arithmetic to represent field elements. The choice of representation system mainly affected by the hardware in use and the requirements of the cryptosystem, has a great impact on computational performance.

A few number of representation systems for extension binary fields have been proposed in the literature, such as polynomial basis normal basis (NB), redundant basis (RB), and dual basis. In both normal basis (NB) and redundant representation (RB), squaring operation can be performed by applying a simple permutation operation on the coordinates. Moreover, redundant representation is of a special interest because of its unique feature in accommodating ring type operations. This not only offers almost cost-free squaring operation but also eliminates the need for modular reduction in multiplication.

II.EXISTED SYSTEM

Fig. 1 shows the architecture, hereafter referred to as digit-level symmetrical Redundant Basis RB type$-a$ multiplier. From top to bottom, the architecture contains an n-bit circular shift register which should be initialized with the coordinates of operand *B*. This shift register provides inputs to a wire expansion module with n inputs and $w(n - 1)$ outputs followed by $((n - 1)/2)$ identical modules shown inside the dashed boxes.

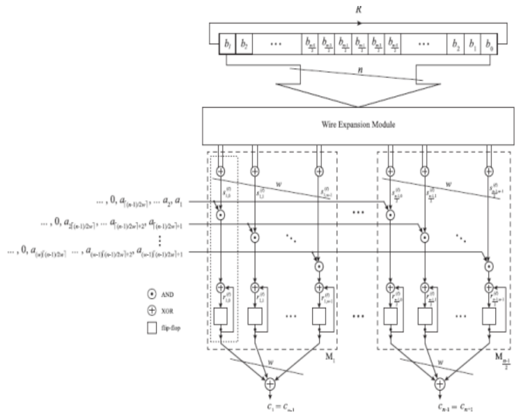


FIG. 1 EXISTED SYSTEM

At the bottom, there is a network of XOR gates adding $2w$ outputs of each module together to form output coordinates. Each module is made of a layer of $2w$ AND gates receiving the outputs of the wire expansion module as their first input set. The second input set is received from certain bits of operand *A* in a digit-serial fashion. Each AND gate is followed by an XOR gate connected immediately to a flip-flop.

The output of the flip-flop is fed back to the XOR gate forming an accumulation unit together. Two AND gates along with their respective accumulation units form a structure responsible to realize the operations. One of these structures is shown in the Fig. 1 inside a dotted block for $j = 0$ and $k = 0$. In total, the architecture contains $w(n - 1/2)$ such structures, each of which consists of two AND gates, two XOR gates, and two flip-flops to generate and store each clock cycle.

III. PROPOSED SYSTEM

A proposed multiplier is a combinational logic circuit used in digital systems to perform the multiplication of two binary numbers. These are most commonly utilized in various applications especially in the field of digital signal processing to perform the various algorithms. Commercial applications such as computers, mobiles, high speed calculators and some general purpose processors require binary multipliers. Compared with addition subtraction, and multiplication multiplication is a complex process.

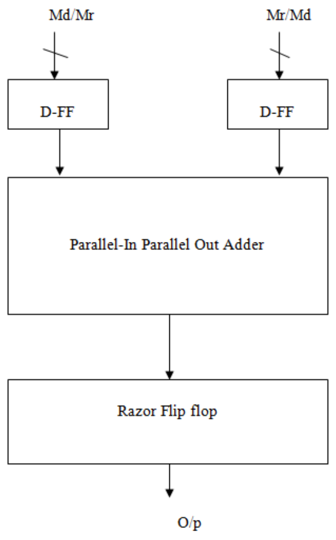


FIG. 2 PROPOSED SYSTEM

In multiplication process, the number which is to be multiplied by the other number is known as multiplicand and the number multiplied is known as multiplier. Similar to the multiplication of decimal numbers, multiplication which is proposed follows the same process for producing a product result of the two binary numbers. The binary multiplication is much easier as it consists of only 0s and 1s.
The multiplication of two binary numbers can be performed by utilizing two common methods, namely partial product addition and shifting, and utilizing parallel multipliers. Before discussing about the types, let us look at the unsigned binary numbers multiplication process. In the above multiplication, partial products are generated for each digit in the multiplier. Then all these partial products are summed to produce the final product value. In the partial product multiplication, when the multiplier bit zero, the partial product is zero, and when the multiplier bit is 1, the resulted partial product is the multiplicand.

As similar to the decimal numbers, each successive partial product is shifted one position left relative to the preceding partial product before summing all partial products. Therefore, this multiplication utilizes n-shifts and adds to multiply n-bit binary number. The combinational circuit implemented for performing such multiplication is known as an array multiplier or combinational multiplier. A multiplier essentially contains two operands, a multiplicand ‘Y’ and a multiplier ‘X’, and generates a product ‘P’. In a conventional multiplier, a number of partial products are formed initially by multiplying the multiplicand with each bit of the multiplier. These partial products are then added together for generating the product ‘P’. In short, we can break down multiplication into two parts, namely partial product generation and partial product accumulation.

Hardware multipliers, based directly on adder architectures, have become indispensable in modern computers. Multiplier circuits are modelled after the “shift and add” algorithm as shown below. In this algorithm, one partial product is created for each bit in the multiplier—the first partial product is created by the LSB of the multiplier, the second partial product is created by the second bit in the multiplier, etc. Partial products are a copy of the multiplicand if the corresponding multiplier bit is a '1', and all 0's if the corresponding multiplier bit is '0'.

Each successive partial product is shifted one bit position to the left. This specific multiplication example is recast in a generalized example on the left below. Each input, partial product digit, and result have been given a logical name, and these same names are used as signal names in the circuit schematics. By comparing the signal names in the multiplication example with the schematics, the behavior of the multiply circuit can be confirmed.

Each bit in the multiplier is AND'ed with each bit in the multiplicand to form the corresponding partial product bits. The partial product bits are fed to an array of full adders (and half adders where appropriate), with the adders shifted to the left as indicated by the multiplication example. The final partial products are added with a CLA circuit. Note that some full-adder circuits bring signal values into the carry-in inputs (instead of carries from the neighbouring stage). This is a valid use of the full-adder circuit; the full adder simply adds any three bits applied to its inputs. You are encouraged to work through a few examples on your own to confirm the adder array and CLA work together to properly sum the partial products. By utilizing Razor flip-flops timing violations occur before the next input pattern arrives can be detected.

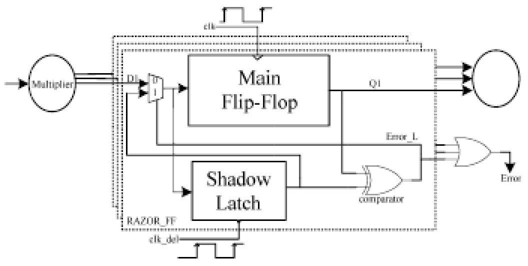


FIG.3: RAZOR FLIP-FLOPS

A 1-bit Razor flip-flop consists of a main flip-flop, shadow latch, XOR gate, and mux. The shadow latch catches the execution result using a delayed clock signal, which is slower than the normal clock signal and the main flip-flop catches the execution result for the combination circuit using a normal clock signal.

The path delay of the current operation exceeds the cycle period, and the main flip-flop catches an incorrect result if the latched bit of the shadow latch is different from that of the main flip-flop. To notify the system the Razor flip-flop will set the error signal to 1 to re execute the operation if any errors occur and notify the circuit that an error has occurred. To detect whether an operation is considered to be a one-cycle pattern can really finish in a cycle we utilize Razor flip-flops. If not, the operation is reexecuted with two cycles. Although the reexecution may seem costly, because of the reexecution frequency is low then overall cost is low.

IV. RESULTS

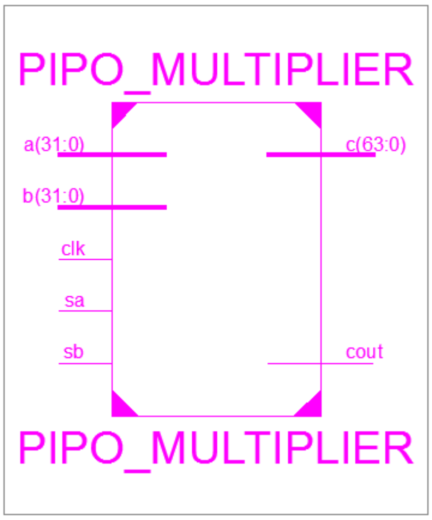


FIG 4. RTL SCHEMATIC

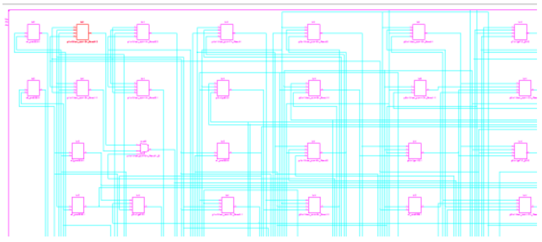
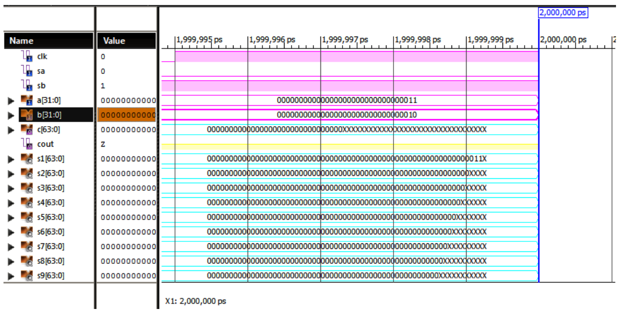


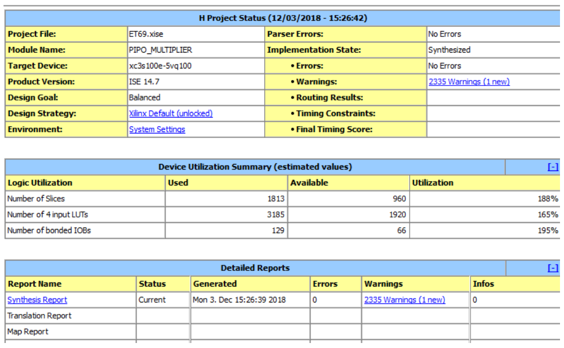
FIG 5. TECHNOLOGY SCHEMATIC



```
Name        Value          1,999,995 ps   1,999,996 ps   1,999,997 ps   1,999,998 ps   1,999,999 ps   2,000,000 ps
clk         0
sa          0
sb          1
a[31:0]     00000000000    00000000000000000000000000000011
b[31:0]     00000000000    00000000000000000000000000000010
c[63:0]     00000000000    000000000000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
cout        Z
s1[63:0]    00000000000    00000000000000000000000000000000000000000000000000000000000011X
s2[63:0]    00000000000    0000000000000000000000000000000000000000000000000000000000XXXX
s3[63:0]    00000000000    000000000000000000000000000000000000000000000000000000000XXXXX
s4[63:0]    00000000000    00000000000000000000000000000000000000000000000000000000XXXXXX
s5[63:0]    00000000000    0000000000000000000000000000000000000000000000000000000XXXXXXX
s6[63:0]    00000000000    000000000000000000000000000000000000000000000000000000XXXXXXXX
s7[63:0]    00000000000    00000000000000000000000000000000000000000000000000000XXXXXXXXX
s8[63:0]    00000000000    0000000000000000000000000000000000000000000000000000XXXXXXXXXX
s9[63:0]    00000000000    000000000000000000000000000000000000000000000000000XXXXXXXXXXX

X1: 2,000,000 ps
```

FIG 6. OUTPUT



H Project Status (12/03/2018 - 15:26:42)			
Project File:	ET69.xise	**Parser Errors:**	No Errors
Module Name:	PIPO_MULTIPLIER	**Implementation State:**	Synthesized
Target Device:	xc3s100e-5vq100	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	2335 Warnings (1 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1813	960	188%
Number of 4 input LUTs	3185	1920	165%
Number of bonded IOBs	129	66	195%

Detailed Reports [-]					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Mon 3. Dec 15:26:39 2018	0	2335 Warnings (1 new)	0
Translation Report					
Map Report					

FIG 7. REPORT

V. CONCLUSION

Two new digit-level PIPO finite field multipliers using redundant representation have been proposed. The new architecture stage layout minimizes the switching activities and reduces the electricity consumption of a Parallel in parallel out multiplier. The good judgment gate substitution technique has additionally utilized in digit-serial PB multiplier. Hence, the area complexity of the finite subject multiplier has been reduced. The proposed low-strength digit multiplier is appropriate for imposing low-electricity EC crypto systems in embedded structures with limited power resources. The proposed low-strength digit multiplier can be used as IP middle for instant implementation of EC cryptosystems. At last it produce effective results compared to existed system.

VI. IREFERENCES

[1] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inf. Theory, vol. 31, no. 4, pp. 469–472, Sep. 2006.

[2] I. F. Blake, G. Seroussi, and N. P. Smart, Elliptic Curves in Cryptography (London Mathematical Society Lecture Note Series). Cambridge, U.K.: Cambridge Univ. Press, 1999.

[3] A. J. Memezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography (Discrete Mathematics and Its Applications). Boca Raton, FL, USA: CRC Press, 1996.

[4] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in G F(2m) using normal basis," Inf. Comput., vol. 78, no. 3, pp. 171–177, 1988.

[5] C. Rebeiro, S. Roy, D. Reddy, and D. Mukhopadhyay, "Revisiting the Itoh–Tsujii inversion algorithm for FPGA platforms," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 8, pp. 1508–1512, Aug. 2011.

[6] E. D. Mastrovito, "VLSI architectures for computations in Galois fields," Ph.D. dissertation, Dept. Electr. Eng., Linköping Univ., Linköping, Sweden, 1991.

[7] J. Omura and J. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent 4 587 627, May 6, 1986.

[8] 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.

[9] D. Jungnickel, A. J. Menezes, and S. A. Vanstone, "On the number of self-dual bases of G F(qm) over G F(q)," Proc. Amer. Math. Soc., vol. 109, no. 1, pp. 23–29, 1990.

[10] S. Gao, J. von zur Gathen, D. Panario, and V. Shoup, "Algorithms for exponentiation in finite fields," J. Symbolic Comput., vol. 29, no. 6, pp. 879–889, 2000.

[11] S. Gao, J. von zur Gathen, and D. Panario, "Gauss periods and fast exponentiation in finite fields," in LATIN Theoretical Informatics (Lecture Notes in Computer Science), vol. 911. Berlin, Germany: Springer, 1995, pp. 311–322.

[12] A. H. Namin, H. Wu, and M. Ahmadi, "Comb architectures for finite field multiplication in (Fm2)," IEEE Trans. Comput., vol. 56, no. 7, pp. 909–916, Jul. 2007.

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

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

[15] J. Xie, P. Meher, and Z.-H. Mao, "High-throughput finite field multipliers using redundant basis for FPGA and ASIC implementations," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 1, pp. 110–119, Jan. 2015.

[16] R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, 2nd ed. New York, NY, USA: Cambridge Univ. Press, 1997.

[17] D. W. Ash, I. F. Blake, and S. A. Vanstone, "Low complexity normal bases," Discrete Appl. Math., vol. 25, no. 3, pp. 191–210, 1989.

[18] H. Wu, M. Hasan, and I. Blake, "Highly regular architectures for finite field computation using redundant basis," in Cryptographic Hard- ware and Embedded Systems (Lecture Notes in Computer Science), vol. 1717, C. K. Koç and C. Paar, Eds. Berlin, Germany: Springer, 1999, pp. 269–279.

[19] C. F. Kerry and P. D. Gallagher, "Digital signature standard DSS," U.S. Dept. Commerce, Nat. Inst. Standards Technol. Tech. Rep. FIPS 186-4, Jul. 2013.

[20] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," IEEE Trans. Comput., vol. 62, no. 4, pp. 744–757, Apr. 2013.