ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

Vol.02, Issue.01
July -2020
Pages: -199-214

LATENCY OPTIMIZED PARALLEL IN PARALLEL OUT EFFICIENT RB MULTIPLIER

[1]PALIVELA SUMA, [2]CH.RAJESH BABU

[1]M.Tech, Dept. of ECE, Godavari Institute of Engineering and Technology, Rajahmundry, A.P

[2]Assistant professor, Dept. of ECE, Godavari Institute of Engineering and Technology, Rajahmundry, A.P

ABSTRACT: Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. Based on a specific feature of redundant representation in a class of finite fields, two new 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 using 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. Further, this project is enhanced by using parallel in parallel out concept for latency optimization for 32 bit multiplier.

INDEX TERMS: Multiplication, latency optimization, Redundant binary, Error correction, Normal Binary.

INTRODUCTION: FINITE FIELD multiplication over Galois Field is a basic operation frequently encountered in modern cryptographic systems such as the elliptic curve cryptography (ECC) and error control coding [1]–[3]. Moreover, multiplication over a finite field can be used further to perform other field operations, e.g., division, exponentiation, and inversion [4]–[6]. Multiplication over 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. There are different types of bases to represent field elements, those are polynomial basis, normal basis, triangular basis and RB, and the choice of representation of field elements has a major impact on the performance of the arithmetic circuits. Several algorithms for basic arithmetic operations in GF (2m) are suitable for both hardware and software implementations have been recently developed. Because of several advantages of the RB based multipliers they have gained significant attention in recent years.

MULTIPLIER ARCHITECTURE, DL-SRB-A:

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 ($M1, M2, \ldots, M(n-1/2)$) shown inside the dashed boxes. At th bottom, there is a network of XOR gates adding $2w$ outputs of each module Mj together to form output coordinates. Each module Mj is made of a layer of $2w$ AND gates receiving the outputs of the wire expansion module as their

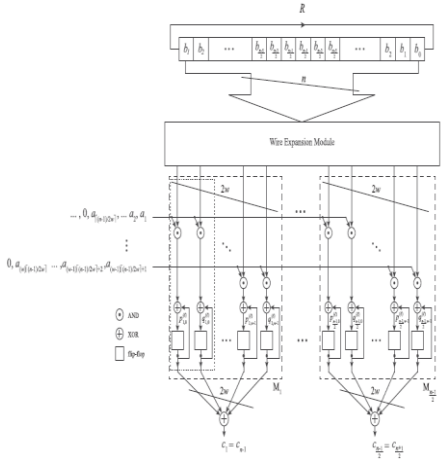


Fig. 1. Proposed architecture for digit-level SIPO RB multiplier, DL-SRB-*a*.

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 performed in Steps 5 and 6 of Algorithm 1. One of these structures is shown in the Fig. 1 inside a dotted block for $j = 0$ and $k = 0$. In total, the proposed 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 $p(\_)\ j,k$ and $q(\_)\ j,k$ in each clock cycle. As mentioned earlier, input A should be fed into the multiplier in a digit-serial fashion (comb style). According to (13), the multiplication operation is performed using $a\hat{}\,i$ coefficients which are necessarily equal to the $(n-1/2)$ coordinates of A starting from coordinate number 1 to $(n-1/2)$. We will refer to this set of coordinates of A as $\hat{}A$. Let $\hat{}A$ be divided into w parts of length d in the same way we did earlier for A, as

Note that $\hat{}A$ is padded with $wd-(n-1/2)$ zeros in the mostsignificant word. In the first clock cycle, the first bits of every word, i.e., $a1, ad+1, \ldots, a(w-1)d+1$ form an input set to the multiplier. In the second clock cycle, the inputs would be the set of second bits of every word, $a2, ad+2, \ldots, a(w-1)d+2$, so on and so forth.

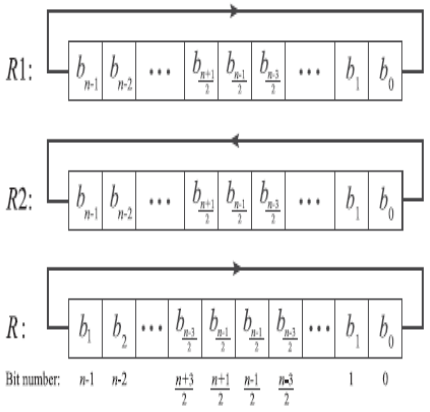


Fig.2. Circular *n*-bit shift register to store coordinates of operand *B*.

For given *j* and *k*, in each clock cycle, the variable of function ϕ in $b_{\phi(j-kd-\ell)}$ decreases by one in Step 5. An *n*-bit circular shift register can be used, as shown in Fig. 2 by *R*1, to generate the required coefficients in Step 5. This circular shift register should be initially loaded as, from left to right, $b_{n-1}, b_{n-2}, \ldots, b_0$. On the contrary, the variable of function ϕ in $b_{\phi(j+kd+\ell)}$ in Step 6 increases by one in each clock cycle In this case, a similar circular shift register, namely, *R*2, with the same initial contents but with the opposite shift direction should be utilized to produce the required coefficients.

MULTIPLIER ARCHITECTURE, DL-SRB-B:

At the expense of a slight increase in the critical path delay, the number of logic gates and flip-flops used in the architecture of Fig. 1 can be significantly reduced. Starting from the closed formula of (13), instead of the decomposition shown in (14), define two intermediate signals $s_{(\ell)j,k}$ and $r_{(\ell)j,k}$, $j = 1, 2, \ldots, (n-1/2)$ and $k = 0, 1, \ldots, w-1$ for $\ell = 1, 2, \ldots, d$ as

$$\begin{cases} s_{j,k}^{(\ell)} = [b_{\varphi(j-kd-\ell)} + b_{\varphi(j+kd+\ell)}] \\ r_{j,k}^{(0)} = 0 \text{ and } r_{j,k}^{(\ell)} = r_{j,k}^{(\ell-1)} + \hat{a}_{(kd+\ell)} s_{j,k}^{(\ell)}. \end{cases}$$

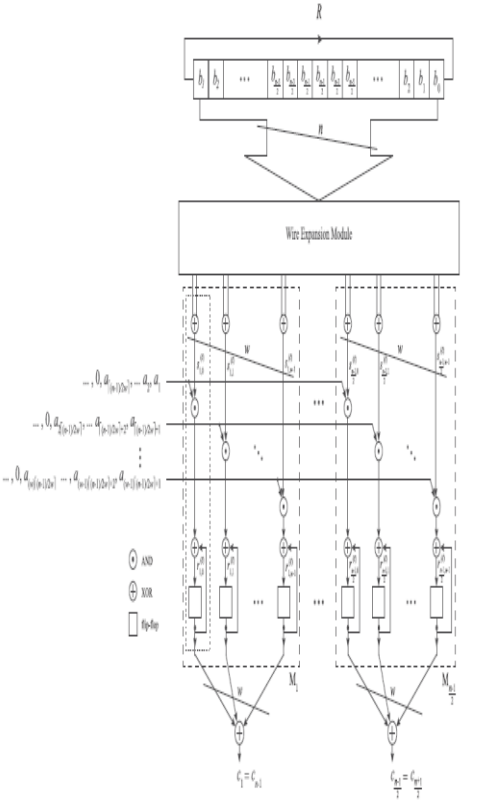


Fig. 3. Proposed architecture for digit-Level SIPO RB multiplier, DL-SRB-*b*.

$r_{(d)j,k}$ holds the value of signal *r* after *d* clock and is equal to

$$r_{j,k}^{(d)} = \sum_{\ell=1}^{d} \hat{a}_{(kd+\ell)} [b_{\varphi(j-kd-\ell)} + b_{\varphi(j+kd+\ell)}].$$

$$c_j = \sum_{k=0}^{w-1} r_{j,k}^{(d)}.$$

The new algorithm can be obtained

$$s_{j,k}^{(\ell)} = [b_{\varphi(j-kd-\ell)} + b_{\varphi(j+kd+\ell)}]$$
$$r_{j,k}^{(\ell)} = r_{j,k}^{(\ell-1)} + \hat{a}_{(kd+\ell)} s_{j,k}^{(\ell)}$$

end for

end for

end for

for all values of $j = 1, 2, \ldots, \frac{n-1}{2}$, compute in parallel

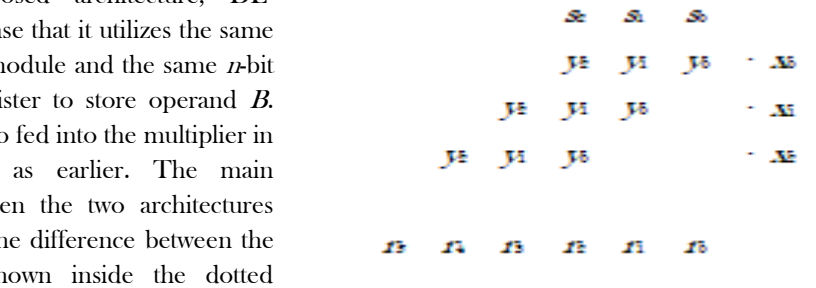
for all values of $k = 0, 1, \ldots, w-1$, compute in serial

$$c_j = \sum_{k=0}^{w-1} r_{j,k}^{(d)}.$$

Note that in each clock cycle, Steps 5 and 6 should be computed in serial. Fig. 3 shows the modified architecture referred to as DL-SRB-*b*. As can be seen from Fig. 3, the new architecture is similar to the previously proposed architecture, DL-SRB-*a*, in the sense that it utilizes the same wire expansion module and the same n-bit circular shift register to store operand B. Operand A is also fed into the multiplier in the same way as earlier. The main difference between the two architectures originates from the difference between the two modules shown inside the dotted boxes in Figs. 1 and 3. In type-*a* architecture, one bit of operand B is multiplied by one bit of operand A, and the resulting partial product is stored separately in its respective accumulation unit. On the contrary, in type-*b* architecture, two bits of operand B are first added together before they enter the AND gate and be fed into the accumulation unit. As a result, the critical path delay of the new architecture changes from $TA + TX$ to $TA + 2TX$. In the recent architecture, the number of accumulation units and AND gates are reduced by half from $w(n-1)$ to $w(n-1/2)$ each. Since half of the addition operations are performed before the accumulation units, the size of the binary XOR tree is also reduced from $2w-1$ to $w-1$. Similar to DL-SRB-*a*, the multiplication delay of DL-SRB-*b* is composed of two parts: d and $d$ex. The first part corresponds to Steps 5 and 6 of the algorithm caused by modules Mj during d clock cycles. The second part corresponds to the time delay of a w-input XOR gate or a binary tree of $(w-1)$ two-input XOR gates. Assuming that a binary tree of two-input XOR gates is used, the total number of clock cycles required to complete a single multiplication

PROPOSED TECHNIQUE:

The following scheme describes the school method for multiplication of two n-bit numbers x and y and addition of a number s for $n = 3$. The generalization to arbitrary n is obvious.



By introducing additional variables for intermediate result bits and additional subscripts to make all variables distinct this scheme is extended to the following scheme.

The values of the new variables of the second scheme are defined in terms of the variables of the first scheme as:

$s_{0j} = s_j$ for $j \in \{0, ..., n-1\}$

$x_{i0} = x_i$ for $i \in \{0, ..., n-1\}$

$y_{0j} = y_j$ for $j \in \{0, ..., n-1\}$

Then we have as the result of the multiplication/addition:

$s_{ij} = r_j$ for $j \in \{0, ..., 2n-1\}$

The resulting multiplication unit is a linear array of processing elements, shown in Figure 1 for an operand length of $n = 3$. Each processing element performs the computation determined by the recurrence equations.

The input variable x is held in the same processor in each time step. Thus, this input has to be provided in a bit-parallel way. However, a timing analysis shows that x_0 is first required at time $Z(0\ 0)^T = 0$, x_1 is first required at time $Z(1\ 0)^T = 1$, and so on. Thus, input x can be shifted into the multiplier serially and latched at appropriate times.

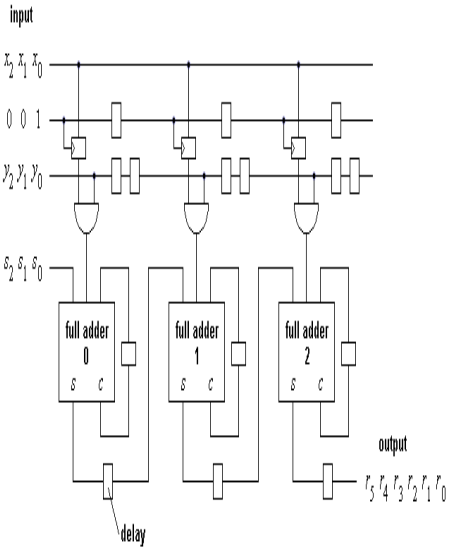
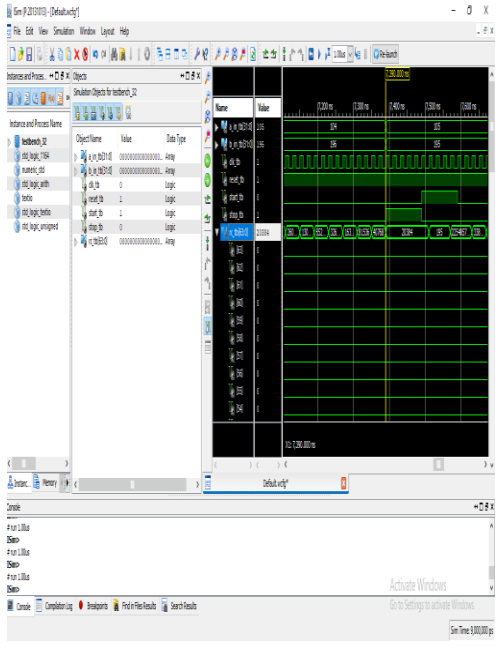


Figure4 : Parallel multiplier for three-bit numbers

For n-bit operands, the multiplier has an execution time of $3n$ cycles. It takes n cycles before the first result bit is produced at the output of the multiplier, and then another $2n$ cycles for output of the $2n$ result bits. However, successive multiplications can be pipelined in a way such that the input is provided while the last n result bits are being output. Thus, the execution time drops to $2n$.

RESULT:



CONCLUSION:

Two new digit-level SIPO finite field multipliers using redundant representation have been proposed. Numerical complexity comparison showed that both new architectures have the lowest delay cost compared with the existing RB architectures. the proposal can show better performance than ONB multipliers, if existed, and can show much better performance than NB multipliers

REFERENCES:

1. J. Xie, P. Meher, and J. He, "Low-complexity multiplier for *G F(2m)* based on all-one polynomials," *IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst.*, vol. 21, no. 1, pp. 168–173, Jan. 2013.

2. J. Menezes, I. F. Blake, S. Gao, S. A. V. R. C. Mullin, and Yaghoobian, Eds., *Applications of Finite Fields*. Boston, MA, USA: Kluwer, 1993.

3. R. Lidl and H. Niederreiter, Eds., *Introduction to Finite Fields TheirApplication*. New York, NY, USA: Cambridge Univ. Press, 1986.

4. [Online]. Available: http://www.csrc.nist.gov/publications

5. S. K. Jain, L. Song, and K. K. Parhi, "Efficient semisystolic architectures for finite-field arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI)Syst.*, vol. 6, no. 1, pp. 101–113, Mar. 1998.

6. L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 19, no. 2, pp. 149–166, 1998.

7. P. K. Meher, "Systolic formulation for low-complexity serial-parallel implementation of unified finite field multiplication over *G F(2m)*," in *Proc. 18th IEEE Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*,Jul. 2007, pp. 134–139.

8. F. Rodriguez-Henriguez and C. K. Koc, "Parallel multipliers based on special irreducible pentanomials," *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1535–1542, Dec. 2003.

9. A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over *G F(2m)*," *IEEETrans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

10. W. Tang, H. Wu, and M. Ahmadi, "VLSI implementation of bit-parallel word-serial multiplier in *G F(2233)*," in *Proc. 3rd Int. IEEE-NEWCASConf.*, Jun. 2005, pp. 399–402.

11. H. Wu, "Low complexity bit-parallel multiplier for a class of finite fields," in *Proc. Int. Conf. Commun., Circuits Syst.*, vol. 4. Jun. 2006, pp. 2565–2568.

12. P. K. Meher, "High-throughput hardware-efficient digit-serial architec-ture for field multiplication over *G F(2m)*," in *Proc. 6th Int. Conf. Inf.Commun. Signal Process. (ICICS)*, Dec. 2007, pp. 1–5.

13. P. K. Meher, "Systolic and super-systolic multipliers for finite field *F(2m)*based on irreducible trinomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 5, pp. 1031–1040, May 2008.

14. R. Katti and J. Brennan, "Low complexity multiplication in a finite field 1