International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 06
May 2017

Design via DLL Multiplier Using Redundant Basis for FPGA and ASIC Implementation

M.JAGADEESH MALLIREDDI*, V.RAMARAO
M.JagadeeshMalliReddy, M-Tech (VLSI), ECE Department College of Engineering and Technology, Eluru JNTUK A.P
V.Ramarao, M-Tech, MIAENG, Assistant-Professor, Department of ECE College of Engineering and Technology, Eluru JNTUK A.P

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 and modular reduction. In this paper we discuss the growth has started the spread of architectures for implementing ECC from FPGA towards ASIC. Computing scalar multiplication and point inversion forms the core ECC architecture. ASIC based implementation of these ECC arithmetic primitives over finite fields GF (2m). we have proposed a Dual Logic Level (DLL) the arithmetic components are designed using Verilog and implemented on field programmable gate array (FPGA) and application specific integrated circuit (ASIC) realization of the proposed designs especially presented in high-throughput up to 50% and 20% savings area-delay-power product (ADPP) implementation over the best of the existing designs, respectively.

Index Terms—ASIC, finite field multiplication, FPGA, high-throughput, redundant basis. Dual Logic Level, digital series RB multiplier.

I. Introduction

Finite field GF (2m) is a field that contains finitely many fields. It is especially useful in translate computer data, which present in the binary form. Finite Field has wide applications in cryptography and error control coding [1], [2]. The key arithmetic unit for multiple systems based on computations of finite field is finite field multiplier because the complex operations like division and inversioncan be broken down into successive multiplication operation. The most common arithmetic is multiplication which is useful to obtain efficient multipliers [3].

A number of structures have been designed for efficient finite field multiplication over finite field based on RB. Semi-systolic Montgomery multiplier ispresented in [4]. Super-systolic multiplier has been reported by Pramod Kumar Mehar. Bit-Serial/Parallel multipliers [8], Comb style architectures are presented formerly and also several other RB multipliers are designed for hardware efficiency and throughput[5] [6]. In this contribute, an efficient high-throughput digit-serial/parallel multiplier designs over finite field based on RB is presented. A novel recursive decomposition scheme is presented, based on that parallel algorithms are obtained for high-throughput digit-serial multiplication [7]. By depicting the parallel algorithm to a regular two dimensional signal-flow-graph (SFG) array go after by projection of SFG to onedimensional processor-space flow graph (PSFG), the algorithm is mapped to three multiplier architectures. In this work, the implementation of 10-bit digit-serial RB multipliers is presented to obtain high-throughput. The organization of this paper is as follows: Mathematical representation is presented in section II. High-throughput structures for digit-serial RB multipliers are derived from the proposed algorithm mentioned in section III. Implementation and Simulation results are presented in section IV.

Introduction about the dual logic level (DLL) High-speed opto coupler is capable of transmitting binary values the full specified operating temperature range. The combination of low input current (1.6 mA) and Active logic-level output is a fit for nearly all logic applications.

APPLICATIONS: Requirements for Standard, Fast, and High* Greater Design Flexibility Ultra-Low Power Consumption Saves Space 8-Bump, 0.4mm Pitch, 0.8mm x 1.6mm WLP Package 8-Pin, 2mm x 2mm Speeds

The Finite Field GF(2^8).

The case in which **n** is greater than one is much more difficult to describe. In cryptography, one almost always takes **p** to be **2** in this case. This section just treats the special case of **p = 2** and **n = 8**, that is. **GF(2^8)**, because this is the field used by the new U.S. *Advanced Encryption Standard* (AES).

Multiplication in GF(2^8).

Multiplication is this field is much more difficult and harder to understand, but it can be implemented very efficiently in

hardware and software. The first step in mutiplying two field elements is to multiply their corresponding polynomials just as in beginning algebra (except that the coefficients are only **0** or **1**, and **1 + 1 = 0** makes the calculation easier, since many terms just drop out). The result would be up to a degree **14** polynomial -- too big to fit into one byte. A finite field now makes use of a fixed degree eight irreducible polynomial (a polynomial that cannot be factored into the product of two simpler polynomials). For the AES the polynomial used is the following (other polynomials could have been used): **(x) = $x^8 + x^4 + x^3 + x + 1$ = 0x11b** (hex). The intermediate product of the two polynomials must be divided by **m(x)**. The remainder from this division is the desired product. This sounds hard, but is easier to do by hand than it might seem (though error-prone). To make it easier to write the polynomials down, adopt the convension that instead of $x^8 + x^4 + x^3 + x + 1$ just write the exponents of each non-zero term. (Remember that terms are either zero or have a **1** as coefficient.) So write the following for **m(x)**: **(8 4 3 1 0)**.

Typical Elliptic Curve Crypto processor:

Elliptic curve crypto systems have a layered hierarchy as shown in Figure1. The bottom layer constituting the arithmetic on the underlying finite field most prominently influences the area and critical delay of the overall implementation[9]. The group operations on the elliptic curve and the scalar multiplication influences the number of clock cycles required for encryption.

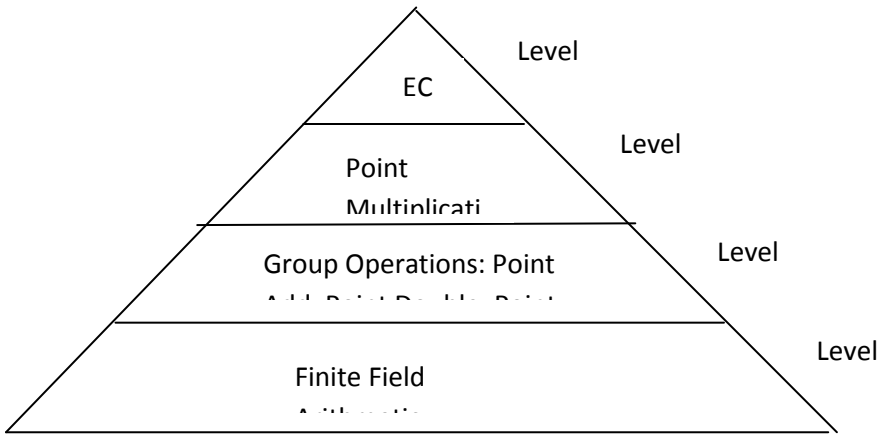


Figure 1: ECC layered hierarchy for arithmetic operations

To be usable in real world applications, implementations of the crypto system must be efficient, scalable, and reusable[10]. Applications such as smart cards and mobile phones require implementations where the amount of resources used and the power consumed is critical. Such implementations should be compact and designed for low power. Computation speed is a secondary criterion. Also, the degree of reconfigurability of the device can be kept minimum [11][12]. This is because such devices have a short lifetime and are generally configured only once. On the other side of the spectrum, high performance systems such as network servers, data base systems etc. require high speed implementations of Elliptic Curve Cryptoporcessor (ECCP)[13][14]. The crypto algorithm should not be the bottleneck on the application‟s performance. These implementations must also be highly flexible. Operating parameters such as algorithm constants, etc. should be reconfigurable

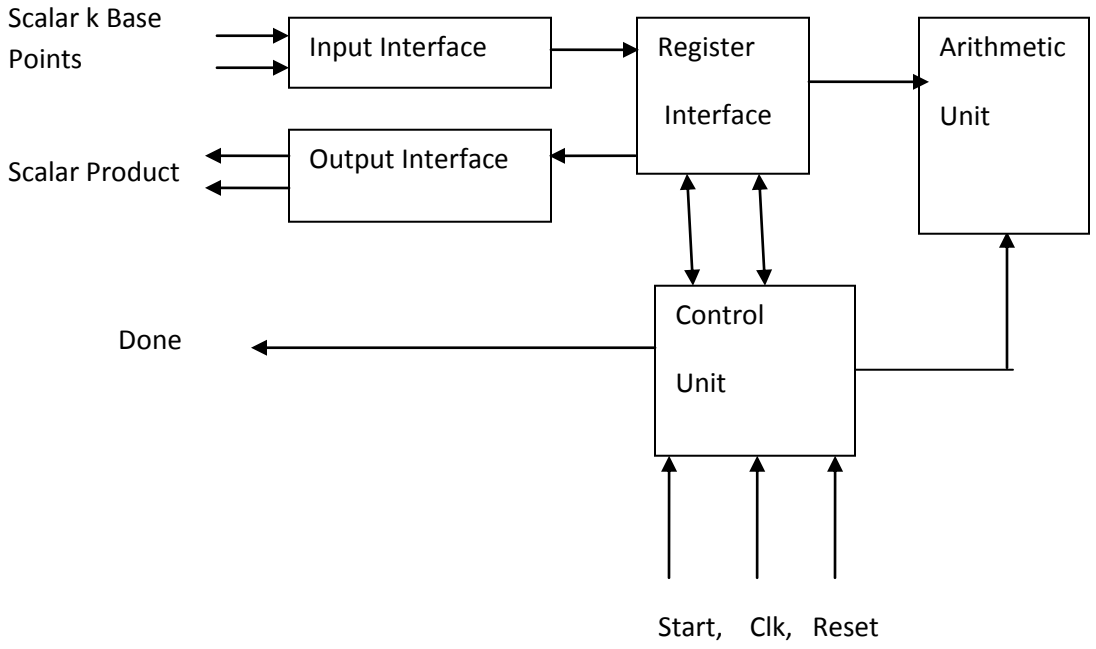


Figure 2: A typical elliptic curve crypto processor

The Arithmetic primitives from the ECCP hierarchy can be grouped to constitute a dedicated hardware. A typical ECCP is given in Figure2. This crypto processor should implement the double and add scalar multiplication algorithm. Point doubling is performed for every iteration loop of the algorithm. Point addition is performed only when the bit is set in the binary expansion of scalar input k. This constitutes the

Arithmetic unit (AU) of ECCP shown in Figure3. The output is the scalar multiplication product kP. Here P is the base point on the curve. Finite Field Arithmetic : Addition, Multiplicaion, Inversion Group Operations: Point Add, Point Double, Point Reflect Level 0 Level 1 Level 2 Level 3 Point Multiplication ECC Input Interface Output Interface Control Unit Arithmetic Unit Register Interface Scalar k Base Points Scalar Product Done Start, Clk, Reset.

Digit-serial RB multiplier:

The proposed digit-serial RB multiplier is derived from the SFG of the proposed algorithm the representation of RB multiplication is by two dimensional SFG in Fig.1. The SFG consists of Q number of arrays which are in parallel; each array is with (P-1) bit-shifting nodes which is S node. The S nodes are two types they are S-I and S-II. The one position circular bit-shifting is carryout by S-I and Q positions circular bit-shifting is carryout by S-II. And it also consists of P multiplication nodes and addition nodes, where M nodes and A nodes.

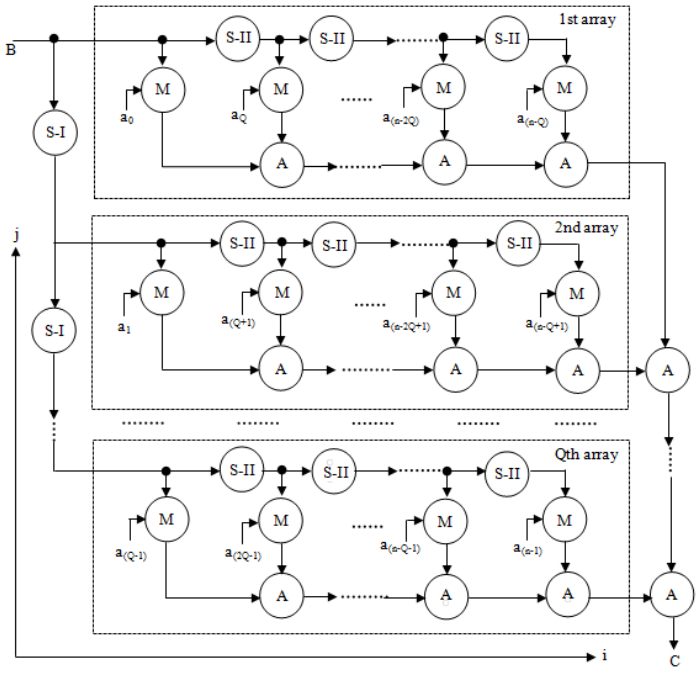
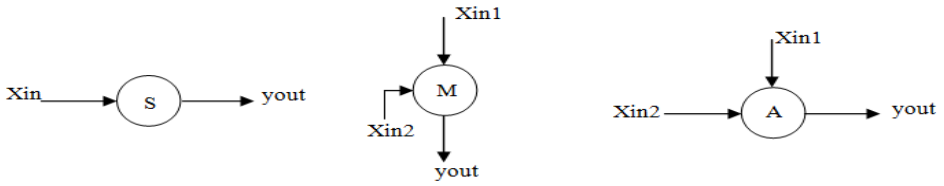


Fig.3. The proposed Signal-flow graph (SFG) for parallel realization of RB multiplication

The role of M nodes and A nodes are described in Fig.4 (b) and 4(c). M node carryout AND operation of each serial-input bits of Awith the B input bits by bit-shifting

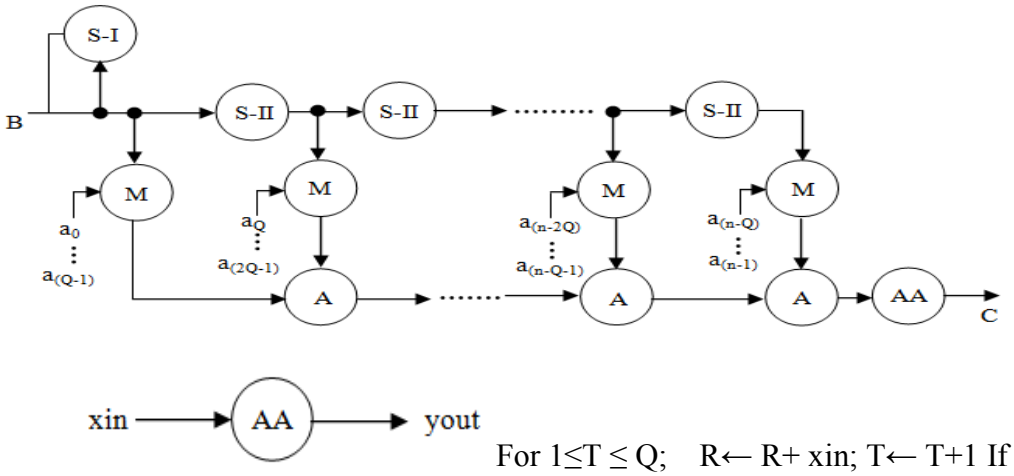
form, and XOR operation is carryout by the each Anode. The final output is obtained by performing the bit by bit XOR operation of the operands Fig.3.By addition of the Q parallel arrays output the required product word is obtained. To obtain the PSFG (Fig.6), the SFG is projected along the jth direction for digit-serial multiplication.In PSFG during each clock cycle the p number of input bits carried in parallel to multiplication node. The PSFG functionality is as same as the SFG inFig.1 It consists of an extra node which is add-accumulation node (AA) and the role of the add-accumulation is to carryout accumulation operation to produce necessary result.



(a) (b) (c)

Bit-shifting (Xin) →youtXin1. Xin2→yout Xin1+Xin2→yout

Fig. 4. (a) Functional representation of S node. (b) Functional representation of M node. (c) Functional Representation of A node



For 1≤T ≤ Q; R← R+ xin; T← T+1 If T=Q then yout← R;
T←0; R←; R← 0; Endif

Fig.5. Processor- space flow graph (PSFG) of digit-serial realization of finite field RB multiplication over GF (2m). (a) The PSFG. (b) Functional representation of add-accumulation (AA) node.

The digit-serial RB multiplier shown in Fig.8, mentioned as structure-I. Structure-I consists of three blocks, which are bit-permutation module (BPM), partial product generation module (PPGM) and finite fieldaccumulator module. The BPM carries out rewiring of inputsB and the output is fed to the partial product generation unit. The PPGM is with the AND, XOR and register cells which carry out the function of M node and the finite field accumulator block consistent with n-bit parallel accumulation units. The recent input which is received is added with the past accumulated result, and the sum is retain in the register cell and used in the next cycle. And successive output is obtained. Fig.11 shows the structure of partial product generation module which consists of XOR cell, AND cell and register cells with n parallel input bits and n parallel output bits.

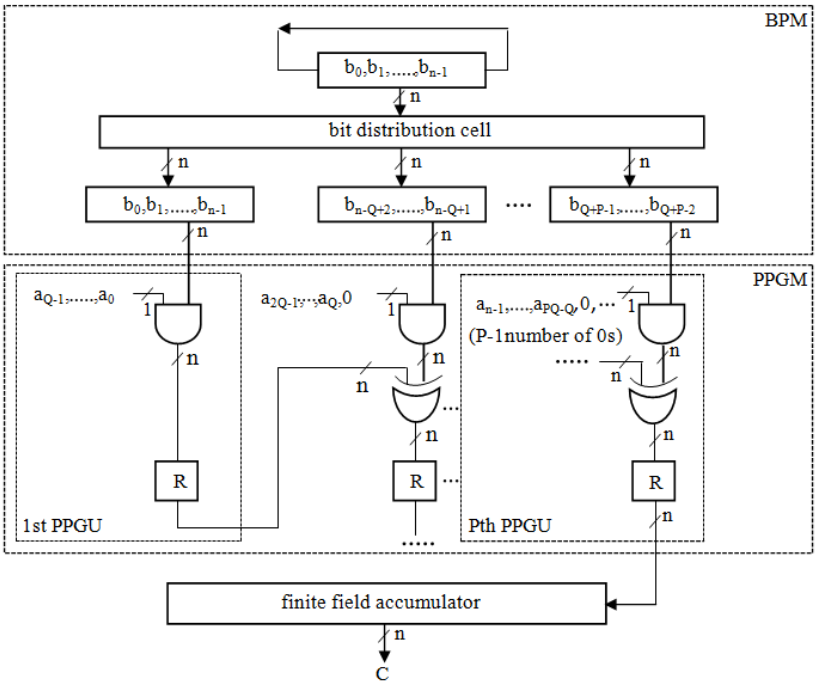


Fig.6. structure-I for digit-serial RB multiplier

Modification of Structure –I for Digit-Serial RB Multiplier

We can have ($P=kd+l$), for any p integer value, where $0\leq\square<d$ and $d<P$. For simpleness, we assume l=0, however can easily extended to the cases where $l \neq 0$. Define $0\leq h \leq\square-1$, and $0\leq\square\leq\square-1$,

the PSFG is modified to obtain appropriate digit-serial multiplier structure Fig.6, a set of

shifting nodes, a set of multiplication nodes and a set of addition nodes of PSFG are combined to form overall node. And these nodes are executed by new PPGU to obtain PPGM of P/2 PPGUs. Suitably, in the structure of Fig.4 the two PPGU are appeared into a new PPGU, and it consists of two AND cells, two XOR cells and it needs only one XOR cell at the first PPGU of the structure-I when d=2. The functionality of the AND, XOR and register cells are same as the structure-I in Fig.4.

Structure- II for digit-serial RB multiplier

The Structure-II for digit-serial RB multiplier is in Fig.9, the (P-1) A nodes of PSFG which are connected serially are combined into the pipeline form of (P-2) A nodes. And these pipeline forms of A nodes are constructed by using the pipeline XOR tree. To meet the time requirement there is no need of padding '0' at input due to the AND cell is organized in parallel. The function is as same as the structure-I.

Structure-III for digit-serial RB multiplier

In this, the bit-addition and bit-multiplication are carried out concurrently and hence the throughput of the desired structure can be increased. The structure-III for digit-serial RB multiplier is shown in Fig.10, which contains (P+1) PPGUs and the each PPGU is with the single AND cell, single XOR cell and two register cells and the first output of this structure-III can be obtained at (P+Q+1) cycles. And at Q cycles the consecutive output is obtained.

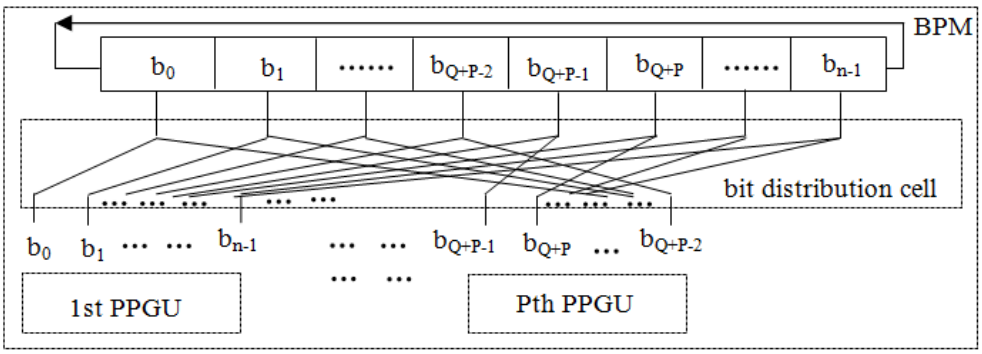


Fig.7. structure of the bit-permutation module (BPM)

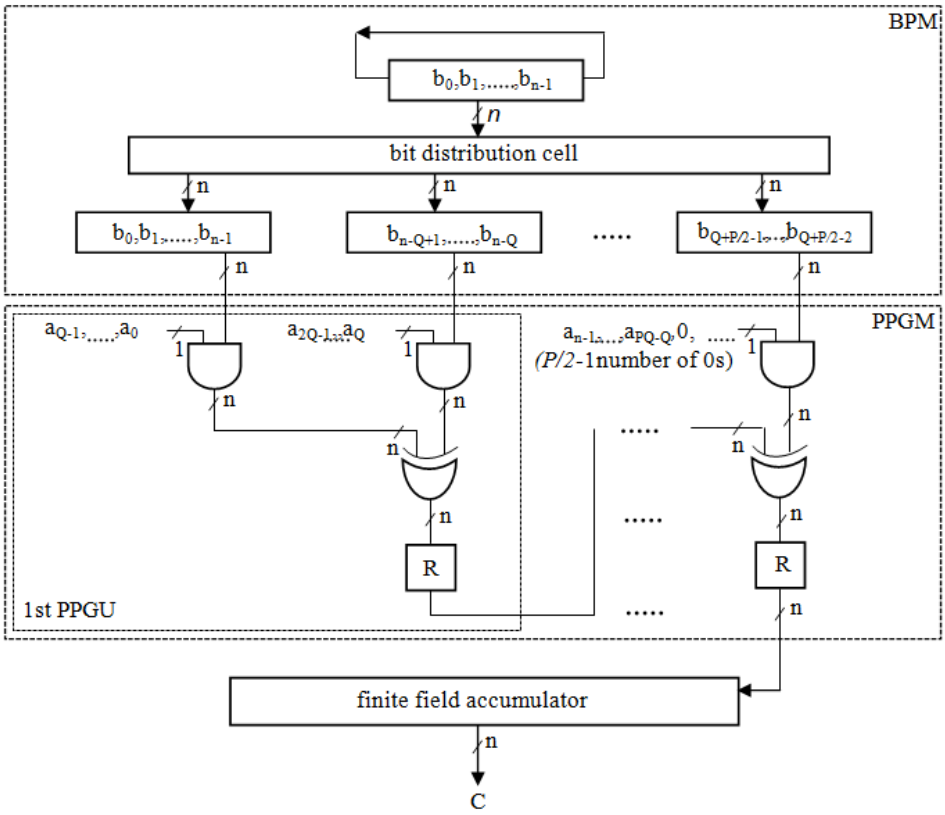


Fig.8. Structure-I for digit-serial RB multiplier when d=2.

The fig.5 shows the structure of the bit-permutation module, and fig.11(a), 11(b) and fig.11(c) shows the structure of AND cell, XOR cell and register cell of PPGM. Which the inputs are given parallel to the AND cell and obtain the output parallel and also which is done similar to the XOR cell and register cell. This consists of n parallel inputs and n parallel outputs.Fig.8. Shows the structure of finite field accumulator, the finite field accumulator also consists of XOR cell and register cell with the parallel inputs and parallel outputs.

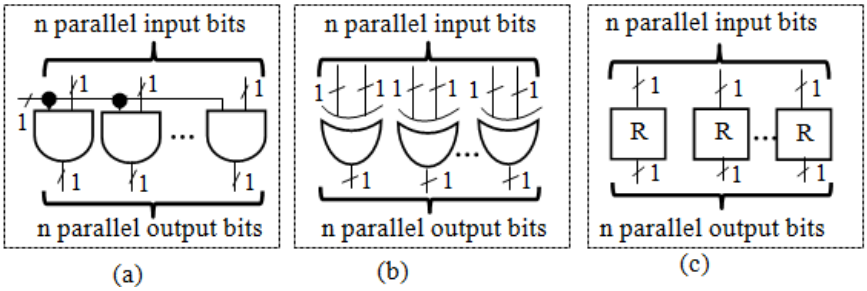


Fig.9 (a) Structure of AND cell in partial product generation module. (b) Structure of XOR cell in partial product generation module. (C) Structure of register cell in partial product generation module.

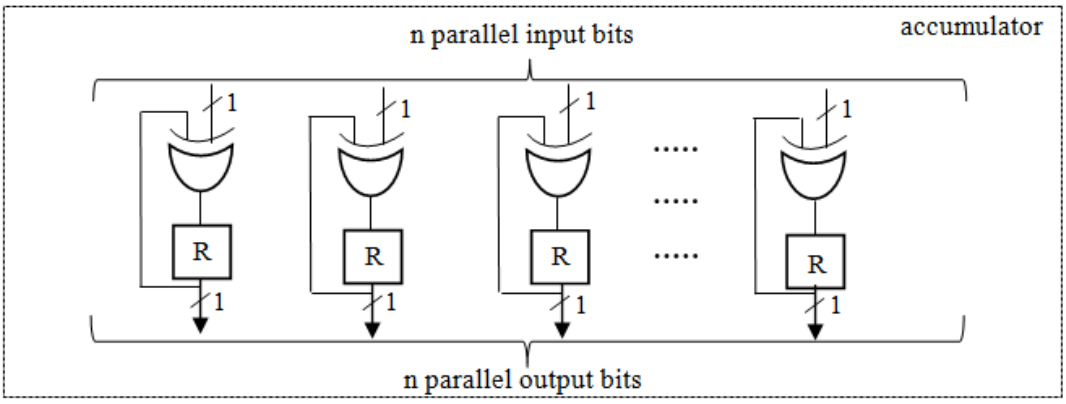


Fig.10. Structure of the finite field accumulator

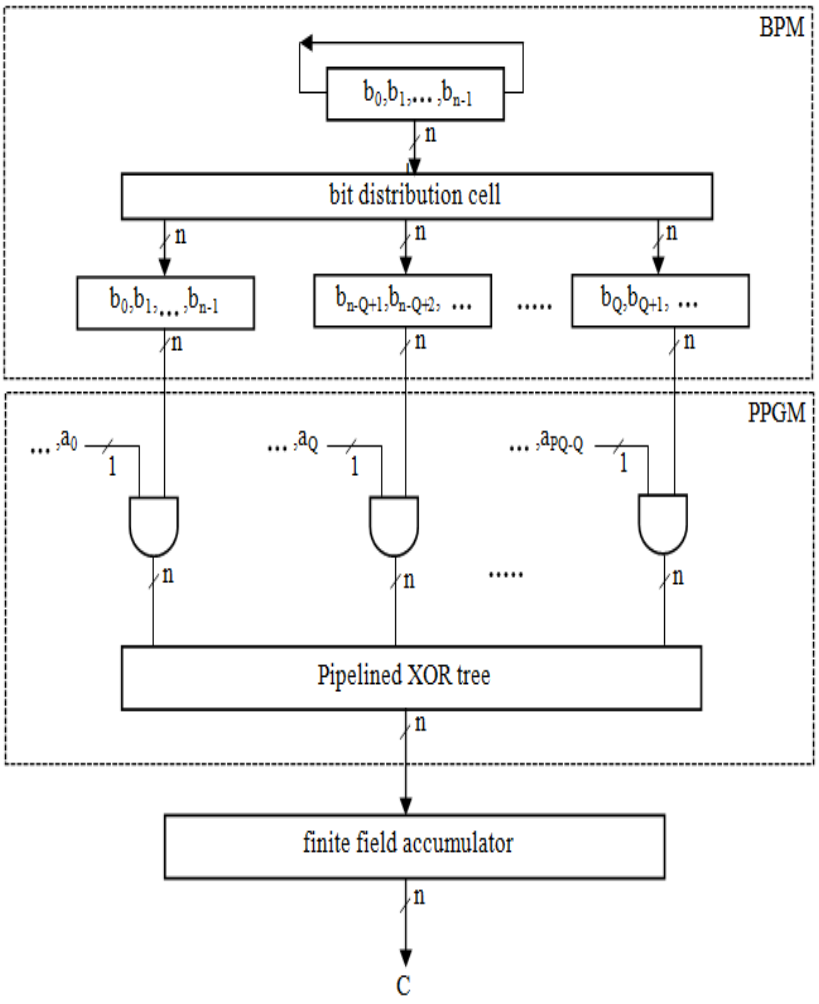


Fig.11. Structure-II for digit-serial RB multiplier

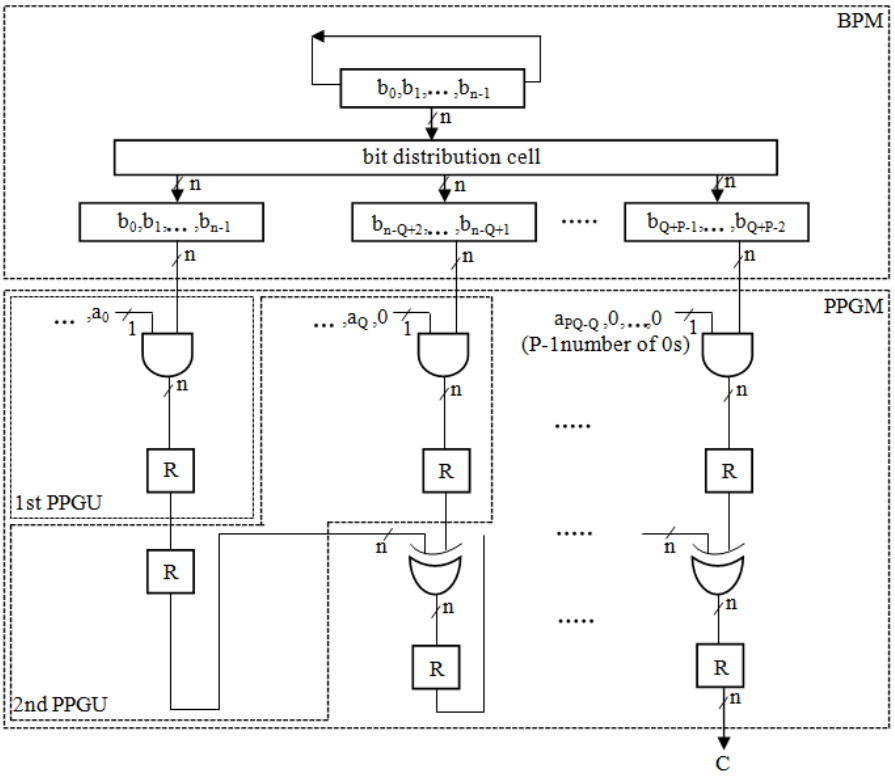


Fig.12. Structure-III for digit-serial RB multiplier

The pipelined tree:

We can further transform the PSFG of Fig. 4 to reduce the latency and hardware complexity of PS-I. To obtain the proposed structure, serially-connected A nodes of the PSFG of Fig. 4 are merged into a pipeline form of A nodes as shown within the dashed-box in Fig. 6(a). These pipelined A nodes can be implemented by a pipelined XOR tree, as shown in Fig. 6(b). Since all the AND cells can be processed in parallel, there is no need of using extra "0"s on the input path to meet the timing requirement in systolic pipeline. The critical path and throughput of PS-II are the same as those of PS-I. Similarly, PS-II can be easily extended to larger values of d to have low register-complexity structures.

Cut-set retiming :

Since the S nodes of Fig. 4 perform only the bit-shifting operations they do not involve any time consumption. Therefore, we can introduce a novel cut-set retiming to reduce the critical path further, as shown in Fig. 7(a). It can be observed that the cut-set retiming allows to perform the bit-addition and bit-multiplication concurrently, so that the critical-path is reduced to

max{T_A+T_X}=T_X, i.e ,the throughput of the design Is increased the pipelined tree for RB multiplier, where "R" denotes a register cell. (a) Modified PSFG. (b) Structure of RB multiplierhave 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 50% and 20%, 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.

DLL Simulation Results

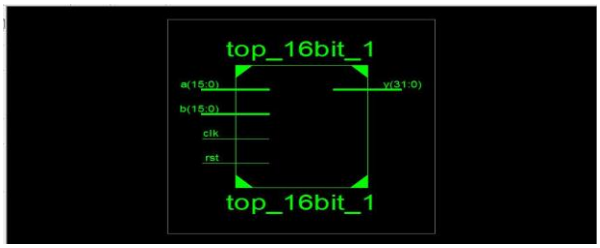


Fig 14: Schematic diagram of DDL

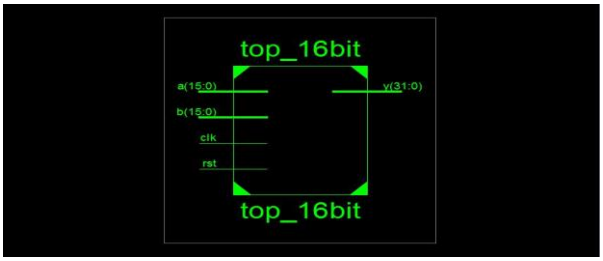
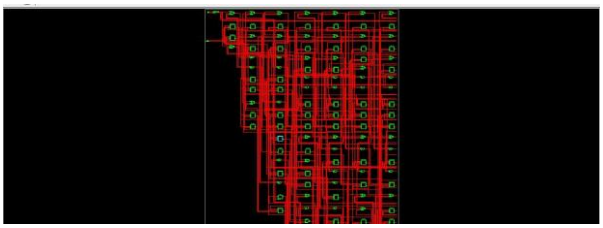


Fig 15: Schematic diagram of DDL-DB

Technology Schematic of DLL



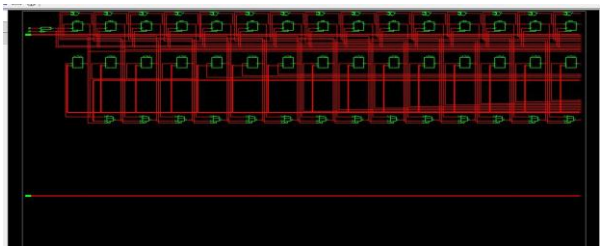


Fig 16: **Technology schematic of DLL merged regular PPGU**

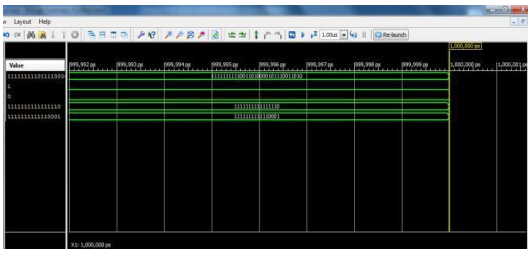


Fig 17: DLL Simulation results of pipelined tree

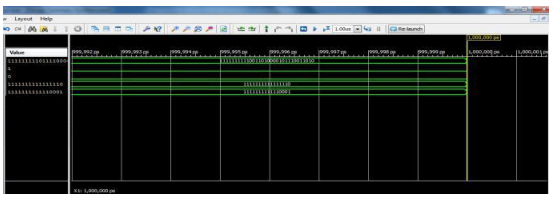


Fig 18: DLL Simulation results of cut-set retiming

CONCLUSION

We have proposed a duel logic level 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 proposed structures can be chosen depending on the requirement of the application environments.

REFERENCES

[1] I. Blake, G. Seroussi, and N. 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-bhaskara 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 and its applications," IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst., vol. 1 7, 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 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," IEEE Trans. Comput., vol. 54, no. 9, pp. 1061–1070, Sep.2005.[8] P. K. Meher, "Systolic and super-systolic multipliers for duel logic level 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, "Lowlatency systolicmontgomerymultiplier for finite field 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 ," 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 epresentation," 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 multiplierF2m in 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.