Area Efficient Implementation of Elliptic Curve Point Multiplication Algorithm

Sunil Devidas Bobade
Research Scholar
S.G.B.Amravati University
Amravati, India

Dr.Vijay R. Mankar
Deputy Secretary
Regional Board of Technical Education
Pune Region, Pune, India

***Abstract*—Elliptic Curve Cryptography (ECC) has established itself as the most preferred and secured cryptography algorithm for the secure data transfer and secure data storage in embedded system environment. Efficient implementation of point multiplication algorithm is crucial activity for designing area efficient, low footprint ECC cryptoprocessors. In this paper, an area efficient implementation of double point multiplication algorithm over binary elliptic curve is presented. Area analysis of double point multiplication algorithm based on differential addition chains method is carried out and area report is generated. Area optimization is achieved by using pipelined structure and by reutilizing idle resources from previous stages in processing unit. The proposed architecture for double point multiplication is implemented on Xilinx Virtex-4 FPGA device. Architecture is modeled in verilog-HDL and synthesized using Xilinx ISE 14.1 design software and is found to be more efficient in terms of area than the existing such architectures.**

Keywords*—Cryptography; Elliptic Curve Cryptography; Double Point Multiplication; Binary Elliptic Curve; Differential Addition Chain

I. INTRODUCTION

Victor Miller and Neal Koblitz proposed the concept of elliptic curve cryptography in the mid of 1980's and was considered as a next big step in public key cryptographic systems. Few algorithms already existed such as DSA and RSA. The main advantage of ECC over RSA is the usage of shorter key and it is aided with a drawback that the design for ECC when implemented in software performs at dead slow speed, whereas if the implementation is done in hardware, the process is much more efficient. Hence ECC is the best choice for cryptographic hardware implementation. Due to these many advantages of ECC, a number of hardware implementations have been proposed, and included in many standards such as IEEE 1363and NIST.

An operation called point addition is defined on an elliptic curve. The point addition is an operation, where two points on the curve are added and a third point, which is also on the curve, is plotted as shown in figure 1. Importantly for cryptography, it is very hard to analyze which two points were added. Furthermore, using consecutive point additions, an operation called "Elliptic curve point multiplication" is defined. The most exorbitant finite field operation for point addition and point doubling is the finite field inversion. However, one way to handle finite field inversion is by transforming it into less expensive finite field operation, such as finite field addition and multiplication by using projective coordinates. The elliptic curve point doubling and point multiplication activities are shown in figure 2 and 3.

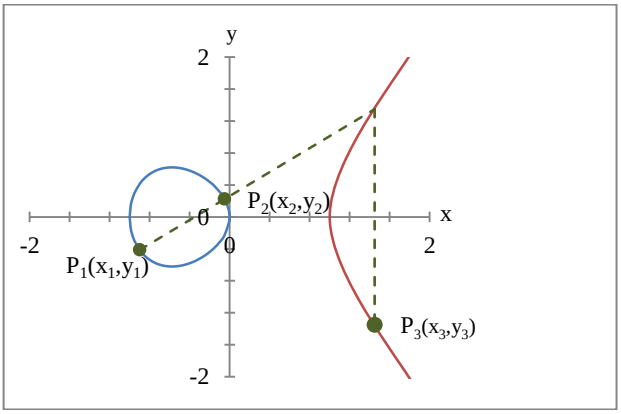

Fig.1. Point addition $P_1+P_2=P_3$

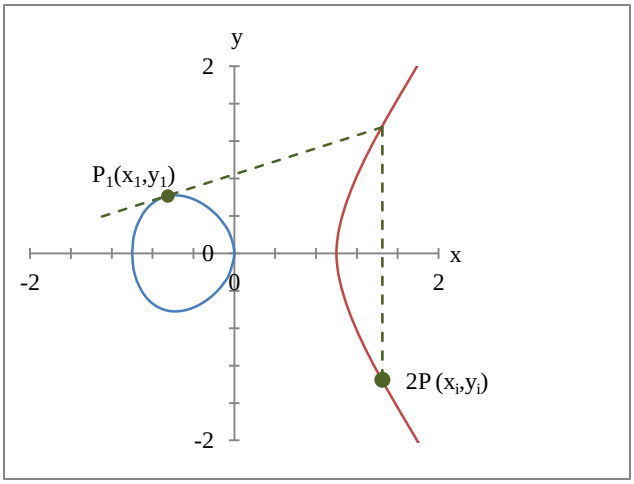

Fig.2. Point doubling 2P

A vast number of resource-constrained and high-performance embedded applications utilize the ECC based public key cryptography due to shorter key sizes.. The core operation in ECC systems is the point multiplication. The security of the cryptosystems like ECC depends mainly on the difficulty of the discrete logarithm problem (DLP). A commonly adopted method of solving DLP is to use the Pollard's rho technique [1].

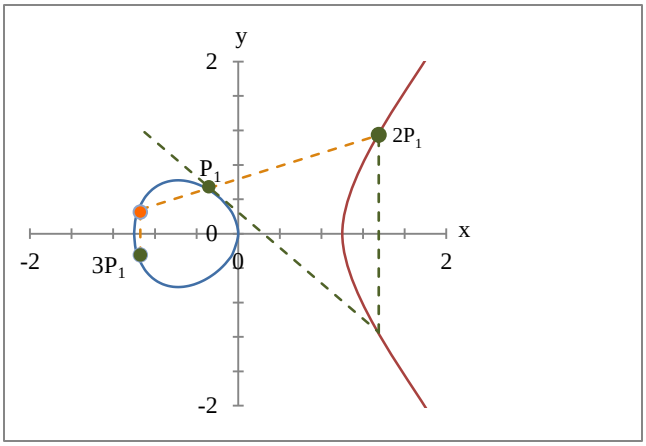

Fig.3. Point multiplication on Elliptic Curve

A traditional technique for computing side channel information is to apply a variant of double-and-add type algorithms with respect to the binary form of the secret exponent '*a*'. Such an algorithm would deteriorate due to power analysis attacks when doubling and addition operations are distinct [2]. One method to provide Diffie-Hellman type protocols with some level of protection against side channel attacks, is to divide the scalar a = *r + (a r)* for some secret random integer *r*, and to compute *aP = rP + (a r)P* [3].

For the sake of generality, let *G* be an additive abelian group. Given an integer *a* and a point *P* belonging to *G*, a (single) point multiplication algorithm computes *aP* belonging to *G*. Given two integers *a*, *b* and two points *P*, *Q* belonging to *G*, a double point multiplication algorithm computes *aP +bQ* belonging to *G*. Having an efficient and secure one double point multiplication algorithm is important for most of the cryptographic schemes. Another scenario where one needs efficient and secure double point multiplication is to speed up single point multiplication over elliptic curves with endomorphism as in [4], [5], [6].

A simple way to implement double point multiplication is by making use of two single point multiplications in parallel. Straus-Shamir's trick [7] and interleaving [8] are two such methods. Straus-Shamir's type simultaneous double point multiplication algorithms are sensitive to side-channel analysis, because of which double and add instructions are not accomplished in a linear fashion. Fortunately, recoding the scalars *a* and *b* allows one to make use of Straus-Shamir's type algorithms in such a way that the same instructions are executed in the same order. Joye and Tunstall [9] introduced several techniques of regular recoding of scalars for regular point multiplication algorithms, which can immediately be adapted to yield regular simultaneous double point multiplication algorithms. Especially, their signed-digit recoding technique with the digit set f1; 3g generate a regular double point multiplication algorithm, referred as JT-f1;3g algorithm. JT-f1; 3g costs 0.5 addition and 1 doubling per scalar bit. Adapting differential addition chains (DAC) is another technique to compute simultaneous double point multiplication [10], [11], and [12]. DAC-method is more prominent as it produces potentially simple power analysis resistant algorithms due to the uniform pattern of operations executed and it is particularly efficient towards elliptic curves setting because of the fact that double and add operations can be computed only using x-coordinates only. Bernstein [12] proposed a double point multiplication algorithm related to the new binary chain, known as the B-NBC algorithm. B-NBC has a uniform framework, and costs 2 additions and 1 doubling per scalar bit. Recently, Azarderakhsh and Karabina [13] designed a simultaneous double point multiplication algorithm based on DAC, the AK-DAC algorithm. AK-DAC has a uniform structure, and costs 1:4 additions and 1:4 doublings per scalar bit.

The above mentioned three double point multiplication algorithms JT-f1; 3g, B-NBC, and AK-DAC are normal, and hence they are potentially resistant towards power analysis attacks. Nevertheless, comparing these algorithms with respect to the efficiency point of view is not straight forward. Although JT-f1; 3g shows the best per-bit cost, B-NBC and AK-DAC have the benefit of being based on DAC. For example, in elliptic curves setting, one can implement B-NBC and AK-DAC by adapting the addition formulas that include only the x-coordinates of the points, and are much more efficient than that of their conventional counterparts. Moreover, JT-f1; 3g cannot be executed in parallel because an addition operation should be always performed following two successive doubling operations. Double and add operations can be completely parallelized in both B-NBC and AK-DAC. If one redistributes 2 parallel addition/doubling units, then the costs of B-NBC and AK-DAC per bit becomes 1A+1D and 1:4A. In the same way, if one redistributes 3 parallel addition/doubling units, then the per-bit cost of B-NBC becomes 1A.

In this paper, hardware architecture of Area efficient Elliptic Curve Point Multiplication using AK-DAC standard Weierstrass binary elliptic curve groups is implemented and is investigated for area occupancy. This will be realized with the promising regular algorithm with low hardware requirement (area).

The rest of the paper is organized as, Section 2 reviews some of the latest research works performed related to proposed work and in Section 3 the motivation and the methodology of research are discussed. Section 4 clearly explains and analyzes the proposed architecture with neat sketches and algorithms and in Section 5 the experimental results are reported and compared with other existing works. Finally the work concludes in Section 6.

II. Related Work

Literature is a significant treasure house of various VLSI architectures for point multiplication in ECC. At this juncture, existing architectures offered in the literature need to be understood. Reza Azarderakhsh and Koray Karabina [13] designed a new double point multiplication algorithm and its application to binary elliptic curves with endomorphism. In this design, the algorithm was based on differential addition chains. The architecture was designed with a uniform structure and has some degree of built-in resistance against side channel

analysis attacks. Their double point multiplication algorithm is based on an adaptation of Montgomery's PRAC algorithm. Work also demonstrated how double point multiplication can be employed to speed up the computation of single point multiplication on elliptic curves with efficiently computable endomorphisms. In design, gain acceleration is 30% and 18% for computing single point multiplication with and without availability of parallel multipliers, respectively.

Efficient elliptic curve point multiplication using digit-serial binary field operations was designed by Gustavo D. Sutter *et.al* [14]. They used a new high-speed point multiplier for elliptic curve cryptography using either field programmable gate array or application-specified integrated circuit technology. Their design adapted a digit-serial approach in GF multiplication and GF division in order to construct an efficient elliptic curve multiplier using projective coordinates. The design involved many basic arithmetic operations in the underlying finite field. There are different acceleration techniques to improve the performance of the ECC operations. Their point multiplication technique used three types of algorithm Montgomery Ladder Algorithm, Point multiplication and Point multiplication using three multipliers and one divisor and precomputing x^{P-1}. This design achieved point multiplication over GF (2^{163}) in 19.38 μs in Virtex-E devices and in 5.48 μs in Virtex-5.

Efficient RNS implementation of elliptic curve point multiplication over GF (p) was designed by Mohammad Esmaeildoust *et.al* [15]. In this design, based on the residue number system (RNS), new hardware architecture for ECPM over GF (p) was established. The designed architecture encompasses RNS bases with various word-lengths to efficiently implement RNS Montgomery multiplication. In that method two versions of fast and area-efficient designs for RNS Montgomery multiplication in six and four-stage pipelined architectures were used. When compared to state-of-the-art implementations, their implemented design achieved higher speeds and better area–delay.

Kimmo U. Järvinen *et.al* [16] suggested efficient algorithm and architecture for elliptic curve cryptography for extremely constrained secure applications. They proposed an efficient implementation of point multiplication on Koblitz curves targeting extremely-constrained, secure applications. In design Gaussian normal basis (GNB) representation of field elements was adopted and employed an efficient bit-level GNB multiplier. The special property of normal basis representation and squarings was rewired in hardware very efficiently. Also, a new technique was introduced for point addition in affine coordinate which required fewer registers. In their newly designed technique extremely small processor architecture for point multiplication was used. Their architecture offered better results compared to the previous works, making it suitable for extremely-constrained, secure environment.

Theoretical modeling of elliptic curve scalar multiplier on LUT-based FPGAs for area and speed efficiency was designed by Sujoy Sinha Roy *et.al* [17]. Two primitives used in elliptic curve scalar multiplier architecture (ECSMA) were implemented on k input lookup table (LUT)-based field-programmable gate arrays to approximate the delay of different characteristic. It was used to determine the optimal number of pipeline stages and the ideal placement of each stage in the ECSMA. In order to perform point addition and doubling in a pipelined data path, suitable scheduling was created. The three stage pipelined architecture for double and add based scalar multiplication is performed on Xilinx Virtex V platforms over GF (2^{163}). The implementation used a novel pipelined bit-parallel Karatsuba multiplier that has subquadratic complexity. In proposed design, efficient choice of scalar multiplication algorithm, optimized field primitives, balanced pipeline stages, and enhanced scheduling of point arithmetic resulted in a high-speed architecture with a significantly small area.

Hossein Mahdizadeh and Massoud Masoumi [18] designed a novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF (2^{163}). In architecture the critical path of the Lopez–Dahab scalar point multiplication architecture was organized and reordered by the maximum architectural and timing improvements, such that logic structures were implemented in parallel and operations in the critical path were diverted to noncritical paths. In the implemented design the execution delay of the LD algorithm has been reduced by parallelization of the multipliers in the implementation of the calculations of projective coordinates. The ECC processor was implemented using synthesizable VHDL codes, and synthesized, placed, and routed using Xilinx ISE 12.1. This design completes the computations in the projective coordinates in 326∗ ([m/G1]) +1304 cycles and coordinate conversion in 15∗([m/G2])+214 cycles. With G1 =33, their new design was four times faster than other designs.

Hybrid binary-ternary number system for elliptic curve cryptosystems was designed by Jithra Adikari *et.al* [19]. The most computational intensive operations in elliptic curve based cryptosystems are Single and double scalar multiplications. The performance of operations was improved by means of integer recoding techniques; with an aim to minimize the scalars density of nonzero digits. Designed system housed three novel algorithms for both single and double scalar multiplications. The first algorithm is w-HBTF and the other two algorithms, namely, HBTJF and RHBTJF. It was used to find the short and sparse representation for a single scalar or a joint representation for a pair of scalars. The output results showed that hybrid algorithms are almost always faster than classical w-NAF methods or JSF.

Kazuo Sakiyama *et.al* [20] implemented a tripartite modular multiplication. In multiplication, for maximizing a level of parallelism, systematic approach was implemented for modular multiplication. The algorithm which is used in this method effectively integrates three different existing algorithms, a classical modular multiplication based on Barrett reduction, the modular multiplication with Montgomery reduction and the Karatsuba multiplication algorithms in order to reduce the computational complexity and increase the potential of parallel processing. In multiprocessor environment for hardware and software implementations, this algorithm is very effective. This algorithm clocks a higher speed when compared to the other algorithms for modular multiplication.

III. PROPOSED METHODOLOGY

Most of the methods implemented for point multiplication include a pre-computation stage before the actual process for point multiplication. The operation of pre-computation stage includes the computation of the intermediate points which are then used for increasing the throughput of the point multiplication process. Hence the need for highly efficient elliptic curve point multiplication is an important activity in the field of cryptography. The traditional and the less complex method for point multiplication is the binary method which is well known as double-and-add method. While, the other double point multiplication algorithm discussed in literature is naive method. But, all these methods only speed up the point multiplication and, since for VLSI architectures the hardware utilization is the major requirement, thus thrust should be on optimizing area required for the proposed system. Thus, in this paper, area efficient implementation of double point multiplication over binary elliptic curves is presented. Area analysis of double point multiplication algorithm based on differential addition chains method is investigated. The performance and efficiency of any scheme is based on the required area. Proposed architecture for double point multiplication is implemented on Xilinx Virtex-4 FPGA device. The proposed architecture is modeled in verilog-HDL and synthesized using Xilinx ISE 14.1 design software.

IV. PROPOSED DOUBLE POINT MULTIPLICATION ALGORITHM

Proposed implementation of Elliptic Curve double point Multiplication algorithm is loosely based on Montgomery's PRAC algorithm [13]. Algorithm is simplified and modified in order to make the design more area efficient than the exiting design. The modified double point algorithm used in proposed work is exhibited in figure 4 as a flowchart.

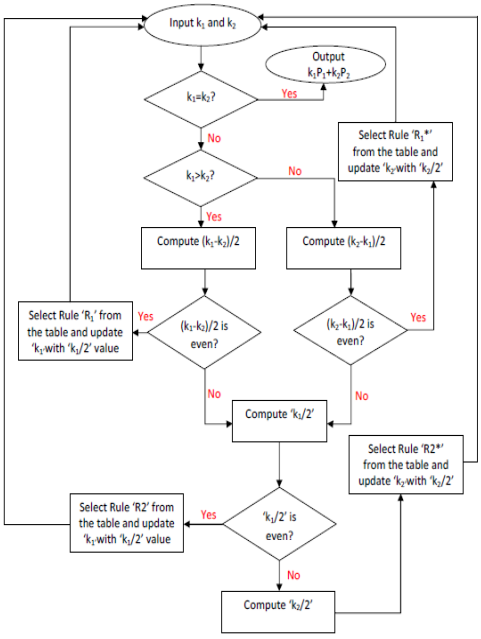

Fig.4. Flowchart for the proposed modified double point multiplication scheme

Let k_1 and k_2 be the two integers such that $k_1 > 0$, and $k_2 > 0$, $k_1, k_2 \in Z$ Where Z is a set of integers. $P_1(X_1, Y_1)$ and $P_2(X_2, Y_2)$ are two points on an Elliptic curve E, such that $P_1, P_2 \in G(2^m)$, Where $G(2^m)$ is Galois Binary extension field. The inputs to proposed double point algorithm are integers k_1, k_2 and the point in the elliptic curve E and the output generated by the algorithm is $k_1P_1 + k_2P_2$ which is another point that lies on the same elliptic graph E such that $k_1P_1 + k_2P_2 \in G(2^m)$. The values of k_1 and k_2 are updated based on conditions mentioned in the flowchart. Whenever the values of k_1 or k_2 are updated, a rule from the Table.I is invoked, which in turn generates a sequence of selector signal for the multiplexers in the architecture. The process continues for various iterations and end up with the output value of double point multiplication $k_1P_1 + k_2P_2 \in G(2^m)$ when k_1 and k_2 becomes equal.

TABLE.I. SELECTOR SEQUENCE AND OPERATIONS BASED ON THE RULES

Rules	Operation			S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
	Register P_1	Register P_2	Register P_d									
Initial	P_1	P_2	P_d	0	0	1	0	0	0	0	0	1
R_1	$2P_1$	$P_{1+}P_2$	P_d	0	1	0	0	1	0	1	0	0
R_2	$2P_1$	P_2	P_1+P_d	0	0	0	0	1	0	0	0	1
R_1*	$P_{1+}P_2$	$2P_2$	P_d	0	1	0	1	0	1	0	1	0
R_2*	P_1	$2P_2$	$P_d+(-P_2)$	1	0	0	1	0	0	0	1	1

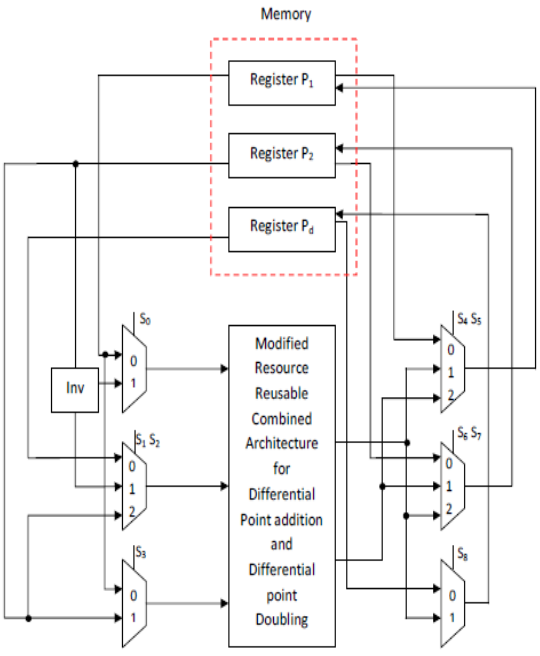

Fig.5. Architecture for Proposed Double point Multiplication unit for ECC

The architecture for proposed area efficient point multiplication scheme using double point multiplication is shown in figure 5. The architecture includes a processing unit, memory unit and a control unit.

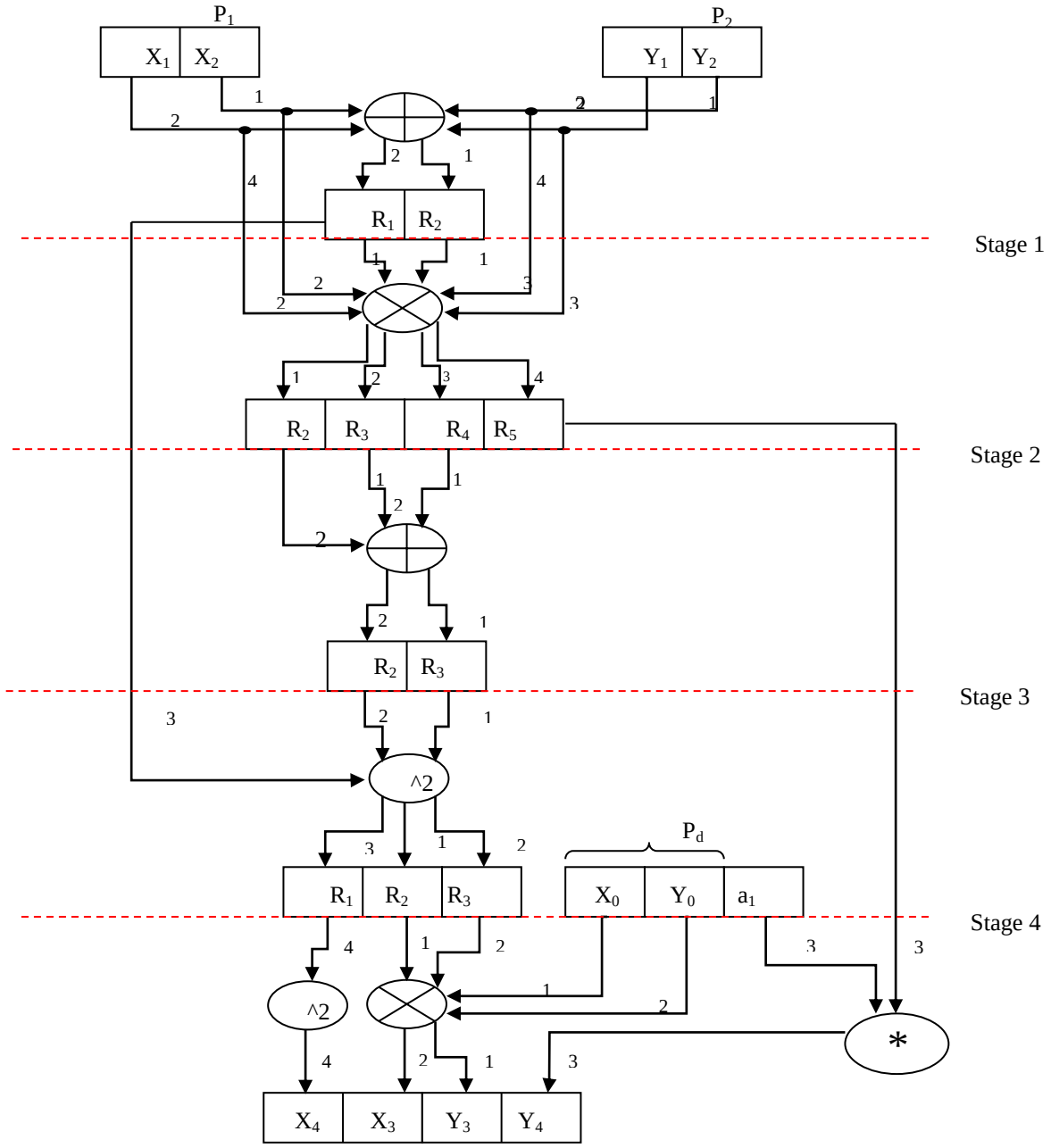

Fig.6. Modified data dependency graph for the processing unit

A. *Processing Unit*

The processing unit is a combined architecture for differential point addition and differential point doubling operations. The major portion of the available slices are occupied by the processing unit because of the involvement of various finite field arithmetic units for computing the output point addition and doubling values. So the main contribution of this work is focused on designing an area efficient processing unit with a reduction in number of incorporated Arithmetic units. The modified area efficient data dependency graph for the processing unit is shown in figure 6.

Proposed data dependency graph for computing double point multiplication employs area efficient finite multipliers, squarers, and adders based on differential point addition and doubling formulae given in [4]. The processing unit is designed with 4 stages of pipeline process in order to reduce the usage of arithmetic units for computation.

The inputs to the processing unit are three points and a difference between two points (the input points values are selected based on the sequence from the control unit). The parameter '*a*' is a constant integer value from the elliptic curve equation considered for cryptography.

When the input is loaded to the processing unit, the processing of the input points takes place in 4 stages. After the completion of the previous stage, the values are stored temporarily in the respective registers (Buffers) and then only the next level of process begins. Hence in proposed architecture, resources such as registers and other arithmetic units that are used in previous stage of process are reused. For example in data flow graph the multiplier used in the first stage of computation can be reused in the stage four and the squarer used in the fourth stage can be reused in the final output computation stage thereby reducing the need for extra multipliers and squarers. The buffers that are used in the previous stage and that are found to be empty in the next stages are reused efficiently for making processing unit area efficient The arithmetic units that are incorporated inside proposed resource reusable combined architecture for differential point addition and differential point doubling are discussed in detail in following sections.

B. Addition Unit

The addition process that takes place in processing unit is a finite field modulo 2 binary additions. Let $A = \sum_{i=0}^{m-1} a_i x^i$ which in binary vector form represented as $A = (a_{m-1}, \ldots\ldots a_1, a_0)$ and $B = \sum_{i=0}^{m-1} b_i x^i$ which in binary vector form represented as $B = (b_{m-1}, \ldots\ldots b_1, b_0)$. Addition of '*A*' and '*B*' produces the result '*C*' as $c_i = a_i \oplus b_i$, Where the symbol ' $\oplus$ ' denotes the 'XOR' operator. Hence in hardware realization of the addition unit, 'XOR' gate array is used as shown in figure 7 for adding two finite field binary elements. The addition process utilizes only one clock cycle for storing the results in the respective output register.

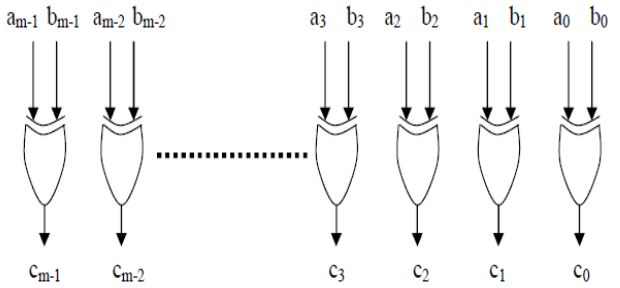

Fig.7. Finite field Adder

C. Squaring Unit

The squaring of an element 'A' in binary finite field is simpler than that of finite field multiplication. Squaring includes two steps of processing; in the first step zeros are inserted between each bit in the bit vector representing 'A' shown in figure 8. In the Second step the bit vector obtained from first step is reduced by taking $\mod f(x)$, where $f(x)$ is a degree-*m* irreducible polynomial. In hardware implementation, reduction can be done by XOR and shifting operation. The squaring operation for 'A' is represented as $A^2 = \sum_{i=0}^{m-1} a_i x^{2^i} \mod f(x)$.

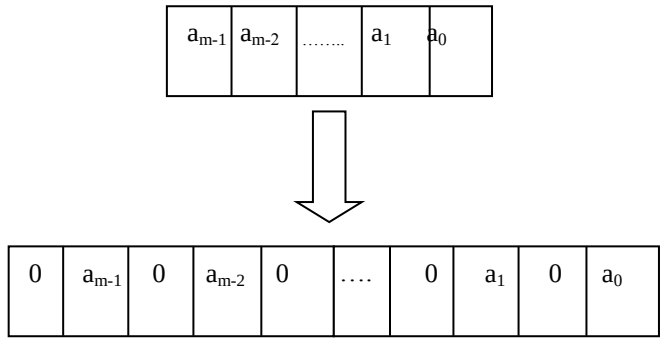

Fig.8. Zero insertion for squaring

D. Multiplication Unit

The design of Finite field multipliers is the complex issue in the implementation of the ECC processor. A number of multipliers with different area and time complexity are reported in the available literatures. In this work, an area efficient architecture for Karatsuba's multiplier which incorporates digit-level polynomial basis multiplier is adopted.

The modified Karatsuba multiplier used in proposed architecture for double point multiplication multiplies 2 finite inputs 'A' and 'B' of m-bit length. In Karatsuba multiplier, each operand is first split into two equal parts and then processed. The internal processor includes 3 multipliers and 4 adders.

The architecture for Karatsuba multiplier is as shown in figure 9.

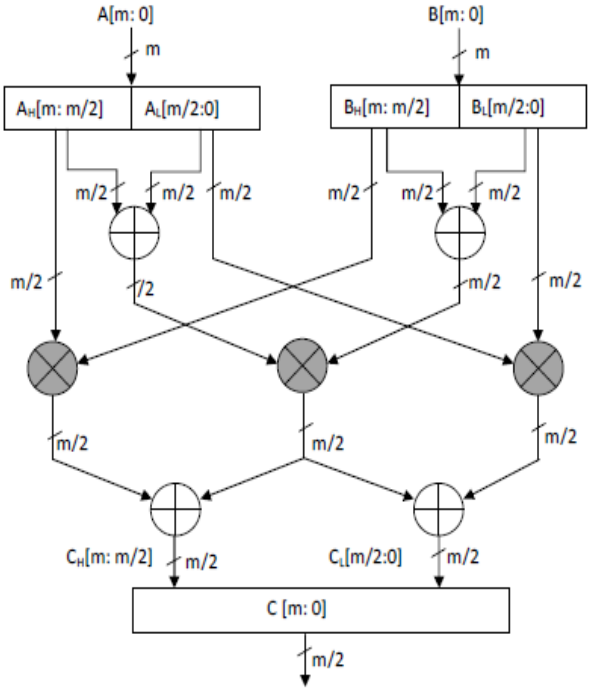

Fig.9. Architecture for $G(2^m)$ Karatsuba multiplier

The multiplier used here is a digit-level polynomial basis multiplier for computing the product of two elements over $G(2^l)$, ($l = m/2$ for this design) using a irreducible polynomial $f(x)$. Hence the input to the digit-level polynomial basis multiplier is of bit length $'m/2'(l = 116)$ and the irreducible polynomial for $G(2^{116})$ is $x^{116} + x^4 + x^2 + 1$ in binary it is represented as (10000……10101). The digit-level multiplier exhibits an area complexity of $O(md)$ and time complexity of $O(m/d)$

The operand '*A*' register is initially loaded with *l*-bits and operand '*B*' register is loaded with *l*-bits. The *D*-block is an array of AND gates as shown in figure 10 which performs the operation $a_i \cdot B$. Hence the output of the D-block is available only if the bit value of the corresponding A-register is '1' and if it is '0' then the output of *D*-block becomes '0'. When all the d-partial products are computed the $* x^i$ blocks perform corresponding shift operations and the $\bmod f(x)$ block performs reduction operation. The $G(2^l)$ adder block add all the partial products obtained in the before step using a array of 'XOR' gates same as that have been used for addition operation for field elements. The main advantage of using this multiplier in proposed technique is that it can operate at higher clock frequencies in comparison to the other multipliers reported in the literature. The architecture for the digit-level polynomial basis multiplier used in proposed technique is shown in figure 11.

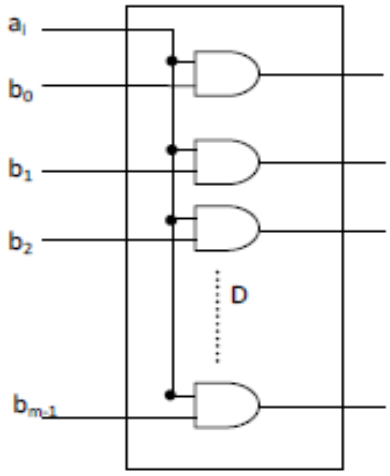

Fig.10. Internal Structure of D-block

E. Inversion Unit

Inversion is the most expensive arithmetic finite field operations. In general, inversion can be computed as,

$$A^{-1} = A^{2m-2}$$

The general computation of inversion requires $m-1$ squaring and $m-2$ multiplications. But the method proposed by Itoh and Tsujii (IT) [20] is the most efficient method of inversion and hence same technique has been adopted for hardware implementation of inversion module in proposed architecture. The IT technique requires only $\lfloor \log_2(m-1) \rfloor + H(m-1) - 1$ and $m-1$ squaring. Where H is the number of ones in the binary representation of $m-1$ bits known as '*Hamming Weight'*. The algorithm of Itoh and Tsujii method for inversion is given in Algorithm 1.

Let, $A \in G(2^5), A \neq 0, m = 2^2 + 1 = 5 \Rightarrow s = 2$

Initially $\alpha(A) = A and\ i = 1\ since (s - 1 = 2 - 1 = 1)$

When $\boldsymbol{i = 0}$, $(A)^{2^{2(1)}} = A^4$ $(2\,Squaring)$

$$A = A^4 . A\,(1\,Finite Field\ Multiplication) = \boldsymbol{A^5 = A^{-1}}$$

Algorithm 1

Input $: \boldsymbol{A \in G(2^m), A \neq 0, m = 2^s + 1}$
Output $: \boldsymbol{A^{-1}}$
1. $\boldsymbol{\alpha(A) = A}$
2. ***For*** $\boldsymbol{i = 1}$ ***to*** $\boldsymbol{s - 1}$ ***do***
begin
3. $\boldsymbol{\beta(A) = \alpha\ (A)^{2^{2i}}}$ $(2^i$ ***cyclic shifts***$)$
4. $\boldsymbol{\alpha(A) = \alpha(A) * \beta(A)}$ $(\boldsymbol{Finite Field Multiplication G(2^m))}$
end
5. $\boldsymbol{\alpha(A) = \alpha(A)^2}$ $(\boldsymbol{Finite Field Multiplication G(2^m))}$
Return $(\boldsymbol{\alpha(A)})$

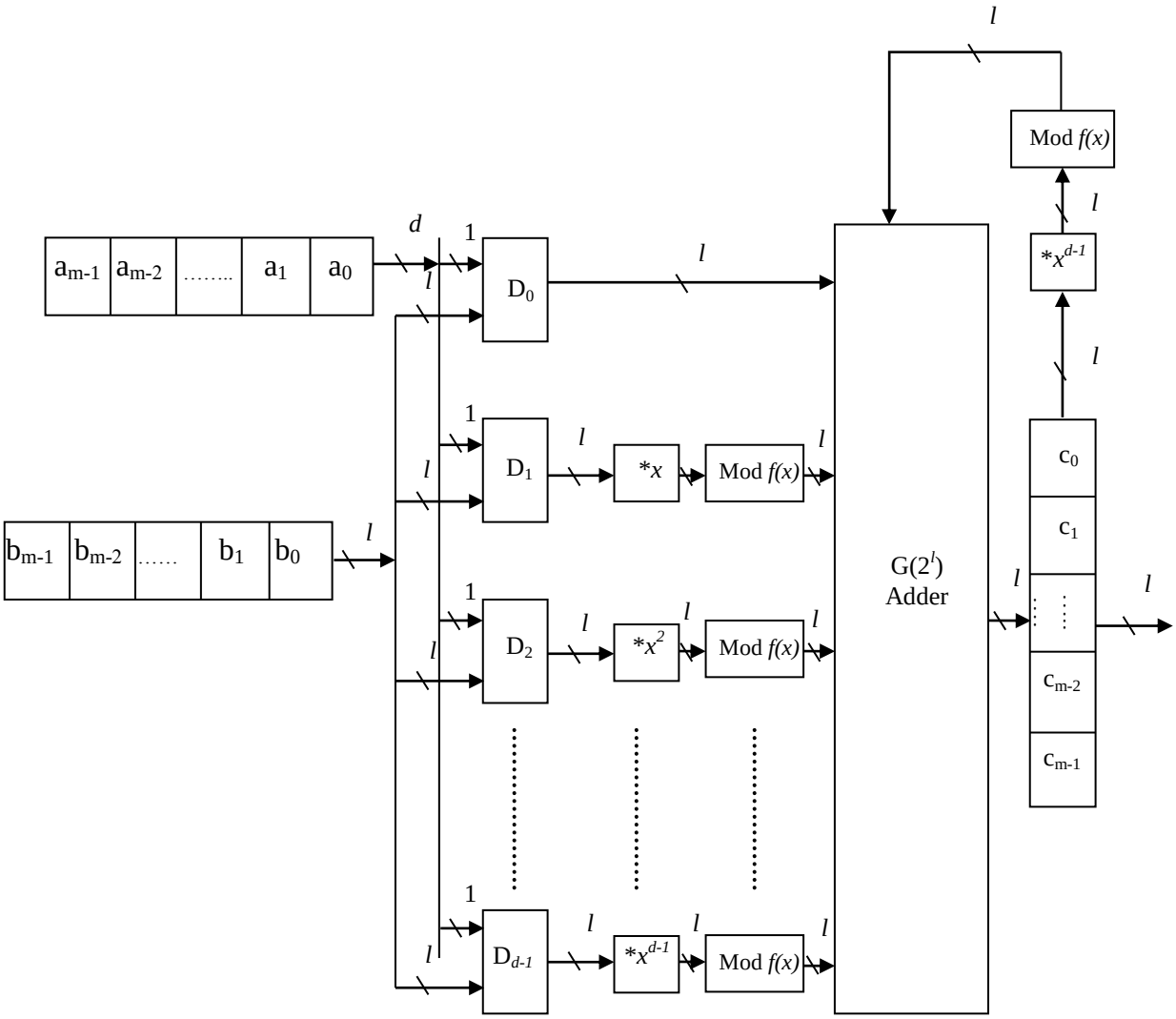

Fig.11. Digit-level polynomial basis multiplier

F. Control and Memory Unit

The control unit designed with LUTs generates the control signals as per the input rule and flow chart given in figure 1. Based on the input rule, appropriate selector signals are generated and are fed to the multiplexers. For each clock pulse, the selectors signals are generated and based on this the contents from the registers are fed to the processing unit. At the start of the process, the Register P_1 and Register P_2 are initialized with the input points P_1 and P_2 respectively. The designed control unit is simple and utilizes only a smaller area than the other units in the architecture.

The block diagram for the control unit is shown in figure 12. For storing the points and all other data needed for the computation, register files are used instead of RAM blocks. This is because the RAM blocks require communication between the memory unit and the processing unit which is not required in case of the register files.

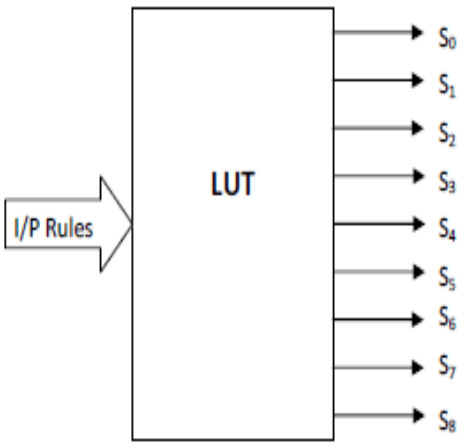

Fig.12. Control Unit

V. Results and Discussion

In this section, proposed architecture for double point multiplication is implemented to analyze its area and power requirements.

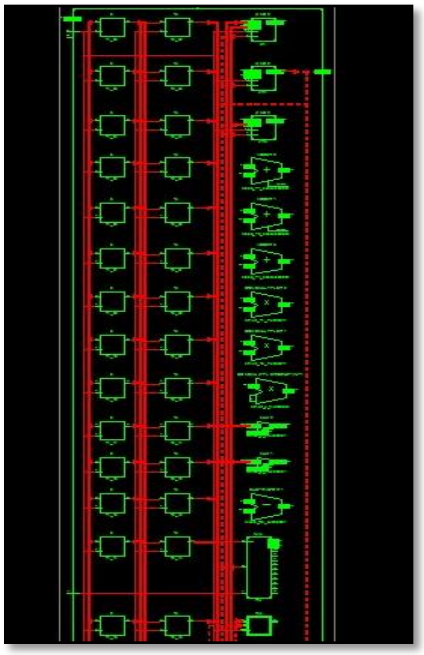

Fig.13. RTL schematic for the proposed double point multiplication architecture

The Xilinx® Virtex™-4 xc4vlx200 device is used as the target FPGA. The proposed architecture is modeled in verilog-HDL and synthesized for different digit sizes using XST™ of Xilinx® ISE™ version 14.1 design software. All the experiments were performed on 3.10GHz Intel(R) i5, 4.00GB RAM, and 32-bit operating system with windows7 professional. Figure 13 exhibits a snapshot of RTL schematic of proposed double point multiplication architecture.

A. Area report of proposed scheme:

Comparison of area utilization of proposed double point multiplication architecture with other existing methods such as Naive Method, B-NBC, JT -{±1, ±3} and AK-DAC is carried out. Target device includes 89,088 Slices (178,176 4 input LUTs and 178,176 Sliced FFs) and 960 bonded IOBs. Each slice contains 2 flip-flops (FFs) and 2 look-up tables (LUTs). The resource utilization comparison as depicted in table II below shows the slices utilized by proposed scheme are much lesser than the other existing methods. Proposed implementation utilizes only an average of 6.5% among the available 89,088 slices in the device. But all other methods report a high percentage of device utilization.

With increase in'd' value from 7 to 13, it is observed that there is increase in proposed architecture footprint. The area comparison of proposed method with other similar existing methods is shown in figure 14 for a clear understanding of the efficiency of proposed method. For digit size of 7, proposed architecture uses 37.95% reduced slices as compared to Naive method, 25.9% fewer in comparison to B-NBC , 8% fewer slices as compared to JT –{±1, ±3} method and 6% lesser than AK-DAC technique.

TABLE.II. Area Comparisons of Different Double Point Multiplication Algorithms over $G(2^m)$ with $m = 233$

d	Naïve Method[13] [*#Slices*]	B-NBC[12] [*#Slices*]	JT –{±1, ±3}[9] [*#Slices*]	AK-DAC[13] [*#Slices*]	Proposed [*#Slices*]
7	6,218	5,207	4,196	4,146	3,858
13	9,693	8,117	6,541	6,462	6,146
18	11,335	9,492	7,649	7,557	6,887
26	16,612	13,911	11,210	11,075	8,733

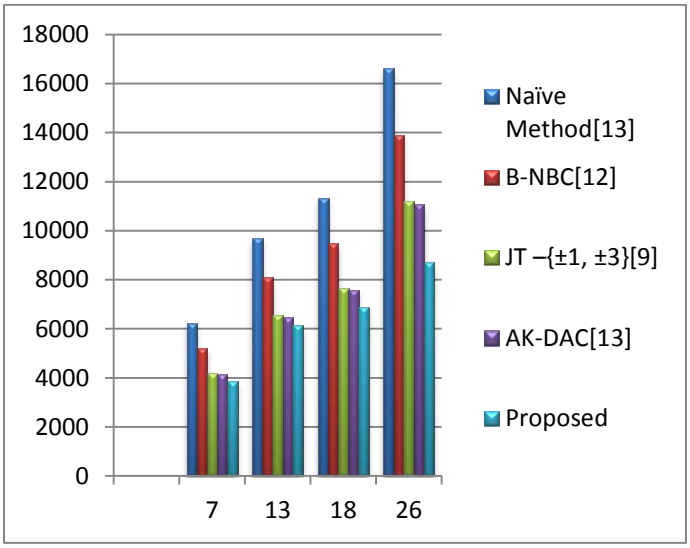

Fig.14. Bar chart showing Area comparison

For digit size of 26, proposed architecture uses 47.42% reduced slices as compared to Naive method, 37.22 % in comparison to B-NBC, 22.09% fewer slices as compared to JT –{±1, ±3} method and 21.14% lesser than AK-DAC technique

B. Power and Performance Report of proposed architecture

The total clock periods required for the computation, frequency and power needed for the implemented architecture is tabulated in table III below.

TABLE.III. Implementation Results for Our Proposed Double Point Multiplication Scheme

d	Clock period(*ns*)	Frequency(*MHz*)	Power(*W*)
7	30.673	32.602	1.420
13	35.473	28.191	1.429
18	39.052	25.607	1.479
26	44.058	22.698	1.503

Figure 15 shows the graph plotted between 'd' along x-axis and Clock periods along y-axis. With the increase in digit size of the multiplier, the clock periods (Computation Time) increases. Hence a large digit size multiplier can boost up the throughput of architecture. But with the increase in digit size the need for registers, AND gates XOR gates and shift logic also increases which contributes to chip area. Since thrust is on area reduction, a low bit size for implementation has been chosen.

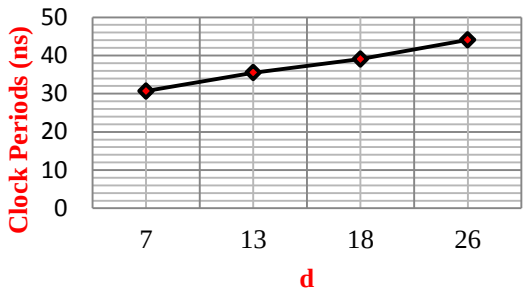

Fig.15. Graph plot between'*d*' and 'Clock Periods (*ns*)'

From the graph shown in figure 16, it is observed that with the rise in the digit size of the multiplier, the operating clock frequency for implementation decreases.

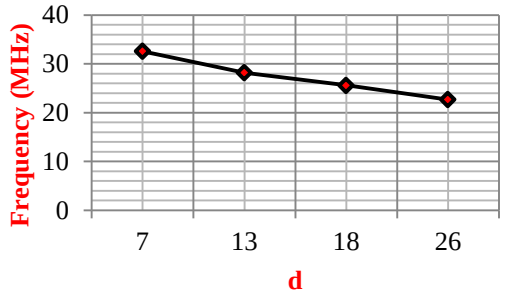

Fig.16. Graph plot between'*d*' and 'Frequency (*MHz*)'

The power consumption by proposed design is mainly due to the leakage power and the clock power. Figure 17 shows the graph plot for digit size (*d*) Vs Total Power consumption by proposed module. It can be observed that with the increase in digit size of the multiplier, the power consumed by architecture increases. From the above analysis, if '*d*' value is selected as a low value then Area and Power consumption decreases but the speed of computation decreases. On the other hand if 'd' value is made high then the area and power consumption increases with a high speed computation. Hence in order to make proposed architecture efficient towards Area, Power and performance, a balanced value of digit size is to be chosen and is set to an average value as possible.

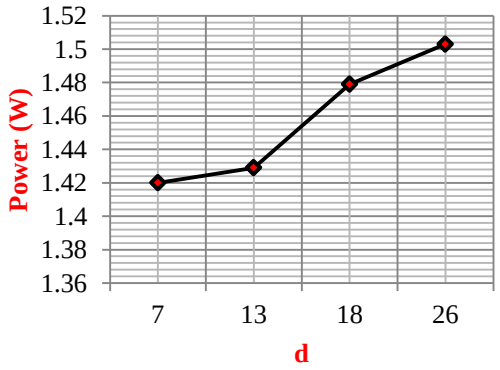

Fig.17. Graph plot between'*d*' and 'Power (*W*)'

VI. Conclusion

In this paper, an area efficient elliptic curve point multiplication architecture using a double point multiplication technique is designed and implemented. Reutilization of idle resources and a pipelined data path scheme for data processing in the combined module for differential point adder and point doubler were presented clearly. The finite field arithmetic operators were designed efficiently to reduce the area utilization. The complete architecture was synthesized and simulated using Xilinx ISE 14.1. Reports were generated in terms of area, power and time by varying the digit size of the multiplier. The results obtained from area report were compared with other similar existing methods reported in the literatures and found to be much better. In future for further area optimization of the proposed architecture, research thrust should be on designing an efficient area optimized finite field multiplier.

References

[1] J.M. Pollard. Monte Carlo, "Methods for Index Computation (mod p)" Mathematics of computation, 32(143):918–924, 1978.

[2] J-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems", Lecture Notes in Computer Science, CHES 1999, 1717:292–302, 1999.

[3] C. Clavier and M. Joye, "Universal Exponentiation Algorithm – A First Step towards Provable SPA-Resistance", Lecture Notes in Computer Science, CHES 2001, 2162:300–308, 2001.

[4] R. Gallant, R. Lambert, and S. Vanstone, " Faster point multiplication on elliptic curves with efficient endomorphisms", Advances in Cryptology – CRYPTO 2011, LNCS, 2139:190–200, 2001.

[5] D. Galbraith, X. Lin, and M. Scott, " Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves", Journal of Cryptology, 24:446–469, 2011.

[6] D. Hankerson, K. Karabina, and A. Menezes, "Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields", IEEE Transactions on Computers, 58:1411–1420, 2009.

[7] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", New York, 1996.

[8] B. Moller, "Algorithms for Multi-exponentiation", Selected Areas in Computer Science SAC 2001, LNCS, 2259:165–180, 2001.

[9] M. Joye and M. Tunstall, " Exponent recoding and regular exponentiation algorithms", Lecture Notes in Computer Science, AFRICACRYPT 2009, 5580:334–349, 2009.

[10] P. Montgomery, "Evaluating recurrences via Lucas chains", www.cwi.nl/ftp/pmontgom/Lucas.ps.gz, December 13, 1983; Revised March, 1991 and January, 1992.

[11] T. Akishita, "Fast Simultaneous Scalar Multiplication on Elliptic Curve with Montgomery Form", Selected Areas in Computer Science SAC 2001, LNCS, 2259:225–267, 2001.

[12] D. Bernstein, "Differential addition chains", Technical report, 2006, Available at http://cr.yp.to/ecdh/diffchain-20060219.pdf.

[13] Reza Azarderakhsh and Koray Karabina,"A new double point multiplication algorithm and its application to binary elliptic curves with endomorphisms", IEEE Transactions on Computers, No.99, May 2013.

[14] Sutter.G.D, Deschamps.J and Imana J.L , "Efficient elliptic curve point multiplication using digit-serial binary field operations", IEEE Transactions on industrial electronics, Vol.60, No.1, Jan 2013.

[15] Mohammad Esmaeildoust, "Efficient RNS implementation of elliptic curve point multiplication over GF(p)", IEEE Transactions on Very Large Scale Integration systems, Vol. 21, No. 8, Aug 2013.

[16] Azarderakhsh. R, Jarvinen K.U and Mozaffari-Kermani. M,"Efficient algorithm and architecture for elliptic curve cryptography for extremely constrained secure applications", IEEE Transactions on circuits and systems- I , Vol. 61, No. 4, April 2014.

[17] Roy. S.S, Rebeiro,C and Mukhopadhyay. D, "Theoretical modeling of elliptic curve scalar multiplier on LUT-based FPGAs for area and speed", IEEE Transactions on Very Large Scale Integration (VLSI) systems, Vol. 21, No. 5, May 2013.

[18] Hossein Mahdizadeh and Massoud Masoumi, "A novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF(2^{163})",IEEE Transactions on very large scale integration systems, Vol. 21, No. 12, Dec 2013.

[19] Adikari. J, Dimitrov.V.S, and Imbert.L,"Hybrid binary-ternary number system for elliptic curve cryptosystems", IEEE Transactions on computers, Vol. 60, No. 2, Feb 2011.

[20] Kazuo Sakiyama, Miroslav Knezevica, Junfeng Fana, , Bart Preneela, and Ingrid Verbauwhedea, "Tripartite modular multiplication", Integration, the VLSI Journal, Vol. 44, No.4, pp: 259–269, September 2011.