Design and Implementation of an Efficient RSD-Based ECC Processor

G.Sindhuja[1] gsindhu14@gmail.com [1]

K. Kishore Kumar[2] kishorekumar20@gmail.com[2]

[1]PG Scholar, VLSI, Dr.K.V.Subba Reddy Institute of technology , Lakshmipuram, Kurnool.

[2]Assistant Professor, Dept of ECE, Dr.K.V.Subba Reddy Institute of technology , Lakshmipuram, Kurnool.

Abstract: **In this paper, an exportable application-specific instruction-set elliptic curve cryptography processor based on redundant signed digit representation is proposed. The processor employs extensive pipelining techniques for Karatsuba–Ofman method to achieve high throughput multiplication. Furthermore, an efficient modular adder without comparison and a high throughput modular divider, which results in a short datapath for maximized frequency, are implemented. The proposed architecture of this paper analysis the logic size, area and power consumption using Xilinx 13.2. The extension for the project is Vedic Sutra – Urdhwa Tiryakbhyam.**

Index Terms**— Application-specific instruction-set processor (ASIP), elliptic curve cryptography (ECC), field-programmable gate array (FPGA), Karatsuba–Ofman multiplication, redundant signed digit (RSD).**

I.INTRODUCTION

In prime field ECC processors, carry free arithmetic is necessary to avoid lengthy datapaths caused by carry propagation. Redundant schemes, such as carry save arithmetic (CSA), redundant signed digits (RSDs) , or residue number systems (RNSs) , have been utilized in various designs. Carry logic or embedded digital signal processing (DSP) blocks within field programmable gate arrays (FPGAs) are also utilized in some designs to address the carry propagation problem. It is necessary to build an efficient addition data path since it is a fundamental operation employed in other modular arithmetic operations. Modular multiplication is an essential operation in ECC.

Two main approaches may be employed. The first is known as interleaved modular multiplication using Montgomery's method. Montgomery multiplication is widely used in implementations where arbitrary curves are desired. Another approach is known as multiply then-reduce and is used in elliptic curves built over finite fields of Merssene primes. Merssene primes are the special type of primes which allow for efficient modular reduction through series of additions and subtractions. In order to optimize the multiplication process, some ECC processors use the divide and conquer approach of Karatsuba–Ofman multiplications, where others use embedded multipliers and DSP blocks within FPGA fabrics.

This paper proposes a new RSD-based prime field ECC processor with high-speed operating frequency. In this paper, we demonstrate the performance of left-to-right scalar point multiplication algorithm. The overall processor architecture is of regular cross bar type with 256 digit wide data buses. The design strategy and optimization techniques are focused toward efficient individual modular arithmetic modules rather than the overall architecture.

The remaining of this paper is organized as follows. Section II provides background information on ECC systems. Section III presents the overall architecture of the proposed processor, the architecture of the modular arithmetic unit (AU) is presented. In Section IV, extension of the project is discussed. Finally, Results and conclusion is drawn in Section V and Section VI.

II. RELATED WORK

Karatsuba–Ofman Multiplication:

The complexity of the regular multiplication using the schoolbook method is $O(n^2)$. Karatsuba and Ofman proposed a methodology to perform a multiplication with complexity $O(n^{1.58})$by dividing the operands of the multiplication into smaller and equal segments. Having two operands of length n to be multiplied, the Karatsuba–Ofman methodology suggests to split the two operands into high-(H)and low-(L) segments.

$$a_H = (a_{n-1}, \ldots, a_{\lceil n/2 \rceil}), \quad a_L = (a_{\lceil n/2 \rceil - 1}, \ldots, a_0)$$
$$b_H = (b_{n-1}, \ldots, b_{\lceil n/2 \rceil}), \quad b_L = (b_{\lceil n/2 \rceil - 1}, \ldots, b_0).$$

Consider β as the base for the operands, where β is 2 in case of integers and β is x in case of polynomials. Then, the multiplication of both operands is performed as follows: considering

$a = a_L + a_H \beta^{\lceil n/2 \rceil}$ and $b = b_L + b_H \beta^{\lceil n/2 \rceil}$ then

$$\boxed{\begin{aligned} C = AB &= (a_L + a_H \beta^{\lceil n/2 \rceil})(b_L + b_H \beta^{\lceil n/2 \rceil}) \\ &= a_L b_L + (a_L b_H + a_H b_L)\beta^{\lceil n/2 \rceil} + a_H b_H \beta^n. \end{aligned}}$$

Hence, four half-sized multiplications are needed, where Karatsuba methodology reformulate (6) to

$$\begin{aligned} C = AB &= (a_L + a_H \beta^{\lceil n/2 \rceil})(b_L + b_H \beta^{\lceil n/2 \rceil}) \\ &= a_L b_L \\ &\quad + ((a_L + a_H)(b_L + b_H) - a_H b_H - a_L b_L)\beta^{\lceil n/2 \rceil} \\ &\quad + a_H b_H \beta^n. \end{aligned}$$

Therefore, only three half-sized multiplications are needed. The original Karatsuba algorithm is performed recursively, where the operands are segmented into smaller parts until a reasonable size is reached, and then regular multiplications of the smaller segments are performed recursively.

Redundant Signed Digits:

The RSD representation, first introduced by Avizienis [32], is a carry free arithmetic where integers are represented by the difference of two other integers. An integer X is represented by the difference of its x+ and x− components, where x+ is the positive component and x− is the negative component. The nature of the RSD representation has the advantage of performing addition and subtraction without the need of the two’s complement representation. On the other hand, an overhead is introduced due to the redundancy in the integer representation, since an integer in RSD representation requires double word length compared with typical two’s complement representation. In radix-2 balanced RSD represented integers, digits of such integers are either 1, 0, or −1.

III.PROPOSED METHODOLOGY

The proposed P256 ECC processor consists of an AU of 256 RSD digit wide, a finite-state machine (FSM), memory, and two data buses. The processor can be configured in the pre-synthesis phase to support the P192 or P224 NIST recommended prime curves [36]. Fig. 1 shows the overall processor architecture. Two sub control units are attached to the main control unit as add-on blocks. These two sub control units work as FSMs for point addition and point doubling, respectively. Different coordinate systems are easily supported by adding corresponding sub control blocks that operate according to the formulas of the coordinate system.

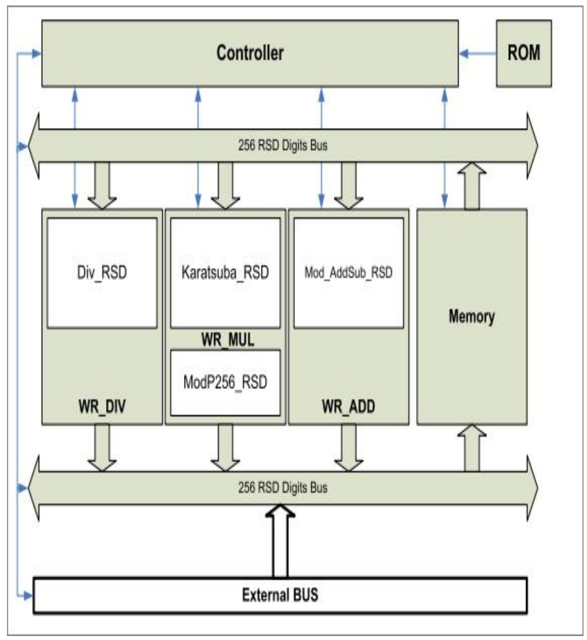


Fig. 1. Overall processor architecture.

ARITHMETIC UNIT.

Modular Addition and Subtraction Addition is used in the accumulation process during the multiplication, as well as, in the binary GCD modular divider algorithm. In the proposed implementation, radix-2 RSD representation system as carry free representation is used. In RSD with radix-2, digits are represented by 0, 1, and −1, where digit 0 is coded with 00, digit 1 is coded with 10, and digit −1 is coded with 01. In Fig. 2, an RSD adder is presented that is built from generalized full adders.

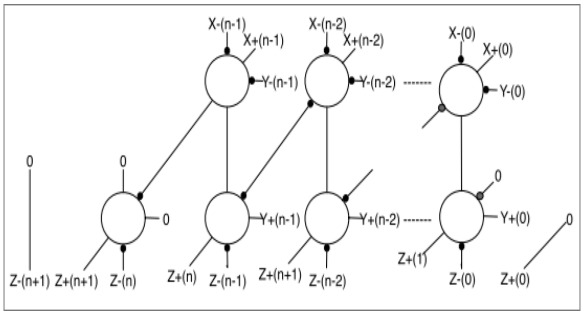


Fig. 2. RSD adder.

Modular Multiplication

Karatsuba's multiplier recursive nature is considered a major drawback when implemented in hardware. Hardware complexity increases exponentially with the size of the operands to be multiplied. To overcome this drawback, Karatsuba method is applied at two levels. A recursive Karatsuba block that works depth wise, and an iterative Karatsuba that works widthwise.

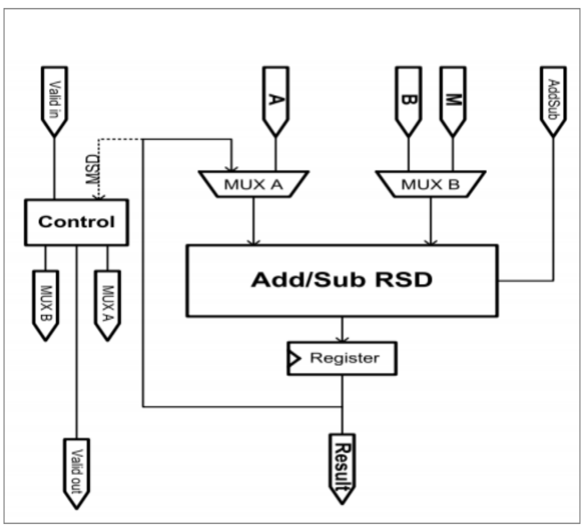


Fig. 3. Modular addition subtraction block diagram.

The block diagram of the recursive Karatsuba multiplier is shown in Fig. 4, where data dependences are clearly noticed. As shown in Fig. 4, Karatsuba method requires performing a subtraction at every level, which is an advantage of the proposed implementation since subtraction is performed with no added cost in RSD representation. The block diagram of the recursive Karatsuba module is built from three half-sized recursive Karatsuba blocks and some RSD adders/subtracters. There is one 1-digit RSD multiplier that is used to multiply the carry digits from the middle addition. According to Fig. 4, the critical datapath of the recursive Karatsuba is divided into two paths. The first path goes through the middle half-sized recursive Karatsuba block, and the other goes through the cross product of the middle addition with multiplexers and some adders.

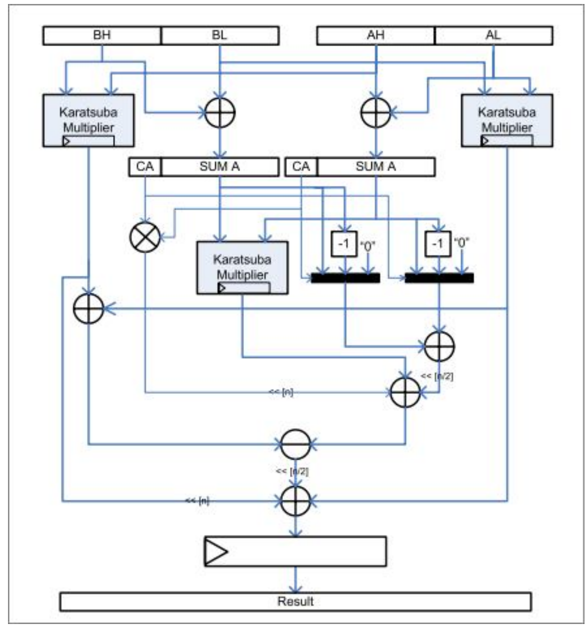


Fig. 4. Karatsuba recursive block

NIST Reduction: Generalized Mersenne primes [19] are the special type prime numbers that allow fast modular reduction. Regular division is replaced by few additions and subtractions. Such primes are represented as p = f (t), where t is a power of 2. The modulus of the P256 curve is Merssene prime $p = 2^{256}-2^{224}+2^{192}+2^{96}-1$.

Due to the redundancy nature of the RSD representation, the multiplication process may produce results that are represented by more than 512 digits and these results are still in the range $-p^2 < A < p^2$. These one or two extra digits are outside the range of the NIST reduction process. Hence, we derived new formulas to include these extra digits in the reduction process. The new reduction process has one extra 256-digit term, D5, along with some modification of the previously existed terms. This term is added conditionally, whether the extra digit is set or not. Thus, two additions are the total overhead required to handle the extra digits caused using the

RSD representation. The modified reduction formula is B = T + 2S1 + 2S2 + S3 + S4 − D1 − D2 − D3 − D4 − D5 mod p, where A16 represents the extra digits produced by RSD Karatsuba multiplier.

$$
\begin{aligned}
T &= (A_7\|A_6\|A_5\|A_4\|A_3\|A_2\|A_1\|A_0)\\
S_1 &= (A_{15}\|A_{14}\|A_{13}\|A_{12}\|A_{11}\|\ 0\ \|\ 0\ \|\ 0\ )\\
S_2 &= (\mathbf{2*A_{16}}\|A_{15}\|A_{14}\|A_{13}\|A_{12}\|\ 0\ \|\ 0\ \|\mathbf{A_{16}})\\
S_3 &= (A_{15}\|A_{14}\|\ 0\ \|\ 0\ \|\mathbf{-2*A_{16}}\|A_{10}\|A_9\|A_8)\\
S_4 &= (A_8\|A_{13}\|A_{15}\|A_{14}\|A_{13}\|A_{11}\|A_{10}\|A_9)\\
D_1 &= (A_{10}\|A_8\|\ 0\ \|\ 0\ \|\mathbf{2*A_{16}}\|A_{13}\|A_{12}\|A_{11})\\
D_2 &= (A_{11}\|A_9\|\ 0\ \|\mathbf{A_{16}}\|A_{15}\|A_{14}\|A_{13}\|A_{12})\\
D_3 &= (A_{12}\|\mathbf{2*A_{16}}\|A_{10}\|A_9\|A_8\|A_{15}\|A_{14}\|A_{13})\\
D_4 &= (A_{13}\|\ 0\ \|A_{11}\|A_{10}\|A_9\|\mathbf{A_{16}}\|A_{15}\|A_{14})\\
\mathbf{D_5} &= (\mathbf{-A_{16}}\|\ \mathbf{0}\ \|\ \mathbf{0}\ \|\ \mathbf{0}\ \|\ \mathbf{0}\ \|\ \mathbf{0}\ \|\ \mathbf{0}\ \|\mathbf{-A_{16}}).
\end{aligned}
$$

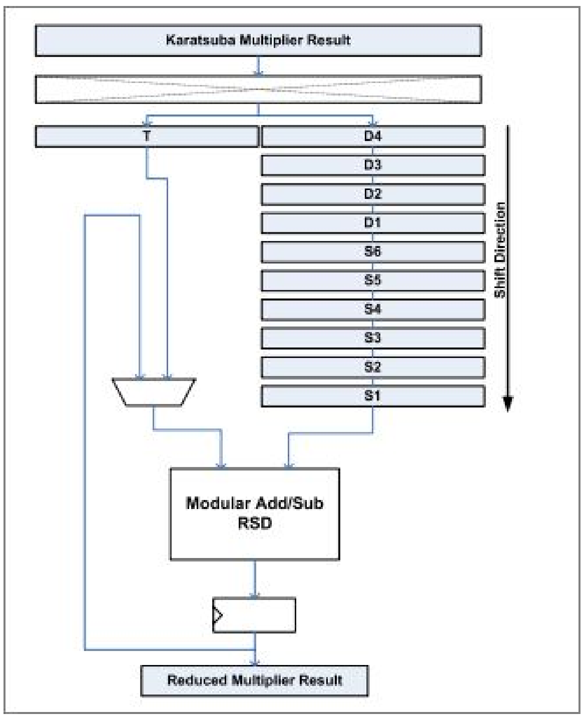


Fig. 5. Mod P256 reduction block.

In order to accommodate the extra digit produced by the RSD Karatsuba multiplier, NIST reduction is reformulated. The resultant reduction scheme consists of three extra additions. However, through reformulation and combining the original terms with the additional terms, the reduction scheme is optimized. Accordingly, the modular multiplier is built with a Karatsuba multiplier, modular RSD adder, and some registers to hold the 256-digit terms. Fig. 5 shows the block diagram of the Mod P256 RSD multiplier. A controller is used to control the flow of the terms to the modular adder and at every turn, the result of the modular addition is accumulated and fed back to the adder. The cross-bar in Fig. 5 shows the wiring of the 32-digit words to their respective locations within the extended NIST reduction registers.

High-Radix Modular Division

Binary GCD algorithm is an efficient way of performing modular division since it is based on addition, subtraction, and shifting operations. The complexity of the division operation comes from the fact that the running time of the algorithm is inconsistent and is input dependent.

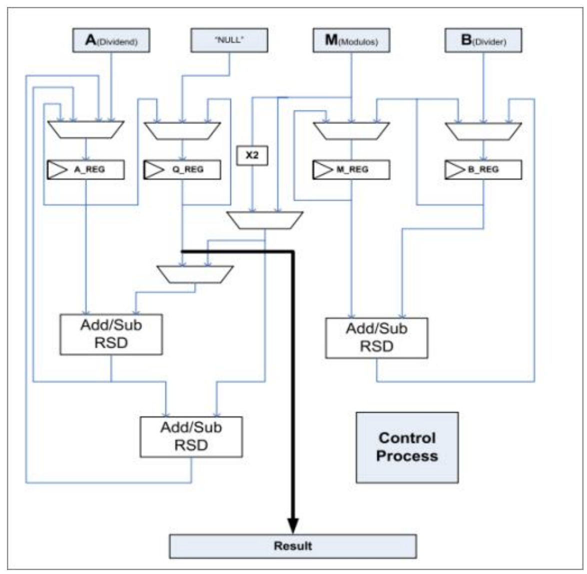


Fig. 6. Modular divider block

IV. Vedic Sutra – Urdhwa Tiryakbhyam

In proposed system we tend to area unit measurement Input Adder Unit, currently it is replaced by sacred text multiplier factor. By doing this we are able to get less power consumption, high accuracy and reduced delay.

The sixteen sacred text Sutras apply to and canopy nearly each branch of arithmetic. They apply even to advanced issues involving an oversized variety of mathematical operations. Among these sutras, Urdhwa Tiryakbhyam Sanskrit literature is that the best for acting multiplication. The use of this Sanskrit literature will be extended to binary multiplication as well. This Sanskrit literature interprets to "Vertical and crosswise". It utilizes solely logical AND operation, 0.5 adders and full adders to perform multiplication wherever the partial merchandise area unit generated before actual multiplication. this protects a substantial quantity of time interval. What is more it's a sturdy methodology of multiplication. Consider 2 8-bit numbers, a (a8-a1) and b (b8-b1) wherever one to eight represents bits from the least important bit to the most important bit. the ultimate Product is represented by P (P16-P1). In Fig.7, the step by step methodology of multiplication of 2 8-bit numbers using Urdhwa Tiryakbhyam sutra is illustrated. The bits of the number and number area unit diagrammatic by dots and also the 2 approach are represents the logical AND operation between the bits that provides the partial product terms. In the typical style of Urdhwa Tiryakbhyam sutra based mostly number, solely full-adders and half-adders area unit used for addition of the partial products. But, the aptitude of full-adder is restricted to addition of solely three bits at a time.

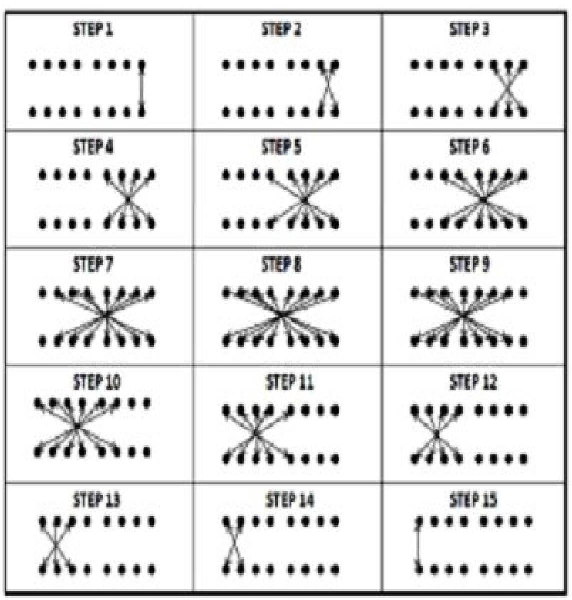


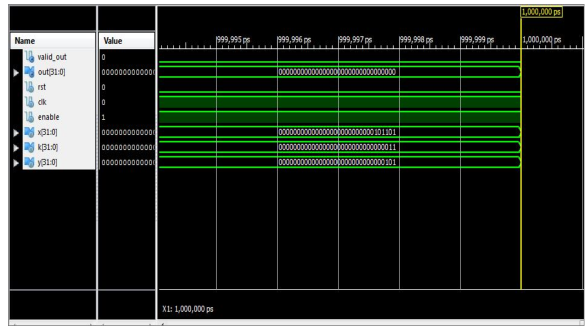
Fig.7. 8-bit binary multiplication using Urdhwa Tiryakbhyam Sutra

So, a large number of stages are required to get the final product. Higher order compressors discussed in next section can be employed to add more than 3 bits at a time (upto 7 bits) and hence can reduce the intermediate stages.

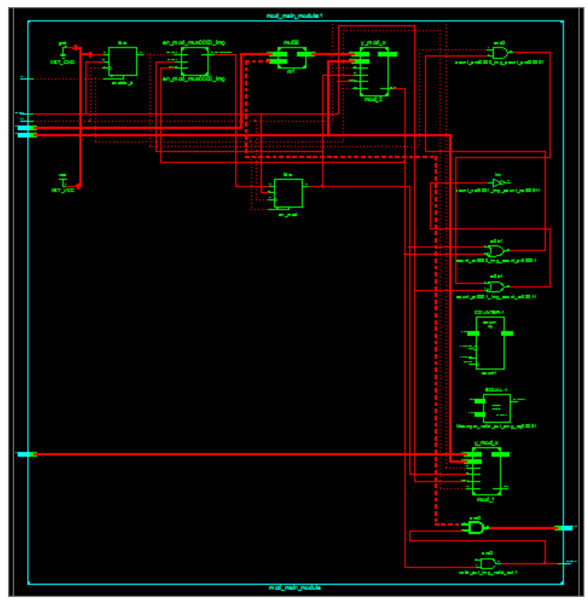
V. EXPERIMENTAL RESULTS

Results of proposed method

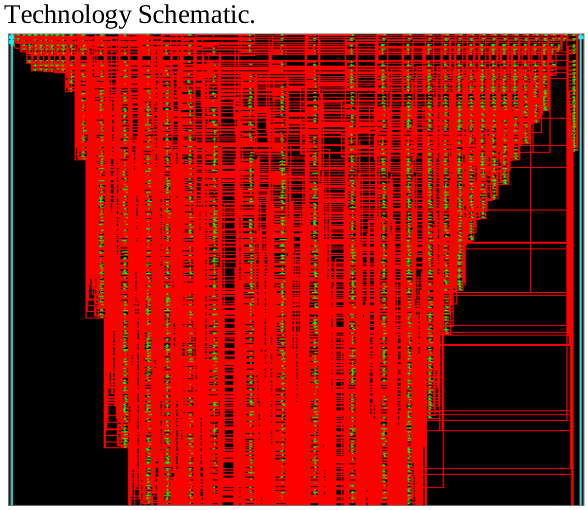
Simulation.



RTL Schematic.



Technology Schematic.



Design Summary.

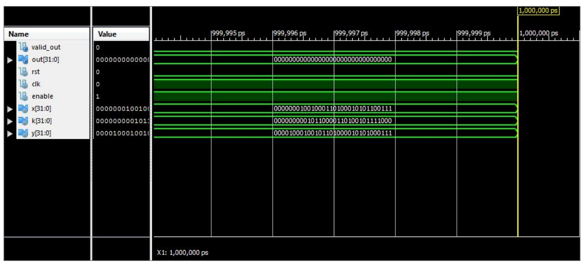
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	930	4656	19%
Number of Slice Flip Flops	177	9312	1%
Number of 4 input LUTs	1741	9312	18%
Number of bonded IOBs	132	232	56%
Number of GCLKs	1	24	4%

Timing Report.

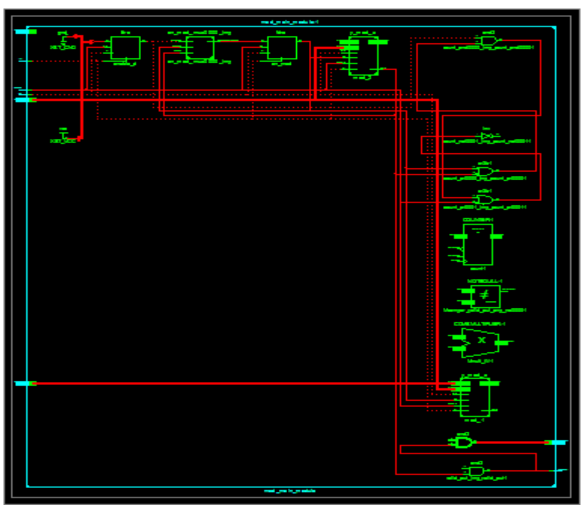
```
================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 2177 / 33
--------------------------------------------------------------------------------
Offset:              8.440ns (Levels of Logic = 19)
  Source:            mod_2/z_0 (FF)
  Destination:       valid_out (PAD)
  Source Clock:      clk rising
```

Extension.

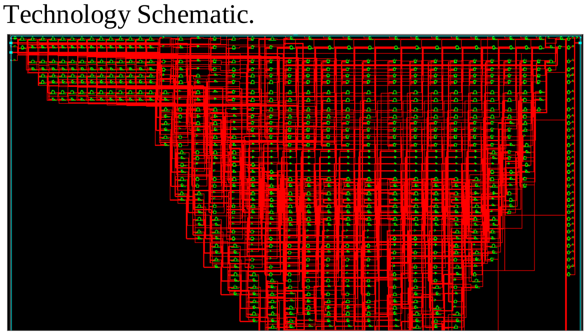
Simulation.



RTL Schematic.



Technology Schematic.



Design Summary.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	233	4656	5%
Number of Slice Flip Flops	165	9312	1%
Number of 4 input LUTs	439	9312	4%
Number of bonded IOBs	132	232	56%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	1	24	4%

Timing Report.

```
================================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 12.488ns (frequency: 80.078MHz)
  Total number of paths / destination ports: 151927 / 324
--------------------------------------------------------------------------------
Delay:               12.488ns (Levels of Logic = 21)
  Source:            count_3 (FF)
  Destination:       mod_2/x_1_31 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising
```

VI. CONCLUSION

In this paper, a NIST 256 prime field ECC processor implementation in FPGA has been presented. An RSD as a carry free representation is utilized which resulted in short datapaths and increased maximum frequency. We introduced enhanced pipelining techniques within Karatsuba multiplier to achieve high throughput performance by a fully LUT-based FPGA implementation.. Furthermore, an efficient modular addition /subtraction is introduced based on checking the LSD of the operands only. A control unit with add-on like architecture is proposed as a reconfigurability feature to support different point multiplication algorithms and coordinate systems. The main advantages of our processor include the exportability to other FPGA and ASIC technologies and expandability to support different coordinate systems and point multiplication algorithms.

REFERENCES

[1] N. Koblitz, “Elliptic curve cryptosystems,” Math. Comput., vol. 48, no. 177, pp. 203–209, Jan. 1987.

[2] W. Stallings, Cryptography and Network Security: Principles and Practice, 5th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Jan. 2010.

[3] C. Rebeiro, S. S. Roy, and D. Mukhopadhyay, “Pushing the limits of high-speed GF(2m) elliptic curve scalar multiplication on FPGAs,” in Proc. Cryptograph. Hardw. Embedded Syst. (CHES), vol. 7428. Jan. 2012, pp. 494–511.

[4] Y. Wang and R. Li, “A unified architecture for supporting operations of AES and ECC,” in Proc. 4th Int. Symp. Parallel Archit., Algorithms Programm. (PAAP), Dec. 2011, pp. 185–189.

[5] S. Mane, L. Judge, and P. Schaumont, “An integrated prime-field ECDLP hardware accelerator with high-performance modular arithmetic units,” in Proc. Int. Conf. Reconfigurable Comput. FPGAs, Nov./Dec. 2011, pp. 198–203.

[6] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi, "Efficient RNS implementation of elliptic curve point multiplication over GF(p)," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 8, pp. 1545–1549, Aug. 2012.

[7] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS implementation of an Fp elliptic curve point multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[8] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "Efficient poweranalysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 1, pp. 49–61, Feb. 2013.

[9] J.-Y. Lai and C.-T. Huang, "Energy-adaptive dual-field processor for high-performance elliptic curve cryptographic applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 8, pp. 1512–1517, Aug. 2011.

[10] S.-C. Chung, J.-W. Lee, H.-C. Chang, and C.-Y. Lee, "A highperformance elliptic curve cryptographic processor over GF(p) with SPA resistance," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2012, pp. 1456–1459.

[11] J.-Y. Lai and C.-T. Huang, "Elixir: High-throughput cost-effective dualfield processors and the design framework for elliptic curve cryptography," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 11, pp. 1567–1580, Nov. 2008.

[12] D. Karakoyunlu, F. K. Gurkaynak, B. Sunar, and Y. Leblebici, "Efficient and side-channel-aware implementations of elliptic curve cryptosystems over prime fields," IET Inf. Secur., vol. 4, no. 1, pp. 30–43, Mar. 2010.

[13] D. Schinianakis and T. Stouraitis, "Multifunction residue architectures for cryptography," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 4, pp. 1156–1169, Apr. 2014.

[14] J. Vliegen et al., "A compact FPGA-based architecture for elliptic curve cryptography over prime fields," in Proc. 21st IEEE Int. Conf. Appl.-Specific Syst. Archit. Process. (ASAP), Jul. 2010, pp. 313–316.

[15] T. Güneysu and C. Paar, "Ultra high performance ECC over NIST primes on commercial FPGAs," in Proc. 10th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES), 2008, pp. 62–78.

[16] P. L. Montgomery, "Modular multiplication without trial division," Math. Comput., vol. 44, no. 170, pp. 519–521, Apr. 1985.

[17] K. Sakiyama, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "Reconfigurable modular arithmetic logic unit for high-performance public-key cryptosystems," in Proc. 2nd Int. Workshop Reconfigurable Comput., Archit. Appl., vol. 3985. 2006, pp. 347–357.

[18] A. Byrne, E. Popovici, and W. P. Marnane, "Versatile processor for GF(pm) arithmetic for use in cryptographic applications," IET Comput. Digit. Tech., vol. 2, no. 4, pp. 253–264, Jul. 2008.

[19] J. Solinas, "Generalized Mersanne number," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. CORR 99-39, 1999. [20] B. Ansari and M. A. Hasan, "High-performance architecture of elliptic curve scalar multiplication," IEEE Trans. Comput., vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[21] N. Smyth, M. McLoone, and J. V. McCanny, "An adaptable and scalable asymmetric cryptographic processor," in Proc. Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP), Sep. 2006, pp. 341–346.

[22] C. J. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processor over GF(p)," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.

[23] K. Ananyi, H. Alrimeih, and D. Rakhmatov, "Flexible hardware processor for elliptic curve cryptography over NIST prime fields," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 8, pp. 1099–1112, Aug. 2009.

[24] M. Hamilton and W. P. Marnane, "FPGA implementation of an elliptic curve processor using the GLV method," in Proc. Int. Conf. Reconfigurable Comput. FPGAs (ReConFig), Dec. 2009, pp. 249–254.

[25] F. Crowe, A. Daly, and W. Marnane, "A scalable dual mode arithmetic unit for public key cryptosystems," in Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC), vol. 1. Apr. 2005, pp. 568–573.

[26] N. Takagi, "A VLSI algorithm for modular division based on the binary GCD algorithm," IEICE Trans. Fundam. Electron., Commun., Comput. Sci., vol. E81-A, no. 5, pp. 724–728, May 1998.