Design and Implementation of an Efficient  
RSD-Based ECC Processor  

G.Sindhuja\textsuperscript{1}  
gsindhu14@gmail.com \textsuperscript{1}  
K. Kishore Kumar\textsuperscript{2}  
kishorekumar20@gmail.com \textsuperscript{2}  

\textsuperscript{1}PG Scholar, VLSI, Dr.K.V.Subba Reddy Institute of technology , Lakshmipuram, Kurnool.  
\textsuperscript{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 Tiryakbyham.  

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\textsuperscript{2}). Karatsuba and Ofman proposed a methodology to perform a multiplication with complexity O(n\textsuperscript{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.

\[ a_H = (a_{n-1}, \ldots, a_{[n/2]}, a_0) \]
\[ a_L = (a_{[n/2]-1}, \ldots, a_0) \]
\[ b_H = (b_{n-1}, \ldots, b_{[n/2]}) \]
\[ b_L = (b_{[n/2]-1}, \ldots, b_0) \]

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

\[ a = a_L + a_H \beta^{[n/2]} \]
\[ b = b_L + b_H \beta^{[n/2]} \]

then

\[
C = AB = (a_L + a_H \beta^{[n/2]}) (b_L + b_H \beta^{[n/2]}) \\
= a_Lb_L + (a_L b_H + a_H b_L) \beta^{[n/2]} + a_H b_H \beta^n.
\]

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

\[
C = AB = (a_L + a_H \beta^{[n/2]}) (b_L + b_H \beta^{[n/2]}) \\
= a_Lb_L \\
+ (a_L + a_H)(b_L + b_H) - a_H b_H - a_L b_L \beta^{[n/2]} + a_H b_H \beta^n.
\]

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.

**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.
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.

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.

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 Mersenne 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 \(-p < A < p^2\). 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, \( D_5 \), 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 
\[ B = T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4 - D_5 \mod p, \]
where A16 represents the extra digits produced by RSD Karatsuba multiplier.

\[
T = (A_7 \| A_6 \| A_5 \| A_4 \| A_3 \| A_2 \| A_1 \| A_0) \\
S_1 = (A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{11} \| 0 \| 0 \| 0) \\
S_2 = (2 \cdot A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| 0 \| 0 \| A_{16}) \\
S_3 = (A_{15} \| A_{14} \| 0 \| 0 \| -2 \cdot A_{16} \| A_{10} \| A_9 \| A_8) \\
S_4 = (A_8 \| A_{13} \| A_{15} \| A_{14} \| A_{13} \| A_{11} \| A_{10} \| A_9) \\
D_1 = (A_{10} \| A_8 \| 0 \| 0 \| 0 \| A_{16} \| A_{13} \| A_{12} \| A_{11}) \\
D_2 = (A_{11} \| A_9 \| 0 \| 0 \| 2 \cdot A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12}) \\
D_3 = (A_{12} \| 2 \cdot A_{16} \| A_{10} \| A_9 \| A_{15} \| A_{14} \| A_{13}) \\
D_4 = (A_{13} \| 0 \| A_{11} \| A_{10} \| A_9 \| A_{16} \| A_{15} \| A_{14}) \\
D_5 = (-A_{16} \| 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| -A_{16}).
\]

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.

**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 Tiryakbbhyam 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 Tiryakbbhyam 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.

V. EXPERIMENTAL RESULTS

Results of proposed method
Simulation.

RTL Schematic.

Technology Schematic.

Fig.7. 8-bit binary multiplication using Urdhwa Tiryakbbhyam 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.
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


