MODELLING OF ELLIPTIC CURVE SCALAR MULTIPLIER ON LUT-BASED FPGAS FOR AREA AND SPEED

Sakthi Shree, E
Student, Dept. Of ECE
Sathyabama University, Chennai, Tamilnadu, India.
esakthietikkan@gmail.com

Jagatheswari, S
Student, Dept. Of ECE
Sathyabama University, Chennai, Tamilnadu, India.

Mrs. P. Kavipriya
Assistant professor, Dept. Of ECE
Sathyabama University, Chennai, Tamilnadu, India.
ecekavipriya@gmail.com

Abstract - The aim of this paper is to design a karatsuba multiplier that reduces the area consumed by modifying the algorithms Montgomery ladder and Itoh-tsujii algorithm. Here, a theoretical model to approximate the delay of different characteristic two primitives used in an elliptic curve scalar multiplier architecture (ECSMA) implemented on k input lookup table (LUT)-based field-programmable gate arrays. Approximations are used to determine the delay of the critical paths in the ECSMA. This is then used to theoretically estimate the optimal number of pipeline stages and the ideal placement of each stage in the ECSMA. This paper illustrates suitable scheduling for performing point addition and doubling in a pipelined data path of the ECSMA. Finally, detailed analyses supported with experimental results, are provided to design the fastest scalar multiplier over generic curves. Experimental results for GF(2163) show that, when the ECSMA is suitably pipelined, the scalar multiplication can be performed in only 13.90 ns on a Xilinx Virtex V.

Keywords - ECSMA, Karatsuba multiplier, Montgomery ladder, Itoh-tsujii algorithm

1 INTRODUCTION

The rapid advances in information technology in the last few decades have led to intensive research on information security. Many technologies and cryptographical systems have been developed, all to secure information and protect it from unauthorized invaders. Public-key cryptography has been widely studied and used since 1975 when Rivest, Shamir, and Adleman invented RSA public key cryptography. This system heavily depends on integer factorization problem (IFP) using big key bits such as 1024 bits and 2048 bits. Later on Diffie-Hellman in developed the public key exchange algorithm using the discrete logarithm problem (DLP). ElGamal also used DLP in encryption and digital signature scheme. In 1985, Koblitz and Miller independently used EC in cryptography using elliptic curves discrete logarithm problem (ECDLP). In recent years, researchers have given more attention to develop the proposed ECC algorithms and improve their efficiency. Improving the efficiency of scalar multiplication in EC is one of the main interests of many researchers in the field of cryptology.

The techniques proposed so far use different methods for representing the scalar k, which clearly shows different levels of computation speed and security. Binary representation is extended to signed binary representation, and its Non-Adjacent Form (NAF) algorithm. Other well-known techniques such as the window methods and the Montgomery method have brought about much improvement in terms of the efficiency of EC arithmetic. When doubling one point P to obtain 2P = R as a new point on E, extra field squaring over prime fields is required, but it is the same cost as in point adding if the curve has been defined over the binary fields. Adding two points P and Q on the same elliptic curve E, requires solving three equations, that involving one field inversion, one field squaring, and two field multiplications, which are costly operations in EC implementation. Some other operations involved in adding two points are addition, subtraction, and multiplication of small integers, which are in most cases negligible operations. The proposed algorithms offers many different formulas for finding point multiplication from the given point P. One can use many doublings 2((2P)...(2(2P))) with one or more extra point additions. The reverse operation also lead to find points such as P from (2P), which is called point halving.

Elliptic curve cryptography is rapidly becoming the standard for public-key ciphers because of the large amount of security provided per key bit. Several accreditation bodies have migrated to ECC for their public-key cryptographic requirements. To match the speed requirements for real-time applications, hardware acceleration of ECC is a necessity. Field-programmable gate arrays Form an ideal platform for hardware implementations of security algorithms such as ECC. However, because FPGAs are resource-constrained, there are many challenges in developing designs to ensure better resource utilization, keeping the...
critical delay minimal. In this paper, we present the design of a high-speed ECC processor for binary fields on FPGA platforms. This paper proposes an efficient implementation of the field multiplier, which is the most important primitive in ECC. The elliptic curve scalar multiplier architecture presented uses a pipelined bit-parallel Karatsuba multiplier, with the objective of improving the utilization of the FPGA’s lookup tables. For the inversion, another important field primitive, the Itoh–Tsujii algorithm, is designed to use optimal exponentiation circuits specific to the LUT size of the underlying FPGA platform. These field primitives are combined to realize the elliptic curve scalar multiplier. In the next part, this paper explores opportunities for pipelining the design for high-speed.

II KARATSUBA MULTIPLIER

Basically Karatsuba stated that if we have to multiply two n-digit numbers x and y, this can be done with the following operations, assuming that B is the base of and m < n.

First both numbers x and y can be represented as x1,x2 and y1,y2 with the following formula.

\[ x = x_1 \cdot B^m + x_2 \quad y = y_1 \cdot B^m + y_2 \]  

---(1)

Obviously now xy will become as the following product.

\[ xy = (x_1 \cdot B^m + x_2)(y_1 \cdot B^m + y_2) \]

\[ a = x_1 \cdot y_1 \]

\[ b = x_1 \cdot y_2 + x_2 \cdot y_1 \]

\[ c = x_2 \cdot y_2 \]  

---(2)

Finally xy will become:

\[ xy = a \cdot B^{2m} + b \cdot B^m + c \]

However a, b and c can be computed at least with four multiplications, which isn’t a big optimization. That is why Karatsuba came up with the brilliant idea to calculate b with the following formula:

\[ b = (x_1 + x_2)(y_1 + y_2) - a - c \]  

---(3)

That make use of only three multiplications to get xy.

Let’s see this formula by example.

\[ 47 \times 78 \]

\[ x = 47 \]

\[ y = 78 \]

\[ x_1 = 4 \]

\[ x_2 = 7 \]

\[ y_1 = 7 \]

\[ y_2 = 8 \]

\[ a = x_1 \cdot y_1 = 4 \cdot 7 = 28 \]

\[ c = x_2 \cdot y_2 = 7 \cdot 8 = 56 \]

\[ b = (x_1 + x_2)(y_1 + y_2) - a - c = 11 \cdot 15 - 28 - 56 \]

Now the thing is that 11 \cdot 15 it’s again a multiplication between 2-digit numbers, but fortunately we can apply the same rules two them. This makes the algorithm of Karatsuba a perfect example of the “divide and conquer” algorithm.

A. Efficiency analysis:

Karatsuba’s basic step works for any base B and any m, but the recursive algorithm is most efficient when m is equal to n/2, rounded up. In particular, if n is \(2^k\), for some integer k, and the recursion stops only when n is 1, then the number of single-digit multiplications is \(3^k\), which is \(n^c\) where \(c = \log_3 3\).

Since one can extend any inputs with zero digits until their length is a power of two, it follows that the number of elementary multiplications, for any \(n\), is at most \(3^{\lceil\log n\rceil} \leq 3n\log_2 n\).

Since the additions, subtractions, and digit shifts (multiplications by powers of B) in Karatsuba’s basic step take time proportional to \(n\), their cost becomes negligible as \(n\) increases. More precisely, if \(t(n)\) denotes the total number of elementary operations that the algorithm performs when multiplying two \(n\)-digit numbers, then

\[ t(n)=3[(n2)] + cn +d \]

---(4)

for some constants \(c\) and \(d\). For this recurrence relation, the master theorem gives the asymptotic bound \(t(n) = O(n\log^2 n)\).

It follows that, for sufficiently large \(n\), Karatsuba’s algorithm will perform fewer shifts and single-digit additions than the longhand method, even though its basic step uses more additions and shifts than the straightforward formula. For small values of \(n\), however, the extra shift and add operations may make it run slower than the longhand method. The point of positive return depends on the computer platform and
context. As a rule of thumb, Karatsuba is usually faster when the multiplicands are longer than 320–640 bits.

III IMPLEMENTATION OF FIELD PRIMITIVES

In polynomial basis representation of binary fields, addition is performed by bitwise EXCLUSIVE OR operation. For fixed irreducible polynomials, squaring circuits can be hardwired, thus making squaring operations easy. Operations such as multiplications and inversions are complex and should be implemented efficiently. This section describes the implementation of the field multiplication and inversion units in the ECSMA.

A. Field multiplication

Multiplication in binary fields involves a polynomial multiplication followed by a modular reduction. For high-speed implementations, bit-parallel multipliers are preferred to serial and word-parallel multipliers. Several algorithms for bit parallel field multiplication are available in the literature. The ECSMA in this paper uses a Karatsuba multiplier because of its sub quadratic complexity. It is shown in fig 3.1.

IV PARAMETRIC KARATSUBA INTEGER MULTIPLIER

The key to our Montgomery multiplier design is the recursive Karatsuba algorithm. This allows us to compute modular multiplication with a complexity approaching. Our design uses multiple-precision arithmetic techniques so that the critical path delay is independent of the multiplier’s bit-width. Unless stated otherwise, we assume we are multiplying two 2k-bit unsigned integers and the limb-widths of all components are w. The number of limbs in a 2k-bit word. We use either a coarse-grained carry-save technique or a pipelined multiple-precision technique in all of our adders and subtracters. The critical path of the circuit primarily depends upon the limb-width.

V PROCESSOR ORGANIZATION

In this section, we describe the construction of the ECSMA, which uses the left-to-right double and add algorithm with binary signed digit representation of the scalar Algorithm 1. The scalar s is the input to the processor and the output is the scalar product sP. Point arithmetic is done in the LD projective coordinate system. At every clock cycle, the register bank feeds the arithmetic unit through six buses. At the end of the clock cycle, the results of the computation are stored in the registers through buses C0, C1, C2, and Qout. Control signals are generated at every clock depending on the state of the machine and key digit. This section elaborates the working of the ECSMA shown in Fig 5.1.

Fig 3.1: Block diagram of recursive Karatsuba multiplier

Fig 5.1: ECSMA

VI PIPELINING THE ECSMA

The ECSMA discussed in the previous section suffers from low operating frequency because of the long combinational paths. The operating frequency can be increased by splitting the critical paths using the pipeline strategy. Data dependencies present in the point arithmetic steps may introduce bubbles in the pipeline, resulting in an increase in the clock cycle requirement. The number of bubbles during execution is likely to
increase with the number of pipeline stages. Thus, with the increase in pipeline stages, though the delay reduces, the clock cycle requirement increases. Therefore designing a pipeline would require a balance between clock cycle requirement and critical delay. This section first identifies the critical paths in the ECSMA and then theoretically estimates the best pipeline strategy for the design.

VII ALGORITHMS

A. Itoh-Tsujii algorithm

The Itoh-Tsujii inversion algorithm is used to invert elements in a finite field. It was introduced in 1988 and first used over GF(2^m) using the normal basis representation of elements, however the algorithm is generic and can be used for other bases, such as the polynomial basis. It can also be used in any finite field, GF(p^m).

The algorithm is as follows:

Input: A ∈ GF(p^m)
Output: A^−1

1. r ← (p^m − 1)/(p − 1)
2. compute A' = A'−1 in GF(p^m)
3. compute A' = A'· A
4. compute (A'−1) in GF(p)
5. compute A−1 = (A')−1 · A'−1
6. return A−1

This algorithm is fast because steps 3 and 5 both involve operations in the subfield GF(p). Similarly, if a small value of p is used a lookup table can be used for inversion in step 4. The majority of time spent in this algorithm is in step 2, the first exponentiation. This is one reason why this algorithm is well-suited for the normal basis, since squaring and exponentiation are relatively easy in that basis.

B. Montgomery ladder

Points on an elliptic curve E, defined over a finite field GF (q), along with a special point called infinity, and a group operation known as point addition, form a commutative finite group. If P is a point on the curve E, and k is a positive integer computing (eqn 5)

kP = P + P + . . . + P

-----(5)

is called scalar multiplication. The result of scalar multiplication is another point Q on the curve E. It is normally expressed as Q = kP. If E is an elliptic curve defined over GF (q), the number of points in E(GF (q)) is called the order of E over GF (q), denoted by #E(GF (q)). For cryptographic applications #E(GF (q)) = rh where r is prime and h is a small integer and P and Q have order r. Scalars such as k are random integers where 1 < k < r − 1. Since r = q, the binary representation of k = k2i has n bits where n = m = [log2 q]. Scalar multiplication is the most dominant computation part of elliptic curve cryptography.

Algorithm A shows the Montgomery scalar multiplication scheme for non-supersingular elliptic curves over binary fields as it was introduced. In this algorithm Madd(X_1,Z_1,X_2,Z_2), Mdouble(X_1,Z_1) and Mxy(X_i,Z_i,X_j,Z_j) are functions for point addition, point doubling and conversion of projective coordinates to affine coordinates.

C. Point multiplication technique

Point Multiplication is the basic computation primitive of elliptic curve cryptography. The definition of corresponding operations depends on a particular field, but they always amount to combinations of arithmetic operation.

kP = P + P + . . . + P

-----(6)

Where P is a point on an elliptic curve E and k is an integer in a range 1 ≤ k < order (P). Accordingly, the elliptic curve point multiplication means that the point P is added to itself k times. The order of the point P is n0 if and only if P multiplied with n0 results in the point at infinity.

D. Elliptic curve point addition & doubling

Point addition:
To add two distinct points P and Q on an elliptic curve.

Point doubling:
The point-doubling operation amounts to squaring operations of any binary number.

E. Finite field arithmetic

A field F (finite field) is equipped with two operations, addition and multiplication. Subtraction of field elements is defined in terms of addition.

VIII RESULTS AND DISCUSSIONS

The simulation results of Karatsuba Multiplier is shown below. The result in figure 8.1 and figure 8.2 shows the area utilized in the existing system.
Figure 8.3 shows the top level module implementation (RTL Schematic) of the Multiplier.

Fig 8.1 Device Utilization summary of existing system

Fig 8.2 Simulation results of existing system

Fig 8.3 RTL Schematic of multiplier

Fig 8.4 Internal RTL Schematic of Multiplier

Fig 8.5 Internal Schematic of ks233

Figure 8.6 shows the device utilization summary of proposed system. Figure 8.7 shows the detailed analysis of the proposed system.
Table 8.1 shows the comparison of existing and proposed system.

**TABLE 8.1 COMPARISON TABLE OF EXISTING AND MODIFIED MULTIPLIER**

<table>
<thead>
<tr>
<th>Logic Utilizion</th>
<th>Number of Used LUTs</th>
<th>Number of Fully Used LUTs</th>
<th>Number of Bonded IOs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXISTING</strong></td>
<td>15336</td>
<td>20480</td>
<td>0</td>
</tr>
<tr>
<td><strong>MODIFIED</strong></td>
<td>4082</td>
<td>20480</td>
<td>0</td>
</tr>
</tbody>
</table>

IX CONCLUSION

This proposed elliptic curve scalar multiplier used for high-speed and area-constrained applications. The implementation uses a novel pipelined bit-parallel Karatsuba multiplier that has sub quadratic complexity. A theoretical model was used to analyze delay of critical paths in the ECSMA and to obtain optimal pipelining. 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.

In future the power analysis can be done by using tools like TSpice and it can be reduced.

X. REFERENCES


