Power-Time Flexible Architecture For GF(2^k) Elliptic Curve Cryptosystem Computation

Adnan Abdul-Aziz Gutub and Mohammad K. Ibrahim
Computer Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, SAUDI ARABIA
E-mail: {gutub,ibrahimm}@ccse.kfupm.edu.sa

ABSTRACT
New elliptic curve cryptographic processor architecture is presented that result in considerable reduction in power consumption as well as giving a range of trade-off between speed and power consumption. This is achieved by exploiting the inherent parallelism that exist in elliptic curve point addition and doubling. Further trade-off is achieved by using digit serial-parallel multipliers instead of the serial-serial multipliers used in conventional architectures. In effect, the new architecture exploits parallelism at the algorithm level as well as at the arithmetic element level. This parallelism can be exploited either to increase the speed of operation or to reduce power consumption by reducing the frequency of operation and hence the supply voltage.

Categories and Subject Descriptors
B.2.4 High-Speed Arithmetic (*Algorithms, Cost/performance)*
C.5.4 VLSI Systems

General Terms
Algorithms, Measurement, Performance, Design, Security.

Keywords
Elliptic Curve Cryptography, Projective Coordinate arithmetic, Parallel architecture, Crypto-Systems Power-time tradeoff.

1. INTRODUCTION
Elliptic Curve Cryptosystem (ECC) has received considerable attention from mathematicians around the world ever since the original proposal by N. Koblitz and V. Miller in 1985 [1-8]. ECC is based on the Discrete Logarithm problem over the points on an elliptic curve. To date, no significant breakthroughs have been made in determining weaknesses in the algorithm. The fact that the problem appears so difficult to crack means that key sizes can be reduced in size considerably, even exponentially [2,7], especially when compared to the key size used by other cryptosystems. This made ECC become a challenge to the RSA, one of the most popular public key methods. Although critics are still skeptical as to the reliability of this method, several encryption techniques have been developed recently using the properties of elliptic curve.

Several cryptographic processors have been proposed in the literature recently [4,5,12]. The conventional approach used in the design of these processors is to adopt serial computations at both the algorithmic level by using a single multiplier, as well as at the arithmetic level by using a serial multiplier. The reason for sequential operation is that it leads low area for large word lengths that is needed for secure encryption (i.e. > 160 bits [8]). This classical approach could lead to the reduction of area, however, the constraint of current technology is not on gate count but power consumption. Reducing area is not necessarily the best approach to reducing power consumption.

In this paper, a power-time flexible architecture is proposed that exploits the parallelism inherent at both the algorithmic level and the arithmetic level of ECC. This is contrary to existing designs [4,5,12], which opt for sequential operations to minimize area. It is strongly believed that these two aspects would lead to an even better trade-off between the time and power consumption.

2. ELLIPTIC CURVES OVER GF(2^k)
It will be assumed that the reader is familiar with the arithmetic over elliptic curve. For good review the reader is referred to [8]. The elliptic curve equation over $GF(2^k)$ is: $y^2+xy = x^3+ax^2+b$; where: $x,y,a,b \in GF(2^k)$ *and* $b \neq 0$. The addition of two different points on the elliptic curve is computed as shown below:

$(x_1 , y_1) + (x_2 , y_2) = (x_3 , y_3)$; where $x_1 \neq x_2$
$\lambda = (y_2 + y_1)/(x_2 + x_1)$
$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$
$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$

The addition of a point to itself (Doubling a point) on the elliptic curve is computed as shown below:

$(x_1 , y_1) + (x_1 , y_1) = (x_3 , y_3)$; where $x_1 \neq 0$
$\lambda = x_1 + (y_1)/(x_1)$
$x_3 = \lambda^2 + \lambda + a$
$y_3 = (x_1)^2 + (\lambda + 1)\, x_3$

For elliptic curve cryptography several point addition and doubling operations are needed [2,6,8]. As seen from the equations above, any point operations over elliptic curve requires inversion, which is the most expensive operation over GF(2^k) [1,8,12]. A common approach is to eliminate the need for

GLSVLSI'03, April 28-29, 2003, Washington, DC, USA.

inversion by representing the elliptic curve points as projective coordinate points [1,4,6,8,12]. This results in replacing the inversion with several multiplication operations. This approach is also adopted in the processor proposed here.

3. GF(2^k) ECC POINT OPERATIONS OVER PROJECTIVE COORDINATES

To eliminate the need for performing inversion in $GF(2^k)$, its coordinates (x,y) are projected to (X, Y, Z), where $x=X/Z^2$, and $y=Y/Z^3$. The projected elliptic curve equation becomes:

$$Y^2 + XYZ = X^3 + a X^2Z^2 + b Z^6$$

The formulas for projective point addition of two elliptic curve points are as follows:

$P=(X_1,Y_1,Z_1);Q=(X_2,Y_2,Z_2);P+Q=(X_3,Y_3,Z_3);$ *where* $P \neq \pm Q$

$(x, y) = (X/Z^2, Y/Z^3) \rightarrow (X,Y,Z)$

$\lambda_1 = X_1 Z_2^2$	2M
$\lambda_2 = X_2 Z_1^2$	2M
$\lambda_3 = \lambda_1 + \lambda_2$	
$\lambda_4 = Y_1 Z_2^3$	2M
$\lambda_5 = Y_2 Z_1^3$	2M
$\lambda_6 = \lambda_4 + \lambda_5$	
$\lambda_7 = Z_1 \lambda_3$	1M
$\lambda_8 = \lambda_6 X_2 + \lambda_7 Y_2$	2M
$Z_3 = \lambda_7 Z_2$	1M
$\lambda_9 = \lambda_6 + Z_3$	
$X_3 = a Z_3^2 + \lambda_6 \lambda_9 + \lambda_3^3$	5M
$Y_3 = \lambda_9 X_3 + \lambda_8 \lambda_7^2$	3M

	20M

The formulas for projective point doubling of P is given by:

$P = (X_1,Y_1,Z_1);\ P+P = (X_3,Y_3,Z_3)$

$Z_3 = X_1 Z_1^2$	2M
$X_3 = (X_1 + bZ_1^2)^4$	3M
$\lambda = Z_3 + X_1^2 + Y_1 Z_1$	2M
$Y_3 = X_1^4 Z_3 + \lambda X_3$	3M

	10M

The complete data flow graph for doubling a point is shown in Figure 1. It requires ten multiplications and four k-bit XOR operations. Figure 2 shows the data flow graph for adding two elliptic curve points. It requires twenty multipliers and seven k-bit XOR gates. From the binary method, any elliptic curve crypto processor that uses projective coordinates must implements the dataflow graphs in Figure 1 and 2 iteratively.

4. PROPOSED ARCHITECTURE

The architecture of the new processor is shown in Figure 3. The new architecture has the following features:

- It has three digit serial-parallel multipliers,
- It can perform multiply-add operation in the same instruction,
- It has a power management unit.

The basic motivation behind the design of the proposed architecture is to exploit, as much as possible, the full parallelism that exists in the ECC. The trade-off between power and time can be achieved by reducing the clock frequency and hence consequently the source voltage. It is well known that reducing the source power is the most effective means of reducing power consumption. In the proposed design here, this is exploited at the algorithmic level by using more than one multiplier. It is also exploited at the arithmetic element level by using serial-parallel multiplier such as those reported in [10,11] rather than the conventional approach of using a serial multiplier. The benefits of parallel implementation of ECC on power are discussed in more details in section 7.

The reason for using three multipliers only is as follows. From Figures 1 and 2, the corresponding critical path of each dataflow diagram is effectively of 5 GF(2^k) multiplications and 7 GF(2^k) multiplications, respectively. Here the time GF(2^k) addition is ignored since it is negligible compared to multiplication. Therefore, the lower bound of the minimum computation time to perform one elliptic point operation in the calculation of nP is 12 GF(2^k) multiplications.

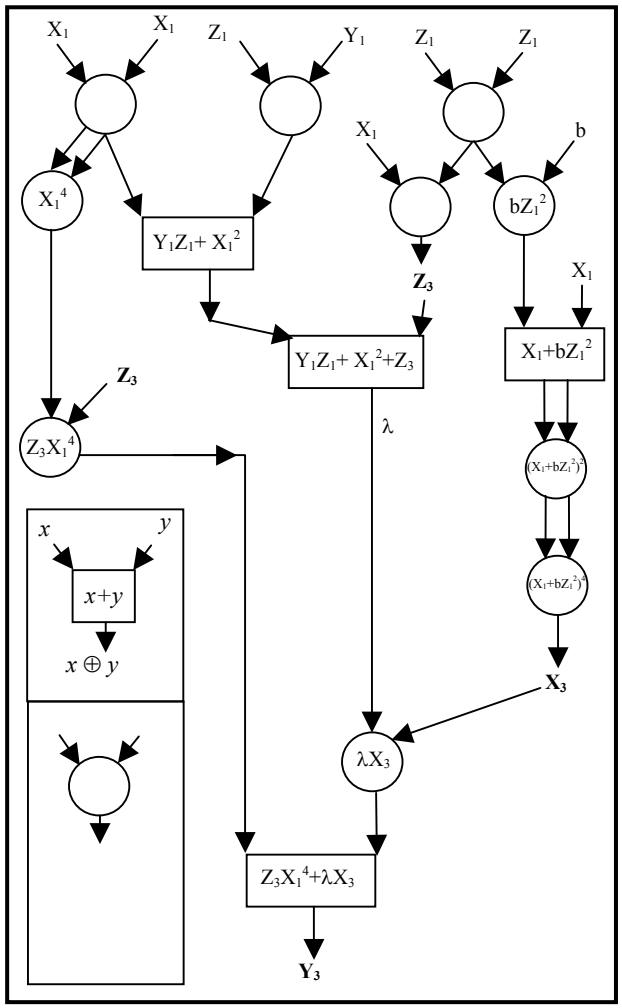

Figure 1. Doubling an elliptic curve point Data flow graph

It can be easily seen from Figures 1 and 2 that performing three multiplications in parallel will meet this lower bound, and any further concurrent multiplications will not actually achieve any further reduction in the computation time. It should also be noted that the utilization of the three multipliers is very high. As can be seen from Figures 1 and 2, all the three multipliers will be used in eight out of the 12 steps, and in only two out of the 12 cycles where a single multiplier is used.

The advantage of performing multiply-add operation in one instruction is that the dataflow in Figures 1 and 2 include many

computations where the addition of the output of two multipliers must be carried out. Such a feature will circumvent the need to store these values back in the registers and fetching them back again for their subsequent addition. This will save both in cycles and power. The purpose of the power management unit is to ensure that the power consumption of blocks that are not

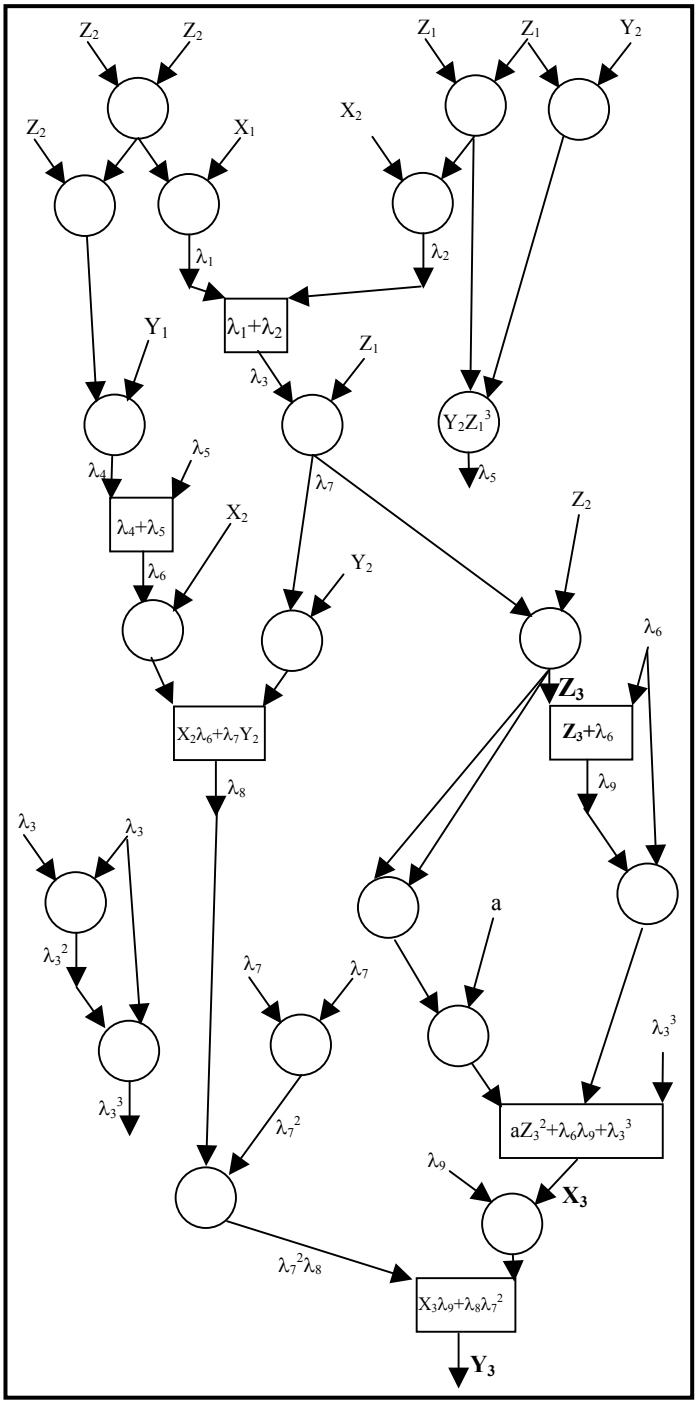

Figure 2. Data flow graph for adding two points

used is kept to a minimum. This is achieved by clock-gating the registers of these blocks and ensuring that the logic in these blocks is static. There are two possible cases where blocks are not used. The first is when not all three multipliers are used, and the second is when the application wordlength is less than the wordlength of the processor.

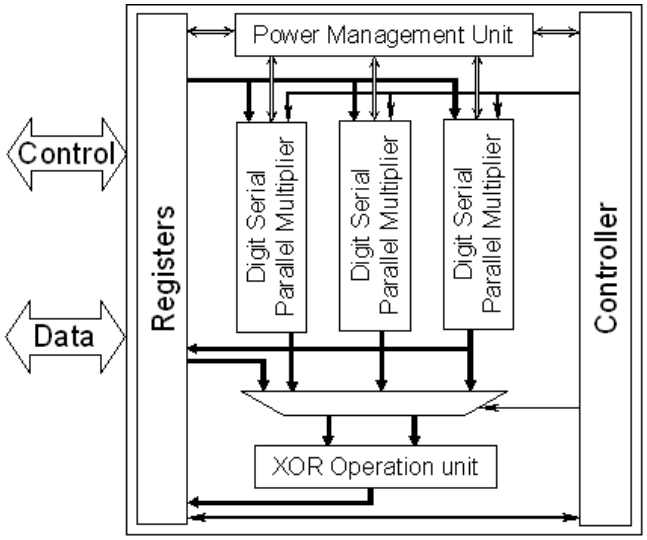

Figure 3. The elliptic curve point operations hardware

5. PERFORMANCE COMPARISONS

5.1 Serial-Parallel Vs. Serial Multiplication

In the crypto processor presented here we also propose to use GF(2^k) digit serial multipliers such as those reported in [10,11]. These digit serial-parallel structures lead to a much better trade-off between power and time. Given N as the wordlength and M as the number of digits of size (N/M), Table 1 shows the comparison of serial-parallel multiplier with a serial one with the *same* digit size.

Table 1. Comparison of digit-serial multiplier [16] vs. serial multiplier[4]. C_o is the capacitance of a single-bit arithmetic.

Type of Multiplier	Normalized Area	Time (Cycles)	Freq	Power = kf^3C
Serial	$(N/M)^2$	M^2	f_o	$k(N^2/M)^2 f_o^3 C_o$
Serial-Parallel	N^2/M	M	f_o/M	$k(N^2/M^4) f_o^3 C_o$

Since Power, $P=fCV_S^2$ and assuming that $V_S=kf_o$, where f_o is the maximum operating frequency for the given V_s, *then* $P=kf^3C$. Given that serial-parallel computation requires M cycles compared to M^2 cycles for the serial multiplier, the clock frequency of the serial parallel multiplier can be reduced by a factor of M for the same execution time. As can be clearly seen from Table 1, operating the serial-parallel multiplier at clock frequency of f_o/M will lead to a reduction of power by a factor of M^2. A further advantage of the proposed architecture is that it allows the designer the flexibility to operate at a higher clock frequency up to f_o, but of course at the expense of higher power consumption. Clearly this demonstrates the superiority of using digit serial-parallel computation.

As with regard to the GF(2^k) modulo adder, it is to be implemented in bit parallel fashion since the area is not significant compared to the multiplier and minimizing the addition time will reduce the overall multiply-add cycle time.

5.2 Parallel Vs Sequential Implementation

The power consumption of using three multipliers is compared with that of using a single multiplier and two multipliers in Figure 4 for different execution times. Here time is computed as follows, time=No. of cycles x f_o. Also, the same assumptions made in Table 1 are used for Figure 4.

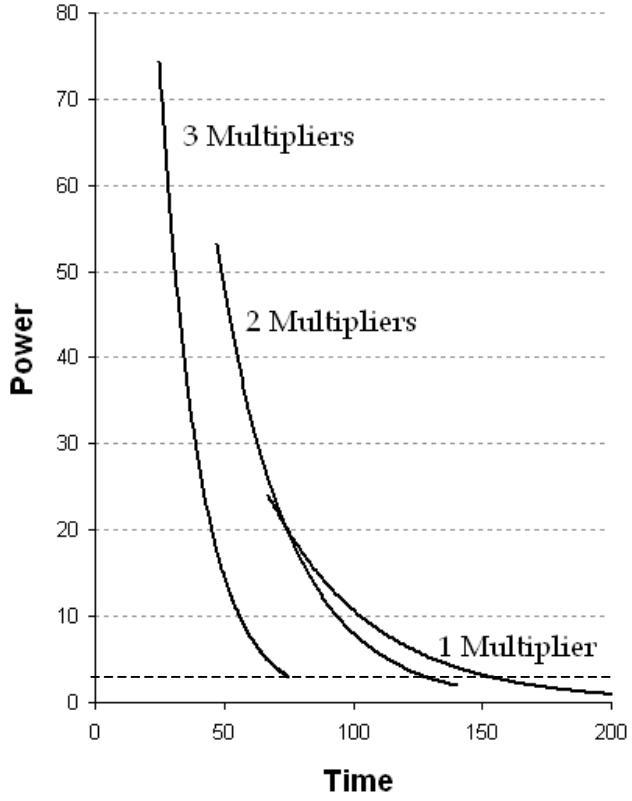

Figure 4. Comparison of using different multipliers.

In existing designs [4,5], a single multiplier is used to perform all the multiplications needed in Figures 1 and 2. The reason is that using more than one multiplier is perceived to be too expensive. However, as can be seen from Figure 4, the proposed architecture would lead to much lower power consumption than using one or two multipliers for the *same* execution time.

It is also clear that using three multipliers gives a wider range of trade-off between power and speed. In fact, the case of using two multipliers does not provide any advantage over the other two options. Finally, the proposed architecture can support a further reduction in power by switching to one multiplier based operation in cases where a further reduction in power is required. In this case the power management unit will simply ensure that the other two multipliers do not consume any dynamic power.

6. CONCLUSION

A novel GF(2^k) elliptic curve crypto processor is proposed in this paper. The new architecture results in considerable reduction in power consumption as well as offering users a range of trade-off between power and time. The basic feature of the new architecture is that it exploits the inherent parallelism in the computation of doubling and adding points over an elliptic curve as well as in multiplication. Performance evaluation shows a considerable advantage over sequential implementation in terms of power consumption and time.

7. ACKNOWLEDGMENTS

The authors would like to thank KFUPM for supporting this work and Dr Muhammad E. S. Elrabaa for discussions on figures of merits for low power designs.

8. REFERENCES

[1] Miyaji. Elliptic Curves over F_P Suitable for Cryptosystems. Advances in cryptology-AUSCRUPT'92, Australia, 1992.

[2] Stallings. Cryptography and Network Security: Principles and Practice. 2nd Edition, Prentice Hall Inc., NJ, 1999.

[3] Chung, Sim, and Lee. Fast Implementation of Elliptic Curve Defined over GF(p^m) on CalmRISC with MAC2424 Coprocessor. Workshop on Cryptographic Hardware and Embedded Systems, Massachusetts, August 2000.

[4] Okada, Torii, Itoh, and Takenaka. Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2^m) on an FPGA. Workshop on Cryptographic Hardware and Embedded Systems, Massachusetts, August 2000.

[5] Orlando, and Paar. A High-Performance Reconfigurable Elliptic Curve Processor for GF(2^m). Workshop on Cryptographic Hardware and Embedded Systems, Massachusetts, August 2000.

[6] Stinson. Cryptography: Theory and Practice. CRC Press, Boca Raton, Florida, 1995.

[7] Paar, Fleischmann, and Soria-Rodriguez. Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents. IEEE Transactions on Computers, 48(10), Oct. 1999.

[8] Blake, Seroussi, and Smart. Elliptic Curves in Cryptography. Cambridge University Press: NY, 1999.

[9] Scott, Norman R. Computer Number Systems and Arithmetic. Prentice-Hall Inc., New Jersey, 1985.

[10] Ibrahim, and Almulhem. Bit-Level Pipelined Digit Serial GF(2^m) Multiplier. IEEE International Symposium on Circuits and Systems, Sidney, Australia, 2001.

[11] Ibrahim, Junaid, Al-Abaji, and Almulhem. Trade-off analysis of a new sign digit serial GF multiplier. Fifth World Multi-conference on Systemics, Cybernetics and Informatics SCI/ISAS, XIV(II):52-56. Orlando, July 2001.

[12] Orlando, and Paar. A scalable GF(p) elliptic curve processor architecture for programmable hardware. Workshop on Cryptographic Hardware and Embedded Systems, Paris, France, May 2001.