A Comparison on FPGA of Modular Multipliers Suitable for Elliptic Curve Cryptography over GF(p) for Specific p Values

Mark Hamilton, William Marnane, Arnaud Tisserand

2011 21st International Conference on Field Programmable Logic and Applications

A Comparison on FPGA of Modular Multipliers Suitable for Elliptic Curve Cryptography over GF(p) for Specific p Values

Mark Hamilton, William P. Marnane
Claude Shannon Institute for Discrete Mathematics, Coding and Cryptography
Department of Electrical & Electronic Engineering
University College Cork, Ireland
{*markh,liam*}*@eleceng.ucc.ie*

Arnaud Tisserand
IRISA, CNRS, INRIA,
Centre Rennes - Bretagne Atlantique,
Université Rennes 1,
6 rue Kérampont, 22305 Lannion, FRANCE
arnaud.tisserand@irisa.fr

***Abstract*—In this paper we provide a comparison of different modular multipliers suitable for use in an elliptic curve processor, when working with a Mersenne prime modulus. Mersenne primes allow for the use of fast modular reduction techniques. Several multipliers are presented that can be implemented solely in slice logic. A design that makes use of the DSP48E blocks on Virtex 5 FPGAs is also described. The different multipliers are compared for speed, area and power consumption when implemented on a Virtex 5 FPGA.**

Keywords: FPGA, Hiasat Multiplier, Elliptic Curve Processor, Mersenne prime, Modular Multiplication.

I. Introduction

The modular multiplier in an elliptic curve processor is generally the component that determines the critical path of the design. A very basic method for computing a single modular multiplication is to perform a full width multiplication and then reduce the result mod *p*. The reduction step is computationally intensive as it requires more or less the division of two numbers. During a modular exponentiation the reduction would have to be performed after each multiplication, this method would be too slow to be used for Elliptic Curve Cryptography (ECC) where the modulus can be several hundred bits in length. The Montgomery algorithm [1] is a much more efficient method as it does not require the computationally intensive trial division operations.

An alternative to the Montgomery method is to choose a modulus of a special form that support fast modular reduction, such as $2^n - 1$, 2^n or $2^n + 1$. For ECC the most interesting form is $2^n - 1$ where *n* is a prime number. This form of number is known as a Mersenne prime and there are two such primes that are currently useful for ECC. In [2], NIST specify a curve over GF(p) with $p = 2^{521} - 1$ as one of the standard curves to be used for ECC. In [3], the authors introduce a method for performing ECC using a curve defined over the extension field GF(p^2) with $p = 2^{127} - 1$. There are many ways of implementing a multiplier modulo a Mersenne prime. Several designs were implemented on a Virtex xc5vlx50 and an xc5lx220 FPGA and compared. The xc5vlx50 is the smallest Virtex 5 FPGA that Xilinx produces.

II. Multipliers for modulo $2^n - 1$ multiplication

In 1992, Hiasat [4] proposed a design that takes advantage of the fast modular reduction that can be performed when working with moduli of the form $2^n \pm 1$. In the case of $2^n - 1$ the multiplication can be performed as follows.

If,

$$Z = XY = \sum_{i=0}^{2n-1} z_i 2^i \tag{1}$$

the modular reduction of Z can be described as,

$$|Z|_{2^n-1} = \left| \sum_{i=0}^{n-1} z_i 2^i + 2^n \sum_{i=n}^{2n-1} z_i 2^{i-n} \right|_{2^n-1} \tag{2}$$

where,

$$|2^n|_{2^n-1} = |1|_{2^n-1} \tag{3}$$

This type of multiplier can be implemented through the use of a full width multiplier followed by correction circuitry that performs the modular reduction step. Methods for performing the $n \times n$ bit multiplication are discussed later in the section. The correction circuitry consists of simple combinational logic. The setup is shown in Fig. 1.

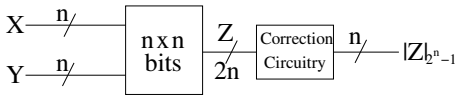

Figure 1. Hiasat Multiplier

We have seen that the modular reduction step above can be performed efficiently in hardware. Next we discuss methods for efficiently implementing the $n \times n$ bit multiplication shown in Fig. 1.

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.55

A. Serial Multiplier

The simplest form of multiplier is one which implements an n-bit multiplication as the sum of n partial products, Fig. 2. The multiplier, Y, is fed into a shift register and the least significant bit (LSB) is scanned. If the current bit is a 1, the multiplicand, X, is added to an accumulator and the multiplier is shifted to the right by one bit position. If a 0 bit is detected, nothing is added to the accumulator and the multiplier is shifted to the right by one bit position, this is represented by the multiplexer in Fig. 2. This process continues until every bit of the multiplier has been scanned. This method is known as the schoolbook method for multiplication. This form of multiplier requires a single n-bit adder and performs the multiplication in n clock cycles. Fig. 2 shows how the output of the adder feeds into a shift register that holds the accumulator value. Y is stored in the LSBs of the accumulator and shifted right. After n clock cycles Y is completely shifted out of the accumulator and only the result of the multiplication, Z, will remain. Z can then be fed into some correction circuitry to obtain the mod $2^n - 1$ result, as shown in Fig. 1. The critical path of the design is the n-bit adder.

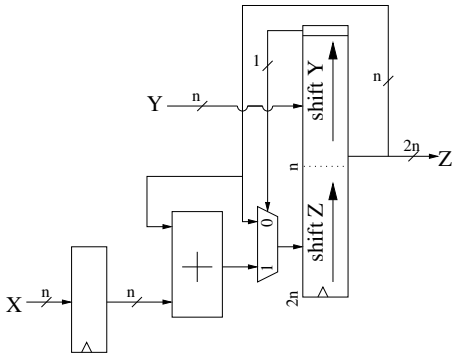

Figure 2. Serial Multiplier

B. Booth Multiplier

In 1951 Booth [5] introduced a multiplication algorithm which reduced the number of additions of partial products to $n/2$, at the cost of some extra hardware. A modified version of Booth's algorithm was introduced in [6]. Both algorithms work by precomputing partial products multiples from a certain set. In the case of the modified Booth algorithm the multiples $\{-2X, -X, 0, X, +2X\}$ are precomputed. Every multiple in this set can be calculated by a bitwise shift, an inversion in 2's complement or both. For the modified Booth algorithm overlapping groups of 3 bits of the multiplicand must be scanned at a time. For a detailed description of Booth recoding schemes see [7]. The circuit is shown in Fig. 3. The critical path of this design is through the adder and the circuitry used for the precomputations.

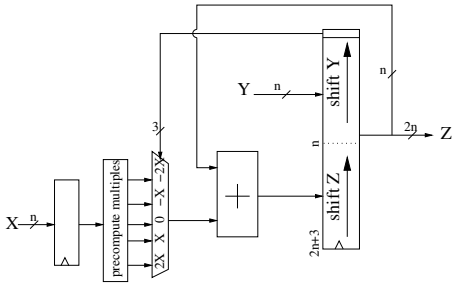

Figure 3. Booth2 Multiplier

III. Montgomery Multiplier

In 1985 Montgomery [1] introduced an algorithm to efficiently compute the modular product of two numbers. The algorithm is very efficient when used for modular exponentiation as it does not require computationally intensive trial division operations. The Montgomery multiplier is one of the most widely used types of multipliers in elliptic curve processors. This is due to its relatively short critical path and its ability to be used with a wide range of different moduli.

Montgomery's method first requires the numbers to be converted to a N-residue form mod p. The result of a Montgomery multiplication will also be in N-residue form. To obtain the correct result the answer must be converted back to standard form. The main operation performed in ECC is scalar multiplication. From [8] we know that for this case, the conditional subtraction at the end of the Montgomery Multiplication algorithm is not required. Algorithm 1 shows how the Montgomery Multiplication is performed without a conditional subtraction.

Algorithm 1: Montgomery Multiplication

Input: $X' = \sum_{i=0}^{k} X'_i 2^i, Y' = \sum_{i=0}^{k} Y'_i 2^i, p$
Output: $Z' = X \cdot Y \cdot 2^{-k+2} \pmod p$
1 $Z' = 0, y_{k+1} = 0;$
2 **for** *i=1 to* $k+2$ **do**
3 $\quad q_i = Z'_{i-1} + y_i X' \pmod 2$
4 $\quad z_i = (Z'_{i-1} + q_i n + y_i X')/2$
5 **end**

A circuit for performing a Montgomery multiplication is shown in Fig. 4. The critical path of this design is through the two adders.

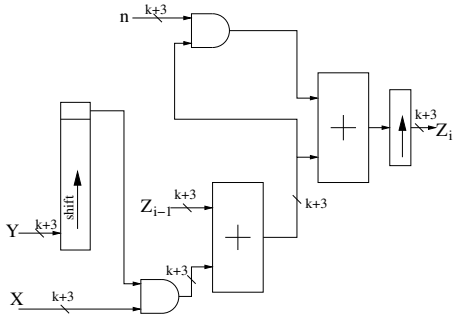

Figure 4. Montgomery Multiplier

IV. Multiplier with BRAM and DSP48Es

All of the previous multipliers that have been discussed can be implemented entirely in slice logic on an FPGA. However, FPGAs also contain large amounts of block RAM and DSP blocks. Incorporating these elements into the design of the multiplier can give an alternative for performing large multiplications.

In a Xilinx FPGA, the DSP48E and BRAM based design uses a number of DSP48E blocks to calculate the partial products. The partial products are then stored in BRAM. An adder can read from the BRAM and add the partial products. To decrease the number of clock cycles required to perform a multiplication, the number of DSP48E blocks used can be increased. An example of the design of the multiplier using four DSP48E blocks is shown in Fig. 5. The design shown consists of four DSP48E blocks, four blocks of BRAM and one adder. If the number of DSP48E blocks and BRAM is increased, the general setup of the design remains the same.

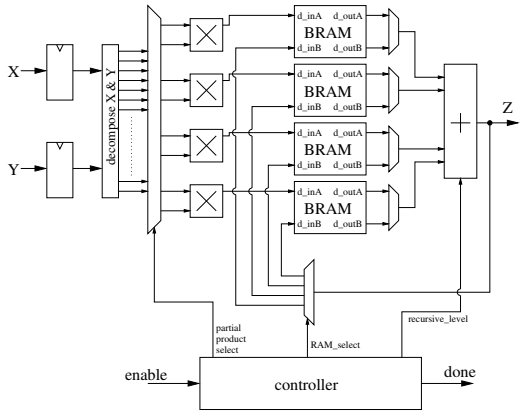

Figure 5. DSP48E and BRAM based Multiplier

Each DSP48E block on a Xilinx Virtex 5 FPGA is capable of performing a full 25×18 bit multiplication, giving a 48 bit result. However in this design we only require the DSP48E blocks to perform 18×18 bit multiplications. To take advantage of the high clock frequency of these DSP48E blocks, a large multiplication must be decomposed recursively until the size of the partial product multiplications is 18×18 bits or less. The maximum frequency of the design will not be limited by the DSP48E blocks but by the critical path of the adder used to sum the partial products. It is possible to pipeline the adder but for large bit lengths, achieving a clock frequency close to that of the DSP blocks is not possible.

The simplest way to decompose the multiplication $x \times y$, is as follows. Let *x* and *y* be represented as a binary string of length *t*. To split *x* and *y* into two equal parts let $n = t/2$, then,

$$\begin{aligned} x &= x_H 2^n + x_L \\ y &= y_H 2^n + y_L \\ x \times y &= (x_H 2^n + x_L)(y_H 2^n + y_L) \\ &= z_a 2^{2n} + z_b 2^n + z_c \end{aligned} \quad (4)$$

where, $z_a = x_H y_H$, $z_b = x_H y_L + x_L y_H$ and $z_c = x_L y_L$. This method can be applied recursively to the partial products until the calculation of z_a, z_b and z_c consist of 18×18 bit, or less, multiplications. For each decomposition there are then four partial products that have to be calculated using the DSP48E blocks, $x_L y_L$, $x_L y_H$, $x_H y_L$ and $x_H y_H$. These four partial products must then be summed according to Eqn. (4). By using the Karatsuba method for decomposition [9], the number of multiplications required to calculate the partial products can be reduced from four to three. However this increases the number of additions that need to be performed. Since the critical path in the design is through the adder, the previous method for decomposition, Eqn. (4), is used.

A. Multiplier Operation

The multiplier design based on DSPE blocks and BRAM is shown in Fig. 5. The inputs X and Y are decomposed by simply routing the correct portions of the array of bits to the multipliers. A multiplexer routes the decomposed X and Y signals into the DSP blocks, to be multiplied. The controller determines which partial products are routed to the multipliers at each clock cycle through the use of the partial product select line. For every DSP48 block in the design, there is a dual port block RAM and for every four DSP48E blocks there is a single adder.

A generator was written in C++ to generate the VHDL code for the multiplier. This was necessary due to the complexity of decomposing the multiplicands and also the multiplexer that routes inputs to the DSP48E blocks. The generator is designed to take as inputs the bit length of the multiplication and number of DSP48E blocks to be used. From these values the generator produces all the VHDL files necessary to implement the multiplier.

V. Implementation

In order to thoroughly compare the different multiplier designs, the circuits were implemented on a Virtex 5 FPGA and the power consumption measured. The SASEBO-GII evaluation board [10] is specifically designed for side channel analysis experiments on FPGAs, consequently the board is also suitable for accurately measuring the power consumption of circuits on an FPGA.

The board allows for access to the Vcc_{int} pin of the FPGA. The Vcc_{int} pin of the FPGA is a 1V power line used only to power the core of the chip. This is useful for measuring the power consumption of a circuit on an FPGA as power to the IO blocks are supplied separately. The resulting value for power consumption is that only of the circuit implemented on the FPGA. The Vcc_{int} pin of the FPGA was connected to a very accurate DC power analyser. The dynamic power consumption of each circuit was measured and averaged over the period of 30 minutes. Each circuit processed random data sent from the control

Multiplier	Bit Length	Area(slices)	Max Freq(MHz)	Clk Cycles	Throughput (Mbits/S)	Power Consumption	
						dynamic (mW)@24MHz	static (mW)
Montgomery	127	**264**	161	129	159	3.45	266.97
Serial	127	352	**186**	128	185	**1.52**	271.17
Booth2	127	353	167	65	327	6.05	**266.78**
DSP_BRAM (4DSPs)	127	1139	110	**31**	**451**	20.31	295.75
Montgomery	521	**957**	54	523	54	17.1	**276.18**
Serial	521	1280	50	522	50	**2.84**	277.59
Booth2	521	1601	**84**	525	**83**	35.59	288.98
*DSP_BRAM (4DSPs)	521	6250	52	**353**	77	-	

Table I
POWER CONSUMPTION RESULTS

FPGA during this time. The static power consumption was also measured while the circuit was not processing any data.

VI. RESULTS

Shown in Table IV-A are post place and route and power consumption measurment results for a Virtex xc5vlx50-1ff324. The clock speed for all power measurements is 24MHz. The results for 521-bit DSP_BRAM circuit, denoted by *, are taken from a xc5vlx220 as this circuit contains too much block RAM to fit on the smaller Virtex xc5vlx50. The area results for the DSP_BRAM circuits do not take into account the BRAM or the DSP48E blocks that are also present in the circuit. The best results in each category are highlighted in bold.

In order to properly route the 521 bit Booth2 circuit, a pipelined design was used. This allowed the design to be routed onto the chip without exceeding the length of the carry chain on the FPGA.

The results show that the Montgomery, Booth2 and Serial multipliers are all closely matched in the power they consume. The circuits that make use of block RAM and DSP blocks are capable of performing the multiplication much faster than the other circuits. However the area used by these multipliers is much higher than the circuits that implement the multiplication in a serial way. The standard deviation of the power consumption of the Booth2 multiplier was lowest for a bit length of 127 bits. This low standard deviation in power consumption would be a desirable property for high security applications as it should reduce the risk of the circuit being susceptible to power analysis attacks such as those described in [11] and [12]. For a comparison of ECC algorithms resistant to Simple Power Analysis attacks see [13].

VII. CONCLUSION

We have shown a comparison of the performance of various different multipliers on a Virtex 5 FPGA. The multipliers were compared for area, throughput and power consumption. The results have shown that there are many ways of implementing a modular multiplier on an FPGA, each of which has their own advantages. The Montgomery multiplier design has the advantage of being able to perform modular multiplications where the modulus is of a general form and also at very low area.

When working with a modulus of special form such as $2^n - 1$, the most efficient design is a Booth2 multiplier. The Booth2 design has a low area similar to a serial multiplier but a much higher throughput due to the recoding used.

In certain applications such as handling large amounts of traffic on a network, high speed encryption is more desirable than low area or low power design. In this situation making use of the BRAM and DSP48E resources on the FPGA may be a desirable way of implementing the multiplier in an ECC unit. Implementing the multiplier in this way also keeps the critical path of the design relatively short. Performing the full $n \times n$-bit multiplication and reduction in a single clock cycle with a fully parallel multiplier would give a very high throughput but also a long critical path. Such a design might hinder operations elsewhere on the chip if only one clock is present.

ACKNOWLEDGMENT

This material is based upon works supported by the Science Foundation Ireland under Grant No. 06/MI/006 and the Mobility Grant 2010 from the *Collège Doctoral International* of the *Université Européenne de Bretagne*.

REFERENCES

[1] P. Montgomery, "Modular multiplication without trial division," in *Mathematics of Computation*, vol. 44, 1985, pp. 519–521.
[2] *Digital Signature Standard (FIPS–186-3)*, NIST, June 2009.
[3] S. D. Galbraith, X. Lin, and M. Scott, "Endomorphisms for faster elliptic curve cryptography on a large class of curves," in *EUROCRYPT 2009, LNCS 5479*, 2009, pp. 518–535.
[4] A. Hiasat, "New Memoryless, mod$(2^n \pm 1)$ Residue Multiplier," in *Electronics Letters*, vol. 28, 1992.
[5] A. D. Booth, "A Signed Binary Multiplication Technique." *Quarterly Journal of Mechanics and Applied Mathematics,*, vol. 4, no. 2, pp. 236–240, 1951.
[6] O. MacSorley, "High-speed arithmetic in binary computers," *Proceedings of the IRE*, vol. 49, no. 1, pp. 67–91, jan. 1961.
[7] G. W. Bewick, "Fast multiplication: Algorithms and implementation," Ph.D. dissertation, Electrical Engineering, Stanford University, 1994.
[8] C. D. Walter, "Montgomery exponentiation needs no final subtractions," *Electronic Letters*, vol. 35, no. 21, pp. 1831–1832, October 1999.
[9] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Proceedings of the USSR Academy of Sciences*, vol. 145, pp. 293–294, 1962.
[10] "Side-channel attack standard evaluation board, SASEBO-GII," http://www.rcis.aist.go.jp/special/SASEBO/.
[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES 2004*, ser. LNCS, vol. 3156, 2004, pp. 16–29.
[12] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO '99*, ser. LNCS, vol. 1666, 1999, pp. 388–397.
[13] A. Byrne, N. Meloni, A. Tisserand, E. M. Popovici, and W. P. Marnane, "Comparison of simple power analysis attack resistant algorithms for an elliptic curve cryptosystem," in *Journal of Computers*, vol. 2, 2007, pp. 52–62.