

MONTGOMERY MULTIPLICATION COPROCESSOR ON RECONFIGURABLE LOGIC

Martin Šimka, Miloš Drutarovský

Department of Electronics and Multimedia Communications,
Technical University of Košice,
Park Komenského 13, 04120 Košice, Slovakia
{Martin.Simka, Milos.Drutarovsky}@tuke.sk

Abstract

In this paper we introduce a scalable Montgomery Multiplication (MM) coprocessor implemented in reconfigurable hardware. A way of connection to Altera Nios embedded processor and some improvements of design are presented.

1. Introduction

Several cryptographic algorithms, such as RSA, Diffie-Hellman key exchange algorithm or Elliptic curve cryptography use modular multiplication and modular exponentiation. The MM algorithm provides certain advantages in the implementation of modular multiplication. Thanks to the implementation in programmable logic devices (PLD) we can obtain very flexible and secure solution.

In following lines we present shortly the MM algorithm, a way of implementation the MM coprocessor and its connection to the Nios processor, improvements of the coprocessor's structure, speed and area results, and finally some conclusions.

2. MM Algorithm

Basic mathematical operation used by RSA is modular exponentiation [1]

$$Z = X^E \bmod M \quad (1)$$

that a binary or general m -nary methods can break into a series of modular multiplications. All of these computations have to be performed with large k -bit integers (typical $k \in \{1024, 2048, \dots\}$).

The well-known MM algorithm [2] speeds-up modular multiplication and squaring required for exponentiation (1). It computes the MM product for k -bit integers X, Y

$$MM(X, Y) = XYR^{-1} \bmod M \quad (2)$$

where $R = 2^k$ and M is an integer in the range $2^{k-1} < M < 2^k$ such that $GCD(R, M) = 1$.

Basic MM (2) can be used for efficient computation of (1) by the standard Montgomery exponentiation algorithm [1]. The starting point of this algorithm is the MM. The faster the MM is performed, the faster the exponentiation process will be accomplished.

Multiple Word Radix-2 Montgomery Multiplication (MWR2MM) algorithm [3] with word length w performs bit-level computations, produces word-level outputs and provides direct support for the scalable MM coprocessor design. For operands with a k -bit precision $e = \lceil k/w \rceil$ words are required. MWR2MM algorithm scans word-wise operand Y (multiplicand) and M , and bit-wise operand X (multiplier).

This work was supported by VEGA grant 1/8130/01 – Digital Signal Processing and Watermarking in Multimedia Communications.

Algorithm 1 Multiple Word Radix-2 Montgomery Multiplication

```
 $S = 0$ 
for  $i = 0$  to  $k - 1$  do
   $C = 0$ 
   $(C, S^{(0)}) = x_i Y^{(0)} + S^{(0)}$ 
  if  $S_0^{(0)} = 1$  then
     $(C, S^{(0)}) = C + S^{(0)} + M^{(0)}$ 
    for  $j = 1$  to  $e - 1$  do
       $(C, S^{(j)}) = C + x_i Y^{(j)} + M^{(j)} + S^{(j)}$ 
       $S^{(j-1)} = (S_0^{(j)}, S_{w-1..1}^{(j-1)})$ 
       $S^{(e-1)} = (C, S_{w-1..1}^{(e-1)})$ 
    else
      for  $j = 1$  to  $e - 1$  do
         $(C, S^{(j)}) = C + x_i Y^{(j)} + S^{(j)}$ 
         $S^{(j-1)} = (S_0^{(j)}, S_{w-1..1}^{(j-1)})$ 
       $S^{(e-1)} = (C, S_{w-1..1}^{(e-1)})$ 
```

3. Radix-2 coprocessor implementation

Implemented radix-2 MM coprocessor is a unit realizing the algorithm 1. Input values (X , Y , and M) as well as result (S) obtained by the coprocessor are stored in coprocessor's memory. Since input and output values can have a high bit count (1024 bits or more per object), we have decided to store inputs and intermediate results in Embedded Memory Blocks (EMBs). Therefore the coprocessor has been split into two blocks: a Radix-2 Montgomery multiplier and a dual port data memory.

In order to reduce storage and arithmetic hardware complexity, data path of MM coprocessor uses X , Y and M in a standard non-redundant form. The internal sum S is received and generated in the redundant Carry-Save form [4]. Therefore the bit resolution of the sum S is effectively doubled. The design of data path is based on the structure presented in [3]. MM unit consists of two layers of carry-save adders and it is shown for $w = 3$ in Fig. 1. Input c represents latched value t that is the least significant bit of the value $S^{(0)} + x_i Y^{(0)}$.

The most important parameter influencing the overall multiplier speed is the memory access time. Since during one cycle previous result S has to be read and current result from the last stage has to be written to the same memory, we have chosen to configure the memory block as a dual port RAM using Altera-specific function `lpm_ram_dp` from the Library of Parameterized Modules (LPM). The memory with registered input/output data has been found as the faster implementation.

After some changes in [5] the pipeline version of the coprocessor has been implemented. The data path is organized as a pipeline structure of MM units separated by pipeline registers (Fig.2). The operands for the first stage are loaded directly from the memory, to the following stage(s) are shifted through the pipeline registers.

After the timing analysis we found out a distribution of X to the stages as a critical path. Therefore this part has been changed by adding an 1-bit register for x_i into the MM unit (see Fig. 1). Thanks to this modification there are no limitations for a number of stages as it was in [6].

The maximum degree of parallelism that can be obtained with this organization is

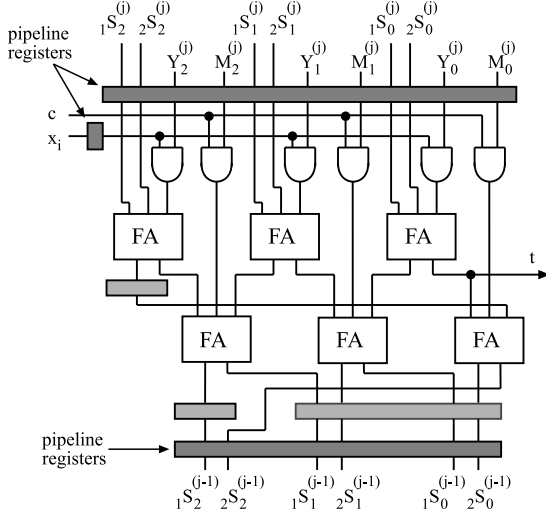


Fig. 1: Structure of MM unit for $w = 3$ (FA – Full Adder)

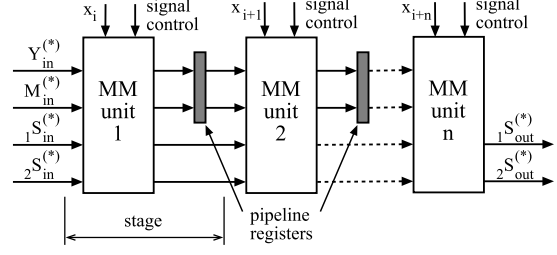


Fig. 2: Block diagram of MM coprocessor data path

found as:

$$p_{max} = \left\lceil \frac{e}{3} \right\rceil \quad (3)$$

When less than p_{max} stages are available, the total execution time will increase, but on the other hand the area occupation of the coprocessor can be changed according area constraints of a target device.

The computation time of the single MM operation when $n \leq p_{max}$ stages are used is:

$$T_{MM} = \frac{1}{f_{clk}} \left(\frac{k^2}{wn} + 3n \right) \quad (4)$$

4. MM coprocessor interface

The way in which the MM coprocessor is connected to the embedded processor is important for a control of the computing process (how the process is started and how the status of the coprocessor is checked), and for an exchange of processing data.

In previous solution [6] the status register has been used for checking the actual status of the coprocessor and the computational process. In this paper we propose the communication using an interrupt. This solution should help us in the future, when several MM coprocessors will be used for parallel computation. In this case the control with interrupts seems to be more suitable. When the computation of MM is finished, the interrupt signal of the processor is asserted until the results are read by the processor within the interrupt routine.

The second area where the most suitable solution need to be found, is the interface between the coprocessor's memory block and the bus of the embedded processor. Since there is a need for faster clocking of the coprocessor, we have assumed the case with two clock signals. The slower one is connected to the processor, and through the bus and the interface to the coprocessor. All the processes concerning the operations between the processor's bus and the coprocessor's memory block are clocked by this signal ($f_{clk_{Nios}}$). The counting operations inside the coprocessor are clocked by external faster clock signal ($f_{clk_{MM}}$). Thanks to such clock signals organisation almost three times higher performance has been obtained compared the previous implementation [6].

5. Implementation results

To map the presented MM coprocessor into Altera PLD, a VHDL-based methodology has been used. All of blocks are fully parameterized to get a solution suitable for custom application and available sources.

Presented results are generated after the synthesis in LeonardoSpectrum 2002b.21 and the Place&Route procedure in Altera Quartus II, v. 2.2 development system. Area occupation expressed in Logic Elements (LEs) and maximal possible PLD clock frequency for Altera PLD APEX 20K200E are given in Table 1 for precision $k = 1024, 2048$ bits. In Table 2 the computational time (in μs) of the single MM operation (according the equation 4) is presented for frequency value of the MM coprocessor clock signal $f_{clk_{MM}} = 90$ MHz.

Table 1: Area occupation (LEs)/max $f_{clk_{MM}}$ (MHz) of the MM coprocessor ($w = 32$ bits)

	$k = 1024$		$k = 2048$	
	LEs	$f_{clk_{MM}}$	LEs	$f_{clk_{MM}}$
$n = 1$	637	99.21	649	96.73
$n = 2$	1339	97.82	1344	92.42
$n = 3$	1927	98.45	1943	96.90
$n = 4$	2441	99.86	2451	95.68

Table 2: Computational time T_{MM} (μs) of the MM operation for $w = 32$ bits, $f_{clk_{MM}} = 90$ MHz

	$k = 1024$	$k = 2048$
$n = 1$	364.12	1456.39
$n = 2$	182.11	728.24
$n = 3$	121.46	485.55
$n = 4$	91.15	364.22

6. Conclusion

The paper describes the implementation of the MM coprocessor optimized for Altera reconfigurable devices. Thanks to the changes of the structure fast and well-parameterised solution has been found. The development of the interface will continue. The MM coprocessor is planned to be a part of a complex cryptographic processor.

References

- [1] J.A. Menezes, P.C. Oorschot, S.A. Vanstone: Applied Cryptography, CRC Press, New York, 1997.
- [2] C.K. Koc, T. Acar: Analyzing and Comparing Montgomery Multiplication Algorithms, IEEE Micro, (16) 3, pp.26-33, June 1996.
- [3] A.F. Tenca, C.K. Koc: A Scalable Architecture for Montgomery Multiplication, In C.K. Koc and C. Paar, editors: Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science No.1717, pp.94-108. Springer, Berlin, Germany 1999.
- [4] C.K. Koc: RSA Hardware Implementation, www.rsa.com, pp.1-28, August 1995.
- [5] M. Drutarovský, V. Fischer: Implementation of Scalable Montgomery Multiplication Coprocessor in Altera Reconfigurable Hardware, International Conference on Signal Processing and Telecommunications, Kosice, Slovakia, pp. 132-135, November 2001.
- [6] M. Šimka, V. Fischer: Montgomery Multiplication Coprocessor for Altera NIOS Embedded Processor, Proceedings of the 5th International Scientific Conference on Electronic Computers and Informatics 2002, Kosice, Slovakia, pp.206-211, October 2002.
- [7] APEX 20K Programmable Logic Family, www.altera.com
- [8] Nios Soft Core Embedded processor, www.altera.com/nios