IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 1, August 2013)

Implementation of Radix-2 Montgomery Multiplier in VHDL

Jaya Bansal
jayabansal3@gmail.com

Jagdish Nagar
jagdishnagar1@gmail.com

Abstract –**Low power consumption and smaller area requirement are prime concern in fabrication of DSP system on FPGA. Modular arithmetic is core operation in cryptosystems since they are efficient when data size is large (1024 bits or greater). In this paper a novel architecture of radix-2 Montgomery multiplier is presented and implemented on Vertex-iv FPGA device. Simulation shows that our design performs faster in terms of clock frequency while it requires lower area.**
Keywords– Radix-2 Montgomery multiplier, VHDL, FPGA.

I. INTRODUCTION

With the advancement in communication systems, security is a prime concern which is offered by public key cryptosystems. These systems offer authentication, confidentiality and privacy. Many cryptosystems including RSA, DSA and ECC systems requires modular multiplication for private key generation. [1] P. Montgomery developed an efficient algorithm for the calculation of (A X B) mod M called Montgomery Multiplication algorithm.
Montgomery multiplication has been used as a fundamental operation of arithmetic operations in RSAalgorithm.
This paper presents FPGA implementation of scalable architecture for radix-2 Montgomery multiplication algorithm for 1024 bit operand.

II. MONTGOMERY MULTIPLICATION

In 1985 a method for modular multiplication using Residue Number System (RNS) representation of integers is proposed by Peter L. Montgomery. In this methodthecostly division operation usually needed to perform modular reduction is replaced by simple shiftoperations by transforming the operands into the RNS domain before theoperation and re-transforming the result after operation.A radix R is selected to be two to the power of a multiple of the word size and greater than the modulus, i.e. $R = 2^w > M$. For the algorithm to work R and M need to be relatively prime, i.e. must not have any common non-trivial divisors. With R a power of two, this requirement is easily satisfied by selecting an odd modulus. This also fits in nicely with the cryptographic algorithms that we are targeting, where the modulus is either a prime always odd with the exception of 2or the product of two primes and therefore odd as well.

RNS representations of integers are called M residues and are usually denominated as the integer variable name with a bar above it. An integer a is transformed into its corresponding M-residue $\bar{a}$ by multiplying it by R and reducing modulo M. The back-transformation is done in an equally straight forward manner by dividing the residue by R modulo M. Thus here are the following equations as transformation rules between the integer and the RNS domain:

$$\bar{a} = aR(mod\ M)$$

$$a = \bar{a}R^{-1}$$

Montgomery Multiplication can be defined simply as the product of two M residues divided by the radix modulo M:

$$\bar{c} = \bar{a}\bar{b}R^{-1}(mod\ M)$$

Division by the Radix is required to make the result again an M-residue.

III. RADIX-2 MONTGOMERY MULTIPLICATION ALGORITHM

M be any odd integer which is greater than zero for satisfying radix-2 operation and X, Y are two operands. Montgomery multiplication involves first transformation of operands into Montgomery domain and then after result is re-transformed into Montgomery domain. This conversion process replaces division by several shift operations.

Let X and Y be two n-bit operands then Montgomery multiplication process is described as follows:

$(X,Y) \bmod M = X'.Y'.2^{-n}$.

Where $X' = X.2^n$.

$Y' = Y.2^n$.

Hence $(X,Y) \bmod M = (X.2^n).(Y.2^n).2^{-n}$

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 1, August 2013)

$=X.Y.2^{n}$

Algorithm 1. Radix-2 Montgomery Multiplication

***Input**: odd M, n = [Log₂ M] +1,*

$$X = \sum_{i=0}^{n-1} x_i 2^i, with\ 0 \le X, Y < M$$

Output:

Z=MP(X, Y, M) ≡ X. Y .2⁻ⁿ (mod M),

With 0≤ Z < M.

S [0] =0;

For i=0 to n-1 do

$q_i = (x_i.Y_0) \oplus S[i]_0;$

S [i+1] =(S[i] +xᵢ. Y +qᵢ. M)/2;

If S[n]>M then

S[n] =S[n]-M;

Return Z=S[n];

IV. MONTGOMERY MULTIPLIER ARCHITECTURE

The interface of Montgomery modular multiplier is shown in Fig.1. It receives the operands *X*, *Y* and *M* and it returns
$R = (X'Y'2^{-n}) \bmod M$. X and Y are 1024 bits respectively.

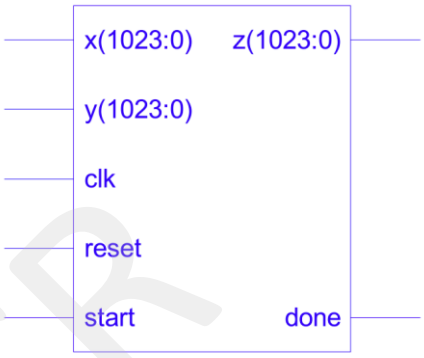

Figure 1:Montgomery Multiplier for 1024 bit data

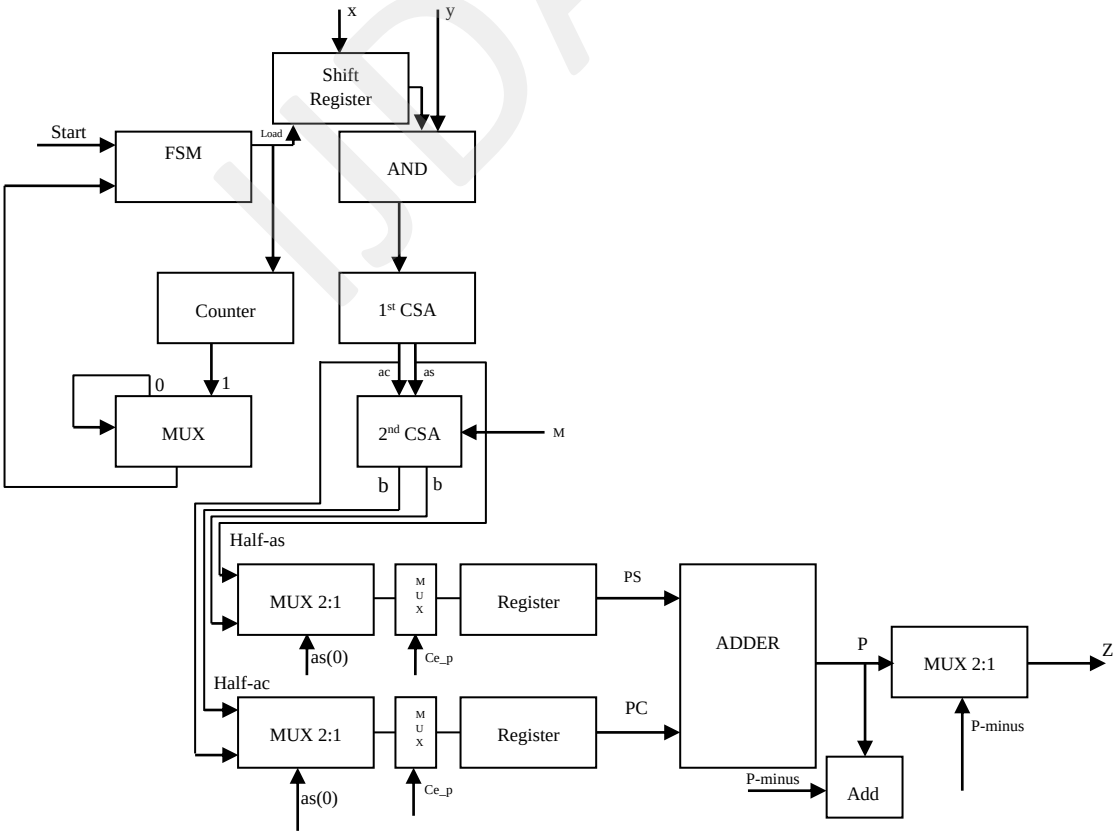

Figure 2: Architecture for Montgomery Multiplier

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 2, Issue 1, August 2013)

The detailed architecture of radix-2 Montgomery multiplier is shown in fig.2. It contains two carry save adders, a shift register, 6-multiplexer, and a control unit.

The input X is passed through shift register while input Y is directly applied to first carry save adder which implements$R+(X_i*Y)$ and second adder gives R+M.

The control unit controls the operation of entire process: Finite State Machine with three states is used to control multiplication process.

- S_0: initialization of the state machine; Ce_p=0;load=0; Go to S_1;
- S_1: load multiplicand and modulus into registers; load multiplicand into shift register; Ce_p=0;load=1; done=0; Go to S_2;
- S_2: wait for $ADDER_1$; Wait for $ADDER_2$; Ce_p=1; load=0; done=0;

V. RESULT

Number of Slices:	6536 out of 10240 63%
Number of Slice Flip Flops:	3224 out of 20480 15%
Number of 4 input LUTs:	12464 out of 20480 60%
Maximum Frequency:	353.576MHz

Performance of 1024-bitMontgomery Multiplier on Vertex-IV FPGA

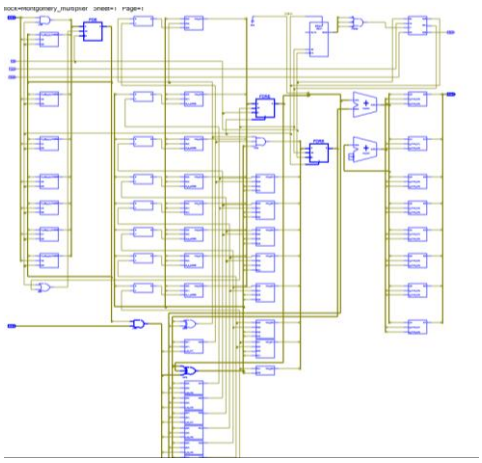

Fig.3. RTL Schematic of Multiplier

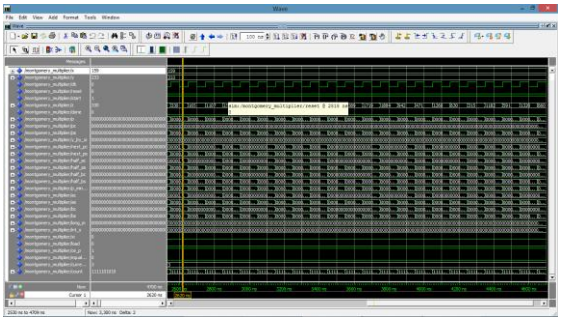

Fig.4. Simulation of Montgomery Algorithm

VI. CONCLUSION

In this paper a novel method for implementing 1024-bit Radix-2 Montgomery multiplier on FPGA is discussed. As clear form the results our design performs best in terms of clock speed while maintaining lower area requirement.

REFERENCE

[1] P. Montgomery, "Modular multiplication without trial division," Mathematics of Computation,vol. 44, pp. 519–21, April 1985.

[2] A.F. Tenca and C, .K. Koc,, "A Scalable Architecture for MontgomeryMultiplication," Proc. First Int'l Workshop Cryptographic Hardwareand Embedded Systems (CHES '99), pp. 94-108, 1999.

[3] A.F. Tenca and C, .K. Koc,, "A Scalable Architecture for ModularMultiplication Based on Montgomery's Algorithm," IEEE Trans.Computers, vol. 52, no. 9, pp. 1215-1221, Sept. 2003.

[4] D. Harris, R. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu,"An Improved Unified Scalable Radix-2 Montgomery Multiplier,"Proc. 17th IEEE Symp. Computer Arithmetic (ARITH), pp. 172-178,June 2005.

[5] Klein, M., Rogers G.J., Moorty S., and Kundur, P., "Analytical investigation of factors influencing power system stabilizers performance", IEEE Transactions on Energy Conversion, Volume: 7 Issue: 3, Sept. 1992, Page(s): 382 -390.

[6] N. Jiang and D. Harris, "Parallelized Radix-2 Scalable MontgomeryMultiplier," Proc. IFIP Int'l Conf. Very Large Scale Integration(VLSI-SoC '07), pp. 146-150, Oct. 2007.

[7] D. Amanor, V. Bunimov, C. Paar, J. PelzlandM. Schimmler, "Efficient HardwareArchitectures for Modular Multiplication onFPGAs", International Conference on FieldProgrammable Logic, ReconfigurableComputing, and Applications. August 24-28,(2005), Tampere, Finland.