Efficient FPGA Implementation of Montgomery Multiplier Using DSP Blocks

Arpan Mondal, Santosh Ghosh,
Abhijit Das, and Dipanwita Roy Chowdhury

Department of Computer Science and Engineering
IIT Kharagpur, WB - 721302, India
arpanmondal@live.com, santosh.ghosh@gmail.com,
{abhij,drc}@cse.iitkgp.ernet.in

Abstract. In this paper, an efficient Montgomery modular multiplier is designed exploiting the efficiency of inbuilt multiplier and adder soft-cores of DSP blocks. 256×256 bit multiplier has been implemented with (i) fully parallel, (ii) pipelined and (iii) semi parallel architectures that consumes upto 16 DSP48E1 64×64 bit soft-cores provided by Xilinx 12.4 ISE Design Suite. Performances with respect to area, operating frequency and design latency have been compared.

Keywords: Montgomery Multiplier, FPGA design, DSP blocks.

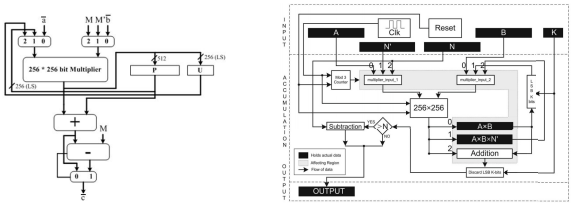
1 Introduction

The most critical operation behind all cryptographic operations is the chained modular multiplication. The efficiency of the implementations is measured by the computing power, energy consumption and memory usage. The proposed method of Peter L. Montgomery[2], in 1985, is a very efficient algorithm for modular multiplication in hardware. The input variables are transformed into Residue Number System (RNS) to replace the costly division operation of hardware by simple shift operations. The re-transformation of the output from RNS to integer domain gives the actual result. Montgomery Multiplication[2] is feasible for the algorithms that involves lot of multiplications with respect to the same modulus as the ratio between transformation overhead and actual modular arithmetic becomes much lower.

2 Efficient Design of Montgomery Multiplier

The implementations of the Montgomery multiplier is classified mainly into two categories : bit-wise and block-wise. Implementation of the entire algorithm without using any explicit multiplication tends to a complex design of the algorithm.

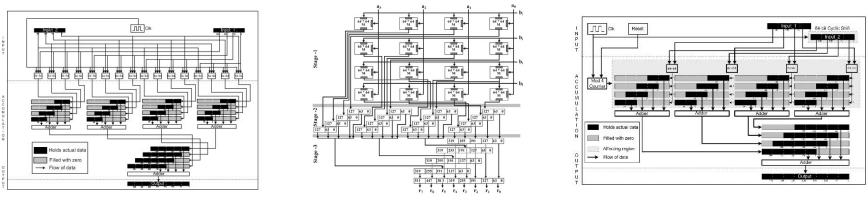
H. Rahaman et al. (Eds.): VDAT 2012, LNCS 7373, pp. 370–372, 2012.



(a) Reduction Block (b) Proposed Architecture

Fig. 1. 256 × 256 bit Montgomery Multiplication

The architecture shown in Fig.1(a) is the basic block diagram of the Montgomery Reduction Algorithm. Detailed architecture for hardware implementation is given in Fig.1(b). The 256 × 256 bit multiplier is the main multiplication block which affects the overall performance of the architecture. Fully parallel, pipelined and semi-parallel multiplication blocks, shown in Fig. 2(a), 2(b) and 2(c) respectively, built using the efficiency of modern FPGAs have been applied in the design to compare and analyze the performance.

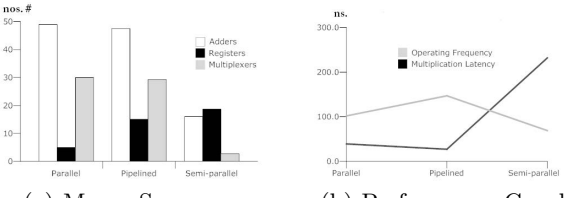


(a) Parallel Design (b) Pipelined Design (c) Semi-parallel Design

Fig. 2. 256 × 256 bit Multiplier Architectures

3 Experimental Results and Analysis

Fig. 3(a) and 3(b) respectively shows the macro statistics and the performance graph of the implemented designs. Comparison with similar FPGA based implementation has been done in Table 1 providing the design details.



(a) Macro Summary (b) Performance Graph

Fig. 3. Diagrammatic Representation of Performance

Table 1. Comparison With Similar FPGA-based Implementation

Implementation	Device	Logic blocks	DSP blocks	Freq. (MHz)
Huang et al.[1]	Xilinx 6000FF1517-4	2176 Slices	64	116.4
Proposed (Semi-parallel)	Xilinx XCHVHX250T	1572 Slices	4	69.94
Proposed (Fully-parallel)	Xilinx XCHVHX250T	2106 Slices	16	102.67
Proposed (Pipelined)	Xilinx XCHVHX250T	2346 Slices	16	147.62

4 Conclusion

In this paper, we have presented a block-based area optimized hardware design of Montgomery Multiplication on DSP platform. The power consumption of the design is low because of the usage of DSP48E1 soft-cores. The performance of the design in terms of computational speed and resource balancing proves the overall effectiveness and the general validity of the approach.

References

1. Huang, G., El-Ghazawi: New Hardware Architectures for Montgomery Modular Multiplication Algorithm. IEEE Transactions on Computers, 923–936 (2011)
2. Montgomery, P.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)