Efficient Area and Delay Profile Architecture of Asynchronous Parallel Self Timed Adder Based Montgomery Multiplication

Vijeta Raichur[1], Prof. Laxminarayan Gahalod[2]

[1]M. Tech. Scholar, [2]Guide

Department of Electronics and Communication Engineering, LNCT, Bhopal

Abstract - With the ongoing digital revolution and advances in high performance computing, powerful desktop computer systems are available to almost everybody at low cost. While there has always been a demand for hardware implementations of public key cryptography, the volume has risen dramatically in recent years, due to a paradigm. Because of its complexity, public–key cryptography is mainly used for digital signatures and the management of secret keys between two points. The encryption of bulk data is mainly established with secret–key cryptosystems, whereas the secret keys to be shared for a pair of users are distributed by public–key cryptosystems. Increasing demand for modular multiplication requires fast modular multiplication algorithms such as Montgomery multiplication the implementation, verification and testing of a Montgomery modular multiplication algorithm for the new efficient area and delay profile architecture of asynchronous parallel self timed adder based Montgomery multiplication reported in this work. This architecture proposes improvements on the well–known Montgomery multiplication algorithm and its previous implementations. The implementation and synthesis of proposed work has completed on Xilinx ISE design suite using hardware descriptive language HDL. The performance of proposed architecture has been evaluated based on area and delay profile.

Keywords- Asynchronous circuit, Parallel self timed adder, Montgomery Multiplication, delay profile, cryptosystems. FFT

I. INTRODUCTION

To perform the complicated application like Public Key Cryptographic (PKC) algorithms, such as RSA, Diffie-Hellman, and Elliptic Curve Cryptography, requires modular multiplication of very large operands (sizes from 160 to 4096 bits) as their core arithmetic operation operation reasonably fast, general purpose processors are not always the best choice. This is why specialized hardware, e.g. in the form of cryptographic co-processors, become more attractive.

Based upon the analysis of recent works on the hardware design for modular multiplication, this work presents a new architecture efficient area and delay profile architecture of asynchronous parallel self timed adder. To our knowledge this is the algorithm for Montgomery's method is realized using asynchronous parallel self timed adder that can perform two different types of arithmetic.

Montgomery proposed an algorithm for modular multiplication. This new algorithm, called Montgomery Multiplication Algorithm, has the advantage of replacing division operations by bit shift operations. If the least significant bits to be shifted out are not zero, Montgomery's algorithm adds multiples of modulus to clear these bits before shifting them out.

In regular modular multiplication, after all bits of the multiplicand are processed, modulus is repeatedly subtracted from the result unless the result is less than the modulus. In Montgomery multiplication, bits are shifted out as each bit of the multiplicand is processed, leaving no need for the subtractions.

The flexibility of architecture to work with different types of arithmetic is a key feature for modern information security applications. A wealth of competing crypto-graphic algorithms exists and has been standardized. Supporting a broad range of these algorithms is no longer optional, but a necessity. The most promising model of addressing this issue in the design of a cryptographic co-processor is to add efficient arithmetic and logic primitives to a standard microprocessor / -controller architecture. This ensures an upgrade path to support future algorithms and changes to existing schemes, while preserving the speed advantages of a specialized design.

Unified Architecture: Architecture is said to be unified when it is able to work with operands in both prime and binary extension fields using the same hardware. It has been shown that a unified multiplier is feasible with only minor modifications to the multiplier for GF(p).

Dual–Radix Architecture: A unified multiplier is said to be dual–radix if it operates with a larger radix value for GF(2n) than the radix used for GF(p). The term, architecture, is used to represent the hardware of the Montgomery multiplier.

Dual radix multiplier design has critical time–area considerations, as the cost of extra radix should not effect the signal propagation time much while keeping the silicon area as low as possible.

II. SYSTEM MODEL

The fundamentals of asynchronous systems such as handshake protocols, bundled-data and delay-insensitive data encoding schemes, modes of operation, various classes of asynchronous circuits based on the timing models adopted, Muller's C-element and the concept of indication, and the notion of a function block are discussed briefly.

a. Handshake Mechanism

Asynchronous systems come in many flavours with the most prominent among them being bundled-data and dual-rail data encoding schemes. The communication protocol among these systems can also assume two forms: 2-phase (transition signalling) and 4-phase (level- sensitive signalling). Bundled-data encoding with 2-phase signalling and dual-rail data encoding with 4-phase signalling have been the popular choices in asynchronous circuit design until now and so they will be described here to provide relevant background information. In fact, dual-rail data encoding with level sensitive signalling continues to attract attention, as it is tolerant to variations in logic elements and communicating signal wires and hence has become attractive for deep submicron technologies.

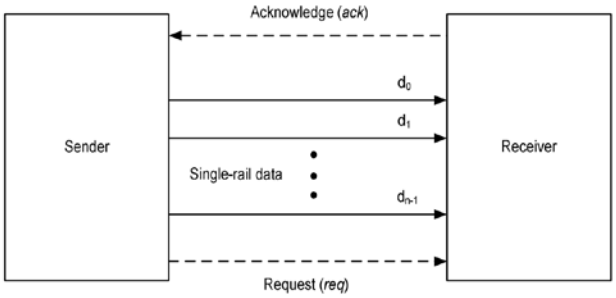

Figure 2.1 Bundled- data encoding and 2- phase handshaking.

b. Bounded and Unbounded Delay Models

Asynchronous circuit design methodologies can generally be categorised based on the timing models. Bounded delay models assume that the delay in all circuit elements and wires is known (thereby bounded). Circuits based on this model, coupled with the fundamental mode assumption, are generally referred to as Huffman circuits. There are two basic assumptions underlying this model: i) only one input to the circuit is allowed to change at a time, and ii) the present-state entries of the combinational logic can change only after the logic has settled in response to a new input – this condition, when viewed along with the first constraint leads to the understanding that multiple input changes would necessitate multiple iterations by the non-regenerative logic thereby increasing the number of cycles required to complete computation.

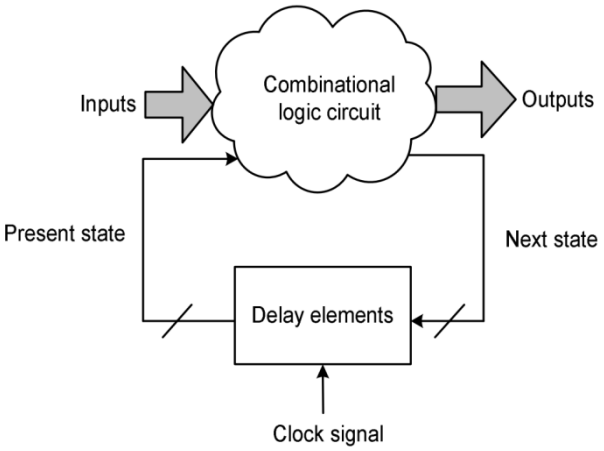

Figure 2.2 Fundamental mode system configuration.

c. C-element and Indicatability

The C-element, introduced by Muller, is an important gate widely used in asynchronous circuits and is the key element for implementing robust asynchronous logic. The symbol, Boolean equation and a transistor level realisation of the 2-input C-element (CE2) with weak feedback are shown in the figure below.

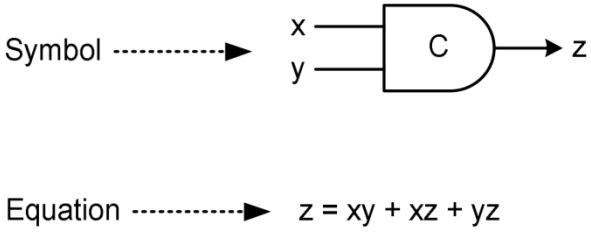

Figure 2.3 input C-element.

The CE2 outputs a 'high', when both its inputs are 'high' and outputs a 'low', when both its inputs become 'low'. In general, a random size C-element waits for all its inputs to become high (low) before producing a similar logic level at its output.

d. PASTA

The classical design of PASTA has been shown in figure. The input selection for two input multiplexers corresponds to the Request handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL=0and will switch to feedback/carry paths for subsequent iterations using SEL=1. The feedback path from the HAs enables the multiple iterations to

continue until the completion when all carry signals will assume zero values.

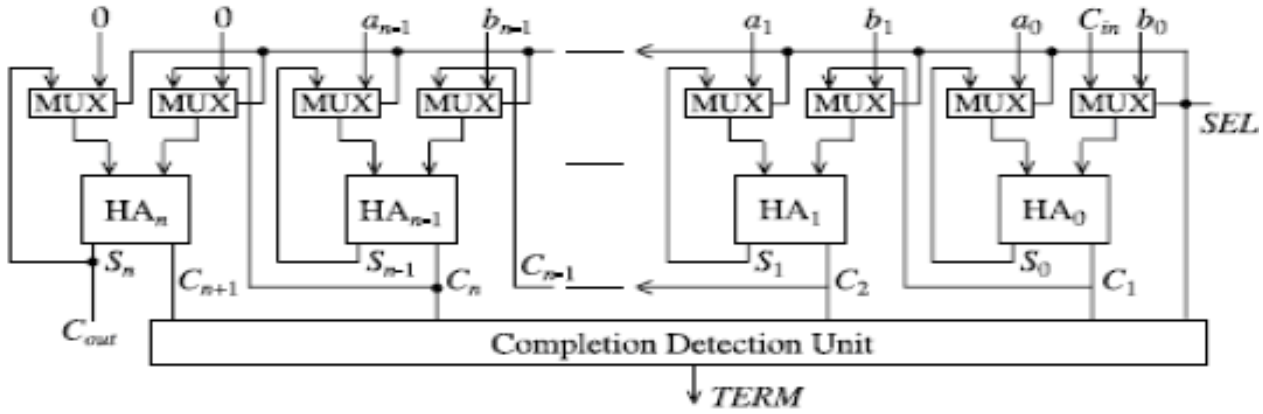

Figure 2.4 Basic block diagram of PASTA.

III. PROPOSED SYSTEM

The simplest way of adapting Montgomery's algorithm to large operand sizes would hence be, to just replace every arithmetic operation by its multi-precision equivalent. The criteria for selecting the most suitable algorithm are not limited to the number of multiplication operations alone. The specific architecture targeted for the implementation also plays an important role. Asynchronous parallel self timed adder based method is the most suitable one for implementation of modular multiplication. An area and delay efficient system has been proposed in this work implemented on Xilinx ISE design suite. RTL schematic of top module of proposed work has been shown in figure 3.1. To implement proposed system an efficient asynchronous parallel adder has been utilized to design a modular multiplication algorithm. As shown in figure there are two input pints in proposed design A (63:0) and B (63:0) are the 64 bit input vectors. M (63:0) is the 64 bit modular multiplier, clk is a clock input. Product (63:0) is the 64 bit product output. The proposed architecture is very useful in complex cryptographic application.

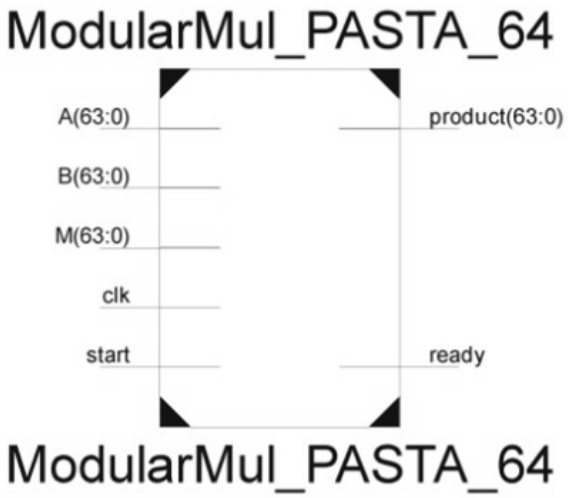

Figure 3.1 RTL Schematic of Top Module of Proposed Architecture

Sub module of proposed module has been shown in Figure 3.2 RTL schematic of sub modules of proposed architecture. RTL schematic of expanded view of sub module has been shown in figure 3.3. Primitive arithmetic operations such as multiplication and addition are limited to a certain word size w. Operands of cryptographic algorithms, on the other hand, tend to be very large, so that multiple precision arithmetic comes into existence. The common trade-off when it comes to implementation of an algorithm in hardware versus one in software is that flexibility is sacrificed for speed. There are a number of different ways to improve on the performance of complex operations in hardware. While logic and arithmetic operations take at least one clock cycle each in software implementations, multiple logic operations can be combined into a single clock cycle in custom built hardware.

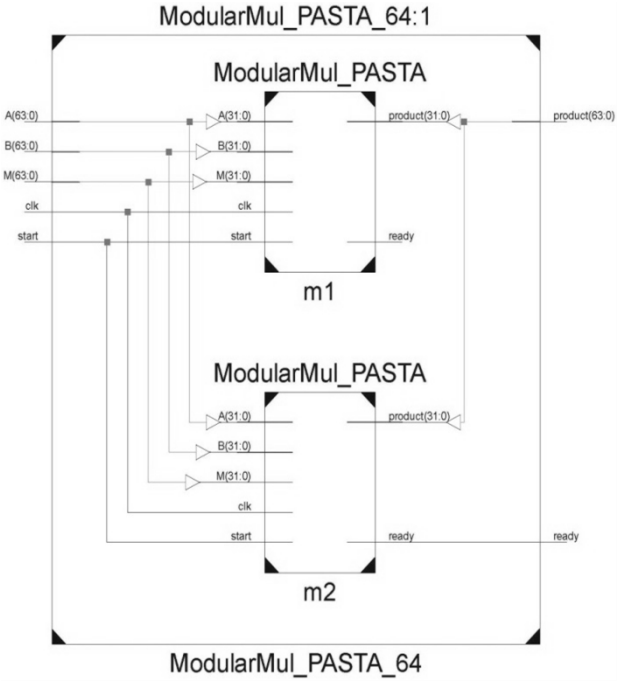

Figure 3.2 RTL Schematic of Sub Modules of Proposed Architecture

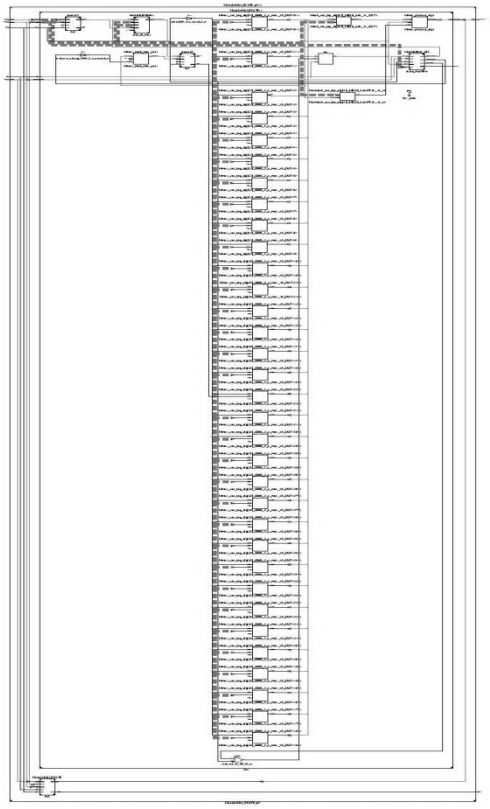

Figure 3.3 RTL Schematic of Expanded View of Sub Modules of Proposed Architecture

SYNTHESIS OUTCOMES

Synthesis of proposed work has been done on Xilinx ISE simulated on ISIM HDL simulation environment. The components of the design were described the in structural VHDL code. This approach makes the performance of the design less dependent on the synthesis features of the VHDL compiler suite. The correct function of the components has been verified using a VHDL testbench. A testbench essentially is a piece of behavioral VHDL code without any signals to the outside, that instantiates the component that is to be tested, also called Device Under Test (DUT), feeds specific data to its inputs (test vectors or patterns) and reads back the results, comparing them to the expected results. Figure 4.1 synthesis screen of proposed work on Xilinx ISE 13.1.

The timing summary of proposed design has been shown in Figure 4.2 device utilization statistics and timing summary of proposed architecture achieved Clock period: 5.867ns (frequency: 170.457MHz). Table 1 shows the Implementation results comparison with the previous architecture.

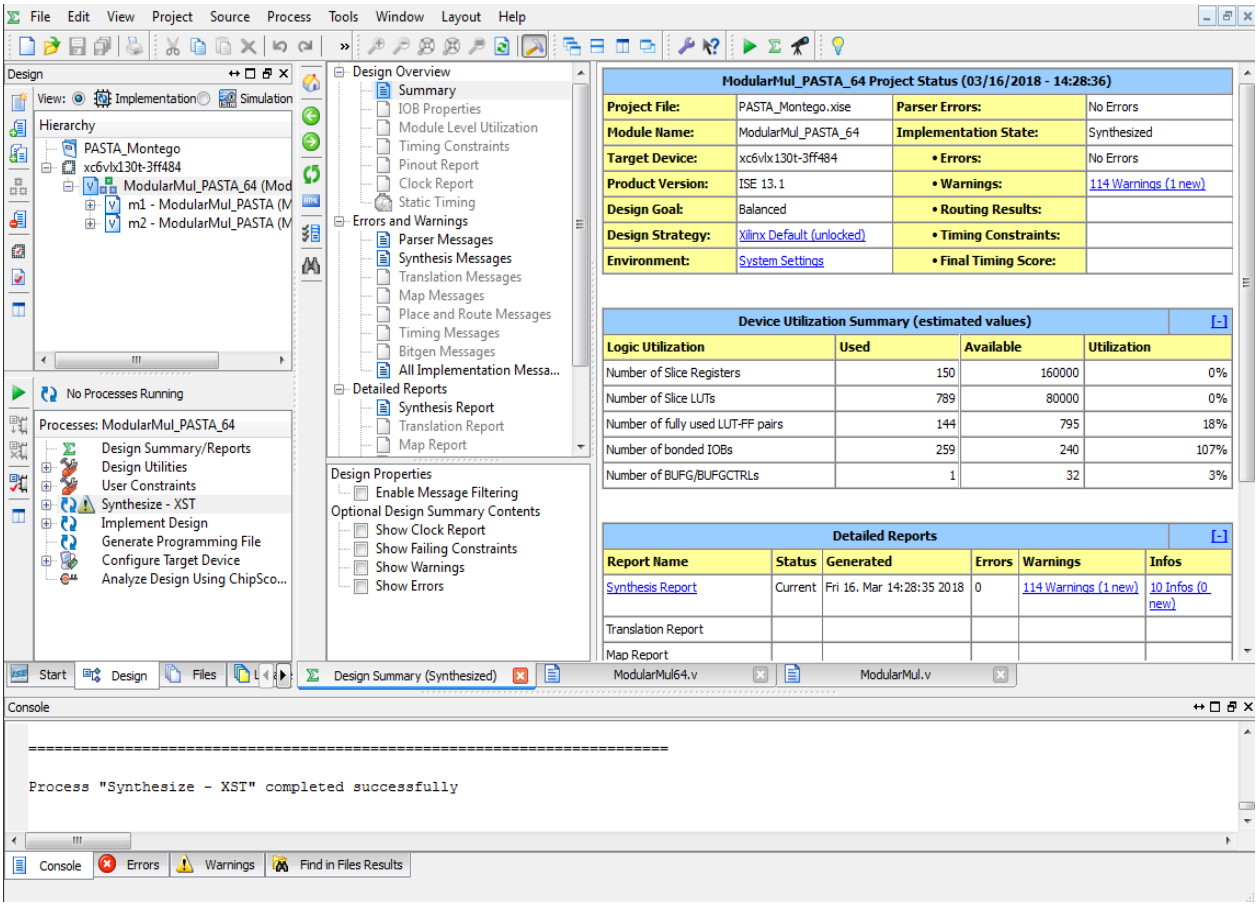

Figure 4.1 Device Utilization Summary of the Implementation on XILINX UI.

```
Device utilization summary:
---------------------------

Selected Device : 6vlx130tff484-3


Slice Logic Utilization:
 Number of Slice Registers:             150  out of  160000     0%
 Number of Slice LUTs:                  789  out of  80000      0%
    Number used as Logic:               789  out of  80000      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    795
   Number with an unused Flip Flop:     645  out of    795    81%
   Number with an unused LUT:             6  out of    795     0%
   Number of fully used LUT-FF pairs:   144  out of    795    18%
   Number of unique control sets:         4

IO Utilization:
 Number of IOs:                         259
 Number of bonded IOBs:                 259  out of    240   107%
=========================================================================

Timing Details:
---------------
All values displayed in nanoseconds (ns)
=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 5.867ns (frequency: 170.457MHz)
  Total number of paths / destination ports: 14375 / 214
-------------------------------------------------------------------------
```

Figure 4.2 Device Utilization Statistics and Timing Summary of Proposed Architecture.

Table 1: Implementation Results Comparison with the Previous Architecture

Parameters	Previous Architecture	Proposed Architecture (PASTA)
Transform length / Number of digits (P)	64	64
Bit length of each digit (u)	32	32
LUTs	13975	789
Slices	3928	150
Period	5.34 ns	5.86 ns
Latency / Delay	5.67 us (5670 ns)	17.406 ns

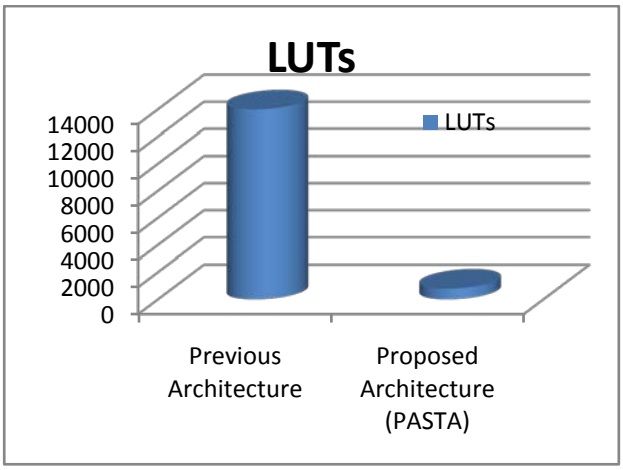

Figure 4.3 Graphical comparison of proposed work with existing work in terms of LUTs

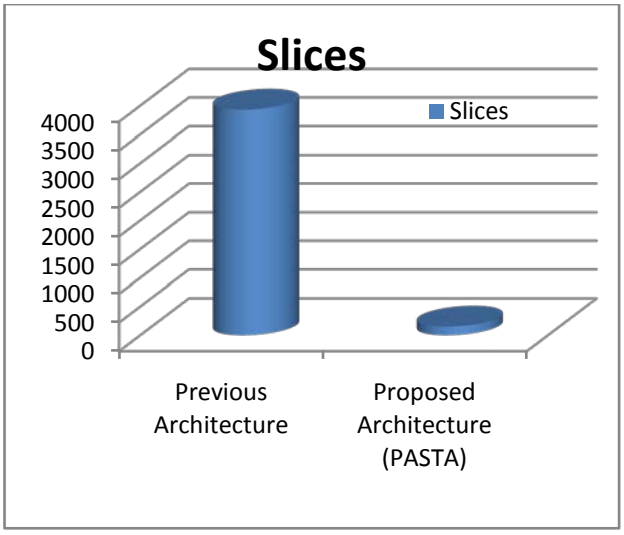

Figure 4.4 Graphical comparison of proposed work with existing work in terms of Slices.

IV. CONCLUSION AND FUTURE SCOPES

A new architecture for efficient area and delay profile architecture of asynchronous parallel self timed adder based Montgomery multiplication algorithm has been proposed, which combines positive features from previously proposed architectures with recent advances in digit multiplier design. The outcome of proposed work has highly scalable design with the ability to perform integer and binary arithmetic multiplication operation at high

speeds. Analysis and comparison of previous work results with proposed work results proved the efficiency and advances of proposed work. The most important outcome of this analysis was to identify delay and area utilized to implement and synthesize design. In this work the architecture has been implemented in VHDL, synthesized and tested successfully.

The architecture presented in this work can be considered for future work to improve performance in terms of high seeped and less area. In the current/future era where issues such as reliability and variability tend to assume greater significance than quality-of-results. The mathematical proof of the control logic for these architectures can be an interesting area of study.

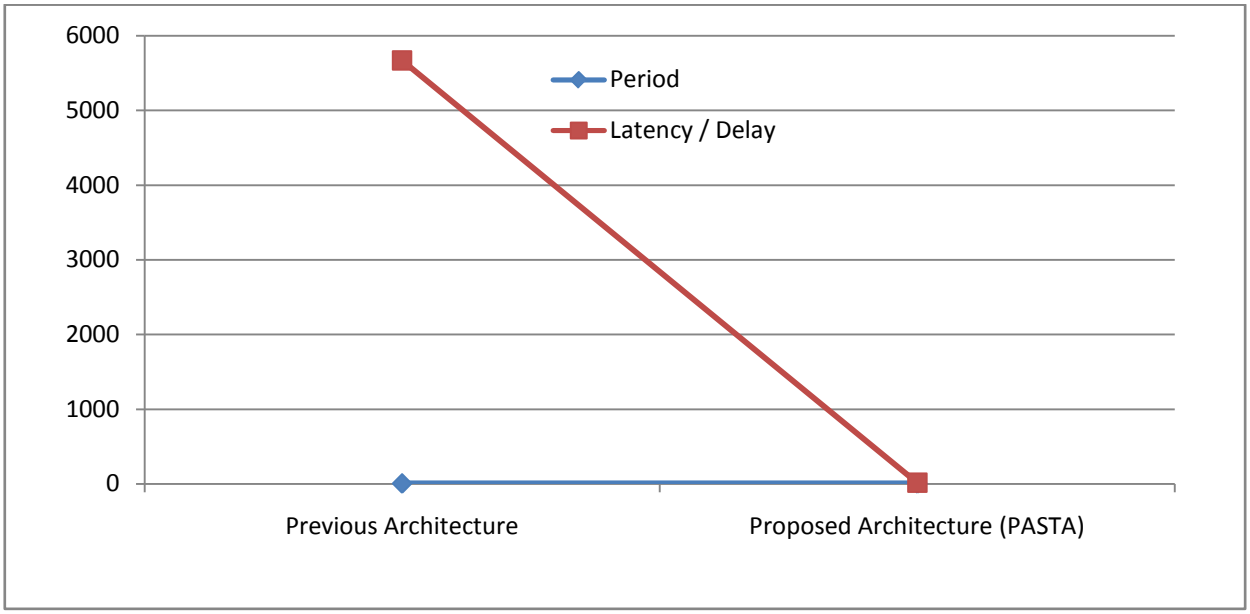

Figure 4.5 Delay comparisons.

REFERENCES

[1] S. Kavyashree and B. V. Uma, "Design and implementation of different architectures of montgomery modular multiplication," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 1101-1105.

[2] P. M. C. Massolino, L. Batina, R. Chaves and N. Mentens, "Area-optimized montgomery multiplication on IGLOO 2 FPGAs," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, 2017, pp. 1-4.

[3] Leelavathi G, Shaila K and Venugopal K R, "Elliptic Curve Cryptography implementation on FPGA using Montgomery multiplication for equal key and data size over GF(2m) for Wireless Sensor Networks," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 468-471.

[4] S. R. Kuang, K. Y. Wu and R. Y. Lu, "Low-Cost High-Performance VLSI Architecture for Montgomery Modular Multiplication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 2, pp. 434-443

[5] J. C. Néto, A. F. Tenca and W. V. Ruggiero, "CRT RSA decryption: Modular exponentiation based solely on Montgomery Multiplication," 2015 49th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2015, pp. 431-436

[6] W. Dai, H. Wu and R. C. C. Cheung, "Time-efficient computation of digit serial Montgomery multiplication," 2014 International Symposium on Integrated Circuits (ISIC), Singapore, 2014, pp. 212-215

[7] M. Mohammadi and A. S. Molahosseini, "Efficient design of Elliptic Curve Point Multiplication based on fast Montgomery modular multiplication," ICCKE 2013, Mashhad, 2013, pp. 424-429.

[8] J. Han, S. Wang, W. Huang, Z. Yu and X. Zeng, "Parallelization of Radix-2 Montgomery Multiplication on Multicore Platform," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 12, pp. 2325-2330.

[9] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, 1978.

[10] R. L. Rivest, “A description of a single-chip implementation of the RSA cipher,” Lambda, vol. 1, no. Oct.–Dec., pp. 14–18, 1980.

[11] P. L. Montgomery, “Modular multiplication without trial divi- sion,” Mathematics Comput., vol. 44, no. 170, pp. 519–521, 1985.

[12] A. Karatsuba and Y. Ofman, “Multiplication of multidigit num- bers on automata,” Soviet Physics Doklady, vol. 7, 1963, Art. no. 595.

[13] S. A. Cook and S. O. Aanderaa, “On the minimum computation time of functions,” Trans. Amer. Math. Soc., vol. 142, pp. 291–314, 1969.

[14] A. Scho€nhageand V. Strassen, “Schnelle multiplikation Großer Zahlen,” Computing, vol. 7, no. 3/4, pp. 281–292, 1971.

[15] M. Fu€rer, “Faster integer multiplication,” SIAM J. Comput., vol. 39, no. 3, pp. 979–1005, 2009.
[16] D. Harvey, J. van der Hoeven, and G. Lecerf, “Even faster integer multiplication,” CoRR, vol. abs/1407.3360, 2014. [Online]. Available: http://arxiv.org/abs/1407.3360