International Journal of Engineering, Science and Mathematics
Vol. 6Issue 8, December 2017, (Special Issue)
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijesm.co.in, Email: ijesmj@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Design of RNS Montgomery Multiplication Architecture

N.Nandini*

B.V.R.Gowri**

B.Siva Prasad***

Abstract

A design methodology for incorporating Residue Number System(RNS)/polynomial Residue Number (PRNS) in Montgomery modular multiplication in r respectively, as well as a VLSI architecture of a dual -field residue arithmetic Montgomery multiplier are presented in this paper. As analysis of input/output conversions to/from residue representations, along with the proposed residue Montgomery multiplication algorithm, revels common multiply-accumulate data paths both between the converters and between the two residue representations. A versatile architecture is derived that supports all operations of Montgomery multiplication in and, input/output conversions, Chinese Reminder Theorem(CRT)/Mixed Radix Conversion(MRC) for integers and polynomials, dual -field modular exponentiation and inversion and same hardware. Detailed comparisons withstate-of-the-art implementations prove the potential of residue arithmetic exploitation in dual-field modular multiplication. Based on irreducible pentanomials a low latency Montgomery multiplication is presented over GF(2^m).To decompose the multiplications into a number of independent units an efficient algorithm is presented to achieve the parallel process. "pre-computed addition" to reduce the latency further method is introduced. Comparing with other existing methods the proposed method having the lesser area delay and power delay. In case of NIST recommended pentanomials, it having the same or shorter critical path and involve nearly one-fourth of the latency.

Keywords:

Xilinx;RNS Value;

Montgomery;

Mixed radix conversion.

Author correspondence:

N.Nandini.
PG Scholar,
Baba Institute of Technology and Sciences,Visakhapatnam.

1.Introduction

Everyday transactions from few years ago required some physical existence have been replaced by electronic applications. Users may now use friendly, fast and safe interfaces to perform easily numerous tasks, varying from money transfers using the web and remote health checks to e-learning and e-commerce. Being

exposed in an unprecedented number of threats and frauds, safe connectivity for all network-based systems has now become a predicate necessity. The science of cryptography provides the necessary tools and means towards this direction. Cryptographic hardware and software play now a dominant role in e-commerce, mobile phone communications, military applications, and private emails, digital signatures for e-commerce, ATM cards, and web banking, maintenance of health records and so on.

The systematic arithmetic realization over finite fields of the form GF (2n) , where n ϵ Z and n $\geq$ 1 , or the form GF (p) here p is a prime. For the security purpose cryptographic application form special case, for this they need large integer operands. For achieving a favorable cryptosystem performance, the effective field multiplication with large operands is essential, since multiplication consumes most time and area to employ modular arithmetic there is a need to increase cryptosystems conveying modular arithmetic.

The parallelization of their operations is an obvious technique to achieve this method. Due to their performing ability of fast and parallel arithmetic RNS and PRNS have renewed scientific interest due their ability. There is no need for exchanging information between paths but using RNS/PRNS a given path serving a larger range of data is replaced by parallel paths. It results residue system helps to reduce the complexity and power consumption of larger word length arithmetic units. Otherwise, RNS/PRNS executions tolerate the extra cost of output converters to interpret from PRNS or RNS to binary representations and input converters to interpret numbers from a standard binary format into residues.

In this paper a dual field Montgomery modular multiplication algorithm for integer GF(p) and for polynomial GF(2n) are presented in a new methodology for embedding residue arithmetic. Valid PRNS or RNS incorporation are examined which are need to be satisfied mathematical conditions. The resulting architecture is highly parallelizable and versatile, as it supports RNS/PRNS-to-binary and binary-to-RNS/PRNS conversions, Mixed Radix Conversion (MRC) for integers and polynomials, dual-field Montgomery multiplication in the same hardware.

2.SYSTEM DESIGN

Recently, numerous number of systems have been anticipated to make computers more powerful. One of these systems, which has benefits in computing large numbers, is Residue Number System (RNS). Residue Number System is a particular interest in computing large numbers due to its properties of parallelism, carry-free, and high-speed arithmetic. In RNS, we first choose a set of relatively prime moduli to be the base of this system. Then, the numbers in RNS are represented by the residue of each

modulus, and the computations can be performed on each residue independently. Thus, RNS can be applied to many applications, such as digital signature scheme and signal processing .To apply RNS in large number arithmetic, the conversion between RNS and the decimal number system is an important issue. Recently, many researches [2, 3, 9] have been proposed to simplify and accelerate the conversion by choosing a specific set of moduli. Most of these researches choose each modulus being near the power of 2 to reduce the conversion time, such as {2n-1, 2n, 2n+1} and {2n-1, 2n, 2n+1} [2, 3]. However, for the general moduli set, the conversion can only be done by the Chinese Remainder Theorem (CRT) or Mixed Radix Conversion (MRC). To convert RNS to the decimal number system, MRC is strictly performed in the sequential processes. On the other hand, CRT can be executed by parallel processors to make it faster than MRC. Thus, most of the previous researches [1, 7, 11] are based upon CRT when the general moduli.

3.PRIME SIZE FINITE FIELD:GF(p)

The rules for a finite field with a prime number (p) of elements can be satisfied by carrying out the arithmetic modulo-p. If we take any two elements in the range 0 to p - 1, and either add or multiply them, we should take the result modulo-p.

Example 1: Table 1 and 2 shows MODULE-2 addition and multiplication respectively for GF(2) , here p equals 2:-

TABLE 2.3
MODULO-2 ADDITION

+	0	1
0	0	1
1	1	0

TABLE 2.4
MODULO-2 MULTIPLICATION

·	0	1
0	0	0
1	0	1

Example 2: In Tables 3 and 4, the results for GF(3) are shown where p is equal to 3.

Table 5.1 Addition in GF(3)

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Thus in GF(3), the additive inverse of 0 is 0 and 1 is 2 vice versa. The multiplicative inverse can be found by identifying from the table pairs

of elements whose product is 1 . In the case of GF(3), we see that the multiplicative inverse of 1 is 1 and the multiplicative inverse .Another approach can be adopted to finding the multiplicative inverse that will be more generally useful and will lead towards the method for constructing other field sizes. In any prime size field, it should be shown that there is everytime at least one element whose powers constitute all the nonzero elements of the field. This element is said to be primitive. For example, in the field GF(7), the number 3 is primitive powers of 3 just repeat the pattern as 36 = 1 . Note that we can carry out multiplication by adding the powers of 3, thus 6 x 2 = 33 x 32 = 35 = 5. Hence we can find the multiplicative inverse of any element as 3i as 3-i = 36-i. Thus in GF(7) the multiplicative inverse of 6 (33) is 6 (33), 4 (34) is 2 (32), and 5 (35) is 3 (31).

Example 3: show how you can make subtraction and division operatons over GF(7) by constructing addition and multiplication

Sol:- the elements of GF(7) are (0,1,2,3,4,5,6) since p=7. The addition and multiplication over

MODULO-7 ADDITION

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

The addition table shown above is used also for subtraction. If we subtract 6 from 3, we first use the addition table to find the additive inverse of 6, which is 1. Then we add 1 to 3 to get the result [i.e., 3-6=3+(-6)=3+1=4]. For division, we use the multiplication table. To obtain the result as [3÷2=3.(2-1)=3.4=5] we have to find the multiplicative inverse of 2 which is 4 and then we multiply 3 by 4.

4.Montgomery Multiplication

For computing a, b and mod m positive integers a, b and m Montgomery multiplication is used. When there are a large number of multiplications to be done with the same modulus m, and with a small number of multipliers this method reduces the execution time on a computer. But if possible there can be a large enough number to be valuable by speeding up. costly execution time is result of reductions in modulo m which are essential division operations.

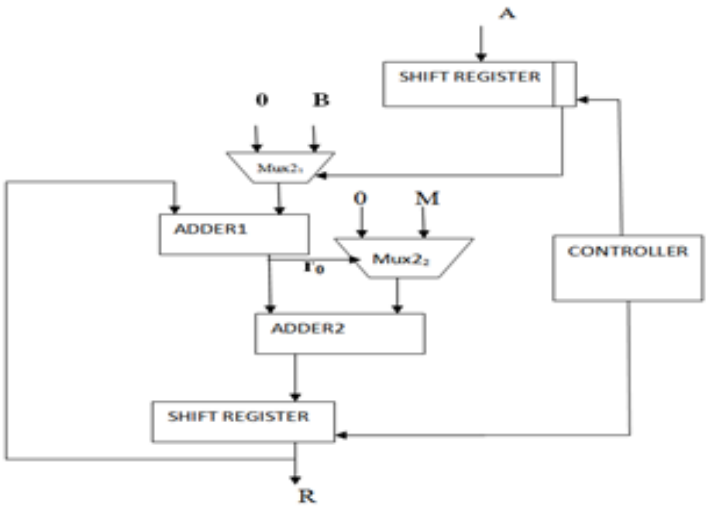


Fig2: Montgomery multiplication

The multipliers a and b should be less than the modulus m for the use of montgomery multiplication. We must have gcd(r, m) = 1and introduces integer which must be greater than m.. This technique, basically changes the reduction modulo r from a reduction modulo m. If r is a power of 2, we must have m odd, to meet the gcd requirement, Usually r is chosen to be an integral power of 2, so the reduction of modulo r is just a masking operation; that is, retaining the lg(r) low-order bits of an intermediate result, and removing higher order bits.

The method:

1. find two integers r^{-1} and m′ such that $rr^{-1} - mm' \square 1$.

This can be done by the prolongedgcd algorithm. There is a binary prolonged gcdalgorithm which does no divisions, and modifysignificantly when one argument (r) is a power of 2 and the other (m) is odd. In arithmetic computation, Montgomery reduction is an algorithm introduced in 1985 by Peter Montgomery that allows modular arithmetic to be performed efficiently when the modulus is large (typically several hundred bits).A single application of the Montgomery algorithm (Montgomery step) is faster than a "naive" modular multiplication

$$c \equiv a \times b \pmod{n}.$$

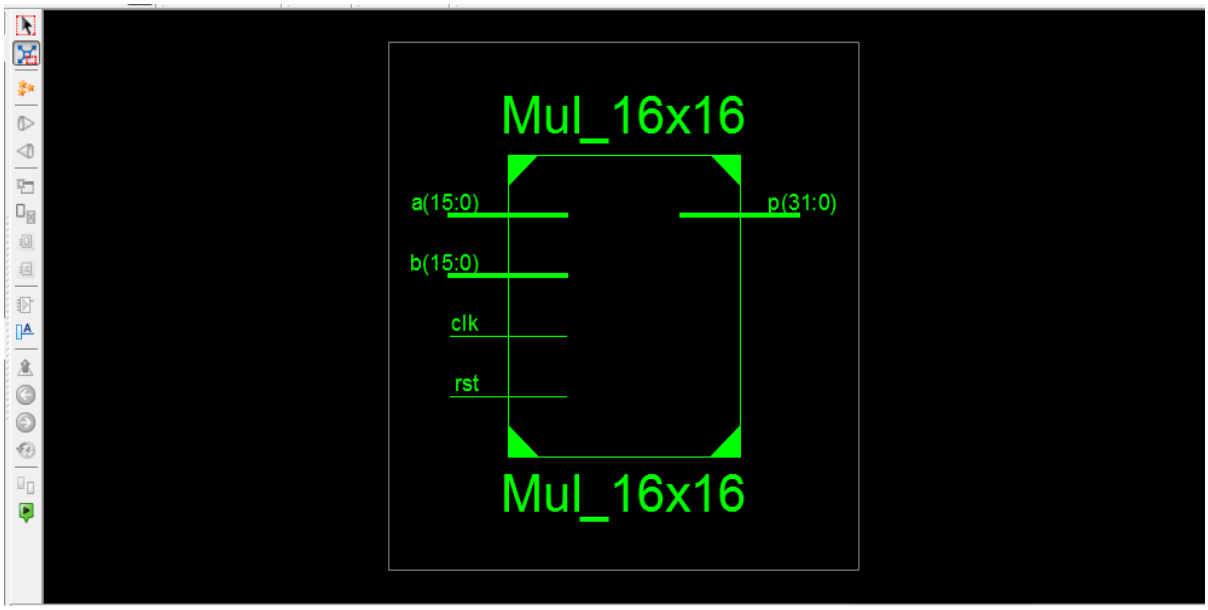
Because numbers have to be converted to and from a particular form suitable for performing the Montgomery step, a single modular multiplication performed using a Montgomery step is actually slightly less efficient than a "naive" one. In this case the greater speed of the Montgomery steps far outweighs the need for the extra conversions.Working with n-digit numbers to base d, a Montgomery step calculates $a \times b \div d^n \pmod{r}$. For the purpose of exposition, we shall illustrate with $d = 10$ and $n = 4$. To turn this into a modular operation with a modulus r, add immediately before each shift, whatever multiple of r is required to make the value in the accumulator a multiple of 10. The result will be that the final value in the accumulator will be an integer (since only multiples of 10 have ever been divided by 10) and equivalent (modulo r) to 472 × a ÷ 10000. Finding the proper multiple of r is a simple operation of single-digit arithmetic. The Montgomery step is faster than the methods of "naive" modular arithmetic because the decision as to what multiple of r to add is taken purely on the basis of the least significant digit of the accumulator. This permit the use of carry-save adders, which are more faster than the conventional kind but are not instantly able to give close values for the more significant digits of the result.

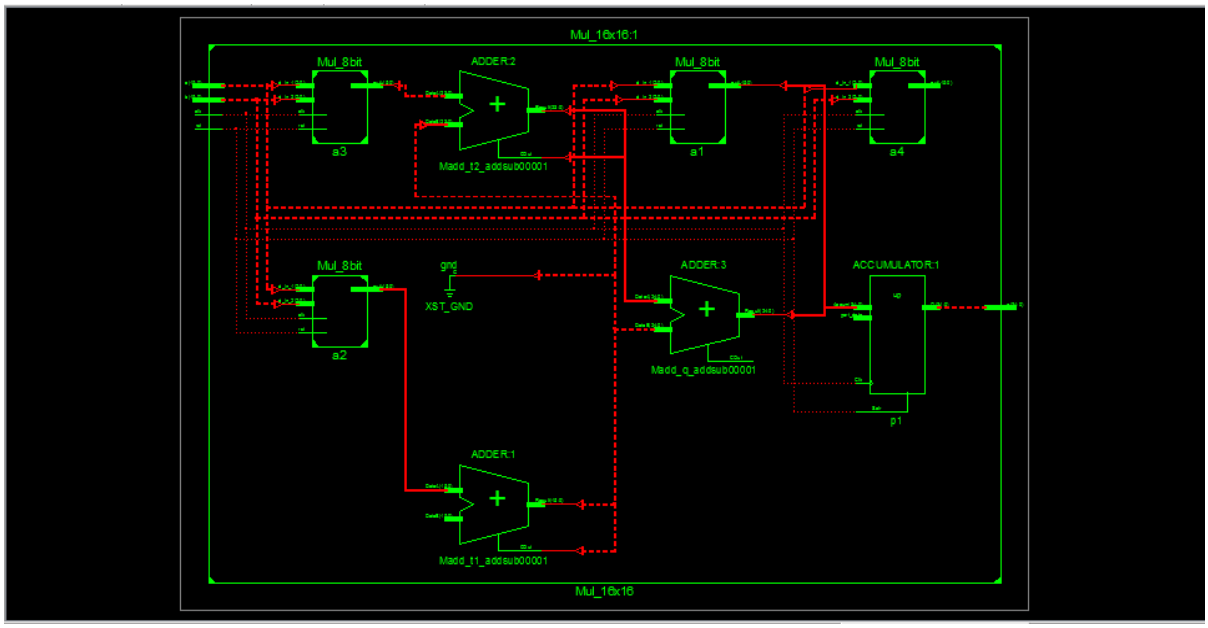
This note tells about the practice and theory of Montgomery multiplication. Montgomery multiplication is a method for computing *ab* mod *m* for positive integers *a*, *b*, and *m*.[1]Itdecreases the execution time on a computer when there are a huge number of multiplications to be done with the same modulus *m*, and with a small number of multipliers. In particular, it is useful for computing a^nmod *m* for a large value of *n*. The number of multiplications modulo *m* insuch a computation can be reduced to a number significantly less than *n* by successively squaring and multiplying according to the pattern of the bits in the binary expression for *n*. But it can still be a large enough number to be worthwhile speeding up if possible. The difficulty is in the reductions modulo *m*, which are, essentially, division operations, which are costly in execution time. If one defers the modulus operation to the end, then the products will grow to very large numbers, which slows down the multiplications and also the final modulus operation.

The multiplier a and b should be less than the modulus m for the use of Montgomery multiplication. We should have gcd(r, m) = 1 and presentadditional integer m which must be lesser than r. The technique, essentially, changes the reduction modulo m to r. If r is a integral power of 2, we must have m odd, to satisfy the gcd requirement, Usually r is chosen to be an integral power of 2, so the decreased modulo r is simply a masking operation; i.e, retaining the lg(r) lower order bits of an midway result, andremoval of higher order bits.

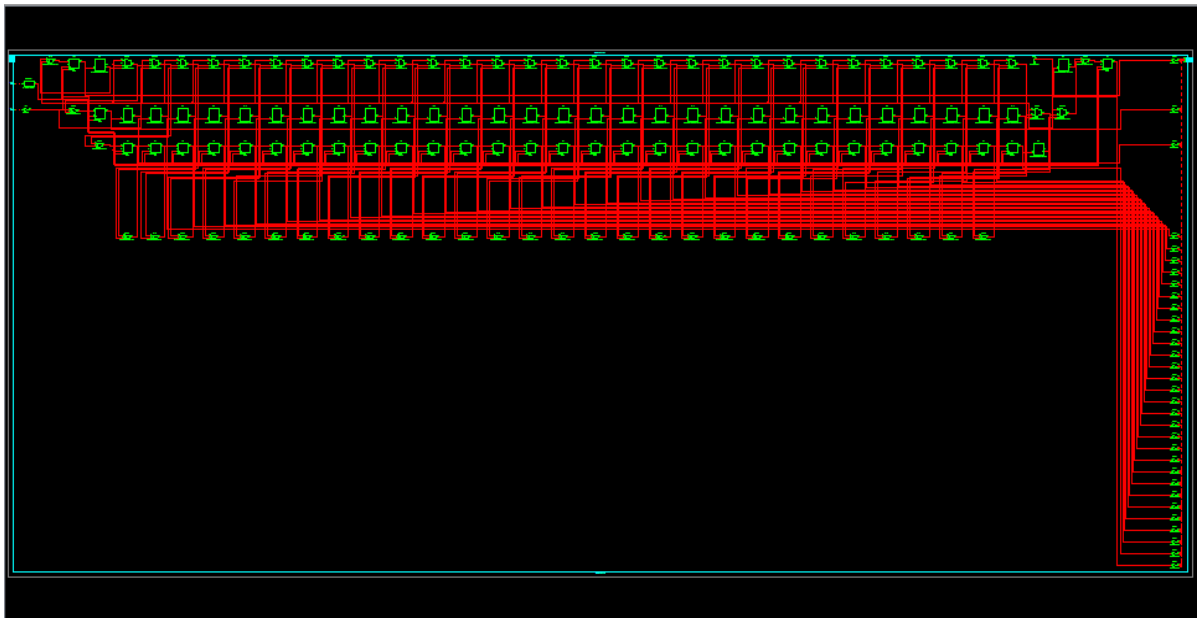
5.RESULT

RTL SCHEMATIC

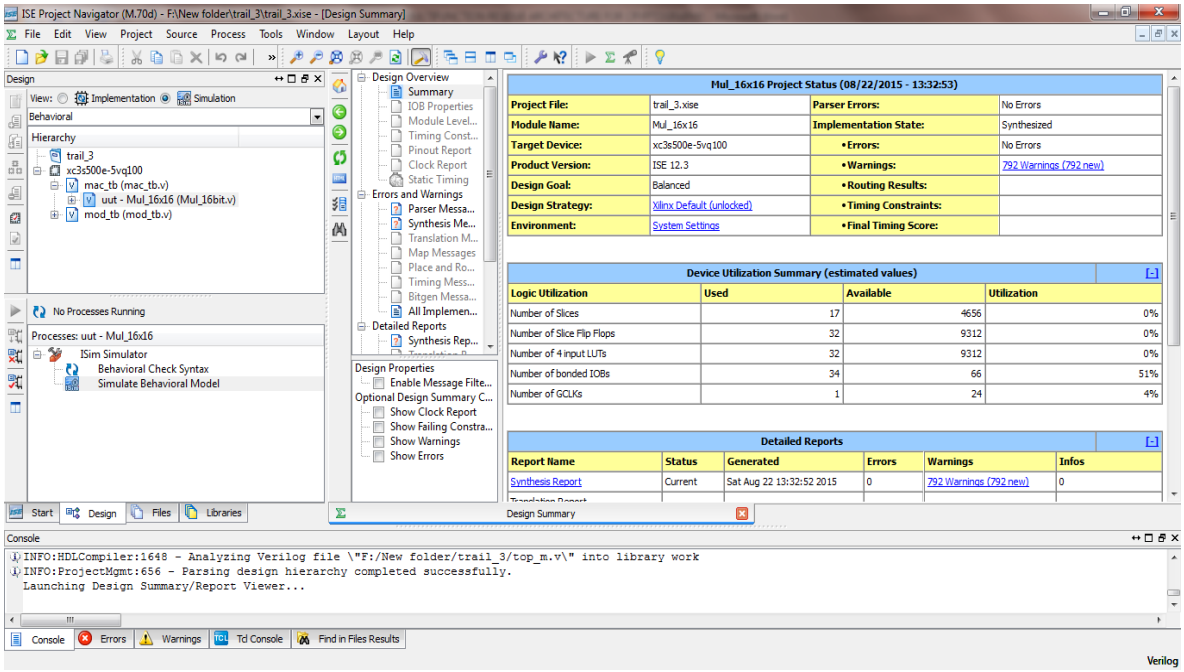




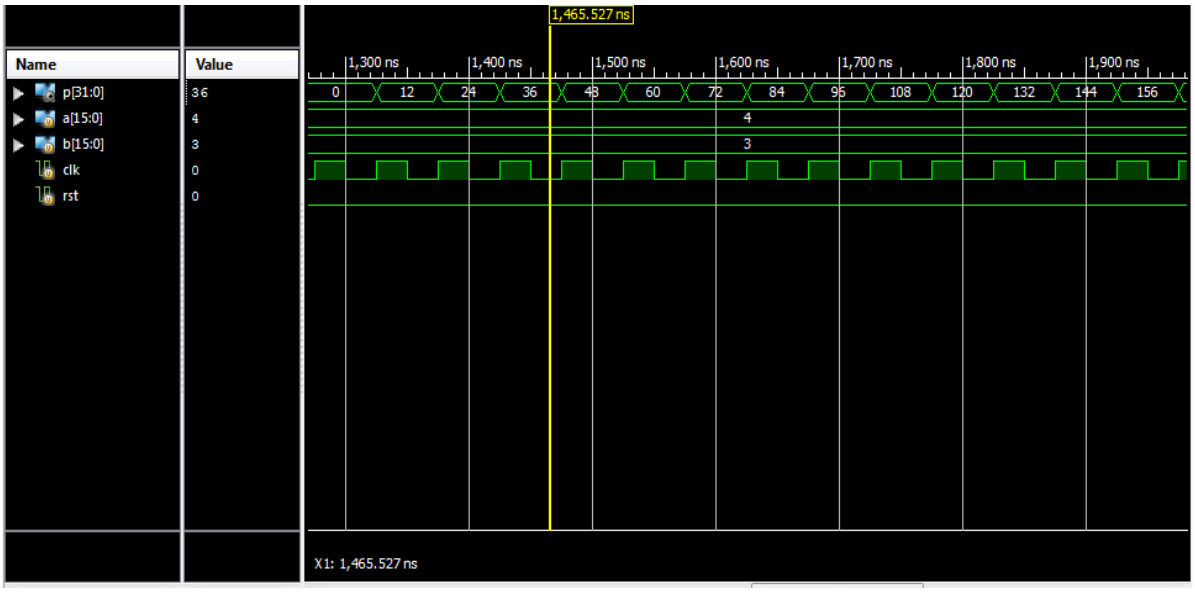
TECHNOLOGICAL SCHEMATIC VIEW



DESIGN SUMMURY



6.SIMULATION



7. Conclusion:

In this paper, for montgomery multiplication we presented a novel on PCA technique and modular reduction scheme with RNS over GF(2^m)based on irreducible pentanomials. To elaborate the efficiency of the plannedmethod, we have intended the multiplier aimed at the irreducible pentanomial $f(x) = x^{13} + x^4 + x^3 + x^2 + 1$, for ease of conversation. In this the Montgomery multiplication is rotten by RNS into two simultaneous blocks and we have resulting a lower-latency multiplier using the recommended modular reductionsystem using PCA. In this method we have lower area-delay and power-delay difficulties than the anewstated multiplier for irreducible pentanomial, with nearly one-fourth of the delay of the other, in terms of the National Institute of Standards and Technology recommended (NIST) pentanomials.

8.References

[1] F. Gandino, F. Lamberti, G. Paravati, J. Bajard, and P. Montuschi, *"An algorithmic and architectural study on Montgomery exponentiation in RNS,"* IEEE Trans. Comput., vol. 61, no. 8, pp. 1071–1083, 2012.

[2] D. Schinianakis and T. Stouraitis, *"Harware-fault attack handling in RNS-based Montgomery multipliers,"* in Proc. IEEE Int. Symp. Circuits and Systems, 2013.

[3] D. Schinianakis, A. Skavantzos, and T. Stouraitis, *"Montgomery multiplication using polynomial residue arithmetic,"* in Proc. IEEE Int. Symp. Circuits and Systems, 2012, pp. 3033–3036

[4] D. Schinianakis and T. Stouraitis, *"A RNS Montgomery multiplication architecture,"* in Proc. IEEE Int. Symp. Circuits and Systems, 2011, pp. 1167–1170.

[5] Y. Tong-jie, D. Zi-bin, Y. Xiao-Hui, and Z. Qian-jin, *"An improved RNS Montgomery modular multiplier*," in Proc. 2010 Int. Conf. Computer Application and System Modeling (ICCASM), 2010, vol. 10, pp. V10-144–V10-147.

[6] N. Guillermin, *"A high speed coprocessor for elliptic curve scalar multiplications over,"* in Cryptographic Hardware and Embedded Systems, CHES 2010, 2010, pp. 48–64, Lecture Notes in Computer Science 6225

[7] M.-D. Shieh and W.-C.Lin, *"Word-based Montgomery modular multiplication algorithm for low-latency scalable architectures,"* IEEE Trans. Comput., vol. 59, no. 8, pp. 1145–1151, aug. 2010.

[8] D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas, and T. Stouraitis, *"An RNS implementation of an elliptic curve point multiplier,"* IEEE Trans. Circuits Syst. I, vol. 56, no. 6, pp. 1202–1213, Jun.2009.