INTERNATIONAL JOURNAL OF SCIENTIFIC PROGRESS AND RESEARCH (IJSPR) ISSN: 2349-4689
Issue 96, Volume 34, Number 02, 2017

Sign Detection using Residue Number System with Lower Delay with Mixed Radix Conversion

Nirbhay Hardaha[1], Dr. Rita Jain[2]
[1]*Mtech. Scholar,* [2]*Research Guide*
Department of Electronics and Communication, LNCT, Bhopal

Abstract- A new advance modified RNS modular fast sign detection algorithm based on the mixed radix conversion (MRC) algorithm and new Chinese reminder theorem (CRT). The RNS is a special, non-weighted, convey free number framework that gives parallel, rapid and fault tolerant arithmetic operations. The sign data is disguised in every residue digit in a RNS, so sign detection in a RNS is more troublesome than that in the weighted number framework, where the sign bit is the most significant bit (MSB). Sign detection assumes a critical part in magnitude examinations, data overflow identification, and in spreading operations.

Keyword- RNS, Moduli set, CRT, MRC, FPGA, Sign detection,

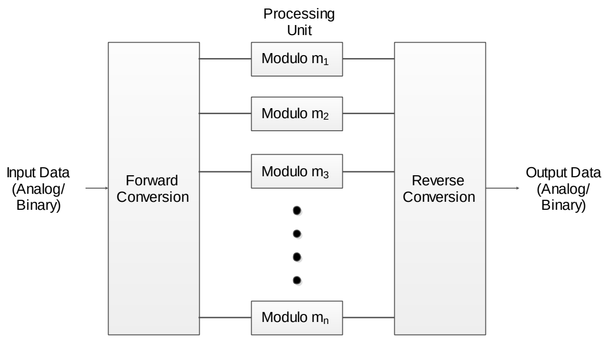
I. INTRODUCTION

Embedded systems today have transformed from simple, single-function control systems to highly complex, multipurpose computing platforms. Higher performance is no longer the only important criterion in such designs. The advent and popularity of personal wireless communication and handheld, portable multimedia and communication devices in the last decade has created stringent requirements on performance, power, cost and time- to-market. The omnipresence of such battery-powered devices has created a perpetual demand for cheap, high performance, and power efficient embedded processors.

The mathematical procedure of obtaining the answer 23 in this example from the set of integers 2, 3, and 2 is what was later called the Chinese Remainder Theorem (CRT). The CRT provides an algorithmic solution of decoding the residue encoded number back into its conventional representation. This theorem is considered the cornerstone in realizing RNSs. Encoding a large number into a group of small numbers results in significant speed up of the overall data processing. This fact encourages the implementation of RNS in some applications where intensive processing is inevitable.

A general structure of a typical RNS processor is shown in Figure 1.1.

The RNS represented data is processed in parallel with no dependence or carry propagation between the processing units. The process of encoding the input data into RNS representation is called Forward Conversion, and the process of converting back the output data from RNS to conventional representation is called Reverse Conversion. The conversion stages are very critical in the evaluation of the performance of the overall RNS.



Conversion circuitry can be very complex and may introduce latency that offsets the speed gained by the RNS processors. For a full RNS based system, the interaction with the analog world requires conversion from analog to residue and vice versa. Usually, this is done in two steps where conversion to binary is an intermediate stage. This makes the conversion stage inefficient due to their increased latency and complexity. To build an RNS processor that can replace the digital processor in a certain application; need to develop conversion circuits that perform as efficient as the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) in the digital binary-based systems.

RNS Representation

An RNS is defined by a set of relatively prime integers called the moduli. The moduli-set is denoted as $\{m_1, m_2,...,m_n\}$ where m_i is the ith modulus. Each integer X can be represented as a set of smaller integers called the residues. The residue-set is denoted as $\{r_1 ,r_2,...,r_n\}$ where is the residue. The residue is defined as the least positive remainder when is divided by the modulus. This relation can be notation ally written based on the congruence:

$$X \, mod \; m_i = \; r_i \qquad \text{................................}(1.1)$$

The same congruence can be written in an alternative notation as:

$$|X|_{m_i} = r_i \qquad \text{......................................}(1.2)$$

The RNS is capable of uniquely representing all integers X that lie in its dynamic range. The dynamic range is determined by the moduli-set $\{m_1, m_2, \ldots, m_n\}$ and denoted as M where:

$$M = \prod_{i=1}^{n} m_i \qquad \text{..............................(1.3)}$$

The RNS provides unique representation for all integers in the range between 0 and M — 1. If the integer X is greater than M — 1, the RNS representation repeats itself. Therefore, more than one integer might have the same residue representation.

Base Extension

Both Jullien (1978) and Shenoy and Kumaresan (1989) describe a method for base extension they call the Szabo-Tanaka method first presented in Szabo and Tanaka (1967).

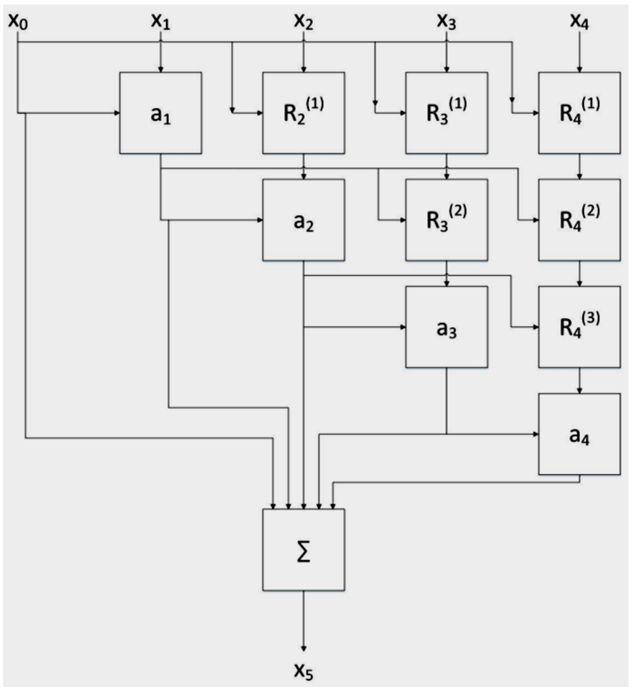


Figure 1.2 : Block diagram of base extension algorithm.

This method uses a recursive way to calculate the residue of the extended base. Take a RNS number X, coded with the moduli $\{m_1, m_2, \ldots, m_n\}$. This number is limited by x < fln= 0 m1. This number is to be base extended to also include the residue of the modulo mn+1. Figure 1.2 Demonstrate the block diagram of base extension algorithm.

II. SYSTEM MODEL

RNS Functional Units

1) Conversion Units

Any RNS-based design involves conversions from binary to RNS, commonly known as forward conversion, and vice versa, known as reverse conversion. The conversion units need to be extremely efficient as conversion operations are an overhead in performance and power in RNS-based systems. Reverse conversion, which is an application of the Chinese Remainder Theorem, is especially expensive. It is therefore natural that a lot of research has addressed not only discovering efficient moduli-set, but also faster and power-efficient reverse converter circuits.

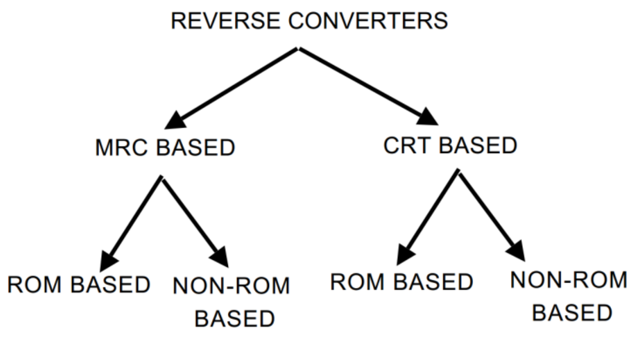


Figure 2.1 Classification of reverse converter.

A classification of the various kinds of techniques is shown in Figure l.12. The first level of classification is made on the basis of what mechanism, whether the mixed radix conversion (Equation (I.1)) or Chinese Remainder theorem (Equation (I.2)), is used to compute the binary number from residues. An additional classification can be made on the actual hardware used in the implementation of the reverse converter. The first reverse converters were based on read-only memories (ROMs) to hold pre-computed moduli used in computation of the terms of Equation (I.1) and Equation (I.2). These converters, though fast, do not scale well for large modulo operations as the size of the ROM is directly proportional to the modulus. Non-ROM based converters, instead, computes all the terms of the equations and are more scalable.

More recent reverse converters have utilized an alternative simpler formulation of the Chinese Remainder Theorem, called the New Chinese Remainder Theorem. This new formulation transforms the original CRT equation, which needs a large modulo operation, to use terms with smaller values of moduli.

2) Computational Units

A large number of adders and multipliers have been proposed over the years for a variety of moduli. These can, in general, be classified into ROM and non-ROM based implementations. A review of some of the early modulo adders and multipliers. More recently, Hiasat has proposed high-speed modular adders suitable for VLSI implemenation. Efstathiou et. al have proposed fast modulo adders based on parallel-prefix carry computation units. Zimmermann proposes circuits for addition and multiplication modulo $2^n - 1$ and $2^n + 1$. Adders are based on parallel-prefix architectures and multipliers are enhanced so that speed-up techniques like Wallace trees and Booth-recoding can be used.

III. PROPOSED METHODOLOGY

A standard RNS is fully particular for positive integer's numbers in the range [0, M). A verifiable signed number framework might be thought to be part into a positive half and negative portion of the range to hold negative integer numbers. The dynamic range M of the moduli set $\{m_1, m_2, \dots, m_{N-1}, m_N = 2n\}$ is even. An implicit symbol of the actual result Y can be obtained in its range$[-M/2, M/2 - 1)$,after conversion from residue number to the weighted number ensuing the no integer X in the interval $[0, M/2)$ as follows:

$$Y = \begin{cases} X\ , \text{if } 0 \leq X < M/2 \\ X - m\ , \text{if } M/2\ < M \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (4.1)$$

The mixed radix CRT obtained of a residue number X=(x_1 ,x_2,.,,,,,,,,,,,x_N)

It converts residue numbers to weighted numbers and it requires only modulo m_i operations. The computation system for each blended radix j is free of others, so the blended radix coefficients can be registered in a completely parallel strategy. Figure 4.1 represent the sign detection unit for Moduli set.

The proposed sign detection system for moduli set has the following major component. The component is as follows for efficient fast sign detection of integer numbers. Demonstrated in figure 4.1 .

(1) A carry generation unit

(2) A carry Correction Unit

(3) Comparator Unit

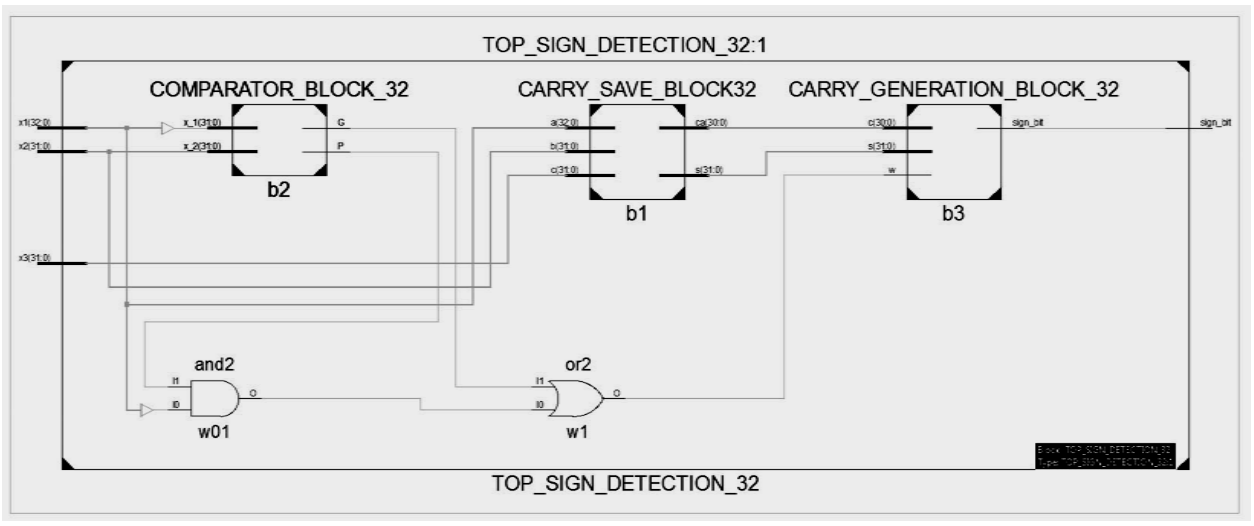


Figure 3.1 Sign Detection unit for the moduli set.

(1) Carry Generation Unit

The carry generation unit which is used for the generation of the carry, the carries CiT (i= 1, 2...n) of A+B+T can be achived with the carry era and carry propagation bits from the pre-preparing unit. Any Existing prefix structure can be utilized to get the conveys CiT. It merits calling attention to that the carry-out piece of SCSA in the preprocessing unit is not included in the prefix Computation. Rather, CSCSA consolidated with the carry-out piece of the prefix tree is required to.

(2) Carry Correction Unit

The carry correction unit is utilized to get the real carries without error for each bit needed in the final sum computation stage. In order to reduce the area and get the carries of A+ B by correcting the carries of in the carry correction unit.

(3) Comparator Unit

Usually, the whole calculation is as same as that in Prefix based binary adder. Be that as it may, grain is the amendment result when is taken into account. That is C out= 0, if, is the Carry bit of $A + B$. Otherwise, it is the carry bit of $A + B + T$. Thus the partial sum bits of A+B and A+B+T are both required in the last entirety calculation. Let PiO and Pil (i = 0,1, 2...n - 1) be the partial sum bits of $A + B$ and $A + B + T$ respectively.

IV. SIMULATION RESULT

The simulation of the proposed system has done on the Xilinx simulation tool the comparison table for the execution speed and sign detection waveform screen captured during the simulation of the proposed system has given in the figure 4.1 to Figure 4.7.

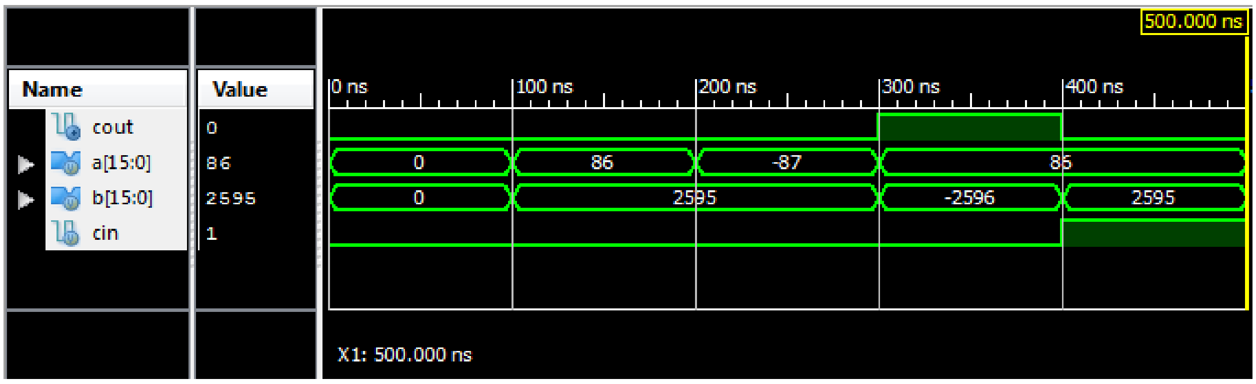


Figure 4.1 Timing waveform of carry generator.

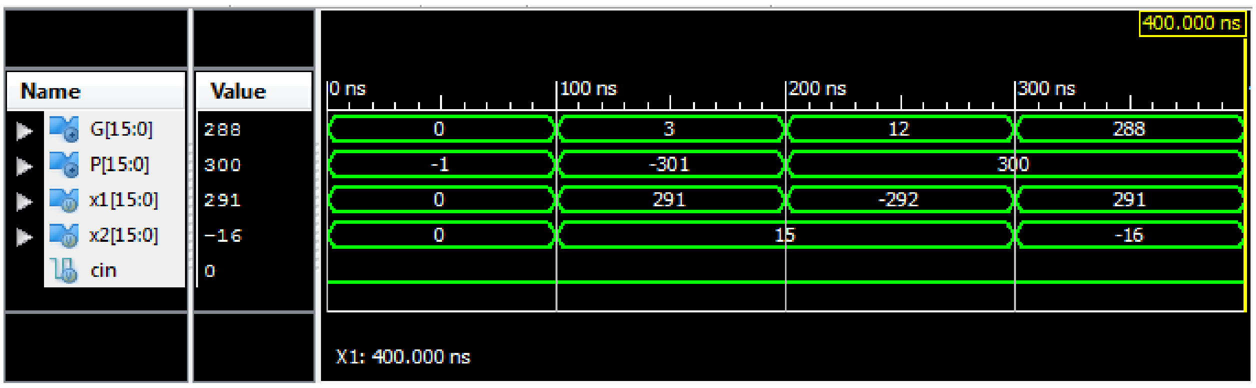


Figure 4.2 Timing waveform comparator.

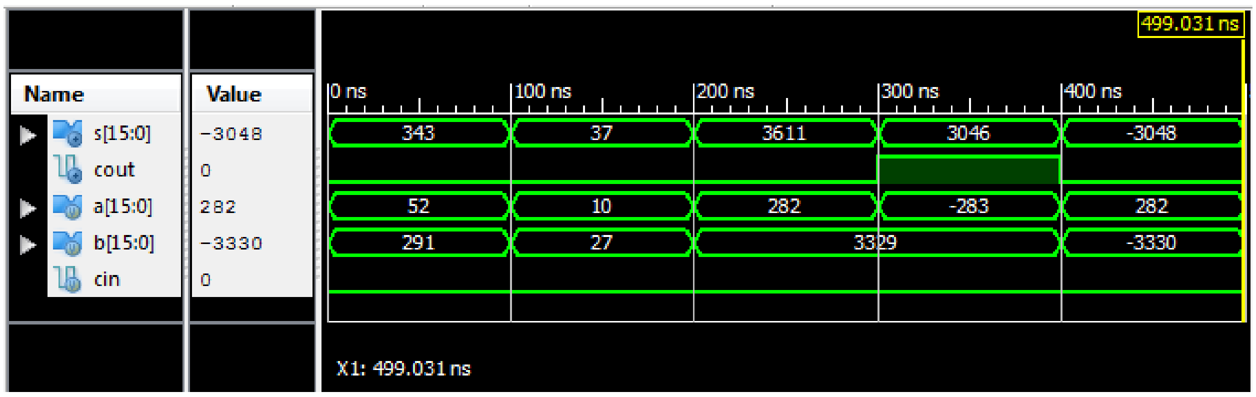


Figure 4.3 Timing waveform of CSA.

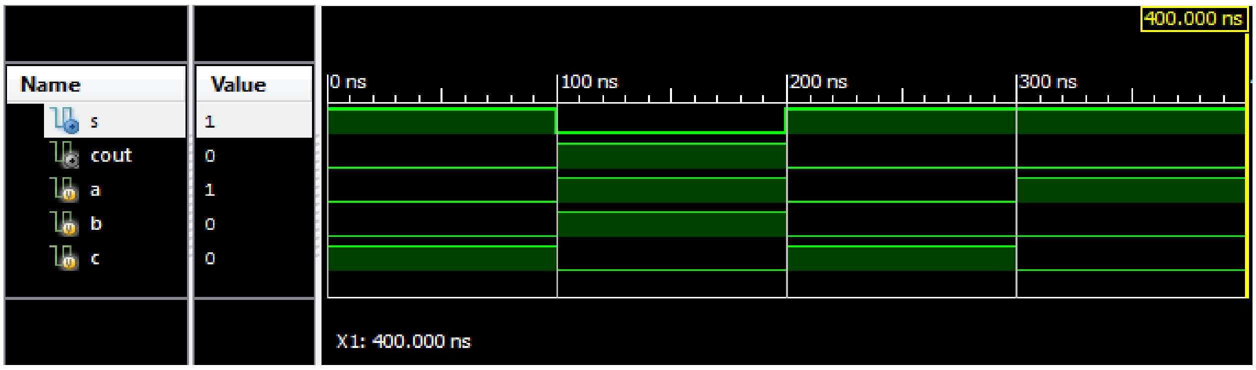


Figure: 4.4 Timing waveform of Full Adder.

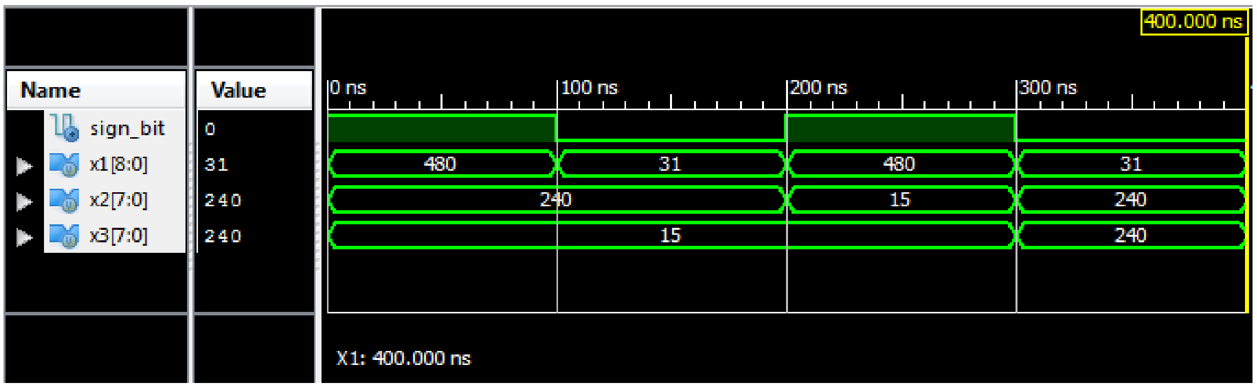
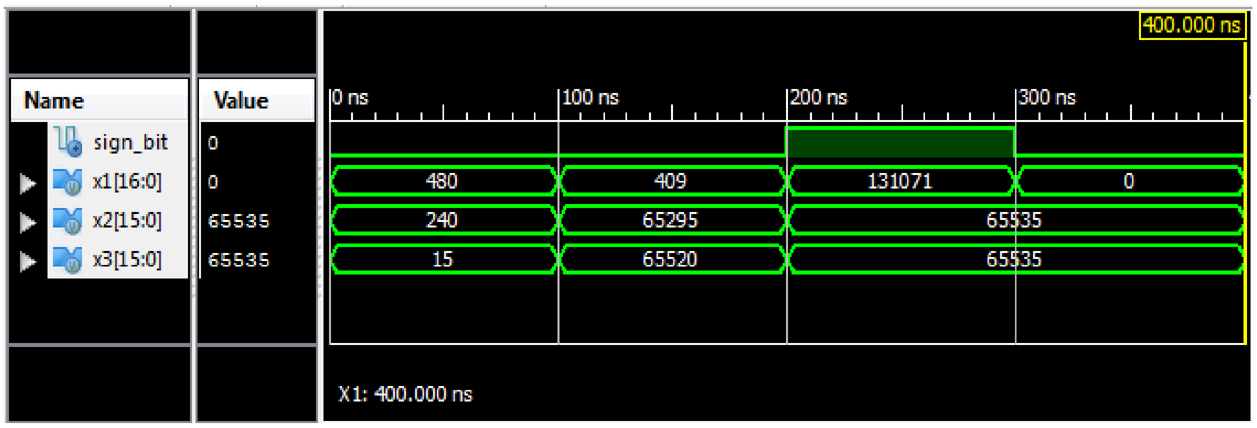


Figure 4.5 Sign Detection waveform of 8 bit.



Figurep:4.6 Sign Detection waveform of 16 bit.

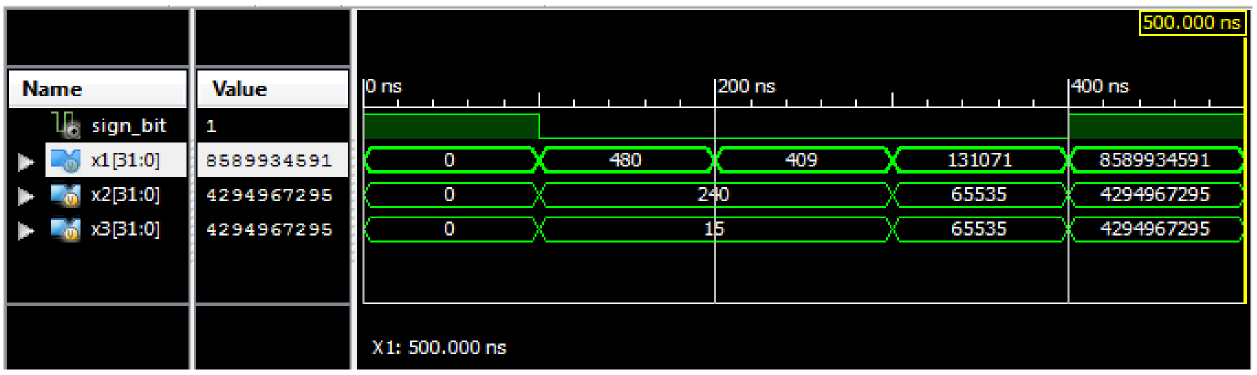


Figure 4.7 Sign Detection waveform of 32 bit.

Table 4.1 Timing Summary.

Cell :in->out	Fan out	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	4	0.003	0.574	x1_0_IBUF (x1_0_IBUF)
LUT5:I0->O	1	0.040	0.349	b3/p1/Y<1>33_SW0 (N01)
LUT5:I3->O	1	0.040	0.467	b3/p1/Y<1>33_SW1 (N16)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>33 (b3/p1/Y<1>33)
LUT6:I5->O	1	0.040	0.467	b3/p1/Y<1>34 (b3/p1/Y<1>34)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>35 (b3/p1/Y<1>35)
LUT6:I5->O	1	0.040	0.467	b3/p1/Y<1>36 (b3/p1/Y<1>36)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>37 (b3/p1/Y<1>37)
LUT6:I5->O	1	0.040	0.467	b3/p1/Y<1>38 (b3/p1/Y<1>38)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>39 (b3/p1/Y<1>39)

LUT6:I5->O	1	0.040	0.467	b3/p1/Y<1>40 (b3/p1/Y<1>40)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>41 (b3/p1/Y<1>41)
LUT6:I5->O	1	0.040	0.467	b3/p1/Y<1>42 (b3/p1/Y<1>42)
LUT6:I2->O	1	0.040	0.292	b3/p1/Y<1>43 (b3/p1/Y<1>43)
LUT4:I3->O	1	0.040	0.292	b3/p1/Y<1>44_SW0 (N2)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>44_SW1 (N14)
LUT6:I1->O	1	0.040	0.434	b3/p1/Y<1>44 (b3/p1/Y<1>44)
LUT6:I3->O	1	0.040	0.292	b3/p1/Y<1>46 (b3/p1/Y<1>46)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>47_SW0 (N4)
LUT6:I1->O	1	0.040	0.560	b3/p1/Y<1>47 (b3/p1/Y<1>47)
LUT6:I1->O	1	0.040	0.000	b3/p1/Y<1>48_G (N19)
MUXF7:I1>O	1	0.196	0.292	b3/p1/Y<1>48 (b3/p1/Y<1>48)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>49_SW0 (N6)
LUT6:I1->O	1	0.040	0.560	b3/p1/Y<1>49 (b3/p1/Y<1>49)
LUT6:I1->O	1	0.040	0.000	b3/p1/Y<1>50_G (N21)
MUXF7:I1>O	1	0.196	0.292	b3/p1/Y<1>50 (b3/p1/Y<1>50)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>51_SW0 (N8)
LUT6:I1->O	1	0.040	0.560	b3/p1/Y<1>51 (b3/p1/Y<1>51)
LUT6:I1->O	1	0.040	0.000	b3/p1/Y<1>52_G (N23)
MUXF7:I1>O	1	0.196	0.292	b3/p1/Y<1>52 (b3/p1/Y<1>52)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>53_SW0 (N10)
LUT6:I1->O	1	0.040	0.349	b3/p1/Y<1>53 (b3/p1/Y<1>53)
LUT6:I4->O	1	0.040	0.292	b3/p1/Y<1>54 (b3/p1/Y<1>54)
LUT5:I4->O	1	0.040	0.560	b3/p1/Y<1>55_SW0 (N12)
LUT6:I1->O	1	0.040	0.349	b3/p1/Y<1>55 (b3/p1/Y<1>)
LUT5:I3->O		0.040	0.279	b3/p1/Mxor_SIGN_BIT_xo<0>1 (sign_bit_OBUF)
OBUF:I->O		0.002		sign_bit_OBUF (sign_bit)

The proposed fast sign detection algorithm shown in this dissertation is synthesis on the XILINX FPGA simulation tool. The synthesis outcome in terms of delay of the proposed design is quite better than the existing work. The comparison is shown in the Table 5.2 below.

The modulo operations in the sign detection cmethodology are limited. a sign detection scheme utilizes the nth blended radix digit in mixed-radix conversion (MRC) to distinguish the sign function. just short to utilize the combinational rationale to execute a sign recognition calculation in light of moduli set. However, the technique can't be reached out to other moduli sets

Table 4.2: Comparison Overall Delay and Unit Gate Delay.

Architecture	Overall Delay	Delay
Previous Design	19 ns	1 ns
Our Proposed Design	15.56 ns	0.196 ns

V. CONCLUSION

In this short a proficient fast sign detection algorithm for the residue number system (RNS) moduli set is presented. The proposed algorithm which permits parallel usage and incorporate modulo 2n augmentations. In view of existing sign detection algorithm, an effective sign detection algorithm is proposed. The sign detection unit can be executed utilizing one carry save adder, one comparator and one prefix adder. Here effectiveness accomplished is superior to other algorithm for sign detection. Table 1 and Table 2 are the comparison table of outcome illustrate the outperformance of the proposed sign detection system of (RNS) Moduliset .

REFERENCES

[1] M. Xu, Z. Bian and R. Yao, "Fast Sign Detection Algorithm for the RNS Moduli Set {2^{n+1}-1, 2^n-1, 2^n} ," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 2, pp. 379-383, Feb. 2015.

[2] C. H. Chang and S. Kumar, "Area-efficient and fast sign detection for four-moduli set RNS {2^n −1,2^n, 2^n +1,22^n +1}," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne VIC, 2014, pp. 1540-1543.

[3] S. R. Kotha, A. Singhvi and S. K. Sahoo, "A new approach for high performance RNS-FIR filter using the moduli set {2^k − 1, 2^k, 2^k−1 − 1}," 2014 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), Penang, 2014, pp. 136-140.

[4] P. Patronik and S. J. Piestrak, "Design of Reverse Converters for the New RNS Moduli Set {2^n+1,2^n-1,2^n,2^n-1+1} (n odd)," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 12, pp. 3436-3449, Dec. 2014.

[5] L. Sousa and P. Martins, "Efficient sign identification engines for integers represented in rns extended 3-moduli set {2n - 1, 2n + k, 2n + 1}," in Electronics Letters, vol. 50, no. 16, pp. 1138-1139, July 31 2014.

[6] T. F. Tay, C. H. Chang and J. Y. S. Low, "Efficient VLSI Implementation of {2^n} Scaling of Signed Integer in RNS {2^n-1, 2^n,2^n+1} ," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 10, pp. 1936-1940, Oct. 2013. [1] T. Tomczak, “Fast sign detection for RNS {2n — 1, 2n, 2n + 1},” IEEE Tra”s. Circuits Syst. I, Reg. Papers, vol. 55, no. 6, pp. 1502-1511, Jul. 2008.

[7] P. Mohan, “RNS-to-binary converter for a new three-moduli set {2n+1 — 1, 2n, 2n — 1},” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 9, pp. 775-779, Sep. 2007.

[8] S. Bi and W. Gross, “The mixed-radix Chinese remainder theorem and its applications to residue comparison,” IEEE Trans. Comput., vol. 57, no. 12, pp. 1624-1632, Dec. 2008.

[9] S. Piestrak, “Design of residue generators and multioperand modular adders using carry-save adders,” IEEE Trans. Comput., vol. 43, no. 1, pp. 68-77, Jan. 1994.

[10] R. Zimmermann, “Efficient VLSI implementation of modulo (2n ± 1) addition and multiplication,” in Proc. 14th IEEE Symp. Comput. Arithmetic, 1999, pp. 158-167.

[11] K. Furuya, “Design methodologies of comparators based on parallel hardware algorithms,” in Proc. 10th ISCIT, Oct. 2010, pp. 591-596.