International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

Academic Science

Role of Residue Number System in High-speed Spanning tree Parallel prefix Adder

1.**B.Raghavaiah** , Research scholar , Department of Electronics and Communication Engineering , Shri JJTU, Rajasthan ,India.
2.**Dr. Om Praksh** ,HOD, Department of ECE, Shri JJTU, Rajasthan, India.

ABSTRACT

The modern computation system is growing rapidly. Every computation depends on the general arithmetic operations that are indirectly involves decimal numbers. The decimal numbers are represented in binary notation. The decimal number system is very flexible to everyone and computations are easier than any other system. Even though there is problem while dealing with this decimal number system. With increase value of decimal number complexity increases. When these higher valued numbers are converted to binary number for computer computations, this leads huge usage of resources.

The solution for this problem is using RESIDUE number system. The RNS converts the huge valued number into group of small numbers, this makes computations easy and uses of less resources, which is the main advantage of the RNS system.

INTRODUCTION

The fundamental idea under- lying RNS is arithmetic applications. This paper involves the conversation of fundamental aspects of the RNS and then with main concepts of residue number system. The practical use of the unconventional number systems in computer arithmetic is described briefly. We have two objectives in these preliminary discussions. Firstly we should differentiate the RNS with the conventional number system. Secondly for forming basic residue number system arithmetic, we need to follow some basic rules of arithmetic. In number system, every series corresponds to precisely one number and in the normal decimal number system that is used to split the integer and fraction parts of a representation. In RNS, a given sequence may match to more than a few numbers: the radix point position is not fixed, and each position in a digit-sequence indicates the particular number represented.

Residue number systems and arithmetic

Congruence is the basic thing on which RNS is based on, which is defined as follows. Let us consider two a and b, if the difference between a and b is exactly same to result of division of a and b with m, then we can say that a and b are congruent module of m; to write a =b (mod m) to denote this. Thus, for example,10 =7 (mod 3), 10 =4 (mod 3), 10 =1 (mod 3), and 10 = -2 (mod 3). The m is the base or the modulus and it is assumed to leave out unity, this produces slight congruencies.

Choice of moduli

More “practical", issues, the best moduli are probably prime numbers at least from a purely mathematical perspective. A particularly useful property of such moduli is that of “generation". If a modulus, m, is prime, then there is at least one primitive root (or generator), p < m -1, such that the set $\{|p^i|_m: i= 0,1,2,3\ldots\ldots m-2\}$ is the set of all the non-zero residues with respect to m. As an example, if m = 7, then we may take p = 3, since $\{|3^0|_7 = 1, |3^1|_7 =3, |3^2|_7 =2, |3^3|_7=6, |3^4|_7 =4, |3^{5-}|_7 =5\} = \{1,2,3,4,5,6\}$; 5 is also a primitive root. Evidently, for such moduli, multiplication and powering of residues may be carried out in terms of simple operations on indices of the power of the primitive root, in a manner similar to the use of logarithms and anti-logarithms in ordinary multiplication. More on the subject will be found in Chapters 2 and 5.

It is also useful to have moduli that simplify the implementation arithmetic operations. This always means, arithmetic on residue digits should not deviate too far from conventional arithmetic, which is just arithmetic modulo a power of two. A common choice of prime modulus that does not complicate arithmetic and which has good

International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

representational efficiency is $m_i = 2^i - 1$. Not all pairs of numbers of the form $2^i - 1$ are relatively prime, but it can be shown that that $2^j - 1$ and $2^k - 1$ are relatively prime if and only if j and k are relatively prime. Many moduli sets are depends on these choices, but there are other possibilities; for example, moduli-sets of the form $\{2^n - 1, 2^n, 2^n + 1\}$ are among the most popular in use.

Conversion

The most direct way to convert from a conventional representation to a residue one, a process known as forward conversion, is to divide by each of the given moduli and then collect the remainders. This, however, is likely to be a costly operation if the number is represented in an arbitrary radix and the moduli are arbitrary. On the other hand, the number is represented in radix-2 (or a radix that is a power of two) and the moduli are of a suitable form (e.g. 2^n-1), then there procedures that can be implemented with more efficiency. The conversion from residue notation to a conventional notation, a process known as reverse conversion, is more difficult (conceptually, if not necessarily in the implementation) and so far has been one of the major impediments to the adoption use of RNS. The use of the Chinese Remainder Theorem, which is the basis for many algorithms for conversion from residue to conventional notation; this too involves, in essence, the extraction of a mixed-radix representation.

Forward conversion

Numbers that are initial inputs to, or final outputs from, residue computations will usually be in some conventional notation. Forward conversion is the process of translating from conventional notation, here binary or decimal, into residue notation. The basic principle in the computation of residues is division, with the moduli as the divisors. But division is an expensive operation in hardware and so is rarely used in the computation of residues, whence the significance of the special moduli. Division is avoidable in the case of other moduli as well, but the hardware required will not be as simple as in the case of the special moduli.

$\{2^{n-1}, 2^n, 2^{n+1}\}$ moduli-sets

We have seen above how to obtain the residues relative to each of the moduli 2^{n+1} and 2^{n-1}. The only other modulus in the basic special set is 2n. Residues with respect to this modulus are obtained easily by dividing the given binary number by 2n, which "division" is just an n-bit right-shift of the given binary number, X. So forward conversion in the $\{2^{n-1}, 2n, 2^{n+1}\}$ moduli-set is straightforward and simple logic circuits, involving modular adders, will suffice for the implementation. If we define, $m1 = 2n + 1$, $m2 = 2^n$ and $m3 = 2^n-1$, then any integer X within the

dynamic range, $M = [0, 2^{3n} - 2^n - 1]$ (where the upper end of the range is m3m2m1), is uniquely defined by a residue-set {r1, r2, r3}, where $r_i = |X|_{mi}$ and X is a 3n-bit:

$X = x_{3n-1}x_{3n-2}\ldots..x_{2n}x_{2n-1}\ldots\ldots x_nx_{n-1}\ldots\ldots x_0$

example

Consider the moduli-set f7; 8; 9g, and let X = 167. The binary representation of X is 10100111. Since n = 3 in the given moduli-set, we partition X into 3-bit blocks, starting from the right: B1 =010 B2 = 100 B3 =111. Applying Equation 3.4, we get (in decimal)

$|167|_{2^3+1} = |167|_9$

$= |2 -4 + 7|_9$

$= 5$

The residue with respect to 8, which 23; is obtained by shifting the binary equivalent of 167 three bits to the right and taking the three bits shifted out: 7 in decimal.

Reverse conversion

Reverse conversion is the process, usually after some residue-arithmetic operations, of translating from residue representations back to conventional notations. It is one of the more difficult RNS operations and has been a major, if not the major, limiting factor to a wider use of residue number systems. The main methods for reverse conversion are based on the Chinese Remainder Theorem (CRT) and the Mixed-Radix Conversion (MRC) technique. The basic theoretical foundations of the main methods, the chapter also includes some discussions on architectural implementations.

Consider the moduli-set {3, 5, 7}, and suppose we wish to find the X whose residue representation is (1, 2,3). To do so, we first determine the M_is and their inverses:

International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

$M1 = {}^{M}/_{m1}$

$= \frac{3 \times 5 \times}{3}$

$= 35$

$M2 = {}^{M}/_{m2}$

$= \frac{3 \times 5 \times}{5}$

$= 21$

$M3 = {}^{M}/_{m3}$

$= \frac{3 \times 5 \times}{7}$

$= 15$

Whence

$|M_1M_i^{-1}|_3=1$

$|35M_1^{-1}| =1$

$M_1^{-1}= 2$

Similarly

$|M_2M_2^{-1}=2$

$|21M_2^{-1}|5 = 1$

$M^{-1}{}_2 = 1$

and

$|M_3M_3{}^{-1}|_7 = 1$

$|15M_3{}^{-1}|_7 = 1$

$M_3^{-1} = 1$

Then, by the CRT, we have (M = 3 £ 5 £ 7 = 105)

$X = || \sum_{i=1}^{3} x_i \quad |_{150}$

$=|1 \text{ X } 35 \text{ X } 2 + 2 \text{ X } 21 \text{ X } 1 + 3 \text{ X } 15 \text{ X } 1|_{105}$

$=||112|_{105} + 45|_{105}$

$=|7 + 45|105$

$= 52$

Filter design

Introduction

Filters are known to be frequency selective network. A selected frequency bands are allowed attenuating the others. These filters are mainly classified into two different types as the digital and analog filters. Further classification is done as FIR filters and FIR filters.

Digital Filters

In electronic industry digital filters are widely used because of their high potentiality to obtain far better signal to noise ratio when compared to the analog filters and also the intermediate stage noise is more in analog filters which is almost noiseless in digital filters. For noise cancelation, shaping spectrum and for reducing inter symbol interference in communication. By achieving high performances these digital filters have became very popular then the analog filters.

Digital Filters are constructed from 3 fundamental mathematical operations.

- Addition (or subtraction)
- Multiplication (normally of a signal by a constant)
- Time Delay i.e: delaying a digital signal by one or more sample periods the above mentioned mathematical equations represent behavior of the system and graphical representation is as shown in the figure 1.2

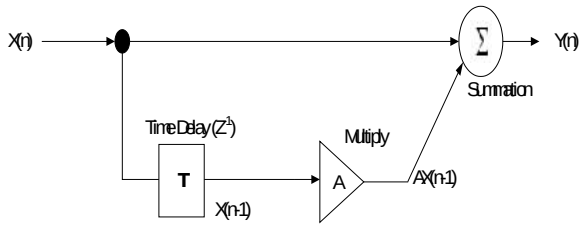


Figure .a simple design of digital filter

h(n) is the output response of a filter which is having *x(n) as input and δ*(*n*) as the unit impulse function. The system response can be calculated by knowing input signal and the impulse response of the system for the sequence *x(n).*By definition, the unit impulse is applied to a system at sample index *n*=0. So, the impulse response is non-zero only for values of *n* greater than or equal to zero i.e *h(n)* is zero for *n*<0. This impulse response is said to be causal otherwise the system would be producing a response before an input applied. The LTI system, time inverse property states that the response of the system will be *h*(*n*-*k*) for the delayed unit impulse response *δ*(*n*-*k*). linear property states that waited sum of the inputs will be equal to the weighted sum of the response of the individual inputs. Therefore, the response of a system to an arbitrary input *x*(*n*) can be written as follows:

$Y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$

Multipliers are the very essential blocks in the digital filter and also in the digital signal processors.

International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

The performance of multiplier play vital role. The whole process depends on the speed of the multipliers. The conventional multiplication system was implemented using sequence of additions and shifting. A series of repeated addition is considered as the multiplication. Multiplicand is the number to be added and the multiplier is the number that indicates how many number of times of addition to yield the final product. Generally the product is twice the width of the input. The shifting and adding process is considered to be the basic process time taking. The necessity of multiplier implementation brought an idea of efficient multiplication algorithm, which divides the multiplication process into two blocks as the partial product generation and the addition.

The basic multiplication principle is twofold, i.e. production of partial products and addition of shifted partial products. This is done by iterative addition of shifted partial product matrix. Shifting of the 'multiplier' is done and gated by the suitable bit of 'multiplicand'. Thus produced partial product matrix is then added to produce final product bits. Two's complement number system is used for performing multiplication for both signed and unsigned multiplications.

The existing multiplier design overwhelmed with complicated switching AND/OR irregular logic design. The 2^n multipliers that are operating in parallel instead of pipelining avoid most of the issues. The pipelining multipliers are M. K. Ibrahim in 1993. Digit level pipelining provides the benefit of constant speed that is not dependent on size of the multiplier. The digit size of the multiplier decides the clock speed of the multiplier which is fixed before implementing the design

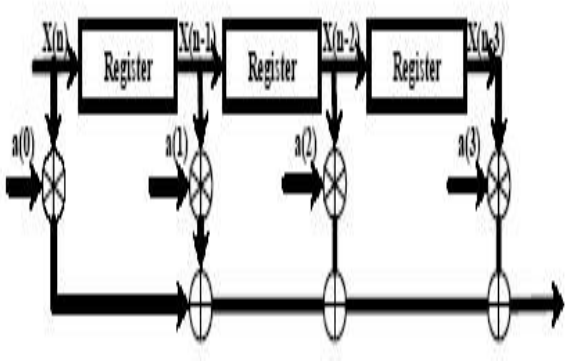


Figure FIR FILTER

For performing addition, a digital circuit adder/ summer is used. In some processors adders also used for calculating addresses, table indices and similar operations. Adders can operate in any radix, in general for computers they are uses binary numbersFor representing negative numbers two's compliment and one's compliments are used

Parallel prefix adders:

The parallel prefix adders are similar to that of carry look ahead adder. Based on the requirement he parallel prefix adders generate in different ways. For increasing the speed of operation, we use tree structures. The fastest and best performing adders in industry are the parallel prefix adders. Addition in parallel prefix adders involves three steps as

1. Pre-processing stage
2. Carry generation network
3. Post processing stage

The f3 stages of parallel prefix addition i.e. pre-calculation of **pi, gi,** calculating carries **ci** and finally combining **ci** and **bi** for producing the final sum **si** is done by a 3-stage butterfly like structure as shown.

1. Pre-processing stage

The generate and propagate signals are produces in this stage for each adder taking inputs as A and B. These signals are given by the equation 1&2.

Pi=Ai Bi..... (1)

GI=Ai .Bi..... (2)

2. Carry generation stage

Carry generation is the most dominant operation. This stage is mainly for generating carries for each bit. The carries are generated in parallel and divided into smaller pieces. Carries are generated using AND gates and an OR gate. It uses propagate and generate as intermediate signals which are given by the equations 3&4.

P(i:k) =P(i:j) . P(j-1:k)…….....(3)

G(i:k) =G(i:j) +(G(j-1:k) . P(i:j))……..(4)

3. Post processing stage

This is the final stage to compute the summation of input bits. it is same for all adders and sum bit equation given

International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

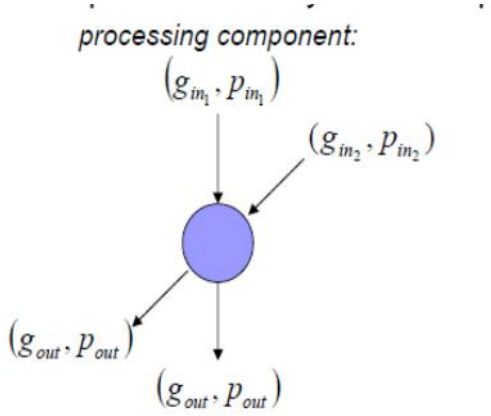
Si=Pi.Ci……..(5)

Ci+1=(Pi.C0)+Gi……(6)

Parallel prefix operations

Calculation of carries –Prefix Graphs

The components usually seen in a prefix graph are the following with the processing component.



The equation will be calculated with **gin** and **pin** values is

$$(g_{out}, p_{out}) = (g_{in_1} + p_{in_1} \cdot g_{in_2}, p_{in_1} \cdot p_{in_2})$$

Carry tree adders is the other name for Parallel-prefix adders, propagate and generate signals are pre-compute. *Fundamental carry operator* (fco) is used for combining generate and propagate signals.

(gL, pL) o $(gR, pR) = (gL + pL \bullet gR, pL \bullet pR)$

These operators can be combined in any form since *fco* accepts associative property. For, example the four-bit carry-look ahead generator is given by:

$c4 = (g4, p4)$ o $[\ (g3, p3)$ o $[(g2, p2)$ o $(g1, p1)]\ ]$

A simple rearrangement of the order of operations allows parallel operation, resulting in a more efficient tree structure for this four bit example:

$c4 = [(g4, p4)$ o $(g3, p3)]$ o $[(g2, p2\ )$ o $(g1, p1)]$

Spanning Tree:

For implementing tree graph we many algorithms, vertex-centric algorithm is one of them.

1. Select a node, say as arbitrary node and make it as the beginning of the tree.

2. Repeat this till all the nodes are marked.

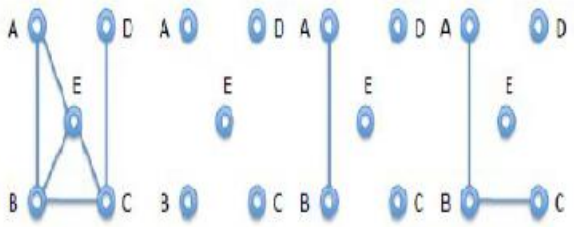
Select the arbitrary node with an edge to a node W in the tree. Mark w as in the tree and add it the spanning tree. Repeat this for n-1 times since we have n-1 vertices which are to be added to the tree. Efficiency of the algorithm is determined by the qualifying selection of the W.

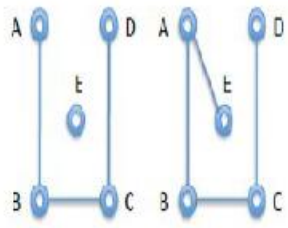
The second algorithm is edge-centric.

1. Collect the singleton trees, each with exactly one node.

2. If we have more than one tree, in the edge connect two trees.

Since we are having n-1 edges, the second step repeats for ‘n’ times Its efficiency is determined by how quickly we can tell if an edge would connect two trees or would connect two nodes already in the same tree. Let’s try this algorithm on our first graph, considering edges in the listed order: (AB, BC, CD, AE, BE, CE).





1. Spanning tree adder:

Use minimum number of multi-input gates

Hybrid adder

2. Manchester Carry Chain (MCC):

use ripple carries

Propagate rapidly

Size of MCC

is a trade of of inputs of levels.

International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

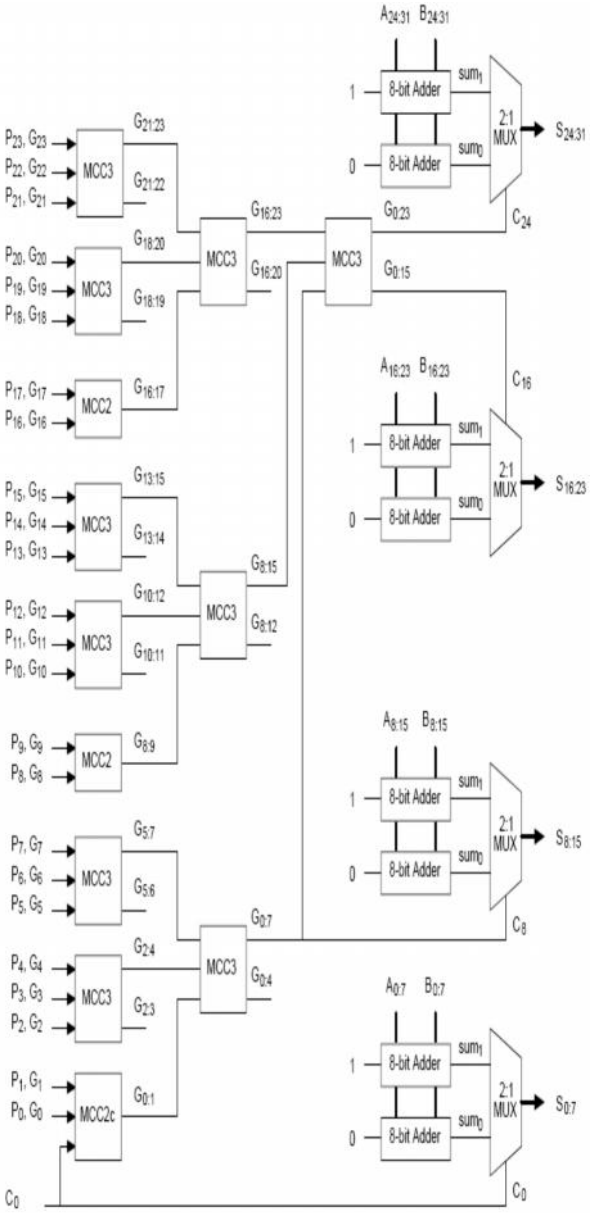


Fig: 32-bit spanning tree adder

pi = Ai XOR Bi

gi = Ai AND Bi

This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic equations below:

Pi:j = Pi:k+1 AND Pk:j

Gi:j = Gi:k+1 OR (Pi:k+1 AND Gk:j)

Present work:

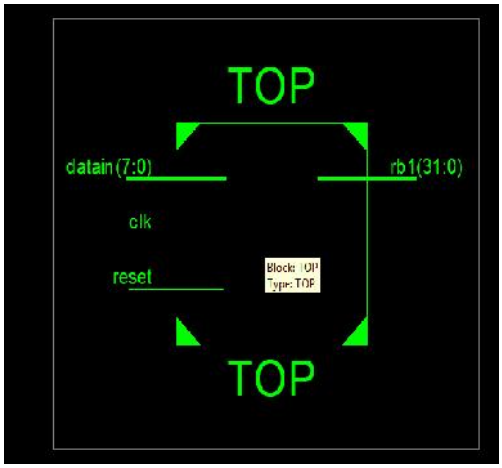
The paper proposes RNS system along with FIR filters and spanning tree adder.

The process involves converting a binary or decimal number to residue number and the giving it to the FIR filter, as we know filter consists of multipliers and adders. The input given to filter flows through several stages of multipliers and adders and produces the output at the final adder. The adder we used here is the efficient spanning tree adder and finally outputs from the filter are converted back to the original radix either binary or the decimal.

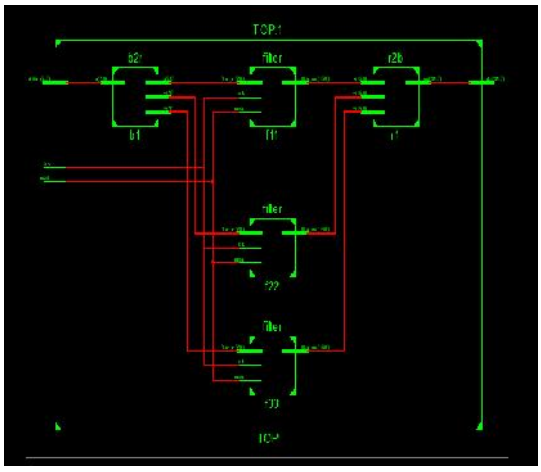
Firstly the non residue number is given to the forward converter with three prime numbers which yields 3 outputs as the modulo division outputs, the three outputs are given to the three FIR filters, the multiply-add process took place in the filters and produces three outputs from three filters. Thus obtained inputs are again reverse converted to their original base using reverse conversion method.

Results

RTL schematic



Internal of RTL schematic

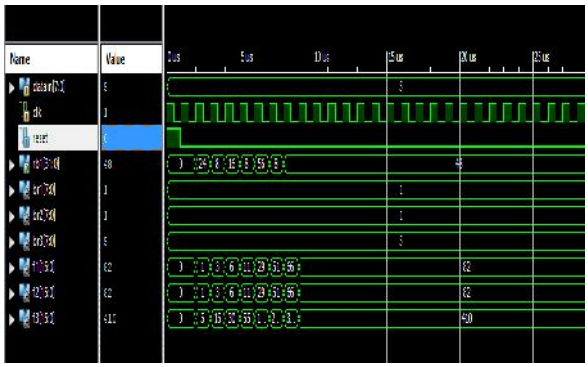


International Journal of Electronics, Electrical and Computational System
IJEECS
ISSN 2348-117X
Volume 6, Issue 7
July 2017

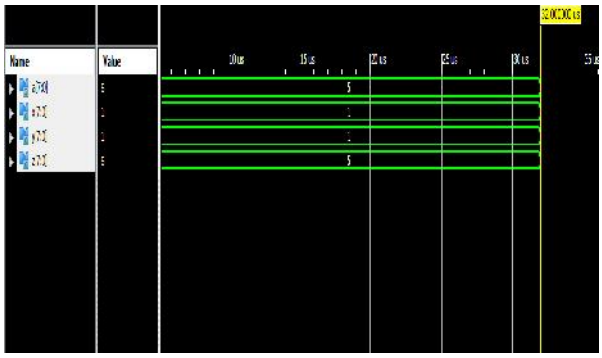
Academic Science

Technology schematic

Top module

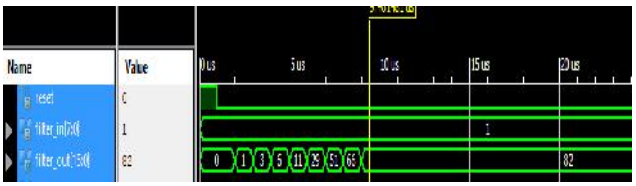


Residue to binary



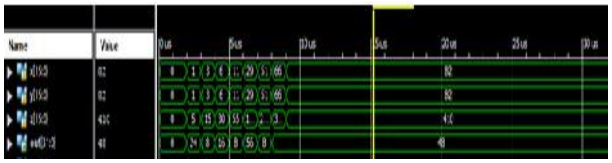
Filter 1

Filte 2



Filter 3 output

Binary to radix



Design results

Sparton3E-xc3s500e-fg320

	No. of LUT's	Power (watts)	Delay(nano sce)
Spanning tree adder	19	1.444	9.569
Carry select adder	26	1.976	9.617

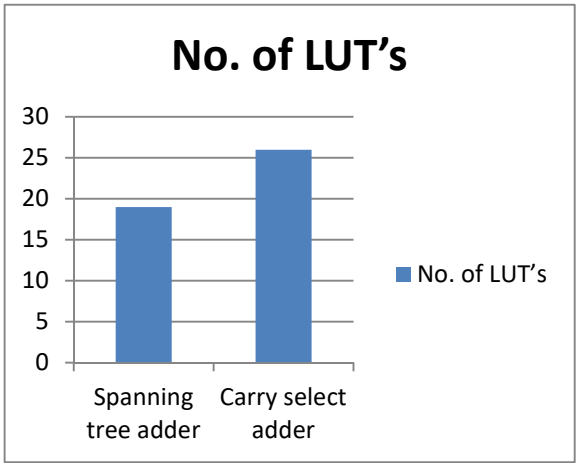


Fig. Graph showing no. of LUT', required for spanning tree and carry select adder.

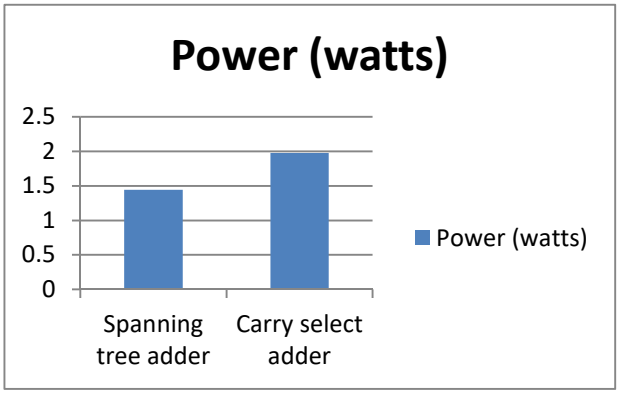


Fig. Graph showing power consumed by spanning tree and carry select adder.

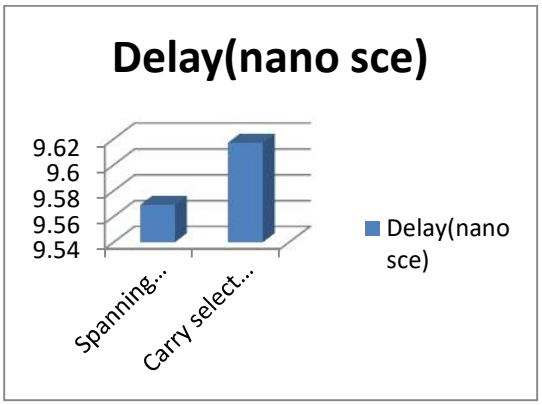


Fig. Graph showing delay produced by spanning tree and carry select adder.

Conclusion:

The paper proposes residue number system, instead of decimal number system and shown that performs arithmetic operations as similar to the decimal number system with similar results and lesser hardware. In the reverse conversion of RNS, we need to use additions and multiplications. The addition circuit plays major role in the reverse conversion system. The paper uses carry select adder and spanning tree adder. The spanning tree adder gets better results when compared to the carry select adder. The results are as shown in table.

Carry select adder implementation uses more hardware then the spanning tree adder. Also delay and power consumption are less in spanning tree approach.

References

1. P.V.Mohan,"novel design for binary to RNS converters " in Pro.IEEE Int.Symp. Circuit system., India,june 1994,pp.357-360.
2. Molahosseini A.S and Sorouri.s.,"Research challenges in next generation residue number system architures" in Proc.IEEE Int. conf.comput.Sci.Educ.vol.69,No.7,pp.1658-1661.
3. B.Ram kumar and H.m Kittur, " low power and area efficient carry select adder". IEEE trans very large scale integer.(vlsi) sysst. Vol.20, no.2, 371-375 feb 2012.
4. B.Chandrika., and G.Poorna Krishna "implementation and estimation of delay power and are of parallel prefix adders ".International Journal for Modern Trends in Science and Technology Volume: 02, Issue No: 11, November 2016
5. Chan.hua vun. And A.B.Premkumar." a new RNS based DA approach for Inner product computation". IEEE. Trans. Circuit and systems, vol.60, no.8, august 2013