ENHANCED MAC DESIGN USING RNS USING KOGGE-STONE

Boddapalli Venkata Ramana[1], Girish Pechetti[2], V S M Madulika Sundaraneedi[3]
[1,2,3]Department of Electronics and Communication Engineering,
Bonam Venkata Chalamayya Institute of Technology & Science,
Amalapuram, Andhra Pradesh, India.

ABSTRACT: This paper presents the design and implementation of 32- bit floating point RNS Multiply and Accumulate (MAC) unit using Vedic Multiplier. MAC is one of the most important components in many of DSP applications like convolution and filtering. RNS (Residue Number System) gained popularity in implementation of fast arithmetic and fault tolerant computing applications because of parallel processing and carry free computations. Floating point RNS arithmetic units have obvious advantage over fixed point MAC units which are key units in Digital Signal Processors. Floating point RNS MAC uses modulo adder for exponent addition and modulo multiplier for mantissa multiplication where operations are performed on moduli. In previous methods, modulo multiplier is implemented using array multiplier that has more delay. Floating Point can be represented as M x BE where M is Mantissa, E is the Exponent and B is the Base. The MAC unit consists of three units -Floating-point multiplier, Conversion unit and an Accumulator. The floating-point multiplier makes use of Brickell's Algorithm, the conversion unit makes use of a parallel conversion for the forward conversion and the Chinese Remainder Theorem for reverse conversion and the accumulator includes an adder unit which can make use of any of the conventional adders that depends on the moduli of the RNS being used. Further, this project is enhanced by using Koggstone adder algorithm to improve latency failures.

INTRODUCTION:

MAC is the main component in many of the digital signal processing applications like convolution, filters and FFT. A basic MAC architecture consists of a multiplier and accumulator. D.L.Chaitanya Associate Professor, Department of ECE, Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, India. A.Chandana M.Tech, VLSI, Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad, India. The products generated by the multiplier are added and stored in accumulator. The performance of MAC can be increased by the optimized design of multiplier and adder. RNS gained popularity because of the parallel processing and carry-free arithmetic. A large bit number can be represented in form of small bit residues and the residues can be processed in parallel and thus the performance of the multiplier or adder can be increased [3]. That is because there is no need of communicating carry information between two residues. So carry free arithmetic can be performed. A high speed MAC capable for handling large range numbers with better precision will be required for many of the DSP applications. There are two types of arithmetic operation which are fixed and floating point operations. Fixed point number was inefficient for large number arithmetic. So floating point arithmetic was invented. Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular algorithms. To achieve speed improvements Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Further by combining both Modified Booth algorithm and Wallace Tree technique we can see advantage of both algorithms in one multiplier. However with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand "serial-parallel" multipliers compromise speed to achieve better performance for area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics. A residue numeral system (RNS) represents a large integer using a set of smaller integers, so that computation may be performed more efficiently. It relies on the Chinese remainder theorem of modular arithmetic for its operation, a mathematical idea from Sunzi Suanjing (Master Sun's Arithmetic Manual) in the 4th century AD. I N THIS PAPER we develop and investigate the properties of a novel system, called the residue code or residue number system. The residue number system is of particular interest because the arithmetic operations of addition and multiplication may be executed in the same time as required for an addition operation. The main difficulty of the residue code relative to arithmetic operations is the determination of the relative magnitude of two numbers expressed in the residue code. The residue code is probably of little utility for general-purpose computation, but the code has many characteristics which recommend its use for special-purpose computations. The residue code is most easily developed in terms of linear congruence. A brief discussion of the pertinent properties of congruence is presented in the next section.

A residue number system is defined by a set of N integer constants, {m1, m2, m3, ... , mN }, referred to as the moduli. Let M be the least common multiple of all the mi. Any arbitrary integer X smaller than M can be represented in the defined residue number system as a set of N smaller integers

{x1, x2, x3, ... , xN} with xi = X modulo misrepresenting the residue class of X to that modulus.

Practical applications

RNS have applications in the field of digital computer arithmetic. By decomposing in this a large integer into a set of smaller integers, a large calculation can be performed as a series of smaller calculations that can be performed independently and in parallel. Because of this, it's particularly popular in hardware implementations. Computation speeds have increased dramatically during the past three decades resulting from the development of various technologies. The execution speed of an arithmetic operation is a function of two factors. One is the circuit technology and the other is the algorithm used. It can be rather confusing to discuss both factors simultaneously; for instance, a ripple-carry adder implemented in GaAs technology may be faster than a carry-look-ahead adder implemented in CMOS. Further, in any technology, logic path delay depends upon many different factors: the number of gates through which a signal has to pass before a decision is made, the logic capability of each gate, cumulative distance among all such serial gates, the electrical signal propagation time of the medium per unit distance, etc. Because the logic path delay is attributable to the delay internal and external to logic gates, a comprehensive model of performance would have to include technology, distance, placement, layout, electrical and logical capabilities of the gates. It is not feasible to make a general model of arithmetic performance and include all these variables. The purpose of this chapter is to give an overview of the different components used in the design of arithmetic operators. The following parts will not exhaustively go through all these components. However, the algorithms used, some mathematical concepts, the architectures, the implementations at the block, transistor or even mask level will be presented. This chapter will start by the presentation of various notation systems. Those are important because they influence the architectures, the size and the performance of the arithmetic components. The well known and used principle of generation and propagation will be explained and basic implementation at transistor level will be given as examples. The basic full adder cell (FA) will be shown as a brick used in the construction of various systems. After that, the problem of building large adders will lead to the presentation of enhancement techniques.

RESIDUE NUMBER SYSTEM (RNS):

RNS refers to Residue Number System. This project mainly deals with the conversion of number system that is from binary to RNS system.RNS system does not any particular base. So, we can call this number system a weight-free number system. The RNS has been considered as an interesting theoretical topic for researchers in recent years. Its importance stems from the absence of carry propagation between its arithmetic units. This facilitates the realization of high-speed, low-power arithmetic. This advantage is of paramount importance in embedded processors, especially those found in portable devices, for which power consumption is the most critical aspect of the design. In this thesis, we aim at developing efficient schemes for the conversion from the conventional representation to the RNS representation and vice versa. The conventional representation can be in the form of an analog continuous-time signal or a digital signal represented in binary format. We present some of the currently available algorithms and schemes of conversion when the signal is in binary representation.

The proposed schemes are aimed to encourage the utilization of RNS in various real-time and practical applications in the future. In this brief, a read-only-memory less structure for binary-to-residue number system (RNS) conversion modulo {2^n±k} is proposed. This structure is based only on adders and constant multipliers. This brief is motivated by the existing {2^n ± k} binary-2-RNS converters, which are particular inefficient for larger values of n. The experimental results obtained for 4n and 8n bits of dynamic range suggest that the proposed conversion structures are able to significantly improve the forward conversion efficiency, with an AT metric improvement above 100%, regarding the related state of the art. Delay improvements of 2.17 times with only 5%area increase can be achieved if a proper selection of the {2^n± k} modulo is performed. RNS refers to the Residue Number System. It always deals with the remainders of the numbers. The remainders in this terminology are known as the “residues” and this type numbering system does not have any particular base. The base in this terminology is known as “modulo”. As it has no particular weight it is known as “weight-free number system”. It is based on “remainder theorem” of modular arithmetic.

AN EFFICIENT VLSI DESIGN FOR A BINARY TO RESIDUE CONVERTER FOR GENERAL BALANCE MODULI {2^n-3, 2^n, 2^n+3}

In the previous session we have looked after the special modulo set {2^n-1, 2^n, 2^+1} .Here we discuss about the improvised version of special modulo set whose modulo set is {2^n-3, 2^n, 2^n+3}. As we have derived some of the formulae for the special modulo set in the same way we have some formulae for this modulo set too.

Therefore the decimal number 2456 with the modulo {13, 16, 19} can be represented in RNS as {12, 8, 8}

Now let us find out the dynamic range for this improvised modulo set.

DYNAMIC RANGE = {13*16*19} = 3952. In this we can find out the RNS representation of a decimal number. There exists more disadvantages than advantages regarding this improvised modulo set {2^n-3, 2^n, 2^n+3}.

REVERSE CONVERSIONS:

In our previous chapters we have learnt how to convert a binary number into RNS number. In this chapter we are about to learn how to convert RNS number into binary number. There are two ways to convert a number from RNS to binary number. They are:

CHINESE REMAINDER THEOREM (CRT) 2. MIXED RADIX CONVERSION (MRC)

CHINESE REMAINDER THEOREM (CRT):

This is one of the best ways to convert a RNS number to binary number. This is also the earlier version for the reverse conversion. RNS to binary conversion can be done with the help of the reverse converter. Let us formulate for this conversion. In order to convert the number we should consider the relatively prime

modulo set $\{m_1, m_2, m_3 \ldots\ldots\ldots\ldots, m_n\}$ and the residue representation is given by $\{r_1,r_2,r_3,\ldots\ldots\ldots\ldots r_n\}$

The representation of residue is given by

$r_i = |X|_{mi}$

Thedecimal number can be obtained by the following formula: $|X|_M = |\sum_{i=1}^{n} r_i * |M_i^{-1}|_{mi} * M_i|_M$

Finally, this is modified as

$|X|_M = \sum_{i=1}^{n} r_i * |M_i^{-1}|_{mi} * M_i$

Where., $M = m_1 * m_2 * m_3 * \ldots\ldots\ldots\ldots m_i$ = Dynamic range $M_i = M/m_i$

It can be explained clearly with an example. Consider the modulo set $\{m_1, m_2, m_3\} = \{3, 4, 5\}$ and the residue set $\{r_1, r_2, r_3\} = \{2,3,1\}$. Now we need to find out the integer X. For this purpose we apply the above formula in that firstly we need find the dynamic range which is the multiplication of the modulo set and it is represented by M.

STEP 1: Finding the dynamic range “M” $M = m_1 * m_2 * m_3 = 3 * 4 * 5 = 60$

STEP 2: Finding the values of “M_i “

Therefore the integer 11 when divided with a modulo set {3,4,5} gives us a residue set {2,3,1}. In this way we can obtain the value of the integer with the help of Chinese Remainder Theorem (CRT)

KOGGE-STONE ADDER: Since we are providing

mantissa and exponent binary inputs separately, they should be converted separately using the moduli mentioned for them respectively. When a 5-bit binary exponent is converted into RNS using moduli set {3, 4, 5 }then the residues each of 3-bit is obtained. So we are considering a 3-bit Kogge-Stone adder in Fig. Similarly when a lO-bit binary exponent is converted into RNS then the residues each of9-bit is obtained. So for this we are considering 9-bit Kogge-Stone adder in below fig.

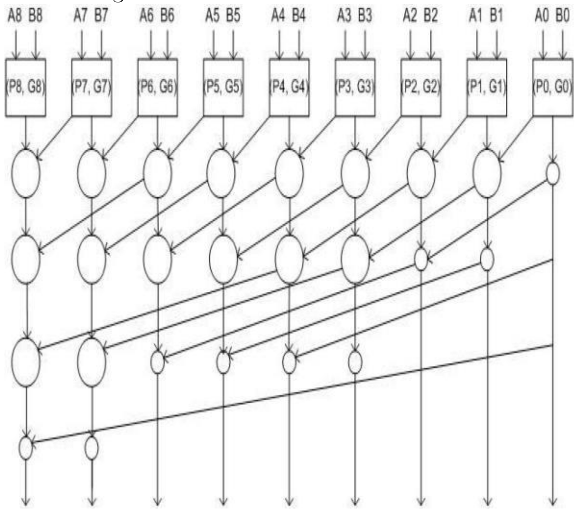
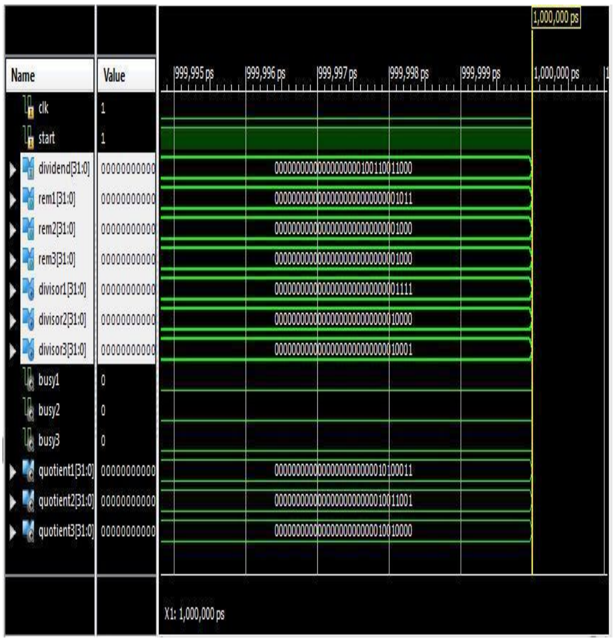


Fig: bit kogg-stone adder

RESULT:


```
Name              Value
clk               1
start             1
dividend[31:0]    00000000000    00000000000000000000100110011000
rem1[31:0]        00000000000    00000000000000000000000000001011
rem2[31:0]        00000000000    00000000000000000000000000001000
rem3[31:0]        00000000000    00000000000000000000000000001000
divisor1[31:0]    00000000000    00000000000000000000000000001111
divisor2[31:0]    00000000000    00000000000000000000000000010000
divisor3[31:0]    00000000000    00000000000000000000000000010001
busy1             0
busy2             0
busy3             0
quotient1[31:0]   00000000000    00000000000000000000000010100011
quotient2[31:0]   00000000000    00000000000000000000000010011001
quotient3[31:0]   00000000000    00000000000000000000000010010000

999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
1,000,000 ps
X1: 1,000,000 ps
```

CONCLUSION:

In this brief, two generic and scalable modulo {2^n±k} binary to RNS conversion structures are proposed for jn-bit Dynamic Ranges. The proposed approach splits the jn input bits into j input sets and computes the residue value using modular additions and constant multiplications, implementing ROM-less structures. To assess the gains achieved by the proposed structures, the experimental results were obtained for 2n and 4n bit dynamic ranges. Neverthless, if a proper selection of the used k values is performed,2.17 times faster conversion operations with only 5% extra area resources can be achieved, with AT performance metric improvements above 100%

REFERENCES:

[1] N. Szabo and R. Tanaka, Residue Arithmetic and Its Applications to Computer Technology.

[2] G. Cardarilli, A. Nannarelli, and M. Re, “Residue number system for low-power DSPapplications”

[3] J. Bajard and L. Imbert, “A full RNS implementation of RSA”

[4] S. Antão, J.-C. Bajard, and L. Sousa, “RNS based elliptic curve point multiplication for massive parallel architectures”

[5] F. E. P. D. Gallaher and P. Srinivasan, “The digit paralell method for fast RNS to weighted number system conversion for specific moduli {2n−1, 2n,2n +1}”

[6] P. A. Mohan, “Reverse converters for the moduli sets {22N − 1, 2N , 22N + 1} and {2N − 3, 2N+ 1, 2N − 1, 2N + 3}”

[7] M.-H. Sheu, S.-H. Lin, C. Chen, and S.-W. Yang, “An efficient VLSI design for a residue to binary converter for general balance moduli {2n − 3, 2n+ 1, 2n − 1, 2n + 3}”

[8] R. Chaves and L. Sousa, “{2n + 1, 2n+k , 2n − 1}: A new RNS moduli set extension”

[9] A. Premkumar and A. P. Vinod, “A memoryless reverse converter for the 4-moduli superset {2n − 1, 2n, 2n + 1, 2n+1 − 1}”

[10] B. Cao, C.-H. Chang, and T. Srikanthan, “An efficient reverse converter for the 4- moduli set {2n−1, 2n , 2n+1, 22n+1} based on the new Chinese remainder theorem”

[11] H. Pettenghi, R. Chaves, and L. Sousa, “RNS reverse converters for moduli sets with dynamic ranges up to (8n+1)-bit”

[12] R. Zimmermann, “Efficient VLSI implementation of modulo {2n ± 1} addition and multiplication”

[13] P. Matutino, H. Pettenghi, R. Chaves, and L. Sousa, “RNS arithmetic units for modulo {2n ± k}”

[14] S. Piestrak, “Design of residue generators and multi operand modular adders using carry-save adders”
[15] A. Premkumar, “A formal framework for conversion from binary to residue numbers”
[16] A. Premkumar, E. Ang, and E.-K. Lai, “Improved memoryless RNS forward converter based on the periodicity of residues”