International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

Design of QSD Number System Addition using Delayed Addition Technique

S.Mallesh
Geethanjali College of Engineering & Technology
Hyderabad
mallesh9549@gmail.com

Dr.C.V.Narasimhulu
Geethanjali College of Engineering & Technology
Hyderabad
narasimhulucv@gmail.com

Abstract: **Quaternary number system is a base-4 numeral system. Using Quaternary Signed Digit (QSD) number system may also execute carry free addition, borrow free subtraction and multiplication. The QSD number system wants a different group of prime modulo based logic elements for each arithmetic operation. In this work we extend this QSD addition to Delayed addition in place of carry free addition. Carry free addition generates intermediate carry and intermediate sum, in this carry propagation is required to generate intermediate sum. To reduce carry propagation we evaluated delayed addition. This delayed addition reduces carry propagation and improves arithmetic calculations. We present both QSD and Floating –point single precision addition using delayed addition. The design work is carried by using Verilog HDL in ISE.**

Keywords: **QSD, DA, CFA and Floating-Point.**

I. INTRODUCTION

Quaternary is the base-4 numeral system. The degree of redundancy usually increases with increase of radix. It uses the digits 0, 1, 2 and 3 to signify any real number it shares with all fixed-radix numeral systems many properties, such as the capability to signify any real number with a canonical illustration (almost unique) and the characteristics of the representations of rational numbers and irrational numbers.

The high speed digital circuit uses the various arithmetic operations. These arithmetic operations are widely used and play significant role in different digital systems such as computers and signal processors. Designing this Arithmetic unit using QSD number representation has attracted the interest of many researchers. Additionally, current advances in technologies for included circuits make large scale arithmetic circuits suitable for VLSI implementation. The propose a high speed QSD adder design. The QSD addition operation employs a fixed number of min terms for any operand size. By using Wallace trees to accumulate results without carry propagation over head. The Wallace tree uses 3:2 or 4:2 compressors to perform addition operation. The fig.1 represents the n bit addition using 3:2 and 4:2 compressors.

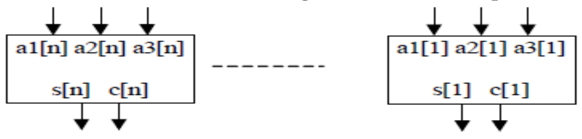

Fig 1.(a)

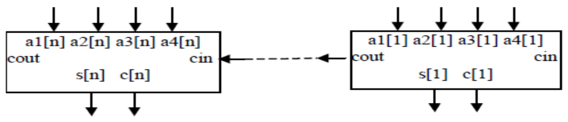

Fig 1.(b)
Fig.1 a) n bit adder using 3:2 compressors
Fig.1 b) n bit adder using 4:2 compressors

In the present work the design of an QSD adder using 3:2 and 4:2 compressors and also we designed floating point adder in Verilog HDL in Xilinx ISE environment based on Spartan 3E FPGA family.

Present work is divided as follows: Section II presents the QSD number system; section III presents the Carry Free addition; section IV is dedicated delayed addition and floating point number system and finally section V is for conclusion of the work.

II. QSD NUMBER SYSTEM

*QSD numbers save 25% storage compared to BSD:*To represent a numeric value N log 4*N* number of QSD digits and 3 log 4*N* binary bits are required while for the same log 2*N* BSD digits and *2* log 2*N* binary bits are required in BSD representation. Ratio of number of bits in
QSD to BSD representation for an arbitrary number N is, 3 |log 4 N| / 2| log 4 N| which roughly equals to ¾. Therefore QSD saves ¼ storage used by BCD.

The proposed QSD adder is better than RBSD adder in terms of number of gates, input connections and delay though both perform addition within constant time. Proposed design has the advantages of both parallelisms as well as reduced gate complexity. The computation speed and circuit complexity increases as the number of computation steps decreases. A two step schemes appear to be a prudent choice in terms of computation speed and storage complexity. Quaternary is the base 4 redundant number system. The degree of redundancy usually increases with the increase of the radix [3]. The signed digit number system allows us to implement parallel arithmetic by using redundancy. QSD numbers are the SD numbers with the digit set as: { 3, 2,1, 0,1, 2, 3 } where 3, 2, and 1 represent -3, -2, and -1 respectively. In

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

general, a signed-digit decimal number D can be represented in terms of an n digit quaternary signed digit number as

$$D = \sum_{i=0}^{n-1} X_i 4^i$$

where *xi* values are { -3, -2,-1, 0,1, 2, 3 }for producing an appropriate decimal representation. For digital implementation, QSD numbers are represented using 3-bit 2's complement notation. A QSD negative number is the QSD complement of the QSD positive number [3]. For example, using primes to denote complementation, we have 3 ' = 3, 3' =3 , 2 ' = 2, 2' = 2 , 1'= 1, 1' =1.

Example: The QSD is applied for any two numbers A = 107 and B = 233 (One number is positive and another number is negative).

First convert the decimal number to their equivalent QSD representation:

(107)10 = (1 2 2 3) QSD

(233)10 = (3 2 2 1) QSD

107+233=(340)10 = (11110)QSD.

The Table 1 representing the Decimal to quaternary number representation.

Decimal Number system	Quaternary Number system
0	0
1	1
2	2
3	3
4	10
5	11
6	12
7	13
8	20
9	21
10	22

Table 1.

III. CARRY FREE ADDITION

Two steps concerned in the carry-free addition. The first step generates an intermediate carry and sum. The second step combines the intermediate sum of the present digit with the carry of the lower significant digit. To avoid carry from further rippling, two rules are used. The first rule contains that the magnitude of the intermediate sum must be less than or equal to 2. The second rule contains that the magnitude of the carry must be less than or equal to 1. Consequently, the magnitude of the second step output cannot be greater than 3 which can be represented by a single-digit QSD number, hence no further carry is required. In step 1, all possible input pairs of the addend and augends are measured.

The range of input numbers can differ from -3 to +3, so the addition result will differ from -6 to +6 which wants two QSD digits. The lower significant digit serves as sum and most significant digit serves as carry. The generation of the carry can be avoided by mapping the two digits into a pair of intermediate sum and intermediate carry such that the nth intermediate sum and the (n-1)th intermediate carry never form any carry generating pair (3,3), (3,2), (3,1), (3 , 3), (3, 2), (3,1). If we restrict the representation such that the intermediate carry is limited to a maximum of 1, and the intermediate sum is restricted to be less than 2, then the final addition will become carry free.

Using 6 variable K-map, the logic equations specifying a minimal hardware realization for generating the intermediate carry and intermediate sum are derived. The min terms for the intermediate carry (IC2 , IC1, IC0) are:

$$IC_2 = a_2 b_2 \left(\overline{a_0 b_0 a_1 b_1}\right) + \left(\overline{a_1 + b_1}\right)\left(a_2 \overline{b_0} + b_2 \overline{a_0}\right)$$

$$IC_1 = a_2 b_2 \left(\overline{a_0 b_0 a_1 b_1}\right) + \left(\overline{a_1 + b_1}\right)\left(a_2 \overline{b_0} + b_2 \overline{a_0}\right)$$

$$IC_0 = IC_2 + \overline{a_2}\,\overline{b_2}\left(a_1 b_1 + b_1 b_0 + b_0 a_1 + b_1 a_0 + a_1 a_0\right)$$

Minterms for intermediate sums are:

$$IS_0 = a_0 \overline{b_0} + \overline{a_0} b_0$$

$$IS_1 = \left(a_1 \overline{b_1} + \overline{a_1} b_1\right)\overline{a_0 b_0} + \left(\overline{a_1 \overline{b_1} + \overline{a_1} b_1}\right) a_0 b_0$$

$$IS_2 = IS_0\left(\overline{a_1} b_1 + a_1 \overline{b_1}\right) + b_2 \overline{a_1}\,\overline{b_0} + a_2 \overline{b_1}\,\overline{a_0} + a_0 b_0 \overline{a_1}\,\overline{b_1}(a_2 +$$

The final sum which is carry free is generated from those outputs i.e. Intermediate carry (IC2, IC1, and IC0) and Intermediate sum (IS2, IS1, and IS0). Therefore it has six input and three output bits.

III. Delayed Addition and Floating Point Addition

Delayed Addition:

A multiply-accumulator unit consists of a multiplier and an Adder, he hardwired ripple-carry adder is the fastest. Before continuing on detailed designs, we will first give a brief review on some basics of Wallace tree and its derivatives. One level of Wallace tree is composed of arrays of ***3-2 adders*** (or **compressors**). The logic of a 3-2 adder is the same as a full adder except the carry-out from the previous bit has now become an external input. For each bit of a 3-2 adder, the logic is:

S[i] = A1[i] Å A2[i] Å A3[i];

C[i] = A1[i]A2[i]+ A2[i]A3[i] + A3[i]A1[i];
For the whole array, S+2C = A1 + A1 +A3

S and C are partial results that we refer to in this paper as the ***pseudo-sum***. They can be combined during a final addition phase to compute a true sum. The total number of inputs across an entire level of a 3-2 adder array is the same as the bit-width of the inputs. The typical logic is:

C_{out}[i] = A1[i]A2[i] + A2[i]A3[i] + A3[i]A1[i] ;
S[i] =A1[i] Å A2[i] Å A3[i] Å A4[i] Å C_{in}[i] ;
C[i] = (A1[i] Å A2[i] Å A3[i] Å A4[i])C_{in}[i] +
Ø(A1[i] Å A2[i] Å A3[i] Å A4[i])A4[i] ;
For the whole array, S + 2C = A1 + A2 + A3 + A4
Pseudo-sum = Pseudo-sum + (the final two partial products for each multiplication).

Floating Point Representation:

A floating-point MAC unit uses too much area to fit on a single FPGA chip. The major reason is that floating-point accumulation is a much more complex process than the integer case, as explained below. Rather than a MAC unit, we instead focus here on a floating-point accumulator using delayed addition.

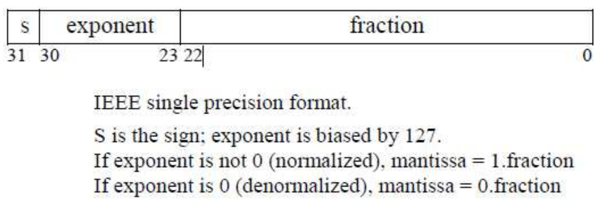
s
exponent
fraction
31
30
23
22
0
IEEE single precision format.
S is the sign; exponent is biased by 127.
If exponent is not 0 (normalized), mantissa = 1.fraction
If exponent is 0 (denormalized), mantissa = 0.fraction

Figure 2: floating point representation

As shown in Fig.2, a traditional floating-point adder would first extract the 1-bit sign, 8-bit exponent and 23- it fraction of each incoming number from the IEEE 754 single precision format. By checking the exponent, the adder determines if each incoming number is de normalized. If the exponent bits are all "0", which means the number is de normalized, the mantissa is 0.fraction, and otherwise, mantissa is 1.fraction. Next, the adder compares the exponents of the two numbers and shifts the mantissa of the smaller number to get them aligned. Sign-adjustments also occur at this point if either of the incoming numbers is negative. Next, it adds the two mantissas; the result needs another sign-adjustment if it is negative. Finally the adder re-normalizes the sum, adjusts the exponent accordingly and truncates the resulting mantissa into 24 bits by the appropriate rounding scheme. The above algorithm is designed for a single addition rather than a series of additions. Even more so than in the integer case, this straightforward approach is difficult to pipeline. One problem lies in the fact that the incoming next element- to-be-summed must be aligned with the current accumulated result. This adds a challenge to our delayed addition technique since we do not keep the accumulated result in its final form, and thus cannot align incoming addends to it. Likewise, at the end of the computation, renormalization also impedes a delayed addition approach.

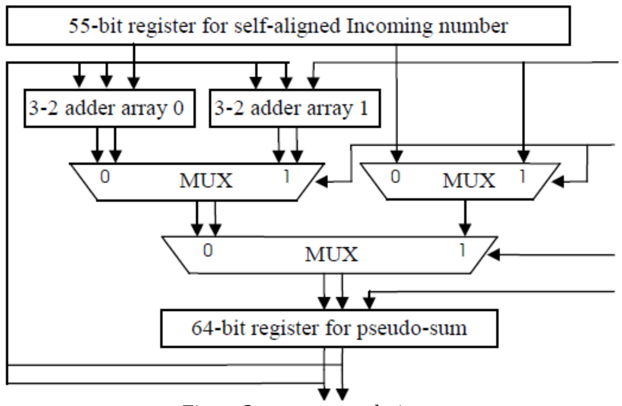
55-bit register for self-aligned Incoming number
3-2 adder array 0
3-2 adder array 1
0
MUX
1
0
MUX
1
0
MUX
1
64-bit register for pseudo-sum

Figure 3: compressor design.

Figure 3 shows the design layout for one of the two compressor units in the design, namely compressor-0. Compressor-1 has essentially identical structure, except that it cross-connects with adder-0 as shown in Figure 4. The running pseudo-sum is stored as the Wallace tree's S and C partial results in the 64-bit registers shown. Were it not for the possibility of either pseudo-sum overflowing, the design would now be complete. Since the accumulated result may exceed the register capacity, we have also devised a technique for recognizing and responding to potential pseudo-sum overflows. Since we are not doing the full carry-propagation of a traditional adder, we cannot use the traditional overflow-detection technique of comparing carry-in and carry-out at the highest bit. In fact, without performing the final add to convert the pseudo-sum to the true sum, it is impossible to precisely know a *priori* when overflows will occur.

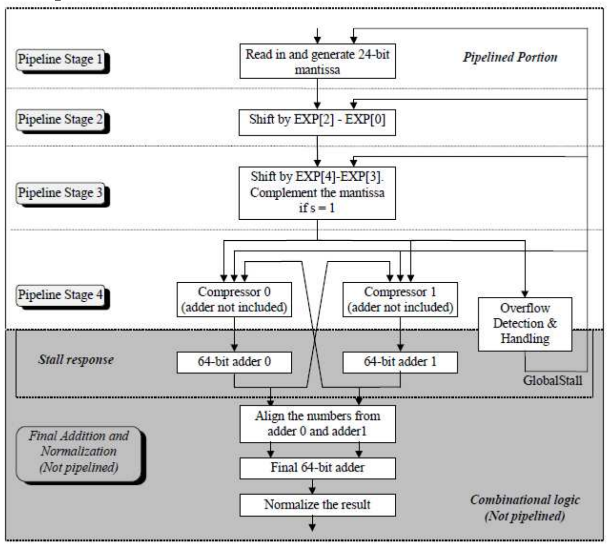
Pipeline Stage 1
Read in and generate 24-bit mantissa
Pipelined Portion
Pipeline Stage 2
Shift by EXP[2] - EXP[0]
Pipeline Stage 3
Shift by EXP[4]-EXP[3]. Complement the mantissa if s = 1
Pipeline Stage 4
Compressor 0 (adder not included)
Compressor 1 (adder not included)
Overflow Detection & Handling
Stall response
64-bit adder 0
64-bit adder 1
GlobalStall
Align the numbers from adder 0 and adder1
Final Addition and Normalization (Not pipelined)
Final 64-bit adder
Normalize the result
Combinational logic (Not pipelined)

Figure 4: floating point accumulation scheme.

V.CONCLUSION

QSD number system addition using delayed addition, and floating point addition using delayed addition has been designed using Verilog HDL the simulations are performed using Active HDL and implementation performed using Xilinx tool.

REFERENCES

[1]. A. Avizinis "signed digit number representation for fast parallel arithmetic", IRE Transactions on Elec. Comp..Vol EC-10,pp 389-400,sept-1961.
[2]. A.A.S. Awwal and J.U. Ahmed, "fast carry free adder design using QSD number system ,"proceedings of the IEEE 1993 national aerospace and electronic conference, vol 2,pp 1085-1090,1993.
[3]. F. Kharbash and G. M. Chaudhry, "Reliable Binary Signed Digit Number Adder Design", IEEE Computer Society Annual Symposium on VLSI, pp 479-484, 2007.
[4]. John Moskal, Erdal Oruklu and Jafar Saniie, "Design and Synthesis of a Carry-Free Signed-Digit Decimal Adder", IEEE International symposium on Circuits and Systems, pp 1089-1092, 2007.
[5]. P. K. Dakhole, D.G. Wakde, " Multi Digit Quaternary adder onProgrammable Device : Design and verification", InternationalConference on Electronic Design, pp. 1-4, Dec 2008.
[6]. IEEE Standards Board. "IEEE Standard for Binary Floating-Point Arithmetic". Technical Report ANSI/IEEE Std. 754-1985, Institute of Electrical and Electronics Engineers, New York, 1985.