U. Abhishek* et al.
(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH
Volume No.4, Issue No.6, October – November 2016, 4341-4344.

BCD Repetitive Strategy To Decrease The Latency

U.ABHISHEK
VLSI&ES Branch
Department of ECE
Holy Mary Institute of Technology
Hyderabad, T.S

Mrs. Y.B.T.SUNDARI M.Tech.,MIEEE
Assistant Professor
Department of ECE
Holy Mary Institute of Technology
Hyderabad, T.S

Abstract: **We present the formula and architecture of the BCD parallel multiplier that exploits some qualities of two different redundant BCD codes to hurry up its computation: the redundant BCD excess-3 code (XS-3), and also the overloaded BCD representation (ODDS). Additionally, new techniques are designed to reduce considerably the latency and section of previous representative high end implementations. Partial goods are generated in parallel utilizing a signed-digit radix-10 recoding from the BCD multiplier using the digit set [-5, 5], and some positive multiplicand multiples (0X, 1X, 2X, 3X, 4X, 5X) created in XS-3. This encoding has lots of advantages. First, it's a self-complementing code, to ensure that an adverse multiplicand multiple could be acquired just by inverting the items of the related positive one. Finally, the partial products could be recoded towards the ODDS representation just by adding a continuing factor in to the partial product reduction tree. Because the ODDS utilize a similar 4-bit binary encoding as non-redundant BCD, conventional binary VLSI circuit techniques, for example binary carry-save adders and compressor trees, could be adapted efficiently to do decimal operations. We reveal that the suggested decimal multiplier comes with an area improvement roughly within the range 20-35 % for similar target delays with regards to the fastest implementation.**

Keywords: **BCD Codes; Multiplier; VLSI Circuit; Adder**

I. INTRODUCTION

Power dissipation is known as a vital parameter in modern VLSI design field. To fulfill MOORE'S law and also to produce electronic devices goods with increased backup and fewer weight, low power VLSI design is essential. Fast multipliers are crucial areas of digital signal processing systems. The rate of multiply operation is crucial in digital signal processing plus the overall purpose processors today, especially because the media processing required off. Previously multiplication was generally implemented using a sequence of addition, Subtraction, and shift operations [1]. Multiplication can be viewed as a number of repeated additions. The delayed, gated demonstration of the multiplicand must be within the same column from the shifted partial product matrix. They're then put into make up the product bit for that particular form. Multiplication thus remains a multi operand operation. To increase the multiplication to both signed and unsigned figures, a handy number system will be the representation of figures in two's complement format. Multipliers are critical factors of numerous high end systems for example FIR filters, microprocessors, digital signal processors, etc. A system's performance is usually based on the performance from the multiplier since the multiplier is usually the slowest clement within the system. In addition, it's usually the most area consuming. Hence, optimizing the rate and part of the multiplier is really a major design issue. To ensure that improving speed results mostly in bigger areas. Consequently, whole spectrums of multipliers with various area-speed constraints are made with fully parallel processing. These multipliers have moderate performance both in speed and area. However, existing digit serial multipliers happen to be affected by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which work on digits inside a parallel fashion rather of bits bring the pipelining towards the digit level and steer clear of the majority of the above problems. The pipelining done in the digit level brings the advantage of constant operation speed regardless of how big the multiplier. The time speed is just based on the digit size that is already fixed prior to the design is implemented. The SPST (Spurious Power Suppression Technique) can be used for digital signal processing (DSP), Transformations of Digital Image Processing and versatile multimedia functional unit (VMFU) etc. The Booth's radix-4 formula, Modified Booth Multiplier, 34-bit CSA are improves speed of Multipliers and SPST adder will lessen the power consumption additionally process. The multiplicand will be multiplied by each digit from the multiplier starting with the rightmost, Least Significant Digit (LSD). Intermediate results (partial-products) are put one atop another, offset by one digit to align digits of the identical weight.

II. BOOTH MULTI-PLIER

The fundamental approach to enhance the performance from the final adder would be to decrease the amount of input bits. The multiple partial goods are compressed right into a sum along with a carry by CSA [2]. The amount of items of sums and carries to become used in the ultimate adder is reduced with the addition of the low items of sums and carries ahead of time inside the range where the efficiency won't be degraded. A Couple-

bit CLA can be used to include the low bits within the CSA. Additionally, to improve the output rate when pipelining is used, the sums and carries in the CSA are accrued rather from the outputs in the final adder in the way the sum and carry in the CSA in the last cycle are inputted to CSA. A CSA architecture is modified to deal with the sign bit the sum and carry in the CSA in the last cycle are inputted to CSA. The multiple partial goods are compressed right into a sum along with a carry by CSA.

$$0 \le l \le e, \quad 9 + e \le m \le 2^4 - 1 (= 15).$$

III. PROPOSED SPST MODEL

VMFU consists of an adder, multiplier as well as an accumulator. Usually adders implemented are Carry- Select or Carry-Save adders, as speed is very important in DSP. One implementation from the multiplier may be as a parallel array multiplier [3]. The inputs for that VMFU should be fetched from memory location and given towards the multiplier block. This whole process will be achieved in one clock cycle. This has been developed in the work. The look includes one 16 bit register, one 16-bit Modified Booth Multiplier, 33-bit accumulator using ripple carry and two16-bit accumulator registers.

IV. METHODOLOGY

The proposed decimal multiplier uses internally a redundant BCD arithmetic to speed up and simplify the implementation. This arithmetic deals with radix-10 ten‘s complement integers of the form:

$$Z = -s_z \times 10^d + \sum_{i=0}^{d-1} Z_i \times 10^i, \qquad (1)$$

Where d is the number of digits, sz is the sign bit, and Zi €[l-e; m-e]is the ith digit, with parameter e is the excess of the representation and usually takes values 0 (non excess), 3 or 6. The redundancy index ρ is defined as ρ=m-l+1-r being r=10.]. On the other hand, the binary value of the 4-bit vector representation of Zi is given by

$$[Z_i] = \sum_{j=0}^{3} z_{i,j} \times 2^j,$$

zi;j being the jth bit of the ith digit. Therefore, the value of digit Zi can be obtained by subtracting the excess e of the representation from the binary value of its 4-bit encoding, that is,

$$Z_i = [Z_i] - e.$$

$$-10^d + \sum_{i=0}^{d-1} (9 - Z_i) \times 10^i. \qquad (3)$$

Note that bit-weighted codes such as BCD and ODDS use the 4-bit binary encoding (or BCD encoding) defined in Expression (2). Thus, Zi=[Zi] for operands Z represented in BCD or ODDS. This binary encoding simplifies the hardware implementation of decimal arithmetic units. For input digits of the multiplicand in conventional BCD, the multiplication by 3 leads to a maximum decimal carry to the next position of 2 and to a maximum value of the interim digit of 9. Therefore the resultant maximum digit is 11. Therefore the redundant BCD representations can host the resultant digits with just one decimal carry propagation. An important issue for this representation is the ten‘s complement operation. Since after the recoding of the multiplier digits, negative multiplication digits may result, it is necessary to negate. The implementation of [9-Zi] leads to a complex implementation, since the Zi digits of the multiples generated may take values higher than 9. A simple implementation is obtained by observing that the excess-3 of the nine‘s complement of an operand is equal to the bit-complement of the operand coded in excess-3. We show how the nine‘s complement can be performed by simply inverting the bits of a digitZi coded in XS-3.

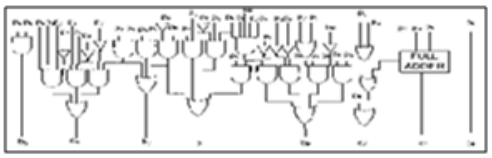

Fig.1.BCD conversion logic

V. PROPOSED SYSTEM

Within our method the sum correction is evaluated concurrently using the binary carry-save additions using posts of binary counters. Essentially we count the amount of carries per decimal column along with multiplication by 6 is conducted. It makes sense added like a correction term towards the creation of the binary carry-save reduction tree [4]. This improves considerably the latency from the partial product reduction tree. The suggested architecture accepts a random quantity of ODDS or BCD operand inputs. A number of PPR tree structures presented also exploit an identical idea. But depend on the custom-designed ODDS adder to do a few of the stage reductions. Our proposal aims to supply an ideal reuse associated with a binary CSA tree for multi operand decimal addition, because it was one out of for that 4221 and 5211 decimal coding. The array ofdþ1ODDS partial products may very well be adjacent digit posts of height hdþ1. Since ODDS digits are encoded in binary, the guidelines for binary arithmetic apply inside the digit bounds, and just carries generated between radix-10 digits (4-bit posts) lead towards the decimal correction from the binary sum. Stage 1) Decimal partial product generation: A SD radix-10 recoding from the BCD multiplier has been utilized. This recoding creates a reduced quantity of partial items those results in a

significant decrease in the general multiplier area. Therefore, the recoding from the d-digit multiplier Y into SD radix-10 digits Ybd-1...Yb0,produces d partial products PP[d-1]...PP[]. One per digit choosing the right multiplicand multiple, . Yet another partial product PP[d] is created by the most important multiplier digit following the recoding, so the final amount of partial products generated is dþ1. As opposed to our previous SD radix-10 implementations, 3X is acquired inside a reduced constant time delay (˜3 XOR-gate delays) using the XS-3 representation. Consequently, the latency is reduced and also the hardware implementation is simplified. The plan suggested also produces 3X in constant time but using redundant signed-digit BCD arithmetic. Stage 2) Decimal partial product reduction. Within this stage, the variety of d 1 ODDS partial goods are reduced to 2 2d-digit words (A,B). Our proposal uses binary carry save adder tree to do carry-free additions from the decimal partial products. The array ofdþ1ODDS partial products may very well be adjacent digit posts of height hdþ1. Since ODDS digits are encoded in binary, the guidelines for binary arithmetic apply inside the digit bounds, and just carries generated between radix-10 digits (4-bit posts) lead towards the decimal correction from the binary sum. The binary sum should be incremented by 6 in the appropriate position to get the correct decimal sum (modulo 10 additions). Two previous designs implement tree structures for adding ODDS operands. A combinational logic block can be used to look for the sum correction in the end the operands happen to be put in a binary CSA tree, using the most of inputs restricted to 19 BCD operands. 2 By comparison, within our method the sum correction is evaluated concurrently. The binary carry-save additions using posts of binary counters. This improves considerably the latency from the partial product reduction tree. Our proposal aims to supply an ideal reuse associated with a binary CSA tree for multioperand decimal addition, because it was one out of for that 4221 and 5211 decimal codlings. Stage 3) Conversion to (non-redundant) BCD. We consider using a BCD carry-propagate adder to do the ultimate conversion to some non-redundant BCD product P=A B -. The suggested architecture is really a 2d-digit hybrid parallel prefix/carry-select adder. Decimal Partial Product Generation: The partial product generation stage comprises the recoding from the multiplier to some SD radix-10 representation, the calculation from the multiplicand multiples in XS-3 code and also the generation from the ODDS partial products [5]. Decimal Partial Product Reduction: The PPR tree includes three parts: (1) a normal binary CSA tree to compute estimation from the decimal partial product sum inside a binary carry-save form (S, C), (2) an amount correction block to count the carries generated between your digit posts. A decimal digit 3:2 compressor which increments the carry-save sum based on the carries count to get the final double-word product (A B -), A being symbolized with excess-6 BCD digits and B being symbolized with BCD digits. The PPR tree may very well be adjacent posts of h ODDS digits each, h to be the column height, and h=d 1. Final Conversion To Bad: The chosen architecture is a2d-digit hybrid parallel prefix/carry-select adder, the BCD Quaternary Tree adder. The delay of the adder is slightly greater towards the delay of the binary adder of8dbits having a similar topology. The decimal carries are computed utilizing a carry prefix tree, while two conditional BCD digit sums are computed from the critical path using 4-bit digit adders which implements [Ai] Bi and [Ai] Bi 1.VLSI Technology: VLSI systems tend to be smaller sized and eat fewer power compared to discrete components accustomed to build electronic systems prior to the 1960s. Integration enables us to construct systems with lots of more transistors, allowing a lot more computing capacity to be relevant to solving an issue. Integrated circuits will also be much simpler to create and manufacture and therefore are more reliable than discrete systems that assists you to develop special-purpose systems which are more effective than general-purpose computers to complete the job at hands.

Fig.2.Technology schematic & Synthesis result

VI. CONCLUSION

Within this paper we've presented the formula and architecture of the new BCD parallel multiplier. The enhancements from the suggested architecture depend on prescribed medication redundant BCD codes, the XS-3 and ODDS representations. Partial products could be generated extremely fast within the XS-3 representation while using SD radix-10 PPG plan: positive multiplicand multiples (0X, 1X, 2X, 3X, 4X, 5X) are recomputed inside a carry-free way. However, recoding of XS-3 partial products towards the ODDS representation is straightforward. The Chances representation uses the redundant digit-set [, 15]. Along with a 4-bit binary encoding (BCD encoding), which enables using a binary carry-save adder tree to do partial product reduction in an exceedingly efficient way. We've presented architectures for IEEE-754 formats, Decimal64 and Decimal128. The region and delay figures believed from both a theoretical model and synthesis reveal that our BCD multiplier presents 20-35 % less area than other kinds for any given target delays.

VII. REFERENCES

[1] A. Aswal, M. G. Perumal, and G. N. S. Prasanna, ―On basic financial decimal operations on binary machines,‖IEEE Trans. Comput., vol. 61, no. 8, pp. 1084–1096, Aug. 2012.

[2] M. F. Cowlishaw, E. M. Schwarz, R. M. Smith, and C. F. Webb, ―A decimal floating-point specification,‖ inProc. 15th IEEE Symp. Comput. Arithmetic, Jun. 2001, pp. 147–154.

[3] M. F. Cowlishaw, ―Decimal floating-point: Algorism for computers,‖ inProc. 16th IEEE Symp. Comput. Arithmetic, Jul. 2003, pp. 104–111

[4] S. Carlough and E. Schwarz, ―Power6 decimal divide,‖ inProc. 18th IEEE Symp. Appl.-Specific Syst., Arch., Process., Jul. 2007, pp. 128–133.

[5] S. Carlough, S. Mueller, A. Collura, and M. Kroener, ―The IBM zEnterprise-196 decimal floating point accelerator,‖ inProc. 20th IEEE Symp. Comput. Arithmetic, Jul. 2011, pp. 139–146.

