International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

A High Performance Binary to BCD Converter for Decimal Multiplication Using VLSI Technique

Mr.Sadashiv Badiger
III Semester, M.Tech (Digital Electronics)
Dept. of Electronics &Communication,
S.D.M.College of Engineering and Technology.
Dharwad, Karnataka.

Prof. B.K.Saptalakar
Assistant Professor,
Dept of Electronics & Communication,
S.D.M. College of Engineering and Technology.
Dharawad, Karnataka.

Abstract-Decimal data processing applications have grown exponentially in recent years thereby increasing the need to have hardware support for decimal arithmetic. Binary to BCD conversion forms the basic building block of decimal digit multiplier.A novel high speed architecture for binary to BCD conversion which is better in terms of delay is presented in this paper. This decimal multiplication in turn is an integral part of commercial, internet and financial based applications.

IINTRODUCTION

In commercial business and internet based applications, decimal arithmetic is receiving significant importance. Decimal arithmetic is important in many applications such as banking, tax calculation, insurance, accounting and many more. Even though binary arithmetic is used widely, decimal computation is essential. The decimal arithmetic is not only required when numbers are presented for inspection by humans, but also it is a necessity when fractions are being used. Rational numbers whose denominator is a power of ten are decimal fractions and most them cannot be represented by binary fractions. For example, the value 0.01 may require an infinitely recurring binary number. Even though the arithmetic is correct, but if binary approximation is used instead of an exact decimal fraction, results can be wrong.

This extensive use of decimal data indicates that it is important to analyse how the data can be used and how the decimal arithmetic can be defined.However, the current general purpose computers perform decimal computations using binary arithmetic. But the problem is that decimal numbers such as 0.3 cannot be represented precisely in binary. The errors that result on conversion between decimal and binary formats cannot be tolerated as long as precision is concerned. Since decimal arithmetic hasgained wide importance in financial analysis,banking, tax calculations. Insurance, telephone billing, such errors cannot be tolerated. This in turn can be overcome by using a decimalArithmetic and logic unit. The operationsrelated to decimal arithmetic are typically slower, more complex and occupy more area and this leads to more power and less speed when implemented in hardware. Hence, enhancing speed and reducing area is the major consideration while implementing decimal arithmetic. In this paper we will be estimating the delay and power.

II BINARY CODED DECIMAL (BCD)

In computing and electronic systems, binary -coded decimalis a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits, usually four or eight, although other sizes have been used historically. Special bit patterns are sometimes used for a sign or for other indications. In byte-oriented systems, the term uncompressed BCD usually implies a full byte for each digit, whereas packed BCD typically encodes two decimal digits within a single byte by taking advantage of the fact that four bits are enough to represent the range 0 to 9. BCD's main virtue is a more accurate representation and rounding of decimal quantities as well as an ease of conversion into human-readable representations. As compared to binary positionalsystems, BCD's principal drawbacks are a small increase in the complexity of the circuits needed to implement basic arithmetic and a slightly less dense storage.BCD was used in many early decimal computers. Although BCD is not as widely used as in the past, decimal fixed-point and floatingpoint formats are still important and continue to be used in financial, commercial, and

industrial computing, where subtle conversion and rounding errors that are inherent to floating point binary representations cannot be tolerate.

III IMPORTANCE OF DECIMAL MULTIPLICATION

Decimal multiplication is an integral part of financial, commercial, and internet-based computations. The basic building block of a decimal multiplier is a single digit multiplier. It accepts two Binary Coded Decimal (BCD) inputs and gives a product in the range [0, 81] represented by two BCD digits. A novel design for single digit decimal multiplication that reduces the critical path delay and area .

Out of the possible 256combinations for the 8-bit input, only hundred combinations are valid BCD inputs. In the hundred valid combinations only four combinations require 4 x 4multiplication, 64 combinations need 3 x 3 multiplication, and the remaining 32 combinations use either 3 x 4 or 4 x3 multiplication. This design leads to more regular VLSI implementation, and does not require special registers for storing easy multiples. This is a fully parallel multiplier utilizing only combinational logic, and is extended to a Hex/Decimal multiplier that gives either a decimal output or a binary output. The accumulation of partial products generated using single digit multipliers is done by an array of multi-operand BCD adders for an (n-digit x n-digit)multiplication.

The main objective of this project is to perform highly efficient fixed bit binary to BCD conversion in terms of delay, power and area. As mentioned earlier, most of the recently proposed multipliers use 7-bit binary to 8-bit/2-digit BCD converters. this project has been specifically designed for such converters.

IVLITERATURE SURVEY

Nowadays, decimal arithmetic is receiving significant attention in the financial, commercial, and internet-based applications. These applications often store data in decimal Format [1]. Currently, general purpose computers do decimal computations using binary arithmetic. But, a number of decimal numbers such as 0.2 cannot be represented precisely in binary. In this world of precision, such errors generated by conversion between decimal and binary formats are no more tolerable. Recently, support for decimal arithmetic has received increased attention due to the growing importance in financial analysis, banking, tax calculation, currency conversion, insurance, telephone billing and accounting which cannot tolerate such errors. This can be overcome by using a decimal arithmetic and logic unit (ALU)[2]. Decimal arithmetic operations are typically more complex, slower and occupy more area leading to more power and less speed when implemented in hardware. Hence, the major consideration while implementing decimal arithmetic is to enhance its speed and reduce area as much as possible. In an iterative decimal multiplier presented in [2], decimal partial products are generated by creating two partial products for each multiplier digit. Multiplying two n-digit Binary Coded Decimal (BCD) numbers requires n iterations, where all iterations consist of two binary carry-save additions and three decimal corrections. After n iterations, the carry and sum are added using a decimal carry-propagate adder to produce the final product. The multiplier presented in [3] generates the partial products by the retrieval of product of BCD digits from look-up tables. Several existing designs for decimal multiplication generate and store multiples of the multiplicand before partial product generation, and then use the multiplier digits to select the appropriate multiple as the partial product [3, 4]. The multiplier in [5] stores intermediate product digits in a less restrictive, redundant format called the overloaded decimal representation that reduces the delay of the iterative portion of the multiplier. From considering all the above factors The proposed converter is flexible and can be plugged into any homogeneous multiplication architectures to achieve better performance irrespective of the method used to generate binary partial products.

V METHODOLOGY

Decimal data processing applications have grown exponentially in recent years thereby increasing the need to have hardware support for decimal arithmetic. Binary to BCD conversion forms the basic building block of decimal digit multipliers. it consist of following methodology

BCD is a decimal representation of a number directly coded in binary, digit by digit. For example, the number (9321)10 = (1001 0011 0010 0001) BCD. It can be seen that each X and Y digit of the decimal number is coded in binary and then concatenated to form the BCD representation of the decimal number. As any BCD digit lies between [0, 9] or [0000, 1001], multiplying two BCD digits can result in numbers between [0, 81]. All the possible combinations can be represented in a 7-bit binary number when multiplied, (81)10 or (1010001)2 being the highest. In BCD multiplication where 4-bitbinary multipliers are used to multiply two BCD numbers with digits, Xi and Yj, respectively, a partial product Pij is generated of the form (p6p5p4p3p2p1p0)2. Conversion of Pij from binary to a BCD number Bi Cj where π (Xi, Yj) = 10Bi + Cj needs fast and efficient BC0D converters. The binary to BCD conversion is generally inefficient if the binary number is very large. Hence the conversion can be done in parallel for every partial product after each BCD digit is multiplied as shown in Figure 1 and the resulting BCD numbers after conversion can be added using BCD adders. Another alternative would be to compress the partial products of all binary terms in parallel and then convert them to BCD.

X= X2 X1 X0 Multiplication of

Y= Y2 Y1 Y0 BCD Digits

P20 P10 P00 Binary Partial

P21 P11 P01 Product

P22 P12 P02Binary to BCD

B2C0 B1C0 B0C0 Conversion

B2C1 B1C1 B0C1 BCD partial

B2C2B1C2 B0C2 Products

VI. IMPLIMENTATION STEPS

This section describes the project architecture in detail. Maximum utilization of the fact that only limited and small numbers of outcomes are possible for conversion has been made in designing the architecture to reduce delay, power and area. As shown in Figure1, p6p5p4p3p2p1p0 are the binary bits to be converted into BCD bits z7z6z5z4 z3z2z1z0. p6, p5 and p4 are the HSBs while p3, p2, p1 and p0 are the LSBs. Let p6p5p4p3p2p1p0 be the seven binary bits to beconverted into two BCD digits. To convert these binary bits into 2-digit BCD we split the binary number into two parts, the first part contains the lower significant bits (LSBs) p3, p2, p1 and p0 while the second part contains the remaining higher significant bits (HSBs) p6, p5 and p4.

The lower significant part (LSBs) has the same weight as that of a BCD digit and can be directly used to represent a BCD digit. The only exception arrives when p3p2p1p0 exceeds (1001)2 or (9)10. To convert the LSBs into a valid BCD number we check whether p3p2p1p0 exceeds (1001)2, and if it does, we add (0110)2 to it. This procedure of adding (0110)2 whenever the number exceeds (1001)2 is called correction in BCD arithmetic.

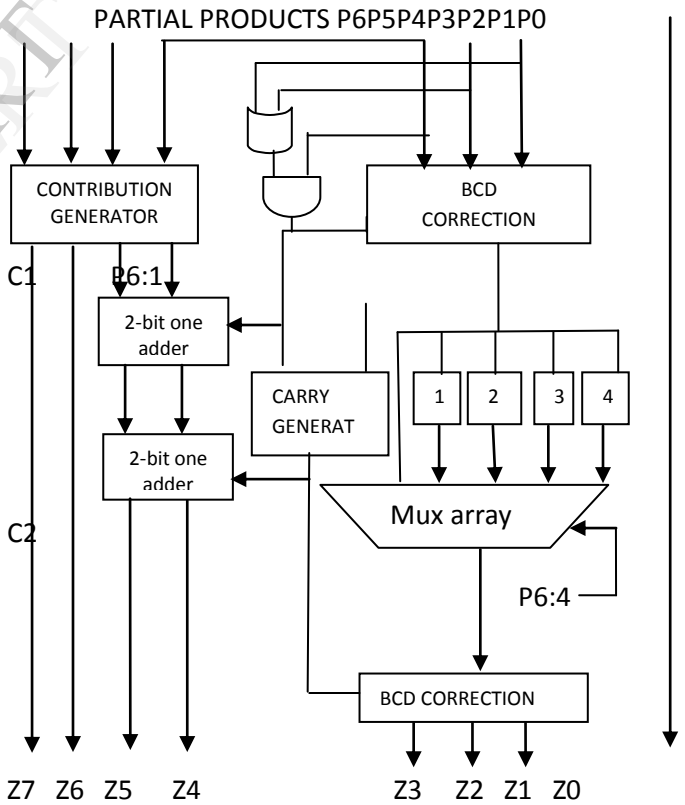

Fig1: proposed architecture

The carry obtained from this procedure is added to the higher significant BCD digit calculated

from the HSBs of the original binary number. The HSBs not only contribute to the higher significant BCD digit but also to the lower significant BCD digit. These contributions of HSBs towards the lower significant digit are added after BCD correction.

The resulting sum is then checked for the case (1001)2 and correction is done if needed to obtain the final lower significant BCD digit. A possible carry from the above operation is added to the higher significant digit resulting in the final higher significant BCD digit.

When two BCD digits are multiplied only six combinations of p6,p5 and p4 (HSBs) are possible, which are 000, 001, 010, 011, 100 and101. Each of these combinations have a different contribution towards the lower and higher significant BCD digits. This contribution can be easily calculated by evaluating the weights of the patterns which are p6x27 + p5x26+ p4x25. Contribution of each of these patterns towards the lower and higher BCD digits is shown in Table.

Table : contribution of hsbs

Higher significant Bits (HSBs)	BCD Weights	
	Higher significant BCD Digits	Lower significant BCD Digits
000	0000	0000
001	0001	0110
010	0011	0010
011	0100	1000
100	0110	0100
101	1000	0000

Z0 is same as p0 and hence no operation is done on p0. {p3, p2 and p1} are used to check whether the LSBs are greater than (1001)2 or not using equation (1) and are sent to the BCD Correction block.

C1 = p3. (p2 + p1) ..... (1)

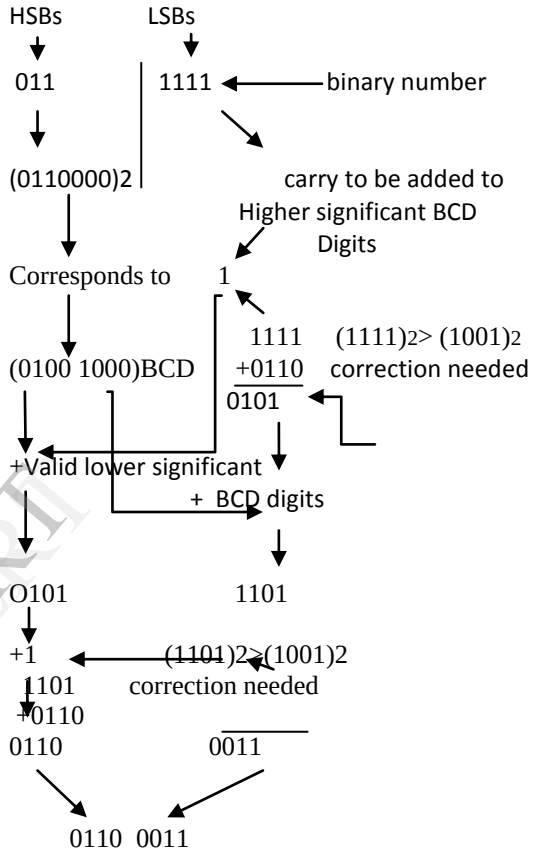

Fig2: Shows an example of the algorithm for number 63 or (0110 0011) BCD

A. BCD CORRECTION

In the BCD Correction block Whenever C1 is high, it adds 011 to the input bits. Below Figure shows the implementation of BCD Correction block.

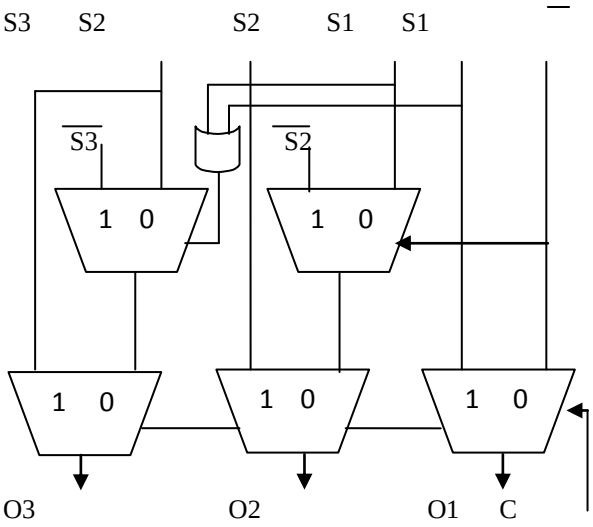

Fig3: BCD correction

B. CONTRIBUTION GENERATION

In parallel, HSBs along with p3 are fed to a simple logic block known as Contribution Generator which produces the higher significant BCD digits. The logic implemented by the Contribution Generator is as follows.

t3 = p6 p4 ..... (2)

t2 = p5 (p4+p3) + p6p4..... (3)

t1 = (p5+p6) p4..... (4)

t0 = p6 p5 p4 + p5p4..... (5)

C1 is the carry from the lower significant digit, so it is added to the higher significant digit t3t2t1t0. It is found that very few cases lead to the propagation of the incoming carry from t1 to t2. Hence, we take advantage of this situation and implement {t3, t2} in combinational logic thus removing the need to add C1 to these terms, thus saving hardware and complexity.

C. 2-BIT ONE ADDRE

A 2-bit One Adder, as shown in Figure 4, is used to add C1 to t0 and t1. There is a possibility of a carry generation, when the contributions of HSBs are added to the corrected LSBs (a3, a2 and a1). This carry is calculated beforehand by a Carry Generator block using C1 and input bits p6 to p1. The logic implemented by Carry Generator is given by the equation below

C2 = C1' (p4 (p3+p2) +p3p5) + p6p3 + p4p3p1.....(6)

C2 is also added to result of the first 2-bit One Adder using another 2-bit One Adder and the final higher significant digit is obtain.

D. ADDER BLOCKS

Contribution of HSBs towards lower significant BCD digit is fixed and unique and is known once HSBs are known. We have implemented four distinct adder units which add only specified values to the inputs in parallel according to the contributions in Table 2. The different adder blocks, +1, +2, +3 and +4 (shown in Figure 5 to 8) add 001, 010, 011 and 100 to the input bits respectively.

Adder blocks take the corrected LSBs (a3, a2, a1) as inputs and add specific numbers to them.The appropriate result is then obtained through a multiplexer whose selection bits are p6, p5 and p4(HSBs). The result from the multiplexer is then fed to BCD Correction block which takes C2 as input to decide whether correction has to be done or not. The results obtained from the BCD Correction block are z3, z2 and z1 which, along with z0, form the final lower significant BCD digit.

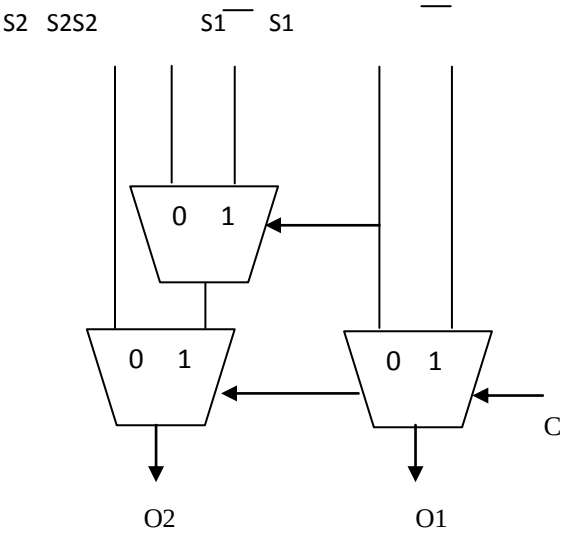

Fig4 : Two-bit adder

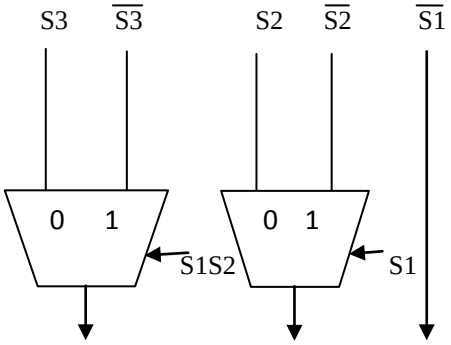

Fig5: +1 Adder block

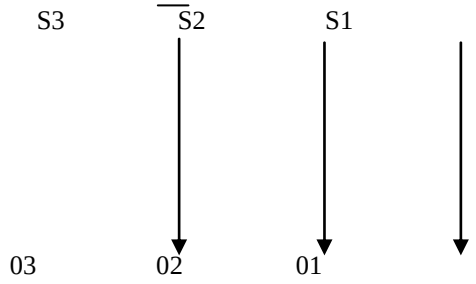

Fig8 : + 4 Adder block

VI.RESULTS

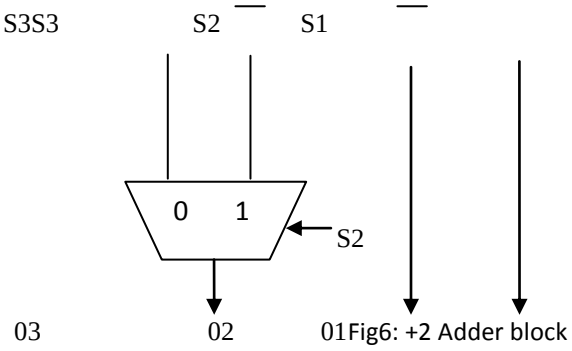

Fig6: +2 Adder block

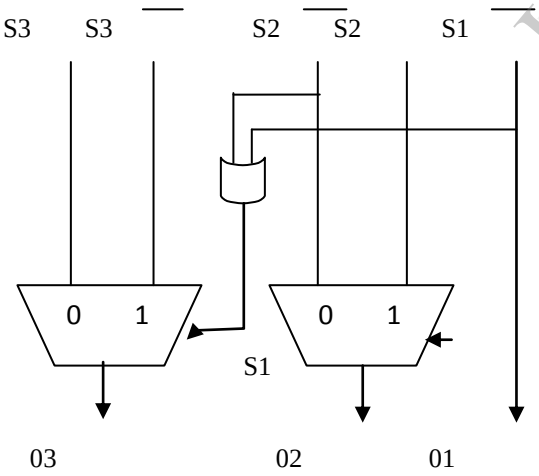

Fig7: +3 Adder block

Device utilization	Numbers	Utilization in %
1. Number of Slices	23	2% [23/768]
2. Number of 4 input LUTs	41	2% [41/1536]
3. Number of IOs	16	12% [16/124]
4. Number of bonded IOBs	16	12% [16/124]
5. Delay	22.148ns (11.424ns logic, 10.724ns route)	
6. Total memory usage	178672 Kbytes	

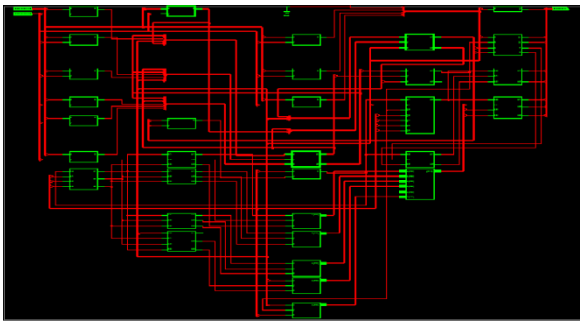

Fig :RTL Schematic

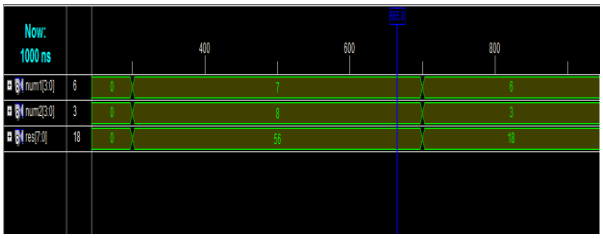

Fig: simulation result

VII.CONCLUSION

This paperpresents architecture for binary to BCD conversion used in decimal multiplication. The proposed converter is flexible and can be plugged into any homogeneous multiplication architectures to achieve better performance irrespective of the method used to generate binary partial products.

VIII.FUTURE ENHANCEMENT

The rapid advances in very large scale integration (VLSI) technology, semi- and fully parallel hardware decimal multiplication units are expected to evolve soon. The dominant representation for decimal digits is the binary-coded decimal (BCD) encoding. The BCD-digit multiplier can serve as the key building block of a decimal multiplier, irrespective of the degree of parallelism .Therefore decimal multiplier can be switched to VLSI technology

REFERENCE

[1].A high performance binary to BCD Converter for Decimal multiplication by Jairaj Bhattacharya , Aman gupta, Anushul singhCenter for VLSI and Embedded system Technology

[2].Binary-coded decimal digit multipliers Jaberipur, G.; Kaivani, A. Computers and Digital Techniques,

[3].early estimation of delay in binary to bcdconvertord. jothsnanarsimham& p. lakshmisarojini.

[4].Erle, M.A.; Schwarz, E.M.; Schulte, M.J., "Decimal multiplication with efficient partial product generation," 17th IEEE Symposium on Computer Arithmetic.

[5]A New Family of High–Performance Parallel Decimal Multipliers by Paolo MontuschiPolitecnico di Torino Dept. of Computer Engineering.

[6]Decimal Multiplication With Efficient Partial Product Generation by Mark A. Erle, Eric M. Schwarz.

Mr.Sadashiv R. Badiger is pursuing his Final year of post graduate studies in the Department of Electronics and Communication Engineering at SDMCET, Dharwad. His research interests include VLSI, Logic Design and Image processing.

PROF. BAIRU K. SAPTALAKAR is working as Assistant Professor in the Department of Electronics and Communication Engineering at SDMCET, Dharwad. His research interests include VLSI and Embedded system, Image Processing.