Design of High Speed Excess-3 Adder Using LUT

Mallireddy Sai Deepika, M. Naresh Babu

Abstract— **Processing of decimal numbers using binary system tends to be costly in terms of area and speed. To realize decimal operations efficiently an improved approach to implement decimal additions is proposed which is based on 6-input LUTs and fast carry chains. A new architecture is proposed with emphasis on critical path delay reduction. The adder architecture is implemented on Xilinx Virtex-6 FPGA for operand sizes from 2 to 18 digits. the design has outperformed other approaches in terms of area and delay. On average, the delay reduction is 13.1% and LUT saving is 28.9% compared to a conventional BCD adder. excess-3 addition is implemented using this system.**

Index Terms— BCD Adders, critical path delay reduction, fast carry chain, 6-input LUT, xilinx virtex-6 FPGA.

I. INTRODUCTION

Decimal computations are required in various applications, such as internet, industrial control, financial and commercial systems. Recently there is an increasing demand for efficient hardware realizations required in these applications. This has also led to the specification revision of the IEEE-754-2008 standard for floating-point arithmetic to incorporate the decimal format [1-2].

As in any hardware realization of real time systems, there is always a requirement to achieve high performance at a low cost. However, decimal arithmetic architectures and the hardware realizations, particularly, in Field Programmable Gate Arrays (FPGAs) have not been fully tackled in the literature. Therefore, efficient methods for the implementation of decimal operations are receiving more attention from hardware designers.

In decimal computation, the most common operation is addition. Earlier decimal adders were designed at the gate level targeting ASICs [3-6]. Binary-Coded-Decimal (BCD) number representation was used in these designs. Some schemes utilized in binary adders were also employed in these BCD additions. In [3], a reduced delay BCD adder was proposed. This approach improved the delay of BCD addition by increasing parallelism. Two 4-bit binary adders, a carry circuit, one AND gate, and one OR gate were used in the critical-path of the adder. In [5], a BCD adder was realized using reversible logic gates. Carry Look-Ahead scheme was employed to speed up the performance. The author in [6] proposed a multi-operand parallel decimal adder, which involved binary to decimal conversion in order to obtain BCD result. The conversion allows for an easy alignment of the sums of adjacent columns.

With the advancement in FPGA technology, the efficiency of the architecture, and availability of various hardware resources, decimal arithmetic can be implemented with high degree of efficiency. In [7] decimal adders/subtractors were proposed based on the use of Look Up Tables (LUTs) in FPGAs. In [8] a multi-operand decimal adder trees were presented and optimized based on the 6-input LUTs with the fast carry chains. Carry-ripple BCD adders were used in the adder tree, which led to an increase in the critical path delay.

In this paper, an improved carry-ripple BCD adder is presented targeting critical path delay reduction. When implemented into Xilinx' Virtex-6 FPGA, we achieved both speed improvement and area reduction.

The organization of this paper is as follows. Section 2 introduces some existing decimal adders that were used in this paper for comparison purpose. The improved BCD adder approach is presented in Section 3. In Section 4, the implementations and comparison of results are described, and the conclusions are given in the last section.

II. RELATED BCD ADDITIONS

A. Conventional BCD Adder

In a conventional 1-digit BCD adder, two BCD operands are added as binary numbers using a binary adder, and then the binary result is converted to BCD number. To perform the binary to BCD conversion, correction logic and another binary adder are required. Fig. 1 is the block diagram of the conventional 1-digit BCD adder.

Manuscript received Aug 18, 2014

M. Sai Deepika, P.G.Student scholar M.Tech (VLSI) ECE Department Sree vidyanikethan engineering college (Autonomous)

M. Naresh Babu, Assistant Professor in sree Vidyanikethan Engineering College, Tirupathi, India

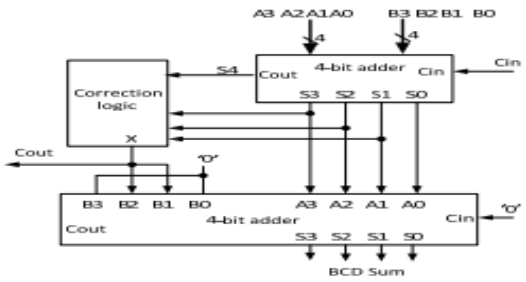


Fig 1. Conventional 1-digit BCD adder.

The correction logic function in Fig. 1 is expressed as:

$C_{out} = X = S_4 + S_3S_2 + S_3S_1$

The sum of the first binary adder is added to (0110)2 when the sum is greater than '9' to generate the correct BCD result.

B. Double-Dibble BCD Adder

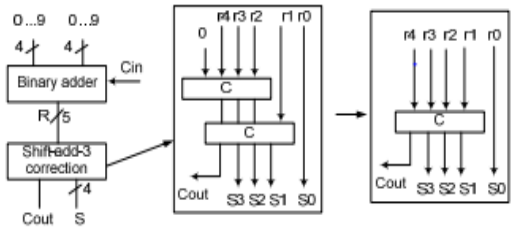
The Double-Dabble (DD) BCD adder uses the Double-Dabble Binary to BCD Conversion algorithm[9] to convert the binary sum to BCD number. This algorithm shifts the binary result one bit left at once, and then compares with '4'. If the value of the shifted bits is great than '4', '3' is added to the shifted bits; otherwise, continue to shift one bit left. Suppose the shifted bits are called "Shifted Unit" (SU), the algorithm is presented as:

if (SU>4) then
SU=SU+3;
else shift left;

It is clear that this algorithm is not efficient for the conversion of large size binary numbers because the number of corrections is based on the number of binary bits. However, for a 1-digit adder operation, we have simplified the design as shown in Fig. 2, where the block C performs the function of adding-3 correction. Originally, by using a binary adder, the sum of two BCD operands is a 5-bit binary number. To convert this 5-bit binary result to a BCD number, two adding-3 correction blocks are required, as shown in Fig. 2 (b) [9]. However, since the maximum number of the binary result to be corrected is

$R = [r_4r_3r_2r_1r_0] = A + B + Cin$
$= 19$
$= (10011)2$

then the maximum value of these first three shifted bits, [r4r3r2] is not greater than 4. Thus, the top adding-3 correction block in Fig. 2 (b) is not necessary. Then, continue to shift left 1-bit into the "Shifted Unit", which means shifting r1 into SU. Hence the four bits, [r4r3r2r1] are used to compare with 4. This will lead to an improved DD conversion architecture, which requires only one adding-3 correction block, as shown in Fig. 2 (c). The function of the adding-3 correction block, named as C, is captured in Table 1.



(a)DD BCD adder (b)Original correction (c)Simplified correction

Fig 2. DD 1-digit BCD adder.

N	r_4 r_3 r_2 r_1	C_{out} S_3 S_2 S_1	Note
0	0 0 0 0	0 0 0 0	
1	0 0 0 1	0 0 0 1	
2	0 0 1 0	0 0 1 0	
3	0 0 1 1	0 0 1 1	
4	0 1 0 0	0 1 0 0	
5	0 1 0 1	1 0 0 0	add-3
6	0 1 1 0	1 0 0 1	add-3
7	0 1 1 1	1 0 1 0	add-3
8	1 0 0 0	1 0 1 1	add-3
9	1 0 0 1	1 1 0 0	add-3
10	1 0 1 0	x x x x	
	.....	.....	
15	1 1 1 1	x x x x	

Table I . Truth table of the correction block

According to the Truth Table, the outputs of the adding-3 correction are expressed as:

$C_{out} = r_{4+}r_3r_{1+}r_3r_2$
$S_3 = r_4r_{1+}r_3r_2\bar{r}_1$
$S_2 = r_4\bar{r}_1+r_3\bar{r}_2+r_2r_1$
$S_1 = r_4\bar{r}_1+r_4\bar{r}_3r_1+r_3r_2\bar{r}_1$

C. 6-LUT-based carry-ripple BCD adder

In newer products of FPGAs, 6-input LUTs are available to be used and combined with fast carry chain. Fig. 3 shows this structure used to implement 1-bit binary adder. By cascading this 1-bit binary adder, an n-bit carry-ripple binary adder can be realized.

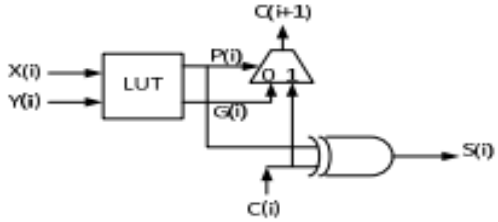


Fig 3. Architecture of FPGA for 1-bit binary adder.

Based on this configuration, a BCD carry-ripple adder has been proposed in [8]. The carry-ripple BCD adder uses the 6-input LUTs to add the most 3 significant bits of the input operands, and correct the result by adding-3 to the sum of the 3-bit addition. In this method, the correction is performed when the sum is equal to or greater than 4. Then, the fast carry chain and XOR gates are used to compute the two least significant bits and the carries. The structure is shown in Fig. 4. Since in this case the sum of the BCD adder, (1000)2 or (1001)2, has been added to (0110)2, the method allowed the 4-bit combinations, (1110)2 and (1111)2, to be valid representations of decimal values '8' and '9', respectively. To correct the final result, a post correction has to be performed.
Thus, the final output of the BCD adder is expressed as:

$$S = Z_3 \times 8 + Z_2 \overline{Z_3} \times 4 + Z_1 \overline{Z_3} \times 2 + Z_0$$

In this approach, the critical path delay is propagated from Cin to Cout, which is equivalent to 4 multiplexer delays.

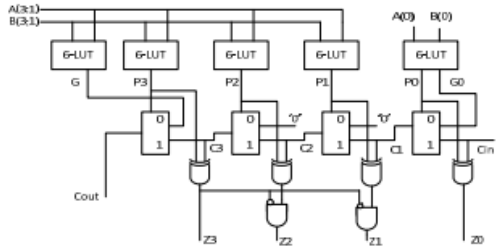


Fig 4. BCD 1-digit adder.

III. PROPOSED BCD ADDER

The proposed BCD adder is also based on 6-input LUTs and the fast carry chains in FPGAs. Assume the input operands of the adder are A and B in BCD format. To use the 6-input LUTs, each of the two input operands is decomposed into two parts:

$$A = [a_3a_2a_1a_0] = (a_3a_2a_1) \times 2 + a_0 = A_1 \times 2 + a_0$$

$$B = [b_3b_2b_1b_0] = (b_3b_2b_1) \times 2 + b_0 = B_1 \times 2 + b_0$$

The output of the BCD adder, named as (C_{out} $S_3S_2S_1S_0$), is presented as:

$$[CoutS_3S_2S_1S_0] = A + B + C_{in} = [A_1 \times 2 + a_0] + [B_1 \times 2 + b_0] + C_{in} = [(a_3a_2a_1) + (b_3b_2b_1)] \times 2 + [a_0 + b_0 + c_{in}]$$

In (4), the expression in the first part represents a 3-bit adder, and the expression in the second part is a full adder. Since the input operands in (4) are BCD numbers, the maximum value of the operands, (a3a2a1) or (b3b2b1) is (100)2. To achieve a BCD output, an adding-3 correction is performed based on the sum of (A1+B1) and the carry of the full adder, named as C1. Thus, the output of the BCD adder is expressed as:

$$[C_{out}S_3S_2S_1S_0] = (A+B+6) \text{ if } A_1+B_1 \geq 5 \text{ and } C_1 = \Phi$$
$$= A+B+6 \text{ if } A_1+B_1=4 \text{ and } C_1=1$$
$$= A+B \text{ if } A_1+B_1=4 \text{ and } C_1=0$$
$$= A+B \text{ if } A_1+B_1<4 \text{ and } C_1=\Phi$$

where adding-6 to (A+B) is the same as adding-3 to (A1+B1). According to (5), two scenarios can be considered separately. One is to design the 3-bit adder with the adding-3 correction without taking into consideration of the carry of the full adder; and the other one is to add the carry of the full adder for the final result of the BCD adder.

First, let's consider the 3-bit adder, (A1+B1). If the sum of (A1+B1) is equal to or greater than 5 while the carry of the full adder is '1' or '0', the final result of the BCD adder is equal to or greater than (101)2×2=5×2=10, and the adding-3 correction is required to be performed to the sum of (A1+B1).Otherwise, we do not perform the adding-3 correction to the 3-bit adder in this case. This 3-bit adder and the correction are merged together as a 6-input function, and implemented using a 6-input LUT. Fig. 5 illustrates this architecture, and Table 2 is the Truth Table of the 3-bit adder with the adding-3 function.

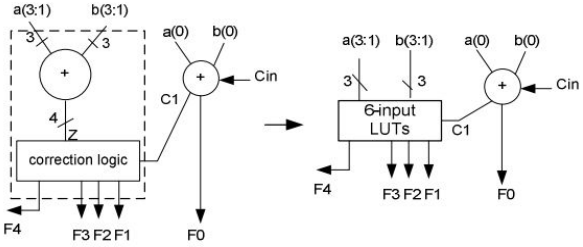


Fig 5. 3-bit adder with the adding-3 correction using 6-input LUTs.

(a_3 a_2 a_1) + (b_3 b_2 b_1)	F_4 F_3 F_2 F_1	Note
0 0 0 0 0 0	0 0 0 0	
0 0 0 0 0 1	0 0 0 1	
0 0 0 0 1 0	0 0 1 0	
0 0 0 0 1 1	0 0 1 1	
0 0 0 1 0 0	0 1 0 0	
0 0 1 0 0 0	0 0 0 1	
0 0 1 0 0 1	0 0 1 0	
.....	.....	
1 0 0 0 0 0	0 1 0 0	add-3
1 0 0 0 0 1	1 0 0 0	add-3
1 0 0 0 1 0	1 0 0 1	add-3

1 0 0 0 1 1 1 0 0 1 0 0	1 0 1 0 1 0 1 1	add-3

Table II. Truth table of 3-bit adder with the adding-3 corrections.

Now, let's take the C1 into account. If C1=0, there is no change to the result of the 3-bit adder. However, if C1=1, the carry has to be added to the sum of the 3-bit adder. This addition is realized by an exclusive OR gate and one multiplexer at each of the outputs of the 3-bit adder. Moreover, if C1=1 and the sum of (A1+B1) is F4F3F2F1=(0100)2=4, the result of the BCD adder should be equal to:

$$[C_{out}S_3S_2S_1S_0] = (F_4F_3F_2F_1+C_1)\times 2+F_0$$
$$=(0100+1)\times 2+F_0$$
$$=(101\ F_0)_2$$
$$=(1\ 000\ F_0)_{BCD}$$

In this case, the carry of the BCD adder, Cout is the same as the carry of the full adder C1=1 and the sum of the BCD adder has to be forced to 0 at the bit positions of S3 and S1.

Considering all the scenarios mentioned above, the final carry of the BCD adder is equal to F4 when (A1+B1)≥5 (in this case F3=0 after adding-3 correction), and equal to C1 when (A1+B1)≤4 (in this case F3=1). Thus, one multiplexer can be used to generate the final carry of the BCD adder. Fig. 6 shows the completed design for the improved BCD adder. The most left-side multiplexer is used to select either the carry of the 3-bit adder F4 for (A1+B1)≥5 and (A1+B1)<4, or the carry of the full adder C1 for (A1+B1)=4. To force the sum of the BCD adder under the condition of (A1+B1) = (F3F2F1) = (100)2 with C1=1, one AND gate with 1-input inverted is connected in the bit positions of S3 or S1.

The improvement of our architecture shown in Fig. 6 over the one proposed in [8] shown in Fig.4 is that our proposed approach has bypassed two multiplexer delays in the critical path of the BCD adder, hence reduced the carry-propagation delay. This has significant impact on the performance of large size BCD adders and multipliers.

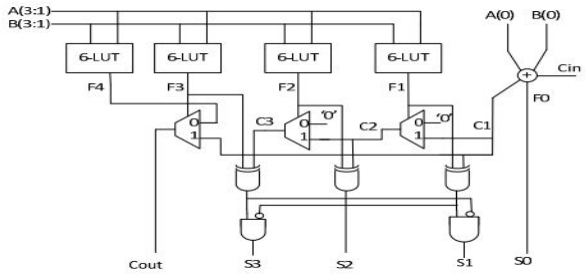


Fig 6. Improved BCD 1-digit adder.

IV. SIMULATION RESULTS

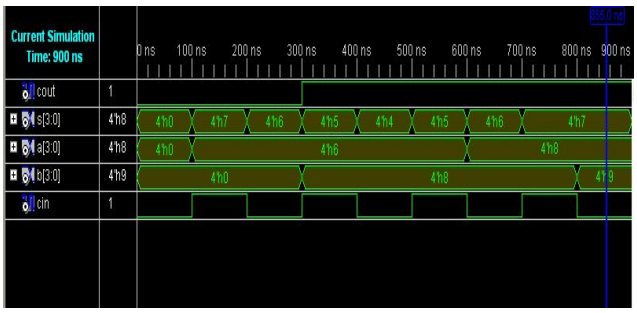


Fig 7: Conventional BCD Adder

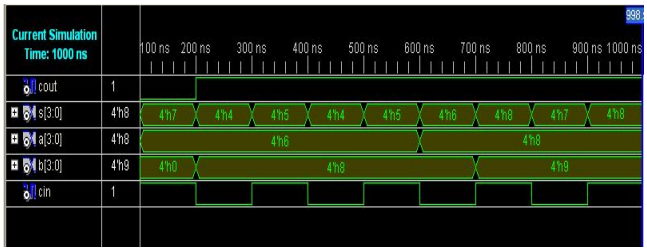


Fig 8: Double-Dibble BCD Adder

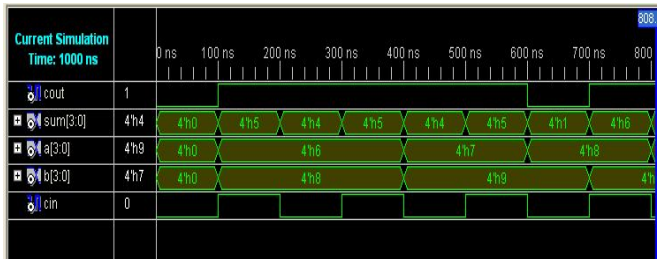


Fig 9: BCD 1-digit Adder

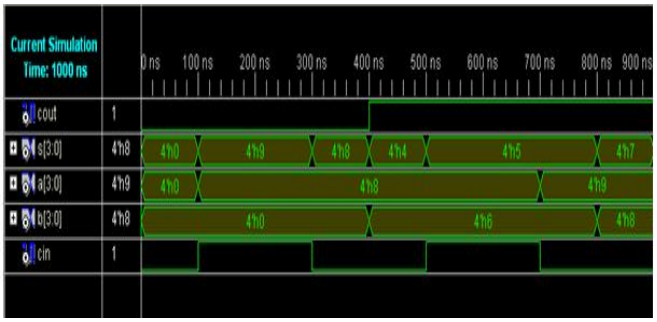


Fig 10: Improved BCD 1-digit Adder

Comparision Table:

Types of BCD adders	Conventional BCD adder	Double-Dibble BCD adder	BCD 1-digit adder	Improved 1-digit BCD adder
Number of LUTs	12	12	12	10
Delay(ns)	11.315ns	10.227ns	10.183ns	9.179ns

Ex-3 addition using BCD adders:

Example1:

		using excess-3 addition
4 - 0100	4+3= 7	0111
3 - 0011	3+3= 6	0110
0111		1101
7		13

Example 2 :

		using excess-3 addition
7 - 0111	7+3=10	1010
6 – 0110	6+3= 9	1001
1101		1 0011
	add 6->	0110
13		1 1001
		1 9

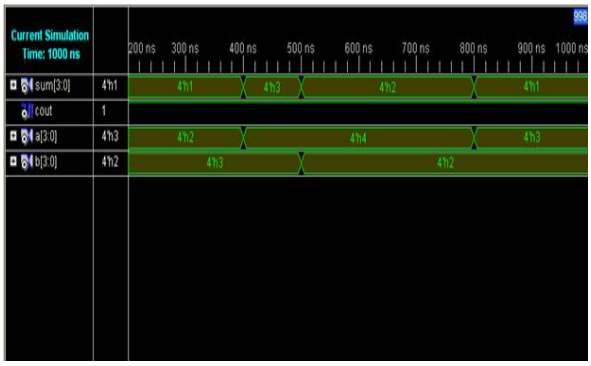


Fig 11: Simulation result using Excess-3 addition.

Comparision Table:

Adder Types	Conventional BCD adder using ex-3 addition	Double-Dibble BCD adder using ex-3 addition	Improved 1-digit BCD adder using ex-3 addition
No.of LUT's	22	22	19
Delay(ns)	13.315ns	12.195ns	11.768ns

CONCLUSIONS

This paper presented an improved BCD adder based on newer Xilinx FPGA architectures. The proposed BCD adder approach efficiently mapped the decimal addition function onto the 6-input LUTs and the fast carry chains in FPGAs. The critical path of the adder has been minimized by bypassing two multiplexers from incoming carry to outgoing carry in the 1-digit BCD adder. The implementation of the proposed approach has resulted in improvements in terms of delay reduction and savings in the number of LUTs. An excess-3 addition is implemented using LUT's which resulted in savings in number of LUT's and delay reduction.

REFERENCES

[1][1] IEEE Computer Society, "IEEE 754-2008 Standard for Floating-Point Arithmetic," Aug.2008.at:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4610935.

[2]M. F. Cowlishaw, "Decimal Floating –Point: Algorism for Computers," 16th IEEE Symposium on Computer Arithmetic, June 2003, page(s): 104-111.

[3]A.A. Bayrakci and A. Akkas, "Reduced Delay BCD Adder," IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP) 2007, page(s): 266-271.

[4]O. Al-Khaleel, Z. Al-Qudah, M. Al-Khaleel, C.A. Papachristou and F.G. Wolff, "Fast and compact binary-to-BCD conversion circuits for decimal multiplication," 2011 IEEE 29th International Conference on Computer Design (ICCD), Oct. 2011, page(s): 226 – 231.

[5]X. Susan Christina, M. Sangeetha Justine, K. Rekha, U. Subha and R. Sumathi, "Realization of BCD adder using Reversible Logic," International Journal of computer theory and engineering, Vol.2, No. 3, June 2010, page(s): 333-337.

[6]L. Dadda, "Multi Operand Parallel Decimal Adders: a mixed Binary and BCD Approach," IEEE Transactions on Computers, vol. 56, Oct. 2007, page(s): 1320–1328.

[7]M. Vazquez, G. Sutter, G. Bioul, J.P. Deschamps, "Decimal Adders/Subtractors in FPGA: Efficient 6-input LUT Implementations," International Conference on Reconfigurable Computing and FPGAs (ReConFig '09), Dec. 2009, page(s): 42 – 47.

[8]Alvaro Vazquez1 and Florent de Dinechin, "Multi-operand Decimal Adder Trees for FPGAs," Inria-00526327, vol. 1, at: http://hal.inria.fr/inria-00526327.

[9]Binary-to-BCD Converter: "Double-Dabble Binary-to-BCD Conversion Algorithm," at: http://edda.csie.dyu.edu.tw/course/fpga/Binary2BCD.pdf.

[10]Xilinx Inc., "Virtex-6 User Guide," UG364 (v1.2), Feb. 2012, at: http://www.xilinx.com/support/documentation/user_guides/ug364.pdf