# Design and Implementation of ALU using Redundant Binary Signed Digit

Suma T.Hegde
VTU Extension centre
UTL Technologies
Bangalore, India

Dr.Siva Yellampalli
VTU Extension centre
UTL Technologies
Bangalore,India

Nandeesh R.
Marvel Soft
Rajajinagar
Bangalore,India

## ABSTRACT

In this paper we present the design of an Arithmetic Logic Unit (ALU) based on Redundant Binary signed Digit (RBSD) Number System. A redundant binary representation is a numeral system that uses more bits than needed to represent a single binary digit because of which most numbers have several representations. This unique feature of RBSD number system allows addition without using a typical carry. The RBSD ALU is designed using VHDL and its RTL view is generated by its FPGA implementation. The FPGA implementation is done in Xilinx ISE environment.

## Keywords

ALU, RBSD, FPGA, RTL, Xilinx, VHDL.nn

## 1. INTRODUCTION

Today's world requires faster processor for the computation purposes to meet the application demand of the digital systems. With the constant growth of computer applications in every field of engineering such as signal processing, communications and neural networks, fast arithmetic logic units (ALU) are increasingly required. The ALU of any processor perform many functions such as Addition, Subtraction, Multiplication, Division and Logical Comparison etc. These arithmetic operations produce carry propagation chains. The speed of operations depends on the implementation of arithmetic algorithms.

ALU can be designed using ripple carry or carry look ahead adder. But in case of ripple carry adder the delay will be more as the carry should be propagated entire bit width. So speed will be reduced. Carry look ahead adders are faster than ripple carry adders but the complexity of the circuitry increases as the number of bits increases. Use of non-conventional number systems in designing ALU is gaining attention in recent years because of their facility to provide carry free addition thus enhancing the achievable processing speed. For making the processing faster a carry free addition technique is adopted by using Redundant Binary Number System [1][2][8]. The property of carry propagation chain elimination tends to make the processing faster.

In this paper, the RBSD based arithmetic and logical unit is designed using VHDL and its RTL view is generated by its FPGA implementation. The FPGA Implementation is done in Xilinx ISE environment. The simulation is done in Model Sim environment.

The rest of this paper is organized as follows: In Section 2, we discuss background information on a novel Redundant Binary Signed Digit (RBSD) number system. In Section 3 the design of RBSD ALU and its FPGA Implementation is discussed and it has been implemented in VHDL. The results are verified and presented in the form of waveforms. Comparison and conclusions part are given in the last Section.

## 2. CARRY FREE ADDITION USING REDUNDANT BINARY SIGNED DIGIT

The redundant binary representation (RBR) is a numeral system that uses more bits than needed to represent a single binary digit so that each number will have several representations. RBR is a place-value notation system. In RBR digit set will have more digits than the radix and digits are pairs of bits, that is, for every place RBR uses a pair of bits. RBR is unlike usual binary numerical systems, including two's complement, which use single bit for each digit.

The value represented by an RBR digit can be found using a translation table as shown in Table 1. This table indicates the mathematical value of each possible pair of bits. As in conventional binary representation, the integer value of a given representation is a weighted sum of the values of the digits. The weight starts at 1 for the rightmost position and goes up by a factor of 2 for each next position. Usually, RBR allows negative values. There is no single sign bit that tells if a RBR represented number is positive or negative. Most integers have several possible representations in an RBR. An integer value can be converted back from RBR using the following formula, where 'n' is the number of digit and $d_k$ is the interpreted value of the $k^{th}$ digit, where 'k' starts at 0 at the right most position [6]:

$$\sum_{k=0}^{n-1} d_k 2^k$$

**Table 1.Translation table for Redundant Binary Signed Digit**

| **Digits** | **Interpreted value** |
|---|---|
| 00 | -1 |
| 01 | 0 |
| 10 | 0 |
| 11 | 1 |

The redundant binary signed digit number (RBSD) representation makes it possible to perform addition with carry propagation chains limited to a single digit position and has been used to speed up the arithmetic operations. In order to cope with the problem of carry propagation the most appropriate approach is elimination of carry propagation. If the numbers can be represented in such a manner that addition does not require carry propagation then the addition is said to be carry-free or carry eliminated addition, In case of RBSD all digit additions can be done simultaneously. The application of interval arithmetic in which carry propagates only one position and no additional carry is generated; makes possible carry free addition [3][4].

The RBSD carry propagation free addition is performed in two steps [5]:

Step 1: In order to eliminate carry, at each position the transfer digit $t_i$ and interim sum digit $w_i$ are determined according to Table 2. If $X_i$ and $Y_i$ are the two operands then the relationship between $X_i$, $Y_i$, $t_i$ and $w_i$ is mathematically represented as

$X_i+ Y_i = 2t_i+w_i$

Step 2: The incoming transfer digit is added with the interim sum to obtain the final sum digit with no new transfer digit. This step is mathematically represented as

$S_i= w_i+t_i$

Where $w_i$ is interim sum, $t_i$ is transfer digit and $S_i$ is sum digit.

**Table 2. Transfer Digit and Interim Sum for Redundant Binary Radix 2**

| $X_i$ $Y_i$ | $X_i+Y_i$ | $t_i$ | $w_i$ |
|---|---|---|---|
| -1 -1 | -2 | -1 | 0 |
| -1 0<br>0 -1 | -1 | 0<br>-1 | -1<br>1 |
| -1 1<br>1 -1 | 0<br>0 | 0<br>0 | 0<br>0 |
| 0 0 | 0 | 0 | 0 |
| 0 1<br>1 0 | 1<br>1 | 0<br>1 | -1<br>-1 |
| 1 1 | 2 | 1 | 0 |

Example: Carry –Free addition using redundant signed radix 2

In conventional binary number system radix 2 number digit set contains 0, 1. The number of digits equal to radix.

Example : Number 6 can be represented in binary as below

0 1 1 0

Number 4 can be represented as below

0 1 0 0

In case of redundant Signed radix 2 number digit set contains {-1,0,1}. The number of digits present in the digit set will be more than the radix. So each number can be represented in many ways.

Example: Number 6 in decimal can be represented in redundant binery as follows.

0 1 1 0 ------- 6

1 0 -1 0 ------- 6

Number 4 can be represented in redundant binary as follows

0 1 0 0 ---------- 4
1 -1 0 0 ---------- 4

In case of conventional binary addition there will be carry propagation. Carry will be propagated till the end.

Addition in case of RBSD is carry free.

In case of RBSD addition the two operands will be added to get the position sum($p_i$). Then the position sum will be divided into interim sum($w_i$) and transfer digit($t_i$). Then interim sum and transfer digit is added to get the final sum.

In case of conventional binary there is no such steps .

Let the two operanads be $X_i$ = 0 1 1 0 and $Y_i$ = 0 1 0 1

$X_i$ = 0 1 1 0 ------- $(6)_{10}$

$Y_i$ = 0 1 0 1 ------- $(5)_{10}$

The final sum should be $(11)_{10}$

Adding these two numbers using binary.

0 1 1 0 ---- $X_i$

0 1 0 1 ---- $Y_i$

-------------------

1 0 1 1 ---- $s_i$

The steps involved in adding the two operands using RBSD are as below.

Step 1. Both the operands are added to get the position sum, $p_i$.

Step 2. Interim sum $w_i$ and transfer digit $t_i$ are determined from Table 2. The interim sum and transfer digit are selected in such a way that they should be within the digit set

i.e -1 0,1 and after adding interim sum and transfer digit the final sum also should be in the selected digit set.

Step 3 Transfer digit is added with the interim sum to get final sum $s_i$. ---- 0 1 0 1 1 --- $(11)_{10}$

```
     0   1  1   0              xi

  +   0   1   0   1            yi
  --------------------------

         0    2   1   1          pi

     0    0    1   1           wi
   /    /   /    /
 0    1    0    0               ti
-----------------------------
0       1     0      1      1        si
```

## 3. DESIGN AND IMPLEMENTATION OF RBSD BASED ALU

### 3.1 Design of one digit RBSD based ALU

The one digit ALU is designed by VHDL and is shown in Figure 1.

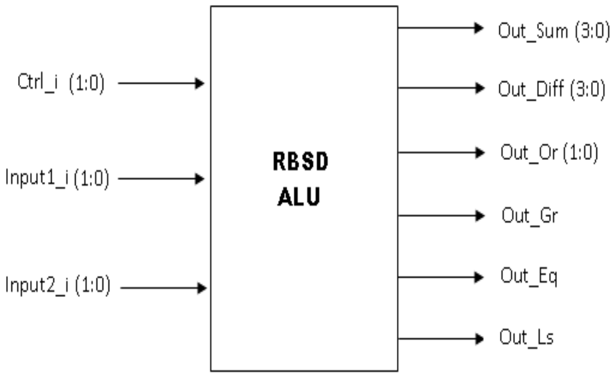


**Figure 1: One digit RBSD ALU pin out**

One digit (two bits) ALU has two input bit vectors Input1_i and Input2_i having width of two. It has one control input Ctrl_i also having width of two. Control input is used to select the required arithmetic operation. The designed ALU has adder, subtractor, OR gate and comparator. If the control is '00' adder unit will be activated. If it is '01' subtractor unit will be activated. If it is '10' OR gate will be activated. If it is '11' comparator will be activated

One digit ALU contains six outputs. Out_Sum is the output of the adder having the width of four. Out_Diff is the output of the subtractor having the width of four. Out_Or is the output of the OR gate having the width of two. Out_Gr, Out_Eq, Out_Ls, are the three outputs of comparator.

The block diagram of one digit RBSD ALU is as shown in Figure 2.

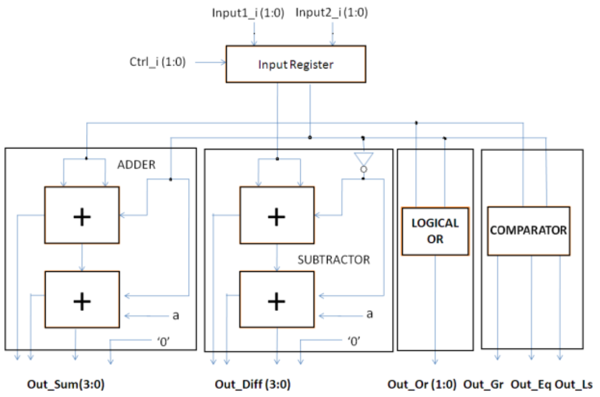


**Figure 2: Block Diagram of One Digit RBSD ALU**

The one digit RBSD ALU contains adder [9], subtractor, logical OR and comparator block. The adder unit contains two full adders. One digit RBSD means two bits . For the first full adder the inputs are two bits from the Input1_ I (1,0) and one bit from Input2_i (1,0), totally three inputs. The output of the first full adder ,one bit from Input2_i (1,0) are the inputs to another full adder. Output of 1 digit RBSD adder will be four bits.

The subtractor unit contains two full adders and one NOT gate. Here NOT gate is used to find the additive inverse of one of the operands. After finding the additive inverse the unit will work same as adder unit. Difference between two operands is found by adding the additive inverse of one of the operands to another.

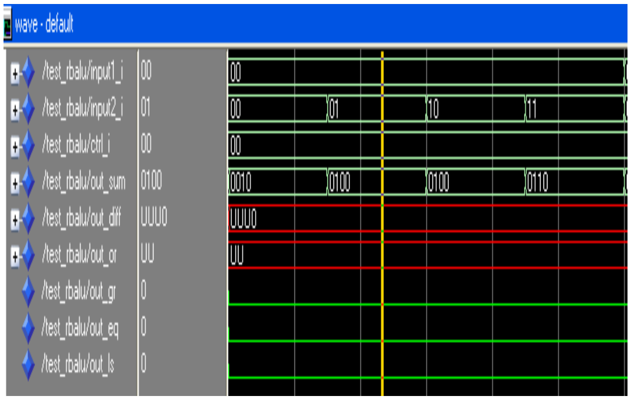
The comparator unit will have two inputs each input of two bits. It has three outputs. It will compare two inputs and depending on the status of the inputs one of the output will be at logic high state.

Simulation is done using ModelSim XE III 6.2g Simulator. The simulation results for one digit ALU are verified with its corresponding logic diagram and are shown below [7].

Figure 3 shows the simulation result for adder unit of one digit RBSD ALU. Input1_i = -1, Input2_i = -1,0,1

Input1 will be -1. As shown in table 1 redundant digit -1 is represented as 00

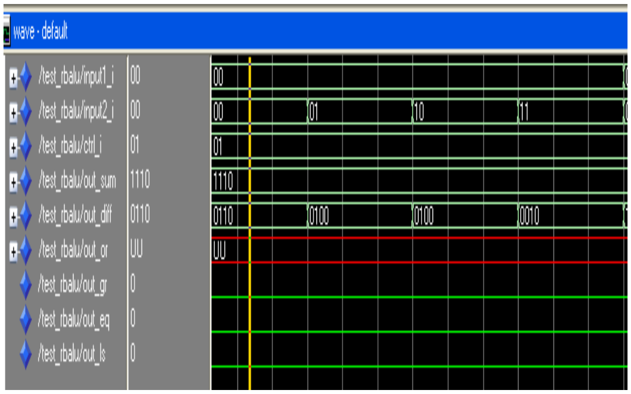
Input2 is (i) -1 (00) , (ii) 0 – (01 or 10) (iii) 1 – (11)



```
wave - default
/test_rbalu/input1_i    00     00
/test_rbalu/input2_i    01     00    01    10    11
/test_rbalu/ctrl_i      00     00
/test_rbalu/out_sum     0100   0010  0100  0100  0110
/test_rbalu/out_diff    UUU0   UUU0
/test_rbalu/out_or      UU     UU
/test_rbalu/out_gr      0
/test_rbalu/out_eq      0
/test_rbalu/out_ls      0
```

**Figure 3: Simulated Result for adder Input1_i=-1 Input2_i =-1, 0, 1**

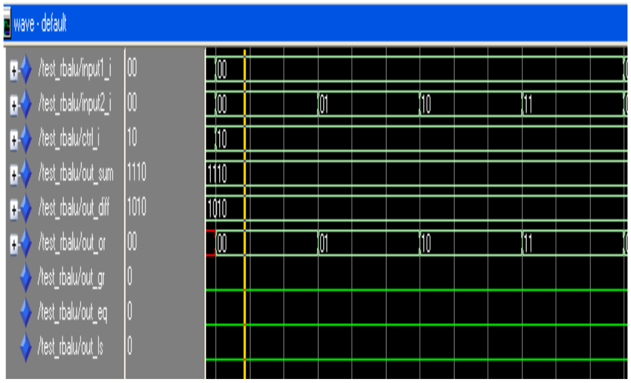
Figure 4 shows the simulation result for subtractor unit of one digit RBSD ALU. Input1_i = -1, Input2_i = -1, 0, 1



```
wave - default
/test_rbalu/input1_i    00     00
/test_rbalu/input2_i    00     00    01    10    11
/test_rbalu/ctrl_i      01     01
/test_rbalu/out_sum     1110   1110
/test_rbalu/out_diff    0110   0110  0100  0100  0010
/test_rbalu/out_or      UU     UU
/test_rbalu/out_gr      0
/test_rbalu/out_eq      0
/test_rbalu/out_ls      0
```

**Figure 4: Simulated Result for subtractor Input1_i=-1 Input2_i =-1, 0, 1**

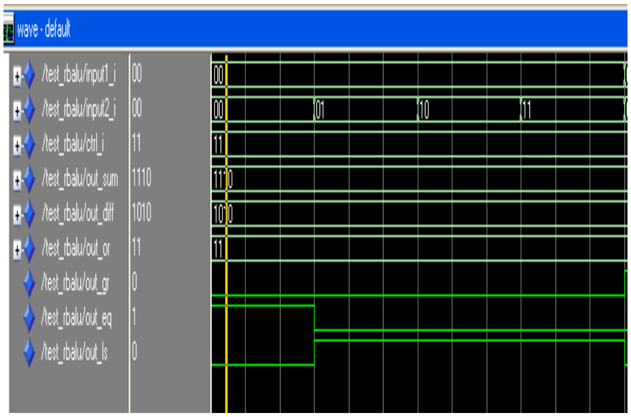
Figure 5 shows the simulation result for OR gate of one digit RBSD ALU. Input1_i = -1, Input2_i = -1, 0, 1



```
wave - default
/test_rbalu/input1_i    00     00
/test_rbalu/input2_i    00     00    01    10    11
/test_rbalu/ctrl_i      10     10
/test_rbalu/out_sum     1110   1110
/test_rbalu/out_diff    1010   1010
/test_rbalu/out_or      00     00    01    10    11
/test_rbalu/out_gr      0
/test_rbalu/out_eq      0
/test_rbalu/out_ls      0
```

**Figure 5: Simulated Result for OR gate Input1_i=-1 Input2_i =-1,0,1**

Figure 6 shows the simulation result for comparator of one digit RBSD ALU. Input1_i = -1, Input2_i = -1,0,1



```
wave - default
/test_rbalu/input1_i    00     00
/test_rbalu/input2_i    00     00    01    10    11
/test_rbalu/ctrl_i      11     11
/test_rbalu/out_sum     1110   1110
/test_rbalu/out_diff    1010   1010
/test_rbalu/out_or      11     11
/test_rbalu/out_gr      0
/test_rbalu/out_eq      1
/test_rbalu/out_ls      0
```

**Figure 6: Simulated Result for Comparator Input1_i=-1 Input2_i =-1, 0, 1**

## 3.2 Design of two digit RBSD based ALU

The two digit ALU is designed by VHDL[10] and is shown in Figure 7.

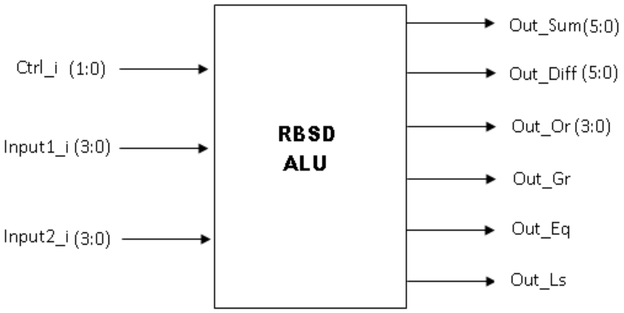


**Figure 7: Two digit RBSD ALU pin out**

Two digit (four bits) ALU has two input bit vectors Input1_i and Input2_i having width of four. It has one control input Ctrl_i also having width of two. Control input is used to select the required arithmetic operation. The designed ALU has adder, subtractor, OR gate and comparator. If the control is '00' adder unit will be activated. If it is '01' subtractor unit will be activated. If it is '10' OR gate will be activated. If it is '11' comparator will be activated

Two digit ALU contains six outputs. Out_Sum is the output of the adder having the width of six. Out_Diff is the output of the subtractor having the width of six. Out_Or is the output of the OR gate having the width of four. Out_Gr, Out_Eq, Out_Ls, are the three outputs of comparator.

The block diagram of two digit RBSD ALU is as shown in Figure 8.

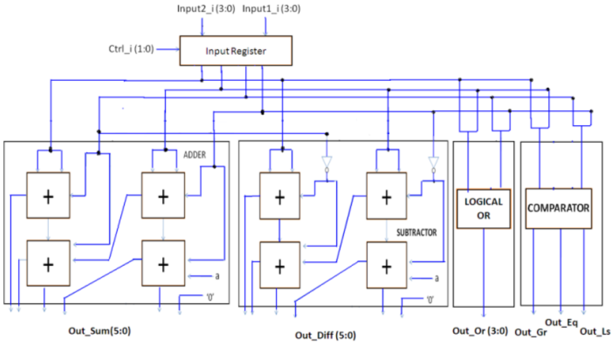


**Figure 8: Block Diagram of Two Digit RBSD ALU**

The two digit RBSD ALU contains adder, subtractor, logical OR and comparator block. Adder unit contains four full adders. Two digit RBSD ALU means four bit ALU. Because each RBSD digit is represented by two bits. The output of adder is six bits

Subtractor unit contains four full adders and two NOT gates. NOT gates are used to find the additive inverse of one of one of the operands. After finding the additive inverse it will be added to the other operand. The output of subtractor is six bits

The comparator unit will have two inputs each input of four bits. It has three outputs. It will compare two inputs and depending on the status of the inputs one of the output will be at logic high state.

Simulation is done using ModelSim XE III 6.2g Simulator. The simulation results for two digit ALU are verified with its corresponding logic diagram and are shown below.

Figure 9 shows the simulation result for adder unit of two digit RBSD ALU. Input1_i = -3, Input2_i =-3,-2,-1,0,1,2,3

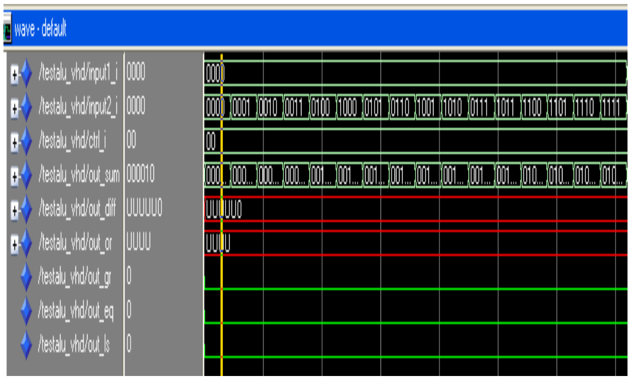


**Figure 9: Simulated Result for Adder of Two Digit ALU Input1_i=-3 Input2_i =-3,-2,-1, 0,1,2,3**

Figure 10 shows the simulation result for subtractor unit of two digit RBSD ALU. Input1_i = -3, Input2_i =-3,-2,-1, 0,1,2,3

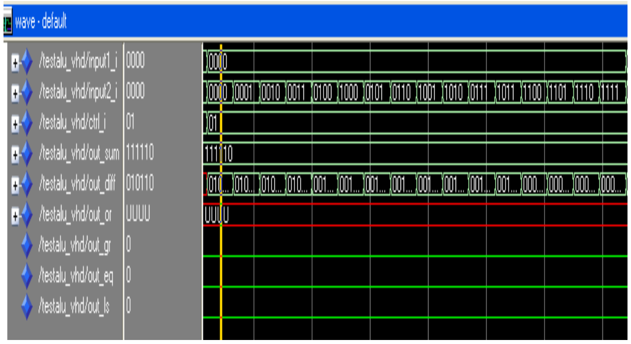


**Figure 10: Simulated Result for Subtractor of Two Digit ALU Input1_i=-3 Input2_i =-3,-2,-1,0,1,2,3**

## 4. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards success of this paper.

## 5. REFERENCES

[1] [Aviz 1961] A. Aviziens, (1961) "Signed Digit Number Representation for Fast parallel Arithmetic," IEEE Trans on Electro. Comp. Vol. EC-10, PP. 389-400.

[2] [Prahami 1990] B.Prahami, (1990) "Generalized signed digit number system,A unifying framework for redundant number representations," IEEE Trans. Computers vol.39, pp.89-98.

[3] Krishna Raj, Brijesh Kumar and Poornima Mittal, "FPGA Implementation and Mask Level CMOS Layout Design of Redundant Binary Signed Digit Comparator" IJCSNS International Journal of Computer Science and Network Security, Vol 9, No.9,pp 107 – 115 , September 2009

[4] G.F.Songster,“Negative base number representationsystems”IEEE Trans. Comput., EC- 12 No. 3, 274-277.

[5] Dhananjay S. Phatak and Israel Koren, “Hybrid Signed–Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounde Carry Propagation Chains”, IEEE Transactions on Computers, vol. 43, No. 8, pp 880-891, August 1994

[6] Louis-Philippe Lessard, “Fast Arithmetic on FPGA Using Redundant Binary Apparatus”, 2008

[7] Pai, Yu-Ting and Chen, Yu-Kumg, “The fastest carry lookahead adder, Electronic Design, Test and Applications”. Second IEEE International Workshop, 2004

[8] Chow , C. Y. Chow and J. E. Robertson, “Logical Design of a Redundant Binary Adder”, Proceedings of 4th Symposium on Computer arithmetic, pp. 109-115, 1978

[9] Samir palnitkar, Verilog HDL “A guide to digital design and Synthesis”,second edition 2003 Pearson education

.