High Speed Reconfigurable ALU Design for Radix ($2^n \pm m$)

Arindam Banerjee[1]*, Swapan Bhattacharyya[1], Arpan Deyasi[2]

[1]Department of ECE, JISCE, Kalyani, West Bengal, India

[2]Department of ECE, RCCIIT, Kolkata, West Bengal, India

*banerjee.arindam1@gmail.com

Abstract: High speed ALU design for ($2^n \pm m$) radix has been reported in this paper. The design has been achieved using the contemporary reconfigurable logic. Here n and m are any positive integer. So far the arithmetic circuit design is concerned; all the architectures have been shown in binary, ternary or quaternary logic. Here we have shown that the conventional logical operations like AND, OR, XOR etc. can be designed in any radix system. Our design follows a generic structure which can be implemented in any radix system. In the proposed design scheme ten arithmetic operations have been incorporated. The design has been verified using Xilinx ISE and implemented using Vertex-7 FPGA.

Keywords: ALU; radix; residue; FPGA; reconfigurable logic.

1 INTRODUCTION

ALU is the core component of all processors used in digital signal processing, image processing, artificial neural network, multimedia application etc [1-10]. From decades ago scientists and researchers are working on the performance improvement of the processors by modifying the ALU structure [2-7, 10]. Common practice for designing ALU is using binary radix system [2, 7], [9-10]. Few researchers have designed ALU in ternary and quaternary radix system [1, 3-6, 8]. But all the designs are dedicated for a particular radix system. The design scheme of the ALU changes with the change in radix system. Most of the design schemes reported so far are based on arithmetic operations but still now there is no design scheme which shows any logical operation.

Here our aim is to propose a design scheme for the ALU which can be designed in any radix system. We have proposed a common technique to design arithmetic unit as well as logical unit in any radix system.

2 BACKGROUND MATHEMATICS

In this paper, it has been shown that the logical operations can be performed in any radix system. In radix-2 system, we perform logical AND, OR, XOR, NOT, NAND and NOR operations. Here it has been shown that these gates can be implemented in any radix system based on some simple mathematics.

Let us start with AND gate. The truth table of the gate is given below.

Table 1. Truth table of the AND gate.

Input A	Input B	Output Y
0	0	0
0	1	0
1	0	0
1	1	1

Here also, it is obvious that Y is the maximum value of A and B. Therefore Y can be defined as follows.

By mere observation, it can be said that the output Y is the minimum value of A and B. Therefore Y can be defined as follows.

$$Y = A \text{ AND } B = \text{Min}(A,B) \quad (1)$$

ISSN: 2222-7059 (print), ISSN: 2222-7067 (online), http://www.aspbs.com/aiem
DOI: 10.7508/aiem.2016.02.003

Similarly, for an OR gate, the truth table is given below.

Table 2. Truth table of the OR gate.

Input A	Input B	Output Y
0	0	0
0	1	1
1	0	1
1	1	1

Here also, it is obvious that Y is the maximum value of A and B. Therefore Y can be defined as follows.

$$Y=A \text{ OR } B=\text{Max}(A,B) \quad (2)$$

Similarly, the NOT operation is defined as follows.

$$Y=\overline{A} = r-1-A \quad (3)$$

where, r is the radix. Then the definitions of all the logic gates can be modified as follows.

$$A \text{ AND } B=\text{Min}(A,B) \quad (4)$$

$$A \text{ OR } B=\text{Max}(A,B) \quad (5)$$

$$A \text{ NAND } B=r-1-\text{Min}(A,B) \quad (6)$$

$$A \text{ NOR } B=r-1-\text{Max}(A,B) \quad (7)$$

$$A \text{ XOR } B=\begin{cases} 0 & \text{for } A=B \\ r-1 & \text{for } A \neq B \end{cases} \quad (8)$$

$$A \text{ XNOR } B=\begin{cases} 0 & \text{for } A \neq B \\ r-1 & \text{for } A=B \end{cases} \quad (9)$$

Based on the above mentioned mathematics, the overall architecture for the ALU has been designed in any radix system.

3 ARCHITECTURAL DESCRIPTION

In this proposed ALU design we have incorporated ten arithmetic and logical operations. These operations are selected based on the control bus ($CTRL_i$) for i=0 to 3. Here it is obvious that the address index (i) has 4 values and therefore 2^4=16 operations can be selected from which 10 operations has been shown. The ten operations are given in Table III.

The operation of the circuit starts by comparing the two operands A and B. If A≥B, then the final borrow output of the subtractor 'Bo' is low else it is high. Now if '$CTRL_0$' is '0' then the MUX_2 output is equal to Min(A,B). Now if '$CTRL_0$' is '1' then the MUX_2 output is equal to Max(A,B).

The inversion operation is performed by the first (ADD/SUB) unit. If the control bus is "0010" then subtraction is performed and the $(ADD/SUB)_1$ output is [Radix-1-Min(A,B)] and else if it is "0011" then the $(ADD/SUB)_1$ output is [Radix-1-Max(A,B)].

To perform XOR or XNOR operation, at first the equality of A and B is checked. If A=B, only then OR array produces low output. Now if the control bus is "0100" then the MUX3 output is also low which the XOR operation is else if the control bus is "0101" then the MUX3 output is [Radix-1] which is the XNOR operation. But if A≠B, then the OR array produces high output. Now if the control bus is "0100" then the MUX3 output is [Radix+1] which is XOR operation else if the control bus is "0101" then the MUX3 output is also low which indicates XNOR operation.

Table 3. List of operations for the ALU.

$CTRL_3$	$CTRL_2$	$CTRL_1$	$CTRL_0$	OPERATION
0	0	0	0	AND
0	0	0	1	OR
0	0	1	0	NAND
0	0	1	1	NOR
0	1	0	0	XOR
0	1	0	1	XNOR
0	1	1	0	ADD
0	1	1	1	SUB
1	1	X	0	INCREMENT
1	1	X	1	DECREMENT

If the control bus is "011X" then AND1 gate passes the two operands A and B to the (ADD/SUB)2 module. CTRL0 selects the type of operation (addition / subtraction). If CTRL0 is low then addition operation is selected else subtraction operation performed. MUX5 produces the final result. Here the size of A and B are taken to be same and that is equal to $[\log_2 \text{Radix}]=[\log_2(2^n \pm m)]$.

Increment / Decrement operation can be performed on any of the two operands simultaneously but the output of these operations are not produced by the output databus 'Result'. There are two separate output buses entitled "INC/DEC A" and "INC/DEC B" which produce the output corresponding to the operand A and B respectively.

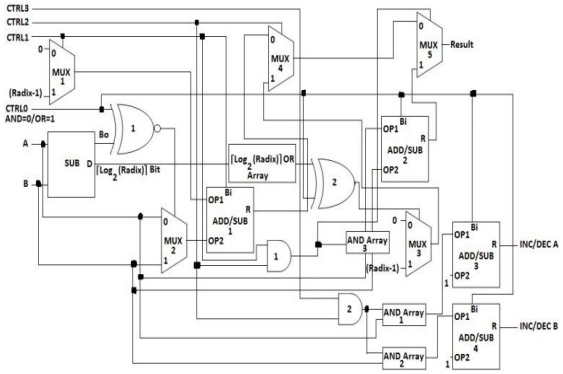

Fig. 1. Architecture for the proposed ALU.

The architecture has several modules:- (i) Subtractor (SUB), (ii) XOR gates, (iii) AND gates, (iv) Multiplexers (MUX), (v) Adder/ Subtractor

(ADD/SUB), (vi) OR array and (vii) AND array. Here XOR gate, AND gate, Multiplexer and subtractor are very common in designing digital circuits. So the main modules which are indispensable in the architecture are discussed below.

3.1 Adder/Subtractor

In adder/ subtractor, there is a control signal which selects the type of operation. In the adder/ subtractor modules shown in Figure 1, 'Bi' is the control signal and op_1 and op_2 are two operands. The governing equation for the adder/ subtractor is given below:

$$D_i = op_{1i} \oplus op_{2i} \oplus B_{oi} \quad (10)$$

$$Bo_{(i+1)} = (B_i \oplus op_{1i})op_{2i} + (B_i \oplus op_{1i} \oplus op_{2i})B_{oi} \quad (11)$$

From equations (10) and (11), it is obvious that if 'B_i' is low then addition is done else subtraction is done. Now for a k-bit number, the result Di can be expressed as $\sum_{i=0}^{K-1} D_i 2^i = \sum_{j=0}^{P-1} E_j r^j$ where $r=(2^n \pm m)$ is the radix. Here m is the residue of the radix 'r' over 2^n and $k=\log_2(2^n \pm m)=n$ if m is negative and (n+1) if m is positive. This is the radix adjustment scheme. Here D_i $(0 \le i < k)$ is in radix 2 whereas E_j $(0 \le j < P)$ is in radix 'r'. In this proposed scheme, it has also been assumed that the binary number D_i, if converted to E_j with respect to radix 'r', has P number of weights. The following algorithm describes the technique to add or subtract and also the radix adjustment scheme.

3.2 Algorithm for Addition/Subtraction

Input: Boolean ctrl;

Input: Boolean Ai and Bi for i=0 to $\log_2(2^n \pm m)-1$

Output: Boolean Yi for i=0 to $\log_2(2^n \pm m)-1$;

Variable: Boolean Si for i=0 to $\log_2(2^n \pm m)$

Variable: Integer D;

for(i=0; i< $\log_2(2^n \pm m)$; i++)

if(ctrl == '0') then $S_i = A_i + B_i$

else Si=A--Bi $S_i = A_i$ -;

end;

end;

D=decimalconvert(Si);

if (D≥(2n±m)

then D=D+m;

Yi=booleanconvert(D);

else Yi=booleanconvert(D);

end;

3.3 OR Array

In Figure 1, the length of the OR array is $\log_2(2^n \pm m)$ if m is negative and (n+1) if m is positive. Figure 2 shows the architecture for the OR array.

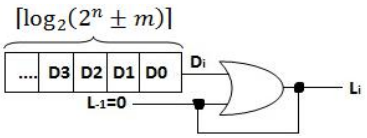

Fig. 2. Architecture for the OR array.

If all the bits of the array of size $2^n \pm m$ are '0' then the output is low else it is high.

3.4 AND Array

The AND array shown in Figure 1 has one control signal which controls the outputs to be set or reset. If the control signal is high then the input bit stream is directly passed to the output else the output is reset.

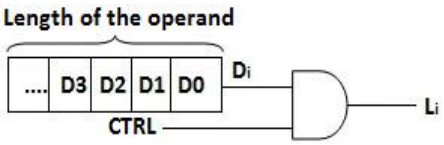

Fig. 3. Architecture for AND array.

3.5 Incrementer/Decrementer (INC/DEC)

As shown in Figure 1, incrementer/ decrementer module has been implemented using the adder/ subtractor module. Here operand1 (OP1) is incremented or decremented by '1' (OP2) based on the control signal 'Bi'.

Table 4. Area, delay and power estimation for different radix.

Radix	Area (No. of slice)	Delay (in ns)	Power (in mW)
6	25	3.149	0.82
8	30	3.237	0.96
12	36	3.367	1.23
16	45	3.473	1.45
20	49	3.823	1.63
32	55	4.083	2.89
40	59	4.236	3.08
48	64	4.531	3.65
64	71	5.472	5.46

4 RESULT ANALYSIS

The proposed design has been verified using Xilinx ISE Design Suite 14.1 and implemented using Vertex-7 FPGA with XC7VX330T device, package FFG1157, speed grade -3. The power consumption of the design has been estimated using Xilinx Xpower Analyser tool by creating post

implementation net list. Figures 4 to 15 show the results for different operations. Here all the waveforms shown below have been generated for

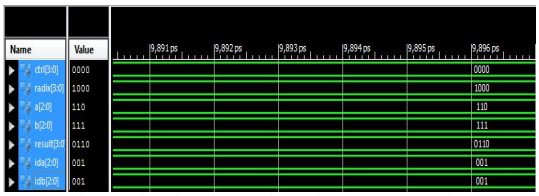

Fig. 4. Simulated waveform for AND operation.

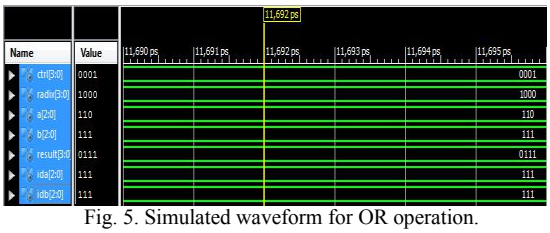

Fig. 5. Simulated waveform for OR operation.

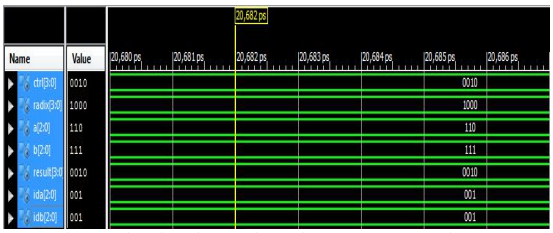

Fig. 6. Simulated waveform for NAND operation.

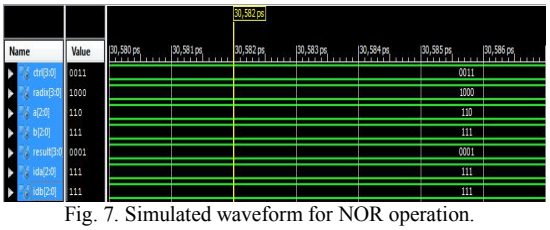

Fig. 7. Simulated waveform for NOR operation.

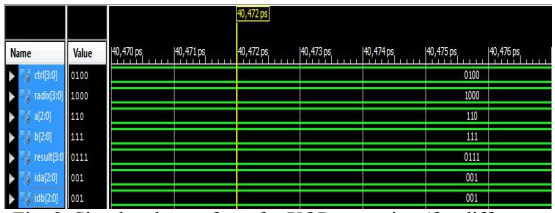

Fig. 8. Simulated waveform for XOR operation (for different operands).

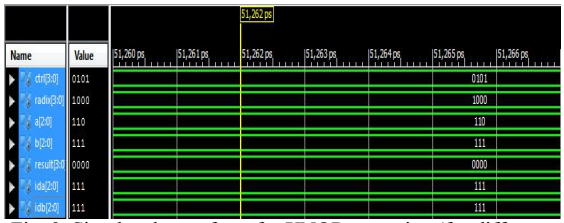

Fig. 9. Simulated waveform for XNOR operation (for different operands).

the radix 8. In Figures 4 to 15, "ctrl" is the control bus for selecting the operations and 'a' and 'b' are the operands. The data bus "result" is the output of the operation. The other two output data buses "ida" and "idb" are for the outputs of the increment/ decrement operations on operand 'a' and 'b' respectively. More over the comparison parameters like area, delay and power have been measured for different radix system. Table IV shows the estimation of the parametric values for different radix system.

5 CONCLUSION

A generic structure for high speed ALU design was reported in this paper. This structure could be used in designing arithmetic as well as logical unit for any radix system. This design scheme was not compared with other contemporary designs because the designs, reported so far, were not incorporated logical unit blocks. All the schemes dwelt with only arithmetic circuits.

ACKNOWLEDGEMENT

The simulation works were performed in the VLSI Laboratory of the Department of ECE, JIS College of Engineering, Kalyani, Nadia, West Bengal, India and also in the Department of ECE, RCCIIT, Kolkata, West Bengal, India. The authors are grateful to the colleagues of the two institutions for their significant co-operations and guidance.

REFERENCES

[1] C. Lazzari, P. Flores, J. Monteiro, and L. Carro, 2010. A new quaternary FPGA based on a voltage-mode multi-valued circuit", in Proc. of Int. Conf. on Design, Automation and Test in Europe, March.

[2] S. Purohit, S. R. Chalamalasetti, and M. Margala, 2009. A 1.2v, 1.02 ghz 8 bit SIMD compatible highly parallel arithmetic data path for multi-precision arithmetic, in Proc. of ACM G.L. Symp. on VLSI, May.

[3] M. Eaton, 2012. Design and Construction of a balanced ternary ALU with potential future cybernetic intelligent systems applications, Int. Conf. on cybernetic Intelligent Systems (CIS), pp. 30-35.

[4] N. Raad and M. M. Mansour, 2011. A Low Power 32-bit Quaternary Tree Adder, Int. Conf. on Energy Aware Computing, pp. 1-2.

[5] H. Shirahama, A. Mochizuki, T. Hanyu, M. Nakajima, and K. Arimoto, 2007. Design of a Processing Element based on Quaternary Differential Logic for a Multi-core SIMD processor, Int. Symp. on Multiple Valued Logic, pp. 43-45.

[6] A. P. Dhande and V. T. Ingole, 2005. Design And Implementation Of 2 Bit Ternary ALU Slice, Int. Conf. on Sciences of

Electronic, Technologies of Information and Telecommunications, Tunisia, March.

[7] Paul Metzgen, 2004. A high performance 32-bit ALU for programmable logic, in Proc. of ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays, February.

[8] L. P. Nascimento, 2001. An Automated Tool for Analysis and Design of MVL Digital Circuits", in Proc. of the 14th Symp. on Integrated circuits and systems design, September.

[9] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, B. Hutchings, 1999. A reconfigurable arithmetic array for multimedia applications, Proc. of ACM/SIGDA Int. Conf. on Field Programmable Gate Arrays, January.

[10] F. Buijs, 1992. ALU synthesis from HDL descriptions to optimized multi-level logic, in Proc. of Int. Conf. on European design automation, October.