Implementation of Energy efficient Arithmetic and Logical Unit on Reconfigurable Logic

Ashutosh Verma, Asst. Prof. G. Ramalakshmi
Dept. of Electronics and Communication Engineering
Bhabha College of Engineering , Bhopal, India

Abstract: This paper deals with low power ALU design and its implementation on 90nm Spartan 3 FPGA. Most of power is consumed in ALU in any processor and hence reduction in ALU power is needed. In this work, we have designed a low power ALU. To reduce dynamic power consumption we disabled the blocks which are not needed in currently selected operation. Also hardware is reused; this will cut down the FPGA resource usage and also reduce the power consumption. By using these methods dynamic power consumption is reduced and less FPGA resources were consumed.

Index Terms— FPGA, ALU, low power, Hardware reuse, tri-state logic, dynamic power consumption.

I. INTRODUCTION

This is an era of hand held devices and equipments, most of these devices runs on battery, this puts a constraint on standby time, to increase standby time more and more battery life is needed, one way of solving this issue is to reduce power consumption of device or equipment. These days almost every device is intelligent, this intelligence came from using processors, and in forthcoming years this trend is likely to be increase. But these processors consume lot of the power of device as lot of switching activity is going inside. ALU (Arithmetic and Logic Unit) is the heart of any processor; this also consumes most of the processor power. In this work we worked in order to reduce power consumption of ALU. We have designed an eight bit optimized ALU, the size of the ALU can be easily increased to 16, 32 or 64 bit. A two level optimization is implemented, first we have reduced the FPGA resource consumption by reusing them for different operations, details are given in forthcoming sections, this will cut down FPGA resource consumption and also power consumption of design, several blocks are designed to implement specified 16 operations, in second level of optimization we enable only one block at a time which is currently selected and all other blocks are disabled, this reduces dynamic power consumption of device and makes our design more greener.

Section 2 of the paper deals with the design of optimized ALU. Section 3 is having results. Section 4 contains concluding remarks of this work. Section 5 contains future scope.

II. DESIGN OF ARITHMETIC AND LOGIC UNIT

The inputs to ALU are A, B (operands), Clk (clock), selection (to select one operation out of sixteen operations). Outputs from ALU are Z (result) and flags. The steps in designing ALU are discussed in next sections.

A. Operations

Our design support sixteen operations, these operations are listed in table 1.

TABLE I
ALU OPERATIONS

Selection	Operation
0000	Clear
0001	Hold B
0010	Complement B
0011	Hold A
0100	Complement A
0101	Decrement A
0110	Increment A
0111	Shift Left A
1000	Add (A + B)
1001	Subtract (A - B)
1010	Add with Carry (A + B + 1)
1011	Subtract with Borrow (A – B - 1)
1100	Logical AND (A AND B)
1101	Logical OR (A OR B)
1110	Logical XOR (A XOR B)
1111	Logical XNOR (A XNOR B)

Different flags are generated depending upon the result of operations.

B. Arithmetic and Logic Unit Design

To support all sixteen operations mentioned in section A, different modules are designed.

Here six operations are similar in nature and can be designed using similar blocks; namely Add, Subtract, Add with carry, Subtract with Borrow, Increment, and Decrement. As we know that subtraction can be implemented using 2's complement, here we have used this property of subtraction and implemented it using adder block. Say we have to implement A – B, this can be implemented as (A + (2's complement of B)). Here we have reused FPGA resources by using same adder block. Similarly increment (A + 1), Decrement (A - 1), Add with carry (A + B + 1) and Subtract with borrow (A – B - 1) is implemented using adder and 2's complement block. This

way above mentioned six operations are implemented using single adder and 2's complement block, and hardware resources are conserved. The high level block diagram of ALU is shown in figure 1

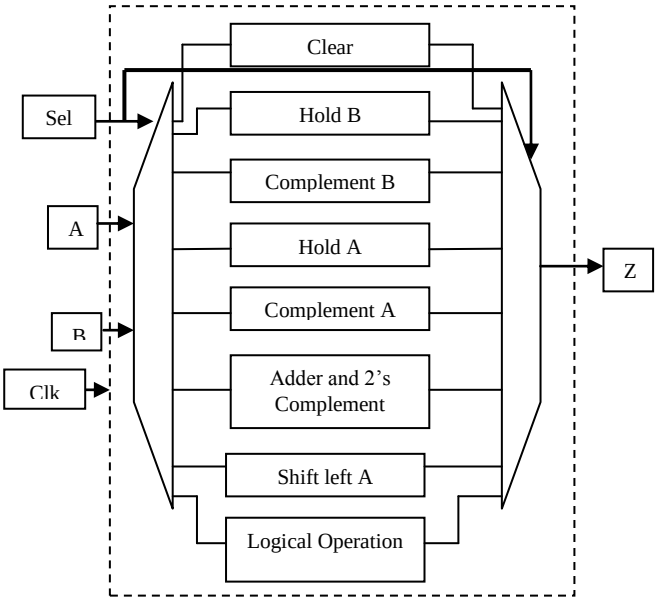

Fig. 1. High Level Block diagram of optimized arithmetic and logic unit (ALU).

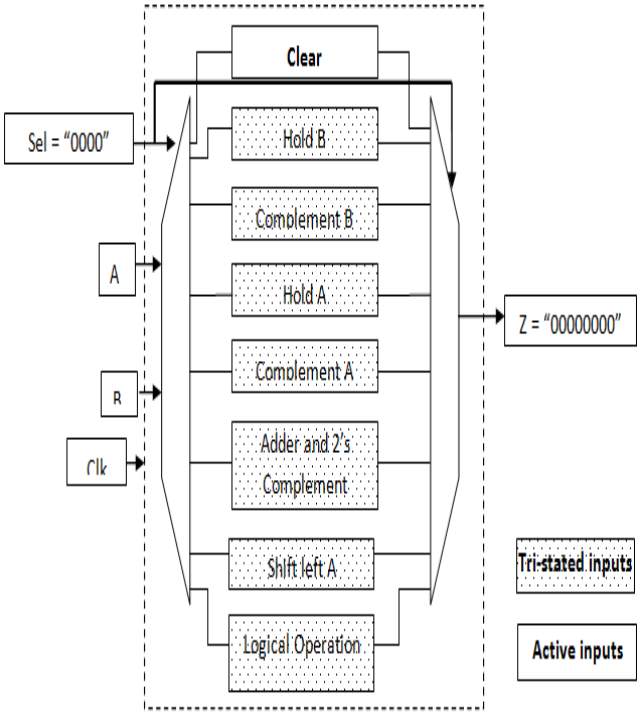

Fig. 2: Clear Operation

High Level Block diagram of optimized arithmetic and logic unit (ALU); here Clear is currently selected operation and as shown, only clear block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

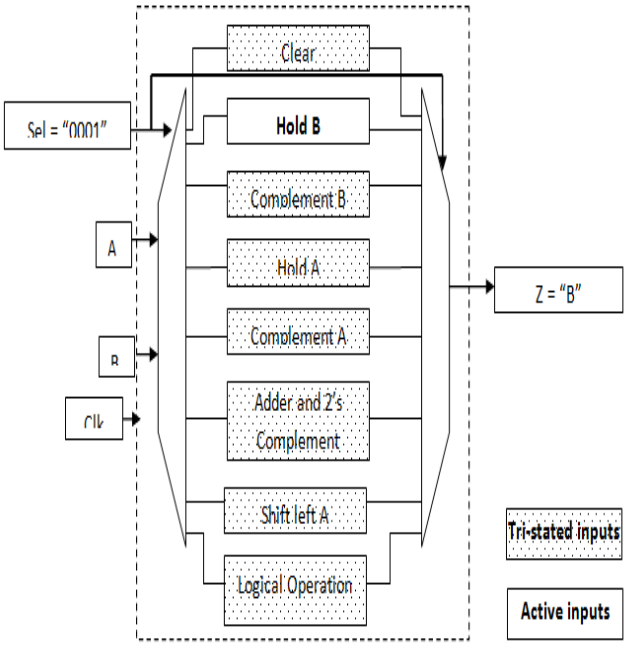

Fig. 3: HOLD D

High Level Block diagram of optimized arithmetic and logic unit (ALU); here hold B is currently selected operation and as shown, only hold B block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

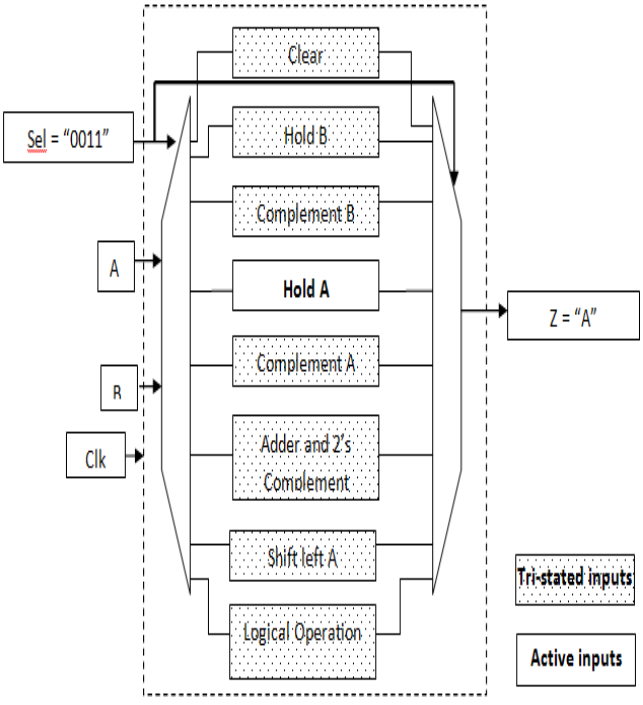

Fig. 4: HOLD A

High Level Block diagram of optimized arithmetic and logic unit (ALU); here hold A is currently selected operation and as shown, only hold A block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

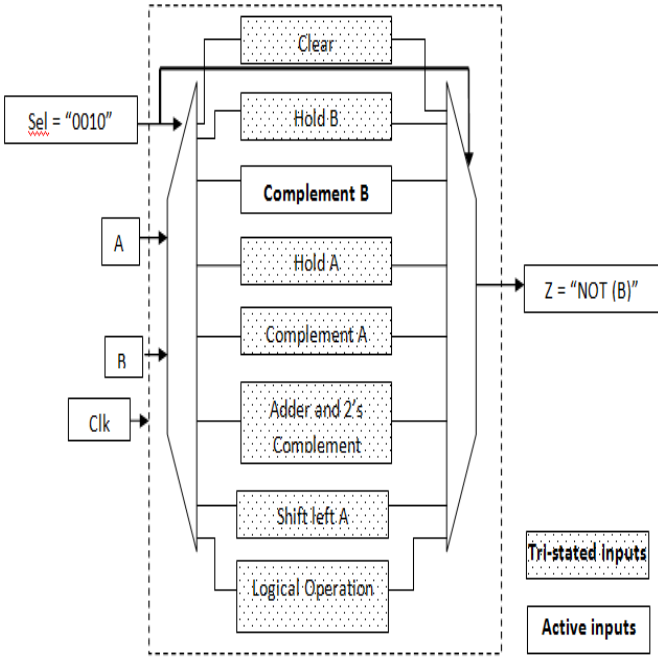

Fig. 5: Complement B

High Level Block diagram of optimized arithmetic and logic unit (ALU); here Complement B is currently selected operation and as shown, only complement B block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

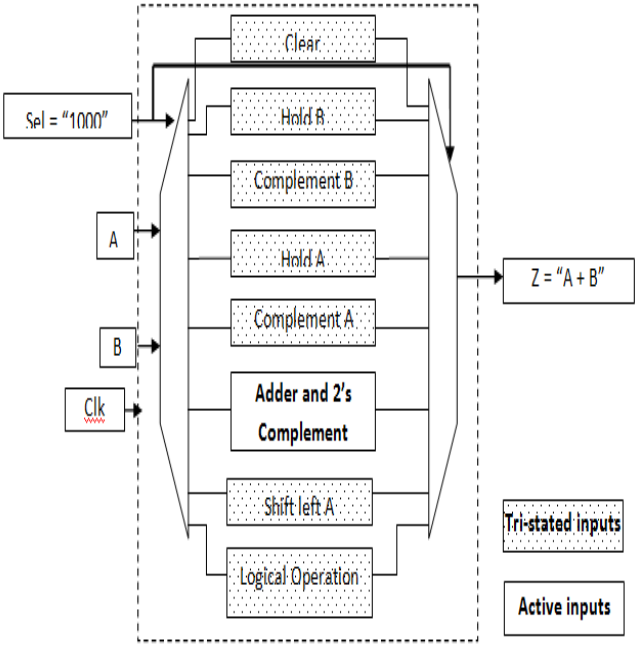

Fig. 7: Add and 2's Complement

High Level Block diagram of optimized arithmetic and logic unit (ALU); six operations will be implemented using this block namely: add, subtract, increment, decrement, subtract with borrow and add with carry. Only this block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption

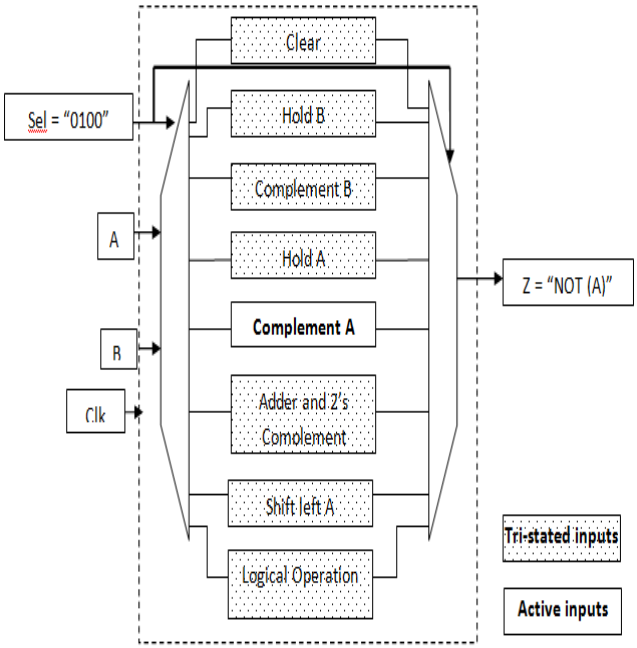

Fig. 6: Complement A

High Level Block diagram of optimized arithmetic and logic unit (ALU); here Complement B is currently selected operation and as shown, only complement B block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

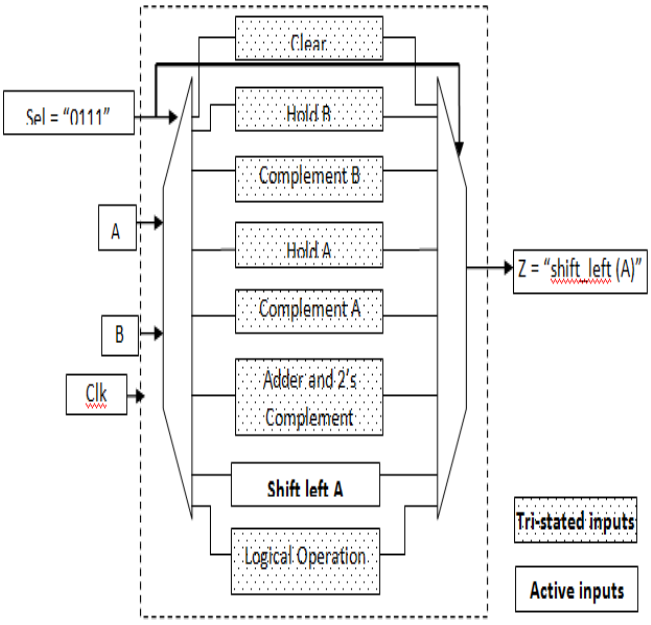

Fig. 7: Shift left A

High Level Block diagram of optimized arithmetic and logic unit (ALU); here shift left is currently selected operation and as shown, only shift left A block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption.

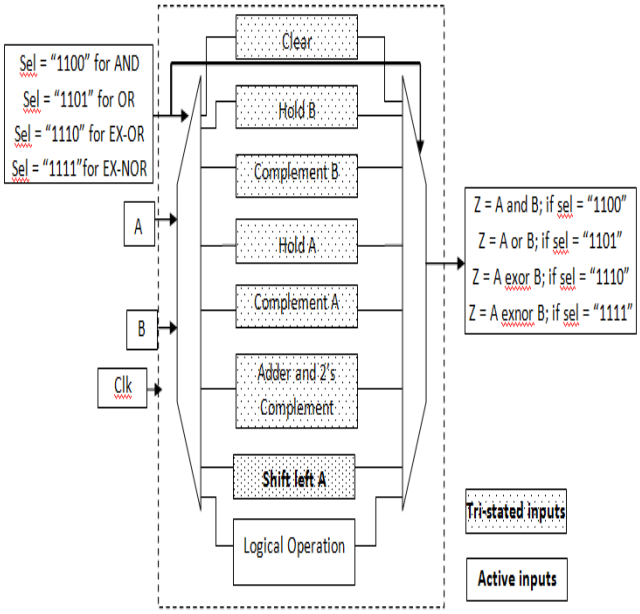

Fig. 8: Logical Operations

High Level Block diagram of optimized arithmetic and logic unit (ALU); here logical operation is currently selected operation and as shown, only logical block is enabled and all other blocks are disabled; this will reduce switching activities and will result in reduction of dynamic power consumption. Four logical operations are performed using this block namely: AND, OR, Exor and Exnor.

Different modules are designed to perform different operations. At a given time only one operation can be performed depending upon the value of 'sel' (selection line to select a particular operation). Here we can utilize this property of ALU to our benefit, we enable one block at a given time and disable all others blocks, this will stop the switching activities of disabled blocks and help in reducing dynamic power consumption ref fig 2. This way dynamic power consumption of ALU can be reduced and makes our design more energy efficient.

III. RESULTS

We have used Xilinx 14.1i to implement the design on 90nm Spartan 3 FPGA. Xpower analyzer is used to perform power analysis. ISIM is used to perform behavioral simulation.

A. Resource Utilization

Table 2 shows the resource utilization summary, due to hardware resource reuse method, resource usage came down.

TABLE 2
RESOURCE UTILIZATION SUMMARY

Logic Utilization	Used
Number of Slice Flip Flops	**28**
Number of 4 input LUT's	**166**
Number of occupied Slices	**114**
Number of bonded IO	**30**

B. Power Report

We used Xpower analyzer tool from xilinx to calculate power consumption of device. The design is operated at different frequencies and power consumption is noted accordingly.

Frequency	Clock (mW)	Logic (mW)	Signals (mW)	IO (mW)	Dynamic Power (mW)
100 Mhz	1	0	1	0	2
1000 Mhz	6	4	6	4	20
10 Ghz	56	23	53	41	173
100 Ghz	557	26	370	411	1364

C. Behavioral Simulation

We have xilinx ISIM for behavioral simulation; we tested our design with different values of input and in all conditions our ALU is behaving as per the specifications. Figure 3 shows the behavioral simulation of optimized ALU. A and B are our two inputs, clk is used as clock source, Z is output port, operation is input by which any operation out of sixteen can be selected and g_reset is used to reset the ALU and it clears all registers inside ALU; output Z after reset is 0. All the values are depicted in hexadecimal. Refer figure 3, all the possible values of operation (0-F) are applied and Z can be seen under all sixteen conditions.

IV. CONCLUSION

ALU is the core of processor, and optimizing ALU can significantly improve the performance of processor. In this work we worked in order to reduce power consumption and resource utilization of FPGA. As we can conclude from power report that by disabling the inactive blocks, dynamic power consumption can be significantly reduced; this is because of decrease in switching activities inside ALU. Next improvement we tried to implement is reduction in FPGA resource usage; we removed few blocks such as subtract, increment, decrement, add with carry and subtract with borrow and implement all these functions using single adder and 2's complement b lock. This fulfills our two purposes; first, reduction of FPGA resource usage and second reduction in power consumption.

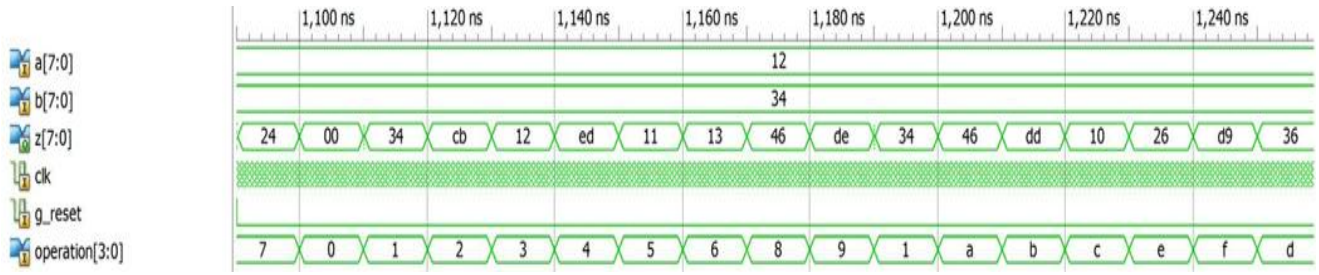

Figure 3. Behavioral simulation of optimized low power ALU. A and B are input, Z is output, operation is input by which any one operation out of sixteen can be selected. Here all sixteen possibilities of operation (0 - F) are applied. #all values in hexadecimal.

V. FUTURE SCOPE

In this work we implemented our design on 90nm Spartan 3 FPGA. One possibility of improvement is designing it on 28nm FPGA. Next possibility is optimizing the code; this will improve the performance of ALU. The size of the ALU can also be increased to 16, 32 or 64 bit. Our amendments will show a better performance on larger size.

REFERENCES

[1] J. P. Oliver, J. Curto, D. Bouvier, M. Ramos, and E. Boemo, “Clock gating and clock enable for FPGA power reduction”, 8th Southern Conference on Programmable Logic (SPL), pp. 1-5, 2012.

[2] J. Shinde, and S. S. Salankar, “Clock gating-A power optimizing technique for VLSI circuits”, Annual IEEE India Conference (INDICON), pp. 1-4, 2011.

[3] J. Castro, P. Parra, and A. J. Acosta, “Optimization of clock-gating structures for low-leakage high-performance applications”, Proceedings of IEEE International Symposium on Efficient Embedded Computing, pp. 3220-3223, 2010.

[4] V. Khorasani, B. V. Vahdat, and M. Mortazavi, ”Design and implementation of floating point ALU on a FPGA processor”, IEEE International Conference on Computing, Electronics and Electrical Technologies (ICCEET), pp.772-776, 2012.

[5] S. Cisneros, J. J. Panduro, J. Muro, and E. Boemo, “Rapid prototyping of a self-timed ALU with FPGAs”, International Conference on Reconfigurable Computing and FPGAs, pp. 26-33, 2012.

[6] B. S. Ryu, J. S. Yi, K. Y. Lee and T. W. Cho,”A design of low power 16-bit ALU”, Proceedings of the IEEE TENCON Conference, pp.868-871, 1999.

[7] T. Lam, X. Yang, W. C. Tang and Y. L. Wu; , "On applying erroneous clock gating conditions to further cut down power," Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific , vol., no., pp.509-514, 25-28 Jan. 2011.

[8] B. Pandey and M. Pattanaik, “Clock Gating Aware Low Power ALU Design and Implementation on FPGA”, 2nd International Conference on Network and Computer Science (ICNCS), Singapore, April 1-2, 2013

[9] E. Arbel, C. Eisner and O. Rokhlenko, "Resurrecting infeasible clockgating functions," Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE , vol., no., pp.160-165, 26-31 July 2009

[10] J. Monteiro, J. Rinderknecht, S. Devadas and A. Ghosh, "Optimization of combinational and sequential logic circuits for low power using precomputation," Advanced Research in VLSI, 1995. Proceedings.