Design and Implementation of FPGA based Barrel shifter

Shridhar Devamane #1, Akshada Hanchate #2, Usha Vagare #3, Shalaka Ujagare #4, Pushpa Teggelli #5
Department of Electronics & Telecommunication, Nagesh Karajagi Orchid College Of Engineering & Technology, Solapur.

Abstract— **A barrel shifter is a Digital circuit that can shift a data word by a specified number of bits in one clock cycle. It can be implemented as a sequence of multipliers (mux), and in such an implementation the output of one mux is connected to the input of the next mux in a way that depends on the shift distance. In arithmetic and logic operations barrel shifters is used to shift a desired number of bits in a desired direction. In this paper an 8-bit & an 16-bit barrel shifter architecture is proposed and implemented using Verilog code.**

Keywords—**2:1 Mux barrel shifter, 8:1 Mux barrel shifter, FPGA, right shifter, right rotator.**

I. INTRODUCTION

A Barrel Shifter is a logic component that performs shift or rotate operations. Barrel shifters are applicable for digital signal processors. This component design is for natural size (4,8,16…) barrel shifters that perform *shift right logical, shift left logical, rotate left and rotate right* operations depending on the instantiation parameters. The left and right operation is implemented through inversion of the input and output vectors. The number of multiplexing stages is relative to the width of the input vector.

Block Diagram

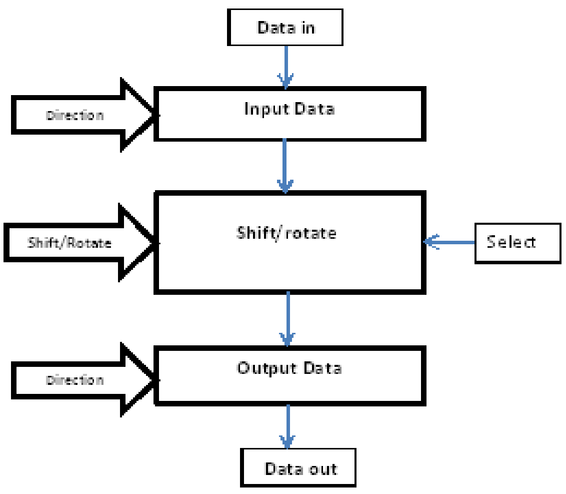


Fig 1- Block Diagram of Barrel Shifter

Shifting and rotating data is required in several applications including arithmetic operations, variable length coding, are bit indexing. Consequently, barrel shifters, which are capable of shifting or rotating data in a single cycle. Select lines are used to specify the amount of shift only. A barrel shifter can be designed by using mux trees.

Architecture

1. Logical right shifter:

A logical right shifter using a fore mentioned approach is shown in the figure 1.The first row corresponds to a shift of one, while the last row corresponds to a shift of four, As required, zeros fill the high order region. Hence, interconnects route zero into the high order multiplexers. The values x_amt[x] represents the bit in position x of the shift amount, and as such represents the value 2^x,

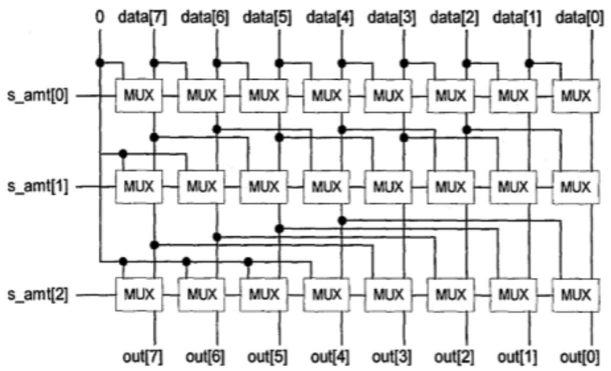


Fig 2- Logical Right Shifter

1.1 Operation of right shifter:

The shift right logical operation is much like a rotate right but without the lower order bits (LSB) being moved to a high order (MSB) position. Instead, the LSB bits are removed. The remaining bits are shifted to the right so as to fill the void created by the loss of the low order bits (LSB). The void created in the MSB region by this shift is filled with zeros.

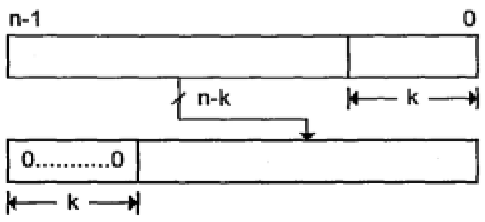


Fig 3- Right Shifter Operation

As shown in example below, the LSB bit is removed from the result and the remaining bits are shifter over. The order bits are set to zero.

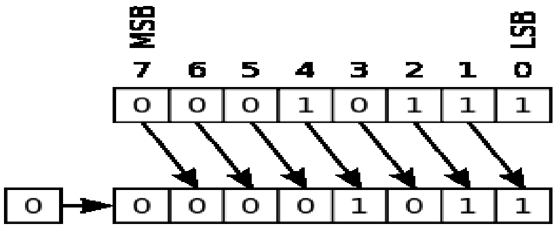


Fig 4- Example of right shifter

2. Right Rotator:

A right rotator is very similar to a logical right shifter. The differences between the two lies in the manner in which interconnect lines are placed, In particular, since all of the input bits are routed to the output, there is no longer a need for interconnect lines carrying the zero signal. Instead, interconnect lines need to be inserted to enable routing of the low order bits from each row to high order region so that rotate can occur. The longer interconnect lines of the rotator, however, can increase both area and delay.

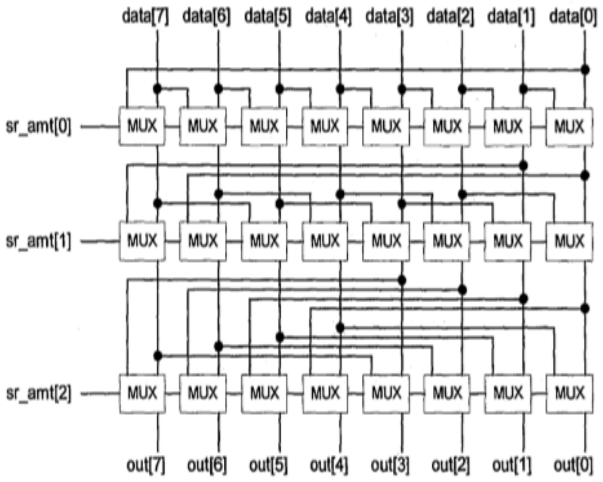

data[7] data[6] data[5] data[4] data[3] data[2] data[1] data[0]
sr_amt[0]
sr_amt[1]
sr_amt[2]
MUX
out[7] out[6] out[5] out[4] out[3] out[2] out[1] out[0]

Fig 5: Right Rotator

2.1 Operation of right rotator:

Rotate is a cyclic shift either to the left or right. This means that as bits are shifted out of the data vector on one side, they are shifted into the data vector on the other side. During this process, all bits from the input are routed to the output. Their position in the output, however, is not necessarily the same as it was in the input.

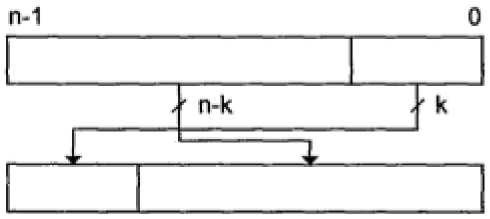


Fig 6- Right Rotator Operation

An example of rotate right can be seen in the figure 6 where a 8-bit word of data has a one bit rotated right.

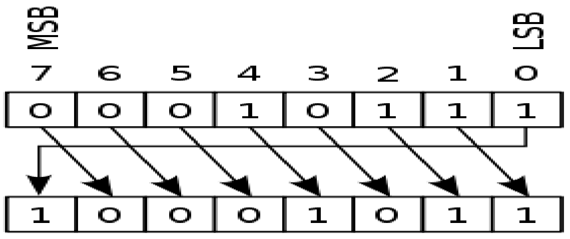


Fig 7- Example for right rotator

II. RESULTS

The following figures show the RTL schematics of 8-bit & 16-bit barrel shifter respectively.

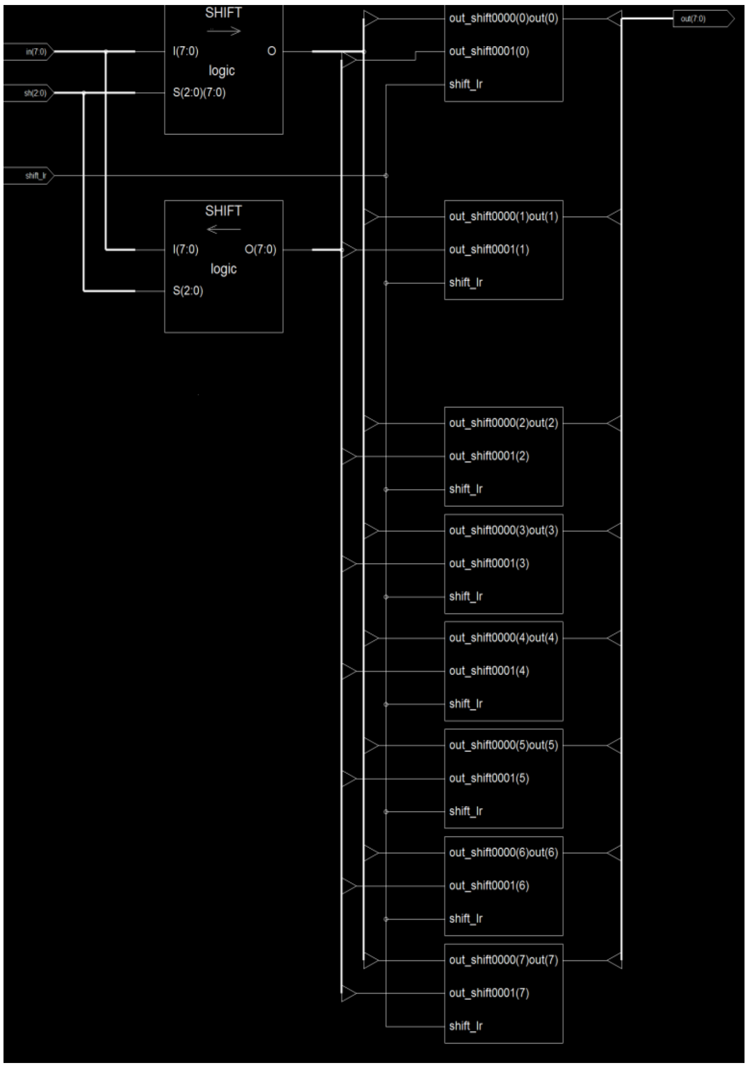


Fig 8- RTL schematic of 8-bit barrel shifter

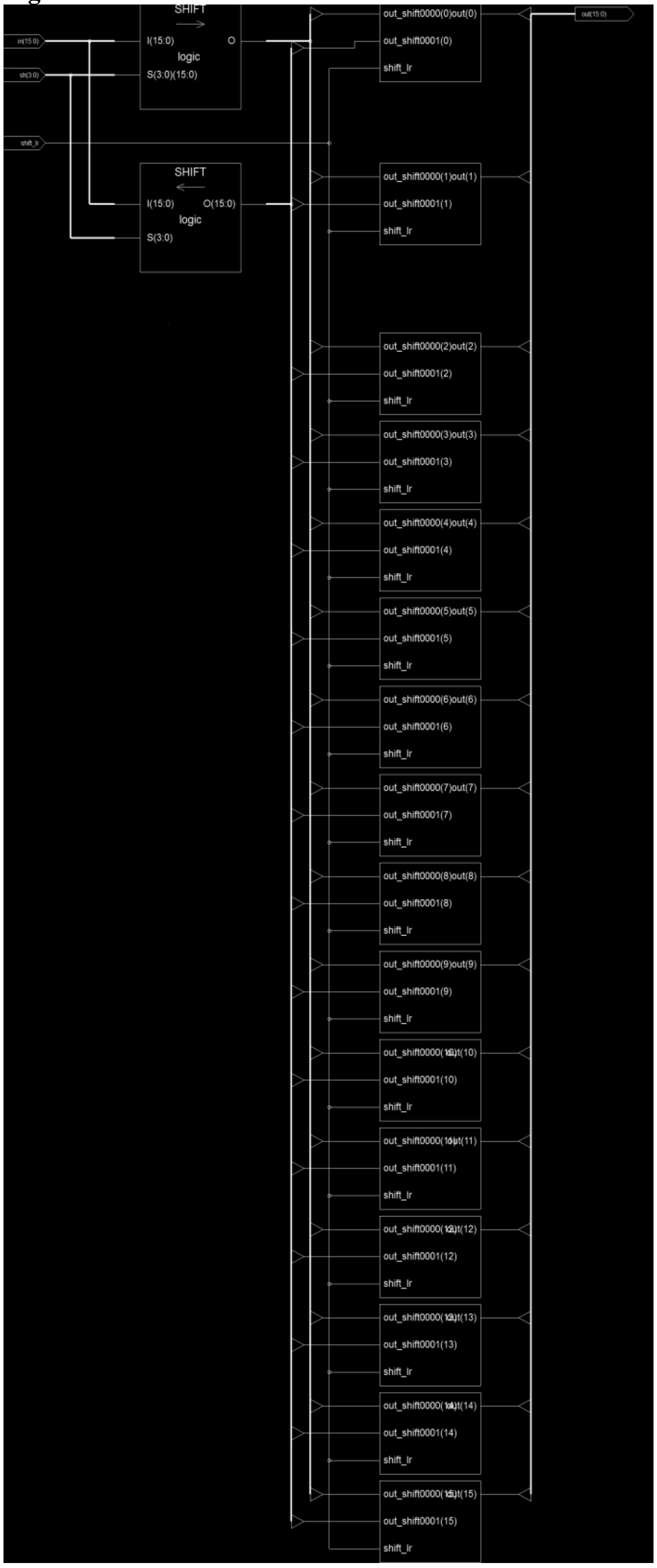
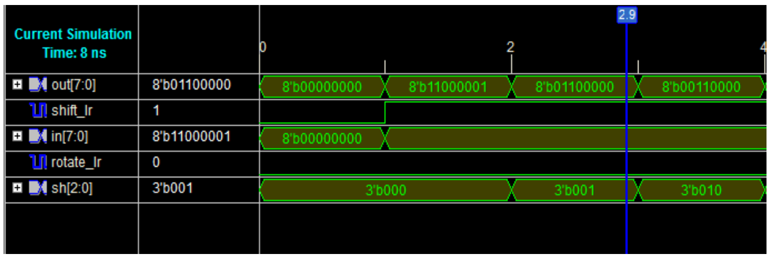


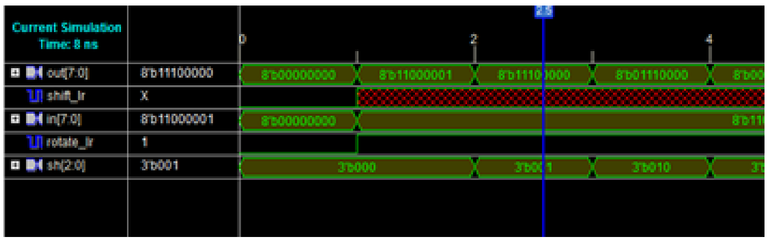
Fig 9- RTL schematic of 16-bit barrel shifter

The results shown below are for the 8 bit and 16 bit barrel shifter which shows output waveforms for different operations such as right shift and right rotate.



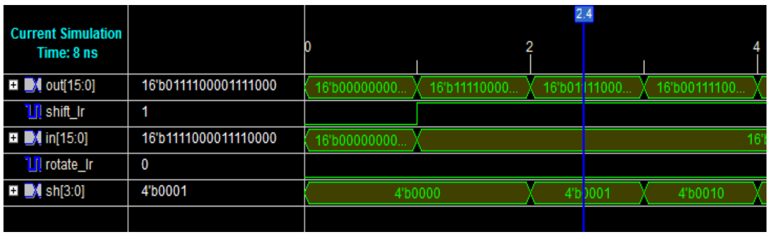
```
Current Simulation
Time: 8 ns
out[7:0]     8'b01100000    8'b00000000  8'b11000001  8'b01100000  8'b00110000
shift_lr     1
in[7:0]      8'b11000001    8'b00000000
rotate_lr    0
sh[2:0]      3'b001         3'b000  3'b001  3'b010
```

Simulation waveform of 8-bit Barrel Shifter (right shift)



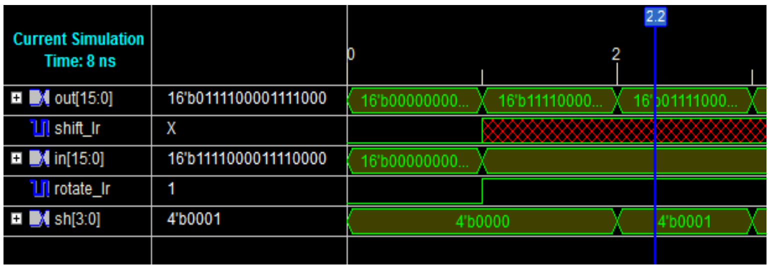
```
Current Simulation
Time: 8 ns
out[7:0]     8'b11100000    8'b00000000  8'b11000001  8'b11100000  8'b01110000
shift_lr     X
in[7:0]      8'b11000001    8'b00000000
rotate_lr    1
sh[2:0]      3'b001         3'b000  3'b001  3'b010
```

Simulation waveform of 8-bit Barrel Shifter (right rotate)



```
Current Simulation
Time: 8 ns
out[15:0]    16'b0111100001111000    16'b00000000...  16'b11110000...  16'b01111000...  16'b00111100...
shift_lr     1
in[15:0]     16'b1111000011110000    16'b00000000...
rotate_lr    0
sh[3:0]      4'b0001                 4'b0000  4'b0001  4'b0010
```

Simulation waveform of 16-bit Barrel Shifter (right shift)



```
Current Simulation
Time: 8 ns
out[15:0]    16'b0111100001111000    16'b00000000...  16'b11110000...  16'b01111000...
shift_lr     X
in[15:0]     16'b1111000011110000    16'b00000000...
rotate_lr    1
sh[3:0]      4'b0001                 4'b0000  4'b0001
```

Simulation waveform of 16-bit Barrel Shifter (right rotate)

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
No. of slices	**22**	**960**	**2%**
No. of 4 input LUTs	**40**	**1920**	**2%**
No. of bonded IOBs	**20**	**66**	**30%**

Table 1- Design summary of 8-bit barrel shifter

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
No. of slices	**65**	**960**	**6%**
No. of 4 input LUTs	**117**	**1920**	**6%**
No. of bonded IOBs	**37**	**66**	**56%**

Table 2- Design summary of 16-bit barrel shifter

III. CONCLUSION

We have proposed and designed a Verilog implementation of FPGA based barrel shifter. Which produce appreciable results. It demonstrated that our approach yields & high speed barrel shifter.

III. ACKNOWLEDGMENT

Authors would like to thank the Department of Electronics and Tele-Communication Engineering and also those individuals who have motivated and helped in carrying this work successfully.

REFERENCES

[1]M.Seckora, Barrel Shifter or Multiply/Divide IC Structure," U.S. Patent 5,465,222, November 1995.
[2] J. Muwafi, G. Fettweis, and H. Neff, "Circuit for Rotating, Left Shifting, or Right Shifting Bits," U.S. Patent 5.978,822, December 1995.
[3]A.Yamaguchi, \Bidirectional Shifter," U.S. Patent 5,262,971, November 1993.
[4] A. Ito, \Barrel Shifter," U.S. Patent 4,829,460, May 1989.
[5]K. Dang and D. Anderson, \High-Speed Barrel Shifter," U.S. Patent 5,416,731, May 1995.
[6] G. F. Burns, \Method for Generating Barrel Shifter Result Flags Directly from Input Data," U.S. Patent 6,009,451, December 1999.
[7]V. Milutinovic, M. Bettinger, and W. Helbig, \Multiplier/Shifter Design Tradeo_s in a 32-bit Micropro-cessor," IEEE Transactions on Computers, vol. 38, pp. 874{880, June 1989.
[8] G. M. Tharakan and S. M. Kang, \A New Design of a Fast Barrel Switch Network," IEEE Journal of Solid-State Circuits, vol. 28, pp. 217{221, February 1992.
[9]F. Worrell, "Microprocessor Shifter using Rotation and Masking Operations," U.S. Patent 5,729,482, March 1998.
[10]K. Dang and D. Anderson, "High-Speed Barrel Shifter," U.s. Patent 5,416,731, May 1995.
[11]S.Palnitkar, "Verilog HDL: A guide to Digital design and synthesis",Prentice Hall,Upper Saddle River,NJ,2003.