ISSN: 2348-4098
DATE OF PUBLICATION: SEP 23, 2014
VOLUME 2 ISSUE 7 SEP-OCT 2014

BARREL SHIFTER

[1]PRAGATI SACHAN, [2]ANCHAL KATIYAR, [3]ANITA DIDAL, [4]PALLAVI GAUTAM

M.Tech Scholar, VLSI, Jayoti Vidyapeeth Women's University Jaipur, Rajasthan, INDIA, E-mail:
[1]sachanpragati.kgi@gmail.com, [2]anchalkatiyar9@gmail.com, [3]anitadidal90@gmail.com, [4]pallavigautam89@gmail.com

ABSTRACT

Barrel shifters are often utilized by embedded digital signal processors and general-purpose processors to manipulate data. This examines design alternatives for barrel shifters that perform the following functions:

Shift right logical, shift right arithmetic, rotate right, shift left logical, shift left arithmetic, and rotate left. Four different barrel shifter designs are presented and compared in terms of area and delay for a variety of operand sizes. This also examines techniques for detecting results that overflow and results of zero in parallel with the shift or rotate operation.

1. INTRODUCTION

A barrel shifter is a digital circuit that can shift a data word by a specified numbers of bits in one clock cycle. It can be implemented as a sequence of multiplexors (mux), and in such an implementation the output of one mux is connected to the next mux in a way that depends on the shift distance. Barrel shifters are often utilized by embedded digital signal processors and general purpose processors to manipulate data. Shifting and rotating data is required in several applications, variable-length coding, and bit indexing. The mux based barrel shifter architecture are designed using 4:1,8:1,16:1,32:1,and 64:1 mux trees. Each mux tree is designed using 2:1 mux. The power consumed by mux trees is quite significant and cannot be ignored. Thus it is important to minimize power dissipation of mux trees within low power designs. Multiplexers are digital circuit that generates an output that exactly reflects state of one of a number of data inputs, based on value of select lines. A multiplexer with two data input and one select line is referred as "2-to-1or 2:1"multiplexer. A barrel shifter primarily offers five operations; rotate right, rotate left, shift right logical, shift left logical, and shift right arithmetic, shift left arithmetic. An n-bit logarithmic barrel shifter uses log2 (n) stages [1, 2]. Each bit of the shift amount, B, controls a different stage of the shifter. The data in to stage controlled by b_k is shifted by 2^k bits if b_k=1; otherwise it is not shifted. Techniques are also presented for detecting results that overflow and results of zero in parallel with the shift or rotate operation.

2. DESIGN SPECIFICATION

Basic shifter and rotator designs are described first followed by Mux-based data-reversal barrel-shifters. The term multiplexer refers to a 1-bit to 2-to-1 multiplexor unless otherwise stated. A row of n multiplexors reverse the order of the data when left=1 to produce the final result. A Mux based data reversal barrel shifter, also detect overflow and results of zero. Overflow only occurs when performing a shift left arithmetic operation and one or more of the shifted-out bits differ from the sign bit. The operation performed by the barrel shifters is controlled by a 3-bit opcode, which consists of the bits left, rotate and arithmetic, additional control signal, sra and sla are set to one when performing shift right arithmetic and shift left arithmetic operations. An n-bit logarithmic barrel shifter uses log2 (n) stages [1, 2]. Each bit of the shift amount, B, controls a different stage of the shifter. The data in to stage controlled by b_k is shifted by 2^k bits if b_k=1; otherwise it is not shifted. Techniques are also presented for detecting results that overflow and results of zero in parallel with the shift or rotate operation.

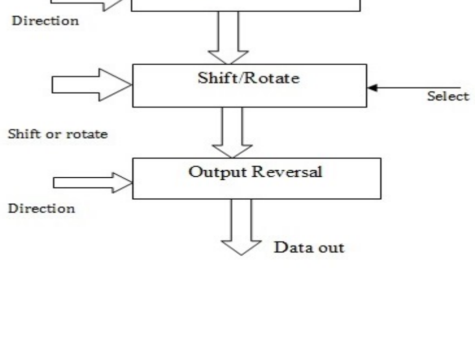
Table no. - 1: Shift and rotate example for A=a7a6a5a4a3a2a1a0 and B=3

Operation	Y
3-bit shift right logical	0 0 0 a7a6a5a4a3
3-bit shift right arithmetic	a7a7a7a7a6a5a4a3
3-bit rotate right	a2a1a0a7a6a5a4a3
3-bit shift left logical	a4a3a2a1a0 0 0 0
3-bit shift left arithmetic	a7a3a2a1a0 0 0 0
3-bit rotate left	a4a3a2a1a0a7a6a5

3. DESIGN METHODOLOGIES

Tools: - Modelsim

Block Diagram:-



ISSN: 2348-4098
VOLUME 2 ISSUE 7 SEP-OCT 2014

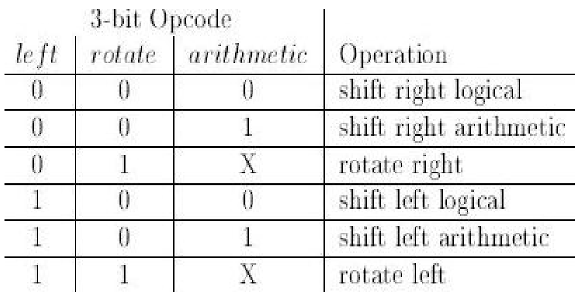
The block diagram showing the Data in, direction, input reversal, shift or rotate, select line, output reversal, data out.

Table NO. – 2: Barrel Shifter Functionality

Mode	Rotation	Direction	Description
Shift Left Logical	0	0	Logic shift left, 0 is shifted through the right most (lsb) bit.
Rotate Left	1	0	Left rotate, the ~~right~~ most bit is shifted back in form the right.
Shift Right Logical	0	1	Logical shift right, 0 is shifted the left most (msb) bit.
Rotate Right	1	1	Right rotate, the right most bit is back in from the left.

The right rotator and the logical right shifter supply different inputs to the more significant multiplexors. With the rotator, since all of the input bits are routed to the output, there is no longer a need for interconnect lines carrying zeros. Instead, interconnect lines are inserted to enable routing of the 2k low order data bits to the 2k high order multiplexors in the stage controlled by bk. Changing from a non-optimized shifter to a rotator has no impact on the theoretical area or delay. The longer interconnect lines of the rotator; however, can increase both area and delay. The logical right shifter can be extended to also perform shift right arithmetic and rotate right operations by adding additional multiplexors. This for an 8-bit right shifter/rotator with three stages of 4-bit, 2-bit, and 1-bit shifts/rotates. Initially, a single multiplexor selects between '0' for logical right shifting and 1 for arithmetic right shifting to produce s. In the stage controlled by bk, 2k multiplexors select between s for shifting and the 2k lower bits of the data for rotating. Before and after the right shifter, when a left shift operation is performed, these multiplexors reverse the data into and out of the right shifter. When a right shift operation is performed, the data into and out of the shifter is not changed.

Table no 3



3-bit Opcode			
left	*rotate*	*arithmetic*	Operation
0	0	0	shift right logical
0	0	1	shift right arithmetic
0	1	X	rotate right
1	0	0	shift left logical
1	0	1	shift left arithmetic
1	1	X	rotate left

3.1 Operation control bit

1. A B-bit shift right logical operation performs a B-bit right shift and sets the upper B bits of the result to zeros.

2. A B-bit shift right arithmetic operation performs a B-bit right shift and sets the upper B bits of the result to an-1, which corresponds to the sign bit of A.

3. A B-bit rotate right operation performs a B-bit right shift and sets the upper B bits of the result to the lower B bits of A.

4. A B-bit shift left logical operation performs a B-bit left shift and sets the lower B bits of the result to zeros.

5. A B-bit shift left arithmetic operation performs a B-bit left shift and sets the lower B bits of the result to zeros. The sign bit of the result is set to an-1.

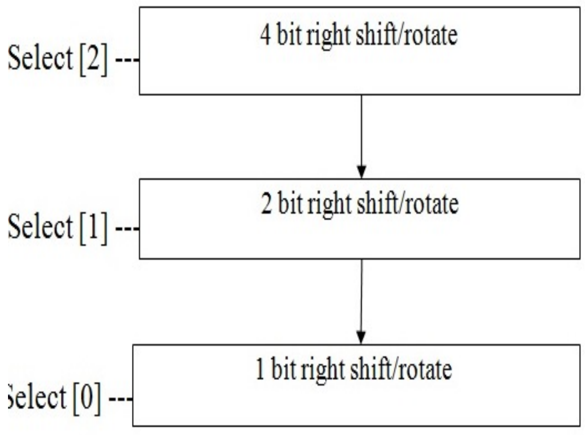
6. A B-bit rotate left operation performs a B-bit left shift and sets the lower B bits of the result to the upper B bits of A.

The operation performed by the barrel shifters is controlled by a 3-bit opcode, which consists of the bits left, Rotate, and arithmetic, as summarized in Table 3. Additional control signals, sra and sla, are set to one when performing shift right arithmetic and shift left arithmetic operations, respectively.

4. DESIGN IMPLEMENTATION

A barrel shifter is often implemented as a:-

Rotate and Shift Direction:-The direction of the rotate and shift operation is implemented by reversing the input and output vector, using this method allows for the shift or rotate logic to be kept simple.



[A] Logical Shift Operation: - The logical shift operation inserts 0 values for each shift operation. The input vector is shifted in the selected direction according to the number of bits in the select indication.

[B] Rotate Operation: - The rotate operation is a shift where the bit which is shifted out of the vector MSB is inserted at its LSB.

[C] Shift and Rotate Operation: - We defined A to be the input operand, B to be the shift/rotate amount, and Y to be the shifted/rotated result. We define A to be an n-bit value, where n is an integer power of two. Therefore, B is a log2(n)-bit integer representing values from 0 to n - 1 we define A to be the input operand, B to be the shift/rotate amount, and Y to be the shifted/rotated result. We define A to be an n-bit value, where n is an integer power of two. Therefore, B is a log2(n)-bit integer representing values from 0 to n-1.

4.1. Figure1- 8-Bit logical right shifter

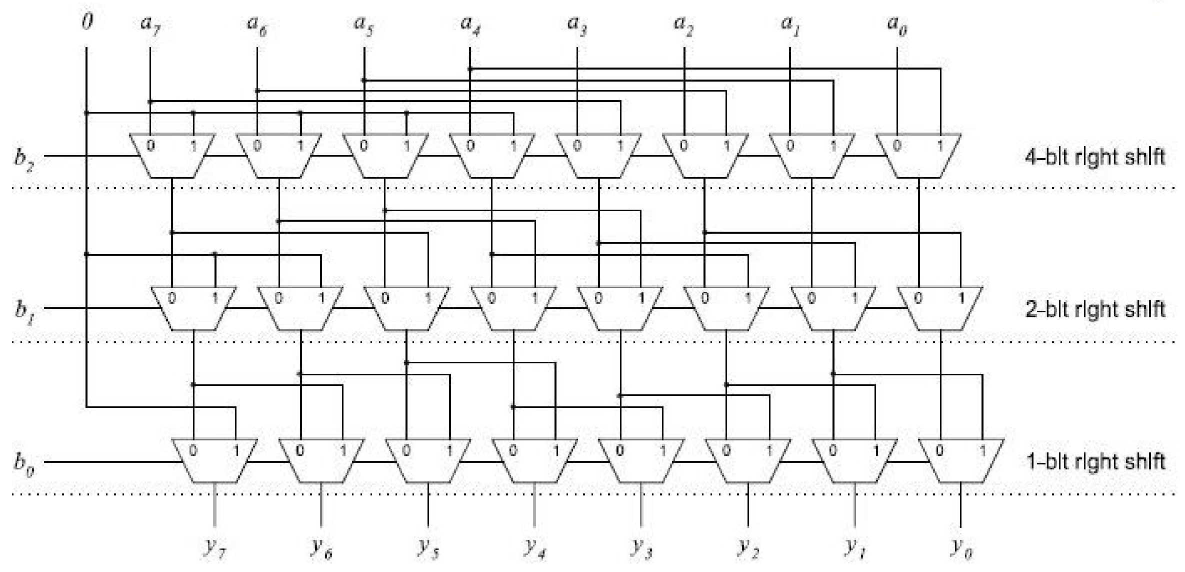


Figure 1. 8-bit logical right shifter.

4.2.Figure2- 8 Bit right rotator

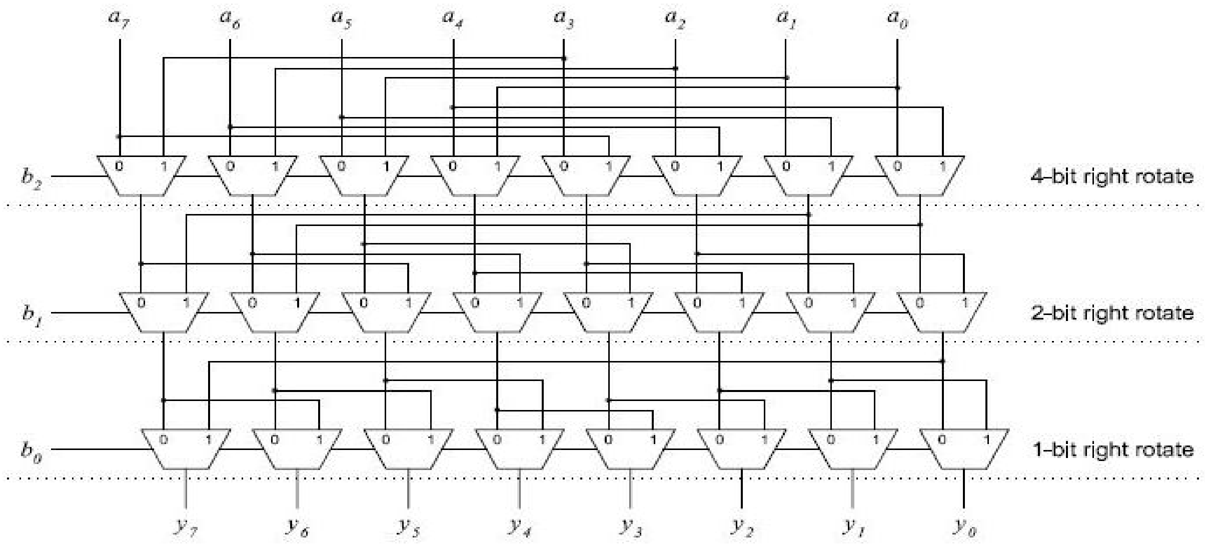


Figure 2. 8-bit right rotator

Figure 1 shows the block diagram of an 8-bit logical right shifter, which uses three stages with 4-bit, 2-bit, and 1-bit shifts. To optimize the design, each multiplexor that has '0' for one of its inputs can be replaced by a 2-input and gate with the data bit and bk as inputs. A similar unit that performs right rotations, instead of right shifts, can be designed by modifying the connections to the more significant multiplexors. Figure 2 shows the block diagram of an 8-bit right rotator which uses three stages with 4-bit, 2-bit, and 1-bit rotates. The right rotator and the logical right shifter supply different inputs to the more significant multiplexors. With the rotator, since all of the input bits are routed to the output, there is no longer a need for interconnect lines carrying zeros. Instead, interconnect lines are inserted to enable routing of the 2k low order data bits to the 2k high order multiplexors in the stage controlled by bk. Changing from a non-optimized shifter to a rotator has no impact on the theoretical area or delay. The longer interconnect lines of the rotator can increase both area and delay. The logical right shifter can be extended to also perform shift right arithmetic and rotate right operations by adding additional multiplexors. This approach is illustrated in Figure 3, for an 8-bit right shifter/rotator with three stages of 4-bit, 2-bit, and 1-bit shifts/rotates. Initially, a single multiplexor selects between '0' for logical right

ISSN: 2348-4098
VOLUME 2 ISSUE 7 SEP-OCT 2014

shifting and a_n-1 for arithmetic right shifting to produce s. In the stage controlled by bk, 2k multiplexors select between s for shifting and the 2k lower bits of the data for rotating.

4.3 Figure3- 8 Bit data reversal

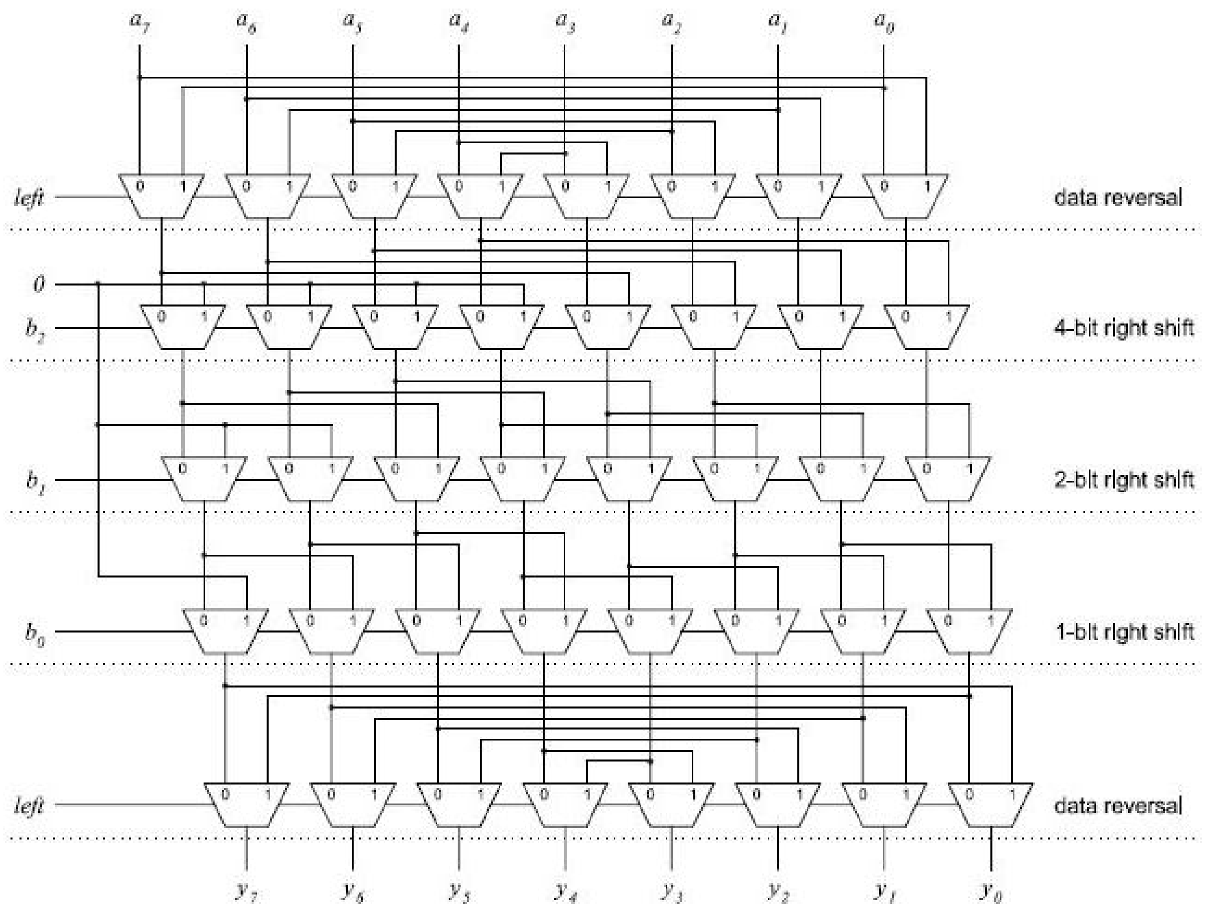
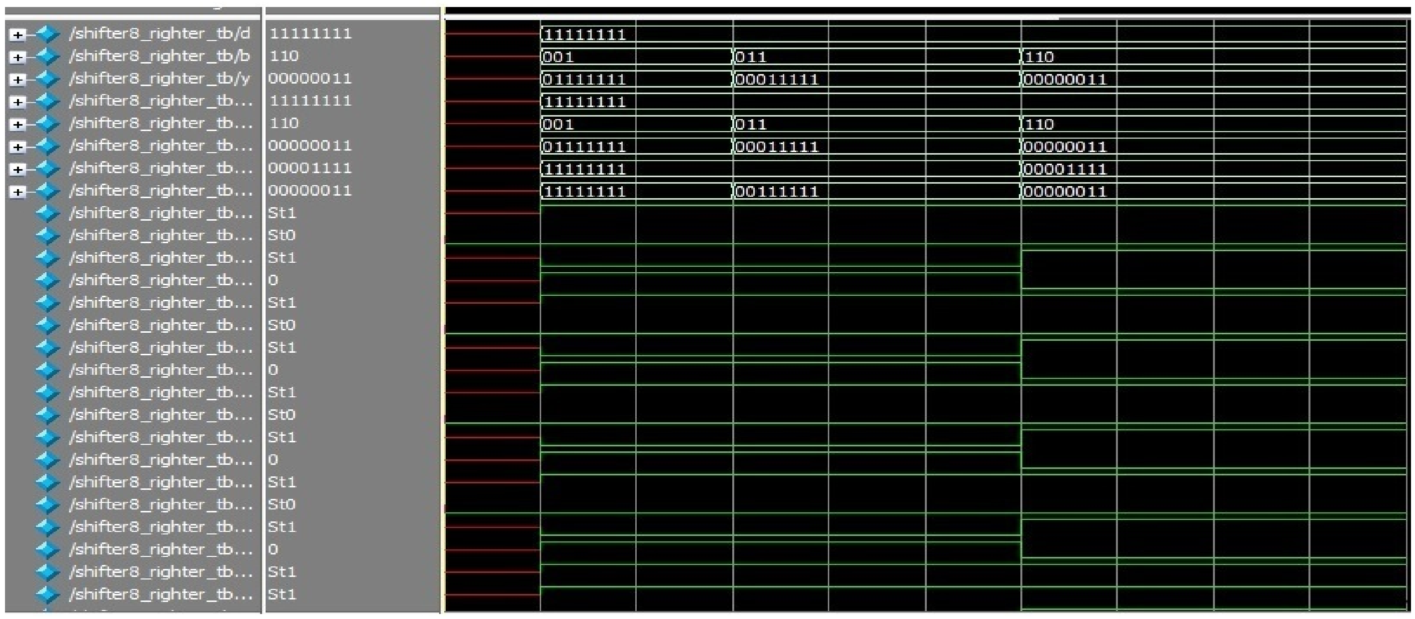


Figure 4. 8-bit data-reversal logical shifter.

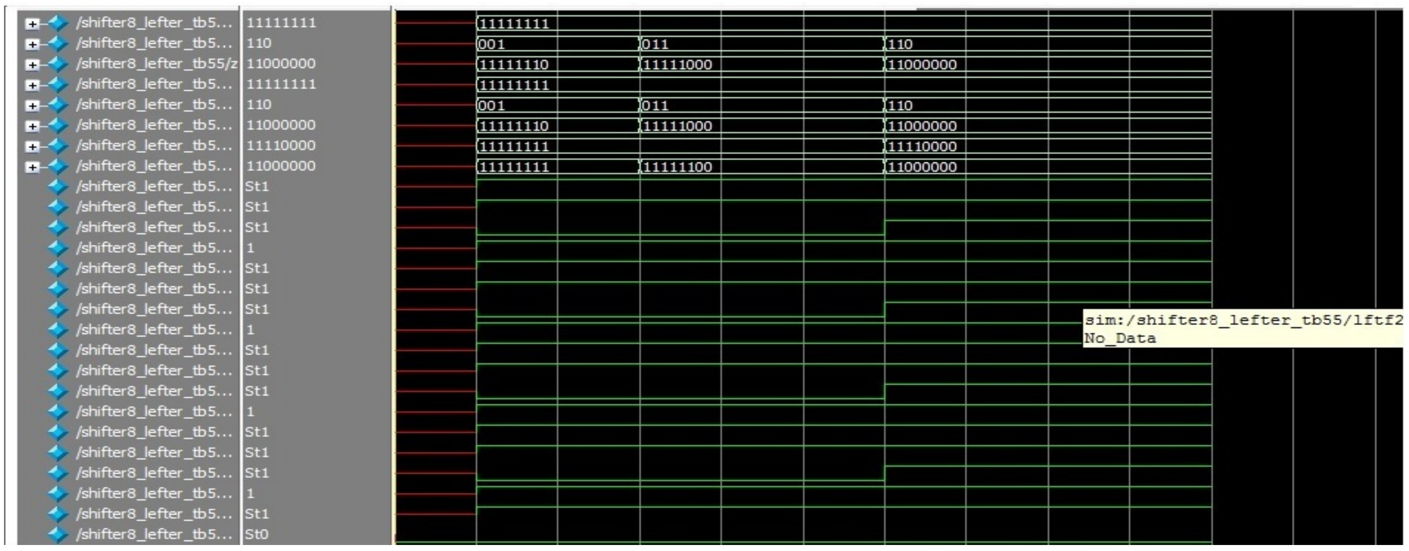
5. SIMULATION AND RESULTS

5.1. Simulation Result (right shift)



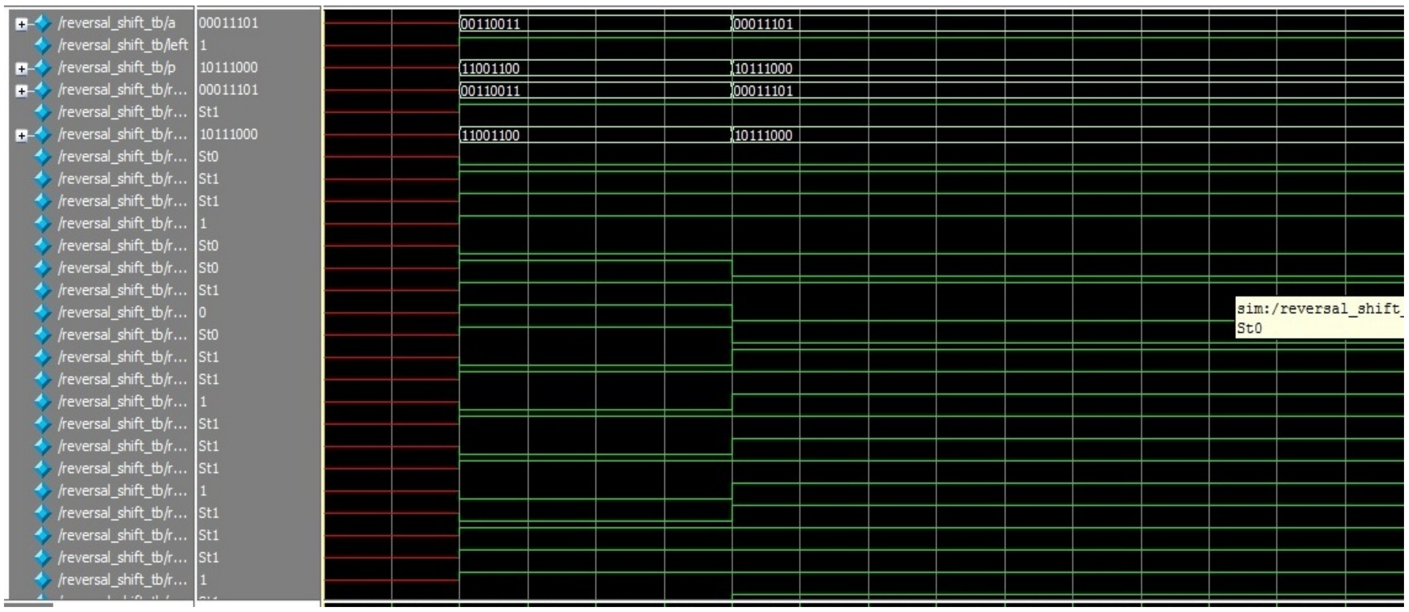
```
/shifter8_righter_tb/d   11111111   11111111
/shifter8_righter_tb/b   110        001        011        110
/shifter8_righter_tb/y   00000011   01111111   00011111   00000011
/shifter8_righter_tb...  11111111   11111111
/shifter8_righter_tb...  110        001        011        110
/shifter8_righter_tb...  00000011   01111111   00011111   00000011
/shifter8_righter_tb...  00001111   11111111              00001111
/shifter8_righter_tb...  00000011   11111111   00111111   00000011
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  St0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  St0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  St0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  St0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  0
/shifter8_righter_tb...  St1
/shifter8_righter_tb...  St1
```

5.2. Simulation Result (left shift)



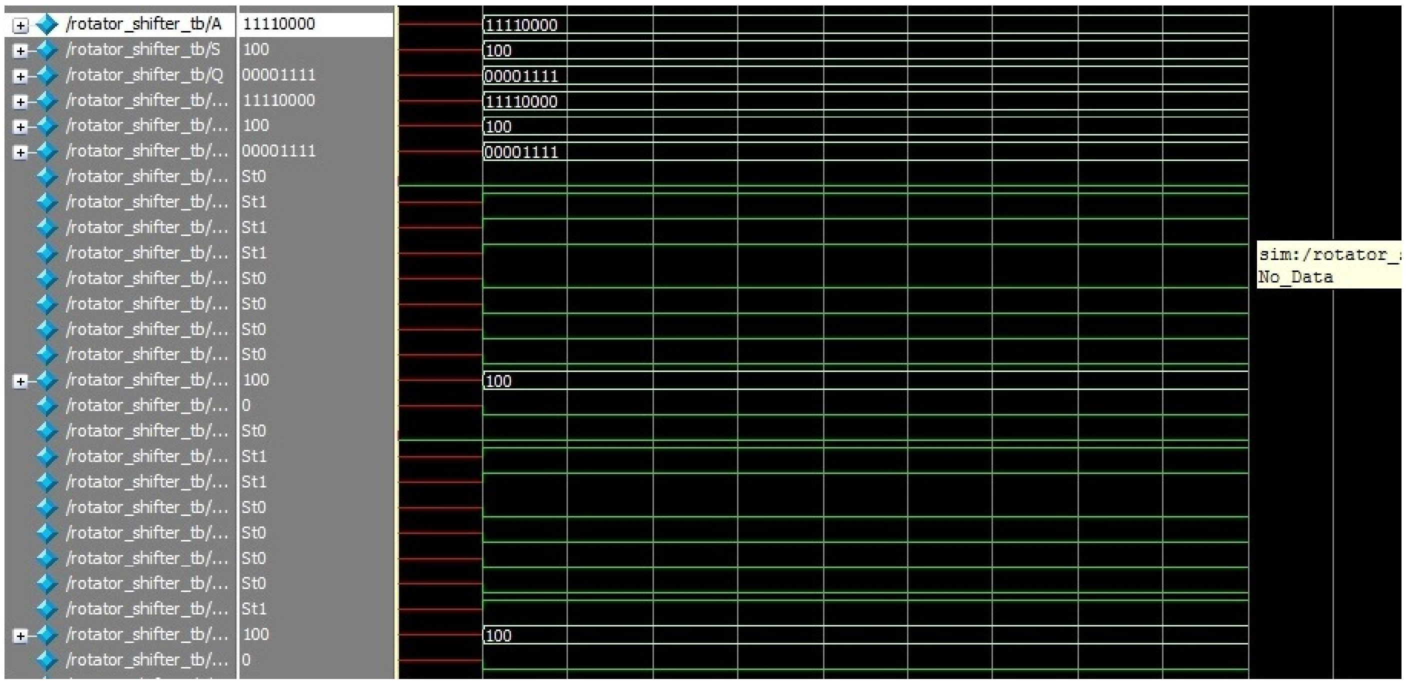
```
/shifter8_lefter_tb5...   11111111
/shifter8_lefter_tb5...   110
/shifter8_lefter_tb55/z   11000000
/shifter8_lefter_tb5...   11111111
/shifter8_lefter_tb5...   110
/shifter8_lefter_tb5...   11000000
/shifter8_lefter_tb5...   11110000
/shifter8_lefter_tb5...   11000000
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   1
/shifter8_lefter_tb5...   St1
/shifter8_lefter_tb5...   St0
```

5.3. Simulation Result (reversal)



```
/reversal_shift_tb/a      00011101
/reversal_shift_tb/left   1
/reversal_shift_tb/p      10111000
/reversal_shift_tb/r...   00011101
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   10111000
/reversal_shift_tb/r...   St0
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   1
/reversal_shift_tb/r...   St0
/reversal_shift_tb/r...   St0
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   0
/reversal_shift_tb/r...   St0
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   St1
/reversal_shift_tb/r...   1
```

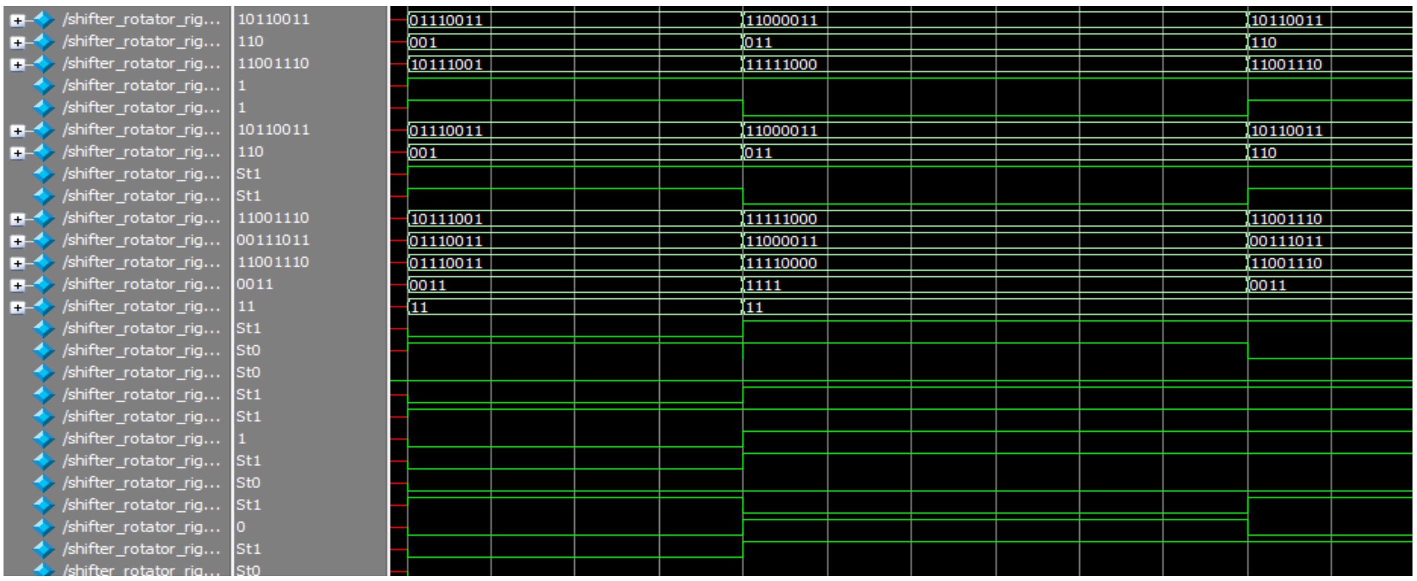
5.4. Simulation Result (rotator)



```
/rotator_shifter_tb/A     11110000
/rotator_shifter_tb/S     100
/rotator_shifter_tb/Q     00001111
/rotator_shifter_tb/...   11110000
/rotator_shifter_tb/...   100
/rotator_shifter_tb/...   00001111
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   100
/rotator_shifter_tb/...   0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St0
/rotator_shifter_tb/...   St1
/rotator_shifter_tb/...   100
/rotator_shifter_tb/...   0
```

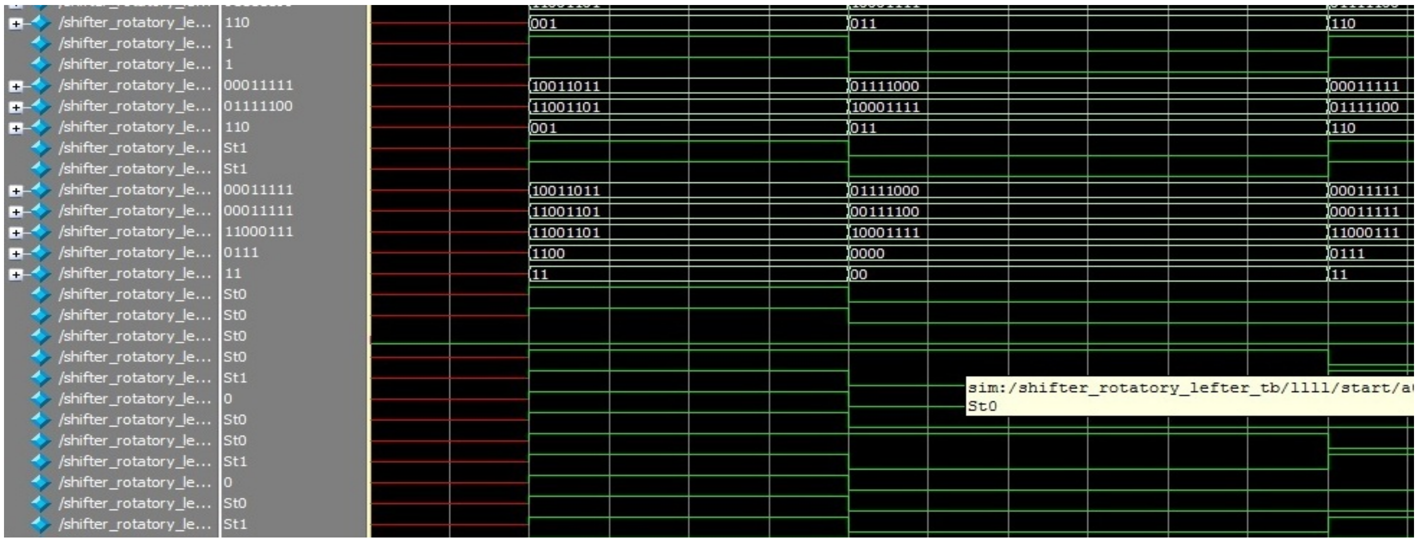
ISSN: 2348-4098
VOLUME 2 ISSUE 7 SEP-OCT 2014

5.5. Simulation Result (shift+rotate right)



```
/shifter_rotator_rig...  10110011
/shifter_rotator_rig...  110
/shifter_rotator_rig...  11001110
/shifter_rotator_rig...  1
/shifter_rotator_rig...  1
/shifter_rotator_rig...  10110011
/shifter_rotator_rig...  110
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  11001110
/shifter_rotator_rig...  00111011
/shifter_rotator_rig...  11001110
/shifter_rotator_rig...  0011
/shifter_rotator_rig...  11
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  St0
/shifter_rotator_rig...  St0
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  1
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  St0
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  0
/shifter_rotator_rig...  St1
/shifter_rotator_rig...  St0
```

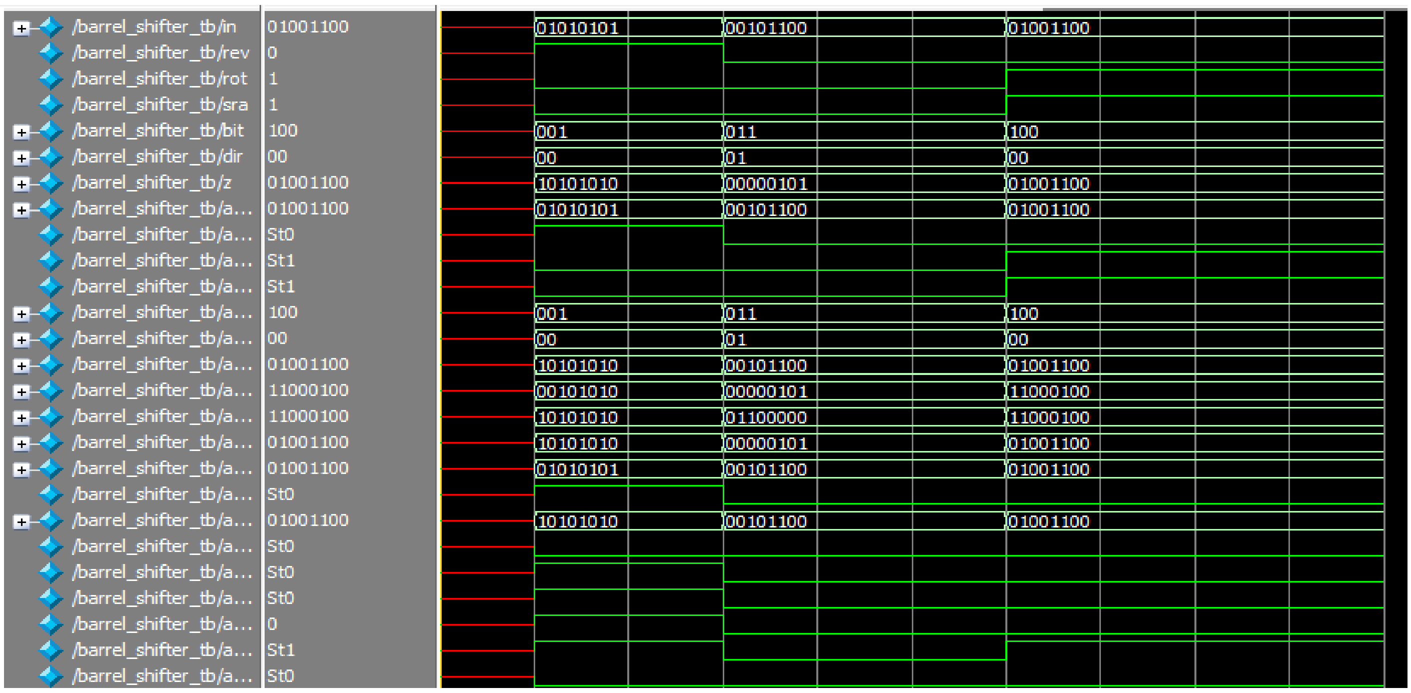
5.6. Simulation Result (shifts+rotates left)



```
/shifter_rotatory_le...  110
/shifter_rotatory_le...  1
/shifter_rotatory_le...  1
/shifter_rotatory_le...  00011111
/shifter_rotatory_le...  01111100
/shifter_rotatory_le...  110
/shifter_rotatory_le...  St1
/shifter_rotatory_le...  St1
/shifter_rotatory_le...  00011111
/shifter_rotatory_le...  00011111
/shifter_rotatory_le...  11000111
/shifter_rotatory_le...  0111
/shifter_rotatory_le...  11
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St1
/shifter_rotatory_le...  0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St1
/shifter_rotatory_le...  0
/shifter_rotatory_le...  St0
/shifter_rotatory_le...  St1

sim:/shifter_rotatory_lefter_tb/llll/start/a
St0
```

5.7. Simulation Result (barrel shifter)



```
/barrel_shifter_tb/in    01001100
/barrel_shifter_tb/rev   0
/barrel_shifter_tb/rot   1
/barrel_shifter_tb/sra   1
/barrel_shifter_tb/bit   100
/barrel_shifter_tb/dir   00
/barrel_shifter_tb/z     01001100
/barrel_shifter_tb/a...  01001100
/barrel_shifter_tb/a...  St0
/barrel_shifter_tb/a...  St1
/barrel_shifter_tb/a...  St1
/barrel_shifter_tb/a...  100
/barrel_shifter_tb/a...  00
/barrel_shifter_tb/a...  01001100
/barrel_shifter_tb/a...  11000100
/barrel_shifter_tb/a...  11000100
/barrel_shifter_tb/a...  01001100
/barrel_shifter_tb/a...  01001100
/barrel_shifter_tb/a...  St0
/barrel_shifter_tb/a...  01001100
/barrel_shifter_tb/a...  St0
/barrel_shifter_tb/a...  St0
/barrel_shifter_tb/a...  St0
/barrel_shifter_tb/a...  0
/barrel_shifter_tb/a...  St1
/barrel_shifter_tb/a...  St0
```

6. FUTURE EXPECTS

The future expects of barrel shifter is that it minimize the area and power delay of the circuit. Area and delay estimates, based on synthesis of structural level VHDL, indicate that data-reversal barrel shifters have less area than two's complement or one's complement barrel shifters and that mask-based data-reversal barrel shifters have less delay than the other designs. As the operand size increases, the delay of the shifters increases as O(log(n)) and their area increases as O(n log(n)).In to the future expectation we attach a overflow detection logic, so the data should not be waste.

7. CONCLUSION

This paper named "Barrel Shifter" undertook by the student of M.Tech (VERI LARGE SCALE INTEGERATION) THIRD SEMESTER under the guidance and support of our teacher.

The reason behind undertaking this project simply lies with the fact that there are so many circuits that have more power consumption and delay, so to minimize the area and delay we are using shifting or rotation. Here we are doing shift right logical, shift right arithmetic, rotate right, shift left logical, shift left arithmetic, and rotate left. Four different barrel shifter designs are presented and compared in terms of area and delay for a variety of operand sizes. This is also examines techniques for detecting results that overflow and results of zero in parallel with the shift or rotate operation.

To resolve this purpose we have made this very project, so that if such a kind of system is used then at least it may be able to sense the shifting or rotation and accordingly necessary conditions can be undertaken.

REFERENCES

1. www.realwordtech.com.

2. Barrel Shifter or Multiply/Divide IC Structure.

3. Circuit for Rotating, Left Shifting, or Right Shifting Bits.

4. Multilevel Barrel Shifter for CORDIC Design.

5. High-Speed Barrel Shifter.