1 Introduction

1.1 Operations

Barrel shifters designed perform a maximum of four operations: rotate right, rotate left, shift logical left, shift logical right. Also arithmetic shift operations can be performed using barrel shifters but are not used here.

1.2 Right Rotate

For a right rotation by k bits on an n bit data the lower k bits are placed in the upper k bits of the result whereas the remaining (n-k) bits occupy the lower remaining bits of the result. Consider data as shown below

a	b	c	d	e	f	g	h

Right rotate by 2 results in

g	h	a	b	c	d	e	f

1.3 Left Rotate

For a left rotation by k bits on an n bit data upper k bits are placed in the lower k bits of the result whereas the remaining (n-k) bits occupy the upper remaining bits of the result. Consider data as shown below

a	b	c	d	e	f	g	h

Left rotate by 2 results in

c	d	e	f	g	h	a	b

1.4 Logical Right Shift

For a logical right shift by k bits on an n bit data the lower k bits are lost and k zeros fill the upper k bits of the result whereas the remaining (n-k) bits occupy the lower remaining bits of the result. Consider data as shown below

a	b	c	d	e	f	g	h

Logical right shift by 2 results in

0	0	a	b	c	d	e	f

1.5 Logical Left Shift

For a logical left shift by k bits on an n bit data the upper k bits are lost and k zeros fill the lower k bits of the result whereas the remaining (n-k) bits occupy the upper remaining bits of the result. Consider data as shown below

a	b	c	d	e	f	g	h

Logical left shift by 2 results in

c	d	e	f	g	h	0	0

1.6 Area Delay Product

Since Multiplexor (Mux) is used in all designs a Mux rather than a gate is considered as fundamental unit of logic here. All the designs use 2:1 Mux only except for Masking based rotator where in addition to 2:1 Mux a 4:1 Mux is used to perform one of the four operations mentioned above.

Area is defined as number of Mux in the design. Latency (Delay) is defined as number of levels of Mux through which input has to pass to reach output.Area delay product is defined as product of area and delay.

Area Delay Product = Area * Delay

2 Unidirectional Designs

2.1 Logical Right Shifter

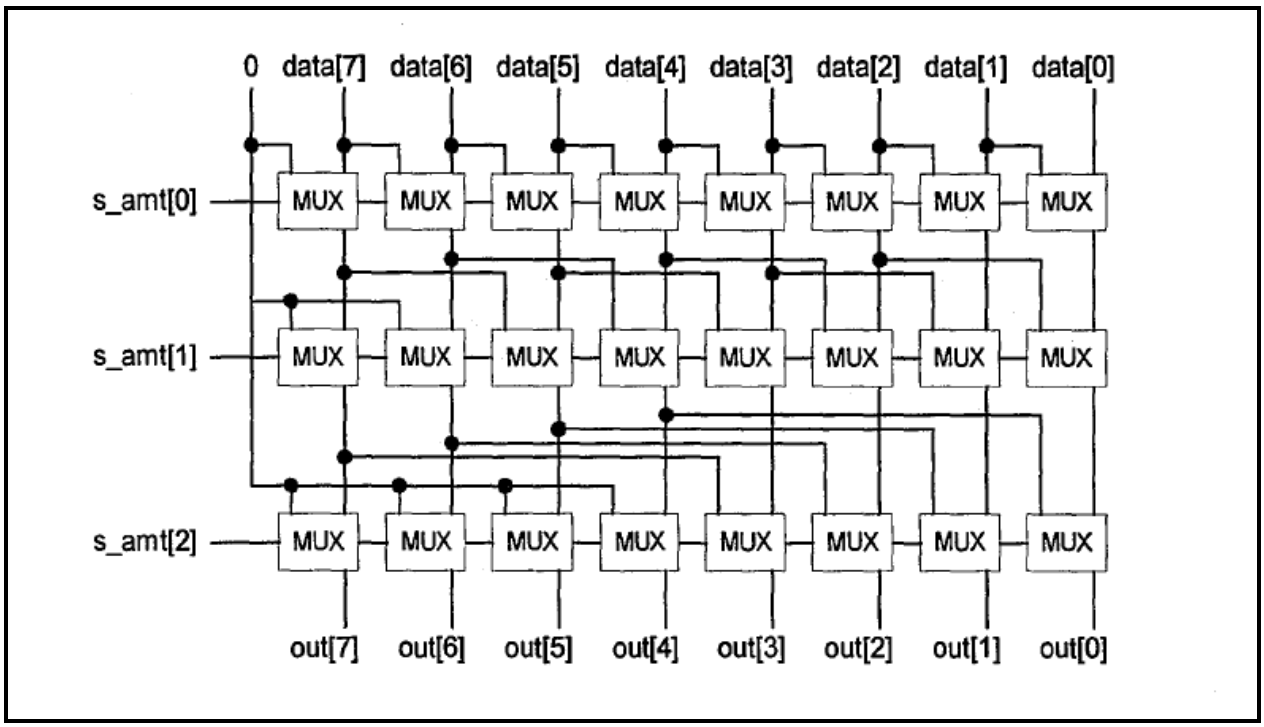

Figure 2.1 Logical Right Shifter

First row corresponds to a shift of 1, second row to a shift of 2 and third row to a shift of 4. Interconnects shown above enable these desigred shifts in each row. As required the higher order bits are filled with zeros. The select lines s_amt denotes the desired shift. If the select line is one the corresponding shift for the row occurs or else the data is passed as it is.

2.2 Right Rotator

The only difference lies in the Hardwired zeros. Instead the lost bits are rotated back in upper positions as shown below. The operation is similar to that of right shifter.

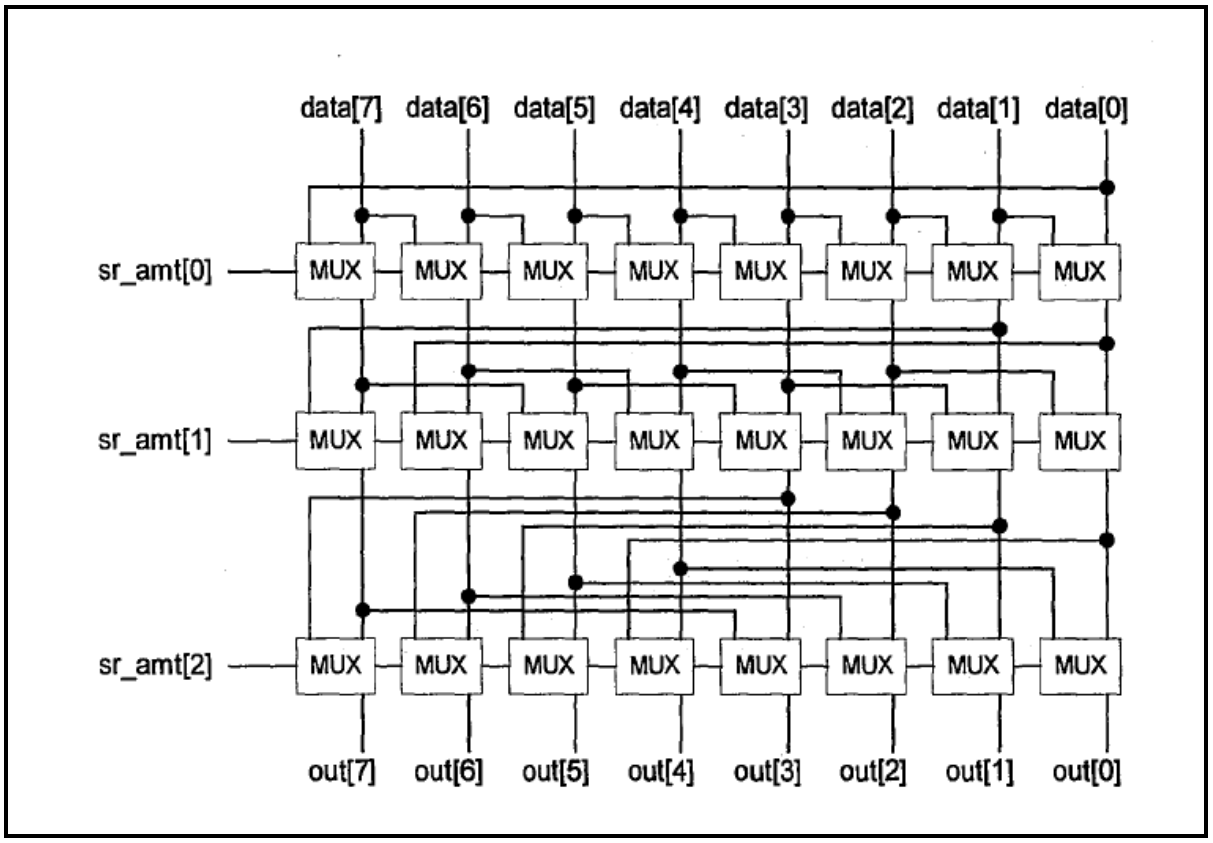

Figure 2.2 Right Rotator

2.3 Left Shifters and Rotators

The only difference lies in the interconnects to ensure left shift and rotate operations instead of right shift and rotate opeartions.

3 Bidirectional Designs

3.1 Series Bidirectional Logical Shifter

It uses a logical right shifter and a logical left shifter connected in series as shown. Input s_amt is given as select lines to both the shifter via and gates and input right and right bar as shown.

right =1	Upper right shifter is active and lower left shifter merely passes the data	Logical right shift occurs
right = 0	Lower left shifter is active and the upper right shifter merely passes the data	Logical left shift occurs

Table 3.1.1 Operation of Series Bidirectional Logical Shifter

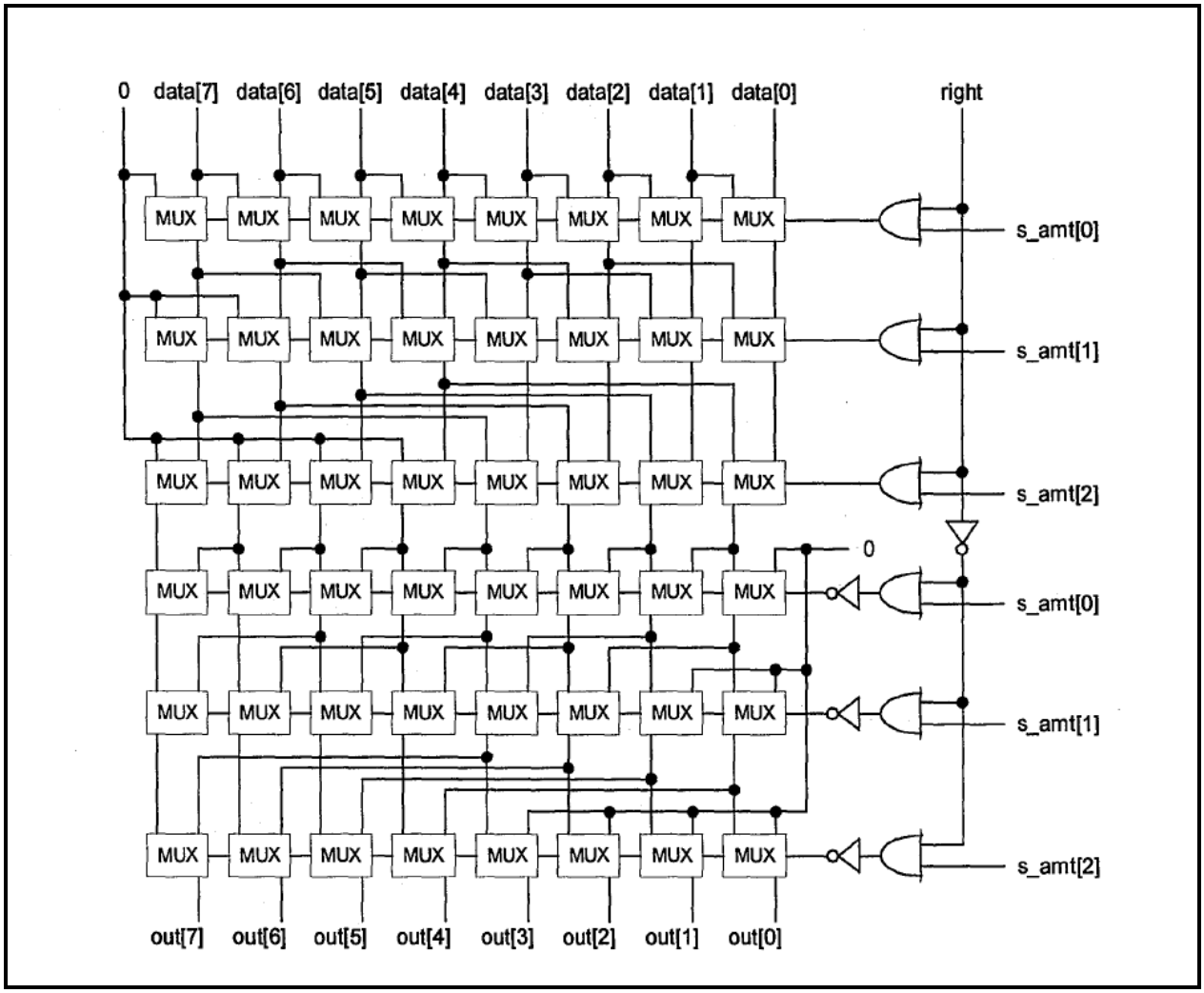

Figure 3.1 Series Bidirectional Logical Shifter

Parameter	Generalized Formula	n=8
Area	$2n\log_2(n)$	48
Delay	$2\log_2(n)$	6
Area Delay Product	$8n\log_2(n)$	288

Table 3.1.2 Area Delay Product for Series Bidirectional Logical Shifter

3.2 Parallel Bidirectional Logical Shifter

It uses a logical right shifter and a logical left shifter connected in parallel as shown. Input sr_amt gives the desired shift as select lines to both right and left shifter units. Data is fed simultaneously to both the units which are always enabled and give their respective results. A final row of 8 Mux selects one of the two outputs of logical right and logical left shifter using an input signal right as its select line.

right =1	Right and left shifter both active but output of right shifter is selected	Logical right shift occurs
right = 0	Right and left shifter both active but output of left shifter is selected	Logical left shift occurs

Table 3.2.1 Operation of Parallel Bidirectional Logical Shifter

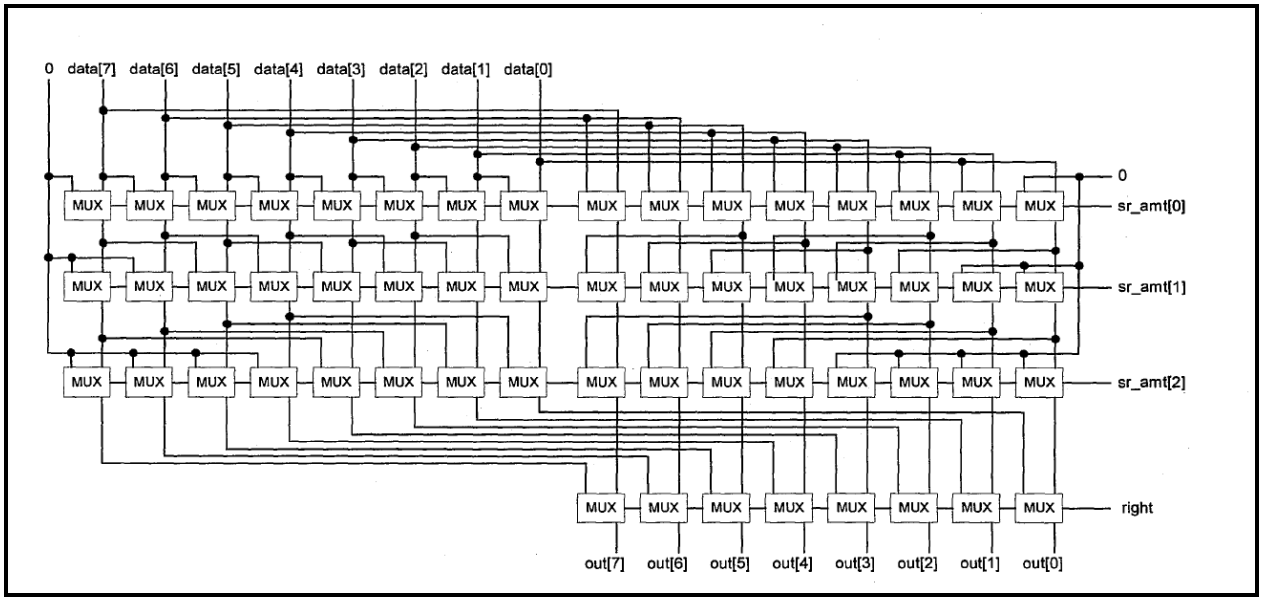

Figure 3.2 Parallel Bidirectional Logical Shifter

Parameter	Generalized Formula	n=8
Area	$2n\log_2(n)+n$	56
Delay	$\log_2(n)+1$	4
Area Delay Product	$7n\log_2(n)+n$	224

Table 3.2.2 Area Delay Product for Parallel Bidirectional Logical Shifter

3.3 Data Reversal Bidirectional Logical Shifter

Previous two designs use two unidirectional shifters to accomplish a bidirectional shifter. Such designs although serve the purpose are expensive. Hence an optimum solution would be the one in which a unidirectional shifter through some mechanism provide a bidirectional shift. Hence a unidirectional shifter must adjust the data such that a shift in supported direction gives the desired shift in the opposite direction. Data Reversal Bidirectional Logical Shifter provides such a mechanism.

Solution is to reverse the order of the bits in the data if it is to be shifted in the direction opposite to that which the single shifter supports both before and after the shifter. The data is manipulated so that a right shift with two bit reversals performs the desired left shift. For shift in the desired direction the row of Mux before and after right shifter passes its input as it is without reversal. Consider output of first row of Mux as step I, output of right shifter as step II and final output as step III for the case of left shift by 1.

Input data

a	b	c	d	e	f	g	h

Output of step I for left=1

h	g	f	e	d	c	b	a

Output of step II for s_amt=001

0	h	g	f	e	d	c	b

Output of step III for left=1

b	c	d	e	f	g	h	0

Table below shows that delay of this design is larger than that of the Parallel Bidirectional Logical Shifter, but less than that of the Series Bi-directional Logical Shifter. Also area of this design is less than the others. The area-delay product shows that this method is preferred as it provides a good trade-off between area and delay.

Parameter	Generalized Formula	n=8
Area	$n\log_2(n)+2n$	40
Delay	$\log_2(n)+2$	5
Area Delay Product	$6n\log_2(n)+4n$	200

Table 3.3 Area Delay Product for Data Reversal Bidirectional Logical Shifter

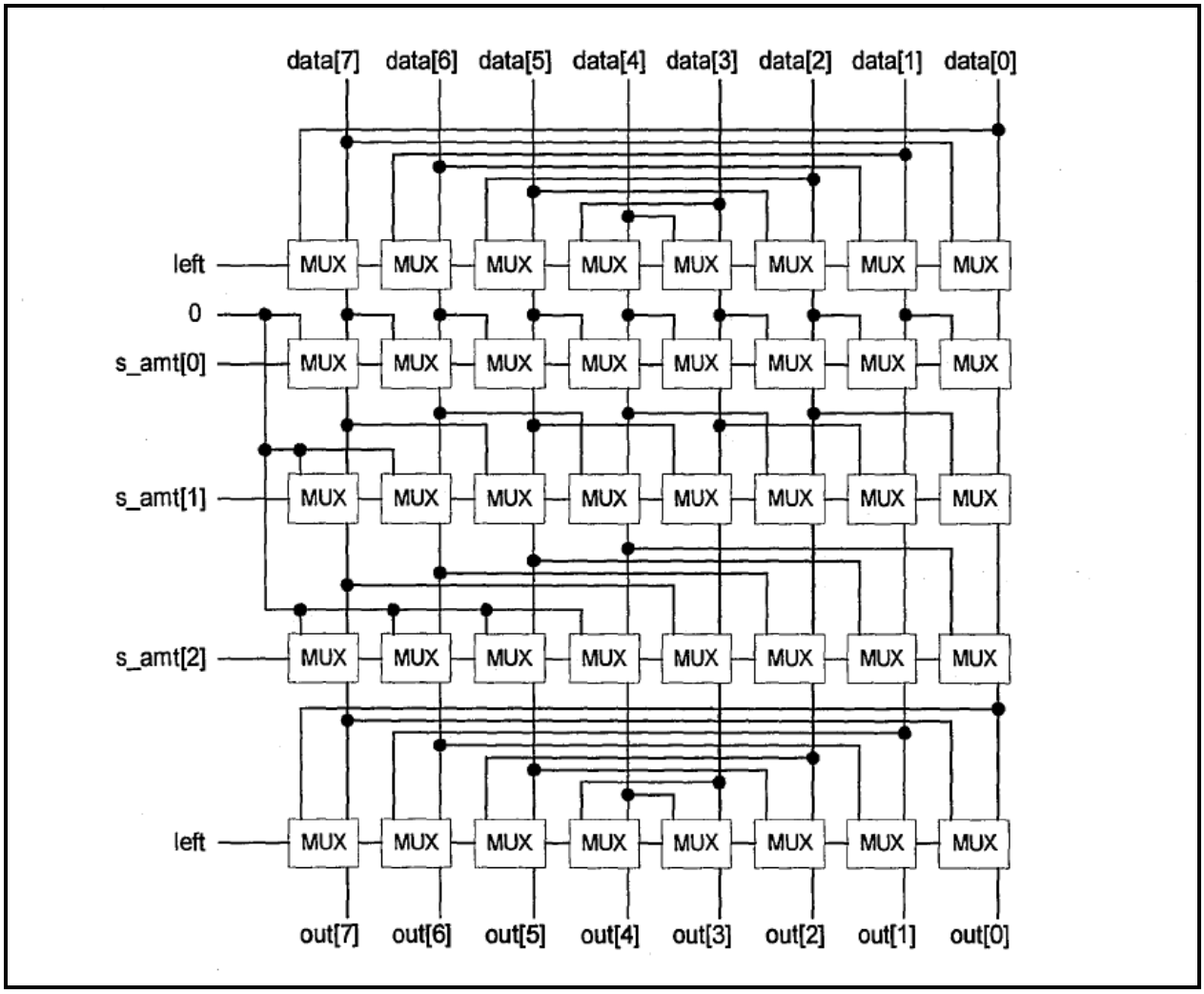

Figure 3.3 Data Reversal Bidirectional Logical Shifter

3.4 One's Complement Bi-directional Rotator

Just as the aforementioned Data Reversal Bi-directional Logical Shifter manipulates the data so as to require only a single shifter, the One's Complement Bi-directional Rotator manipulates the rotate amount to accomplish the same goal. This is done by supporting one rotate direction directly and handling the opposite case by calculating the amount that the data must be rotated in the supported direction to emulate a rotate in the opposite direction.

One's complement of the rotate amount is determined. This computation is more easily accomplished, by Ex-oring with left input so that the inverse of the rotate amount is taken for a rotate left. An extra rotate-by-one stage is added to the rotator for a rotate left when the unit supports rotate right. Consider output of first row of Mux as step I and output of right rotator as step II for the case of left rotate by 1. For right rotation left=0 and hence first row of Mux just passes the data as it is. Also the rotator r_amt is passed as it is to the right rotator and the desired right rotation takes place.

Ex-or gates are ignored in Area and Delay calculations.

Input data

a	b	c	d	e	f	g	h

Output of step I for left=1

h	a	b	c	d	e	f	g

Output of step II for r_amt=001 and left=1

b	c	d	e	f	g	h	a

Parameter	Generalized Formula	n=8
Area	$n\log_2(n)+n$	32
Delay	$\log_2(n)+1$	4
Area Delay Product	$4n\log_2(n)+n$	128

Table 3.4 Area Delay Product for One's Complement Bi-directional Rotator

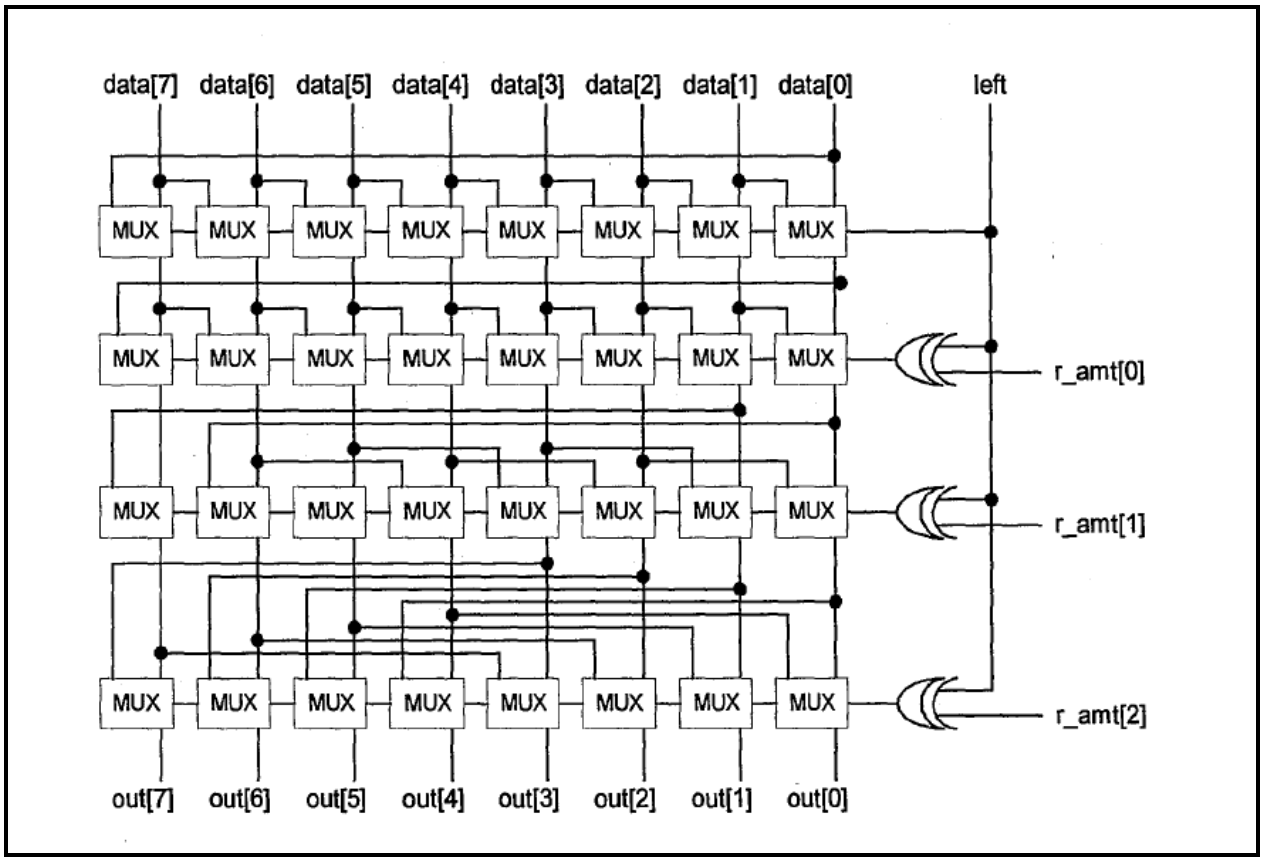

Figure 3.4 One's Complement Bi-directional Rotator

4 Masking Rotating Shifter

Previous designs offer insight into the creation of an efficient bi-directional mechanism for shifts and rotates, they do not offer insight into how all the required operations can be implemented in a single unit. The Masking Rotating Shifter offers that. By using masks to manipulate the results of a rotate, it is possible to emulate logical and arithmetic shifts.

First we use One's Complement Bi-directional Rotator to get the desired right or left rotated result. Now in order to get logical shifts we define mask for right and left logical shift independently. Bitwise AND of the mask with the rotated result gives the desired logical shift.

We define count as the desired number either shift or rotate in a desired direction. Mask is obtained by setting the count number of high order bits to 1 and rest to 0. Ones complement of the mask gives inverse mask. Right mask is same as inverse mask. Left mask is left rotated result of inverse mask determined by count.

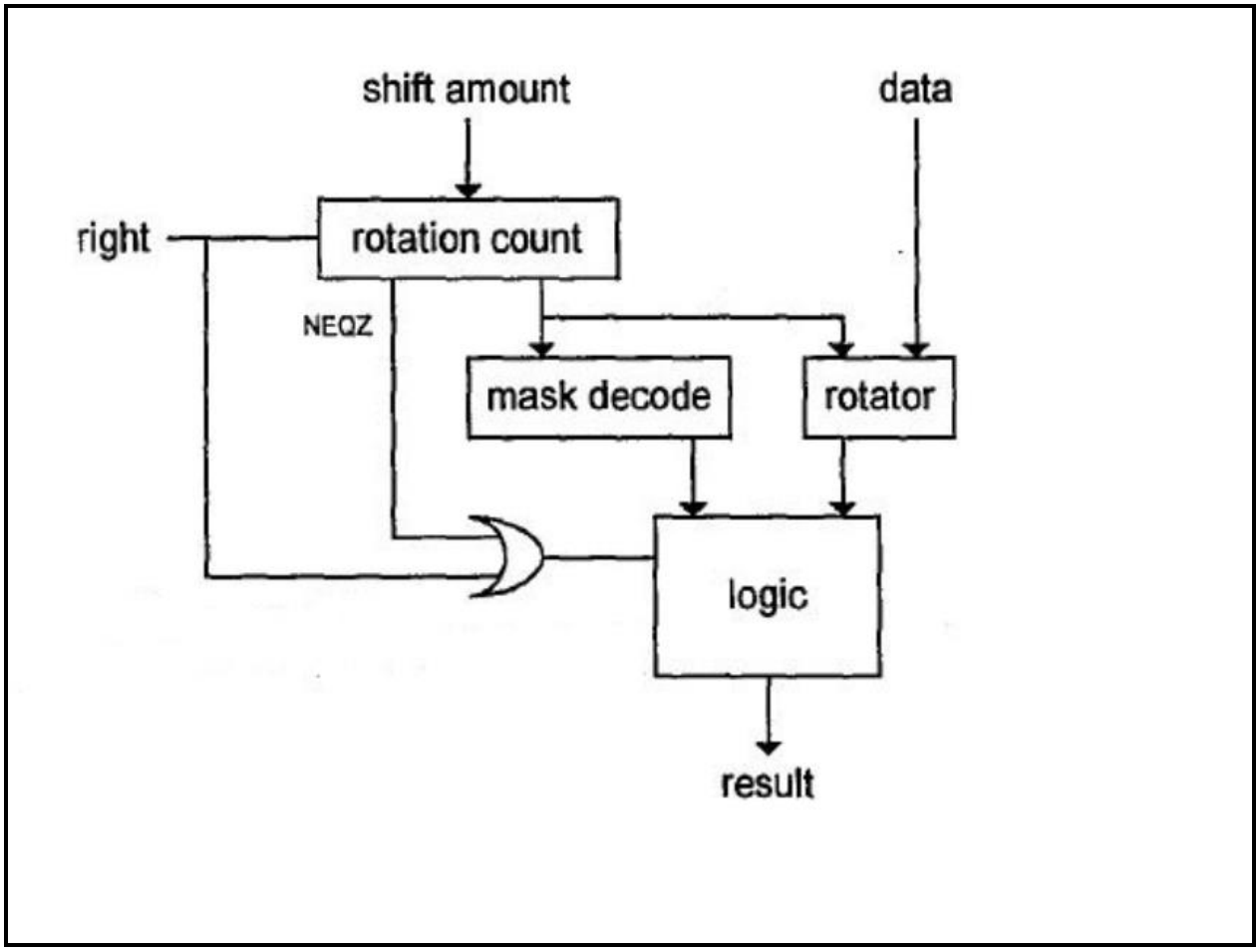

Figure 4 Masking Rotating Shifter

Hence effectively we require a One's Complement Bi-directional Rotator, left rotator and few AND, NOT gates to achieve all four operations viz. Right Rotate, Left Rotate, Logical Right Shift and Logical Left Shift.

It is difficult to compare this design in terms of Area Delay Product as in other designs due to structure of this barrel shifter. But the approach to this design, however, is the most useful piece of information gained since it offers a method to design a complete barrel shifter, not yet explored.

5 Results

Comparisons using Area Delay Product for the designs mentioned above is presented in tables 3.1.2, 3.2.2, 3.3 and 3.4. Synthesis was performed on structural level Verilog modules using ISE 14.7 by Xilinx and simulation results were observed on ISIM Xilinx.

Simulation results for the designs mentioned above are given here.

5.1 Logical Right Shifter

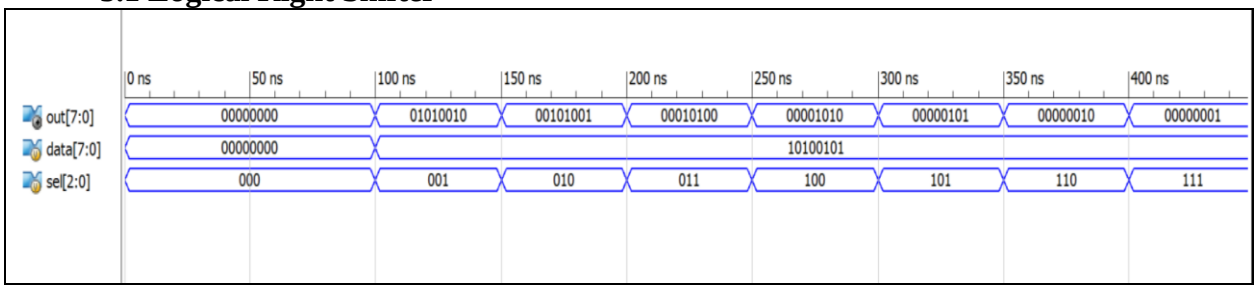

Figure 5.1 Simulation for Logical Right Shifter

5.2 Right Rotator

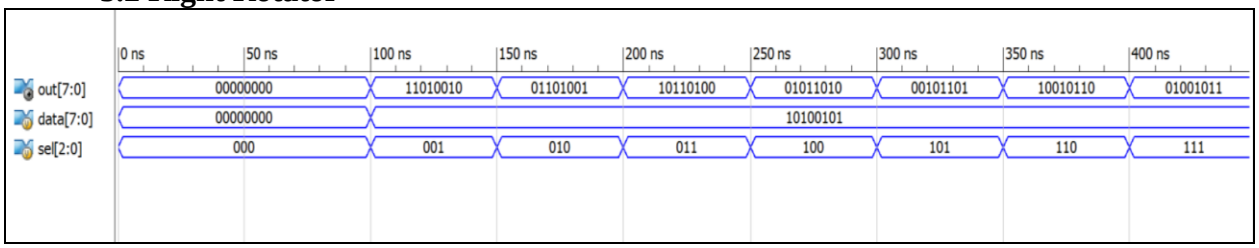

Figure 5.2 Simulation for Right Rotator

5.3 Logical Left Shifter

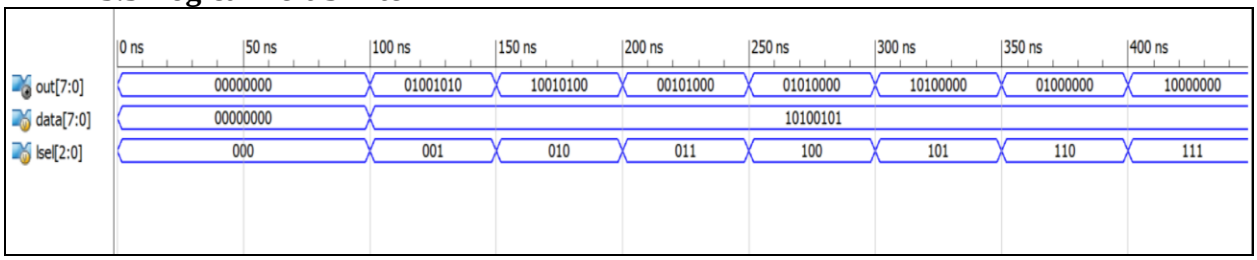

Figure 5.3 Simulation for Logical Left Shifter

5.4 Left Rotator

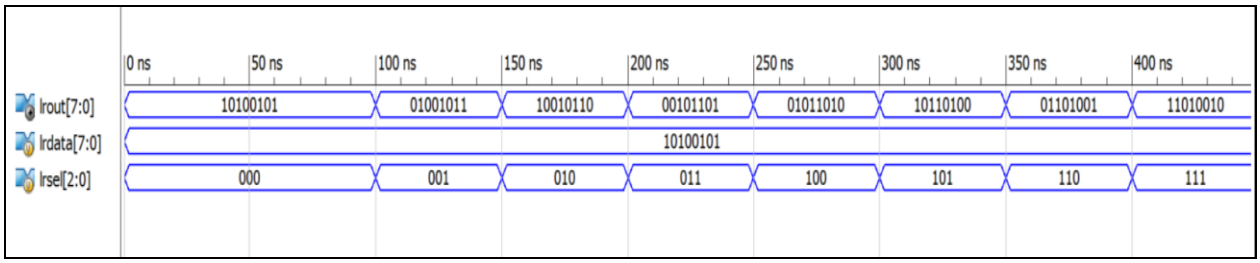

Figure 5.4 Simulation for Left Rotator

5.5 Series Bidirectional Logical Shifter

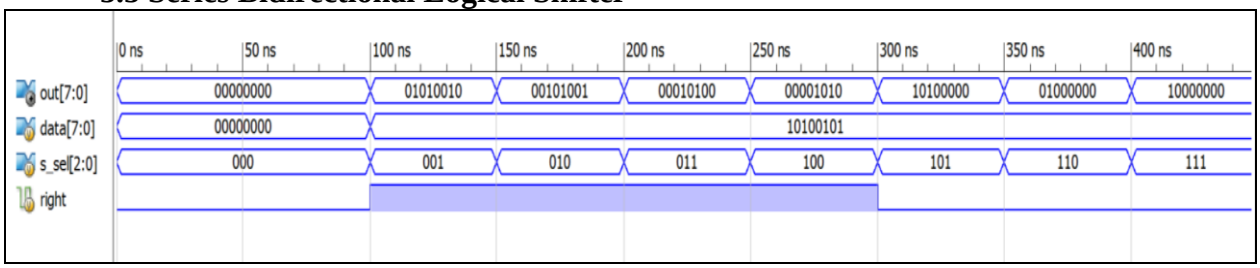

Figure 5.5 Simulation for Series Bidirectional Logical Shifter

5.6 Parallel Bidirectional Logical Shifter

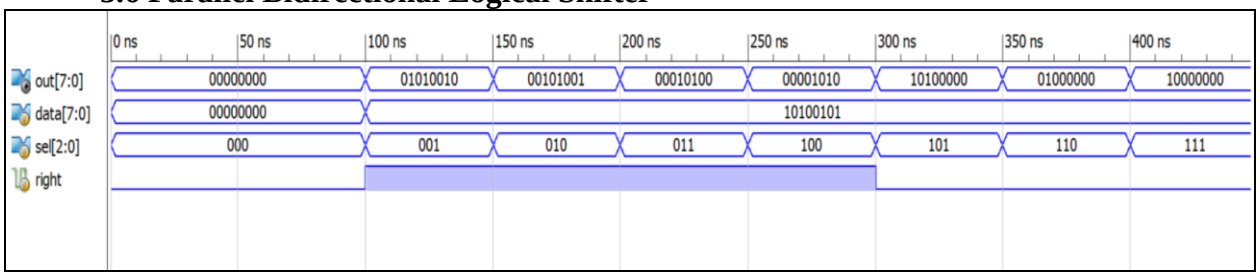

Figure 5.6 Simulation for Parallel Bidirectional Logical Shifter

5.7 Data Reversal Bidirectional Logical Shifter

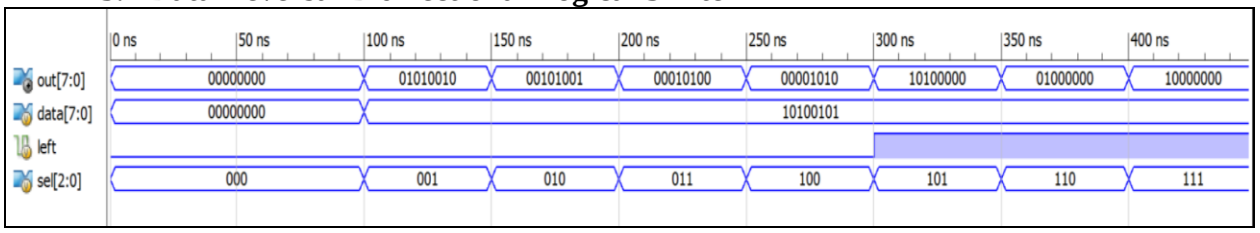

Figure 5.7 Simulation for Data Reversal Bidirectional Logical Shifter

5.8 One's Complement Bi-directional Rotator

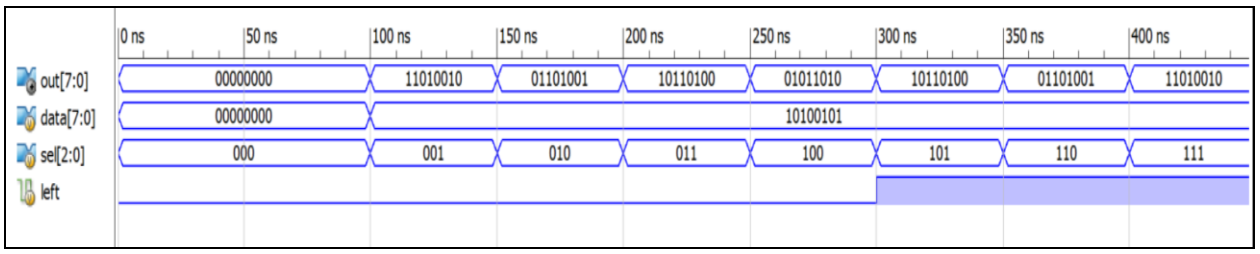

Figure 5.8 Simulation for One's Complement Bi-directional Rotator

5.9 Masking Rotating Shifter

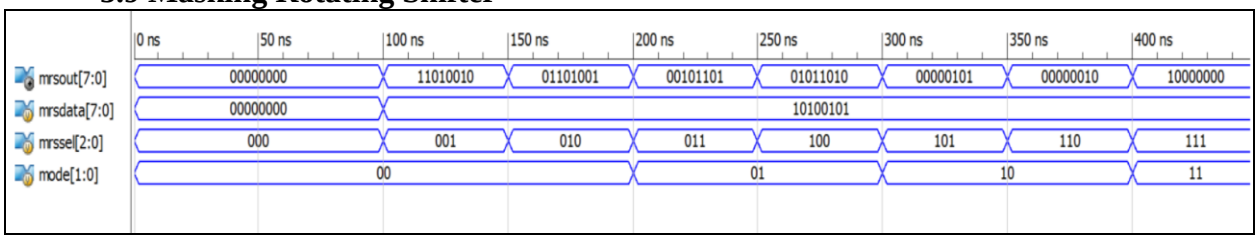

Figure 5.9 Simulation for Masking Rotating Shifter

6 Conclusion

Area Delay Product proves to be an efficient parameter to determine the cost and select an appropriate design suitable to ones need and application. A complete barrel shifter provides means to achieve all four operations viz. Right Rotate, Left Rotate, Logical Right Shift and Logical Left Shift as in case of Masking Rotating Shifter. But all of this operations might not be always desired. In such cases other mentioned designs can be used to suit ones need.

7 Future Work

Having designed a Barrel Shifter, I hope to take up the next step towards building a microprocessor. As in case of ARM architecture Barrel Shifter can be used outside the Arithmetic and Logical Unit (ALU) to provide fast and efficient shift and rotate results frequently required in processor applications. Designing a pipelined architecture, Fast adders, Multipliers are some of the tasks we hope to take in future. Also instead of just limiting to terminal level of design process using a Hardware Description Language I hope to work at layout level too using CAD tools like Microwind in future.

8 References

8.1 Theses and Dissertations

2001, Barrel shifter design, optimization, and analysis by Matthew Rudolf Pillmeier *Lehigh University*

8.2 Books

Digital Design With an Introduction to the Verilog HDL, FIFTH EDITION, M. Morris Mano and Michael D. Ciletti, Pearson.

8.3 Manuals

ISE Simulator (ISim) In-Depth Tutorial UG682 (v1.0) April 27, 2009, Xilinx

8.4 Websites and Forums

http://www.asic-world.com/verilog/
https://electronics.stackexchange.com/