Realizing Adders, MUX in structural and RTL Verilog

CASEST

हैदराबाद विश्वविद्यालय
University of Hyderabad

Lab Report No. 1

MV407: IC Design Lab-1 (Digital)

Submitted by:

Full Name	Enrollment No.
Abhinav M	23PMMT11(@uohyd.ac.in)

Lab Instructor: Dr. Bhawna Gomber

Teaching Assistant: Ms. Bhargavi

Center for Advanced Studies in Electronics Science and Technology
University of Hyderabad
Hyderabad, India
September 23, 2023

Contents

1 Introduction 1

2 Half adder (structural) 1
2.1 Objectives . . . 1
2.2 Schematic . . . 1
2.3 Truth table . . . 2
2.4 Code . . . 3
2.5 Elaborated design . . . 4
2.6 Timing diagram . . . 4
2.7 Utilization summary . . . 4
2.8 Remarks . . . 4

3 Full adder (structural) 5
3.1 Objectives . . . 5
3.2 Schematic . . . 5
3.3 Truth table . . . 5
3.4 Code . . . 6
3.5 Elaborated design . . . 8
3.6 Timing diagram . . . 8
3.7 Utilization summary . . . 8
3.8 Remarks . . . 8

4 Full adder using half adder (structural) 9
4.1 Objectives . . . 9
4.2 Schematic . . . 9
4.3 Truth table . . . 9
4.4 Code . . . 9
4.5 Elaborated design . . . 11
4.6 Timing diagram . . . 11
4.7 Utilization summary . . . 11
4.8 Remarks . . . 12

5 Half adder (RTL) 13
5.1 Objectives . . . 13
5.2 Schematic . . . 13
5.3 Truth table . . . 13
5.4 Code . . . 13
5.5 Elaborated design . . . 14
5.6 Timing diagram . . . 14
5.7 Utilization summary . . . 15
5.8 Remarks . . . 15

6 Full adder(structural) using half adder(RTL) 16
6.1 Objectives . . . 16
6.2 Schematic . . . 16
6.3 Truth table . . . 16
6.4 Code . . . 16
6.5 Elaborated design . . . 17
6.6 Timing diagram . . . 18
6.7 Utilization summary . . . 18
6.8 Remarks . . . 18

7 2x1 MUX using structural and RTL **19**
7.1 Objectives 19
7.2 Schematic 19
7.3 Truth table 19
7.4 Code 20
7.5 Elaborated design 22
7.6 Timing diagram 22
7.7 Utilization summary 23
7.8 Remarks 23

8 4x1 MUX using 2x1 MUX in RTL **24**
8.1 Objectives 24
8.2 Schematic 24
8.3 Truth table 24
8.4 Code 25
8.5 Elaborated design 26
8.6 Timing diagram 27
8.7 Utilization summary 27
8.8 Remarks 27

9 2x1 MUX using input A, B as one vector with 2 bits defined **28**
9.1 Objectives 28
9.2 Schematic 28
9.3 Truth table 28
9.4 Code 28
9.5 Elaborated design 29
9.6 Timing diagram 29
9.7 Utilisation Summary 30
9.8 Remarks 30

10 References **31**

A Appendix A: GitHub Repo **32**

1 | Introduction

This lab report details the implementation of basic adders and multiplexers in Verilog Hardware Description Language(HDL) in structural and Register Transfer Level (RTL) / Dataflow paradigms. The code was written and synthesized on Xilinx Vivado v2020 on a Zynq 7000 xc7c020clg484-1 platform with 53200 Look Up Tables (LUTs) and 400 Input Output Blocks (IOBs). This report is a part of the MV407 course at CASEST, University of Hyderabad (2023).

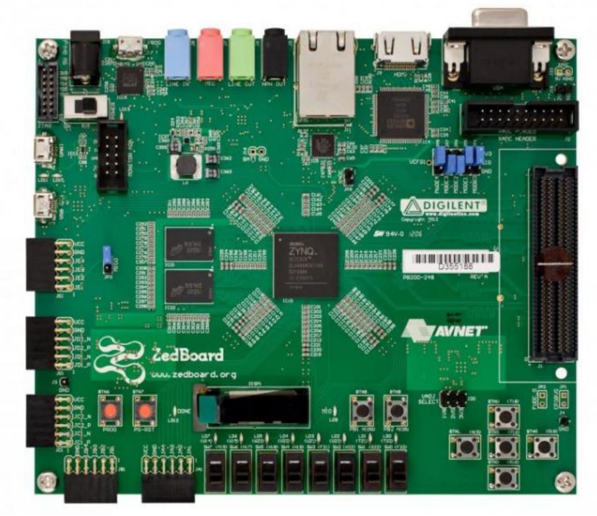


Figure 1.1: Zedboard: A common FPGA platform powered by the Zynq 7000 family.

2 | Half adder (structural)

13-09-2023

2.1 | Objectives

- To design a half adder in structural Verilog and verify its functionality using a testbench.

2.2 | Schematic

The half adder takes two 1-bit binary inputs, A and B, and produces two 1-bit outputs: the sum (S) and the carry (C). For example, when A=1 and B=0, the half adder outputs S=1 and C=0, demonstrating basic binary addition.

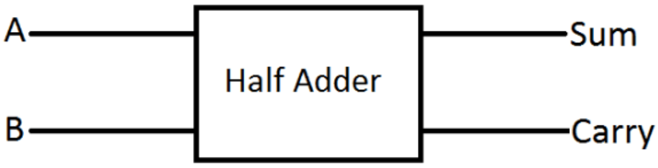


Figure 2.1: Block diagram of the half adder.

2.3 | Truth table

The truth table can be used to derive the Boolean expression:

A	B	S (Sum)	C (Carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 2.1: Truth table for the half adder.

2.3.1 | Reduced Boolean expression using K-Map

For Sum (S)

		A	
		0	1
B	0	0	1
	1	1	0

$$S = \overline{A}B + A\overline{B} + AB$$

$$S = A \oplus B$$

For Carry (C)

		A	
		0	1
B	0	0	0
	1	0	1

$$C = A \cdot B$$

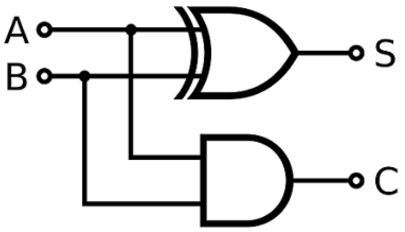


Figure 2.2: Realization of half adder using OR and XOR gates.

2.4 | Code

2.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 02:56:56 PM
// Design Name:
// Module Name: ha
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ha( //better to write outputs first then inputs
    input s,
    input c,
    output a,
    output b
    );
    xor xr_1 (s,a,b); // A XOR B = SUM
    and an_1 (c,a,b); // A.B = CARRY

endmodule
```

2.4.2 | Testbench

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 03:40:37 PM
// Design Name:
// Module Name: test_ha
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test_ha();
    reg a,b;
    wire  s,c;
    ha uut(s,c,a,b);

    initial
    begin
    #10 a=0;b=0;
    #10 a=0;b=1;
    #10 a=1;b=0;
    #10 a=1;b=1;


    #10 $finish;
    end

endmodule
```

2.5 | Elaborated design

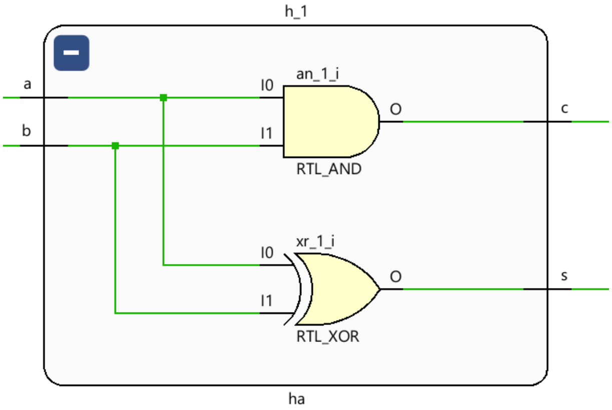
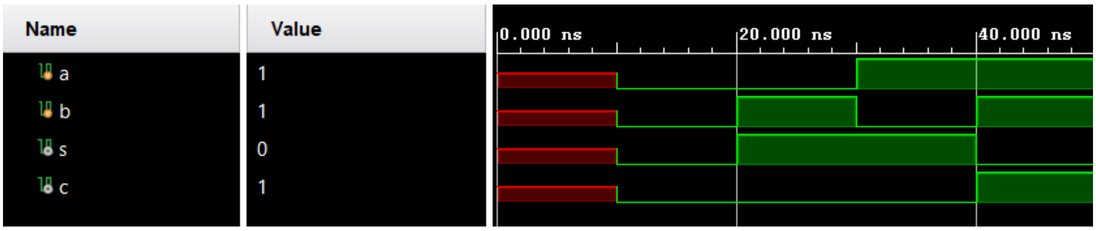


Figure 2.3: Elaborated design of half adder.

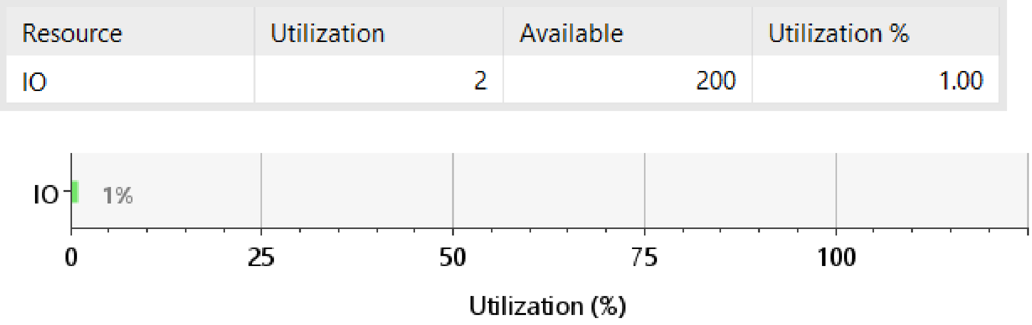
2.6 | Timing diagram



Name	Value
a	1
b	1
s	0
c	1

Figure 2.4: Timing diagram of the half adder obtained after simulating the testbench.

2.7 | Utilization summary



Resource	Utilization	Available	Utilization %
IO	2	200	1.00

Figure 2.5: Resource utilization summary of the synthesized design.

2.8 | Remarks

The design and testing of half adder in structural Verilog has been carried out. The device is showing expected behaviour.

3 | Full adder (structural)

13-09-2023

3.1 | Objectives

- To design a full adder in structural Verilog and verify its functionality using a testbench.

3.2 | Schematic

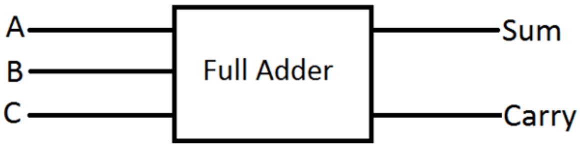


Figure 3.1: Block diagram of the full adder.

3.3 | Truth table

The truth table can be used to derive the Boolean expression:

A	B	C_{in} (Carry in)	S (Sum)	C_{out} (Carry out)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 3.1: Truth table for the full adder.

3.3.1 | Reduced Boolean expression

For Sum (S)

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = A \cdot \overline{B} \cdot \overline{C_{in}} + \overline{A} \cdot B \cdot \overline{C_{in}} + \overline{A} \cdot \overline{B} \cdot C_{in} + A \cdot B \cdot C_{in}$$

Simplifying:

$$S = A \oplus B \oplus C$$

For Carry (C)

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$$

Simplifying:

$$C_{out} = A \cdot B + C_{in}(A \oplus B)$$

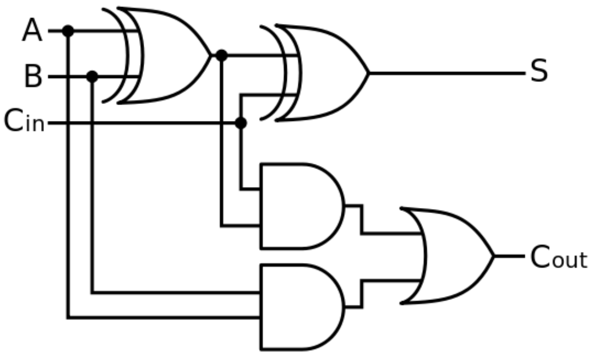


Figure 3.2: Realization of half adder using AND, OR and XOR gates.

3.4 | Code

3.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 02:55:03 PM
// Design Name:
// Module Name: fa
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

```

```verilog

module fa(
    output s,
    output c_out,
    input a,
    input b,
    input c_in
    );
    wire w_1,w_2,w_3,w_4;

    xor x_1(w_1, a,b); //A XOR B
    xor x_2(s, w_1,c_in); // A XOR B XOR C_IN = SUM

    and a_1(w_3,w_1,c_in); // (A XOR B) .C_IN
    and a_2(w_4,a,b); // A.B
    or o_1(c_out,w_3,w4); // // (A XOR B) .C_IN + AB = C_OUT

endmodule
// SUM = A XOR B XOR C_IN | C_OUT = (A XOR B) C_IN + A.B
```

3.4.2 | Testbench

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 03:32:44 PM
// Design Name:
// Module Name: test_fa
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test_fa(

    );
    reg a,b,c_in;
    wire  s,c_out;
    fa uut(s,c_out,a,b,c_in);

    initial
    begin
    #10 a=0;b=0;c_in=0;
    #10 a=0;b=0;c_in=1;
    #10 a=0;b=1;c_in=0;
    #10 a=0;b=1;c_in=1;
    #10 a=1;b=0;c_in=0;
    #10 a=1;b=0;c_in=1;
    #10 a=1;b=1;c_in=0;
    #10 a=1;b=1;c_in=1;

    #10 $finish;
    end

endmodule
```

3.5 | Elaborated design

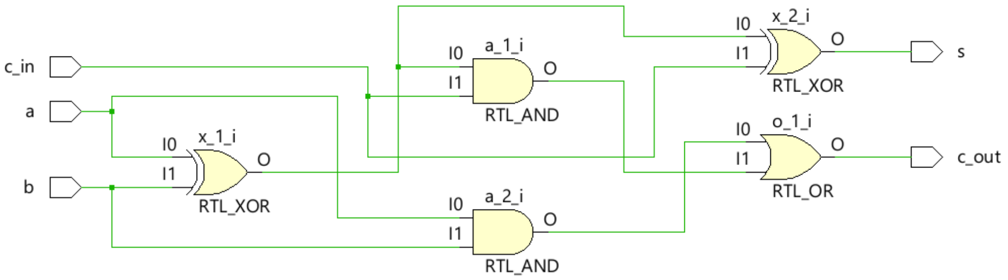
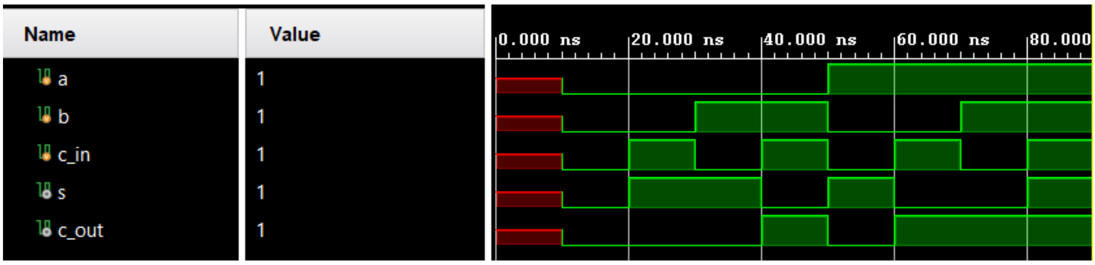


Figure 3.3: Elaborated design of the full adder.

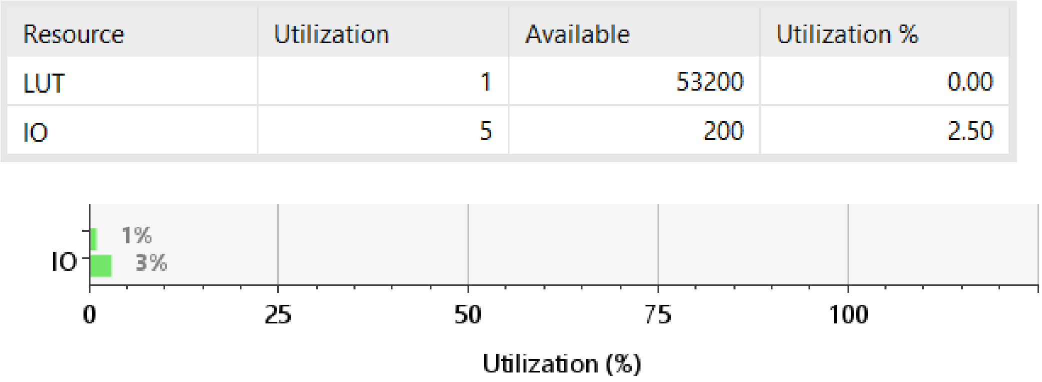
3.6 | Timing diagram



Name	Value
a	1
b	1
c_in	1
s	1
c_out	1

Figure 3.4: Timing diagram of the full adder obtained after simulating the testbench.

3.7 | Utilization summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	5	200	2.50

Figure 3.5: Resource utilization summary of the synthesized design.

3.8 | Remarks

The design and testing of full adder in structural Verilog has been carried out. The device is showing expected behaviour.

4 | Full adder using half adder (structural)

13-09-2023

4.1 | Objectives

- To design a full adder using the half adder module from Half adder (structural) and verify its functionality.

4.2 | Schematic

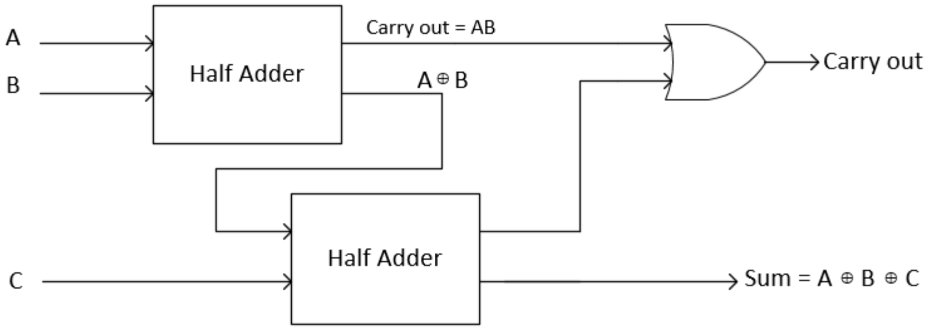


Figure 4.1: Block diagram of a full adder using two half adders.

4.3 | Truth table

Truth table for the full adder is given in Table 3.1.

4.3.1 | Reduced Boolean expression

Reduced Boolean expression for the full adder is given in Section 3.3.1.

4.4 | Code

4.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 02:57:44 PM
// Design Name:
// Module Name: fa_ha
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module fa_ha(
    input a,
    input b,
    input c_in,
    output s,
    output c_out
    );
    wire s_1,c_1,c_2;

    ha h_1(a,b,s_1,c_1); // OUTPUT OF FIRST HA

    ha h_2(s_1,c_in,s,c2); // SUM FROM SECOND HA = FINAL SUM

    or o_1(c_out, c_1,c_2); // ADD CARRYS = FINAL CARRY

endmodule
```

4.4.2 | Testbench

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 03:39:23 PM
// Design Name:
// Module Name: test_fa_ha
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////



module test_fa_ha();
    reg a,b,c_in;
    wire  s,c_out;
    fa_ha uut(a,b,c_in,s,c_out);

    initial
    begin
    #10 a=0;b=0;c_in=0;
    #10 a=0;b=0;c_in=1;
    #10 a=0;b=1;c_in=0;
    #10 a=0;b=1;c_in=1;
    #10 a=1;b=0;c_in=0;
    #10 a=1;b=0;c_in=1;
    #10 a=1;b=1;c_in=0;
    #10 a=1;b=1;c_in=1;

    #10 $finish;
    end

endmodule
```

4.5 | Elaborated design

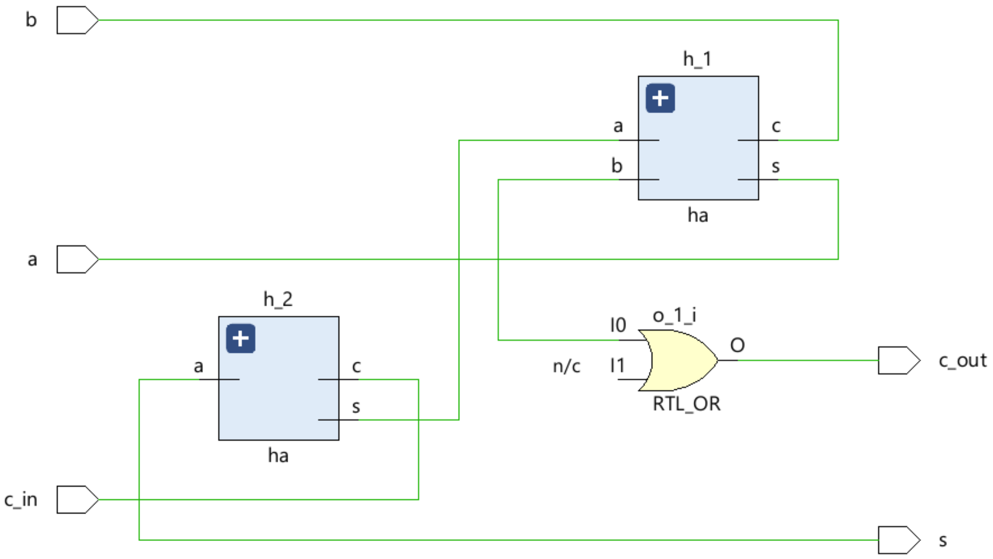
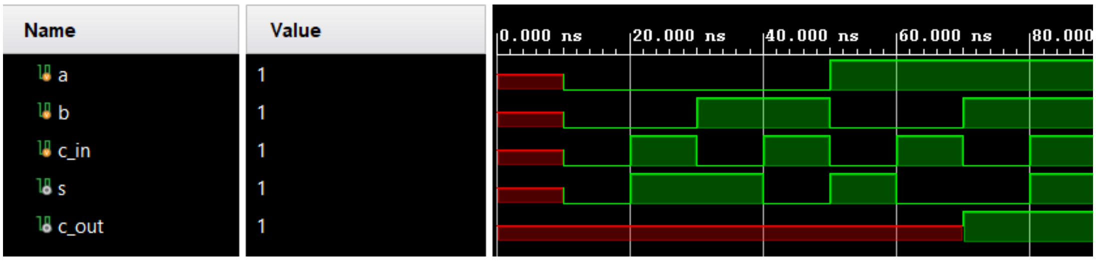


Figure 4.2: Elaborated design of the full adder using two half adders. Internals of the half adders can be seen in Figure 2.3

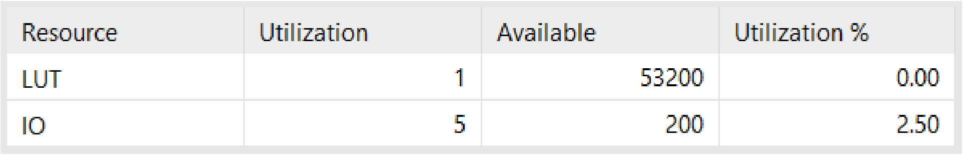
4.6 | Timing diagram



Name	Value
a	1
b	1
c_in	1
s	1
c_out	1

Figure 4.3: Timing diagram of full adder using half adders.

4.7 | Utilization summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	5	200	2.50

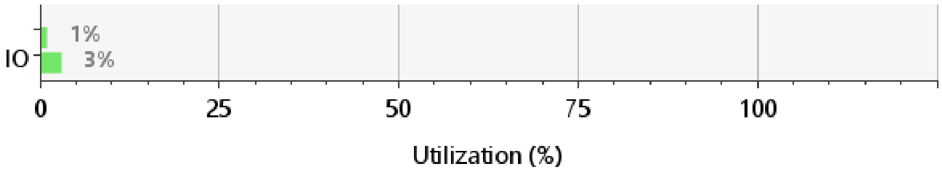


Figure 4.4: Utilization summary of the full adder using two half adders.

4.8 | Remarks

The design and testing of full adder using half adders in structural Verilog has been carried out. The device is showing expected behaviour.

5 Half adder (RTL)

20-09-2023

5.1 Objectives

- To design a half adder in RTL Verilog and test its functionality.

5.2 Schematic

Schematic for half adder is given in Half adder (structural).

5.3 Truth table

Truth table for half adder is given in Table 2.1.

5.3.1 Reduced Boolean expression

Reduced Boolean expression for half adder is given in Half adder (structural).

5.4 Code

5.4.1 Design

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 02:25:25 PM
// Design Name:
// Module Name: ha_rtl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ha_rtl(
    output s,
    output c,
    input a,
    input b
    );

    assign s = a^b;   // the RTL way, as opposed to using xor xr_1 (s,a,b);
    assign c = a&b;

endmodule
```

5.4.2 Testbench

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 02:27:24 PM
// Design Name:
// Module Name: test_ha_rtl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
```

```verilog
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module test_ha_rtl();
    reg a,b;
    wire  s,c;
    ha_rtl uut(s,c,a,b);

    initial
    begin
    #10 a=0;b=0;
    #10 a=0;b=1;
    #10 a=1;b=0;
    #10 a=1;b=1;


    #10 $finish;
    end

endmodule
```

5.5 | Elaborated design

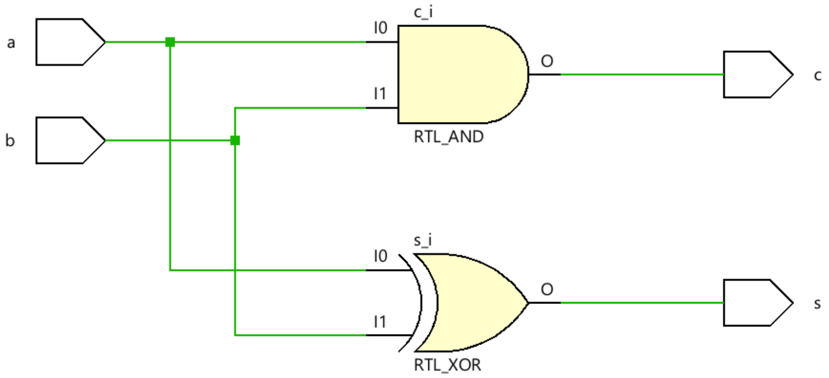
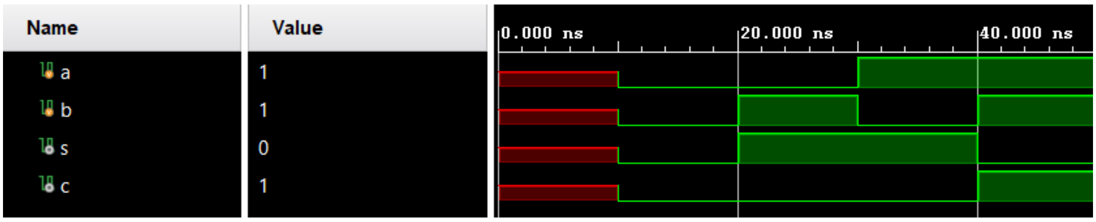


Figure 5.1: Elaborated design of half adder in RTL Verilog.

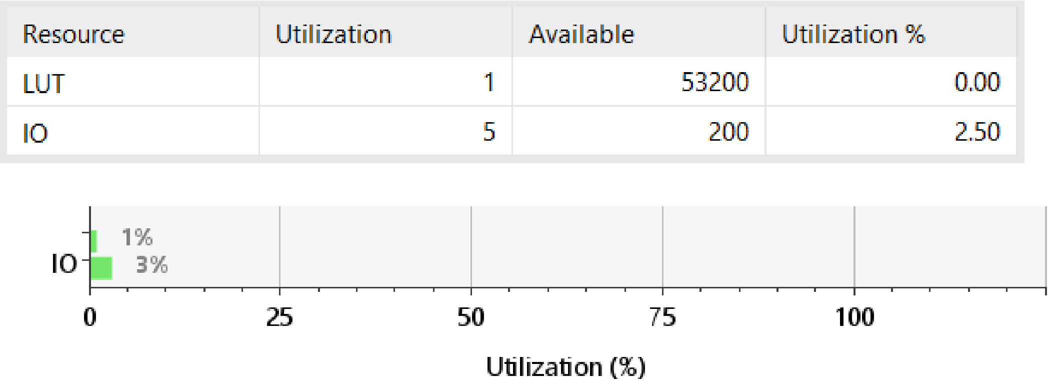
5.6 | Timing diagram



Name	Value
a	1
b	1
s	0
c	1

Figure 5.2: Timing diagram for half adder in RTL Verilog.

5.7 | Utilization summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	5	200	2.50

Figure 5.3: Utilization summary of half adder in RTL Verilog.

5.8 | Remarks

The design and testing of half adder in RTL Verilog has been carried out. The device is showing expected behaviour.

6 | Full adder(structural) using half adder(RTL)

20-09-2023

6.1 | Objectives

- To design a full adder in structural Verilog using half adder modules from Section 5. written in RTL Verilog and test its functionality.

6.2 | Schematic

Truth block diagram for the full adder using half adders is given in Figure 4.1.

6.3 | Truth table

Truth table for the full adder is given in Table 3.1.

6.3.1 | Reduced Boolean expression

Reduced Boolean expression for the full adder is given in Section 3.3.1.

6.4 | Code

6.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/13/2023 02:57:44 PM
// Design Name:
// Module Name: fa_ha
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module fa_str_ha_rtl(
    output s,
   output c_out,
    input a,
    input b,
    input c_in
    );
    wire s_1,c_1,c_2;


    ha_rtl h_1(s_1,c_1,a,b); // OUTPUT OF FIRST HA

    ha_rtl h_2(s,c2,s_1,c_in); // SUM FROM SECOND HA = FINAL SUM

    or o_1(c_out, c_1,c_2); // ADD CARRYS = FINAL CARRY

endmodule
```

6.4.2 | Testbench

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 02:41:06 PM
// Design Name:
// Module Name: test_fa_str_ha_rtl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module test_fa_str_ha_rtl();
    reg a,b,c_in;
    wire  s,c_out;
    fa_str_ha_rtl uut(s,c_out,a,b,c_in);

    initial
    begin
    #10 a=0;b=0;c_in=0;
    #10 a=0;b=0;c_in=1;
    #10 a=0;b=1;c_in=0;
    #10 a=0;b=1;c_in=1;
    #10 a=1;b=0;c_in=0;
    #10 a=1;b=0;c_in=1;
    #10 a=1;b=1;c_in=0;
    #10 a=1;b=1;c_in=1;

    #10 $finish;
    end

endmodule
```

6.5 Elaborated design

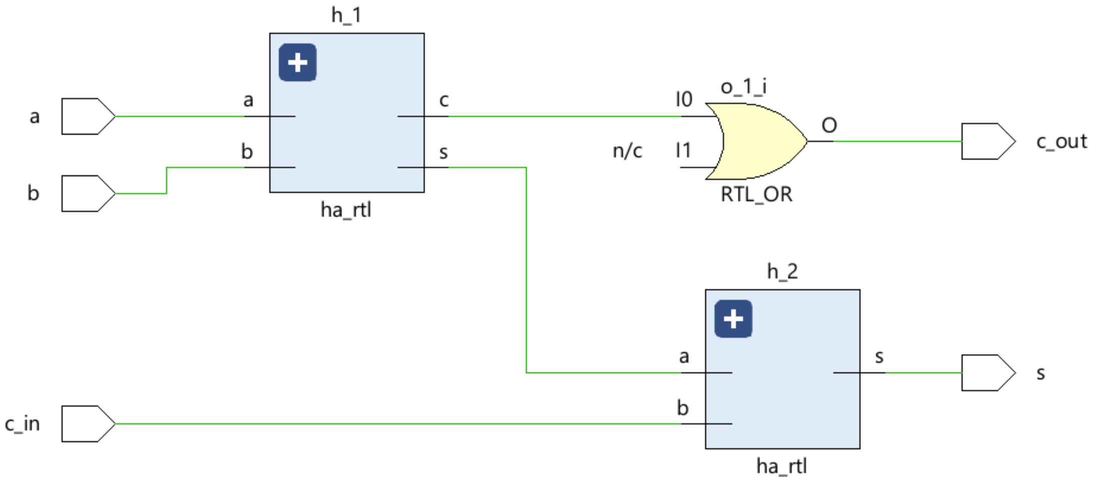
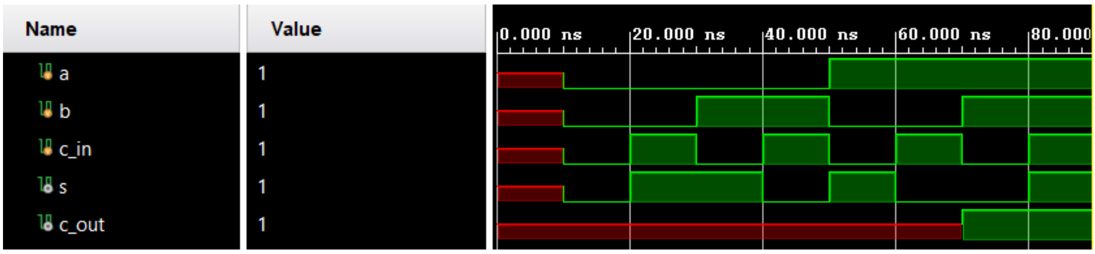


Figure 6.1: Elaborated design of structural full adder using RTL half adder modules.

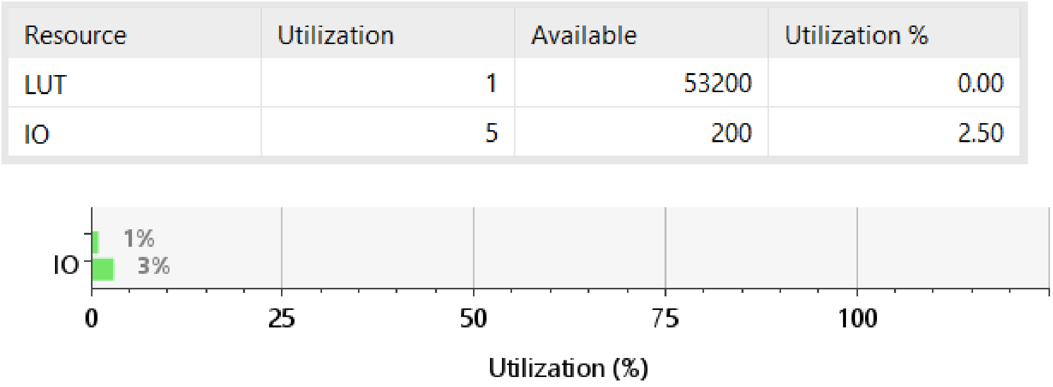
6.6 | Timing diagram



Name	Value
a	1
b	1
c_in	1
s	1
c_out	1

Figure 6.2: Timing diagram of structural full adder using RTL half adder modules.

6.7 | Utilization summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	5	200	2.50

Figure 6.3: Utilization summary of structural full adder using RTL half adder modules.

6.8 | Remarks

The design and testing of structural full adder using RTL half adders in Verilog has been carried out. The device is showing expected behaviour.

7 | 2x1 MUX using structural and RTL

20-09-2023

7.1 | Objectives

- To design a 2x1 multiplexer using structural Verilog and verify its functionality.
- To design a 2x1 multiplexer using RTL Verilog and verify its functionality.

7.2 | Schematic

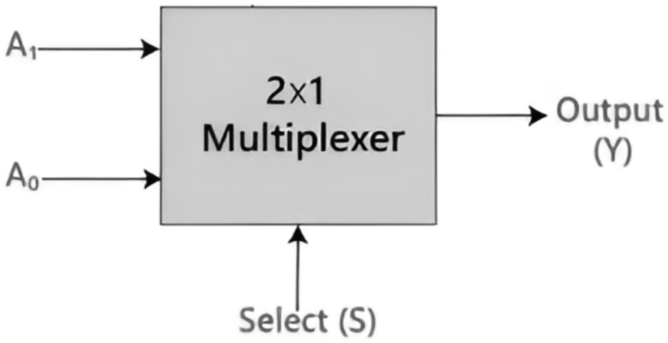


Figure 7.1: Block diagram for 2x1 MUX.

7.3 | Truth table

S	A_1	A_0	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Table 7.1: Truth Table for 2x1 Mux

7.3.1 | Reduced Boolean expression

A_0A_1

S	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$$Y = \overline{S} \cdot A_1 + S \cdot A_0$$

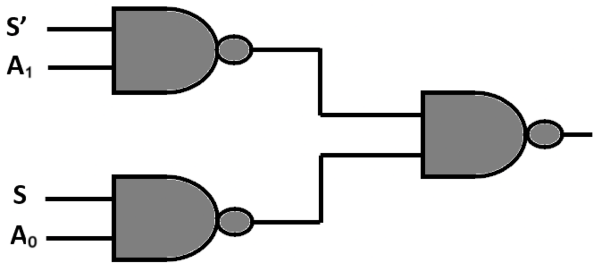


Figure 7.2: NAND logic for structural 2x1 MUX.

7.4 | Code

7.4.1 | Structural Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 03:07:37 PM
// Design Name:
// Module Name: 2x1_mux_str
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module _2x1_mux_str(
    output y,
    input s,
    input a0,
    input a1
    );
    wire w1,w2,nw_1;

    not n1(nw_1,s); //s'

    nand nd1(w1,a0,nw_1); //s' ND a0

    nand nd2(w2,s,a1); //s ND a1

    nand nd3(y,w1,w2); // final ND to give y


endmodule
```

7.4.2 | RTL Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 03:26:05 PM
// Design Name:
// Module Name: _2x1_mux_rtl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module _2x1_mux_rtl(
    output y,
    input s,
    input a0,
    input a1
    );

    assign f = s?a0:a1;

endmodule
```

7.4.3 | Testbench

```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 03:14:40 PM
// Design Name:
// Module Name: test__2x1_mux_str
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test__2x1_mux_str();

    reg a1,a0,s;
    wire  y;
    _2x1_mux_str uut(y,s,a0,a1);

    initial
    begin

    #10 s=0;a0=0;a1=0;
    #10 s=0;a0=0;a1=1;
    #10 s=0;a0=1;a1=0;
    #10 s=0;a0=1;a1=1;
    #10 s=1;a0=0;a1=0;
    #10 s=1;a0=0;a1=1;
    #10 s=1;a0=1;a1=0;
    #10 s=1;a0=1;a1=1;


    #10 $finish;
    end

endmodule
```

7.5 | Elaborated design

7.5.1 | Structural

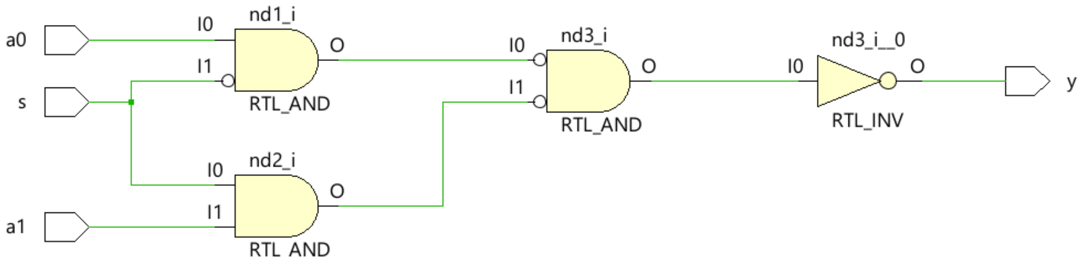


Figure 7.3: Elaborated design for structural 2x1 MUX based on NAND logic.

7.5.2 | RTL

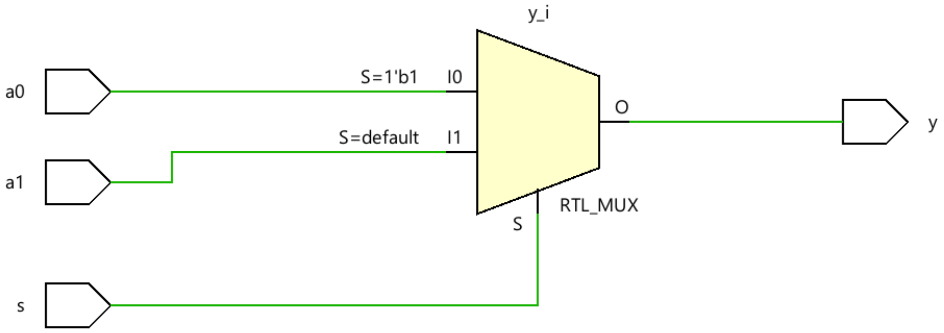
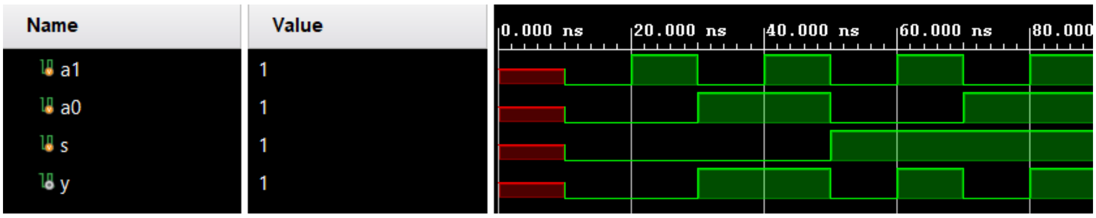


Figure 7.4: Elaborated design of 2x1 MUX in RTL Verilog.

7.6 | Timing diagram

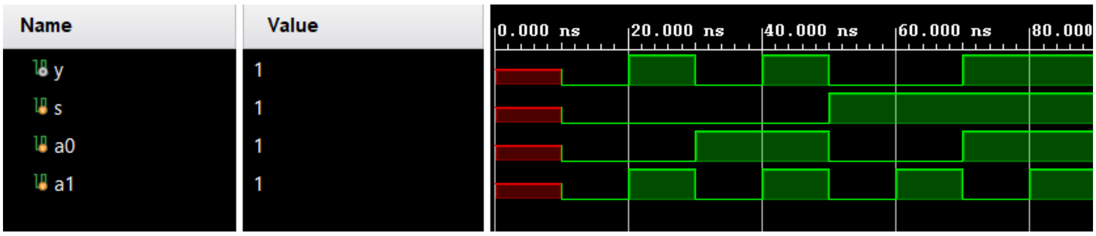
7.6.1 | Structural



Name	Value
a1	1
a0	1
s	1
y	1

Figure 7.5: Timing diagram for structural 2x1 MUX.

7.6.2 | RTL

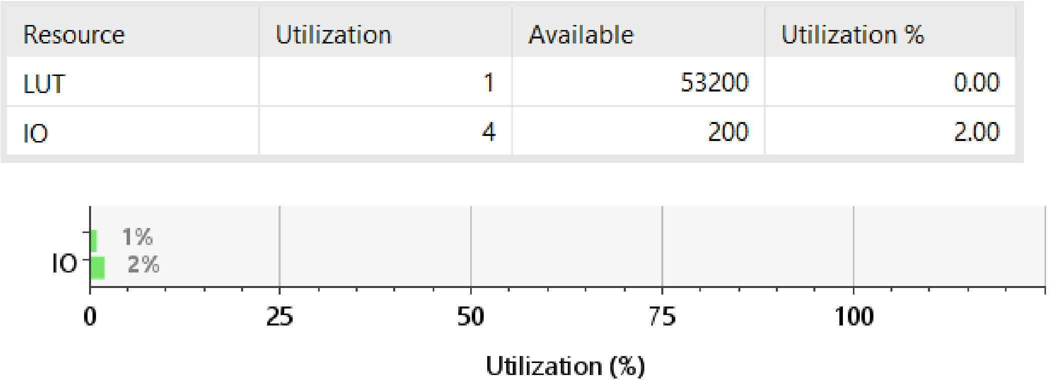


Name	Value
y	1
s	1
a0	1
a1	1

Figure 7.6: Timing diagram for RTL 2x1 MUX.

7.7 | Utilization summary

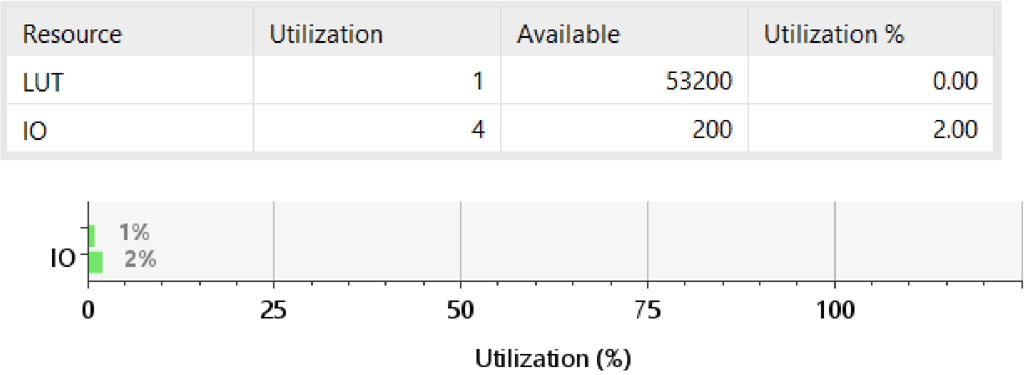
7.7.1 | Structural



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	4	200	2.00

Figure 7.7: Utilization summary for structural 2x1 MUX.

7.7.2 | RTL



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	4	200	2.00

Figure 7.8: Utilization summary for RTL 2x1 MUX.

7.8 | Remarks

The design and testing of 2x1 MUX has been done in structural and RTL Verilog. The device is showing expected behaviour.

8 | 4x1 MUX using 2x1 MUX in RTL

20-09-2023

8.1 | Objectives

- Design a 4x1 multiplexer in RTL Verilog using 2x1 MUX designed in Section 7.4.2 and verify its functionality.

8.2 | Schematic

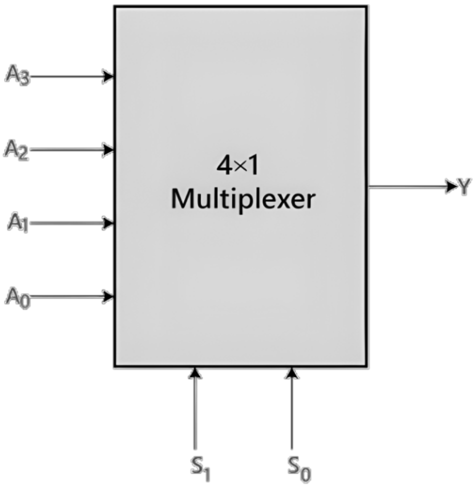


Figure 8.1: Block diagram 4x1 MUX

8.3 | Truth table

S_0	S_1	A_0	A_1	A_2	A_3	Y
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

Table 8.1: Truth Table for 4x1 Multiplexer

8.3.1 | Reduced Boolean expression

$$Y = \overline{S_0} \cdot \overline{S_1} \cdot A_0 + \overline{S_0} \cdot S_1 \cdot A_1 + S_0 \cdot \overline{S_1} \cdot A_2 + S_0 \cdot S_1 \cdot A_3$$

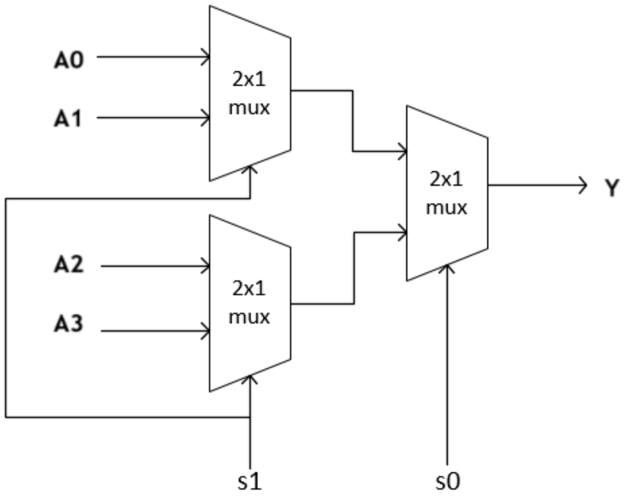


Figure 8.2: 4x1 MUX using three 2x1 MUX.

8.4 | Code

8.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 24.09.2023 08:48:20
// Design Name:
// Module Name: 4x1_mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module _4x1_mux(input s0, s1,input  A0,A1,A2,A3,output y);

  wire y0, y1;

mux_2_1 m1(s1, A2, A3, y1);
mux_2_1 m2(s1, A0, A1, y0);
mux_2_1 m3(s0, y0, y1, y);
endmodule
```

8.4.2 | Testbench

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 24.09.2023 08:52:10
// Design Name:
// Module Name: test_ab_mx4
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test_ab_mx4(

    );

  reg s0, s1;
  reg A0,A1,A2,A3;
  wire y;

  _4x1_mux uut(s0, s1, A0, A1, A2, A3, y);

  initial begin

    {A3,A2,A1,A0} = 4'b1010;

    {s0, s1} = 00;#100;
     {s0, s1} = 01;#100;
     {s0, s1} = 10;#100;
     {s0, s1} = 11;#100;

  end
endmodule
```

8.5 Elaborated design

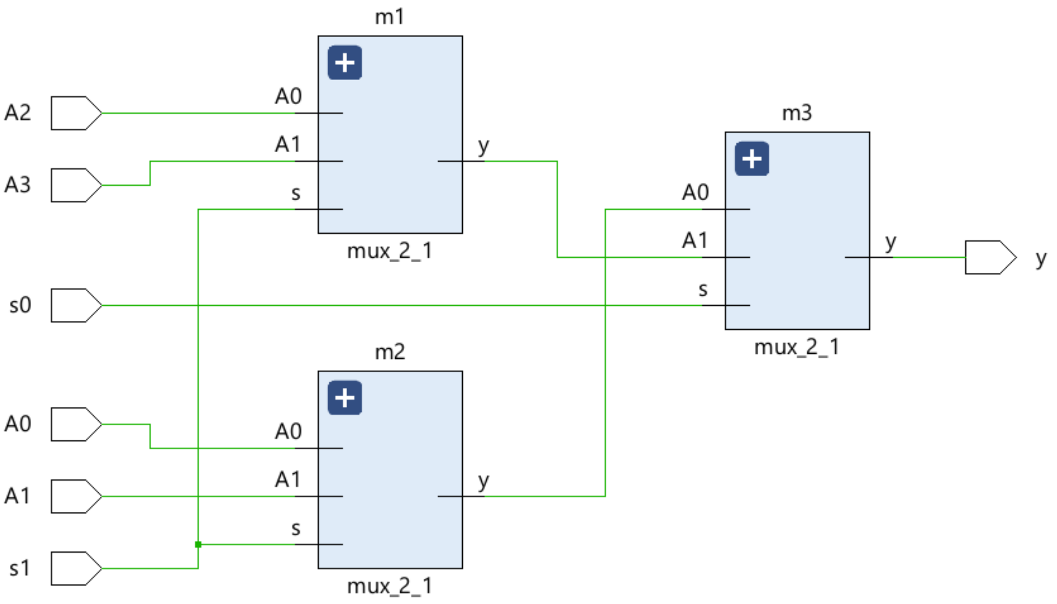
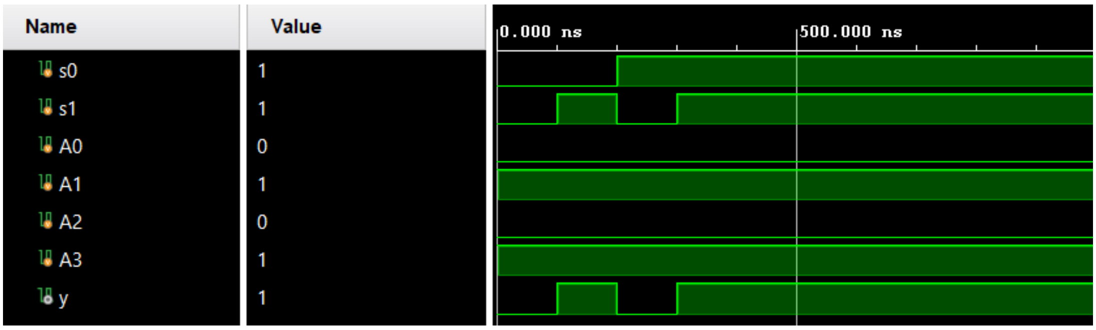


Figure 8.3: Elaborated design of 4x1 MUX using three 2x1 MUX.

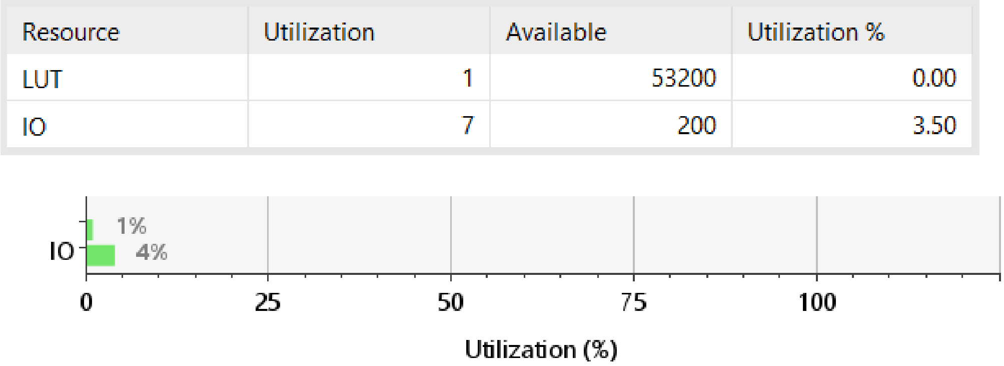
8.6 | Timing diagram



Name	Value
s0	1
s1	1
A0	0
A1	1
A2	0
A3	1
y	1

Figure 8.4: Timing diagram 4x1 MUX.

8.7 | Utilization summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	7	200	3.50

Figure 8.5: Utilization summary 4x1 MUX.

8.8 | Remarks

The design and testing of 4x1 MUX using three 2x1 MUXes has been done in RTL Verilog. The device is showing expected behaviour.

9 | 2x1 MUX using input A, B as one vector with 2 bits defined

20-09-2023

9.1 | Objectives

- To design and test 2x1 MUX using a 2 bit vector input in RTL Verilog.

9.2 | Schematic

Schematic for 2x1 MUX is given in Section 7.

9.3 | Truth table

Truth table for 2x1 MUX is given in Section 7.

9.3.1 | Reduced Boolean expression

Reduced Boolean expression for 2x1 MUX is given in Section 7.

9.4 | Code

9.4.1 | Design

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 09/20/2023 04:17:47 PM
// Design Name:
// Module Name: _2x1_mux_vector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module _2x1_mux_vector(
    output y,
    input s,
    input [1:0]a
    );

    assign y = s ? a[1] : a[0];

endmodule
```

9.4.2 | Testbench

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: CASEST, University of Hyderabad
// Author: Abhinav M
//
// Create Date: 24.09.2023 09:32:56
// Design Name:
// Module Name: vex_2x1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vex_2x1();

    reg [0:1]a;
    reg s;
    wire  y;
    _2x1_mux_vector uut(y,s,a[0:1]);

    initial
    begin

    #10 s=0;a=2'b00;
    #10 s=0;a=2'b01;
    #10 s=0;a=2'b10;
    #10 s=0;a=2'b11;
    #10 s=1;a=2'b00;
    #10 s=1;a=2'b01;
    #10 s=1;a=2'b10;
    #10 s=1;a=2'b11;



    #10 $finish;
    end

endmodule
```

9.5 | Elaborated design

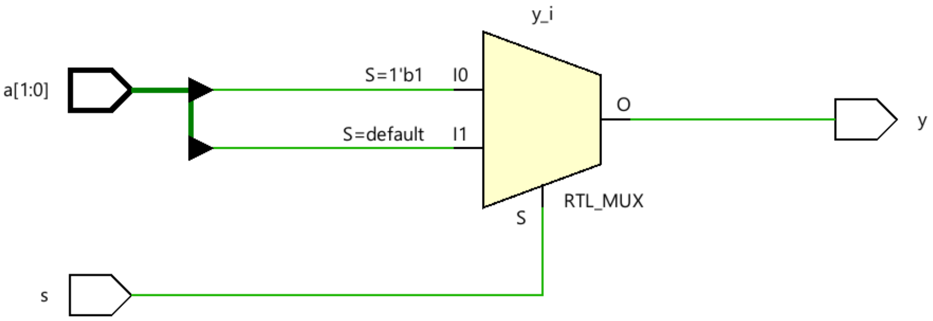
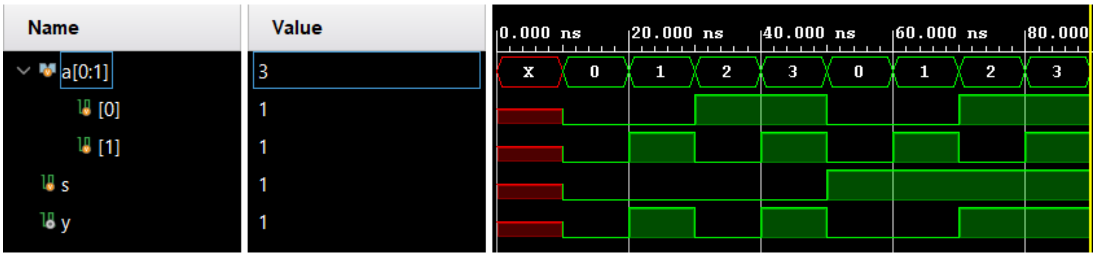


Figure 9.1: Elaborated design of 2x1 MUX taking vector input.

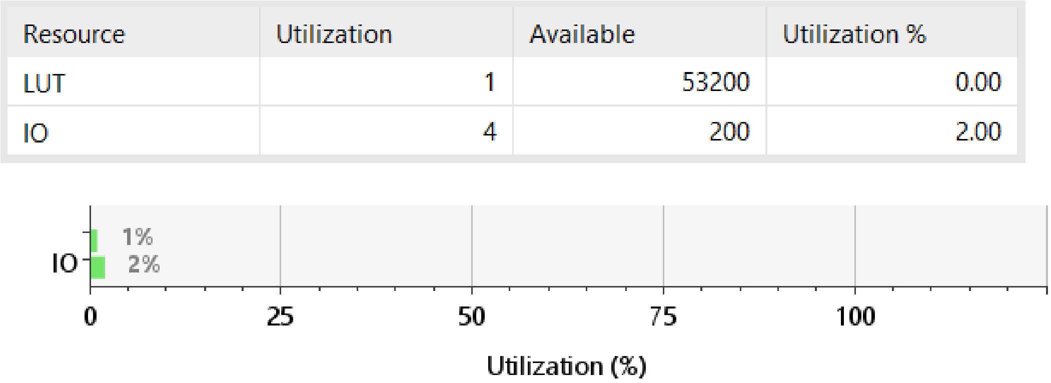
9.6 | Timing diagram



Name	Value
a[0:1]	3
[0]	1
[1]	1
s	1
y	1

Figure 9.2: Timing diagram Elaborated design of 2x1 MUX taking vector input.

9.7 | Utilisation Summary



Resource	Utilization	Available	Utilization %
LUT	1	53200	0.00
IO	4	200	2.00

Figure 9.3: Utilization summary of 2x1 MUX taking vector input.

9.8 | Remarks

The design and testing of 2x1 MUX using a 2 bit vector input has been done in RTL Verilog. The device is showing expected behaviour.

10 | References

Image of the Zedboard and some block diagrams are taken from public sources.

A | Appendix A: GitHub Repo

The GitHub repo referenced below contains the Verilog code and testbenches mentioned in this report.
https://github.com/AbhinavM2000/MV401D/