

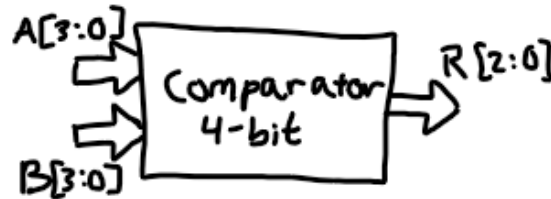


## Laboratory Report #4

Name: Josh Ratificar Date Completed: 10-02-2023

Laboratory Exercise Title: Dataflow Modeling of Combinational Circuits

### Block Diagrams:



*Above includes the block diagrams for this Laboratory Exercise. Based on the diagrams, the following can be concluded:*

### 4-Bit Comparator

#### 8-Input Ports:

- A[0], A[1], A[2], A[3], B[0], B[1], B[2], B[3]

#### 3-Output Ports:

- R[2], R[1], R[0]

### NOTE

G (“greater than” is TRUE) | R [2] = G (1 if A > B, else 0)

E (“equal to” is TRUE) | R [1] = E (1 if A = B, else 0)

L (“less than” is TRUE) | R [0] = L (1 if A < B, else 0)



## Exercise 4A:

```
1  /*****
2  *   FILE:                FourBitComparator.v
3  *   AUTHOR:              Josh Ratificar
4  *   Class:               Gr.3 CpE 3101L Introduction to HDL
5  *   Group/Schedule       Friday, 7:30 AM to 10:30 AM
6  *   Description:         FourBitComparator.v module
7  *****/
8  module FourBitComparator(
9      input    [3:0]A,
10     input    [3:0]B,
11     output   [2:0]R
12 );
13     wire      [2:0]common;
14     // A > B @ R[2] | R[2] = G (1 if A > B, else 0)
15     assign R[2] = (A > B)? 1'b1 : 1'b0;
16
17
18     // A = B @ R[1] | R[1] = E (1 if A = B, else 0)
19     assign R[1] = (A == B)? 1'b1 : 1'b0;
20
21     // A < B @ R[0] | R[0] = E (1 if A < B, else 0)
22     assign R[0] = (A < B)? 1'b1 : 1'b0;
23 endmodule
24
```

Figure 1.0 – FourBitComparator.v Script

Quartus Prime Lite Edition - C:\Users\josh\OneDrive\Desktop\University of San Carlos\USC Year 3 - Sem 1\HDL\Laboratories\LE4\Verilog\FourBitComparator\FourBitComparator - FourBitComparator

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

Tasks Compilation FourBitComparator.v tb\_FourBitComparator.v Compilation Report - FourBitComparator

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
  - Flow Messages
  - Flow Suppressed Messages

Flow Summary

<<Filter>>	
Flow Status	Successful - Mon Oct 02 20:05:24 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	FourBitComparator
Top-level Entity Name	FourBitComparator
Family	MAX 10
Device	10M08DAF484C8G
Timing Models	Final
Total logic elements	9
Total registers	0
Total pins	11
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Messages

System (12) Processing (12)

Running Quartus Prime Analysis & Synthesis  
Command: quartus\_map --read\_settings\_files=on --write\_settings\_files=off FourBitComparator -c FourBitComparator

18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM\_PARALLEL\_PROCESSORS to 1.

20030 Parallel compilation is enabled and will use 4 of the 4 processors detected

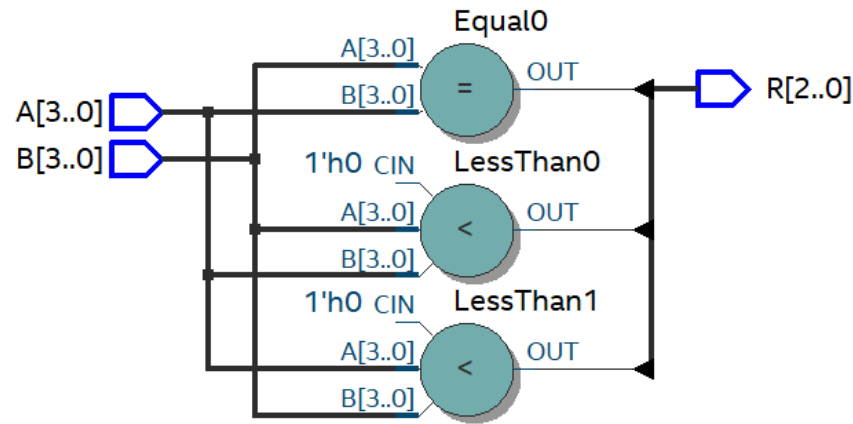
12021 Found 1 design units, including 1 entities, in source file FourBitComparator.v

12021 Found 1 design units, including 1 entities, in source file tb\_FourBitComparator.v

12127 Elaborating entity "FourBitComparator" for the top level hierarchy

100% 00:00:12 8:05 PM

Figure 1.1 – FourBitComparator.v Analysis and Elaboration Test Results



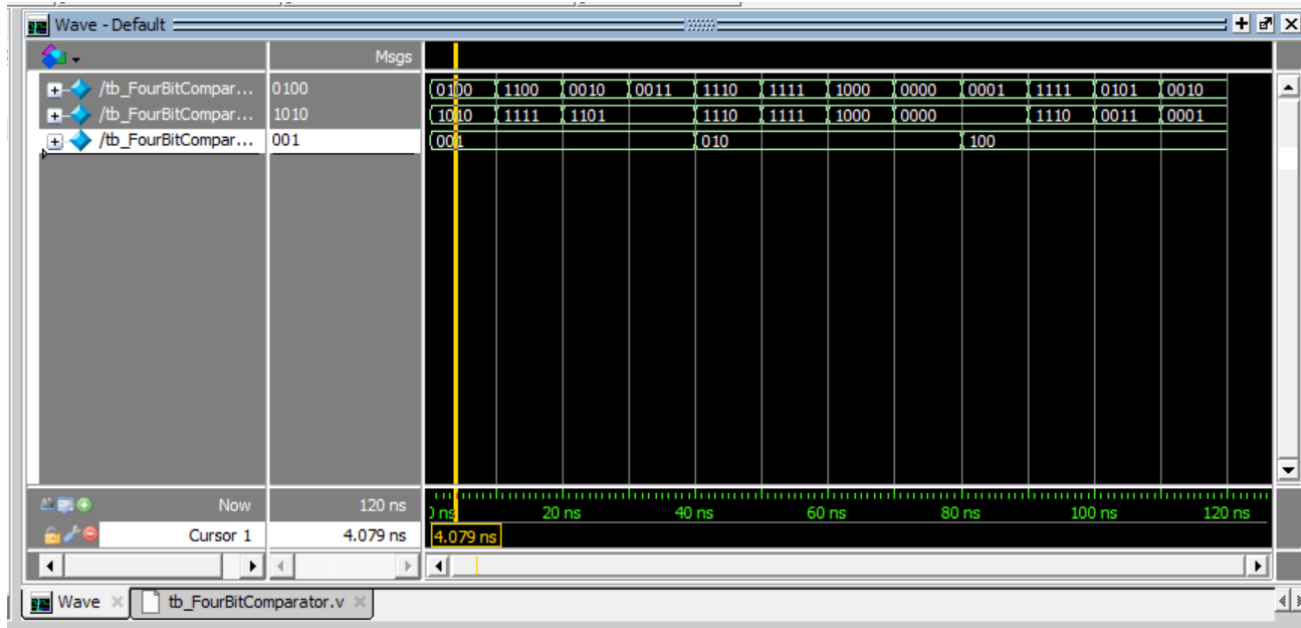
**Figure 1.2 – FourBitComparator RTL View Output**

```

1  /*****
2  *   FILE:                tb_FourBitComparator.v
3  *   AUTHOR:              Josh Ratificar
4  *   Class:               Gr.3 CpE 3101L Introduction to HDL
5  *   Group/Schedule       Friday, 7:30 AM to 10:30 AM
6  *   Description:         tb_FourBitComparator.v module
7  *****/
8  `timescale 1 ns / 1 ps
9  module tb_FourBitComparator();
10     reg    [3:0] A, B;
11     wire   [2:0] R;
12     FourBitComparator UUT(
13         .A(A),
14         .B(B),
15         .R(R)
16     );
17     initial begin
18         $display("Test Bench | tb_FourBitComparator...");
19         // Test cases for A < B
20         $display("A < B");
21         A = 4'b0100; B = 4'b1010; #10; // A < B
22         A = 4'b1100; B = 4'b1111; #10; // A < B
23         A = 4'b0010; B = 4'b1101; #10; // A < B
24         A = 4'b0011; B = 4'b1101; #10; // A < B
25
26         // Test cases for A = B
27         $display("A = B");
28         A = 4'b1110; B = 4'b1110; #10; // A = B
29         A = 4'b1111; B = 4'b1111; #10; // A = B
30         A = 4'b1000; B = 4'b1000; #10; // A = B
31         A = 4'b0000; B = 4'b0000; #10; // A = B
32
33         // Test cases for A > B
34         $display("A > B");
35         A = 4'b0001; B = 4'b0000; #10; // A > B
36         A = 4'b1111; B = 4'b1110; #10; // A > B
37         A = 4'b0101; B = 4'b0011; #10; // A > B
38         A = 4'b0010; B = 4'b0001; #10; // A > B
39         $stop;
40     end
41     initial begin
42         $monitor("Time = %2d ns\t A = %d | [%b]\t B = %d [%b]\t R = %b", $time, A, A, B, B, R);
43     end
44 endmodule
45

```

**Figure 1.3 – tb\_FourBitComparator.v Script for Testing Module**



**Figure 1.4** – Four Bit Comparator RTL Simulation Output

```
# Test Bench | tb_FourBitComparator...
# A < B
# Time = 0 ns A = 4 | [0100] B = 10 [1010] R = 001
# Time = 10 ns A = 12 | [1100] B = 15 [1111] R = 001
# Time = 20 ns A = 2 | [0010] B = 13 [1101] R = 001
# Time = 30 ns A = 3 | [0011] B = 13 [1101] R = 001
# A = B
# Time = 40 ns A = 14 | [1110] B = 14 [1110] R = 010
# Time = 50 ns A = 15 | [1111] B = 15 [1111] R = 010
# Time = 60 ns A = 8 | [1000] B = 8 [1000] R = 010
# Time = 70 ns A = 0 | [0000] B = 0 [0000] R = 010
# A > B
# Time = 80 ns A = 1 | [0001] B = 0 [0000] R = 100
# Time = 90 ns A = 15 | [1111] B = 14 [1110] R = 100
# Time = 100 ns A = 5 | [0101] B = 3 [0011] R = 100
# Time = 110 ns A = 2 | [0010] B = 1 [0001] R = 100
# ** Note: $stop : C:/Users/joshz/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/FourBitComparator/tb_FourBitComparator.v(39)
# Time: 120 ns Iteration: 0 Instance: /tb_FourBitComparator
# Break in Module tb_FourBitComparator at C:/Users/joshz/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/FourBitComparator/tb_FourBitComparator.v line 39
VSIM 2>
```

**Figure 1.5** – Four Bit Comparator Test Bench Monitor Output (Annotations to **Figure 1.4**)

### Discussion of Results (Exercise 4A)

By observing the Test bench script (**Figure 1.3**), we expect that the respective R bit reflect the comparison between two four-bit inputs. This is the best-case desired behaviour expected which after simulating through RTL simulation (**Figure 1.4**), R appropriately and respectively changes every 40 ns. This is how the test bench was setup, and we can furthermore observe this behaviour in **Figure 1.5**. In conclusion, our design for our comparator works even when we applied Data Flow design in HDL Verilog.



## Exercise 4B:

```

1  /******
2  * FILE:          n_bit4x1Multiplexer.v
3  * AUTHOR:       Josh Ratificar
4  * Class:        Gr.3 CpE 3101L Introduction to HDL
5  * Group/Schedule Friday, 7:30 AM to 10:30 AM
6  * Description:  n_bit4x1Multiplexer.v module
7  * *****/
8  module n_bit4x1Multiplexer
9  #(parameter n = 4)
10  (
11      input [(n-1):0] A, B, C, D,
12      input [1:0] S,
13      output [(n-1):0] Y
14  );
15      assign Y = (S == 2'b00) ? A : (S == 2'b01) ? B : (S == 2'b10) ? C : (S == 2'b11) ? D : {n{1'b0}};
16  endmodule
17

```

Figure 2.0 – *n\_bit4x1Multiplexer.v Script*

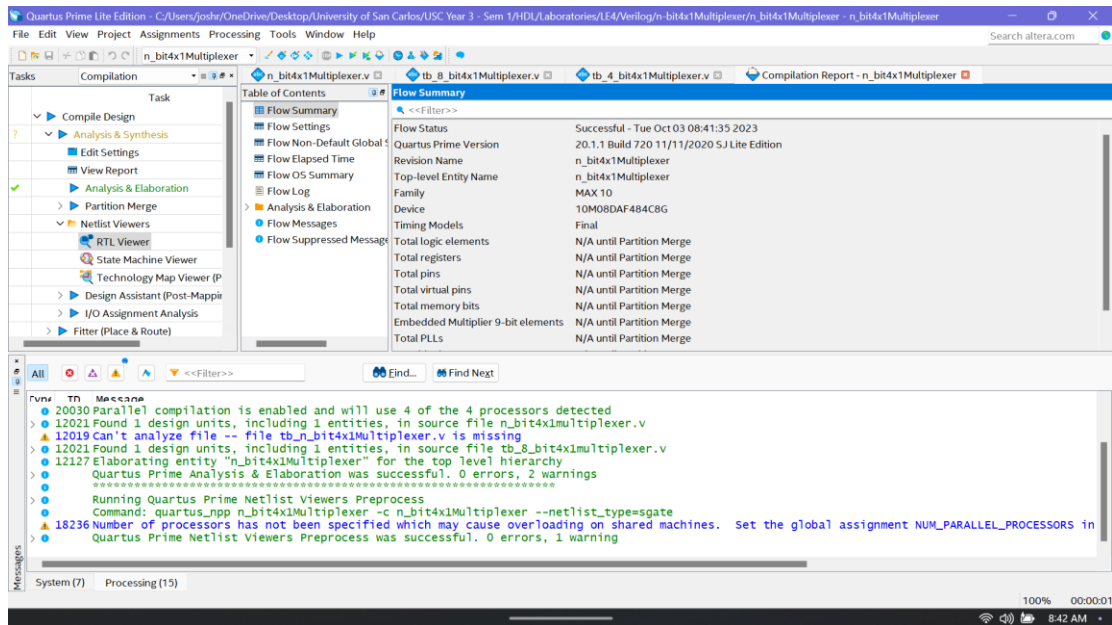


Figure 2.1 – *n\_bit4x1Multiplexer.v Analysis and Elaboration Test Results*

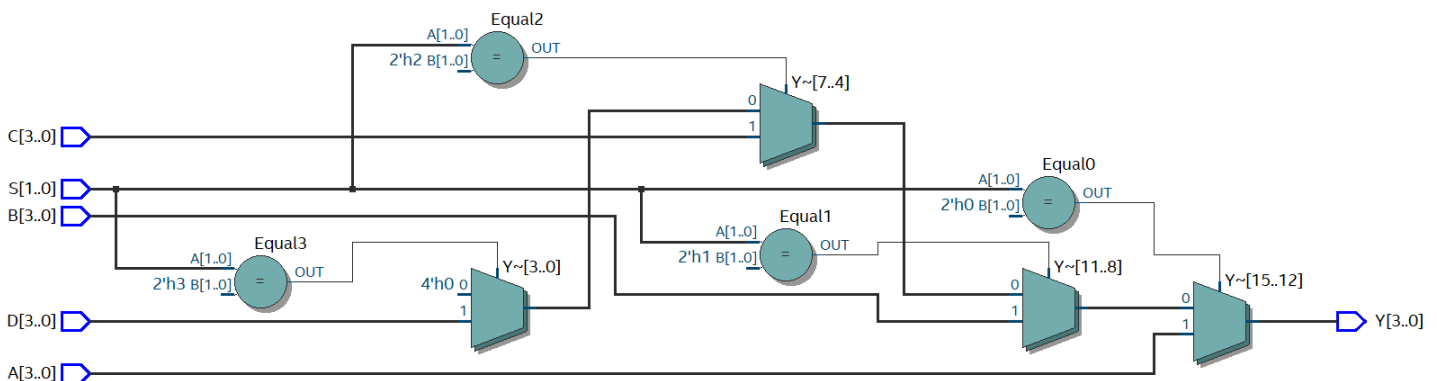


Figure 2.2 – *n\_bit4x1Multiplexer RTL View Output*



```
1 *****
2 FILE:          tb_4_bit4x1Multiplexer.v
3 AUTHOR:        Josh Ratificar
4 Class:         Gr.3 CpE 3101L Introduction to HDL
5 Group/Schedule Friday, 7:30 AM to 10:30 AM
6 Description:    tb_4_bit4x1Multiplexer.v test bench for n_bit4x1Multiplexer
7 *****
8 module tb_4_bit4x1Multiplexer();
9     reg [3:0] A, B, C, D;
10    reg [1:0] S;
11    wire [3:0] Y;
12
13    n_bit4x1Multiplexer #(n(4)) UUT(
14        .A(A),
15        .B(B),
16        .C(C),
17        .D(D),
18        .S(S),
19        .Y(Y)
20    );
21
22    initial begin
23        $display("Test Bench | tb_4_bit4x1Multiplexer...");
24        $display("Test case 1: Select A");
25        A = 4'b1010; B = 4'b0110; C = 4'b1101; D = 4'b0001; S = 2'b00; #10;
26
27        $display("Test case 2: Select B");
28        A = 4'b1010; B = 4'b0110; C = 4'b1101; D = 4'b0001; S = 2'b01; #10;
29
30        $display("Test case 3: Select C");
31        A = 4'b1010; B = 4'b0110; C = 4'b1101; D = 4'b0001; S = 2'b10; #10;
32
33        $display("Test case 4: Select D");
34        A = 4'b1010; B = 4'b0110; C = 4'b1101; D = 4'b0001; S = 2'b11; #10;
35        $stop;
36    end
37    initial begin
38        $monitor("Time = %2d ns\t | A = %d [%b]\t | B = %d [%b]\t | C = %d [%b]\t | D = %d [%b]\t | S = %d [%b]\t | Y = %d [%b]");
39    end
40 endmodule
```

Figure 3.0 – *tb\_4\_bit4x1Multiplexer.v* Script for Testing Module

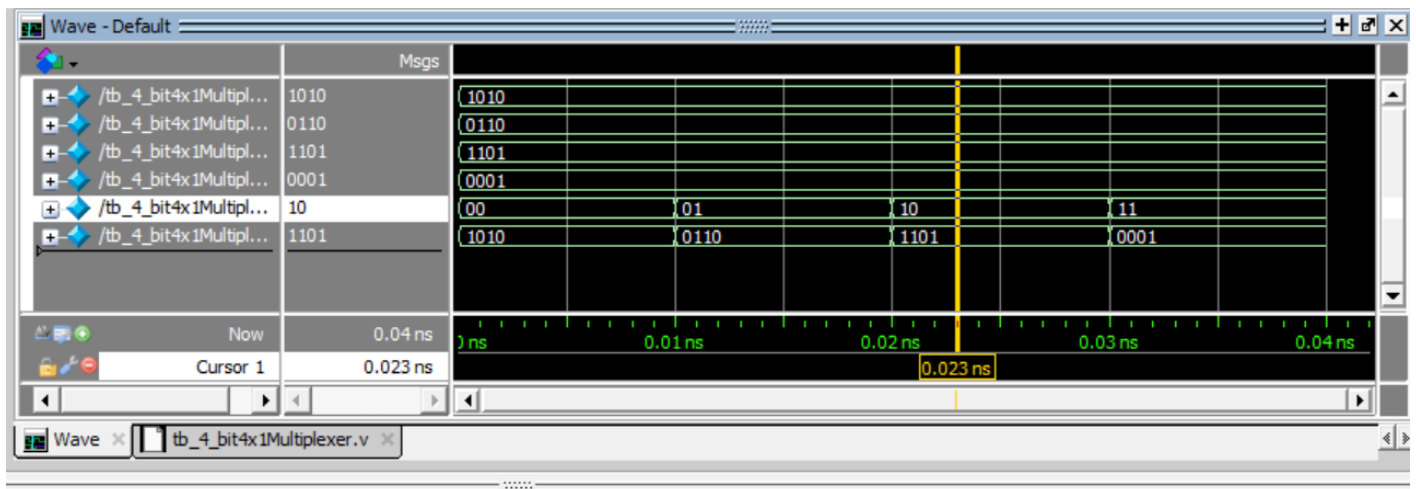


Figure 3.1 – *tb\_4\_bit4x1Multiplexer* RTL Simulation Output



```
# Test Bench | tb_4_bit4x1Multiplexer...
# Test case 1: Select A
# Time = 0 ns |A = 10 [1010]| |B = 6 [0110]| |C = 13 [1101]| |D = 1 [0001]| |S = 0 [00]| |Y = 10 [1010]|
# Test case 2: Select B
# Time = 10 ns |A = 10 [1010]| |B = 6 [0110]| |C = 13 [1101]| |D = 1 [0001]| |S = 1 [01]| |Y = 6 [0110]|
# Test case 3: Select C
# Time = 20 ns |A = 10 [1010]| |B = 6 [0110]| |C = 13 [1101]| |D = 1 [0001]| |S = 2 [10]| |Y = 13 [1101]|
# Test case 4: Select D
# Time = 30 ns |A = 10 [1010]| |B = 6 [0110]| |C = 13 [1101]| |D = 1 [0001]| |S = 3 [11]| |Y = 1 [0001]|
# ** Note: $stop : C:/Users/joshzr/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/n-bit4x1Multiplexer/tb_4_bit4x1Multiplexer.v(34)
# Time: 40 ps Iteration: 0 Instance: /tb_4_bit4x1Multiplexer
# Break in Module tb_4_bit4x1Multiplexer at C:/Users/joshzr/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/n-bit4x1Multiplexer/tb_4_bit4x1Multiplexer.v line 34
```

Figure 3.2 – *tb\_4\_bit4x1Multiplexer* Test Bench Monitor Output (Annotations to Figure 3.1)

```
1  /*****
2  *   FILE:                tb_8_bit4x1Multiplexer.v
3  *   AUTHOR:              Josh Ratificar
4  *   Class:               Gr.3 CpE 3101L Introduction to HDL
5  *   Group/Schedule       Friday, 7:30 AM to 10:30 AM
6  *   Description:         tb_8_bit4x1Multiplexer.v test bench for n_bit4x1Multiplexer
7  *****/
8  module tb_8_bit4x1Multiplexer();
9      reg [7:0] A, B, C, D;
10     reg [1:0] S;
11     wire [7:0] Y;
12
13     n_bit4x1Multiplexer #(n(8)) UUT(
14         .A(A),
15         .B(B),
16         .C(C),
17         .D(D),
18         .S(S),
19         .Y(Y)
20     );
21
22     initial begin
23         $display("Test Bench | tb_8_bit4x1Multiplexer...");
24         $display("Test case 1: Select A");
25         A = 8'b10101010; B = 8'b01100110; C = 8'b11011101; D = 8'b00010001; S = 2'b00; #10;
26         $display("Test case 2: Select B");
27         A = 8'b10101010; B = 8'b01100110; C = 8'b11011101; D = 8'b00010001; S = 2'b01; #10;
28         $display("Test case 3: Select C");
29         A = 8'b10101010; B = 8'b01100110; C = 8'b11011101; D = 8'b00010001; S = 2'b10; #10;
30         $display("Test case 4: Select D");
31         A = 8'b10101010; B = 8'b01100110; C = 8'b11011101; D = 8'b00010001; S = 2'b11; #10;
32         $stop;
33     end
34     initial begin
35         $monitor("Time = %2d ns\t |A = %d [%b]|\t |B = %d [%b]|\t |C = %d [%b]|\t |D = %d [%b]|\t |S = %d [%b]|\t |Y = %d [%b]|");
36     end
37 endmodule
```

Figure 4.0 – *tb\_8\_bit4x1Multiplexer.v* Script for Testing Module

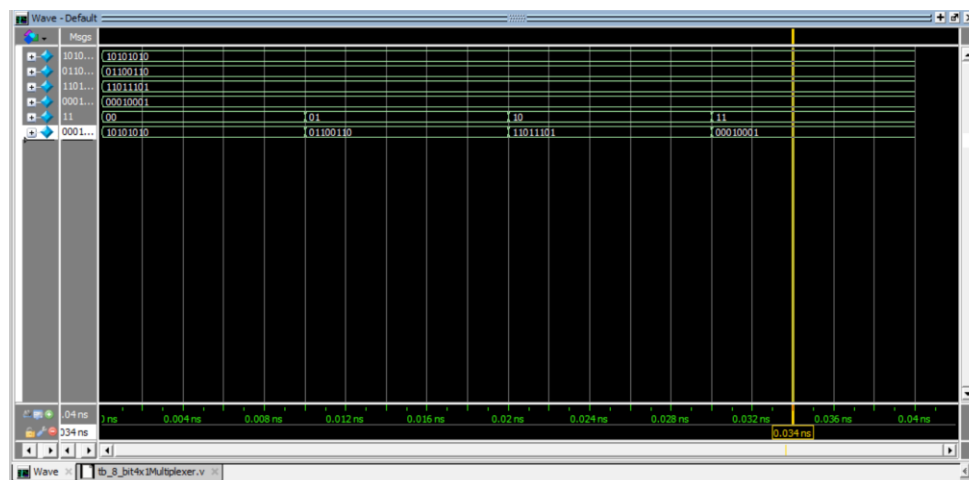


Figure 4.1 – *tb\_4\_bit4x1Multiplexer* RTL Simulation Output





```
# Test Bench : tb_8_bit4x1Multiplexer...
# Test case 1: Select A
# Time = 0 ns |A = 170 [10101010]| |B = 102 [01100110]| |C = 221 [11011101]| |D = 17 [00010001]| |S = 0 [00]| |Y = 170 [10101010]|
# Test case 2: Select B
# Time = 10 ns |A = 170 [10101010]| |B = 102 [01100110]| |C = 221 [11011101]| |D = 17 [00010001]| |S = 1 [01]| |Y = 102 [01100110]|
# Test case 3: Select C
# Time = 20 ns |A = 170 [10101010]| |B = 102 [01100110]| |C = 221 [11011101]| |D = 17 [00010001]| |S = 2 [10]| |Y = 221 [11011101]|
# Test case 4: Select D
# Time = 30 ns |A = 170 [10101010]| |B = 102 [01100110]| |C = 221 [11011101]| |D = 17 [00010001]| |S = 3 [11]| |Y = 17 [00010001]|
# ** Note: $stop : C:/Users/joshz/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/n-bit4x1Multiplexer/tb_8_bi
t4x1Multiplexer.v(31)
# Time: 40 ps Iteration: 0 Instance: /tb_8_bit4x1Multiplexer
# Break in Module tb_8_bit4x1Multiplexer at C:/Users/joshz/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE4/Verilog/n-bit
4x1Multiplexer/tb_8_bit4x1Multiplexer.v line 31
```

**Figure 4.2** – *tb\_4\_bit4x1Multiplexer Test Bench Monitor Output (Annotations to **Figure 4.1**)*

## Discussion of Results (Exercise 4B)

### PART 1:

In **Figure 3.0**, this test bench is setup for a 4-bit 4 to 1 Mux. We are testing every possible combination of S to validate our circuit's behaviour. In our test-bench RTL simulation (**Figure 3.1**), we can observe that we tested four separate cases to select either A, B, C, or D. This behaviour is apparent every 10 ns, where the output across Y reflects its respective selection's four bits. To further strengthen this finding, we utilized the "\$monitor" and "\$display" commands to visually validate the behaviour which can be seen in **Figure 3.2**. Thus, our flexible n-bit 4 to 1 Mux operates as expected when n-bits is set to 4.

### PART 2:

Likewise, **Figure 4.0** is setup for an 8-bit 4 to 1 Mux. The test-benches are set-up similarly, but there were as well additional input/output ports needed to accommodate the additional bits. In **Figure 4.1**, we can see that the same behaviour as described in **Figure 3.1** can be observed. Complementing this finding, **Figure 4.2** monitors which 8-bits are selected across Y, and our four test-cases reflect that only one of the 8-bits can be selected depending on S inputs. From this, we can conclude that our n-bits 4 to 1 Mux as well operates expectedly when n-bits is set to 8.