Ben Linam
Dylan Hall
EEL4712L
11 Nov. 2018
Lab 6

EEL4712L

Lab 6 – Bus Structure

Prelab

Preliminary Setup

Overview

*For this lab, we will be designing a **Bus Structure**. A device which holds a **8-bit input** and performs a **"move or load"** operation storing it in the appropriate **bit register**. Also, the device will keep track of the indicated multiplexer by using input select lines. We will be using **Quartus Prime** to design the **Bus Structure** and implement its logic. Then, we will be using our **Altera DE1-SoC FPGA** board to present our design and confirm its accuracy.*

Materials

- An Altera DE1-SoC kit
- A computer with Quartus Prime

Procedure

*We will be using **ENTITY** and **COMPONENT** designs in this laboratory. We will be designing a **Bus Structure.** This entails a design for a **shift register**, a **multiplexer** as well as a **control circuit.** Once designed, we will be using the **Pin Assignment** utility provided by **Quartus Prime** to determine the **Software → Hardware** implementations.*

*Such that, we will be setting the logic inputs/outputs from our **VHDL** designs to physical pins on our **Altera DE1-SoC** boards to test and simulate the designs.*

Bus Structure

*For the purposes of this lab, we will be implementing an **8-bit bus structure.** For this design, we be implementing a **Shift Register**, a **Multiplexer**, a **Control Circuit** as well as the main **bus structure** program. There will be four shift registers, each of which will cascade their output lines into the input pins of our **Multiplexer.** The **Multiplexer** will also have one more input, for a total of five, which will be data input from the user. The **Shift Registers** will be controlled with a **RegisterControl** bit which will indicate which register is being operated from/on. The **Control Circuit** simply is used to determine which operations and components to utilize.*

- Background:

Description:

*The user will be able to input an **8-bit** input for **data,** as well as a **5-bit** input **function.** This function will allow the user to perform **move** and **load** operations on specified registers. For the **load** function, **(MSB = '1')** the last two bits don't affect anything. Only one register will be loaded and its oriented by the next two bits after the **MSB.** These two bits also indicate which register data will be moved to and the last two bits indicate which register the data will be moved from.*

Function Bit String							
Operation		*Operand*		*Register*	*Operator*		*Register*
X = 0	*X = 1*						
Move	*Load*	*0*	*0*	*0*	*0*	*0*	*0*
Move	*Load*	*0*	*1*	*1*	*0*	*1*	*1*
Move	*Load*	*1*	*0*	*2*	*1*	*0*	*2*
Move	*Load*	*1*	*1*	*3*	*1*	*1*	*3*

***Table 1**. **Function** input descriptor.*

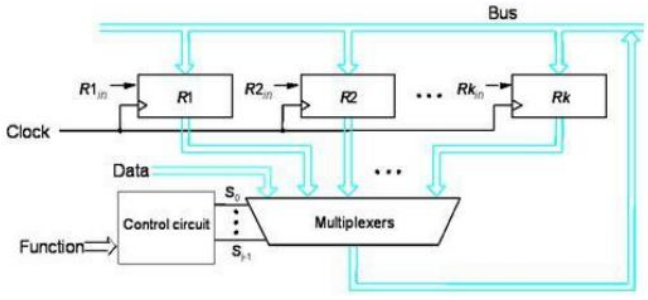

Figure 1. A digital system with k registers.

From this black box, one will be able to follow the direct application of this device.

- *Rin will control the registers*
- *The clock will drive the registers*
- *The Control circuit will drive the data which is present on the bus*

Figure 1. Screen capture of the **Bus Structure** design.

- Realization/Simulation –

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY control IS
6      PORT
7      (
8          F  : IN STD_LOGIC_VECTOR(4 DOWNTO 0);   -- function send to the control circuit
9          R  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- register control
10         S  : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)  -- register select lines
11     );
12
13 END control;
14
15 ARCHITECTURE bhv OF control IS
16     SIGNAL Operand    : INTEGER;
17
18 BEGIN
19 -- Decode the function
20     Operand <= to_integer(unsigned(F(3 DOWNTO 2))); -- which register to operate on
21
22     R <=
23         "0001" WHEN (Operand = 0) ELSE
24         "0010" WHEN (Operand = 1) ELSE
25         "0100" WHEN (Operand = 2) ELSE
26         "1000" WHEN (Operand = 3) ELSE
27         "----";
28
29     S <= F(4) & F(1 DOWNTO 0);
30
31 END bhv;
```

Figure 2. Screen capture displaying the **Quartus Prime logic design** for the **Control Circuit** design.

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3
4   ENTITY shift_reg IS
5       PORT(
6       -- Sequential Sychrounous Logic
7           clk, reset, load  : IN STD_LOGIC;
8       -- Parallel input/output
9           pD     : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
10          pQ     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
11  END shift_reg;
12
13  ARCHITECTURE bhv OF shift_reg IS
14      BEGIN
15      PROCESS(clk, reset)
16          BEGIN
17          IF(reset = '1') THEN
18              pQ <= (OTHERS => '0');
19          ELSIF(RISING_EDGE(clk)) THEN
20              IF(load = '1') THEN
21                  pQ <= pD;
22              END IF;
23          END IF;
24      END PROCESS;
25  END bhv;
26
```

Figure 3. Screen capture displaying the **Quartus Prime logic design** for the **Shift Register** design.

- Realization/Simulation –

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3
4   ENTITY multiplexer IS
5
6       PORT(
7       -- input/output
8           SEL    : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
9           Dout   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
10          Rin0, Rin1, Rin2, Rin3, Din : IN STD_LOGIC_VECTOR(7 DOWNTO 0));
11
12  END multiplexer;
13
14  ARCHITECTURE bhv OF multiplexer IS
15  BEGIN
16      PROCESS(SEL)
17      BEGIN
18          CASE SEL IS
19              WHEN "000" =>
20                  Dout <= Rin0;
21              WHEN "001" =>
22                  Dout <= Rin1;
23              WHEN "010" =>
24                  Dout <= Rin2;
25              WHEN "011" =>
26                  Dout <= Rin3;
27              WHEN OTHERS =>
28                  Dout <= Din;
29          END CASE;
30      END PROCESS;
31  END bhv;
```

Figure 4. Screen capture displayinng the **Quartus Prime logic design** for the **Multiplexer** design.

- Realization/Simulation –

```
 2   USE IEEE.STD_LOGIC_1164.ALL;
 3
 4   ENTITY bus_structure IS
 5       PORT
 6       (
 7         Clk, Reset : IN STD_LOGIC;                              -- clock and reset control
 8         Func       : IN STD_LOGIC_VECTOR(4 DOWNTO 0);       -- function control input
 9         BusData    : BUFFER STD_LOGIC_VECTOR(7 DOWNTO 0);   -- data loaded on the bus
10         Data       : IN STD_LOGIC_VECTOR(7 DOWNTO 0);          -- input data to load
11
12         DTest1, DTest2, DTest3, DTest4 : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
13       );
14   END bus_structure;
15
16   ARCHITECTURE bhv OF bus_structure IS
17
18   COMPONENT shift_reg
19       PORT
20       (
21       -- Sequential Sychrounous Logic
22         clk, reset, load  : IN STD_LOGIC;                    -- register control inputs
23       -- Parallel input/output
24         pD     : IN STD_LOGIC_VECTOR(7 DOWNTO 0);            -- parallel input
25         pQ     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);           -- parallel output
26
27         pTest : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
28       );
29   END COMPONENT;
30
31   COMPONENT multiplexer
32       PORT
33       (
34       -- input/output
35         SEL    : IN STD_LOGIC_VECTOR(2 DOWNTO 0);         -- mux select lines input
36         Dout   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);           -- mux output
37         Rin0, Rin1, Rin2, Rin3, Din : IN STD_LOGIC_VECTOR(7 DOWNTO 0)
38       );
39   END COMPONENT;
40
41   COMPONENT control
42       PORT
43       (
44         F  : IN STD_LOGIC_VECTOR(4 DOWNTO 0);    -- function send to the control circuit
45         R  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- register control
46         S  : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)  -- register select lines
47       );
48   END COMPONENT;
49
50       SIGNAL RegSelect : STD_LOGIC_VECTOR(3 DOWNTO 0);    -- register being loaded
51       SIGNAL MuxSelect : STD_LOGIC_VECTOR(2 DOWNTO 0);    -- mux select lines
52       SIGNAL Rout0, Rout1, Rout2, Rout3 : STD_LOGIC_VECTOR(7 DOWNTO 0);
53
54   BEGIN
55
56       MUX:  multiplexer PORT MAP(MuxSelect,BusData,Rout0,Rout1,Rout2,Rout3,Data);
57       S0:   shift_reg PORT MAP(Clk,Reset,RegSelect(0),BusData,Rout0, DTest1);
58       S1:   shift_reg PORT MAP(Clk,Reset,RegSelect(1),BusData,Rout1, DTest2);
59       S2:   shift_reg PORT MAP(Clk,Reset,RegSelect(2),BusData,Rout2, DTest3);
60       S3:   shift_reg PORT MAP(Clk,Reset,RegSelect(3),BusData,Rout3, DTest4);
61       CNT:  control PORT MAP(Func,RegSelect,MuxSelect);
62
63   END bhv;
64
65
```

Figure 5. Screen capture displaying the **Quartus Prime logic design** for the **Bus Structure** design.

- Realization/Simulation –

Below, one will observe the operation of our design. The values selected were proposed by the lab instructor, and a few were randomly selected to further test the design and its operation.

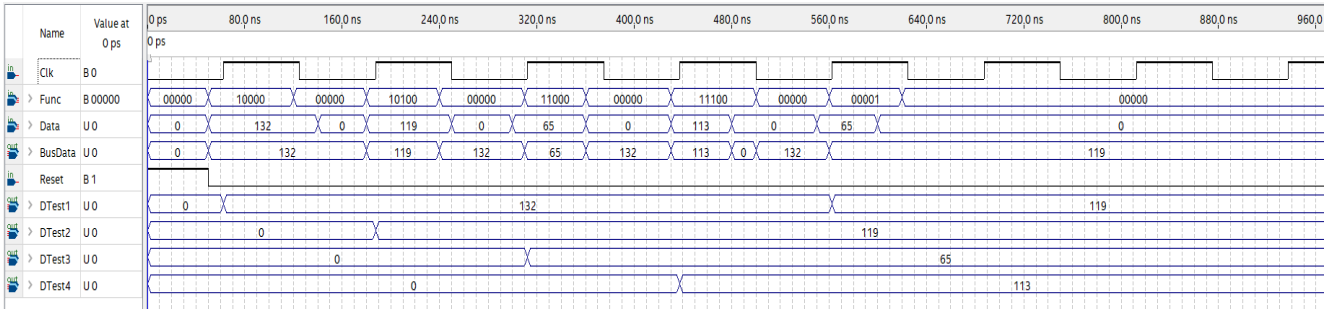

Figure 6. Screen capture disaplying the **Simulation Waveform** using test inputs.

- Summary -

In summation, this lab was a great tool and a particularly more complex design. It granted a better grasp of component and combinational logic designs.

*At first, we were having a hard time simulating the design. There was a misconception that the **VWF** simulation software used would register and track component design ports to be monitored in the simulation. But this is not the case. So, we had to create new ports in the top-level main design entity. After this addition, the simulation ran smoothly and exactly as expected.*