COMPUTER ARCHITECTURE

CSE 590

SPRING 2020

PROJECT 2: GUESSING GAME

BY

ROHITH KUMAR GADALAY

UBIT: rohithku

UB Person no: 50314899

PROCEDURE:

I have created a project with entity as GuessingGame and included all the required items like switches, leds ,anodes, clock, pushbuttons in the constraints file. I have created different variables like btnL, btnR, btnD, btnC, btnU, seg, anode_led ,clock, dp, switch inside the entity and mapped them to the items in constraints file accordingly.

In the architecture, I have declared all the shared variables that are used in the program and initialized them accordingly. The variable **state_value** is used for transition from one state to another. Below table shows the different states and its appropriate actions

state_value	Action Done
1	Display PL 1
2	Latching of player 1
3	Display PL 2
4	Latching of player 2
5	Display 2 HI
6	Display 2 LO
7	WIN state, shows number of tries and lighten the leds
8	Display LOSE for 3-4 seconds and redirect to state 1(PL 1)

When state_value:=1: (Display PL 1)

I have initialized the state_value to one(1) and whenever the bit file is run into the basys-3 board, **PL 1** is displayed. Anode 0 is used to display **1**, Anode 2 is used to display **L** and Anode 3 is used to display **P**. After this, the latching starts when slider switches are turned on and the state_value moves to two(2).

When state_value:=2: (Latching Player 1)

Latching of the player 1 starts in this state when the slider switches are turned on. **The latching works when at first D0 slider switch is turned on and btnL(left push button) is pressed**. After that any slider switch can be turned on and the digits are latches accordingly. Initially the anode segments show as **"0000"** and the value changes and gets displayed in the anode when the slider switches are turned on and the respective push buttons are pressed. The left push button(**btnL**) is used to latch D0, the upper push button is used to latch D1(**btnU**), the lower push button is used to latch D2(**btnD**), the right push button is used to latch D3(**btnR**).

I have created a temporary variable player1_data2 and initialized it to zero. I have created 4 different variables for storing values of 4 different slider switches result. So when the state_value is 2, and player1_data2 is zero I have run a case statement for the Anode 0 related value and stored the temporary output variable accordingly for different input(0-F) of slider switches. After the case statement ends, the acquired output variable is assigned to the actual segment variable(seg). The value

player1_data2 is then incremented and then Anode 1 related information is stored. This cycle runs for 3 loops and at the last iteration, the variable player1_data2 is made to zero so that the loop continues.

Whenever the player 1 decides to confirm a value, the center push button(**btnC**) is pressed and the entire value displayed in the anodes is stored in result1. Now the state_value is made to three(3) so that the next step(displaying PL 2) starts.

When state_value:=3: (Display PL 2)

When the center push button(btnC) is pressed in the state 2 the result of player 1 is stored and state_value moves to 3 and **PL 2** is displayed. Anode 0 is used to display **2**, Anode 2 is used to display **L** and Anode 3 is used to display **P**. After this, the latching starts for player 2 when slider switches are turned on and the state_value moves to four(4).

When state_value:=4: (Latching Player 2)

Latching of the player 2 starts in this state when the slider switches are turned on. **The latching works when at first D0 slider switch is turned on and btnL(left push button) is pressed**. After that any slider switch can be turned on and the digits are latches accordingly. Initially the anode segments show as **"0000"** and the value changes and gets displayed in the anode when the slider switches are turned on and the respective push buttons are pressed. The left push button(**btnL**) is used to latch D0,the upper push button is used to latch D1(**btnU**),the lower push button is used to latch D2(**btnD**),the right push button is used to latch D3(**btnR**).

I have created a temporary variable player2_data2 and initialized it to zero. I have created 4 different variables for storing values of 4 different slider switches result. So when the state_value is 4, and player2_data2 is zero I have run a case statement for the Anode 0 related value and stored he temporary output variable accordingly for different input(0-F) of slider switches. After the case statement ends, the acquired output variable is assigned to the actual segment variable(seg). The value player2_data2 is then incremented and then Anode 1 related information is stored. This cycle runs for 3 loops and at the last iteration, the variable player2_data2 is made to zero so that the loop continues.

Whenever the player 2 decides to confirm a value, the center push button(**btnC**) is pressed and the entire value displayed in the anodes is stored in result2. I have declared a variable **tries** to store the number of attempts of player 2 and player 1 result comparison.

After the result2 is stored, the values of player2 and player1 are compared. If player2 value is greater than player 1 value, then the state_value becomes **5** and **2 HI** is displayed. If player 2 value is lesser than player 1 value, then the state_value becomes **6** and **2 LO** is displayed. If player 2 value is equal to player 1 value, then the state_value becomes **7.**

If the attempts made by player 2 are greater than 4 then the state_value is moved to **8.**

When state_value:=5: (Display 2 HI)

In this state, **2 HI** is displayed when the player 2 stored value is greater than player 1 stored value. Anode 0 is used to display **I**, Anode 1 is for **H** and Anode 3 is used to display **2**. After this, the state_value moves to 4 and latching for player 2 can be done again and the same procedure continues. The tries variable is incremented accordingly.

When state_value:=6: (Display 2 LO)

In this state, **2 LO** is displayed when the player 2 stored value is lesser than player 1 stored value. Anode 0 is used to display **O**, Anode 1 is for **L** and Anode 3 is used to display **2**. After this, the state_value moves to 4 and the latching for player 2 can be done again and the same procedure continues. The tries variable is incremented accordingly.

When state_value:=7: (WIN and display number of tries)

In this state, when the player 2 value is equal to player 1 value then all the leds are lighten and the number of tries is displayed as a decimal value in the anode 0.

When state_value:=8: (Display LOSE and reset to PL 1)

I have considered that the player 2 is given at least **4** attempts for guessing the value of player 1. If the attempts/tries become greater than 4 then **LOSE** is displayed for approximately 3-4 seconds the game is reset and **PL 1** is showed again. Anode 0 is used to display **E**, Anode 1 is used to display **S**, Anode 2 is used to display **O** and Anode 3 is used to display **L**.

I have considered a counter variable for delay of 3-4 seconds and ran it for the condition when counter>600 before showing **PL 1** again.

After **PL 1** is displayed, the latching can be done again for player 1 and the entire procedure from state_value **1** continues.

For the dp segments:

I have considered the condition when each anode value of player 1 is equal to player 2, I have set dp<='0' which indicates that it is a correct guess and if values are not equal then I have set dp<='1' indicating it as an incorrect guess and the respective dp segment will glow.

This scenario is working for only some cases.

Mistakes done and rectified:

- I have taken the constraints file of project 1 in which only the center push button was used. I was getting ports related issue. I forgot to uncomment the remaining push buttons and wasted almost 2 hours searching for the issue. Then I had a look at constraints file and solved the issue.
- I had some issues with elsif loops and they were behaving weirdly even though the logic is right. Then I created individual if loops wherever I was facing the issue.
- While storing the result of player 1 and player 2 I concatenated the outputs of anodes as **Anode0 & Anode1 & Anode2 & Anode3**.I was getting incorrect display values of **2HI** and **2LO** and was searching for the issue for about 24hours. When I identified the issue that the result should be concatenated in order of **Anode3 & Anode2 & Anode1 & Anode0**, the functionality was working as expected.
- I tried to implement the dp segments as stated in the above scenario, but unable to achieve the full functionality of it.