Matthew Simiele
EECE 251
Final Report
12/6/17

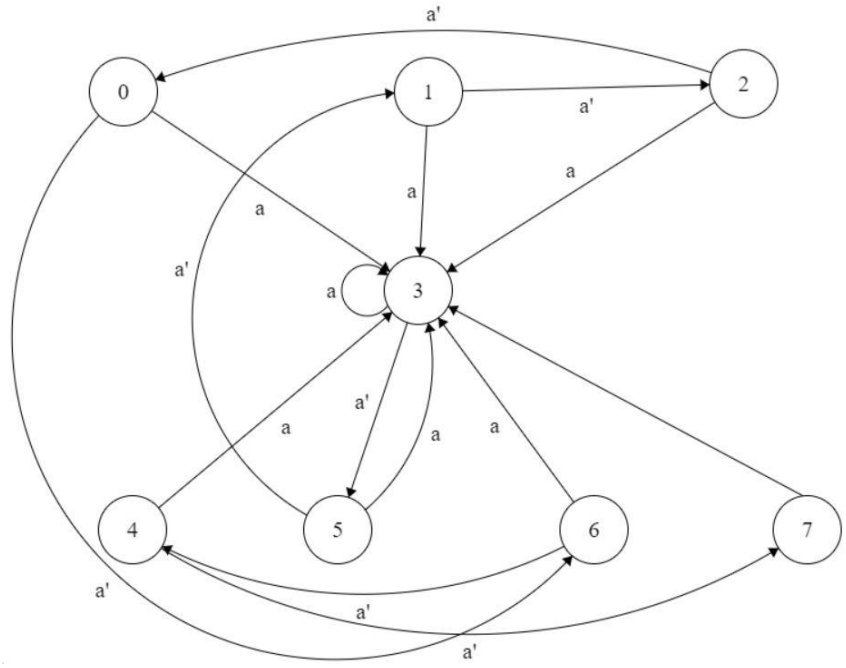
Project Description:

Whac-a-Mole is a popular arcade game in which a player holding a foam mallet attempts to whac moles coming out of their mole holes. In my project the moles were represented by LED's that were switched on and when they were on the player used the switch underneath that LED to "whac" the mole. In my project proposal I stated that every time a person "hit" a mole they would receive 10 points and the score would be increased on the display. This design originally included the timer being just an internal component instead of being outputted to the seven-segment display. I decided that it would be useful to the player of the game to see how many turns of the game has been played. When the user can see this timer, they know how much time they have left to score as many points as they can. The timer counts from 0 to 25 and when it reaches 25 all the LED's will shut off and not turn bac on until the user presses the start game button.

Component Descriptions:

Mole Mover – The mole mover component is an 8 bit register that cycles through LED's turning them on and off to represent the mole moving. This component is influential to this project as without it the mole would stay stationary and not give any challenge to this game. The only input to this circuit is the button to start the game. When the button is pushed in for a second or two the LED 3 as labeled on the Papilio-Duo will turn on and after the button is released the game will start. In the below FSM the current state represents the LED that will be turned on.

Mole Mover FSM:



This circuit has 8 outputs one for each of the LED's these outputs are inputted into the Pulsers and Mole component in the schematic and displayed directly to the LED's above the switches. Each light will only go on if the light previous in the cycle was on and the input button "a" was low so I was able to determine these equations for each light.

Light 7 = Light4 * a'
Light 6 = Light0 * a'
Light5 = Light3 * a'
Light4 = Light6 * a'
Light3 = (Light7 *a') + (Light7' * a) (this light needs to be high in 2 cases the previous light was on or the button to start the game over has been pressed.)
Light2 = Light1 * a'
Light1 = Light5 * a'
Light0 = Light2 * a'

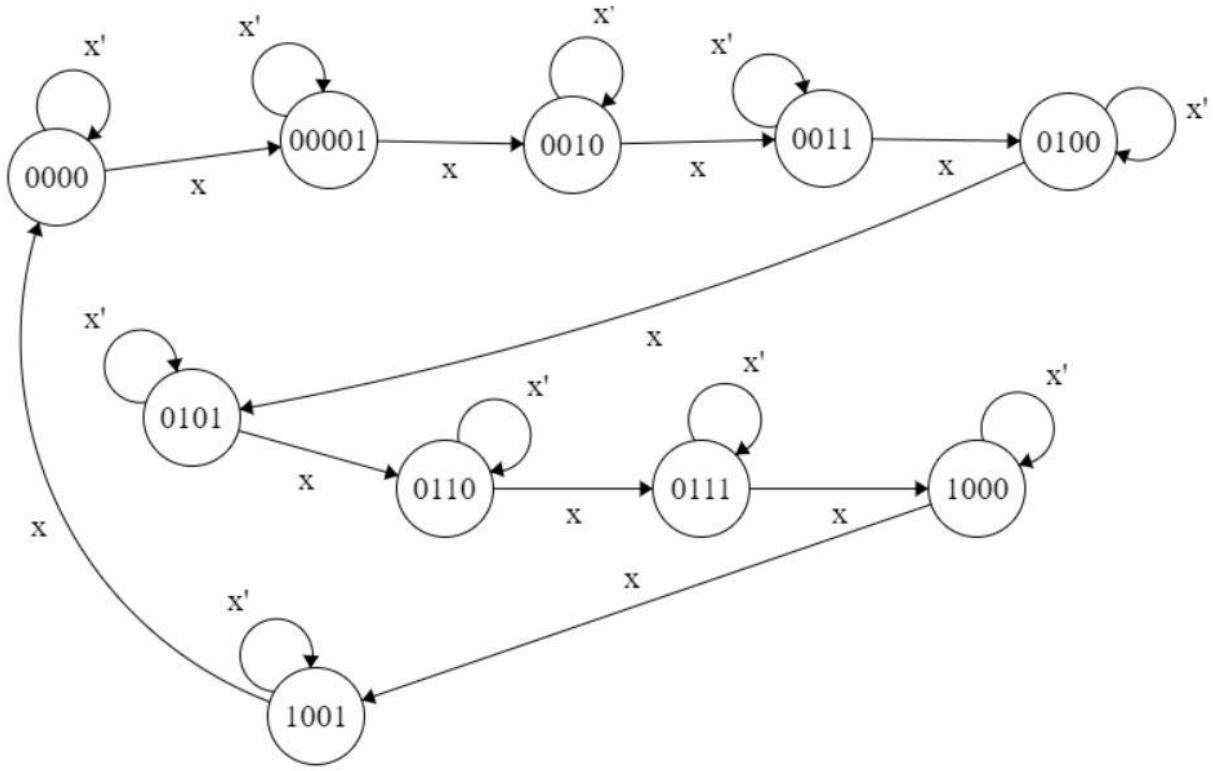
Pulsers and Mole – This component serves two purposes the first purpose is to pulse the input from the switch so that if a switch is left up it does not trigger every time the light above that switch is turned on. The second function is to line up the pulse from the switch to the LED output. To do this an extra D-Flip-Flop was added to the input of which LED is on. The switch inputs need to be connected to a pulser so that when the switch is up it does not continuously increment the score counter. The inputs to this component is each switch and all the values of the LED's. The values of the LED's are put into a flip-flop which extends the value to one clock cycle after so that when the switch inputs are put through a pulser the two values both line up and can be compared correctly. The output of this component drives the score so when any one combination of switch and LED is on the output value must be equal to 1. Therefore, each comparison between switch and LED is ored together.

Clock Divider - The component referred to in the schematic as the "ClockDivideTest" and the setup using the flip-flop and inverter next to it divides the clock down to equal 1 Hz instead of it being equal to 32 MHZ so the game can run at a rate in which it able to be interpreted by the user. This device takes an input which is P94 (the internal clock) and an output which is the clock slowed down to a frequency of 1 Hz. Each output of a flip-flop inside this circuit represents the clock for the next. So, this function can be implemented you need an inverter to switch the input to the flip-flop between 0 and 1 otherwise the clock for the next flip would do nothing because the signal would be constant and would not produce a change from high to low, so the input value could be moved to the output of the flip-flop.

Counter – This component is useful in performing two other functions which are crucial in the operation of the game. The main premise of this device is when an input is high that it will cycle through from 0 to 9 to increment some number whether it be for a score board or a timer. The reason I used this component is that I needed a way to increment my timer and score, so these values could be displayed to the user. The inputs that this component takes in is a value that controls when the number will be incremented by 1. And the output is a 4-bit binary number between the values of 0 and 9. In the below diagram the states are represented by the 4-bit binary number the circuit will output. And the value "x" is the input in which will dictate whether to increment the output value by one. Another output for the counter is the "Carry_Out" which is influential when building a line of counters that will count tens and

even hundreds. Carry_Out is high when the binary number of the current state is 9 and the input to the counter is high.

Counter FSM:



Counter Truth Table:

Q3	Q2	Q1	Q0	X	D3	D2	D1	D0	Carry_out
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	0
0	0	1	1	1	0	1	0	0	0
0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	0	1	1	1	0
0	1	1	1	1	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0
1	0	0	0	1	1	0	0	1	0
1	0	0	1	0	1	0	0	1	0
1	0	0	1	1	0	0	0	0	1

The reduced equations for D[3:0] are as follows by using Boolean Algebra:

D3 = Q3'Q2Q1Q0X + Q3Q2'Q1'Q0' + Q3Q2'Q1'Q0X'
D2 = Q3'Q2'Q1Q0X + Q3'Q2Q1' + Q3'Q2Q0' + Q3'Q2X'
D1 = Q3'Q1'Q0X + Q3'Q1Q0' + Q3'Q1X'
D0 = Q2'Q1'Q0'X + Q2'Q1'Q0X' + Q3'Q0'X + Q3'Q0X'
Carry_Out = Q3Q2'Q1'Q0X

Timer – The timer component of the game consists of 2 counters in which the carry out of one is hooked up to the input of the other in order to count the tens values of the timer. The timer is used to create a time limit for the use to play the game. In the case of this Whac-a-Mole game the timer is set to roughly 25 seconds or 25 cycles of the divided clock. This is done by comparing the output of the tens place to the binary value of 2 and the ones place to the binary value of 5. When both these cases are true the input of the first counter is turned off thus stopping the incrementing of the displayed number. When the number displayed is equal to 25 the LED's all are switched off as well to stop the game from being played. When the player wants to start a new game, the timer is reset by the push of the right button, so the count starts back at 0. The output of these two counters for the timer are placed into the two leftmost seven-segment displays on the sseg component given to us.

Score Counter – The score counter keeps track of the players score as he hits the mole. This component works by taking the output from the pulsers and moles component and using it as its input to increment. Since the clocks for both the components are the same the input will only be high for one cycle while the counter increments by one. This counter is also reset by the switch, so the score is cleared for the next game. The score is also made of two counters so that the input of the second one is the carry out of the previous one. The score counter also outputs a binary number for each place what the score is currently and is connected to the two right most seven segment displays using the sseg component as well.

SSEG – This component is used for the sole purpose of displaying more than one number on the seven-segment display at a time. It takes in inputs for all 4 numbers or characters to display in binary and each decimal point. My basic understanding of this component is that because it runs on a faster clock than the rest of the circuit the display can be updated for all 4 places quickly rather than only displaying one value on all 4 seven-segment displays. The output of this circuit is also a 4-bit binary number than now can be converted into a set of values to be displayed on the seven-segment display.

Bin2Segs – Uses the idea that the seven segment displays idle on ability to convert a 4-bit binary number using a decoder to a set of values for each individual segment. The decoder only has combinational logic up to the value in which the number 9 in binary would trigger it because a number bigger than 9 will never reach it. The reason this component is implemented into the design is that it helps convert a binary number into a set of seven values corresponding to each segment so that the user sees the numbers of 0 – 9. The component takes in 4-bit binary number and displays that number on the seven segment displays. The outputs of this component is the values seen on the seven segment display.

Tests:

The tests used in verifying this project worked were on a component to component base at first. The first component to be designed was the Mole Mover which takes up most of the Whac-a-Mole project file. This component was first tested using the approach of using a debounced switch as the clock to move the mole. The next component to be tested was the counter along with the Bin2Segs components. First one counter was tested on its own to make sure it counted in decimal from 0-9 and then went back to 0. This was first done using the same approach as with the MoleMover using a debounced switch as its clock input. Then two counters were added together to make sure that the implementation I was looking for when I needed the counter to count to 25 would work. This implementation of the two counters first was tested using the debounced switch as its clock method. When it came down to testing the score counter since all other components in the counter were working I was just testing to make sure that moles and switch inputs lines up. When at first when I flipped up a switch with the LED on and it didn't register the point I knew I needed to add a flip flop to the mole signal to produce a signal on the output when I wanted on to occur. After this was fixed the game was working!