Lab #6

Design Project

Lab Report

Gurkirat Dhillon and Bradley Manzo
917578751 916953788

TABLE OF CONTENTS

INTRODUCTION

OBJECTIVES

- To design a dice game by implementing various components such as a turn counter, 1-to-6 counter, full adder, score register, and a control module. The game is about scoring exactly 15 to win or more to lose. The design of the control module is a mealy machine to have an instant dice roll. The 1-to-6 counter includes a reset state at 0 which moves to 1 in the counter. The counter will be clocked by the internal clock of the DE-10 board to simulate a random number generator.

PROCEDURE

DESIGN AND TEST PROCEDURE

- The design for the dice game involves 5 main modules: the turn counter in *Figure 10*, the 1-6 Dice Roll Counter designed in Lab 5 in *Figure 2*, the full adder in *Figure 7*, the score register in *Figure 6*, and a control module in *Figure 12*.
- The control module controls the specific states of the game determining what state the FSM is in, what it's next state will be, and what inputs are allowed in what states. The control module consists of state expressions shown in *Figure 15* .
    - State 0 is the reset state, and the beginning state of our machine. The machine will return to this state whenever the reset button is pressed.
    - State 1 of the game checks for when the user has inputted Rb, the roll button, to roll a dice number from 1 to 6. This was implemented with a moore state machine counting from 1-to-6. This counter was then clocked with the DE-10 board's internal clock of 50 MHz to simulate a random number generator for the dice roll. After the user has pressed roll, the counter will stop at a randomized number from 1-to-6 on the HEX display.

    - State 2 of the game checks the dice number rolled and requests whether to add the rolled number or to pass and roll again. This state also increments the turn by one to keep count of turns from 0 to 99 turns which is shown on the HEX displays. If the user chooses to pass the dice roll, the roll button is released and the user can press roll again to roll another dice number.
    - In state 3, if the user chooses to add the dice roll, the dice roll is sent to the full adder to be added. The result is sent to the score register that keeps track of the score in binary which is shown on 4 LEDs. This action cannot be repeated with the same dice roll, in other words, a dice roll can be only added once to the score register. After the user has accumulated a score of 15 or more, the score is checked to be 15.
    - Since 15 is the maximum value of a 4-bit binary, any value over will produce a remainder. If a remainder is produced, this determines that the user has exceeded a score of 15 and has lost. If a remainder was not produced, this determines that the user has gotten a score of 15 and won the game. The score check in *Figure 4* checks if the score is less than or equal to 15.
    - States 4 and 5 are the win and lose conditions, and an LED will tell the player if they have won or lost. The user cannot roll again and has to reset the game.
    - Additionally, this game also has a special dice roll, six. If the user rolls a six, then the six is automatically added to the score register. This is implemented as an add signal which the user can roll again after rolling a six.
- All the inputs used as a switch were “debounced” by converting the signal to a pulse and can be seen in *Figure 13*. This is required because of the clock. Since the clock repeatedly toggles between high and low, the signal from a switch can be held for several clock cycles instead of a single cycle. In other words, the input is held for a few clock cycles. This is converted to a pulse by using a level to pulse converter shown

in *Figure 12*. The component shown in *Figure 13* was a level to pulse converter connected to another level to pulse converter without knowing that the level to pulse converter was a debouncer. As such, the circuit is called a debouncer to level to pulse converter. This causes issues which were resolved once the mistake was realized.

- The score register in *Figure 6* is made of D-flip flops with an enable. This allows the register to only store a score when the Add signal is high. This is shown in *Figure 8*. By feeding the output of the adder into this register, our FSM will remember the current score and send it back to the full adder to add to the new roll number.
- In addition to debouncing our switch inputs Add and Roll, we have implemented combinational logic in *Figure 5* to limit the states of the machine each can be triggered in. By inhibiting the roll button with the lose conditions notted into an and gate, we are allowed to roll as long as we have not exceeded the score 15. The add input is inhibited by the roll button so that it can only be triggered when Roll is in the high position. Furthermore, Add is anded with state 2 meaning Add will only output a high signal when this specific state.
- The win and lose conditions in *Figure 9* work in a similar way, outputting a high signal on the win or lose LEDS when the conditions in *Figure 4* are met

ANALYSIS/RESULTS

DATA

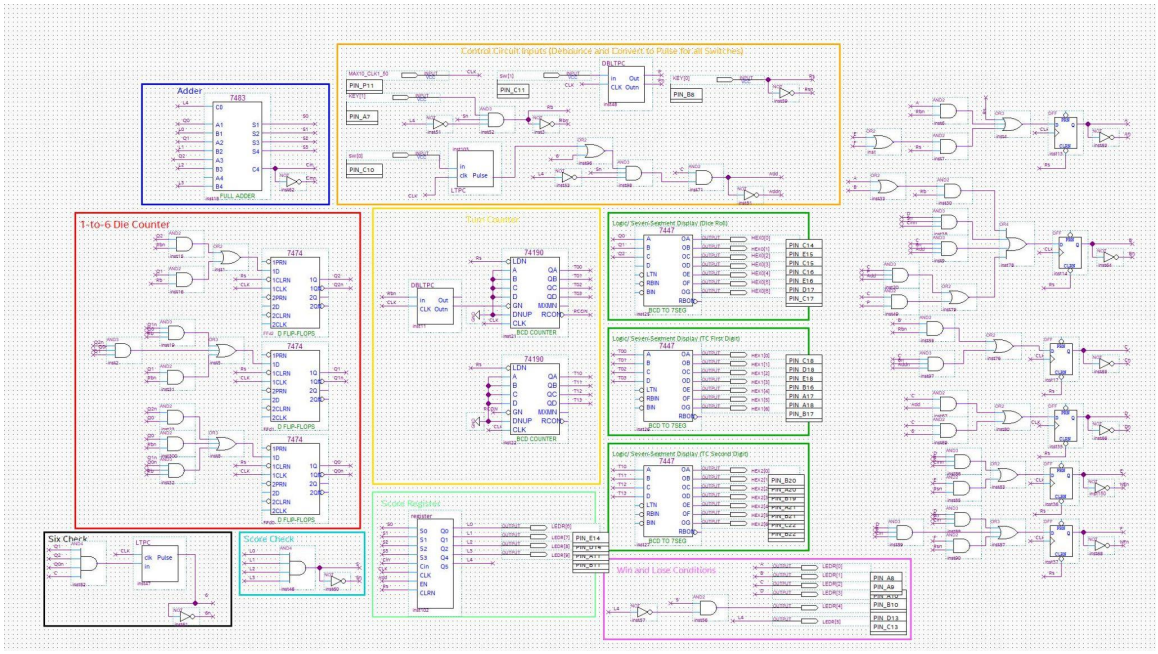


Figure 1: Top Level Circuit Design

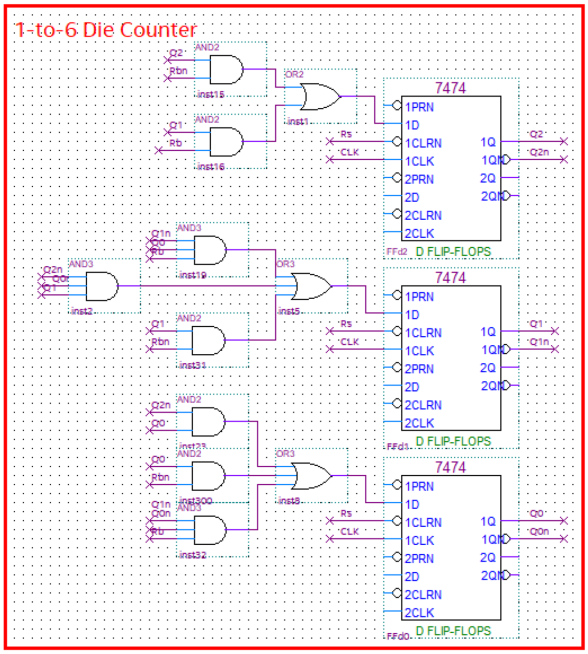


Figure 2: 1-6 Dice Roll Counter from *Lab 5*

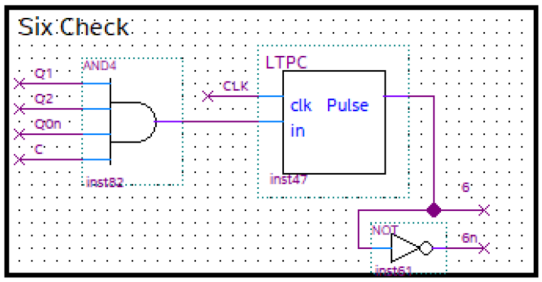
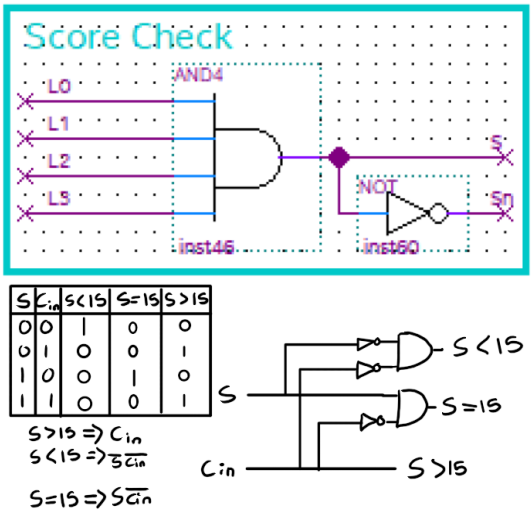


Figure 3: Roll 6 Checker



S	C_{in}	S<15	S=15	S>15
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	0	0	1

$S>15 \Rightarrow C_{in}$

$S<15 \Rightarrow \overline{S}\,\overline{C_{in}}$

$S=15 \Rightarrow S\overline{C_{in}}$

Figure 4: Score Condition Checker

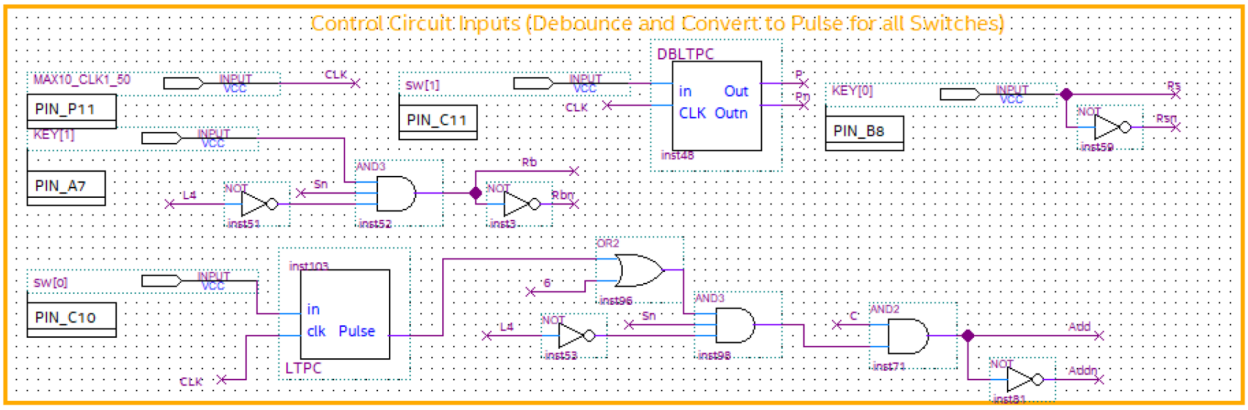


Figure 5: DE10-Lite Inputs

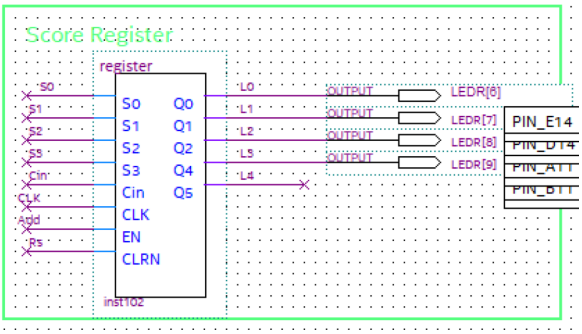


Figure 6: Score Register

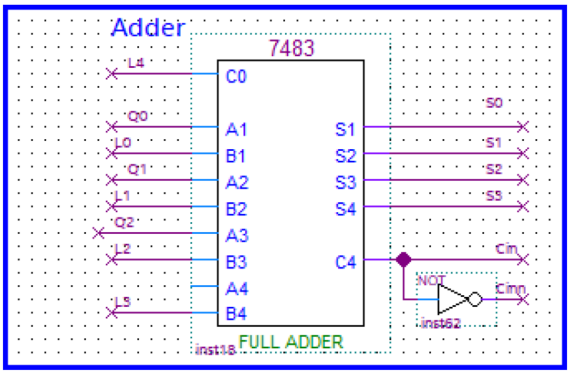


Figure 7: Full Adder

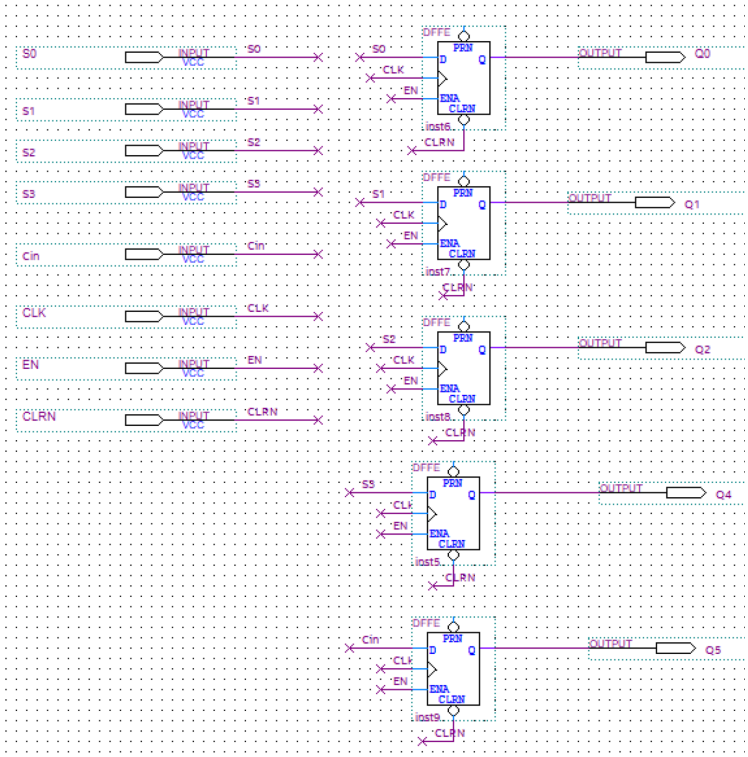


Figure 8: Score Register Lower Level

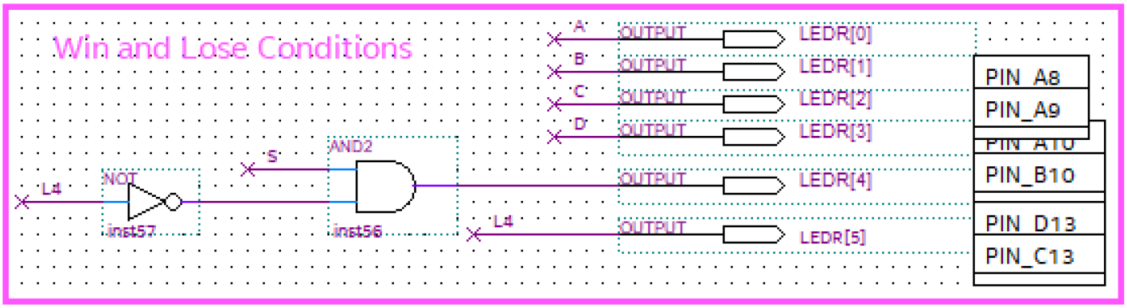


Figure 9: Win and Lose Conditions

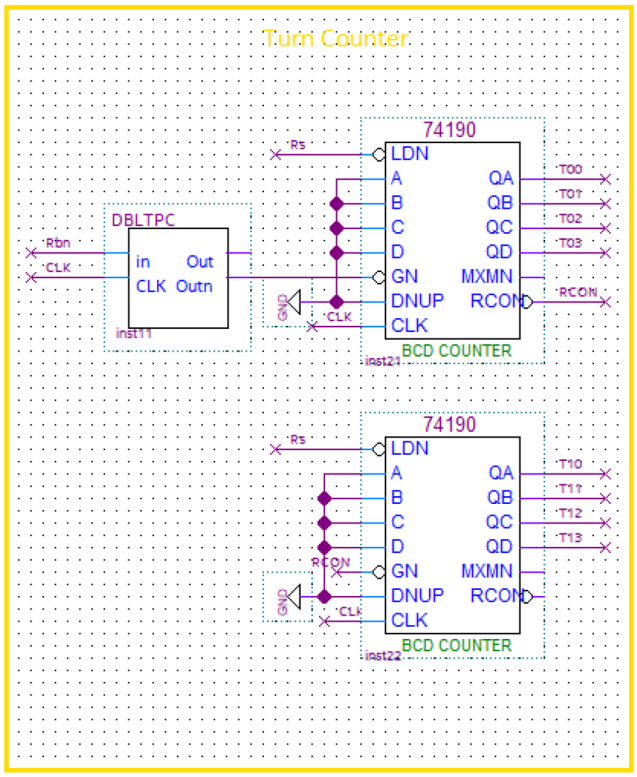


Figure 10: Turn Counter

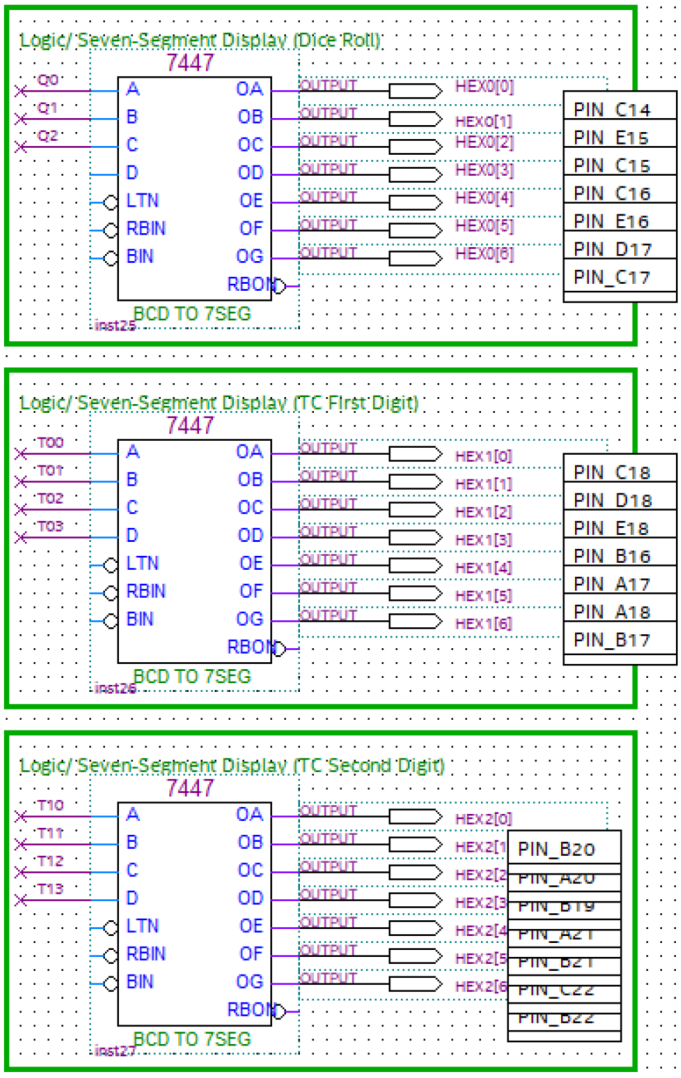


Figure 11: Roll and Turn Seven-Segment Displays

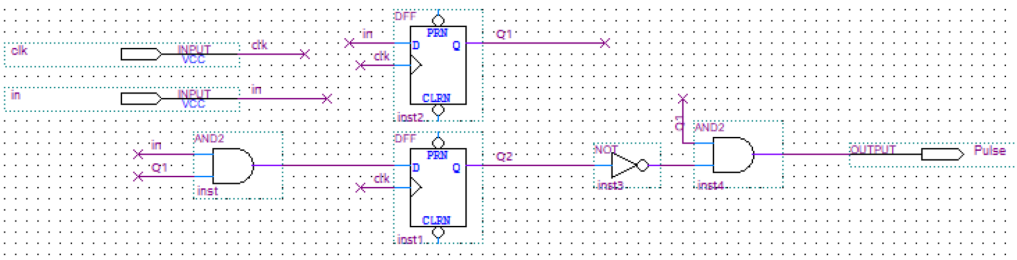


Figure 12: LTPC Lower Level

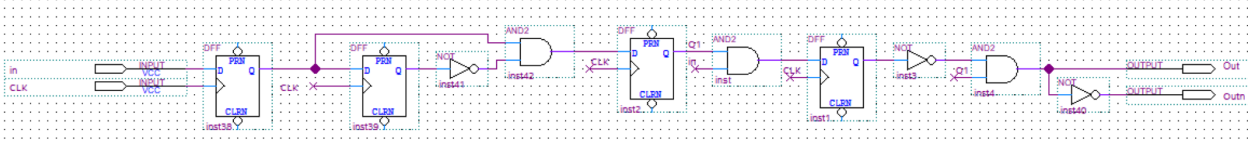


Figure 13: DBLTPC Lower Level

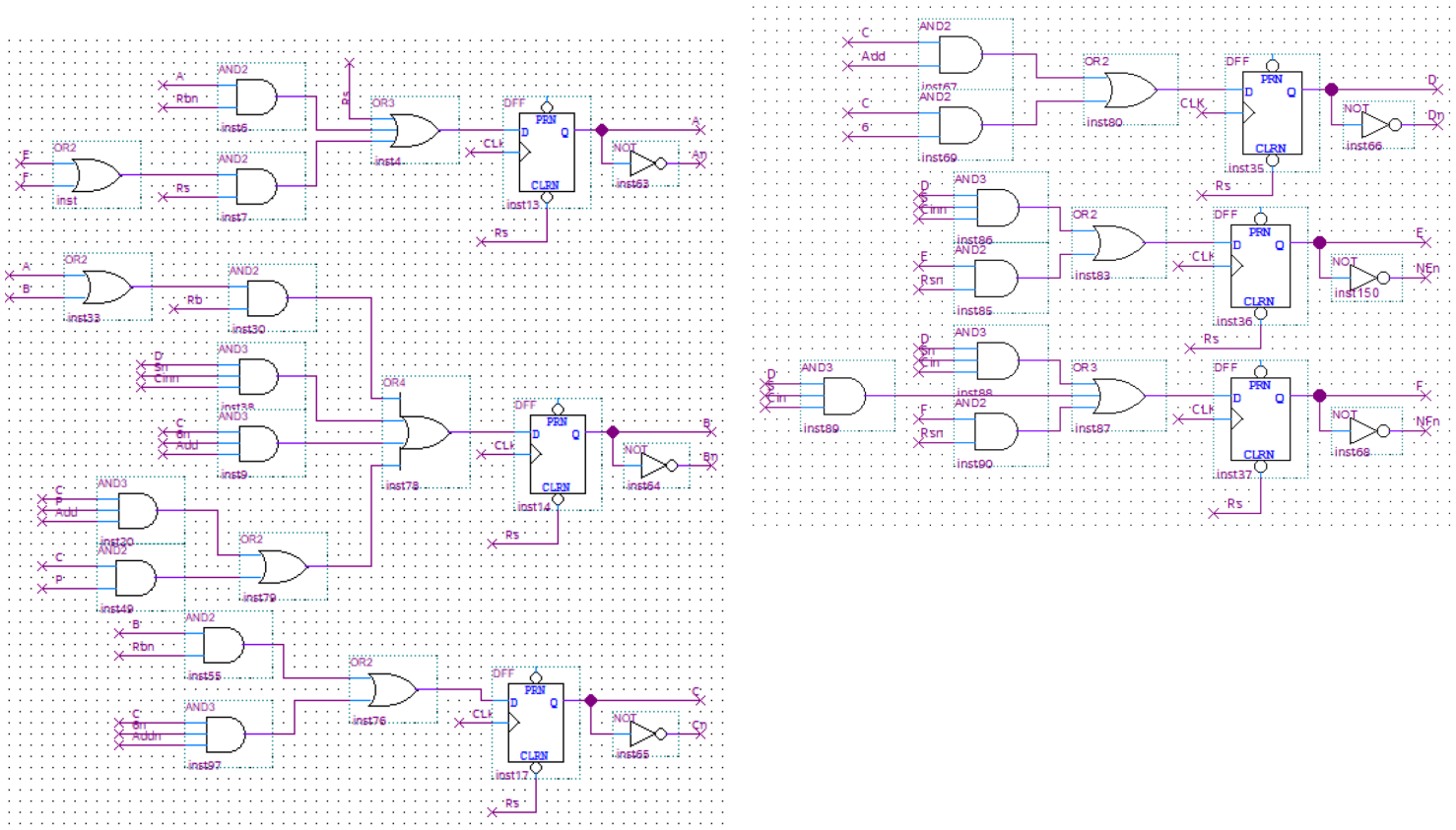
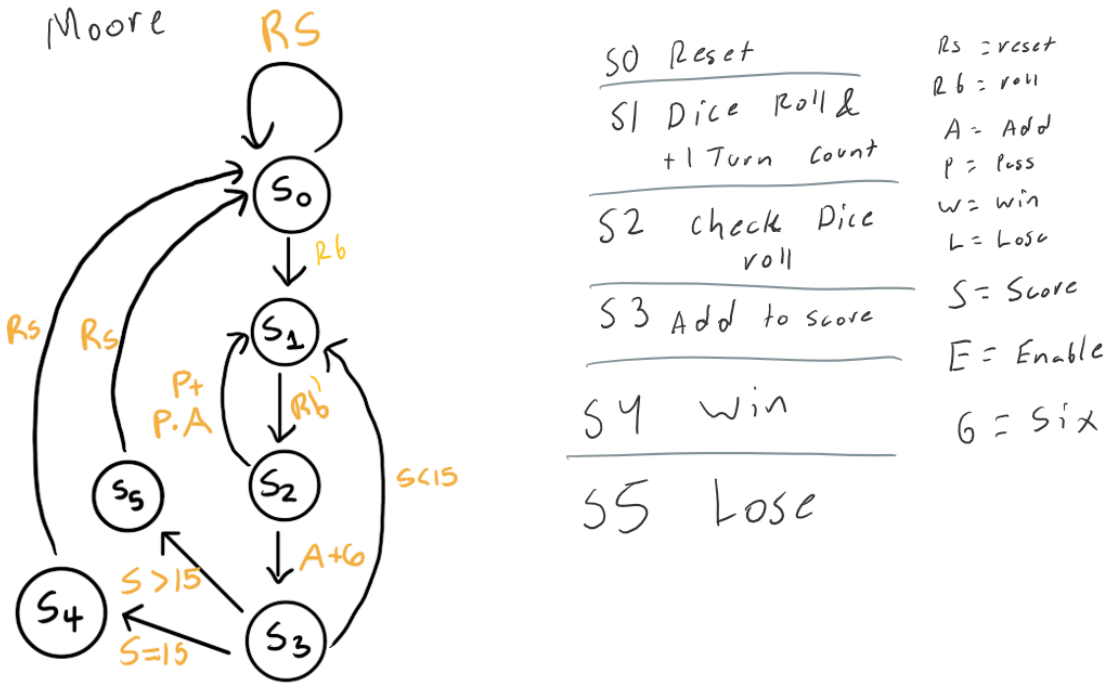


Figure 12: Control Module Design



S0 Reset

S1 Dice Roll & +1 Turn Count

S2 check Dice roll

S3 Add to score

S4 win

S5 Lose

Rs = reset
Rb = roll
A = Add
P = Pass
W = win
L = Lose
S = Score
E = Enable
6 = six

Figure 13: Moore FSM Design

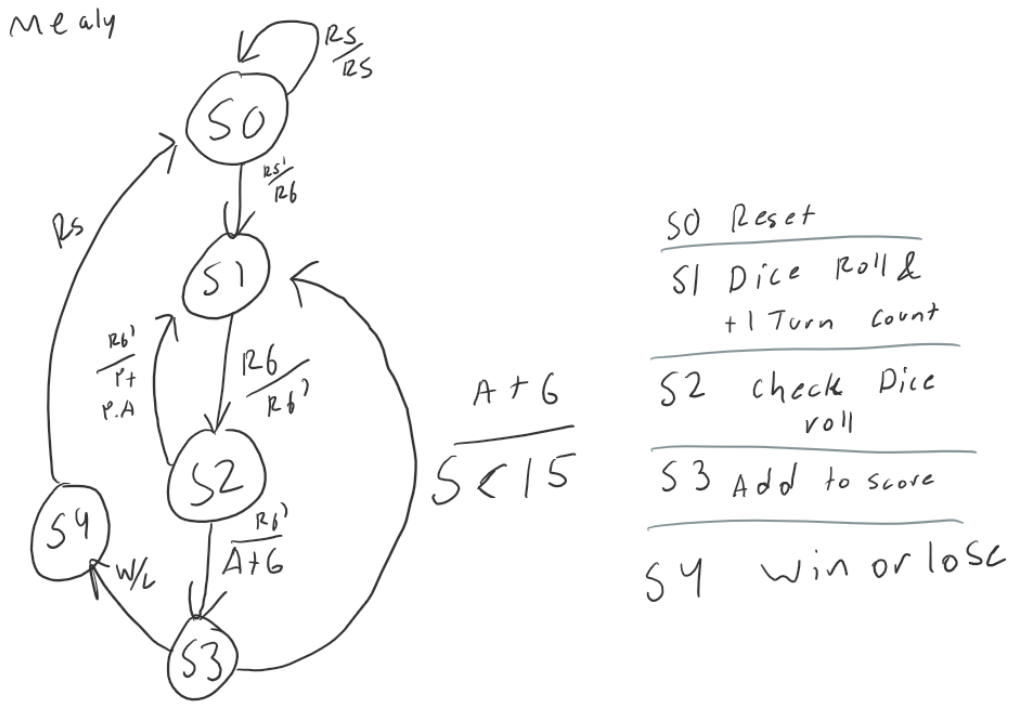


Figure 14: Mealy FSM Design

$$S0 \quad NA = Rs + A.\overline{Rb} + A.\overline{Rs}$$

$$S1 \quad NB = Rb.(A+B) + B.\overline{Rs} + D.\overline{S}.\overline{Cin} + C.Add.P \quad + C.P$$

$$S2 \quad NC = B.\overline{Rb} + C.\overline{6}.\overline{P}.\overline{Add} + .C\,\overline{Rs}$$

$$S3 \quad ND = C.Add + C.6 + D.\overline{Rs}$$

$$S4 \quad NE = D.S.\overline{Cin} + E.\overline{RS}$$

$$S5 \quad NF = D.\overline{S}.Cin + D.S.Cin + F.\overline{RS}$$

Figure 15: Control Circuit State Equations

ANALYZING RESULTS

- With the results we found from designing the dice game in Quartus Prime, we are concluding that the game is functional. The game works when a six is not rolled, meaning our dice roll is functional, and displays the rolled value and counts the turns, the score is

added using SW0 to the register which remembers the previous roll, the win conditions stop the game after the player has earned exactly or exceeded 15 points. The score is displayed on the LEDs, and the When attempting to fix this issue, the game would then refuse to add other dice rolls other than six. This was apparent when checked by a TA.

CONCLUSION

- A pass button was not implemented because the design nature of our dice game is a hybrid of a moore and mealy state machine. In other words, the dice roll acted like a mealy machine and “kept rolling” until stopped by the roll button. The roll button would hold the dice value and the user could decide to add the value or pass by releasing the roll button. Since the pass button is required for check off, this grants partial credit for the implementation of a pass button. The rest of the game is treated as a moore machine. Overall, the game works and can be played based on the schematic in figure 1. We learned about how to effectively utilize store information with D-flip flops which ended up as the control of the schematic design. We also deepened our understanding of finite state machines and one hot encoding in particular due to this lab due to the large emphasis on specific states for the dice game. By implementing the game, this only deeped and solidified any concepts about finite state machines.