UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC 18 **DIGITAL SYSTEMS I** **Fall 2024**

Lab 6: Dice Game 19

Objective: In this lab, you will design a dice-based game called 19. The object of the game is to score exactly 19 points in as few rolls of a simulated die as possible. After each roll, the player can choose to add the current die value to his or her score or not, unless the die value is six, in which case it is automatically added to the player's score. If the player's score exceeds 19 or the total turn count reaches 7, the player loses. If the player's score is equal to 19, the player wins.

Preparation (Pre-lab)

- Draw a preliminary state diagram for the game as either a Moore or Mealy finite state machine.
- Determine the components (gates, etc.) you will use for the turn counter, die counter, and score circuits, and list them out.

Description

To start the game, the player resets the system using the *reset* input. The score will be set to zero ("00"), the *ready_to_roll* LED will be on, and the *win*, *lose_19*, and *lose_turns* LEDs will be off. The player then rolls the die by asserting the *roll* signal. The die counter operates with a clock frequency of 50 MHz so that the player cannot control the outcome. When *roll* is de-asserted, the roll ends. The signal *turn_count* is then incremented by one to indicate that the player has taken a roll. If the turn counter reaches 7, the *lose_turns* LED is lit and remains on until the system is reset.

The die-rolling circuit must operate only once and only at the proper time. For example, a player cannot roll again after a roll until either the *accept* or *reject* button is toggled, or unless a six is rolled. Also, the player cannot roll the die after the game has ended.

If the player rolls a six, it is automatically added to his or her score. Otherwise, the player can toggle the *accept* switch to add the die value to his or her score or the player can toggle the *reject* switch to leave the score unchanged. After each roll, the score will be greater than, less than or equal to 19. If the score is equal to 19, the player wins – the *win* LED is turned on and remains on until the system is reset. If the score is greater than 19, the player loses – the *lose_19* LED is turned on and remains on until the system is reset.

If the score is less than 19 and the turn count is less than 7, the player will roll again. After each roll, the player toggles either the *accept* switch or the *reject* switch (unless a six is rolled) before the die can be rolled again.

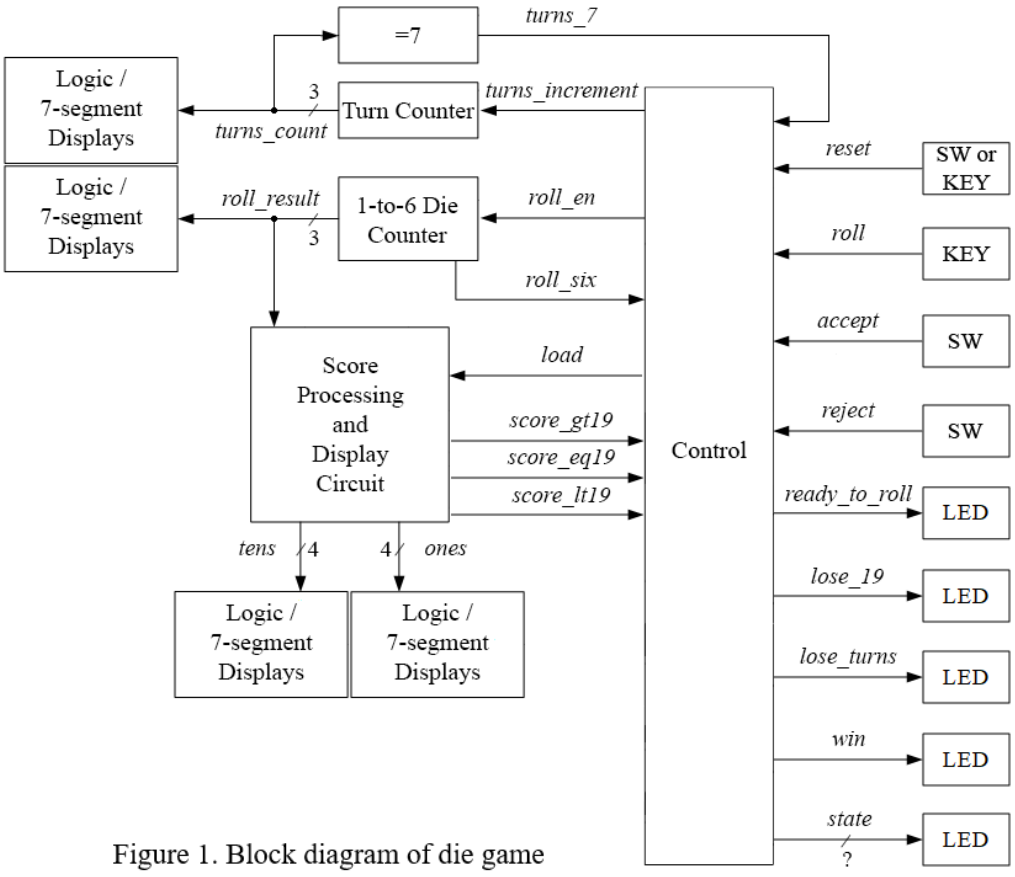

Figure 1. Block diagram of die game

Play continues until the player's score equals or exceeds 19 or the player has used 7 turns.

A block diagram of the system is shown in Figures 1 and 2 and a flowchart is shown in Figure 3. While three *score_** signals are shown so the controller knows the relevant range of the current score, all three are not necessarily needed.

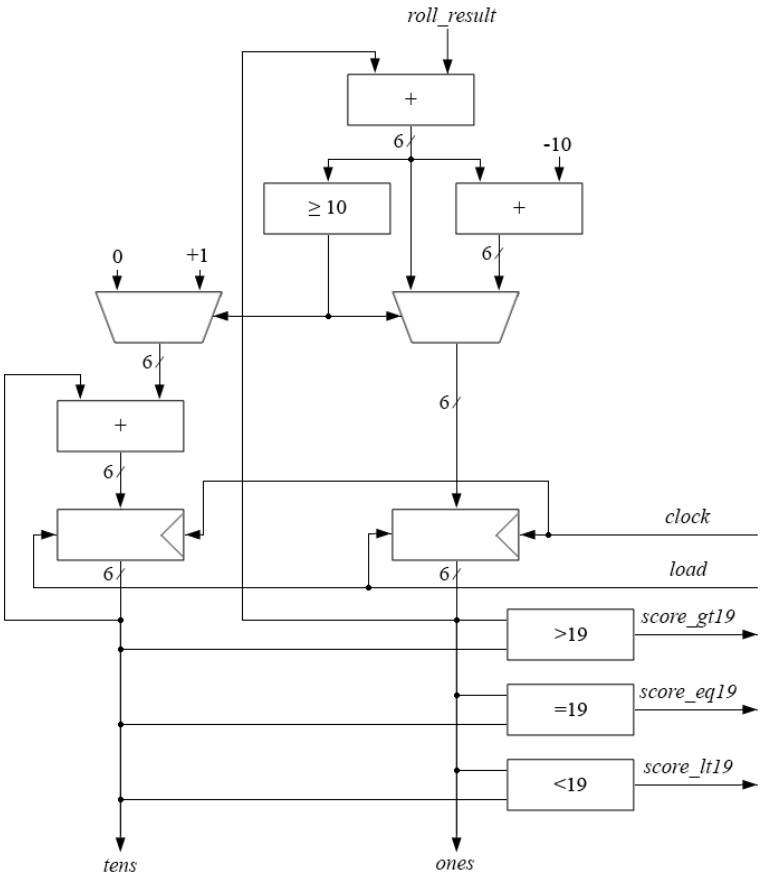

Figure 2. Schematic of Score Processing and Display Circuit

Design Requirements

1. Use the counting sequence assigned to you in the previous lab for your die.
2. Use only flip-flops clocked by a single 50 MHz system clock for all single-bit memories. The clock signal cannot be "gated"—that is, the single clock must directly clock all flip-flops.
3. Assume the pushbutton switches, KEY[1..0], are bounceless since they have been debounced on the DE10-Lite board using a Schmitt trigger circuit. However the SW[] slide switches are *not* debounced—in other words, the output when switching from one value to another will typically make many transitions before settling at the final value (e.g., ...0000-1-0-1-0-1-0-1111...). Design your state machine to work properly under these circumstances.
4. Display the number of turns in *decimal* ranging from 0 to 7 on one 7-segment display.
5. Display the score in *decimal* on two 7-segment displays. You must correctly display scores up to the highest possible score. (What is it?) Follow the schematic in Figure 2 to design the Score Processing and Display Circuit. Use Ripple Carry Adders created from Full Adders for your design. After a dice roll, add the counter output to the BCD ones score register. If the sum is less than 10, the ones and tens values are correct. On the other hand, if the sum is 10 or greater, subtract 10 from the sum to find the correct ones. The BCD tens score register then needs to be incremented by 1 in this case.
6. Make sure that your game is "well-behaved" under unspecified circumstances, for example when *accept* and *reject* are activated at the same time following a roll. Based on the flowchart in Figure 2, the *reject* switch should have priority over the *accept* switch so that if *reject* is pressed, the state machine should transition directly to a new roll state without updating the score even if the *accept* switch is also pressed simultaneously.
7. Display any useful information you can think of on unused LEDs to help in debugging.

8. Implement your design in a Quartus schematic using *only* the following components: Basic logic gates (AND, OR, NAND, NOR, NOT, etc.), DFF, DFFE, 7447, 7474, 7483, 74163, 74190, WIRE, INPUT, OUTPUT.

Design Guidelines

1. Carefully evaluate design alternatives. For example, determine if you will implement your control circuit as a Mealy or a Moore finite state machine. Also, determine which state assignment encoding will be optimal from choices such as one-hot, binary or other encodings. Note that using a one-hot encoding can greatly simplify the design process and the fact that it uses extra flip-flops doesn't matter since you are implementing your design in an FPGA with more than enough resources.
2. If your design doesn't work and you cannot debug it by using LEDs on the state register outputs and other signals, try manually clocking it with a debounced switch.
3. Design and test your system by breaking it into small modules and testing them individually, as needed. You should also simulate your design in ModelSim before implementing on the FPGA board.
4. Remember to import the pin assignments after you add all your input and output components!

Extra Credit

- (+5 max) Additional interesting and useful debug capability
- (+5 max) Additional interesting new game feature

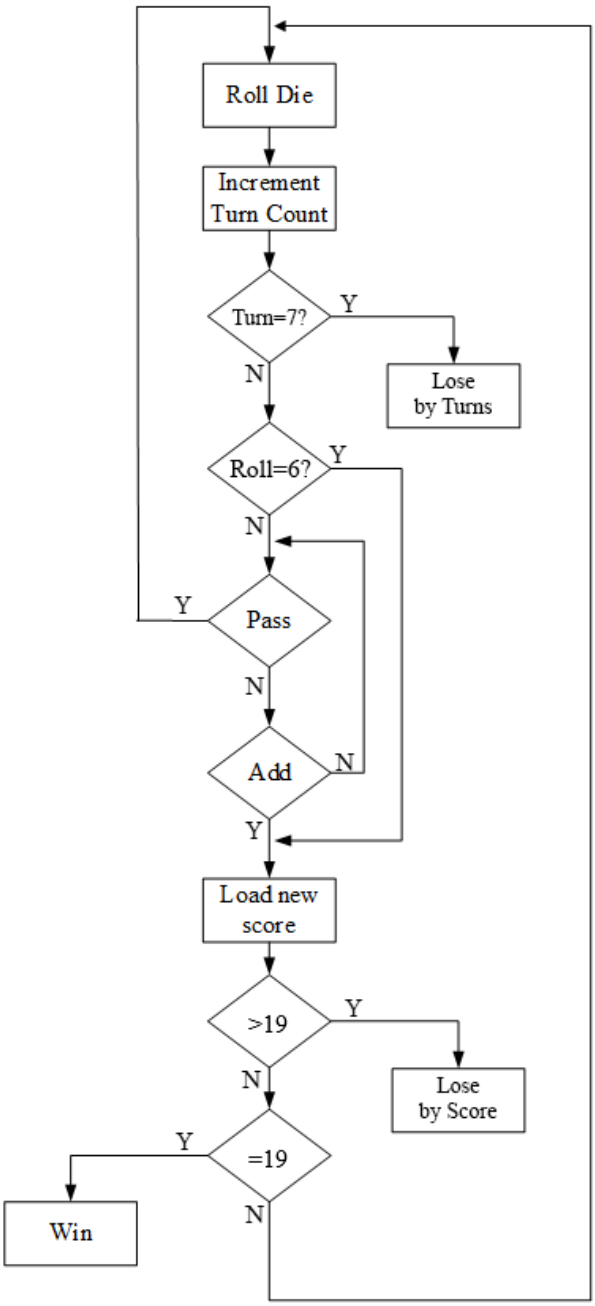

Figure 3. Flowchart for die game

Lab Report

Each individual must submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Also include the following items:

- Graded pre-lab
- Complete documentation of your design including:
  - State diagram
  - State assignments
  - State table
  - Next state equations
  - Tables, Kmaps and all other work showing how you made your design
  - Complete Quartus schematic
- Explain how and why you chose the state assignment encoding that you did.
- Based on your design, what happens if the Add and Pass switches are both activated simultaneously after a roll? Explain how this is implemented in your design.
- Suggest at least one new feature you would add if you had more time.

Grading (200 maximum + 10 possible extra credit)

Prelab	25 points
Lab Verification TA sign off	100 points
Lab Report	75 points

2024/12/02 Added three signals originating from the Score Processing and Display Circuit