Game Show Controller

Nhat Nguyen & Brandon McCoy & Tseveendorj Munkhtur
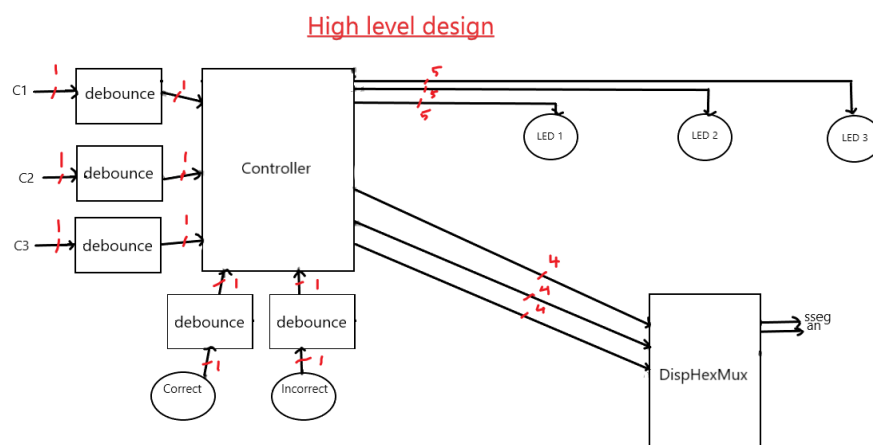
1. Problem Statement (5 pts)

Jeopardy! It is an American television game show. It was created by Merv Griffin in 1964. Students will demo a GameShowController for three contestants and each contestant has a switch of their own.

So we have a FSM with three inputs going into it, which are the three switches. Then we have 3 LEDs (LED1, LED2, and LED3); so when a contestant pushes their button, their corresponding LED lights up. Depending on whoever pushes their button first, only one LED will light up because it blocks out the other two. Once one of the contestants buzzes in, he or she will answer the question and for this we will have a Correct and Incorrect button. If the answer is correct we increment the score and if the answer is incorrect we decrement the score.

Once the game host pushes either the Correct or Incorrect button, the answering player's LED light will go off. In order to achieve this, we have to have 3 scores for each contestant (score1, score2, and score3) and we will display these scores on a 7-segment display.
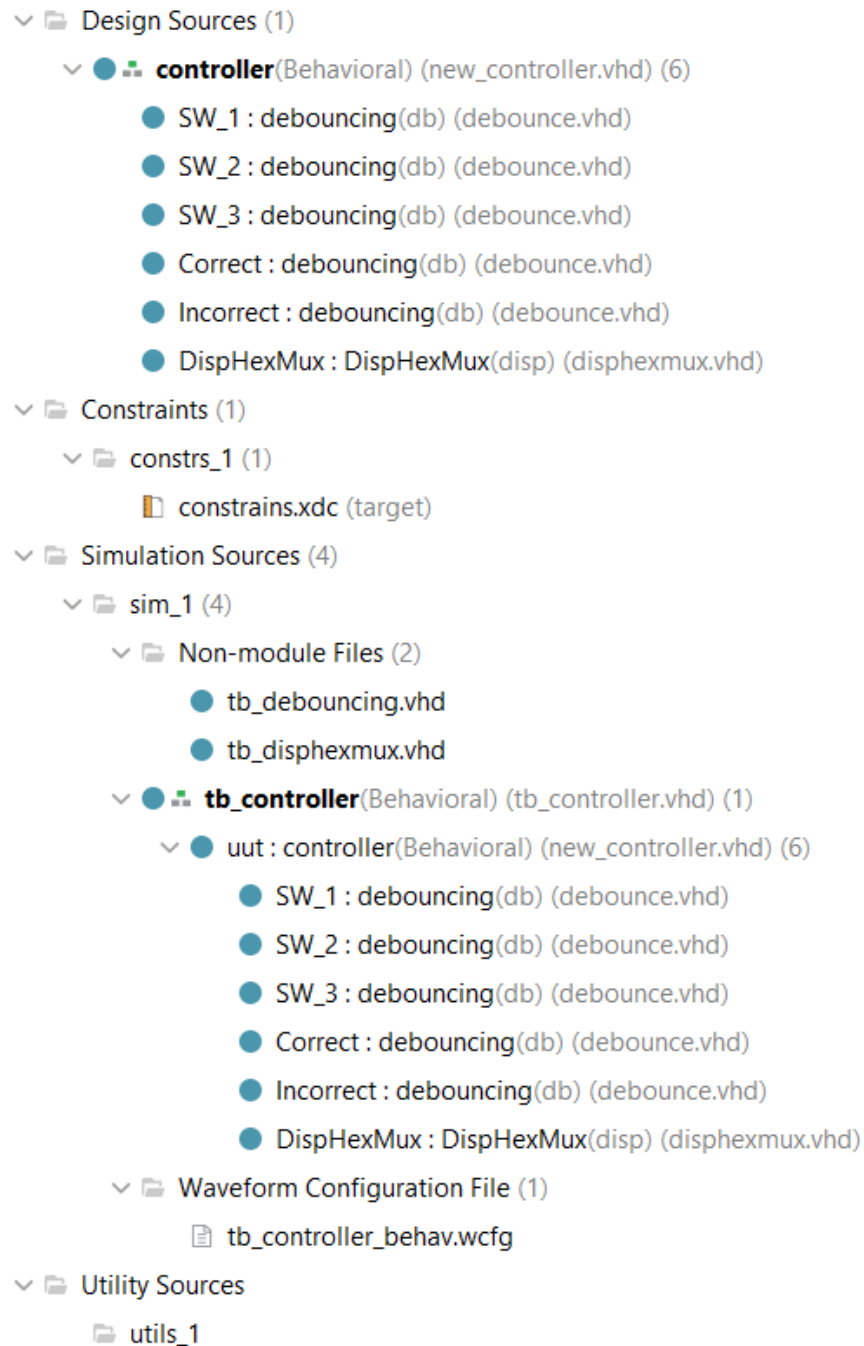
All this is given in the High Level Block Diagram shown in figure 1.

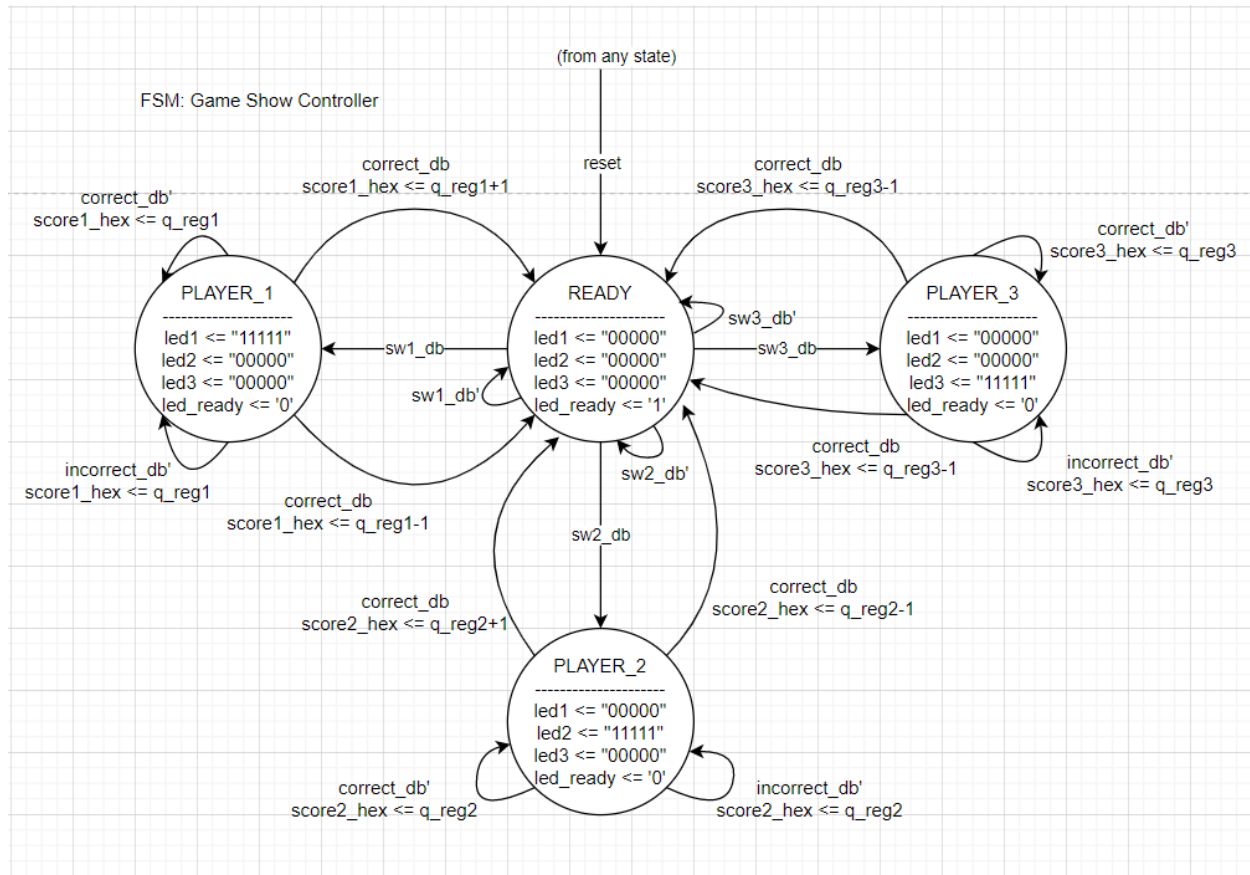2. High level block diagram (like we did in class) (5 pts)

Figure 1 *: High Level Diagram*

3. Diagram showing connected VHDL modules (5 pts)

Design Sources (1)
- controller(Behavioral) (new_controller.vhd) (6)
  - SW_1 : debouncing(db) (debounce.vhd)
  - SW_2 : debouncing(db) (debounce.vhd)
  - SW_3 : debouncing(db) (debounce.vhd)
  - Correct : debouncing(db) (debounce.vhd)
  - Incorrect : debouncing(db) (debounce.vhd)
  - DispHexMux : DispHexMux(disp) (disphexmux.vhd)

Constraints (1)
- constrs_1 (1)
  - constrains.xdc (target)

Simulation Sources (4)
- sim_1 (4)
  - Non-module Files (2)
    - tb_debouncing.vhd
    - tb_disphexmux.vhd
  - tb_controller(Behavioral) (tb_controller.vhd) (1)
    - uut : controller(Behavioral) (new_controller.vhd) (6)
      - SW_1 : debouncing(db) (debounce.vhd)
      - SW_2 : debouncing(db) (debounce.vhd)
      - SW_3 : debouncing(db) (debounce.vhd)
      - Correct : debouncing(db) (debounce.vhd)
      - Incorrect : debouncing(db) (debounce.vhd)
      - DispHexMux : DispHexMux(disp) (disphexmux.vhd)
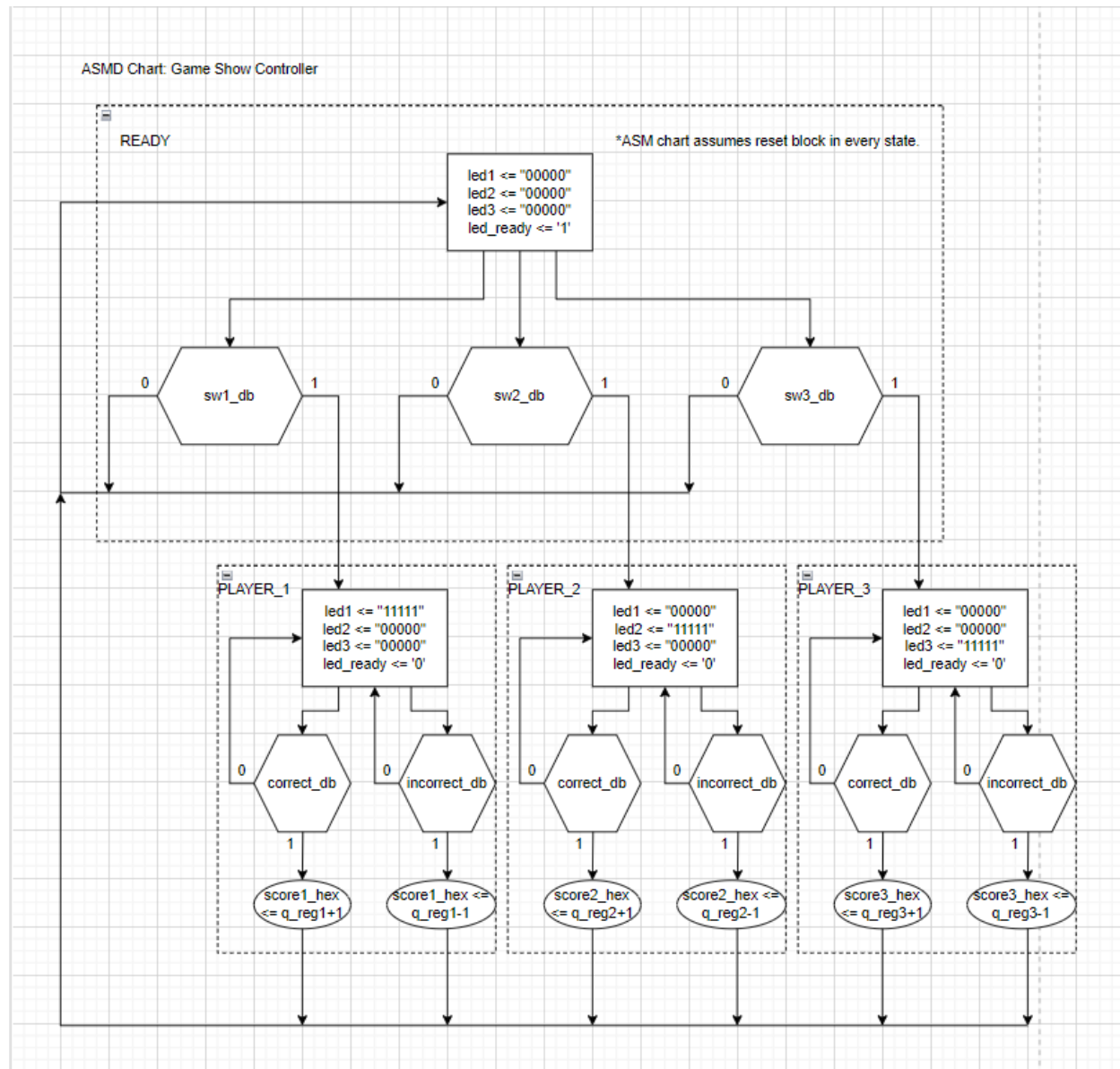 - Waveform Configuration File (1)
    - tb_controller_behav.wcfg

Utility Sources
- utils_1

Figure 2 : VHDL Modules

4. State Machine Diagram of GameController only (NOT DispHexMux, db_fsm, etc...) (5 pts)

FSM: Game Show Controller
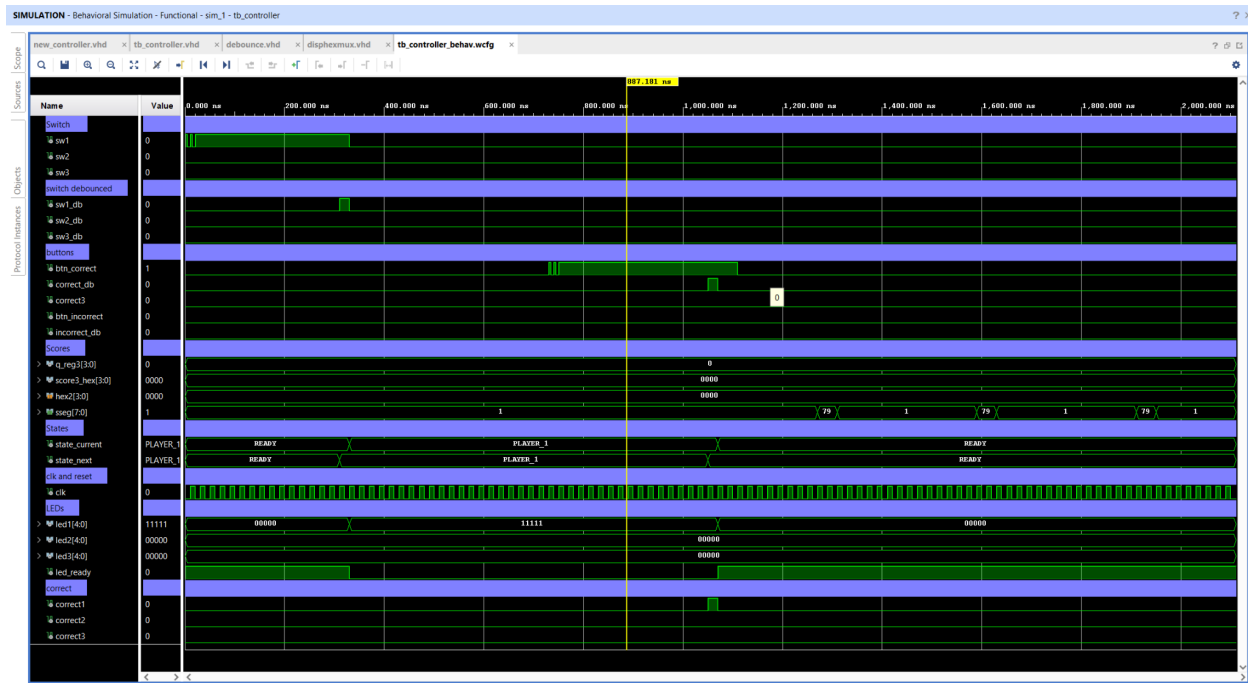
(from any state)

READY
----
led1 <= "00000"
led2 <= "00000"
led3 <= "00000"
led_ready <= '1'

PLAYER_1
----
led1 <= "11111"
led2 <= "00000"
led3 <= "00000"
led_ready <= '0'

PLAYER_2
----
led1 <= "00000"
led2 <= "11111"
led3 <= "00000"
led_ready <= '0'

PLAYER_3
----
led1 <= "00000"
led2 <= "00000"
led3 <= "11111"
led_ready <= '0'

reset

sw1_db
sw1_db'
sw2_db
sw2_db'
sw3_db
sw3_db'

correct_db
score1_hex <= q_reg1+1

correct_db'
score1_hex <= q_reg1

incorrect_db'
score1_hex <= q_reg1

correct_db
score1_hex <= q_reg1-1

correct_db
score2_hex <= q_reg2+1

correct_db
score2_hex <= q_reg2-1

correct_db'
score2_hex <= q_reg2

incorrect_db'
score2_hex <= q_reg2

correct_db
score3_hex <= q_reg3-1

correct_db'
score3_hex <= q_reg3

correct_db
score3_hex <= q_reg3-1

incorrect_db'
score3_hex <= q_reg3

Figure 3 : State Machine Diagram of GameController

5. ASM Chart of GameController only (5 pts)

ASMD Chart: Game Show Controller

READY   *ASM chart assumes reset block in every state.

led1 <= "00000"
led2 <= "00000"
led3 <= "00000"
led_ready <= '1'

sw1_db   sw2_db   sw3_db

PLAYER_1

led1 <= "11111"
led2 <= "00000"
led3 <= "00000"
led_ready <= '0'

correct_db   incorrect_db

score1_hex <= q_reg1+1   score1_hex <= q_reg1-1

PLAYER_2

led1 <= "00000"
led2 <= "11111"
led3 <= "00000"
led_ready <= '0'

correct_db   incorrect_db

score2_hex <= q_reg2+1   score2_hex <= q_reg2-1

PLAYER_3

led1 <= "00000"
led2 <= "00000"
led3 <= "11111"
led_ready <= '0'

correct_db   incorrect_db

score3_hex <= q_reg3+1   score3_hex <= q_reg3-1

Figure 5: **ASDM chart**

6. Simulations of various scenarios (5 pts)
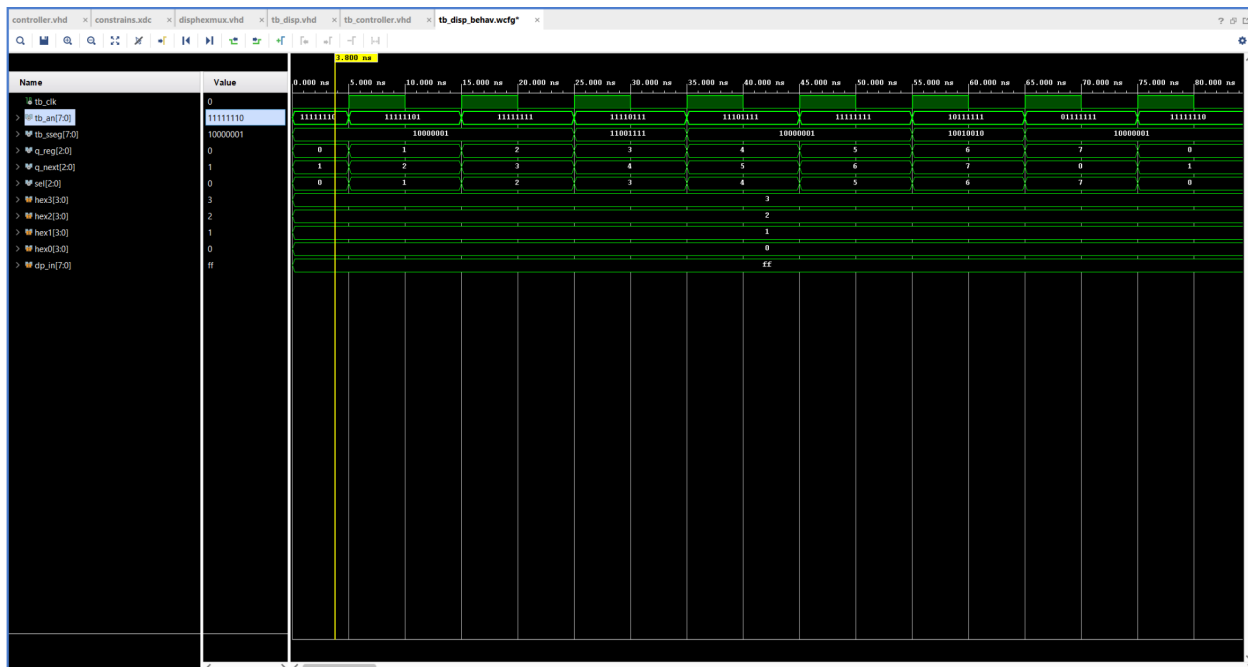

Figure 5: **SW1 on and correct is pushed**

Figure 6: **SW2 on and correct is pushed**

Figure 7: **SW3 on and correct is pushed**

Figure 8: DispHexMux Simulation

Note: The debounce simulation is seen in Figure 5, 6, and 7.
Also, the .png files of the simulations can be found in the submission for better viewing.

7. Practical Implementation on the NEXYS_7. Input switches need to be debounced.
(50 pts)

- Showed in the lab on **Thursday (12/2) at around 2 pm.**