Lab 05: Da Vinci Code

(Oct 30, 2018)

Submission deadlines:

Source Code:	18:30, Oct 30, 2018
Report:	23:59, Nov 4, 2018

Objective

- ➢ To be familiar with modeling finite state machines with Verilog.
- ➢ To be familiar with the FPGA design flow, the demo-board I/Os

Description

Da Vinci Code is a well-known mathematical game for the education. Therefore, we want you to design the Da Vinci Code game for us and implement it on the FPGA board with I/Os (switches, buttons, LEDs and seven-segment display). The game starts with guessing an untold secret number among the initial range of 00 ~ 99, and the range will become smaller and smaller as long as you make more and more guesses.

- In the game, player has 8 chances to guess the secret number.
  The number of light-up LEDs are corresponding to player's chances remain. Whenever the player makes a wrong guess, he/she will lose 1 chance and the two rightmost light-up LEDs will turn off.
  (Ex: If the player has 8 chances, all the LEDs will light up. If the player has 6 chances, all the LEDs except the rightmost four LEDs will light up.)

- After reset, the 7-segment will display "- - - -" (four dashes) and all the LEDs will not light up initially.

- When the **enter_btn** is pressed, the game is started and the LEDs will light up according to the number of chance. Generate a number among 01 and 98 randomly and update it to the number **goal**. This is the secret number to be guessed! (Ex: the **goal** is 27.)

- The game can divide into two parts as below. (SHOW_RANGE and GUESS)

- SHOW_RANGE:

  The 7-segment will display the range of the secret number **goal**.

  The leftmost two digits show the lower bound of the range and the rightmost two digits show the upper bound of the range (Both are exclusive).

  For example, The display “2391” indicates that the number to be guessed is from 24 to 90 (23 < **goal** < 91).

  Player can press **enter_btn** to make a guess. (refer to the GUESS part)

- GUESS:

  (Note: The underline “_” indicates blank digit on the 7-segment)

  The 7-segment will display “_ _ 0 1” initially. The rightmost two digits indicate the number you guess.

  You can press **digit0_btn** to increase the ones-digit by 1 and press **digit1_btn** to increase the tens-digit by 1.

  Note that when any digit has value 9 and its corresponding button is pressed, the digit will become 0 and also that when the ones-digit has value 9 and **digit0_btn** is pressed, the tens-digit will hold and need not to add 1.

  (Ex: If the display is “_ _ 9 1” and player press **digit1_btn**, the display will become “_ _ 0 1”. If the display is “_ _ 6 9” and player press **digit0_btn**, the display will become “_ _ 6 0”.)

  Once you have get the number you guess on the display, press **enter_btn** to finish your guess.

  If you make a wrong guess, the game will go back to SHOW_RANGE and the range will shrink according to the number you guessed. You will lose 1 chance as well.

  If you make a right guess, the 7-segment will display “GOAL” for a clock cycle (with frequency of clk/(2^28)) and then go back to the beginning of the game, which displays “- - - -”.

- If you lose all the chances, the 7-segment will display “LOSE” for a clock cycle (with frequency of clk/(2^28)) and then go back to the beginning of the game, which displays “- - - -”.

- There are 2 ways for players to cheat in this game:
  (1). If the **cheat_sw** is turn to 1, you will not lose chances when making wrong guesses.

  (2). In SHOW_RANGE, When the **cheat_btn** is pressed, the 7-segment will display the **goal** on the two rightmost digits and the two leftmost digits will be blank. When the **cheat_btn** is released, the display will show range as in SHOW_RANGE part.

Demo video: https://youtu.be/CBWjytUd3to

I/O signal specification:

- **rst: asynchronous** active-high reset (connected to **BTNU**). When reset is high, reset the secret number **goal** to 0, reset all LEDs to turn-off and reset player's chance to 8. After reset, the 7-segment will display "- - - -"(four dashes).
- **clk**: clock signal with the frequency of 100MHz (connected to pin **W5**)
- **enter_btn**: used for entering the game (after reset), starting to make a guess (in SHOW_RANGE) and finishing guessing (in GUESS). (connected to **BTNC**)
- **digit0_btn**: used for increasing the ones-digit by 1 when making a guess. (connected to **BTNR**)
- **digit1_btn**: used for increasing the tens-digit by 1 when making a guess. (connected to **BTNL**)
- **cheat_btn:** In SHOW_RANGE, when the **cheat_btn** is pressed, the 7-segment will display the **goal** on the two rightmost digit and the leftmost digit will be blank. When the **cheat_btn** is released, the display will show range as in SHOW_RANGE part (connected to **BTND**).
- **cheat_sw:** If the cheat_sw is turn to 1, player will not lose chances when making wrong guesses (connected to **SW0**)
- **DISPLAY[6:0]**: signals to show the digits on the 7-segment displays.
- **DIGIT[3:0]**: signals to enable one of the 7-segment displays.
- **LED[15:0]**: indicating how many chances player has.

Note:

1. All the signals from button (except rst) should be properly processed (debounce, one pulse).
2. You may use clk/2^16 for your debounce and onepulse and use clk/2^13 for your 7-segment display.

Hint:

1. You may have at least four or five states in your FSM design.
2. You can generate the random number using a counter. Use the **enter_btn** to sample a number from the counter. Make sure it is within the range of 01~98. (Simply adjust it if not.)

You can use the following template for your design.

```
module lab05(input wire rst,
             input wire clk,
             input wire enter_btn,
             input wire digit0_btn,
             input wire digit1_btn,
             input wire cheat_btn,
             input wire cheat_sw,
             output [6:0] DISPLAY,
             output [3:0] DIGIT,
             output [15:0] LED);

  //add your design here
  //…
endmodule
```

Attention:

1. You should hand in only lab05.v. If you have several modules for your design, put them in lab05.v together.
2. You should also hand in your report as lab05_report_StudentID.pdf (i.e., lab05_report_105080001.pdf).
3. You should be able to answer questions of this lab from TA during the demo.
4. You need to generate bitstream before demo.