CSE100 Lab 4 – Stop It

Contents

1 Submission

- Due: Tuesday 11/12/2024 at the end of your lab section.
- Submit your Lab Write-Up to Gradescope.
- Submit your code to the Gradescope autograder. (You have unlimited submission attempts).
- Demonstrate your implementation to a TA.

2 Goals

You will implement a state machine as part of a sequential circuit for a game called *Stop It*.

Once you finish your implementation, you will program it to a Basys 3 FPGA Board.

3 Prelab

Read the following sections before starting your lab.

3.1 Overview

Watch this video to understand what to do for this lab: https://youtu.be/GrlDUsAk_Ig.

The video is summarized here:

- At the start of each round the Game Counter is displayed on the rightmost two digits of the BASYS3 7-segment display. The left two digits are off.
- A `go_i` signal is given (pushbutton `btnC` is pressed) to start the next round.
- In each round, a random 5-bit binary value, the target number, is selected and displayed on the two leftmost digits and the 5-bit Game Counter (still displayed on the rightmost digits) is set to `1f`.
- After 2 seconds the Game Counter begins to decrement every quarter second.
- The Game Counter will keep decrementing, rolling under to `1f` (31 decimal) after reaching 0.
- When the `stop_i` signal is given (pushbutton `btnU`) the Game Counter stops decrementing.
- At this point, if the value of the Game Counter matches the target number, then all 4 display digits flash for four seconds in unison.
- If the value of the Game Counter does not match the target number, then all 4 display digits flash for four seconds, with the target number and Game Counter digits alternating as they flash.
- The flashing continues for four seconds and then the leftmost digits are again blank and a new round can begin with a `go_i` signal.
- Each time the player succeeds in matching the target, one more LED lights up beginning with the rightmost.
- If all 16 LEDs are lit, and the target number was matched, then the game has been won. After the digits flashes for 4 seconds, all 16 LEDs flash, and no button, except `btnR` will have an effect.
- To make the game easier to win (without 17 matches), pressing `btnL` will be a cheat switch that will load the switches into the LEDs (or actually the shift register holding the values of the LEDs).

On the BASYS3 board,

- PushButton btnC drives the go_i signal.
- PushButton btnU drives the stop_i signal.
- PushButton btnL will load the switches into the LEDs.
- PushButton btnR drives the synchronous active-low global reset rst_ni.
- The BASYS3 clock clk_100 is an input of your top level, but will not be part of your logic. It is connected only to the clk_100M_to_clk_1k and clk_100M_to_clk_4 clock divider modules, which divides the 100 MHz clock into 1 kHz and 4 Hz respectively.

3.2 Random Number Generator

You shall finish the random number generator in "rtl/lfsr.sv".

You will use a Linear Feedback Shift Register (LFSR) to generate a random 8-bit binary number. Below is an 8-bit linear feedback shift register. This LFSR is simply an 8-bit shift register where the input to the first register is the XOR of specific bits in the register. If all the bits in the registers are 0 then the LFSR output will always be 0's. Otherwise, it will go through a sequence of all 255 non-zero states before it repeats. This sequence is not random, but reading the LFSR at random times (assuming it cycles through enough states fast enough) will give you a pseudorandom 8-bit number. The choice of inputs into the XOR gate is not arbitrary so be sure to use the inputs exactly specified in the figure.

You should only use 5 bits from the LFSR as the target number, but you still need the 8-bit linear feedback shift register.

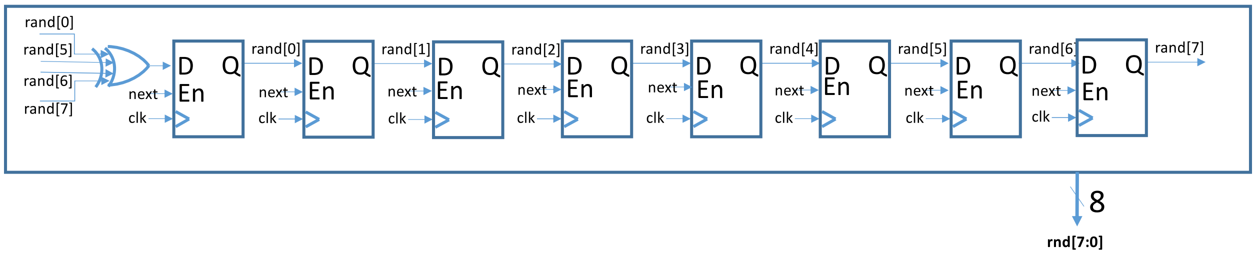

Figure 1: Linear Feedback Shift Register

3.3 Game Counter

You shall finish the Game Counter in "rtl/game_counter.sv".

The Game Counter is the counter that decrements on the 7 segment display. When the en_i is active, Game Counter should start decrementing. When the rst_ni is active, the Game Counter should be initialized to 1f.

3.4 Time Counter

You shall finish the Time Counter in "rtl/time_counter.sv".

The Time Counter is the counter that keeps track of how many quarter-seconds have past. When the en_i is active, Time Counter should start incrementing. When the rst_ni is active, the Time Counter should be initialized to 0.

3.5 7 Segment Display

You shall finish the 7 Segment Display driver in `"rtl/basys3/basys3_7seg_driver.sv"`. Feel free to use your `hex7seg` module from an earlier lab.

You will need to display the target number as well as the Game Counter and make them flash appropriately depending on the game state. You can control whether a digit of the display is on or off by providing appropriate logic for its `an` port. This is the only module that uses the 1 kHz clock.

Refer to previous labs and the Basys3 Reference Manual if you have questions: https://digilent.com/reference/programmable-logic/basys-3/reference-manual#seven_segment_display.

3.6 LED Display

You shall finish the LED display in `"rtl/led_shifter.sv"`.

The LEDs can be controlled using a 16-bit shift register with the `shift_i` signal. Each time the target is matched, a 1 will be shifted in. There is also a parallel load input (`ld_i`) to load a 16-bit bus. The `off_i` signal, which forces the `LED_Shifter` output low, is useful for making the LEDs flash.

3.7 Stop It State Machine

You shall finish the state machine in `"rtl/stop_it.sv"`.

Read the State Machine section in the lowRISC SystemVerilog style guide: https://github.com/lowRISC/style-guides/blob/master/VerilogCodingStyle.md#finite-state-machines.

You may either implement the state machine using the lowRISC style, with a large `unique case` statement inside of an `always_comb` block; or you may simply use several `assign = ? :` statements.

Advice: Getting a state machine right usually requires several iterations. It is likely that in demonstrating your design the TA will discover some case that is not properly handled. Often changes to the state machine are not simple. A complete redesign may be necessary. Trying to patch it by changing one signal here or there, or inserting a gate/FF, or more, almost always makes things worse. Please leave yourself enough time. Hurrying will introduce more bugs that you will need to hunt down. It is strongly suggested that your entire design be entered and simulating **before** your third section.

4 Lab

You will need to complete and submit the following files:

- `"synth/basys3/Basys3_Master.xdc"`
- `"rtl/game_counter.sv"`
- `"rtl/led_shifter.sv"`
- `"rtl/lfsr.sv"`
- `"rtl/stop_it.sv"`
- `"rtl/time_counter.sv"`
- `"rtl/basys3/basys3_7seg_driver.sv"`
- `"rtl/basys3/hex7seg.sv"`

1. Complete the auxiliary modules and ensure they pass the tests mentioned in the README.
2. Draw a state diagram for *Stop It* using all the states mentioned in `stop_it_pkg::state_t` in `"rtl/stop_it_pkg.sv"`. Be sure to denote on each arrow what condition is required to enter the next state. There is no need to assign an encoding in your state diagram; simply use the `stop_it_pkg::state_t` enum labels.
3. Implement the state machine you drew. You may either implement the state machine using the lowRISC style, with a large `unique case` statement inside of an `always_comb` block; or you may simply use `assign = ? :`.
4. The Basys3 constraints file `"synth/basys3/Basys3_Master.xdc"` was downloaded from Digilent's GitHub: https://github.com/Digilent/Basys3/blob/master/Resources/XDC/Basys3_Master.xdc. Complete it to match your `basys3` module. Refer to previous labs and the Basys3 Reference Manual if you have questions: https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual.
5. Simulate your design using commands specified in the `"README.md"`. Submit to the autograder until you get 100%. Note that passing the autograder will not guarantee that your implementation is correct.
6. Synthesize your design and program it to the Basys3 using commands specified in the `"README.md"`. Your design may not work the first time. Review the waveforms for any issues that the autograder did not catch.
7. Get checked off by a TA.
8. Finish the writeup.

Important Please remember to turn off the power to BASYS3 board when you are done.

5 Write-Up

1. Include a state bubble diagram. The diagram can be hand drawn or drawn digitally (https://draw.io/).
2. Describe each state of your machine in words.

Example: "The machine is in state `DECREMENTING` while the Game Counter is counting down. It stays in this state until the `stop_i` signal (`btnU`) goes high."

3. Provide screenshots of your waveform. Display buses as buses in hex.

The following input and output signals should be included:

- `stop_it.*_i`
- `stop_it.*_o`
- `stop_it.lfsr.rand_o`
- `stop_it.game_counter.count_o`
- `stop_it.time_counter.count_o`

Highlight the following moments:

a. Player pressing `go_i` then the next LFSR output generating.
b. Player pressing `stop_i` and matching the target.

c. Player pressing `stop_i` and not matching the target.
d. Player winning after 17 correct stops.