CS223 Fall 2014
Some Sample Projects

(These are some project suggestions from the book "FPGA PROTOTYPING BY VERILOG EXAMPLES" by Chu P.P 2008)

These projects are given as examples to guide you in selecting a project . You do not have to do these projects.

1-Mouse-controlled seven-segment LED display

We can use the mouse to enter four decimal digits on the four-digit seven-segment LED display. The circuit functions as follows:
Only one of the four decimal points of the LED display is lit. The lit decimal point indicates the location of the selected digit.
The location of the selected digit follows the x-axis movement of the mouse.
The content of the select seven-segment LED display is a decimal digit (i.e., 0, . . . ,9) and changes with the y-axis movement of the mouse.
Design and synthesize this circuit and verify its operation.

2-part (a) ROM-based *sin(x)* function

One way to implement a sinusoidal function, *sin(x),* is to use a look-up table. Assume that the desired implementation requires 10-bit input resolution [i.e., there are 1024 (2") points between the input range of 0 and 2.1 and 8-bit output resolution [i.e., there are 256 *(28)*p oints between the output range of -1 and +I]. Let the input and output be the 10-bit .c signal and the 8-bit y signal. The relationship between *x* and y is

$$\frac{y}{2^7} = \sin\left(2\pi \frac{x}{2^{10}}\right)$$

Because of the symmetry of the *sin* function, we only need to construct a 28-by-7 table for the first quadrant (i.e., between 0 and π/2) and use simple pre- and postprocessing circuits to obtain the values in other quadrants. Design this circuit as follows:
1. Write a program in a conventional programming language to generate a case statement that incorporates the 28-by-7 look-up table for the first quadrant.
2. Follow the ROM template in Listing 12.6 to derive the HDL code for the look-up table.
3. Derive a testbench to generate the sinusoidal output for three complete periods. This can be done by using a 10-bit counter to generate the 10-bit ROM address for $3 * 2^{10}$ clock cycles.

2-part(b) ROM-based *sin(x)* and *cos(x)* functions

In many communication modulation schemes, the *sin(x)* and *cos(x)* functions are needed at the same time. The new circuit has two outputs ys and yc:

$$ys = \sin\left(2\pi \frac{x}{2^{10}}\right)$$
$$yc = \cos\left(2\pi \frac{x}{2^{10}}\right)$$

Although we can follow the previous procedure and create a new ROM for the *cos(x)* function, a better alternative is to share the same ROM for both *sin(x)* and *cos(x)* functions. This is based on the observations that *cos(x)* is only a phase shift of *sin(x)* and that the FPGA's block RAM can provide dual-port access.
Note that this circuit requires essentially a "dual-port ROM." No HDL behaviorial template is given for this type of memory. It may be necessary to use Core Generator program to achieve this goal.

Projects using VGA Monitor

3- (a)Ball-in-a-box circuit

The ball-in-a-box circuit displays a bouncing ball inside a square box. The square box is centered on the screen and its size is 256-by-256 pixels. The ball is an 8-by-8 round ball. When the ball hits the wall, the ball bounces back and the wall flashes (i.e., changes color briefly). The ball can travel at four different speeds, which are selected by two slide switches, and its direction changes randomly when a pushbutton switch is pressed. Derive the HDL code and then synthesize and verify operation of the circuit.

3- (b)Two-balls-in-a-box circuit

We can expand the circuit in Ball in a box to include two balls inside the box. When two balls collide, the new directions of the two balls should follow the laws of physics. Derive the HDL code and then synthesize and verify operation of the circuit.

4- Two-player pong game

The two-player pong game replaces the left wall with another paddle, which is controlled by the second player. To better accommodate two players, we can use the keyboard interface as the input device. Four keys can be defined to control vertical movements of the two paddles. Derive the HDL code and then synthesize and verify operation of the circuit.

5- Breakout game

The breakout game is somewhat like the pong game. In this game, the left wall is replaced by several layers of "bricks." When the ball hits a brick, the ball bounces back and the brick disappears. The basic screen is shown in Figure 13.11. As in the code of Listing 13.5, we assume that the game runs continuously. Derive the HDL code and then synthesize and verify operation of the circuit.

6-Complete Breakout Game

The free-running breakout game is described in 5. Follow the procedure of the pong game in 4 to derive the complete system. This should include the design of a new text display subsystem and the design of a top-level FSM controller. Derive the HDL description and then synthesize and verify operation of the circuit.

7-Simple four-trace logic analyzer

A logic analyzer displays the waveforms of a collection of digital signals. We want to design a simple logic analyzer that captures the waveforms of four input signals in "freerunning"
mode. Instead of using a trigger pattern, data capture is initiated with activation of

a pushbutton switch. For simplicity, we assume that the frequencies of the input waveform are between 10 kHz and 100 kHz. The circuit can be designed as follows:
1. Use a sampling tick to sample the four input signals. Make sure to select a proper rate so that the desired input frequency range can be displayed properly on the screen.
2. For a point in the sampled signal, its value can be encoded as a tile pattern by including the value of the previous point. For example, if the sampled sequence of one signal is "00001 1 1 1000", the tile patterns become "00 00 00 01 1 l 11 1 l 10 00 OO", as shown in Figure 14.7(b).
3. Follow the procedure of the preceding square-wave experiment to design the tile memory and video interface to display the four waveforms being stored .
4. Derive the HDL description and then synthesize the circuit.
To verify operation of the circuit, we can connect four external signals via headers around the prototyping board. Alternatively, we can create a top-level test module that includes a 4-bit counter (say, a mod-10 counter around 50 kHz) and the logic analyzer, resynthesize the circuit, and verify its operation

Some samples from YOU Tube where you can find Verilog tutorials and projects.

FPGA Snake game http://www.youtube.com/watch?v=niQmNYPiPw0&noredirect=1

http://www.youtube.com/watch?v=Mf6buDkbyAQ

FPGA game console   http://www.youtube.com/watch?v=KmNaEnPSM9o&noredirect=1

Simple Games with 8x8 LED matrix
http://www.youtube.com/watch?v=Mf6buDkbyAQ&noredirect=1