ECE275 Project progress report (Due Nov 30th)

Instructor: Vikas Dhiman

November 22, 2022

1 VGA module

Watch the tutorial below and replicate it on the Altera FPGA board,

- VGA Video Tutorial (Must be logged in to you Umaine account to view) : Video Tutorial. There is a mistake in the video when instantiating the make box module, it should be make_box make_first_player_paddle(and not module make_first_player make_box(
- Example simple top level : VGA_top.v
- DE0 VGA Driver Module : DE0_VGA.v
- PLL (Phase Locked Loop) Verilog File : PLL_PIXEL_CLK.v
- QSF File : VGA_top.qsf

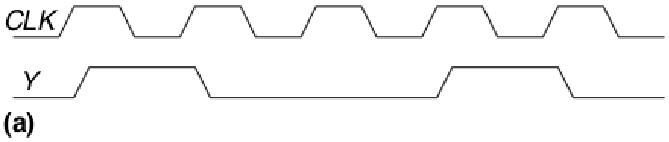
Required submissions for project progress report on Nov 30 before class.

1. A working version of VGA_top.v and VGA_top.qsf that can draw a box on the screen.

2 Breaking application into smaller parts

2.1 Slow clock

Problem 1. *A divide-by-N counter has one output and no inputs. The output Y is HIGH for one clock cycle out of every N. In other words, the output divides the frequency of the clock by N. The waveform for a divide-by-3 counter is shown here:*



Sketch circuit designs for such a counter

Problem 2. *Repeat the above problem in Verilog and show a clock being reduced from 50 MHz to 10 Hz in ModelSim simulator. Example files covered in class, testslowclock.v slowclock.v.*

Required submissions for project progress report on Nov 30 before class.

1. Modified testslowclock.sv to generate a fastclock of frequency 50MHz.
2. Modified slowclock.sv, that outputs a slowclock of frequency 10Hz.
3. Screen shot of the waveform that you generated using ModelSim.

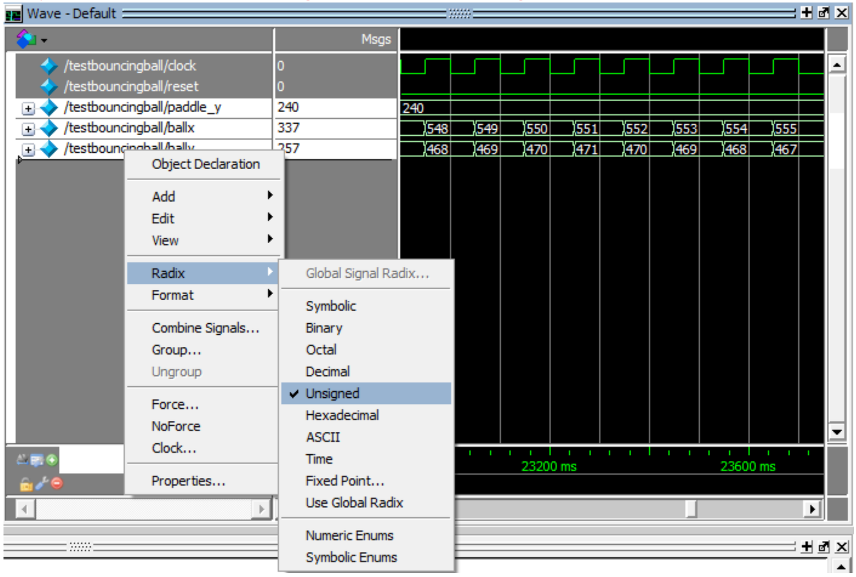
2.2 Bouncing Ball

Problem 3. *Design a circuit for a 1 pixel bouncing ball on a 4x4 pixel screen. You cannot design this circuit by hand (Why?). How will you design it using Verilog? Write verilog code and test it using a testbench?*

Problem 4. *Once you have the VGA example working, extend the above problem to 640x480 resolution and ball represented by a box your chosen size. Connect this module's output to the VGA module.*

Required submissions for project progress report on Nov 30 before class.

1. A bouncingball.sv that bounces the ball around a screen of resolution 640x480. The output of the module must be the 10-bit XY coordinates of the ball, 10-bit ballx and 10-bit bally.
2. A testbouncingball.sv that runs at 10Hz and shows the ballx moving between 0 and 640 and bally moving between 0 and 480. Note that you can right click on wave-variable and select radix to be Unsigned decimal. You may use the code from Figure 1 as a starting point.



Wave - Default

	Msgs	
/testbouncingball/clock	0	
/testbouncingball/reset	0	
/testbouncingball/paddle_y	240	240
/testbouncingball/ballx	337	548 549 550 551 552 553 554 555
/testbouncingball/bally	257	468 469 470 471 470 469 468 467

Radix: ✓ Unsigned

```
`timescale 10ms/1ms
module testbouncingball();
  reg clock;
  reg reset;
  reg [9:0] paddle_y;
  initial begin
    clock = 0;
    paddle_y = 10'd240;
    #5 reset = 1;
    #15 reset = 0;
  end
  always #5 clock = ~clock;

  wire [9:0] ballx;
  wire [9:0] bally;

  bouncingball bball( clock, reset, paddle_y, ballx, bally);
endmodule
```

Figure 1: Sample testbouncingball.sv

3. Screenshot of the waveform generated using ModelSim.

2.3 Moving paddle

Problem 5. *Design a circuit for 1 pixel paddle on a 4x4 pixel screen. Assume that it can take two inputs from BUTTONs, one for moving up and another one for moving down. You can design this circuit by hand (Why)?*

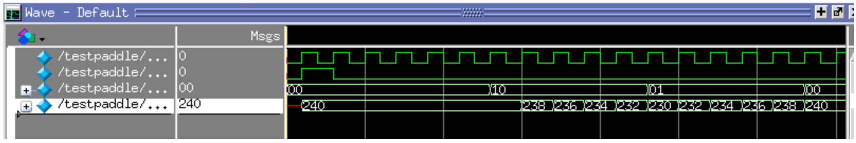
Problem 6. *Design the above circuit using Verilog.*

Problem 7. *Once you have the VGA example working, extend the above problem to 640x480 resolution with you*

Required submissions for project progress report on Nov 30 before class.

1. A paddle.sv that takes 2 bit input up_down for up button and down button and outputs the Y-coordinate of the paddle which moves according to the button pressed.

2. A testpaddle.sv that runs at 50Hz and shows the paddle_y to increase and decreases witch change in up and down input. You may use the code from Figure 2 as a starting point.



```
/testpaddle/...  0
/testpaddle/...  0
/testpaddle/...  00
/testpaddle/...  240
```

.

```
`timescale 10ms/1ms
module testpaddle();
   reg clock;
   reg reset;
   reg [1:0] up_down; // up down buttons
   wire [9:0] paddle_y; // y coordinate of paddle y
   initial begin
      clock = 0;
      reset = 0;
      up_down = 2'b00;
      #1 reset = 1;
      #2 reset = 0;
      #10 up_down = 2'b10; // after delay of 100ms press up
      #10 up_down = 2'b01; // after delay of 100ms press down
      #10 up_down = 2'b00; // after delay of 100ms release both
   end
   always #1 clock = ~clock; // 20ms clock time period

   paddle pdl( clock, reset, up_down, paddle_y);
endmodule
```

Figure 2: Sample testpaddle.sv

3. Screenshot of the waveform generated using ModelSim.

2.4 Ball paddle collision

Problem 8. *Combine the bouncing ball problem with the moving paddle problem and bounce the ball only if the ball is about to hit the paddle, otherwise game is over. If the ball hits the paddle increment a score counter by 1.*