

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory (Spring 2006)

Laboratory 4 Check Off Sheet

Student Name:

6.111 Staff Member Signature/Date:

Part A: VGA Interface

You must show a TA the following for check off:

- State transition diagram of your VGA interface
- Verilog code for the VGA interface
- A “screenshot” of a Pong game: a border, ball, paddle, and MIT logo

Part B: Pong

- State transition diagrams for your major-minor FSMs
- Design methodology, design partitioning, and testing
- Your Pong game working correctly

Be able to respond to any of the following questions:

- What is the advantage of the major-minor FSM setup?
- Explain briefly how VGA works

Laboratory 4 - MIT Pong

Issued: March 17, 2006

Part A Checkoff: recommended by March 24, 2006

Part A and B Checkoff Due: April 7, 2006

Report Due: April 10, 2006 (1PM in 38-107)

1. Introduction

The purpose of this lab is to become familiar with VGA, the display system of a typical PC. You will use a major-minor FSM setup to build and debug a version of the classic video game, Pong (Figure 1). As with previous labs, you will use FSMs to control the system. By the end of the laboratory, you will be familiar with how VGA works, will be able to implement digital systems involving VGA, and will understand how to construct complex systems. We will use the Major/Minor FSM concept described in lecture.

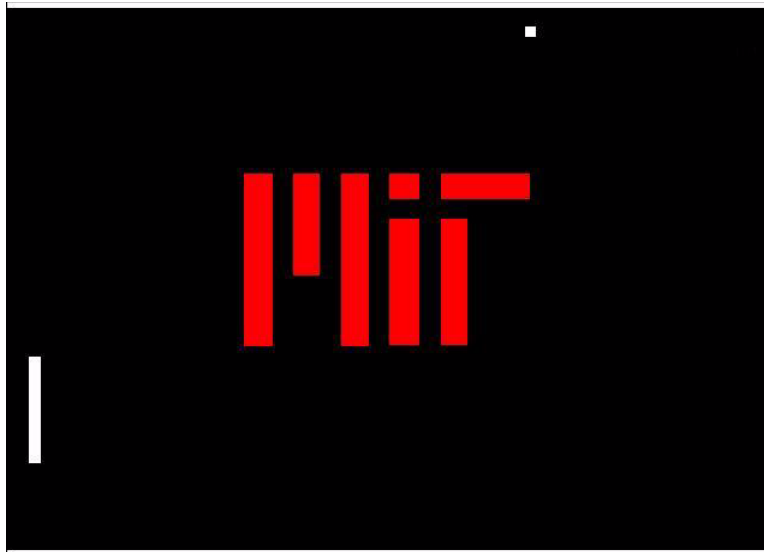


Figure 1. Screenshot of Pong

2. Procedure

The laboratory consists of three phases. The first phase is the design phase. You should read through the lab and plan your design. It will be helpful to review your design with a member of

the teaching staff. In particular, it would be helpful to read the documentation of how VGA works at the website: <http://www-mtl.mit.edu/Courses/6.111/labkit/vga.shtml>

The second phase is to interface with the VGA. Your goal is to create a “screenshot” of a Pong game, which consists of the border, the paddle, the ball, and the MIT logo in the background.

The third phase will build on top of that, where you will actually implement the game. You are asked to build the logic so the paddle and ball move in a similar fashion to those in Pong. Your design should be structured so that a top-level FSM controls other minor FSMs which control the rest of your system.

You will be required to turn in a detailed report of this laboratory.

3. Task Description

Your first goal, Part A, is to create a still image of a Pong game in progress. You will learn how the VGA works and how to interface to VGA (you may use problem set 3, problem 4 to code the VGA interface). You’ll want to try to make the foundation you make in Part A is easily extendable to Part B. Ideally, once you finish Part A, all the details of the VGA interface will be abstracted away so you can supply information for your Part B logic with little concern about the display. One way to do this is to have a block that draws all of the components, and you input locations of the ball and paddle to that block. The only difference between Part A and Part B will be whether you input a fixed location (still-image) or a dynamic location from Part B (game).

Your ultimate goal is to create a one-player version of Pong. The user will be able to control a paddle on the left side of the screen, and use the paddle to hit and direct a ball to the wall on the right side of the screen. The ball should be able to bounce freely off of the top and bottom borders, and should bounce off the wall back toward the left side, so the user may hit the ball again. Should the user miss the ball with the paddle, the ball will continue and freeze once it reaches the very edge of the left side, signaling the game is over.

Your overall system should have several user inputs:

- 1) A **reset** button, which the user presses to restart the game, either in the middle of a game or when the game is over.
- 2) **up** and **down** buttons which the user hits to control the paddle.
- 3) A 4-bit **speed** switch which controls the initial speed of the ball during gameplay. The speed sets the initial velocity of the ball in units of pixels per frame (both in the x and y directions)

4. System Organization

A logical block diagram is shown in Figure 2. As mentioned earlier, you are to use a major-minor FSM setup, though how to partition it is up to you. Unlike previous labs, we have intentionally left some details undefined to give you flexibility in your design. There are certain tradeoffs such as complexity, modularity, and system performance. Be ready to justify your partitioning in the

lab report. Also note the user interface: a 4-bit switch to control the initial speed of the ball and the necessary buttons to allow the user to move paddles.

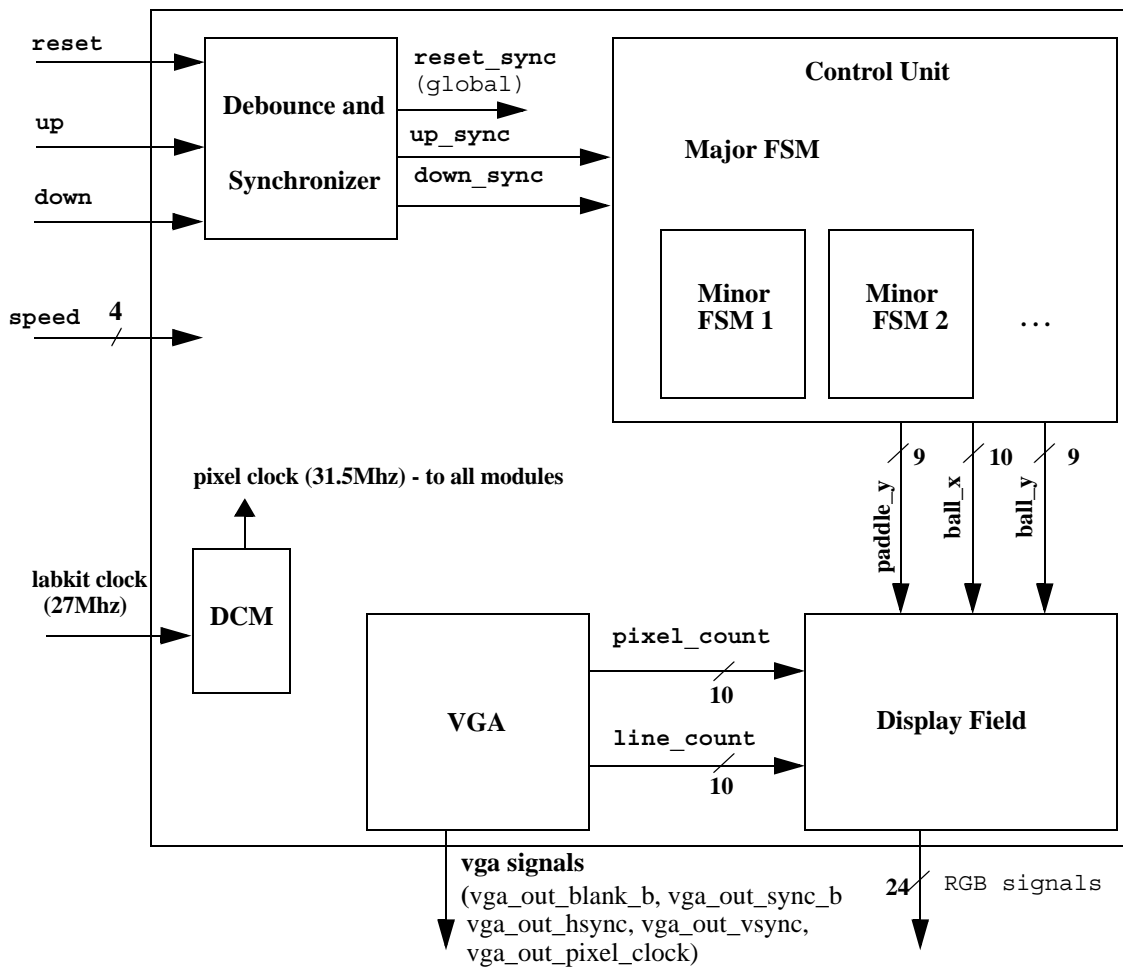


Figure 2. Block Diagram.

5. VGA Output and Checkoff

Part A of the lab is properly displaying to the VGA. Your objective is to produce a “screenshot” of the field. The field is 640x480, the ball is an 8x8 pixel square, and the paddle is a 8x64 rectangle. The MIT logo should be red, the background black, and the paddle, ball, and border should be white. The full pixel layout is shown in Figure 5.

(For the MIT logo, you may use the MIT colors:

MIT Red - R: 8'b0101_1111, G: 8'b0001_1111, B: 8'b0001_1111 and

MIT Gray - R: 8'b0100_1111, G: 8'b0100_1111, B: 8'b0011_1111)

The values of vga_out_red, vga_out_green, vga_out_blue (each 8 bit values) determines the color of the pixel. Notice that for this video DAC, the number of unique colors is $2^8 * 2^8 * 2^8 = 2^{24}$ or 16 million colors. This is often referred to as 24 bit or true color. White is composed of pixels with all ones. Black is all zeros.

VGA Block

The VGA Block is responsible for correctly sending various signals to the VGA display, such as the horizontal and vertical blanking signals. You probably will want to implement this using a finite state machine. Additionally, you might want the block to also keep track of and produce pixel counts (horizontal position) and line counts (vertical position), so you will be able to know which pixel you are dealing with on the display at a particular time. Note that on the labkit page (<http://www-mtl.mit.edu/Courses/6.111/labkit/vga.shtml>), there are several possible timing schemes for the VGA. You should use the third 640x480 setup with the 31.5 MHz clock. The respective timings are shown in the following figures.

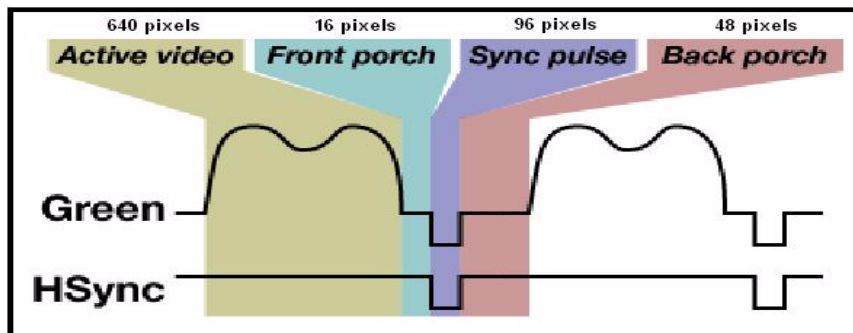


Figure 3. Horizontal timing

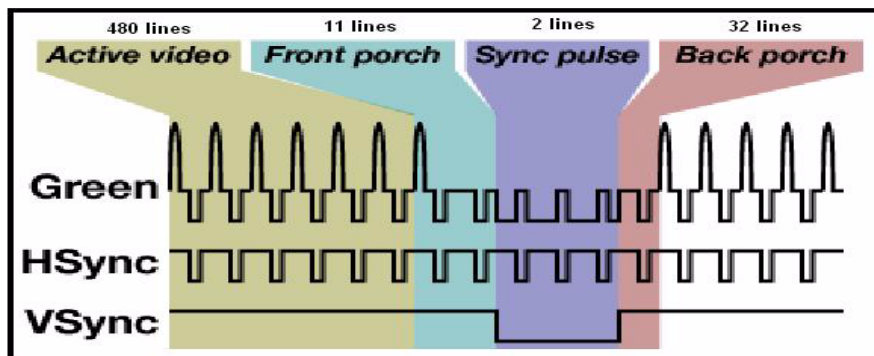


Figure 4. Vertical Timing

DCM

When constructing your VGA interface, you will probably need to use the DCM (Digital Clock Manager) to create the appropriate pixel clock. You can create a 31.5MHz clock signal by using the following code in your top level labkit.v file:

```
DCM pixel_clock_dcm(.CLKIN(clock_27mhz), .CLKFX(pixel_clock));  
//synthesis attribute CLKFX_DIVIDE of pixel_clock_dm is 6  
//synthesis attribute CLKFX_MULTIPLY of pixel_clock_dcm is 7  
//synthesis attribute CLK_FEEDBACK of pixel_clock_dcm is "NONE"  
//synthesis attribute CLKIN_PERIOD of pixel_clock_dcm is 37
```

The 31.5MHz pixel_clock should be used to clock all the modules in your design not just the VGA interface. A template file, lab4_labkit.v, is provide which include the DCM. See course website for details.

Display Field Block

The Display Field block is responsible for actually producing the RGB values for the VGA display. The block should take in the pixel count and line count values from the VGA block, so you may determine which RGB pixel value is currently being generated. For easier adaptation to Part B, it may also be helpful to have the Display Field block take in position values for the paddle and ball, and have the logical block draw around those points. The Display Field block can also be used to draw the MIT logo.

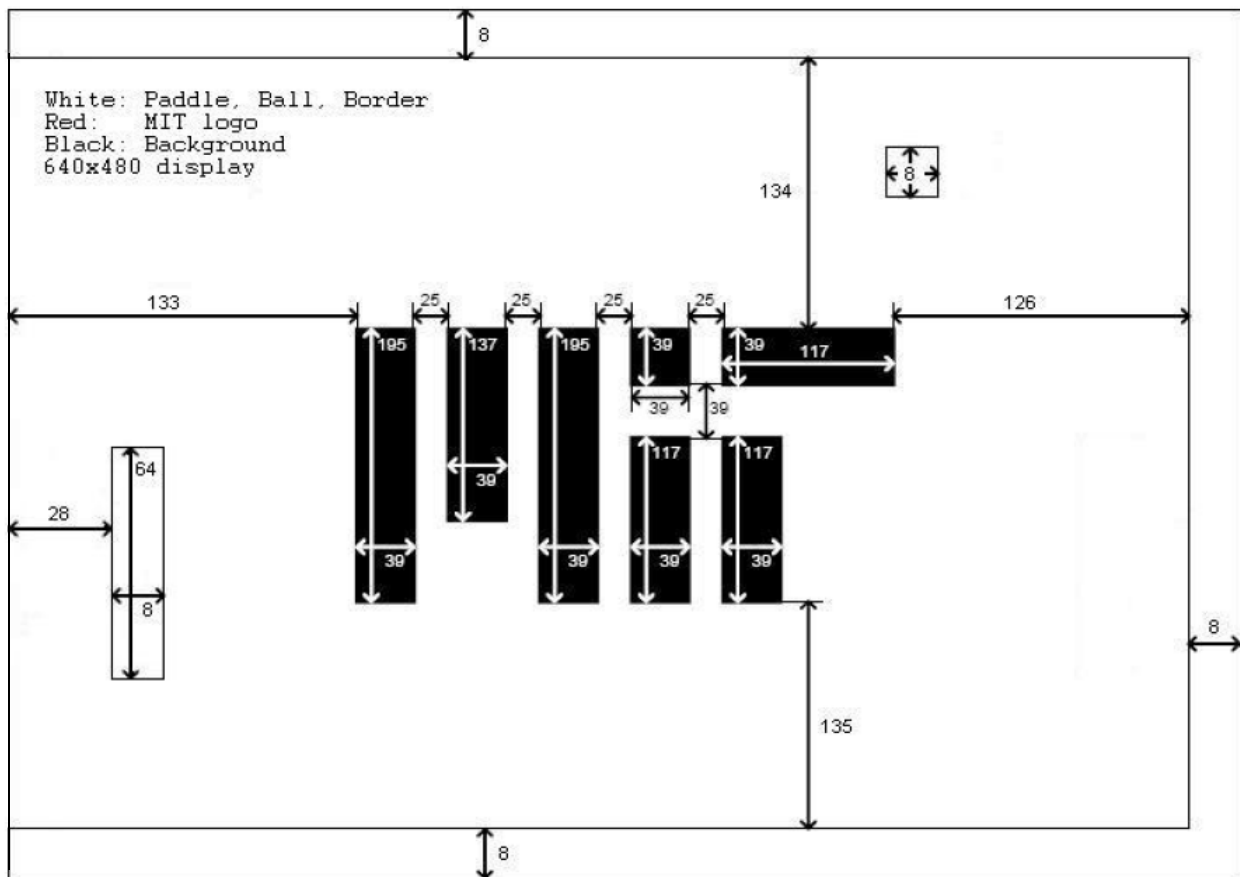


Figure 5. Pixel Specifications (not to scale)

Figure 6 shows the specification for the video timing for the resolution and refresh rate for this lab. For future reference, you might choose to keep the various parameters (front porch, etc.) programmable to adapt your design to different screen resolutions. However, this is not required for the lab checkoff.

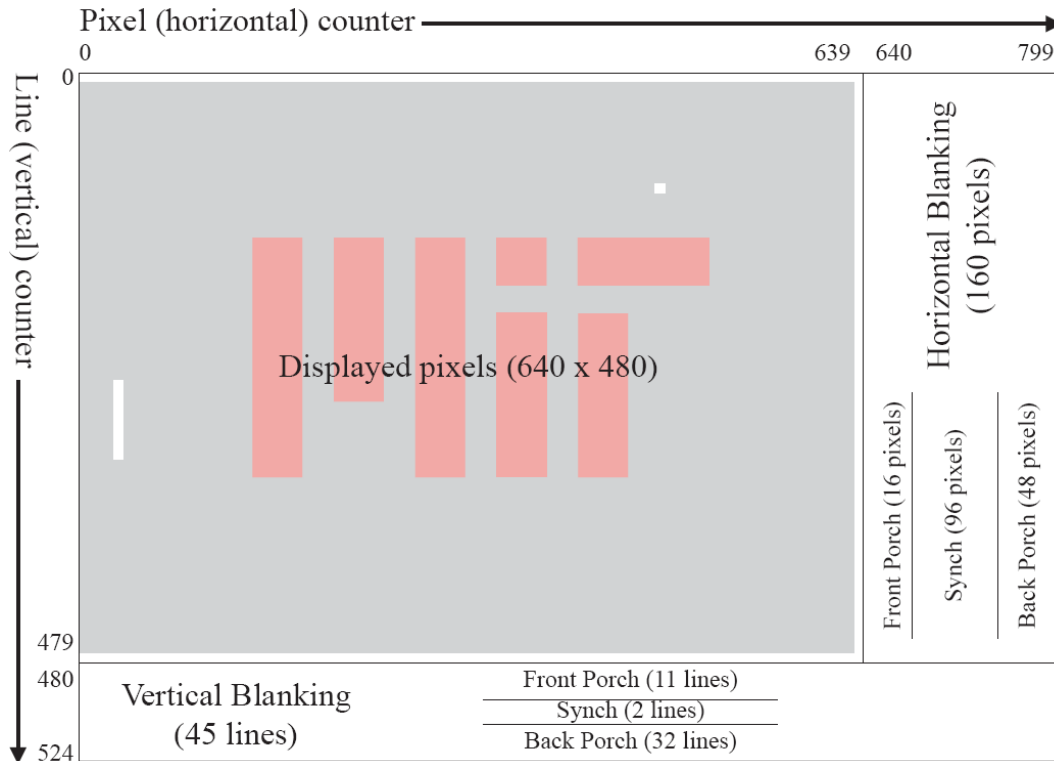


Figure 6. Specifications for video timing.

6. Game Mechanics and Checkoff

The goal of this game is to keep the ball away from the left edge of the screen (i.e., past the paddle). The user can use the paddle to defend the goal and hit the ball back to the right side. Because there is no second player on the right side, the ball will inevitably return when it bounces off of the stationary wall, and the user must once again defend the goal. The user loses if the paddle misses the ball and the ball reaches the left edge of the screen. Unfortunately, the user will never win (for checkoff).

When the ball hits the wall on the right edge of the screen, the ball should respond as one might expect. The ball should bounce back towards the left edge of the screen, and the ball should retain any vertical velocity it had when hitting the wall. If the ball hits the ceiling or floor, the vertical velocity should change sign, while the ball retains the same horizontal velocity. To summarize, when the ball hits the border, the direction of the ball will change, but the actual speed is not affected.

The paddle, however, affects both the speed and direction of the ball. When the ball hits the user's paddle, the ball should bounce back towards the right. However, there is a slight twist: The velocity of the ball changes depending where the ball hits the paddle. Think about breaking up the paddle into 8 regions. Hitting the ball with the center of the paddle increases the horizontal velocity, so the game gets progressively harder. As the ball hits the edges of the paddle, the vertical

velocity is affected. Hitting the ball with the upper section of the paddle creates more upwards velocity. Hitting the ball with the lower section of the paddle creates more downwards velocity.

To further clarify, consider an example demonstrating this idea. Suppose the ball is heading in a downwards direction when the paddle hits the ball. If the ball hits the bottom section of the paddle, the paddle increases the ball's vertical velocity downwards, so the ball is heading downwards at a faster speed than before it hit the paddle. If the ball hits the upper section of the paddle, the ball actually heads downwards at a slower speed than before it hit the paddle. In fact, if the ball has very little vertical velocity downwards, hitting the ball on the very edge of the upper section of the paddle may actually change the ball's velocity so the ball starts heading upwards when it hits the paddle. These properties are what most pong games do, and are not very intuitive.

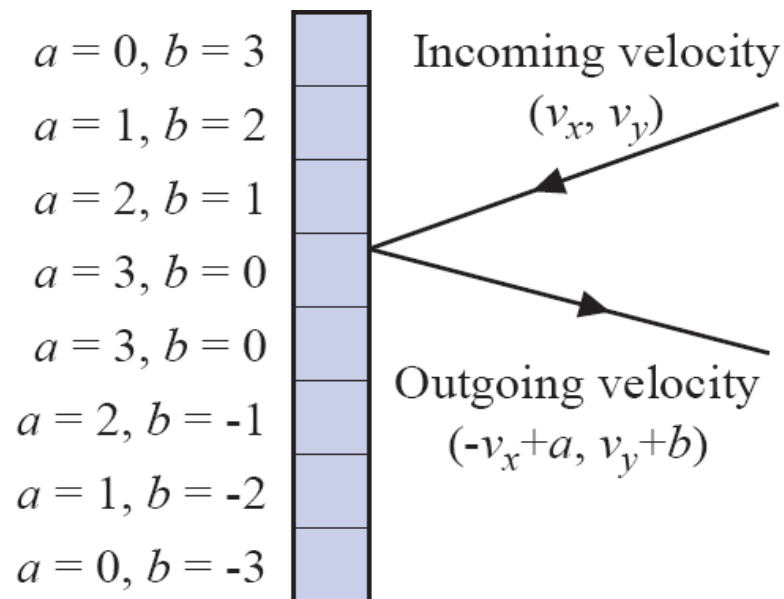


Figure 7. Possible Paddle Configuration

It would be helpful to read these properties carefully to make sure you fully understand how the ball should react to being hit before you start coding.

In terms of horizontal velocity, when the ball hits the paddle, the ball should bounce back towards the right. Hitting close to the center increases horizontal velocity by a large amount. Hitting farther from the center increases horizontal velocity by a lesser amount. Figure 7 illustrates a possible setup for how the different sections of the paddle can affect the velocities. You may use this setup or something similar that illustrates the same concept. However, you must clearly specify it in the report.

If you wish, you might want to create a simple version of Pong initially, where the paddle behaves identically to the three walls. That is, it simply changes the x-direction on contact, without change in the magnitude of the velocity.

Major-Minor FSMs

You are required to structure your system in a Major-Minor FSM setup. The major FSM should be fairly simple, and control the minor FSMs. The minor FSMs should control the rest of the system, and you should be careful when considering how to modularize.

7. Laboratory Report Requirements

We require that you turn in a detailed report for this lab. Your report should emphasize both the theory of the design and practical implementation considerations.

Cover Page/Abstract

Include a cover page with a title, your name, the name of your TA, the course name, and the date.

Introduction

Give a brief description of the problem and a block diagram of your system.

Module Description/Implementation

(a) Describe your control FSMs

- Define your major and minor FSMs.
- Include a state transition diagram for each FSM.
- Describe (in words) the operation of each FSM. Include your commented Verilog code.

(b) Detailed block diagrams of logic implemented in your FPGA

Try to make a reasonable compromise between legibility and detail. You do not have to draw detailed equivalent circuits to describe the contents of the FPGA. You should include a detailed block diagram showing major functional units with annotated widths. Each block should be accompanied by a paragraph describing the function of the FPGA circuitry.

(c) Timing diagrams for major signals - refer to these timing diagrams in your detailed descriptions.

Testing/Debugging

Describe how you tested your digital system. Provide us with a description of the design methodology you used in the creation of your digital system, and how you planned on testing each block in the design stage and how you actually ended up testing it. Be sure to include specific details.

Conclusion¹

We are particularly interested in hearing about what you learned from the completion of this lab and what you think are the important concepts to take away from the design of this digital system.

1. If you have any suggestions about how to improve this lab, please email anantha@mtl.mit.edu (Anantha Chandrakasan), cdeonier@mit.edu (Christian R Deonier) or nickes@mit.edu (Nathan Ickes)
2. We would like to thank Gim Hom, Kyle Gilpin and Javier Castro for their help in the development and debugging of this lab.