EE 371 Autumn 2016 - Lab 3

William Li, Dawn Liang, Jun Park

November 15, 2016

Signatures We certify that the work in this report is our own, and that any work that is not ours is cited.

William Li	Dawn Liang	Jun Park
_____	_____	_____
Signature	*Signature*	*Signature*
_____	_____	_____
Date	*Date*	*Date*

Contributions William helped with initial hardware design, debugging C code, and the write-up. Dawn wrote the verilog code and tested in iverilog and gtkwave, helped load onto the board and debug with SignalTap, and wrote and compiled the write-up. Jun helped design the system and draw diagrams, wrote verilog code and the C program part of the lab, and helped load onto the board and debug with SignalTap.

Contents

1 Abstract **1**

2 Introduction **1**

3 Discussion **1**
3.1 Design . . . 1
3.1.1 Design Specification . . . 1
3.1.2 Design Procedure . . . 1
3.1.3 System Description . . . 2
3.1.4 Software Implementation . . . 5
3.1.5 Hardware Implementation . . . 6
3.2 Test . . . 6
3.2.1 Test Plan . . . 6
3.2.2 Test Specification . . . 7
3.2.3 Test Cases . . . 7

4 Results **8**
4.1 Analysis of Errors . . . 9

5 Summary & Conclusion **9**

6 Appendix **10**
6.1 Two-scanner system . . . 10
6.1.1 Block diagrams . . . 10
6.1.2 Verilog code . . . 12
6.1.3 iverilog & gtkwave waveforms . . . 41
6.1.4 Signal Tap II data . . . 42
6.2 C Program . . . 43

1 ABSTRACT

In this lab, we were tasked with creating a digital 2-scanner system that interacts with a base station to collect underwater data along the canals of Venice. Additionally, we furthered our knowledge of the C programming language by creating two C programs: a temperature conversion program and a digital circuit delay calculator. These projects helped us develop our skills with the various tools and programs used to create both advanced digital designs and complex programs.

2 INTRODUCTION

The bulk of the lab focused on our digital design project, a digital continuous 2-scanner system, for retrofitting on gondolas to record valuable underwater environment data along the canal. We worked through the design process by first creating a high-level overview of the system, deciding the necessary modules for the design to work. Next, we decided on the inputs and outputs of each module and the relationships between them. We then built each module in verilog, simulated them in iverilog and gtkwave, uploaded our complete design into Quartus and tested on the DE1-SoC, and used SignalTap II to probe our design and detect errors. Finally, we furthered our knowledge of the C language by writing making two programs: a temperature conversion program and a digital circuit delay calculator.

3 DISCUSSION

3.1 Design

3.1.1 Design Specification

VHDL The continuous 2-scanner system must constantly be scanning the underwater environment. It does so by trading off scanning and transferring/flushing between the two scanners. Each scanner emits a signal telling the home base that it is ready to transfer as its data buffer reaches 50% full. When a scanners buffer reaches 80% full, it wakes up the other scanner, and at 90% it signals the other scanner to start scanning. At 100%, it enters the idle state, where it waits for a signal to let it begin transferring data, or it is told to flush out its buffer by the other scanner as it reaches 50%. It takes at least half as long to flush or transfer data as it does to acquire; each scanner must also flush data quickly enough that it is ready by the time the other scanner issues a command.

C Program The temperature conversion program is required to take an input and output temperature scale (F, C, or K) and an input temperature value, and output the temperature converted into F, C, and K. The gate delay calculator must take the number of gates in a signal path as input, and output the total gate delay in that path. As an extra feature, both programs prompt the user for a new input if the user inputs an invalid value. The re-prompt message explains why the earlier input was invalid so that the user can correct their mistake.

3.1.2 Design Procedure

VHDL The main components of the digital scanning system are the two scanners. Each scanner consists of a control system and a memory system. The control system cycles through various states: low power, standby, scanning, idle, transferring, and flushing. Thus, we used an FSM to model it. The memory system was implemented using a basic counter, the value of which indicated the percent full of the data buffer. To model different speeds for scanning, transferring, and flushing, we made various clock speeds that connected to the counter depending on the current state of the scanner. The overall scanner module handled the scanners input/output signals, such as the signal indicating that it is ready to transfer and the signals communicating with the second scanner.

C Program For temperature.c we first had to understand the equations to convert from one scale to another. The following equations were used: `tempC = (tempF-32.0)* (5.0/9.0);` and `tempK = tempC + 273.15;`

For calculateDelay.c we had to include the delay introduced by the trace in calculating the total delay by using this equation: `totalDelay = gates*5000 + (gates-1)*18;` The totalDelay is in units of picoseconds, and this was converted to nanoseconds before being displayed to the user for easy viewing.

After discovering that it was very easy to input an invalid value, we decided to include a way to capture invalid user inputs. In case, by re-prompting the user. For example, inputting a string instead of an integer when the program is expecting an integer should not crash the program, but instead cause the program to re-prompt.

3.1.3 System Description

VHDL The digital scanning system continuously collects data from the underwater environment by cycling between two scanners.

Inputs to the system

User inputs

- reset: the system resets to be inactive, so both scanners are in the low power state and the data buffer is clear
  - SA0: the system will never reset to known values at startup, so it cannot begin normal operation
  - SA1: the system will be in constant reset mode
- initialOn: the system is initially inactive. It switches on and begins scanning when it receives an initial on signal, at which point in begins continuously scanning and cycling between the two scanners.
  - SA0: the system will never turn on initially, so it will never begin scanning
  - SA1: the system will always be on; upon reset, the system will not become inactive but rather begin scanning immediately, and otherwise operate normally
- startTransfer: once the scanners have reached 100% (data buffer full), they must wait for a signal from the base station in order to transfer data
  - SA0: the scanners will never transfer data, always flushing it out
  - SA1: the scanners will always transfer their data, never flushing

Outputs of major functions

Communication between the scanners

- goToStandby: when one scanner reaches 80%, it issues a goToStandby signal to the other scanner, telling it to wake up and prepare to scan
  - SA0: the scanners will not tell each other to wake up, so there will not be continuous cycling of scanning
  - SA1: the scanners will immediately go to standby mode once they finish flushing/transferring, otherwise operating normally
- startScan: when one scanner reaches 90%, it issues a startScan signal to the other scanner, telling it to begin scanning and collecting data
  - SA0: the scanners will not tell each other to start scanning, so there will not be continuous cycling of scanning
  - SA1: the scanners will immediately start scanning once they have gone to standby mode, otherwise operating normally
- flush: when one scanner reaches 50%, it sends a flush signal to the other scanner. if the other scanner is at 100% (has not transferred its data), it will empty its data buffer.

  - SA0: the scanners will not tell each other to flush, so they will remain in idle until they receive a startTransfer signal, breaking the continuous scanning cycle
  - SA1: the scanners will always flush their data once they reach the idle state, never tranferring to the base station

Overall scanner system

- readyToTransfer: when a scanner reaches 50%, it sends a readyToTransfer signal to the base station.
  - SA0: the base station will never think the scanner is ready to transfer, so it will never issue a startTransfer signal and the scanner system will never tranfer its data (always flush)
  - SA1: the base station will think the scanner is always ready to transfer, so it will constantly issue a startTransfer signal; otherwise, the system operates normally
- Scanner states & progress: each scanner outputs which state it is currently in and its buffer fill percentage
  - if any of these values fault, the scanner will break fatally

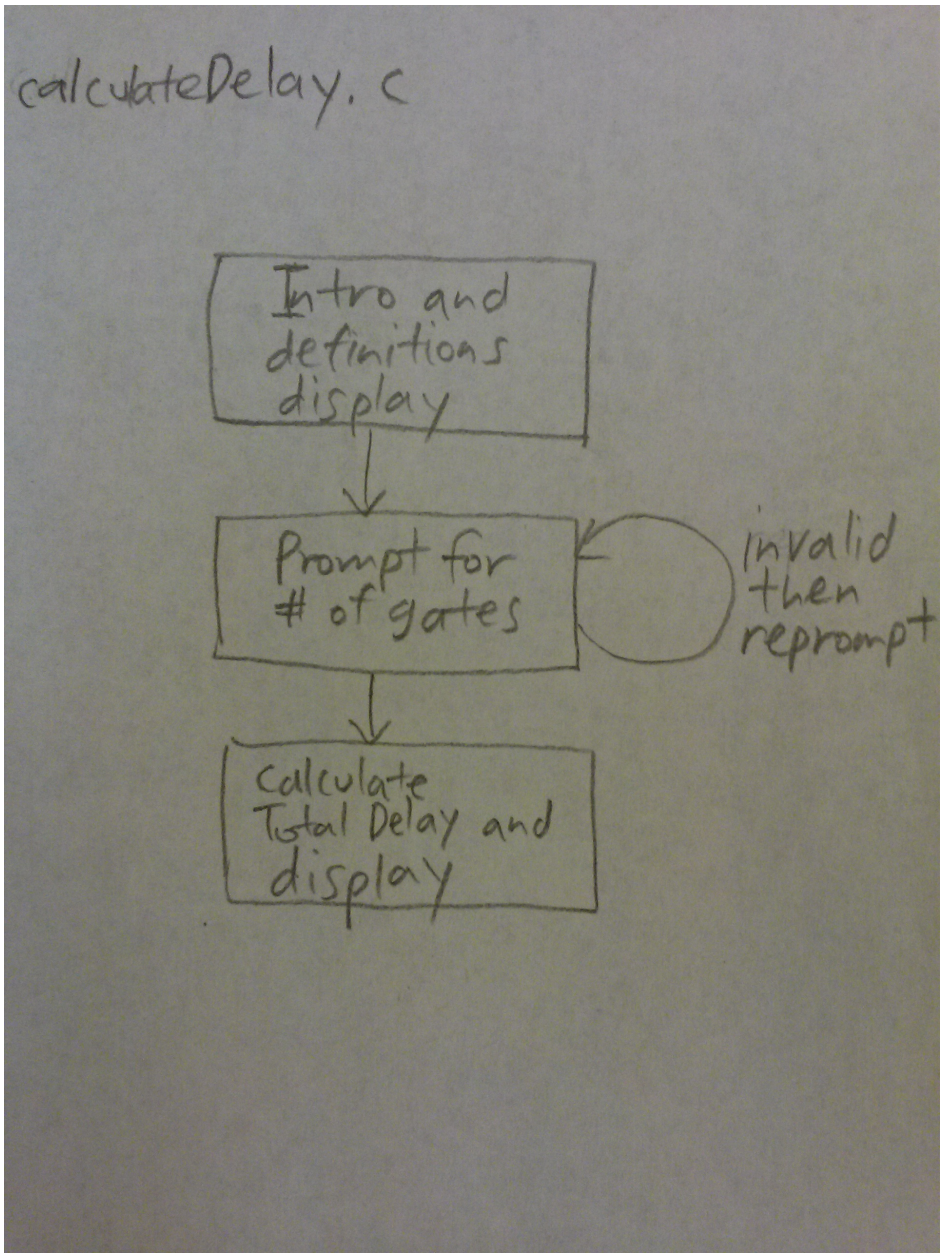

Figure 1: Block diagram of calcDelay.c program

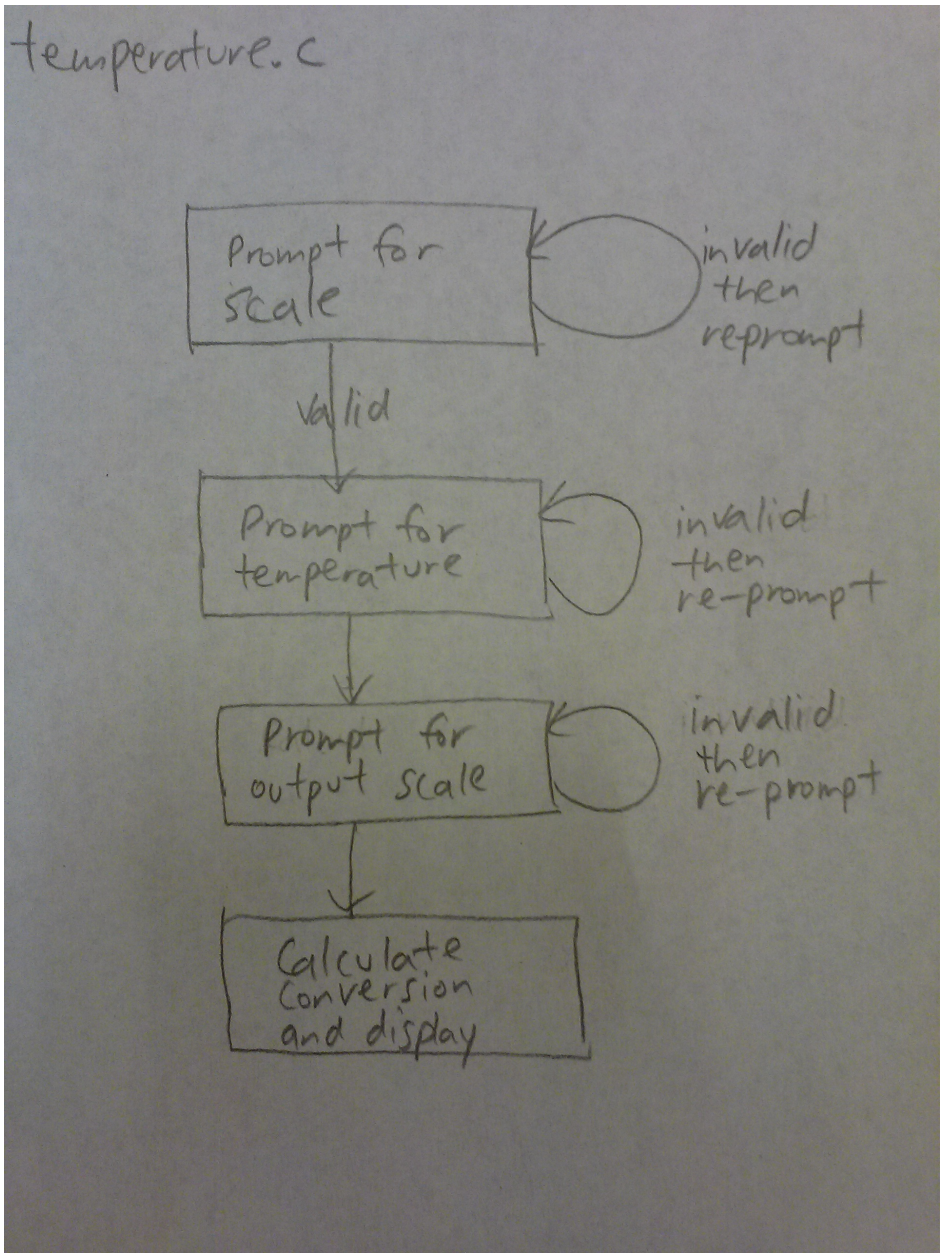

Figure 2: Block diagram of temperature.c program

3.1.4 Software Implementation

The temperature.c program prompts the user for a temperature scale, the temperature, and a temperature scale to convert to. The program then converts and displays the temperature in the scale requested by the user. The program re-prompts the user if the user inputs an invalid temperature scale. The program also re-prompts if the user inputs an invalid temperature. When a valid scale and a valid temperature is input by the user, the program prompts for a temperature scale to convert to. The program re-prompts the user if the user inputs an invalid scale. Once all user inputs are correct, the program converts the temperature and displays the result.

The calculateDelay.c program prompts the user for the number of logic gates in a signal path, then calculates and displays the total delay along that path both in picoseconds and nanoseconds. The program re-prompts the user if the

user inputs an invalid number of gates. The program also displays a definition of how the total delay is calculated upon startup.

3.1.5 Hardware Implementation

The digital scanner contains two scanners that transfer data to a base station and alternate in scanning cycles. They communicate with the base station to transfer data, and communicate with each other to indicate when to wake up, start scanning, or flush data.

Each scanner has a control system and a memory system.

The control system determines the scanners state: low power, standby, scanning, idle, transferring, or flushing. Based on input signals from the other scanner and from base station, as well as the current buffer percentage, the control system transitions the scanner between these states.

The state of the scanner then determines the behaviour of the memory system - whether it is collecting data, holding, or transferring/flushing data. Since collecting data must take at least twice as long as transferring/flushing data, the memory system must also be modeled using variable clock speeds. We did this using a counter that connected to a different speed clock depending on the scanners state.

The scanner outputs various signals to communicate both with the other scanner and the base station, as well as display its current state and progress. These signals are implemented through combinational logic, as they occur when the scanners buffer has reached a specific percentage in the scanning state (i.e. `out = (buffer == percentage && state == desiredState)`).

For viewing on the DE1-SoC, each scanners state and progress are displayed on three HEX displays per scanner, and the 1-bit output signals are displayed using LEDs. The user input signals (reset, initialOn, and startTransfer) are assigned to KEYs and SWs, to model base station signals or turning the system on initially.

3.2 Test

3.2.1 Test Plan

VHDL The digital scanner system was tested under normal operating conditions: it should begin scanning when prompted. Upon reaching 50%, it must issue a readyToTransfer signal as well as a signal for the other scanner to flush. At 80%, it should wake up the other scanner, and at 90% it should tell the other scanner to start scanning. At 100%, the scanner should enter the idle state, where it waits for either a startTransfer signal from the base station or a flush signal from the other scanner, whichever comes first. It should then enter the lowPower state, and go to standby and begin scanning at the prompting of the other scanner. As a detail, it is important that each scanner flushes quickly enough (when the other scanner reaches 50%) that it is ready to go to standby when the other scanner prompts it to (at 80%).

There were relatively few inputs, so failure conditions were few. Of these, we tested that the system only transferred once at 100% in the idle state (did not transfer if it received the signal while not idle), that it issued the output/scanner communication signals only while scanning, and what occurred when the scanner received both the startTransfer and flush signal at the same time. Finally, the reset input should put the system in the reset state.

C Program The temperature.c program needs to be run with the various temperature scale and temperature inputs to make sure it is outputting the correctly converted value to the correct output scale. The calculateDelay.c program needs to be run with various gate number inputs to make sure it is outputting the correct Total Delay time both in picoseconds and nanoseconds. Both temperature.c and calculateDelay.c need to be tested with non valid inputs to ensure they are properly handling invalid inputs with a re-prompt to the user.

3.2.2 Test Specification

VHDL The primary outputs of the two-scanner system, the state and data buffer percentage of each scanner, were monitored. Additionally, each individual scanners output signals for communication between the scanners were monitored. As there were only three inputs, it was relatively simple to test the system constraints.

The scanner should transition through the states in order, as depicted in the state diagram (see state transition diagram). Additionally, it should be filling the data buffer (increasing percentage) in the scanning state, and emptying the data buffer (decreasing percentage) in the transferring and flushing states. The data buffer should empty at least twice as fast as it fills, and for flushing, it should empty by the time the other scanner reaches 80% and sends a goToStandby signal. On reset, the scanner should go to the lowPower state and the data buffer should go to 0%.

Each scanners communication signals should only output when in the scanning state (e.g. it should not issue a readyToTransfer signal when it reaches 50% while flushing). Additionally, the signals should become active at the correct percentage of data buffer full: 50% for readyToTransfer and otherFlush, 80% for otherGoToStandby, and 90% for otherStartScan.

C Program The temperature.c program needs to be tested with various input temperature scales, temperature inputs, and output temperature scales to ensure that it is converting the temperatures correctly and according to the scale specified by the user. For example, inputs: F, 32, C should output 0 degrees on the Celsius scale.

The calculateDelay.c program needs to be tested with various integer inputs to ensure it is calculating the total path delay correctly while following the definitions that: each gate has a 5ns delay, 1in of trace has a delay of 180ps, and that the output of one device to the input of the next device is connected by 0.1in of trace. For example, input of 5 should output 25072ps and 25.072ns.

Both programs need to be tested with likely invalid user inputs to make sure the programs are handling the invalid inputs with a correct re-prompt. For example, inputting a non valid scale like j into temperature.c should cause it to re-prompt for a valid scale, while inputting a non valid gate number like 4.3 into calculateDelay.c should cause it to re-prompt for a valid gate number.

3.2.3 Test Cases

VHDL

Pass cases

- Reset case
  - Input: reset = 1
  - Measure: state and data buffer
  - Pass condition: state = lowPower, data_buffer = 0 (lowPower mode and empty data buffer)
- turning on the scanner system
  - Input: initialOn = 1
  - Measure: states of each scanner
  - Pass condition: one scanner should be in scanning mode, and the other should be in lowPower mode
- transferring data
  - Input: startTransfer = 1
  - Measure: state of the scanner
  - Pass condition: if idle, the scanner should go to transferring mode. Otherwise, the scanner should hold its state

Fail cases

- transferring while not on
  - Input: initialOn = 0, startTransfer = 1
  - Measure: state of the scanner
  - Pass condition: nothing should happen, as the scanners havent been turned on yet
- transferring while not in the idle state
  - Input: startTransfer = 1 while not in the idle state
  - Measure: state of the scanner
  - Pass condition: nothing should happen, since the scanner isnt in the idle state
- simultaneous startTransfer and flush signals
  - Input: startTransfer = 1 & flush = 1
  - Measure: state of the scanner
  - Pass condition: the scanner should prioritise transferring over flushing, so it should go to the transferring state

C Program

- Address all compiler errors then compile and run temperature.c and calculateDelay.c.
- Valid temperature scale inputs for temperature.c are: f, F, c, C, k, K where each character represents Fahrenheit, Celsius, or Kelvins. Try each of these inputs.
- Valid gate number input for calculateDelay.c are non-negative integers. For example, you cant have half a gate, nor a negative number of gates. Try the valid inputs (non-negative integers) first.
- temperature.c should output a correctly converted decimal value representing the temperature in the specified scale rounded to two decimal places.
- calculateDelay.c should output a correctly calculated Total Delay and output that value in both picoseconds and nanoseconds.
- For both programs, intentionally input an invalid value and make sure that the program re-prompts instead of crashing or giving an incorrect value. For example, inputting a string when an integer or double is expected by the program and observing what the program does.

4 RESULTS

VHDL The digital scanning system performed as expected under normal operating conditions and under edge and fail cases.

When the system is reset, it goes to the expected reset state (lowPower & 0% buffer). Nothing happens until it is turned on (initialOn = 1), at which point one scanner begins scanning while the other stays in lowPower. When the first scanner reaches 50%, it issues a readyToTransfer signal to the base station and a flush signal to the other scanner. When it reaches 80%, it wakes up the other scanner, and at 90% it tells the other scanner to begin scanning. At 100%, it goes into the idle state while the other scanner is scanning, and waits for a startTransfer signal from the base station or a flush signal from the other scanner, whichever comes first. It transfers data/flushes accordingly, then goes to the lowPower state and waits to be woken up by the other scanner. The scanners trade off and cycle through these states accordingly, so that there is always at least one scanner scanning at all times.

If a scanner is given an input before it has been turned on, it doesnt do anything, as expected. Additionally, each scanner only transfers once it has reached the idle state; it does not transfer until it has reached 100%. Finally, when a scanner is given both the startTransfer and flush signal simultaneously, it prioritises transferring over flushing, and goes into the transfer state.

C Program temperature.c correctly converts and displays the temperature according to the specified input and output scales. calculateDelay.c correctly calculates and displays the total delay according to the specified number of gates. Both programs handled invalid user inputs correctly by explaining why the input was incorrect and asking the user for another value.

4.1 Analysis of Errors

VHDL When trying to load SignalTap onto the board, we ran into some problems, which we solved by reloading the project and remembering to reset the board between loading through the programmer and through SignalTap.

For the design of the scanner system, we had trouble figuring out how to get our state machine to skip the standby mode when initially turned on. We solved this by refactoring our code and placing the normal operating mode at higher priority than initialisation of the cycle.

C Program The initial error we ran into for temperature.c was that it was converting the temperature to the different scales correctly but that the program wasnt displaying only the correct scale, but instead dumping all the conversions to the output. This was fixed by if/else statements checking to see what the user specified as an output scale.

The handling of invalid user inputs was the most challenging, but it was fixed by a while/break combination.

5 SUMMARY & CONCLUSION

Summary The focus of this project was to design a scanner system for gondolas to collect data from the underwater environment. The scanner systems continuously collect data between the two scanners, and communicate with an external base station operated by the user. The base station turns the scanners on and signals when to transfer data. As a second component, the project involves writing a C program, working with different data types and variable handling.

Conclusion Overall, this project gave us practice in building more advanced VHDL designs. The scanner system had a very simple workflow, with enough variables to make it testable for failure conditions. We were able to practice working with iverilog/gtkwave and Quartus/SignalTap more, as well as learn improve our programming skills in the C programming language.

6 APPENDIX

6.1 Two-scanner system

6.1.1 Block diagrams

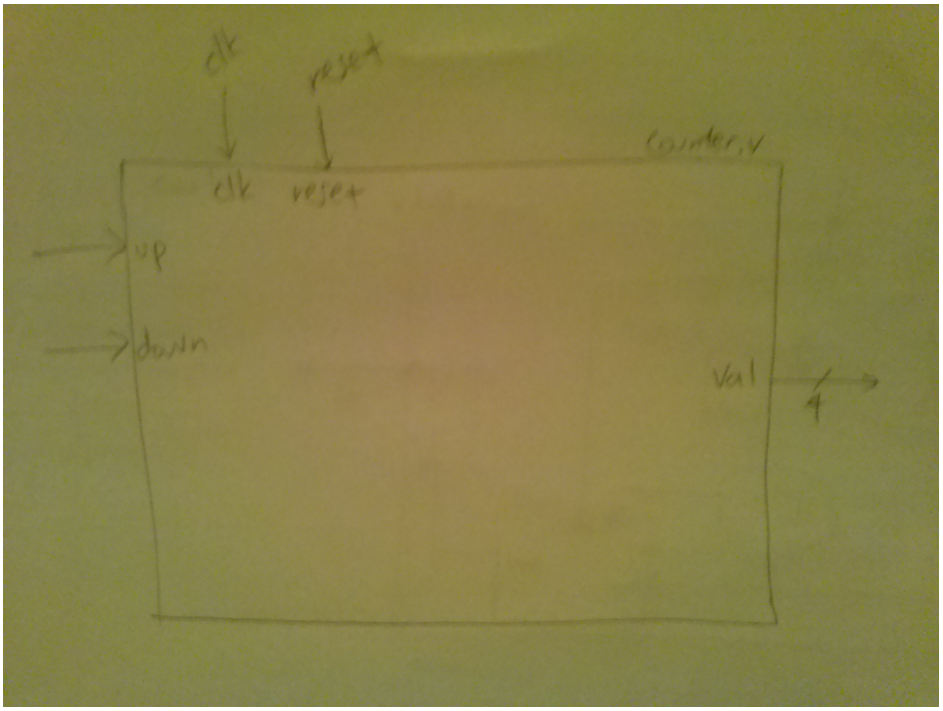

Figure 3: counter module block diagram

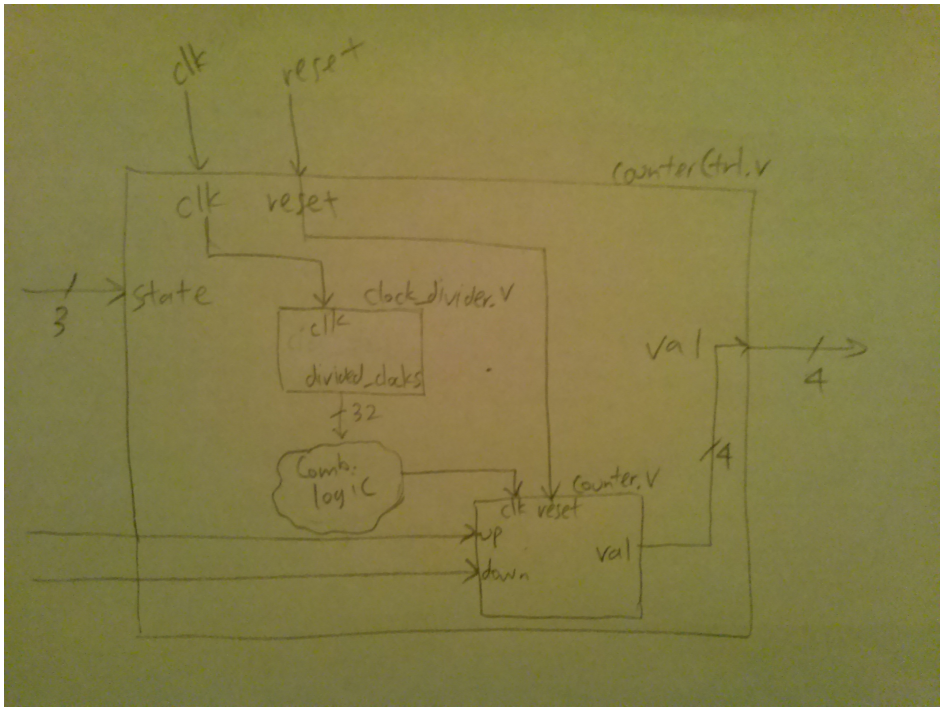

Figure 4: counterCtrl module block diagram

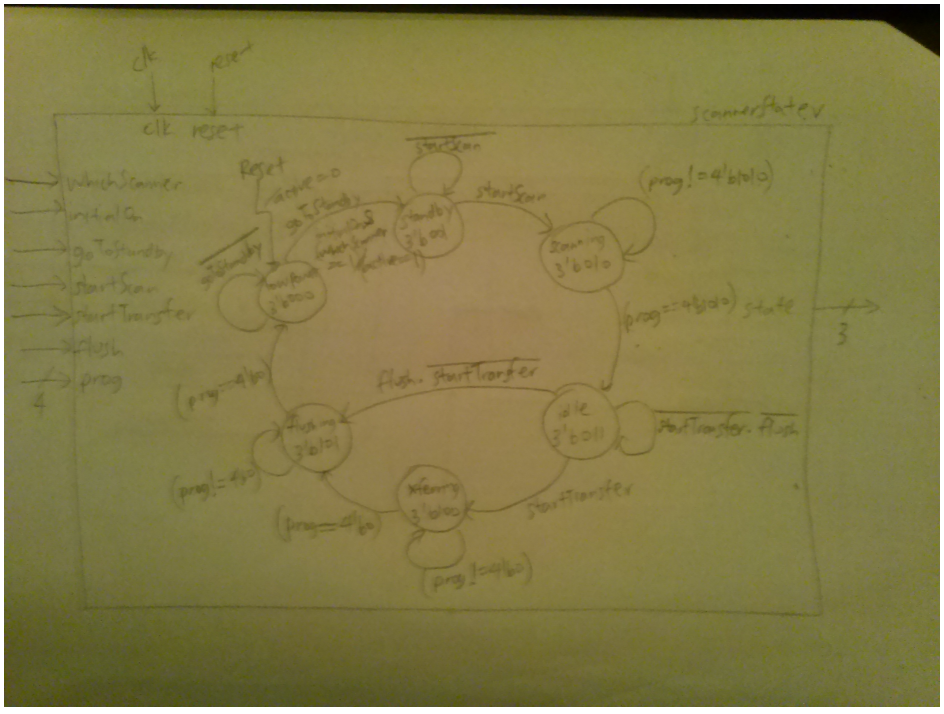

Figure 5: state transition diagram for the scanners

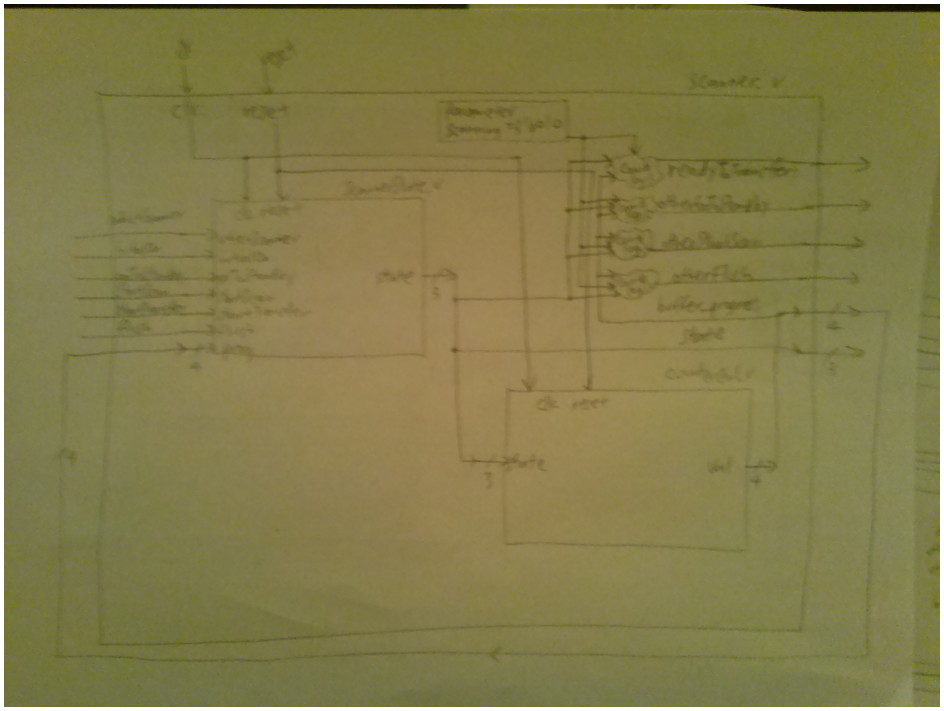

Figure 6: scanner module block diagram

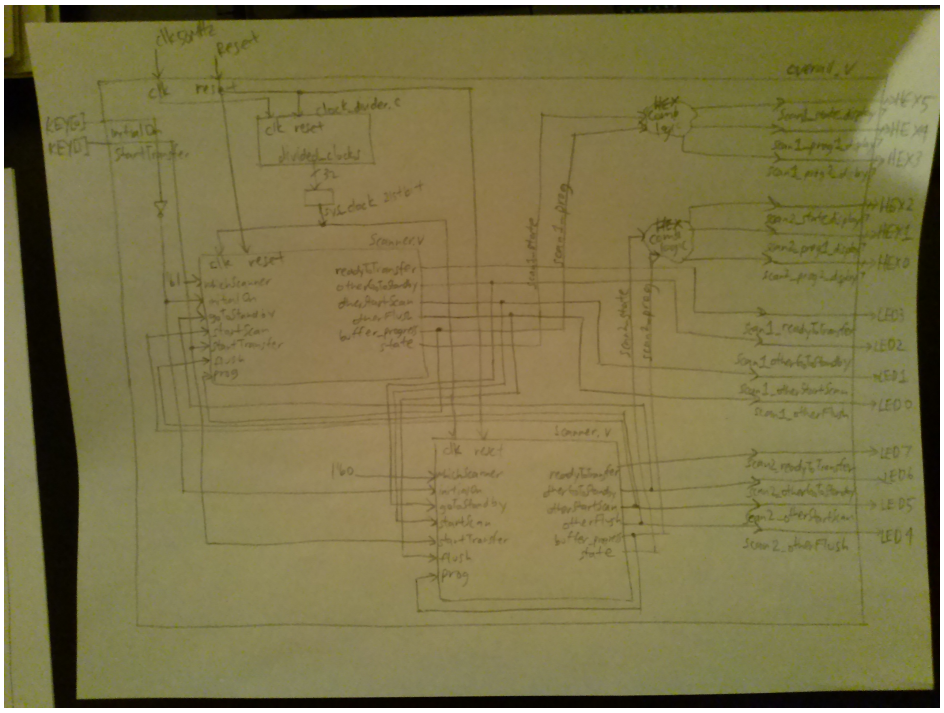

Figure 7: overall module block diagram

6.1.2 Verilog code

```
// EE371 Lab3 Autumn 2016
```

```
// Authors: Dawn Liang, Jun Park, William Li
// Date: 13 Nov 2016
//
// counter that counts every clock edge up/down to a min/max (0/10)
// resets to 0
module counter(val, up, down, clk, reset);
  output [3:0] val;
  input up, down, clk, reset;

  // combinational logic: count every clock edge
  // min/max set at 0/10
  reg [3:0] ps, ns;
  always@(*) begin
    if (up && ~down && (ps < 4'b1010)) begin
      ns = ps + 1'b1;
    end else if (down && ~up && (ps > 4'b0)) begin
      ns = ps - 1'b1;
    end else begin
      ns = ps;
    end
  end

  // output logic
  assign val = ps;

  // sequential logic; reset & next state
  always@(posedge clk) begin
    if (reset) begin
      ps <= 4'b0;
    end else begin
      ps <= ns;
    end
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, William Li, Jun park
// Date: 13 Nov 2016
//
// testbench for the counter module; tests all counting conditions
module counter_testbench();
  reg up, down, clk, reset;
  wire [3:0] val;
  wire final_clock;

  counter dut (.val(val), .up(up), .down(down), .clk(clk), .reset(reset));

  // set up clock
  parameter CLOCK_PERIOD = 10;
  initial clk = 0;
  always begin
    #(CLOCK_PERIOD/2);
    clk = ~clk;
  end

  // begin simulation
  initial begin
    reset <= 1;            @(posedge clk);
```

```
                        @(posedge clk);
    reset <= 0; up <= 0; down <= 0;  @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
          up <= 1;      @(posedge clk); // 0
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk); // 5
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk); // 10
                        @(posedge clk);
                        @(posedge clk);
               down <= 1;  @(posedge clk);
          up <= 0;      @(posedge clk); // 10
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk); // 5
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk);
                        @(posedge clk); // 0
                        @(posedge clk);
               down <= 0;  @(posedge clk);
                        @(posedge clk);
    $finish;
  end

  // gtkwave filedump
  initial begin
    $dumpfile("counter.vcd");
    $dumpvars;
  end
endmodule
```

```
// EE371 Lab 3 Autumn 2016
// Authors: Jun Park William Li Dawn Liang
// Date: 13 Nov 2016
//
// controls a counter that counts up/down at different speeds depending on
// the state input
module counterCtrl(val, state, clk, reset);
  input [2:0] state;
  input clk, reset;

  output [3:0] val;

  // state encodings
  parameter lowPower = 3'b000,
        standby = 3'b001,
        scanning = 3'b010,
```

```
        idle = 3'b011,
        xferring = 3'b100,
        flushing = 3'b101;

  // clock division
  wire [31:0] divided_clocks;
  clock_divider div (.divided_clocks(divided_clocks), .clk(clk));
  reg final_clock;

  // control signals
  reg up, down;
  always@(*) begin
    case (state)
      lowPower: begin
                final_clock = divided_clocks[0];
                up = 0; down = 0;
              end
      standby: begin
                final_clock = divided_clocks[0];
                up = 0; down = 0;
              end
      scanning: begin                    // count up at 1x speed
                final_clock = divided_clocks[3];
                up = 1; down = 0;
              end
      idle:  begin
                final_clock = divided_clocks[0];
                up = 0; down = 0;
              end
      xferring: begin                    // count down at 4x speed
                final_clock = divided_clocks[1];
                up = 0; down = 1;
              end
      flushing: begin                    // count down at 8x speed
                final_clock = divided_clocks[0];
                up = 0; down = 1;
              end
      default: begin
                final_clock = divided_clocks[0];
                up = 0; down = 0;
              end
    endcase
  end

  // counter
  counter prog (.val(val), .up(up), .down(down), .clk(final_clock), .reset(reset));
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, Jun Park, William Li
// Date: 13 Nov 2016
//
// testbench for counterCtrl module; tests all possible state inputs
module counterCtrl_testbench();
  reg [2:0] state;
  reg clk, reset;
  wire [3:0] val;
```

```
counterCtrl dut (.val(val), .state(state), .clk(clk), .reset(reset));

// set up clock
parameter CLOCK_PERIOD = 10;
initial clk = 0;
always begin
  #(CLOCK_PERIOD/2);
  clk = ~clk;
end

initial begin
  reset <= 1;          @(posedge clk);
                      @(posedge clk);
  reset <= 0; state <= 3'b000; @(posedge clk); // lowPower
                      @(posedge clk);
                      @(posedge clk);
        state <= 3'b001; @(posedge clk); // standby
                      @(posedge clk);
                      @(posedge clk);
        state <= 3'b010; @(posedge clk); // scanning
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
        state <= 3'b011; @(posedge clk); // idle
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
                      @(posedge clk);
```

```
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
          state <= 3'b100; @(posedge clk); // xfer
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
          state <= 3'b101; @(posedge clk); // flush
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
                    @(posedge clk);
    $finish;
  end

  // gtkwave filedump
  initial begin
    $dumpfile("counterCtrl.vcd");
    $dumpvars;
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, Jun Park, William Li
// Date:
//
// FSM controlling the state of the scanner
module scannerState(state, whichScanner, initialOn, goToStandby, startScan, prog,
    startTransfer, flush, clk, reset);
  output [2:0] state;
  input whichScanner, initialOn, goToStandby, startScan, startTransfer, flush;
  input [3:0] prog;
  input clk, reset;

  // state encodings
  parameter lowPower = 3'b000,
        standby = 3'b001,
        scanning = 3'b010,
        idle = 3'b011,
        xferring = 3'b100,
        flushing = 3'b101;

  // combinational logic: next state
  reg [2:0] ps, ns;
  reg active;
  always@(*) begin     // cycling between scanners
    if (active) begin
      case (ps)
```

```
        lowPower: begin
                  if (goToStandby) ns = standby;
                  else ns = ps;
                end
        standby: begin
                  if (startScan) ns = scanning;
                  else ns = ps;
                end
        scanning: begin
                  if (prog == 4'b1010) ns = idle;
                  else ns = ps;
                end
        idle:  begin
                  if (startTransfer) ns = xferring;
                  else if (flush) ns = flushing;
                  else ns = ps;
                end
        xferring: begin
                  if (prog == 4'b0) ns = lowPower;
                  else ns = ps;
                end
        flushing: begin
                  if (prog == 4'b0) ns = lowPower;
                  else ns = ps;
                end
        default: ns = ps;
      endcase
    end else begin         // initialising: start the scanner cycles
      if (initialOn) begin
        if (whichScanner) ns = scanning;
        else ns = lowPower;
      end else begin
        ns = lowPower;
      end
    end
  end

  // output logic
  assign state = ps;

  // sequential logic: reset, next state
  always@(posedge clk) begin
    if (reset) begin
      ps <= lowPower;
      active <= 0;
    end else begin
      ps <= ns;
      if(initialOn) active <= 1;
      else active <= active;
    end
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, William Li, Jun Park
// Date: 13 Nov 2016
//
// testbench for scannerState FSM; tests all state transitions
```

```
module scannerState_testbench();
  reg clk, reset;
  reg goToStandby, startScan, startTransfer, flush, initialOn;
  reg [3:0] prog;
  wire [2:0] state;

  // set up clock
  parameter CLOCK_PERIOD = 10;
  initial clk = 0;
  always begin
    #(CLOCK_PERIOD/2);
    clk = ~clk;
  end

  scannerState dut (.state(state), .initialOn(initialOn), .prog(prog),
     .goToStandby(goToStandby), .startScan(startScan),
    .startTransfer(startTransfer), .flush(flush), .clk(clk), .reset(reset));

  initial begin
    reset <= 1;                                                    @(posedge clk);
                                                                 @(posedge clk);
    reset <= 0; goToStandby <= 0; startScan <= 0; startTransfer <= 0; flush <= 0;
        initialOn <= 0; prog <= 4'b0; @(posedge clk);
                                              initialOn <= 1;        @(posedge
                                                  clk);
          goToStandby <= 1;                     initialOn <= 0;        @(posedge
              clk);
          goToStandby <= 0;                                            @(posedge clk);
                     startScan <= 1;                                     @(posedge
                         clk);
                     startScan <= 0;                          prog <=
                         4'b1010;@(posedge clk);
                                                                  @(posedge clk);
                               startTransfer <= 1;                    @(posedge
                                   clk);
                               startTransfer <= 0;            prog <= 4'b0;
                                   @(posedge clk);
          goToStandby <= 1;                                          @(posedge clk);
          goToStandby <= 0; startScan <= 1;                              @(posedge
              clk);
                     startScan <= 0;                          prog <=
                         4'b1010;@(posedge clk);
                                        flush <= 1;                  @(posedge clk);
                                        flush <= 0;       prog <= 4'b0; @(posedge
                                            clk);
                                                                  @(posedge clk);
    $finish;
  end

  // gtkwave filedump
  initial begin
    $dumpfile("scannerState.vcd");
    $dumpvars;
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, William Li, Jun Park
```

```
// Date: 13 Nov 2016
//
// scanner module including FSM and counter for control and memory systems
// outputs current state & buffer progress, and control signals for other
// scanner
module scanner(buffer_progress, state, readyToTransfer, otherGoToStandby,
    otherStartScan, otherFlush,
         whichScanner, initialOn, goToStandby, startScan, startTransfer, flush, clk,
             reset);
  input clk, reset;
  input initialOn, goToStandby, startScan, startTransfer, flush, whichScanner;

  output readyToTransfer, otherGoToStandby, otherStartScan, otherFlush;
  output [3:0] buffer_progress;
  output [2:0] state;

  parameter scanning = 3'b010;

  scannerState statedet (.state(state), .whichScanner(whichScanner),
      .initialOn(initialOn), .goToStandby(goToStandby), .startScan(startScan),
    .prog(buffer_progress), .startTransfer(startTransfer), .flush(flush), .clk(clk),
        .reset(reset));
  counterCtrl prog (.val(buffer_progress), .state(state), .clk(clk), .reset(reset));

  assign readyToTransfer = (buffer_progress == 5 && state == scanning);
  assign otherGoToStandby = (buffer_progress == 8 && state == scanning);
  assign otherStartScan = (buffer_progress == 9 && state == scanning);
  assign otherFlush = (buffer_progress == 5 && state == scanning);
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, William Li, Jun Park
// Date: 13 Nov 2016
//
// testbench for scanner module: tests normal operationg conditions
module scanner_testbench();
  reg clk, reset;
  reg initialOn, goToStandby, startScan, startTransfer, flush;
  wire readyToTransfer, otherGoToStandby, otherStartScan, otherFlush;
  wire [3:0] buffer_progress;
  wire [2:0] state;

  // set up clock
  parameter CLOCK_PERIOD = 10;
  initial clk = 0;
  always begin
    #(CLOCK_PERIOD/2);
    clk = ~clk;
  end

  scanner dut (.buffer_progress(buffer_progress), .state(state),
      .readyToTransfer(readyToTransfer),
    .otherGoToStandby(otherGoToStandby), .otherStartScan(otherStartScan),
        .otherFlush(otherFlush), .whichScanner(1'b1),
    .initialOn(initialOn), .goToStandby(goToStandby), .startScan(startScan),
        .startTransfer(startTransfer), .flush(flush),
    .clk(clk), .reset(reset));
```

```
initial begin
  reset <= 1;                                                @(posedge clk);
                                                           @(posedge clk);
  reset <= 0; initialOn <= 0; goToStandby <= 0; startScan <= 0; startTransfer <= 0;
      flush <= 0;  @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                  goToStandby <= 1;                          @(posedge clk);
                  goToStandby <= 0;                          @(posedge clk);
                                                           @(posedge clk);
         initialOn <= 1;                                       @(posedge clk);
         initialOn <= 0;                                       @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
                                                           @(posedge clk);
```

```
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
            startTransfer <= 1;           @(posedge clk);
            startTransfer <= 0;           @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
```

```
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
goToStandby <= 1;                         @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
           startScan <= 1;                   @(posedge clk);
goToStandby <= 0;                         @(posedge clk);
                                        @(posedge clk);
           startScan <= 0;                   @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
                                        @(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                flush <= 1;  @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                flush <= 0;  @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
                                                    @(posedge clk);
    $finish;
  end

  // gtkwave filedump
  initial begin
    $dumpfile("scanner.vcd");
    $dumpvars;
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, Jun Park, William Li
// Date: 13 Nov 2016
//
// overall module for digital 2-scanner system
// creates 2 scanners & interprets their outputs into the DE1-SoC HEX displays
module overall(scan1_state, scan1_prog, scan2_state, scan2_prog,
       scan1_otherGoToStandby, scan1_otherStartScan, scan1_otherFlush,
       scan2_otherGoToStandby, scan2_otherStartScan, scan2_otherFlush,
       scan1_state_display, scan1_prog1_display, scan1_prog2_display,
           scan1_readyToTransfer,
```

```
     scan2_state_display, scan2_prog1_display, scan2_prog2_display,
         scan2_readyToTransfer,
     initialOn, startTransfer, clk, reset);
input clk, reset;      // 50MHz clock, SW[0]
input initialOn, startTransfer;  // KEY[0], KEY[1]

output reg [6:0] scan1_state_display, scan1_prog1_display, scan1_prog2_display, //
    HEX 5:3
              scan2_state_display, scan2_prog1_display, scan2_prog2_display; // HEX
                  2:0
output  scan1_readyToTransfer, scan1_otherGoToStandby, scan1_otherStartScan,
    scan1_otherFlush, // LED[3:0]
       scan2_readyToTransfer, scan2_otherGoToStandby, scan2_otherStartScan,
           scan2_otherFlush; // LED[7:4]

// HEX encodings 6543210
parameter HEXS = 7'b0010010, // S s stands for Scanning state
       HEXt = 7'b0000111, // t t stands for transferring state
       HEXF = 7'b0001110, // F F stands for Flushing state
       HEXd = 7'b0100001, // d d stands for idle state
       HEXb = 7'b0000011; // b b stands for standby state

parameter HEXOFF = 7'b1111111,// All segments are turned off
       HEXON = 7'b0000000; // All segments are turned on

parameter  HEX0 = 7'b1000000, // 0
       HEX1 = 7'b1111001, // 1
       HEX2 = 7'b0100100, // 2
       HEX3 = 7'b0110000, // 3
       HEX4 = 7'b0011001, // 4
       HEX5 = 7'b0010010, // 5
       HEX6 = 7'b0000010, // 6
       HEX7 = 7'b1111000, // 7
       HEX8 = 7'b0000000, // 8
       HEX9 = 7'b0010000; // 9

// state encodings
parameter lowPower = 3'b000,
       standby = 3'b001,
       scanning = 3'b010,
       idle = 3'b011,
       xferring = 3'b100,
       flushing = 3'b101;

// 50 MHz clock division
wire [31:0] divided_clocks;
clock_divider cdiv (.divided_clocks(divided_clocks), .clk(clk));
wire sys_clock = divided_clocks[21];

// connecting wires
output [3:0] scan1_prog, scan2_prog;
output [2:0] scan1_state, scan2_state;

// 2 scanners
scanner scan1 (.buffer_progress(scan1_prog), .state(scan1_state),
    .readyToTransfer(scan1_readyToTransfer),
  .otherGoToStandby(scan1_otherGoToStandby), .otherStartScan(scan1_otherStartScan),
      .otherFlush(scan1_otherFlush),
  .whichScanner(1'b1), .initialOn(~initialOn),
```

```
      .goToStandby(scan2_otherGoToStandby), .startScan(scan2_otherStartScan),
   .startTransfer(startTransfer), .flush(scan2_otherFlush), .clk(sys_clock),
      .reset(reset));
scanner scan2 (.buffer_progress(scan2_prog), .state(scan2_state),
    .readyToTransfer(scan2_readyToTransfer),
   .otherGoToStandby(scan2_otherGoToStandby), .otherStartScan(scan2_otherStartScan),
      .otherFlush(scan2_otherFlush),
   .whichScanner(1'b0), .initialOn(~initialOn),
      .goToStandby(scan1_otherGoToStandby), .startScan(scan1_otherStartScan),
   .startTransfer(startTransfer), .flush(scan1_otherFlush), .clk(sys_clock),
      .reset(reset));

// output logic
always@(*) begin
  case (scan1_state)
    lowPower: begin
              scan1_state_display = HEXOFF;
              scan1_prog1_display = HEXOFF;
              scan1_prog2_display = HEXOFF;
            end
    standby: begin
              scan1_state_display = HEXb;
              scan1_prog1_display = HEX0;
              scan1_prog2_display = HEX0;
            end
    scanning: begin
              scan1_state_display = HEXS;
              case (scan1_prog)
              4'b0:  begin           // 0
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX0;
                     end
              4'b0001: begin         // 1
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX1;
                     end
              4'b0010: begin         // 2
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX2;
                     end
              4'b0011: begin         // 3
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX3;
                     end
              4'b0100: begin         // 4
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX4;
                     end
              4'b0101: begin         // 5
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX5;
                     end
              4'b0110: begin         // 6
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX6;
                     end
              4'b0111: begin         // 7
                       scan1_prog1_display = HEX0;
                       scan1_prog2_display = HEX7;
```

```
            end
          4'b1000: begin         // 8
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX8;
                   end
          4'b1001: begin         // 9
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX9;
                   end
          4'b1010: begin         // 10
                     scan1_prog1_display = HEX1;
                     scan1_prog2_display = HEX0;
                   end
          default: begin
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX0;
                   end
          endcase
        end
idle:   begin
          scan1_state_display = HEXd;
          scan1_prog1_display = HEX1;
          scan1_prog2_display = HEX0;
        end
xferring: begin
          scan1_state_display = HEXt;
          case (scan1_prog)
          4'b0:  begin           // 0
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX0;
                   end
          4'b0001: begin         // 1
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX1;
                   end
          4'b0010: begin         // 2
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX2;
                   end
          4'b0011: begin         // 3
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX3;
                   end
          4'b0100: begin         // 4
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX4;
                   end
          4'b0101: begin         // 5
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX5;
                   end
          4'b0110: begin         // 6
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX6;
                   end
          4'b0111: begin         // 7
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX7;
                   end
```

```
            4'b1000: begin         // 8
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX8;
                   end
            4'b1001: begin         // 9
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX9;
                   end
            4'b1010: begin         // 10
                     scan1_prog1_display = HEX1;
                     scan1_prog2_display = HEX0;
                   end
            default: begin
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX0;
                   end
            endcase
          end
flushing: begin
            scan1_state_display = HEXF;
            case (scan1_prog)
            4'b0:  begin           // 0
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX0;
                   end
            4'b0001: begin         // 1
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX1;
                   end
            4'b0010: begin         // 2
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX2;
                   end
            4'b0011: begin         // 3
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX3;
                   end
            4'b0100: begin         // 4
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX4;
                   end
            4'b0101: begin         // 5
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX5;
                   end
            4'b0110: begin         // 6
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX6;
                   end
            4'b0111: begin         // 7
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX7;
                   end
            4'b1000: begin         // 8
                     scan1_prog1_display = HEX0;
                     scan1_prog2_display = HEX8;
                   end
            4'b1001: begin         // 9
                     scan1_prog1_display = HEX0;
```

```
                    scan1_prog2_display = HEX9;
                end
            4'b1010: begin          // 10
                    scan1_prog1_display = HEX1;
                    scan1_prog2_display = HEX0;
                end
            default: begin
                    scan1_prog1_display = HEXOFF;
                    scan1_prog2_display = HEXOFF;
                end
            endcase
        end
    default: begin
              scan1_state_display = HEXOFF;
              scan1_prog1_display = HEXOFF;
              scan1_prog2_display = HEXOFF;
            end
endcase

case (scan2_state)
  lowPower: begin
            scan2_state_display = HEXOFF;
            scan2_prog1_display = HEXOFF;
            scan2_prog2_display = HEXOFF;
        end
  standby: begin
            scan2_state_display = HEXb;
            scan2_prog1_display = HEX0;
            scan2_prog2_display = HEX0;
        end
  scanning: begin
            scan2_state_display = HEXS;
            case (scan2_prog)
            4'b0:  begin            // 0
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX0;
                end
            4'b0001: begin          // 1
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX1;
                end
            4'b0010: begin          // 2
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX2;
                end
            4'b0011: begin          // 3
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX3;
                end
            4'b0100: begin          // 4
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX4;
                end
            4'b0101: begin          // 5
                    scan2_prog1_display = HEX0;
                    scan2_prog2_display = HEX5;
                end
            4'b0110: begin          // 6
                    scan2_prog1_display = HEX0;
```

```
                scan2_prog2_display = HEX6;
              end
          4'b0111: begin         // 7
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX7;
              end
          4'b1000: begin         // 8
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX8;
              end
          4'b1001: begin         // 9
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX9;
              end
          4'b1010: begin         // 10
                scan2_prog1_display = HEX1;
                scan2_prog2_display = HEX0;
              end
          default: begin
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX0;
              end
          endcase
        end
idle:   begin
          scan2_state_display = HEXd;
          scan2_prog1_display = HEX1;
          scan2_prog2_display = HEX0;
        end
xferring: begin
          scan2_state_display = HEXt;
          case (scan2_prog)
          4'b0:  begin           // 0
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX0;
              end
          4'b0001: begin         // 1
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX1;
              end
          4'b0010: begin         // 2
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX2;
              end
          4'b0011: begin         // 3
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX3;
              end
          4'b0100: begin         // 4
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX4;
              end
          4'b0101: begin         // 5
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX5;
              end
          4'b0110: begin         // 6
                scan2_prog1_display = HEX0;
                scan2_prog2_display = HEX6;
```

```
            end
          4'b0111: begin         // 7
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX7;
                   end
          4'b1000: begin         // 8
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX8;
                   end
          4'b1001: begin         // 9
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX9;
                   end
          4'b1010: begin         // 10
                     scan2_prog1_display = HEX1;
                     scan2_prog2_display = HEX0;
                   end
          default: begin
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX0;
                   end
          endcase
        end
flushing: begin
          scan2_state_display = HEXF;
          case (scan2_prog)
          4'b0:  begin           // 0
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX0;
                   end
          4'b0001: begin         // 1
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX1;
                   end
          4'b0010: begin         // 2
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX2;
                   end
          4'b0011: begin         // 3
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX3;
                   end
          4'b0100: begin         // 4
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX4;
                   end
          4'b0101: begin         // 5
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX5;
                   end
          4'b0110: begin         // 6
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX6;
                   end
          4'b0111: begin         // 7
                     scan2_prog1_display = HEX0;
                     scan2_prog2_display = HEX7;
                   end
          4'b1000: begin         // 8
```

```
                        scan2_prog1_display = HEX0;
                        scan2_prog2_display = HEX8;
                    end
                4'b1001: begin          // 9
                        scan2_prog1_display = HEX0;
                        scan2_prog2_display = HEX9;
                    end
                4'b1010: begin          // 10
                        scan2_prog1_display = HEX1;
                        scan2_prog2_display = HEX0;
                    end
                default: begin
                        scan2_prog1_display = HEXOFF;
                        scan2_prog2_display = HEXOFF;
                    end
                endcase
            end
        default: begin
                scan2_state_display = HEXOFF;
                scan2_prog1_display = HEXOFF;
                scan2_prog2_display = HEXOFF;
            end
    endcase
  end
endmodule
```

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, William Li, Jun Park
// Date: 13 Nov 2016
//
// testbench for overall module; tests normal operating conditions
module overall_testbench();
  reg clk, reset;
  reg  initialOn, startTransfer;
  wire [6:0] scan1_prog1_display, scan1_prog2_display, scan2_prog1_display,
      scan2_prog2_display, scan1_state_display, scan2_state_display;
  wire [3:0] scan1_prog, scan2_prog;
  wire [2:0] scan1_state, scan2_state;
  wire scan1_readyToTransfer, scan2_readyToTransfer;

  // set up clock
  parameter CLOCK_PERIOD = 10;
  initial clk = 0;
  always begin
    #(CLOCK_PERIOD/2);
    clk = ~clk;
  end

  overall dut (.scan1_state(scan1_state), .scan1_prog(scan1_prog),
      .scan2_state(scan2_state), .scan2_prog(scan2_prog),
        .scan1_state_display(scan1_state_display),
            .scan1_prog1_display(scan1_prog1_display),
            .scan1_prog2_display(scan1_prog2_display),
            .scan1_readyToTransfer(scan1_readyToTransfer),
        .scan2_state_display(scan2_state_display),
            .scan2_prog1_display(scan2_prog1_display),
            .scan2_prog2_display(scan2_prog2_display),
            .scan2_readyToTransfer(scan2_readyToTransfer),
```

```
      .initialOn(initialOn), .startTransfer(startTransfer), .clk(clk),
         .reset(reset));

// begin simulation
initial begin
  reset <= 1;                 @(posedge clk);
                            @(posedge clk);
  reset <= 0; initialOn <= 0; startTransfer <= 0; @(posedge clk);
                            @(posedge clk);
        initialOn <= 1;         @(posedge clk);
        initialOn <= 0;         @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
                            @(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
startTransfer <= 1; @(posedge clk);
           @(posedge clk);
           @(posedge clk);
startTransfer <= 0; @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
           @(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
@(posedge clk);
```

```
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);
                                    @(posedge clk);

    $finish;
  end

  // gtkwave filedump
  initial begin
    $dumpfile("overall.vcd");
    $dumpvars;
  end
endmodule
```

6.1.3 iverilog & gtkwave waveforms

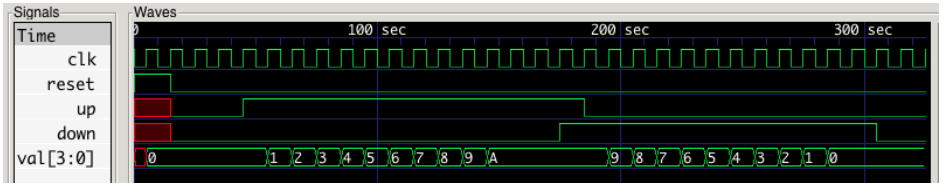

Figure 8: counter module waveform

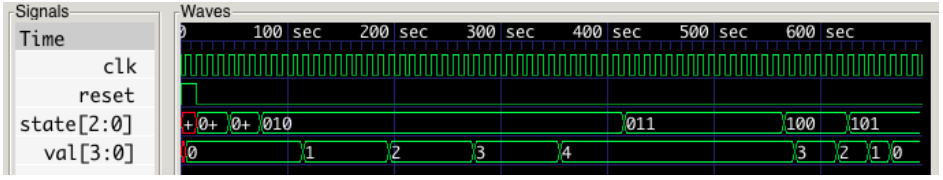

Figure 9: counterCtrl module waveform

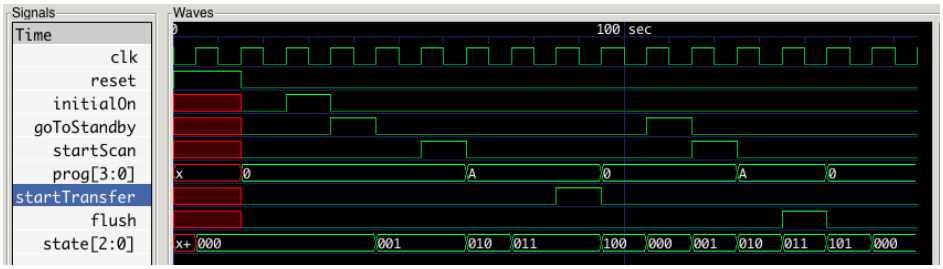

Figure 10: scannerState module waveform

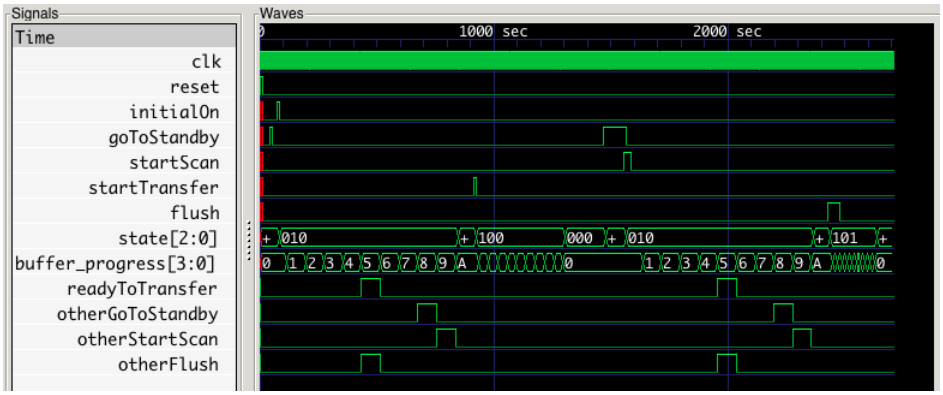

Figure 11: scanner module waveform

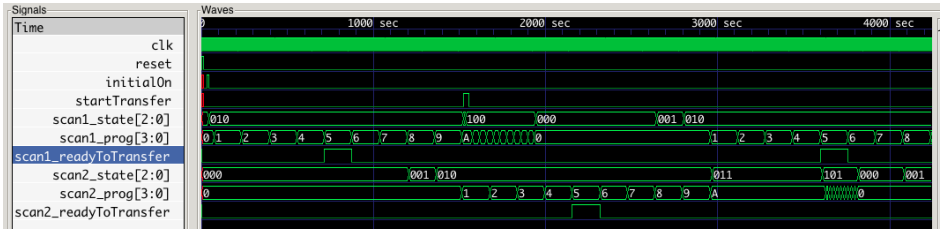

Figure 12: overall module waveform

6.1.4 Signal Tap II data

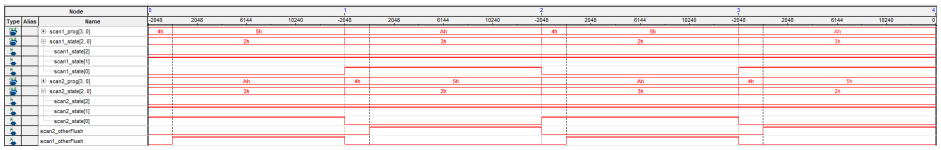

Figure 13: SignalTap II data for flushing the scanners

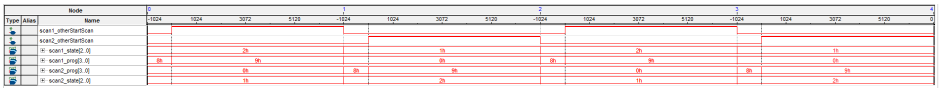

Figure 14: SignalTap II data for starting scanning

6.2 C Program

```c
/* Jun Park, Dawn Liang, William Li
  EE371 Lab3 Propagation Delay Calculator
  Displays total propagation delay caused
  by trace lengths and logic devices.
*/
#include <stdio.h>
#include <stdlib.h>

int main() {
   printf("Propagation Delay Calculator\n");
   printf("Definitions:\n");
   printf("There is 16 ps of delay per 0.1 inches of trace.\n");
   printf("There is 0.1 inches of trace between logic devices.\n");
   printf("Each logic device has a 5 ns delay or 5000 ps delay.\n");
   printf("\n");
   printf("Number of logic devices in the signal path?");
   char *p, s[100];
   int gates;
   while (fgets(s, sizeof(s), stdin)) { // re-prompt until non-negative integer
      gates = strtol(s, &p, 10);
      if (p == s || *p != '\n') {
         printf("Please enter a non-negative integer1: ");
      } else if (gates < 0) {
         printf("Please enter a non-negative integer2: ");
      } else {
         break;
      }
   }
   int totalDelay;
   if (gates == 0) {
      totalDelay = 0;
   } else {
      totalDelay = gates*5000 + (gates-1)*18;
   }
   printf("\n");
   printf("Total Propgation Delay: %dps\n", totalDelay);
   printf("Or equivalent to : %.3lfns\n", (totalDelay/1000.0));
   return 0;
}
```

```c
/* Jun Park, Dawn Liang, William Li
  EE371 Lab3 Temperature Calculator
  Displays Fahrenheit, Celsius, and Kelvin
*/
#include <stdio.h>
int main() {
   char scale;
   while (scale != 'f' || scale != 'F' || scale != 'c' || scale != 'C' || scale !=
       'k' || scale != 'K') {
      printf("Current temperature scale(C, F, or K)? ");
      scanf(" %c", &scale);
      if (scale == 'f' || scale == 'F' || scale == 'c' || scale == 'C' || scale ==
          'k' || scale == 'K') {
         break;
      } else {
```

```
            printf("Scale you entered was not a valid scale. Try again.\n");
        }
    }
    double temperature;
    printf("Current Temperature? ");
    while(scanf("%lf", &temperature) != 1) {
        scanf("%*s"); //ignore the non-double value
        printf("Temperature you entered was not valid. Try again.\n");
        printf("Temperature?");
        if(scanf("%lf", &temperature) == 1) {
            break;
        }
    }

    double tempC;
    double tempF;
    double tempK;
    if (scale == 'f' || scale == 'F') { //Converts F to C and K
        tempF = temperature;
        tempC = (tempF-32.0) * (5.0/9.0);
        tempK = tempC + 273.15;
    } else if (scale == 'c' || scale == 'C') { //Converts C to F and K
        tempC = temperature;
        tempK = tempC + 273.15;
        tempF = tempC*(9.0/5.0) + 32;
    } else { //(scale == 'k' || scale == 'K') //Converts K to F and C
        tempK = temperature;
        tempC = tempK - 273.15;
        tempF = tempC*(9.0/5.0) + 32;
    }

    char outScale;
    while (outScale != 'f' || outScale != 'F' || outScale != 'c' || outScale != 'C' ||
        outScale != 'k' || outScale != 'K') {
        printf("Convert to which temperature scale(C, F, or K)? ");
        scanf(" %c", &outScale);
        if (outScale== 'f' || outScale == 'F' || outScale == 'c' || outScale == 'C' ||
            outScale == 'k' || outScale == 'K') {
            break;
        } else {
            printf("Scale you entered was not a valid scale. Try again.\n");
        }
    }

    if (outScale == 'f' || outScale == 'F') {
        printf("Temperature in Fahrenheit: %.2f degrees\n", tempF);
    } else if (outScale == 'c' || outScale == 'C') {
        printf("Temperature in Celsius : %.2f degrees\n", tempC);
    } else {
        printf("Temperature in Kelvin : %.2f degrees\n", tempK);
    }
    return 0;
}
```

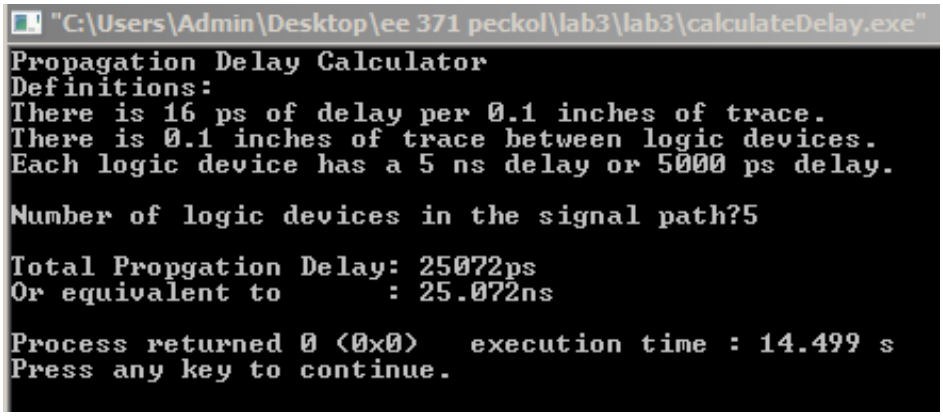

Figure 15: Normal output of calculateDelay program

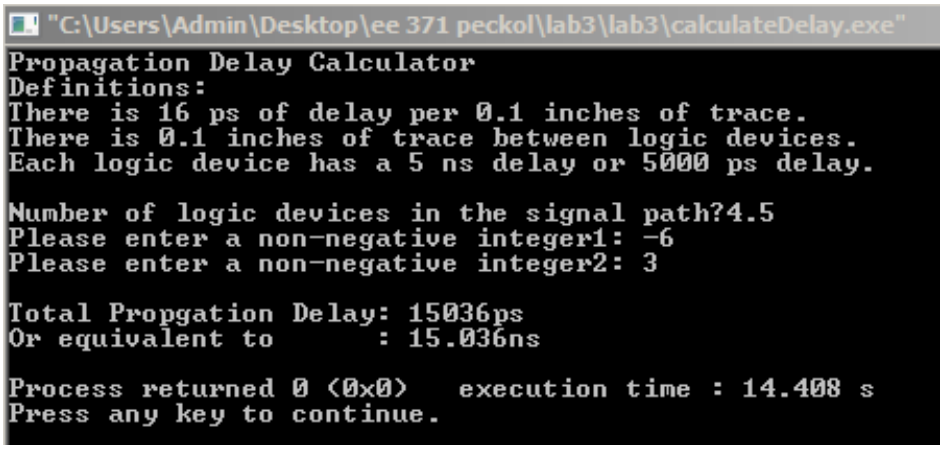

Figure 16: Error handling output of calculateDelay program

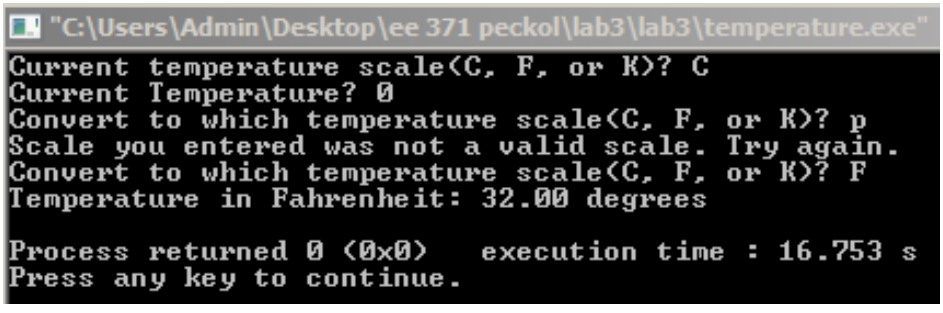

Figure 17: Output of temperature program: C to F

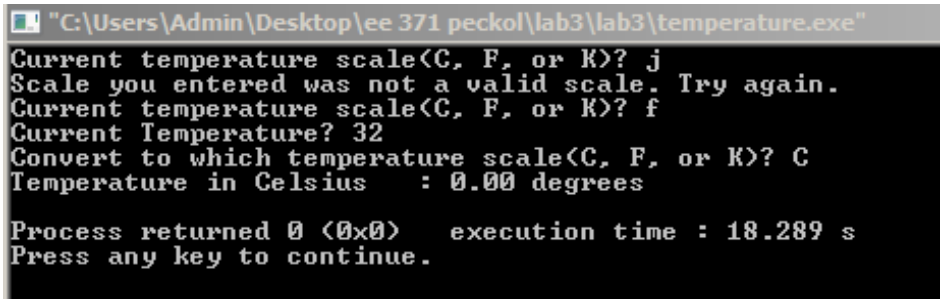

Figure 18: Output of temperature program: F to C

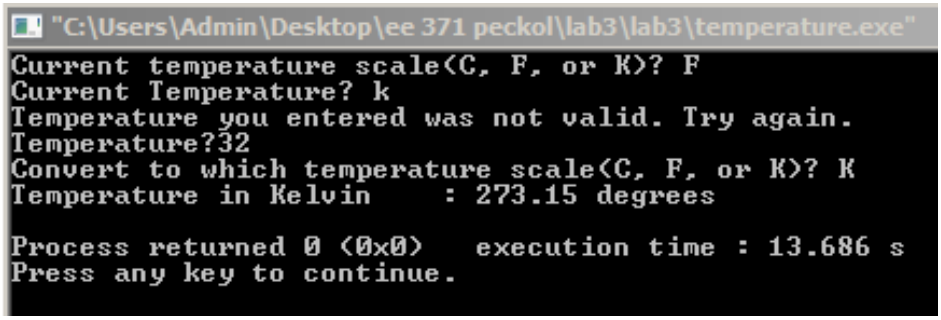

Figure 19: Output of temperature program: F to K

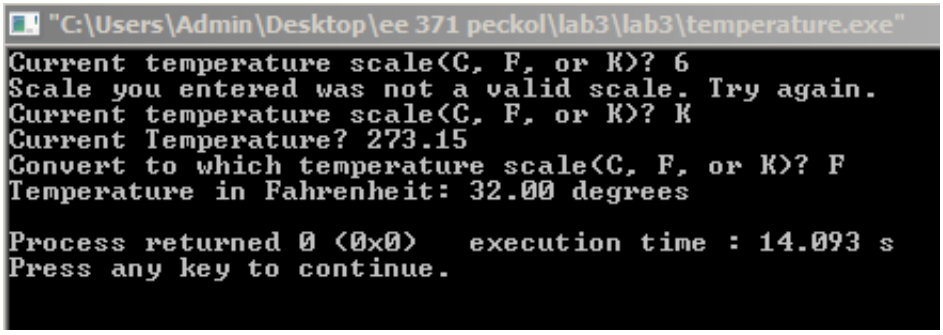

Figure 20: Output of temperature program: K to F