EE 371 Autumn 2016 - Lab 2

William Li, Dawn Liang, Jun Park

November 12, 2016

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

Contents

1 ABSTRACT

2 INTRODUCTION

3 DISCUSSION

3.1 Design

3.1.1 Design Specification

3.1.2 Design Procedure

3.1.3 System Description

3.1.4 Software Implementation

3.1.5 Hardware Implementation

3.2 Test

3.2.1 Test Plan

3.2.2 Test Specification

3.2.3 Test Cases

4 RESULTS

4.1 Analysis of Errors

5 SUMMARY & CONCLUSION

6 APPENDIX

6.1 Two-scanner system

6.1.1 Verilog code

```
// EE371 Lab3 Autumn 2016
// Authors: Dawn Liang, Jun Park, William Li
// Date:
//
// counter that counts every clock edge up/down to a min/max (0/10)
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
```

```
`include "counter.v"
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
//
// counter that counts up/down at different speeds depending on the state input

`include "clock_divider.v"
`include "counter.v"
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[2];
              up = 1; down = 0;
            end
      idle:  begin
              final_clock = divided_clocks[0];
              up = 0; down = 0;
            end
      xferring: begin                    // count down at 2x speed
              final_clock = divided_clocks[1];
              up = 0; down = 1;
            end
      flushing: begin                    // count down at 4x 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
```

```
`include "counterCtrl.v"
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 = 0;
  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
        active = 1;
        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;
    end else begin
      ps <= ns;
    end
  end
endmodule
```

```
`include "scannerState.v"
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
```

```
`include "scannerState.v"
`include "counterCtrl.v"
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;

  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 == 3'b010);
  assign otherGoToStandby = (buffer_progress == 8 && state == 3'b010);
  assign otherStartScan = (buffer_progress == 9 && state == 3'b010);
  assign otherFlush = (buffer_progress == 5 && state == 3'b010);
endmodule
```

```
`include "scanner.v"
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);
                    @(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);
                                                             @(posedge clk);
                                                             @(posedge clk);
                                                             @(posedge clk);
    $finish;
  end

  // gtkwave filedump
  initial begin
```

```
    $dumpfile("scanner.vcd");
    $dumpvars;
  end
endmodule
```

```
`include "scanner.v"
module overall(scan1_state, scan1_prog, scan2_state, scan2_prog,
        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 2:0
             scan2_state_display, scan2_prog1_display, scan2_prog2_display; // HEX 5:3
  output scan1_readyToTransfer, scan2_readyToTransfer; // LED[0], LED[1]

  // 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'b1111100; // 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));
  // reg final_clock = divided_clocks[22];

  // connecting wires
  output [3:0] scan1_prog, scan2_prog;
  output [2:0] scan1_state, scan2_state;
  wire  scan1_otherGoToStandby, scan1_otherStartScan, scan1_otherFlush,
     scan2_otherGoToStandby, scan2_otherStartScan, scan2_otherFlush;

// 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(clk),
      .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(clk),
      .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
```

```
'include "overall.v"
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);
                                @(posedge clk);

    $finish;
  end

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

6.1.2 iverilog & gtkwave waveforms

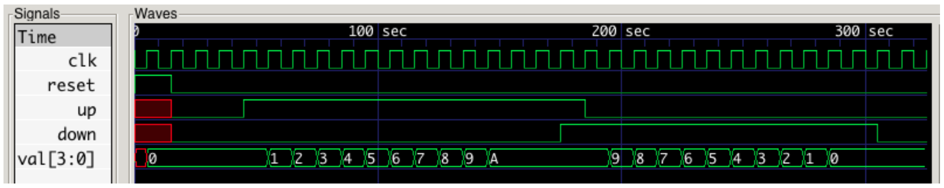


Figure 1: counter module waveform

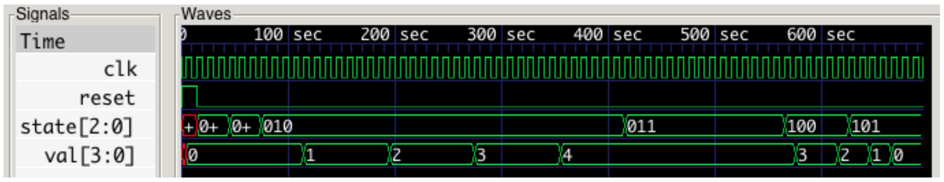


Figure 2: counterCtrl module waveform

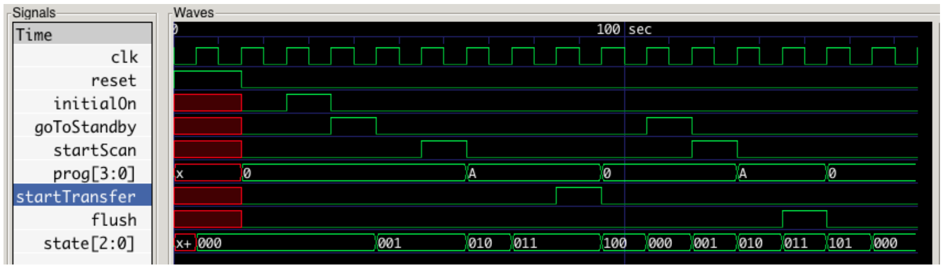


Figure 3: scannerState module waveform

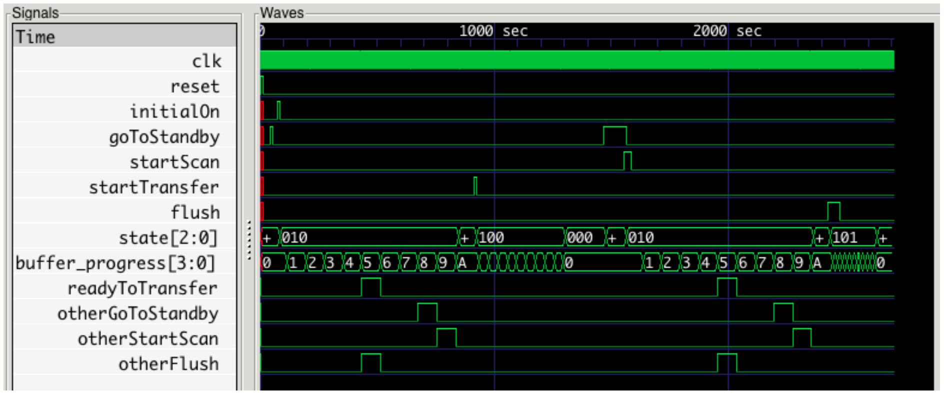


Figure 4: scanner module waveform

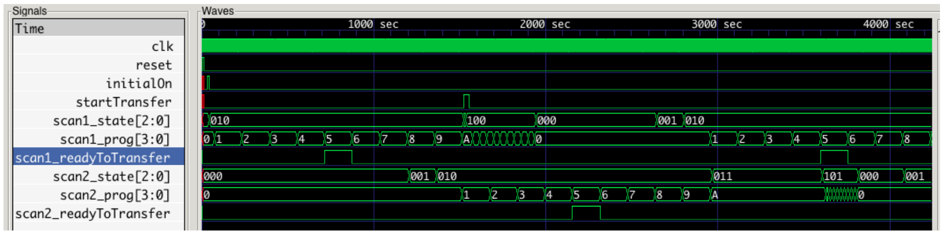


Figure 5: overall module waveform

6.1.3 Signal Tap II data

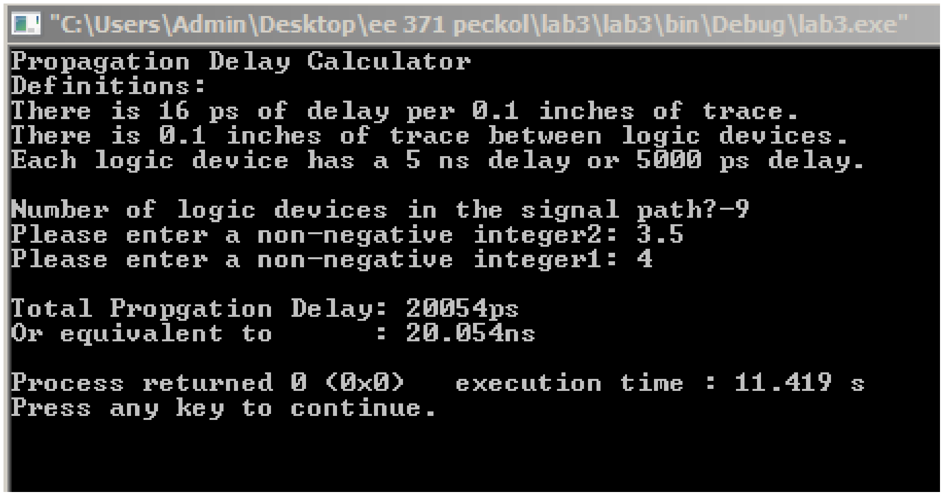
6.2 C Program

```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;
}
```

```
/* 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("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("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;
      }
   }
   printf("\n");
   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;
    }
    printf("Temperature in Fahrenheit: %.2f degrees\n", tempF);
    printf("Temperature in Celsius : %.2f degrees\n", tempC);
    printf("Temperature in Kelvin : %.2f degrees\n", tempK);
    return 0;
}
```



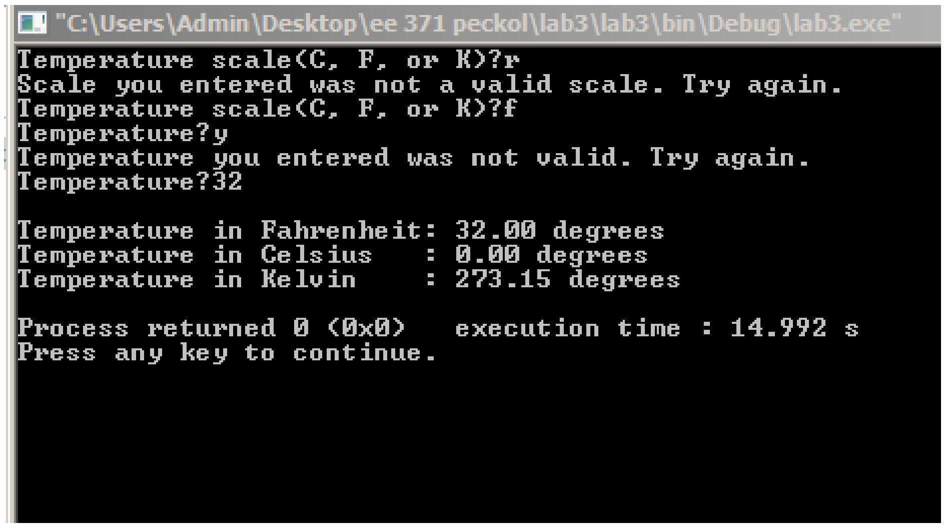
```
"C:\Users\Admin\Desktop\ee 371 peckol\lab3\lab3\bin\Debug\lab3.exe"
Propagation Delay Calculator
Definitions:
There is 16 ps of delay per 0.1 inches of trace.
There is 0.1 inches of trace between logic devices.
Each logic device has a 5 ns delay or 5000 ps delay.

Number of logic devices in the signal path?-9
Please enter a non-negative integer2: 3.5
Please enter a non-negative integer1: 4

Total Propgation Delay: 20054ps
Or equivalent to      : 20.054ns

Process returned 0 (0x0)   execution time : 11.419 s
Press any key to continue.
```

Figure 6: Output of calculateDelay program



```
"C:\Users\Admin\Desktop\ee 371 peckol\lab3\lab3\bin\Debug\lab3.exe"
Temperature scale(C, F, or K)?r
Scale you entered was not a valid scale. Try again.
Temperature scale(C, F, or K)?f
Temperature?y
Temperature you entered was not valid. Try again.
Temperature?32

Temperature in Fahrenheit: 32.00 degrees
Temperature in Celsius   : 0.00 degrees
Temperature in Kelvin    : 273.15 degrees

Process returned 0 (0x0)   execution time : 14.992 s
Press any key to continue.
```

Figure 7: Output of temperature program