TFE4152 Design av integrerte kretser

Semester project: Design of the support circuit for a digital camera

J. L. Selnes T. Nordgård-Hansen

4th November 2019

Contents

---

1 Abstract

This file contains the summary of the report.

I have, at the moment, no idea what is should contain, but it is here at the moment.

To populate the references, the SystemVerilog code was simulated by iverilog [1].

I can also cite the AIMSpice page [2].

2 Introduction

3 Theory

4 Analog design

5 Digital design

6 Simulations

asdf

7 Conclusion

8 References

References

[1] S. Williams. (1998). Icarus verilog, [Online]. Available: `http://iverilog.icarus.com/` (visited on 04/11/2019).

[2] T. Ytterdal. (2019). Aim-spice, [Online]. Available: `http://aimspice.com/` (visited on 04/11/2019).

Appendices

A Spice code

B Verilog code

Listing 1: Main module for camera control testbench

```
/*
 This is the testbench for camera_fsm, the logic that controlls the
    camera.
 */

`define _no_testbench_

`timescale 1ns / 1ps

`include "camera_fsm.v"


module camera_fsm_tb;
   logic clk, init, inc, dec, reset;
   logic expose, erase, nre1, nre2, adc;

   camera_fsm test_camera(clk, init, inc, dec, reset, expose, erase,
      nre1, nre2, adc);

   always @(*)
     #1 clk <= !clk;

   initial begin
      $dumpfile("outfiles/out_camera_fsm_tb.vcd");
      $dumpvars();

      {clk, init, inc, dec, reset} = 5'b00001;
      #2 reset = 0;

      #5 inc = 1;
      #2 dec = 1;
      #30 inc = 0;
      #4 dec = 0;
      #2 init = 1;
      #4 init = 0;

      #30 inc = 1;
      #4 inc = 0;

      #28 reset = 1;
      #2 reset = 0;

      #4 dec = 1;
      #60 dec = 0;

      init = 1;
      #1 init = 0;

      #1 dec = 1;

```

```
49        #38 $finish;
50    end
51
52  endmodule // camera_fsm_tb
```

Listing 2: Main module for camera control

```
1   /*
2    This file contains the controll logic of the FSM of the camera.
3    It depends on the fcd_reg, exp_reg and read_sequencer to function.
4    */
5
6   // `define _no_testbench_
7
8   `ifndef _fsm_logic_v_31_
9    `define _fsm_logic_v_31_
10
11   `include "exp_reg.v"
12   `include "fcd_reg.v"
13   `include "readout_seq.v"
14
15
16  module camera_fsm(input  logic clk,
17                    input logic  init, exponer_inc, exponer_dec, reset,
18                    output logic expose, erase, nre_1, nre_2, adc_enable
                          );
19
20      typedef enum                logic [1:0]
21                                  {idle, exposing, readout, illegal}
                                        statetypes;
22      statetypes current_state, next_state;
23
24      logic                       readout_enable, readout_reset,
           readout_finished;
25
26      logic                       fcd_enable, fcd_dset, fcd_finished;
27      logic [4:0]                 fcd_data;
28
29      logic                       exp_reset, exp_inc, exp_dec;
30
31
32      exp_reg exposure_reg(clk, exp_reset, exp_inc, exp_dec, fcd_data);
33      fcd_reg countdown_reg(clk, fcd_enable, fcd_dset, fcd_data,
           fcd_finished);
34      readout_seq readout_sequencer(clk, readout_enable, readout_reset,
           readout_finished, nre_1, nre_2, adc_enable);
35
36
37      assign exp_inc = (current_state == idle) ? exponer_inc : 0;
38      assign exp_dec = (current_state == idle) ? exponer_dec : 0;
39
40      always @(posedge clk)
41        case (next_state)
```

```
42          idle: begin
43             {expose, erase, readout_enable, readout_reset, fcd_enable,
                  fcd_dset} <= 6'b010101;
44             current_state <= next_state;
45          end
46          exposing: begin
47             {expose, erase, readout_enable, readout_reset, fcd_enable,
                  fcd_dset} <= 6'b100110;
48             current_state <= next_state;
49          end
50          readout: begin
51             {expose, erase, readout_enable, readout_reset, fcd_enable,
                  fcd_dset} <= 6'b001001;
52             current_state <= next_state;
53          end
54          default: current_state <= next_state;
55
56        endcase // case (next_state)
57
58
59     always @(*) begin
60        if (reset)
61          current_state = illegal;
62        else case (current_state)
63               illegal: begin
64                  exp_reset = 1;
65                  readout_reset = 1;
66                  exp_reset = 0;
67                  readout_reset = 0;
68                  next_state = idle;
69               end
70               idle: if (init) next_state = exposing;
71               exposing: if (fcd_finished) next_state = readout;
72               readout: if (readout_finished) next_state = idle;
73               default: next_state = idle;
74             endcase // case (state)
75     end // always @ (*)
76
77     initial
78       current_state = illegal;
79
80  endmodule // camera_fsm
81
82
83  `endif
```

Listing 3: Exposure register

```
1  /*
2   This file contains a simple module for counting the exposure time,
3   it runs from 2 to 30 (requiering 5 bits)
4   */
5
```

```
6  `ifndef _exp_reg_v_25_
7   `define _exp_reg_v_25_
8
9
10 module exp_reg(input  logic clk, reset, inc, dec,
11                output logic [4:0] q);
12
13    always @(posedge clk)
14      if (inc & (q < 30) & !reset)
15        q <= q + 1;
16      else if (dec & (q > 2) & !inc & !reset)
17        q <= q - 1;
18
19    always @(*)
20      if (reset | q > 30 | q < 2)
21        q <= 5'd15;
22
23    initial
24      q = 5'd15;
25
26 endmodule // exp_reg
27
28
29  `ifndef _no_testbench_
30
31
32 module exp_reg_tb;
33    logic clk, reset, inc, dec;
34    logic [4:0] q;
35
36    exp_reg testreg(clk, reset, inc, dec, q);
37
38    initial begin
39       $dumpfile("outfiles/out_exp_reg_tb.vcd");
40       $dumpvars();
41
42       #1 {clk, reset, inc, dec} = 4'b1100;
43       #1 {clk, reset} = 2'b10;
44
45       for (int i = 1; i <= 20; i = i+1) begin
46          #1 {clk, inc, dec} = 3'b111;
47          #1 {clk, inc, dec} = 3'b011;
48       end
49
50       for (int i = 1; i <= 40; i = i+1) begin
51          #1 {clk, inc, dec} = 3'b101;
52          #1 {clk, inc, dec} = 3'b001;
53       end
54
55       #1 reset = 1;
56       #1 reset = 0;
57
58       #1 {clk, reset, inc, dec} = 4'b1101;
```

```
59        #1 {clk, reset} = 2'b00;
60        #1 clk = 1;
61        #1 clk = 0;
62
63        #1 $finish;
64     end
65
66  endmodule // exp_reg_tb
67
68
69   `endif //  `ifndef _no_testbench_
70  `endif //  `ifndef _exp_reg_v_25_
```

Listing 4: Counter for exposure time

```
1   /*
2    This file contains the register used to count down different values
         in the camera.
3    It has 5 bits, can take data inn and counts down on rising clock edge
          when enabled.
4    It has an output for when the internal data is 0.
5    Data set has priority over enable.
6    */
7
8   `ifndef _fcd_reg_v_28_
9    `define _fcd_reg_v_28_
10
11
12  module fcd_reg(input logic clk, enable, dset,
13                 input logic [4:0] dread_data,
14                 output logic      finished);
15
16     logic [4:0]                   data_int;
17
18     always @(posedge clk)
19       if (dset)
20         data_int <= dread_data;
21       else if (enable & data_int > 'b0)
22         data_int <= data_int - 1;
23
24     assign finished = (data_int == 5'd1) ? 1 : 0;
25
26  endmodule // fcd_reg
27
28
29   `ifndef _no_testbench_
30
31
32  module fcd_reg_tb;
33     logic clk, enable, dset, finished;
34     logic [4:0] data_in;
35
36     fcd_reg testreg(clk, enable, dset, data_in, finished);
```

```
37
38      initial begin
39          $dumpfile("outfiles/out_fcd_reg_tb.vcd");
40          $dumpvars();
41
42          #1 {clk, enable, dset, data_in} = 7'b0000111;
43          #1 {clk, dset} = 2'b11;
44          #1 clk = 0;
45          #1 {clk, enable, dset} = 3'b110;
46
47          for (int i = 1; i <= 21; i = i + 1) begin
48              #1 clk = !clk;
49          end
50
51          #1 {clk, enable} = 2'b10;
52          #1 clk = 0;
53          #1 clk = 1;
54
55          for (int i = 1; i <= 5; i = i + 1) begin
56              #1 clk = !clk;
57          end
58
59          #1 {clk, enable, dset} = 3'b111;
60
61          for (int i = 1; i <= 5; i = i + 1) begin
62              #1 clk = !clk;
63          end
64
65          #1 $finish;
66      end // initial begin
67
68  endmodule // fcd_reg_tb
69
70
71
72
73
74   `endif //  `ifndef _no_testbench_
75  `endif //  `ifndef _fcd_reg_v_28_
```

Listing 5: Readout sequencer

```
1  /*
2   This file contains a sequencer for the readout of the 4 pixel camera.
3   It can be enabled and reset.
4   Note that the enable does not imply a reset.
5   */
6
7  `ifndef _readout_seq_v_31_
8   `define _readout_seq_v_31_
9
10
11 module readout_seq(input logic clk,
```

```
12                    input logic  enable, reset,
13                    output logic finished,
14                    output logic nre1, nre2, adc);
15
16    logic [3:0]                  step;
17
18    assign finished = (step >= 8) ? 1 : 0;
19
20    always @(posedge clk)
21       if (!finished && enable && !reset) step <= step + 1;
22
23    always @(*) begin
24       if (reset)
25         step <= 0;
26
27       case (step)
28         0: {nre1, nre2, adc} = 3'b000;
29         1: {nre1, nre2, adc} = 3'b100;
30         2: {nre1, nre2, adc} = 3'b101;
31         3: {nre1, nre2, adc} = 3'b100;
32         4: {nre1, nre2, adc} = 3'b000;
33         5: {nre1, nre2, adc} = 3'b010;
34         6: {nre1, nre2, adc} = 3'b011;
35         7: {nre1, nre2, adc} = 3'b010;
36         8: {nre1, nre2, adc} = 3'b000;
37         default: {nre1, nre2, adc} = 3'b000;
38       endcase // case (step)
39    end // always @ (*)
40
41    endmodule // readout_seq
42
43
44  `ifndef _no_testbench_
45
46
47  module readout_seq_tb;
48    logic clk, enable, reset, finished, nre1, nre2, adc;
49
50    readout_seq test_seq(clk, enable, reset, finished, nre1, nre2, adc)
        ;
51
52    always @(*)
53      #1 clk <= !clk;
54
55    initial begin
56       $dumpfile("outfiles/out_readout_seq_tb.vcd");
57       $dumpvars();
58
59       {clk, enable, reset} = 3'b000;
60       #5 reset = 1;
61       #2 reset = 0;
62
63       #4 enable = 1;
```

```
64          #20 enable = 0;
65          #4 enable = 1;
66          #1 reset = 1; #1
67          #2 reset = 0;
68          #4 enable = 0;
69
70          #4 $finish;
71      end // initial begin
72
73  endmodule // readout_seq_tb
74
75
76   `endif
77  `endif //  `ifndef _readout_seq_v_31_
```