

TFE4152 Design of Integrated Circuits

NTNU



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

J. L. Selnes T. Nordgård-Hansen

7th November 2019



Contents

1	Abstract	3
2	Introduction	3
3	Theory	4
3.1	One digital pixel	4
3.2	Leakage through transistors	4
3.3	Conceptual workings of a camera controller	5
4	Analog design	6
5	Digital design	7
6	Simulations	8
7	Conclusion	9
8	References	10
	Appendices	11
A	Schematics	11
B	Spice code	13
C	Verilog code	18

1 Abstract

This report describes a design of a 4 pixel digital camera. It includes an analog schematics for all the pixels as well as a digital design for the control unit. Coupled with a pulse shaper and $\sqrt{\text{number of pixels}}$ number of ADCs this design can be adapted to an arbitrary quadratic digital camera.

The design was tested with AIM-Spice [1] and Icarus verilog [2] as described in section 6

2 Introduction

There are several components that go into the process of creating a digital camera, among them are the exposure circuits for each circuit, a system to read out the values in turn and a control system for the whole process.

This project aims to give an extensive design example of these systems applied to a $2 \cdot 2$ pixel camera. It does not include any details on the manufacturing process, the analog to digital converters or the long term storage of the images, but focuses instead on the taking on the picture from user input to serialized voltage levels on an analog 2 bit buss.

For reference the analog design is shown as classic schematics in Appendix A and as SPICE net lists in Appendix B, the digital design is defined in SystemVerilog 2012 in Appendix C. In addition, all files related to the project are available on GitHub [3].

3 Theory

3.1 One digital pixel

Each pixel in the camera is constructed as shown in figure 1. The photo diodes detecting the actual light does, in many ways, act as a current source dependent on the light on it, when a picture is taken this current is let through M1 and used to charge CS. Before each picture is taken, M2 is opened to reset the voltage stored over CS.

It is important that M1 and M2 are not let on simultaneously for extended periods of time as this results in a short circuit from VDD to VSS thorough PD1. While the photo diode limits the current, this still might lead to excessive power usage and subsequent heating issues over time.

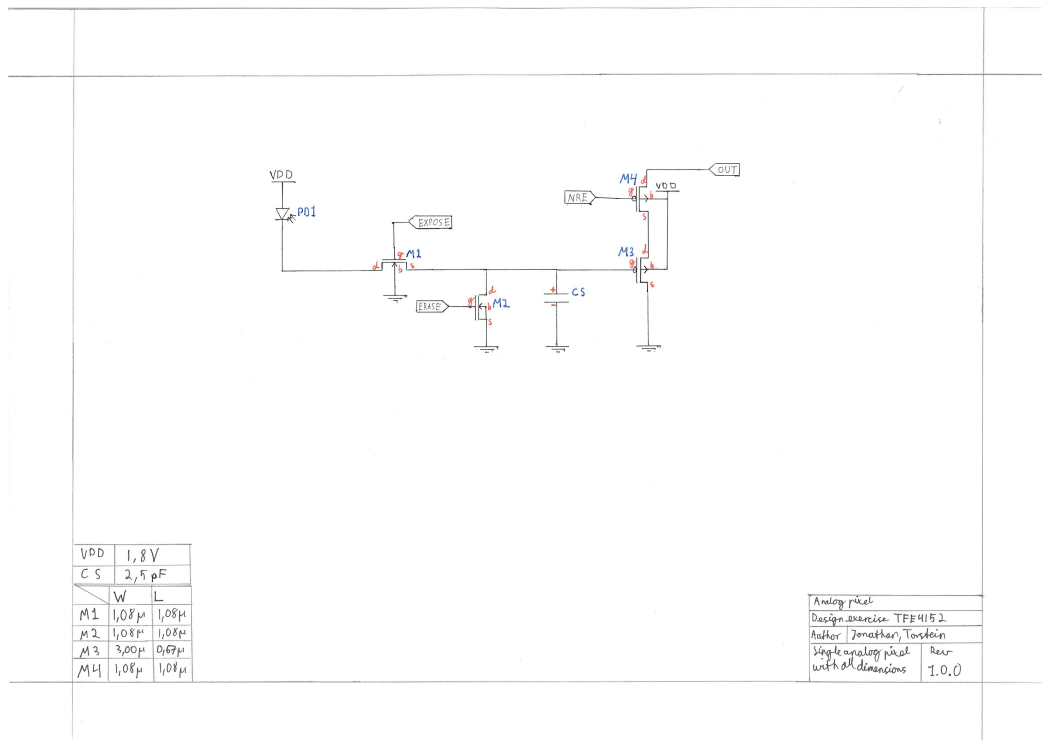


Figure 1: Schematic of one pixel with readout circuit, figure also exists in appendix A

M3 is used to convert the voltage stored over CS into a variable resistance between N3 and VSS for a nondestructive readout of the pixel, M4 functions as a simple switch to isolate the pixel from OUT to free the wire when other pixels in the camera are using it.

3.2 Leakage through transistors

As described in the analog book [4] stuff can happen.

3.3 Conceptual workings of a camera controller

This might work...

4 Analog design

The design exist in appendix B.

5 Digital design

The design exists in appendix C.

6 Simulations

Much simulation, much gud.

7 Conclusion

Such conclusion, very concluded.

8 References

References

- [1] T. Ytterdal. (2019). Aim-spice, [Online]. Available: <http://aimspice.com/> (visited on 04/11/2019).
- [2] S. Williams. (1998). Icarus verilog, [Online]. Available: <http://iverilog.icarus.com/> (visited on 04/11/2019).
- [3] J. L. Selnes and T. Nordgård-Hansen. (2019). Tfe4152-project, [Online]. Available: <https://github.com/torsteinnh/TFE4152-project> (visited on 19/11/2019).
- [4] T. C. Carusone, D. A. Johns and K. W. Martin, *Analog Integrated Circuit Design, 2nd Edition*. Wiley, Nov. 2011, ISBN: 978-11-1821-373-5.

Appendices

A Schematics

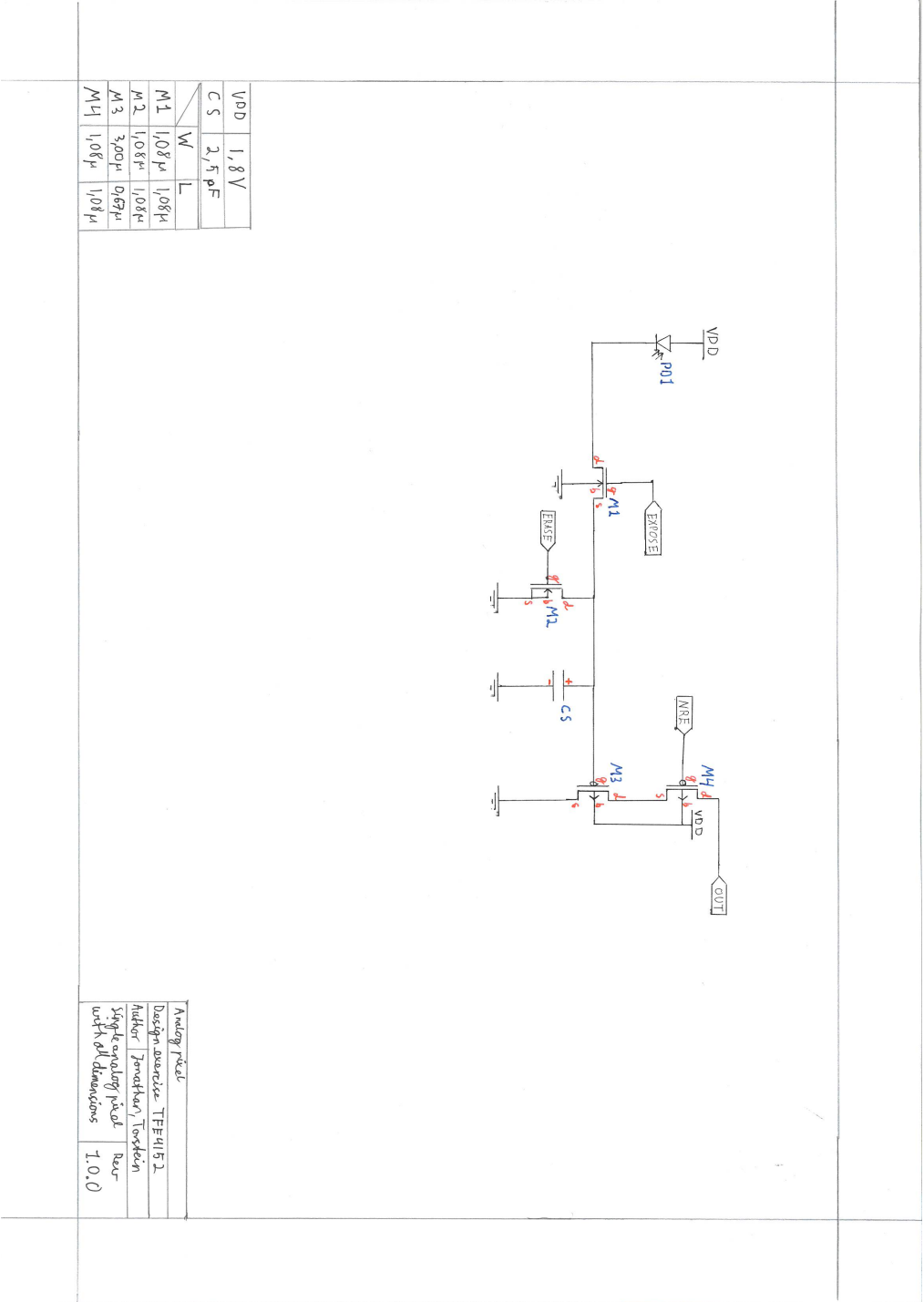


Figure 2: Analog schematic of one pixel

B Spice code

Listing 1: Main simulation of analog pixels

```
* A simulation of all four pixels

.include parameters.cir
.include components.cir

xPixel11 1 0 EXPOSE ERASE NRE_R1 OUT_C1 N211 pixel
xPixel12 1 0 EXPOSE ERASE NRE_R1 OUT_C2 N212 pixel
xPixel21 1 0 EXPOSE ERASE NRE_R2 OUT_C1 N221 pixel
xPixel22 1 0 EXPOSE ERASE NRE_R2 OUT_C2 N222 pixel

xcurrentAmp1 1 0 OUT_C1 currentamp
xcurrentAmp2 1 0 OUT_C2 currentamp

.tran {PERIOD / 100000} PERIOD
.plot tran v(OUT_C1) v(ERASE) v(EXPOSE) v(NRE_R1) v(NRE_R2) v(
    OUT_C2) v(N211)
```

Listing 2: Components in the camera

```
* A file with subcircuits for the camera

.include models/p18_cmos_models.inc
.include models/photo_diode.inc

.subckt pixel VDD GND EXPOSE ERASE NRE OUT N2

xphoto VDD N1 PhotoDiode

M1 N1 EXPOSE N2 GND NMOS W=M1W L=M1L
M2 N2 ERASE GND GND NMOS W=M2W L=M2L
CS N2 GND CSval

M3 N3 N2 GND VDD PMOS W=M3W L=M3L
M4 OUT NRE N3 VDD PMOS W=M4W L=M4L

.ends
```

```
.subckt currentamp VDD GND IO

MC VDD IO IO VDD PMOS W=MOSW L=MCL
CC IO GND CCval

.ends
```

Listing 3: Parameters for the camera

```
* This file contains the parameters and standard components for
  all analog circuits in the project.

* Include models

* Test parameters
.param Ipd_1 = 750p ! Photodiode current, range [50 pA, 750 pA]
.param EXPOSURETIME = 4m ! Exposure time, range [2 ms, 30 ms]

* Derived and fixed test parameters
.param VDD = 1.8 ! Supply voltage
.param TRF = {EXPOSURETIME/100} ! Risetime and falltime of
  EXPOSURE and ERASE signals
.param PW = {EXPOSURETIME} ! Pulsethickness of EXPOSURE and ERASE
  signals
.param FS = 1k; ! Sampling clock frequency
.param CLK_PERIOD = {1/FS} ! Sampling clock period
.param READ_TIME = {CLK_PERIOD}
.param EXPOSE_DLY = {CLK_PERIOD} ! Delay for EXPOSE signal
.param NRE_R1_DLY = {2*CLK_PERIOD + EXPOSURETIME} ! Delay for
  NRE_R1 signal
.param NRE_R2_DLY = {CLK_PERIOD + NRE_R1_DLY + READ_TIME} ! Delay
  for NRE_R2 signal
.param ERASE_DLY = {CLK_PERIOD + NRE_R2_DLY + READ_TIME} ! Delay
  for ERASE signal
.param PERIOD = {ERASE_DLY + EXPOSURETIME} ! Period for testbench
  sources

* Permanent test sources
VDD 1 0 dc VDD
VEXPOSE EXPOSE 0 dc 0 pulse(0 VDD EXPOSE_DLY TRF TRF EXPOSURETIME
  PERIOD)
VERASE ERASE 0 dc 0 pulse(0 VDD ERASE_DLY TRF TRF CLK_PERIOD
```

```

PERIOD)
VNRE_R1 NRE_R1 0 dc 0 pulse(VDD 0 NRE_R1_DLY TRF TRF READ_TIME
PERIOD)
VNRE_R2 NRE_R2 0 dc 0 pulse(VDD 0 NRE_R2_DLY TRF TRF READ_TIME
PERIOD)

```

```

* Parameters for photo cell, L [0.36 , 1.08]u W [1.08 , 5.04]u
.param M1W = 1.08u
.param M1L = 1.08u
.param M2W = 1.08u
.param M2L = 1.08u
.param M3W = 3u
.param M3L = 0.67u
.param M4W = 1.08u
.param M4L = 1.08u
.param M5W = 5.04u
.param M5L = 0.36u

.param CSval = 2.5p
.param CCval = 3p

```

Listing 4: MOSFET models part 1

```

.param proc_delta = 0.95
.param vt_shift = 0.1
.include models/p18_model_card.inc

```

Listing 5: MOSFET models part 2

```

* p18 model card
.MODEL NMOS NMOS
+ VERSION = 3.1
+ LEVEL = 49 NOIMOD = 1 TNOM = 2.70E+01
+ TOX = '4.1E-9/proc_delta' XJ = 1.00E-07 NCH = 2.33E+17
+ VTH0 = '0.36+vt_shift' K1 = 5.84E-01 K2 = 4.14E-03
+ K3 = 1.01E-03 K3B = 2.20E+00 W0 = 1.00E-07
+ NLX = 1.81E-07 DVT0W = 0.00E+00 DVT1W = 0.00E+00
+ DVT2W = 0.00E+00 DVT0 = 1.73E+00 DVT1 = 4.38E-01
+ DVT2 = -3.70E-04 U0 = '260*proc_delta*proc_delta' UA = -1.38
E-09
+ UB = 2.26E-18 UC = 5.46E-11 VSAT = 1.03E+05
+ A0 = 1.92E+00 AGS = 4.20E-01 B0 = -1.52E-09
+ B1 = -9.92E-08 KETA = -7.16E-03 A1 = 6.61E-04
+ A2 = 8.89E-01 RDSW = 1.12E+02 PRWG = 4.92E-01

```

```

+ PRWB = -2.02E-01 WR = 1.00E+00 WINT = 7.12E-09
+ LINT = 1.12E-08 XL = -2.00E-08 XW = -1.00E-08
+ DWG = -3.82E-09 DWB = 8.63E-09 VOFF = -8.82E-02
+ NFACTOR = 2.30E+00 CIT = 0.00E+00 CDSC = 2.40E-04
+ CDSCD = 0.00E+00 CDSCB = 0.00E+00 ETA0 = 3.13E-03
+ ETAB = 1.00E+00 DSUB = 2.25E-02 PCLM = 7.20E-01
+ PDIBLC1 = 2.15E-01 PDIBLC2 = 2.23E-03 PDIBLCB = 1.00E-01
+ DROUT = 8.01E-01 PSCBE1 = 5.44E+08 PSCBE2 = 1.00E-03
+ PVAG = 1.00E-12 DELTA = 1.00E-02 RSH = 6.78E+00
+ MOBMOD = 1.00E+00 PRT = 0.00E+00 UTE = -1.50E+00
+ KT1 = -1.10E-01 KT1L = 0.00E+00 KT2 = 2.19E-02
+ UA1 = 4.28E-09 UB1 = -7.62E-18 UC1 = -5.57E-11
+ AT = 3.30E+04 WL = 0.00E+00 WLN = 1.00E+00
+ WW = 0.00E+00 WWN = 1.00E+00 WWL = 0.00E+00
+ LL = 0.00E+00 LLN = 1.00E+00 LW = 0.00E+00
+ LWN = 1.00E+00 LWL = 0.00E+00 CAPMOD = 2.00E+00
+ XPART = 5.00E-01 CGDO = 6.98E-10 CGSO = 7.03E-10
+ CGBO = 1.00E-12 CJ = '9.8e-4/proc_delta' PB = 7.34E-01
+ MJ = 3.63E-01 CJSW = '2.4e-10/proc_delta' PBSW = 4.71E-01
+ MJSW = 1.00E-01 CJSWG = 3.29E-10 PBSWG = 4.66E-01
+ MJSWG = 1.00E-01 CF = 0.00E+00 PVTH0 = -7.16E-04
+ PRDSW = -6.66E-01 PK2 = 5.92E-04 WKETA = 2.14E-04
+ LKETA = -1.51E-02 PU0 = 3.36E+00 PUA = -1.31E-11
+ PUB = 0.00E+00 PVSAT = 1.25E+03 PETA0 = 1.00E-04
+ PKETA = 6.45E-04 KF = 4.46E-29

```

.MODEL PMOS PMOS

```

+ VERSION = 3.1
+ LEVEL = 49 NOIMOD = 1
+ TNOM = 2.70E+01 TOX = '4.1E-9/proc_delta' XJ = 1.00E-07
+ NCH = 4.12E+17 VTH0 = '-0.39-vt_shift' K1 = 5.50E-01
+ K2 = 3.50E-02 K3 = 0.00E+00 K3B = 1.20E+01
+ W0 = 1.00E-06 NLX = 1.25E-07 DVT0W = 0.00E+00
+ DVT1W = 0.00E+00 DVT2W = 0.00E+00 DVT0 = 5.53E-01
+ DVT1 = 2.46E-01 DVT2 = 1.00E-01 U0 = '110*proc_delta*
    proc_delta'
+ UA = 1.44E-09 UB = 2.29E-21 UC = -1.00E-10
+ VSAT = 1.95E+05 A0 = 1.72E+00 AGS = 3.80E-01
+ B0 = 5.87E-07 B1 = 1.44E-06 KETA = 2.21E-02
+ A1 = 4.66E-01 A2 = 3.00E-01 RDSW = 3.11E+02
+ PRWG = 5.00E-01 PRWB = 1.64E-02 WR = 1.00E+00
+ WINT = 0.00E+00 LINT = 2.00E-08 XL = -2.00E-08
+ XW = -1.00E-08 DWG = -3.49E-08 DWB = 1.22E-09

```



```

+ VOFF = -9.80E-02 NFACTOR = 2.00E+00 CIT = 0.00E+00
+ CDSC = 2.40E-04 CDSCD = 0.00E+00 CDSCB = 0.00E+00
+ ETA0 = 1.12E-03 ETAB = -4.79E-04 DSUB = 1.60E-03
+ PCLM = 1.50E+00 PDIBLC1 = 3.00E-02 PDIBLC2 = -1.01E-05
+ PDIBLCB = 1.00E-01 DROUT = 1.56E-03 PSCBE1 = 4.91E+09
+ PSCBE2 = 1.64E-09 PVAG = 3.48E+00 DELTA = 1.00E-02
+ RSH = 7.69E+00 MOBMOD = 1.00E+00 PRT = 0.00E+00
+ UTE = -1.49E+00 KT1 = -1.09E-01 KT1L = 0.00E+00
+ KT2 = 2.18E-02 UA1 = 4.27E-09 UB1 = -7.68E-18
+ UC1 = -5.57E-11 AT = 3.31E+04 WL = 0.00E+00
+ WLN = 1.00E+00 VW = 0.00E+00 WWN = 1.00E+00
+ WWL = 0.00E+00 LL = 0.00E+00 LLN = 1.00E+00
+ LW = 0.00E+00 LWN = 1.00E+00 LWL = 0.00E+00
+ CAPMOD = 2.00E+00 XPART = 5.00E-01 CGDO = 6.88E-10
+ CGSO = 6.85E-10 CGBO = 1.00E-12 CJ = '1.2e-3/proc_delta'
+ PB = 8.70E-01 MJ = 4.20E-01 CJSW = '2.4e-10/proc_delta'
+ PBSW = 8.00E-01 MJSW = 3.57E-01 CJSWG = 4.24E-10
+ PBSWG = 8.00E-01 MJSWG = 3.56E-01 CF = 0.00E+00
+ PVTH0 = 3.53E-03 PRDSW = 1.02E+01 PK2 = 3.35E-03
+ WKETA = 3.52E-02 LKETA = -2.06E-03 PU0 = -2.19E+00
+ PUA = -7.63E-11 PUB = 9.91E-22 PVSAT = 5.00E+01
+ PKETA = -6.41E-03 KF = 1.29E-29 PETA0 = 7.31E-05

```

Listing 6: Photo diode models

```

.subckt PhotoDiode VDD N1_R1C1
I1_R1C1 VDD N1_R1C1 DC Ipd_1
d1 N1_R1C1 vdd dwell 1
.model dwell d cj0=1e-14 is=1e-12 m=0.5 bv=40
Cd1 N1_R1C1 VDD 30f
.ends

```

C Verilog code

Listing 7: Main module for camera control testbench

```
1  /*
2   This is the testbench for camera_fsm, the logic that controls
   the camera.
3   */
4
5   `define _no_testbench_
6
7   `timescale 1ns / 1ps
8
9   `include "camera_fsm.v"
10
11
12  module camera_fsm_tb;
13      logic clk, init, inc, dec, reset;
14      logic expose, erase, nre1, nre2, adc;
15
16      camera_fsm test_camera(clk, init, inc, dec, reset, expose,
17                             erase, nre1, nre2, adc);
18
19      always @(*)
20          #1 clk <= !clk;
21
22      initial begin
23          $dumpfile("outfiles/out_camera_fsm_tb.vcd");
24          $dumpvars();
25
26          {clk, init, inc, dec, reset} = 5'b00001;
27          #2 reset = 0;
28
29          #5 inc = 1;
30          #2 dec = 1;
31          #30 inc = 0;
32          #4 dec = 0;
33          #2 init = 1;
34          #4 init = 0;
35
36          #30 inc = 1;
37          #4 inc = 0;
38
39          #28 reset = 1;
40          #2 reset = 0;
```

```

41         #4 dec = 1;
42         #60 dec = 0;
43
44         init = 1;
45         #1 init = 0;
46
47         #1 dec = 1;
48
49         #38 $finish;
50     end
51
52 endmodule // camera_fsm_tb

```

Listing 8: 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,
18                   reset,
19                   output logic expose, erase, nre_1, nre_2,
20                   adc_enable);
21
22     typedef enum          logic [1:0]
23                         {idle, exposing, readout, illegal}
24                         statetypes;
25
26     statetypes current_state, next_state;
27
28     logic          readout_enable, readout_reset,
29                   readout_finished;
30
31     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         );
34     fcd_reg countdown_reg(clk, fcd_enable, fcd_dset, fcd_data,
35         fcd_finished);
36     readout_seq readout_sequencer(clk, readout_enable,
37         readout_reset, readout_finished, nre_1, nre_2, adc_enable);
38
39
40     assign exp_inc = (current_state == idle) ? exponer_inc : 0;
41     assign exp_dec = (current_state == idle) ? exponer_dec : 0;
42
43     always @(posedge clk)
44     case (next_state)
45         idle: begin
46             {expose, erase, readout_enable, readout_reset,
47                 fcd_enable, fcd_dset} <= 6'b010101;
48             current_state <= next_state;
49         end
50         exposing: begin
51             {expose, erase, readout_enable, readout_reset,
52                 fcd_enable, fcd_dset} <= 6'b100110;
53             current_state <= next_state;
54         end
55         readout: begin
56             {expose, erase, readout_enable, readout_reset,
57                 fcd_enable, fcd_dset} <= 6'b001001;
58             current_state <= next_state;
59         end
60         default: current_state <= next_state;
61
62     endcase // case (next_state)
63
64     always @(*) begin
65         if (reset)
66             current_state = illegal;
67         else case (current_state)
68             illegal: begin
69                 exp_reset = 1;
70                 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 9: Exposure register

```

1  /*
2   This file contains a simple module for counting the exposure time
3   ,
4   it runs from 2 to 30 (requiering 5 bits)
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 10: 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 11: Readout sequencer

```

1  /*
2   This file contains a sequencer for the readout of the 4 pixel
3   camera.
4   It can be enabled and reset.
5   Note that the enable does not imply a reset.
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,
51                          adc);
52

```

```

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_

```