TFE4152
Design of Integrated Circuits

J. S. Bognæs T. Nordgård-Hansen

11th November 2019

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

Contents

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 **AIMSpice** and Icarus verilog **icarusVL** 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 **githubProject**.

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 thorugh PD1. While the photo diode limits the current, this still might lead to excessive power usage and subsequent heating issues over time.

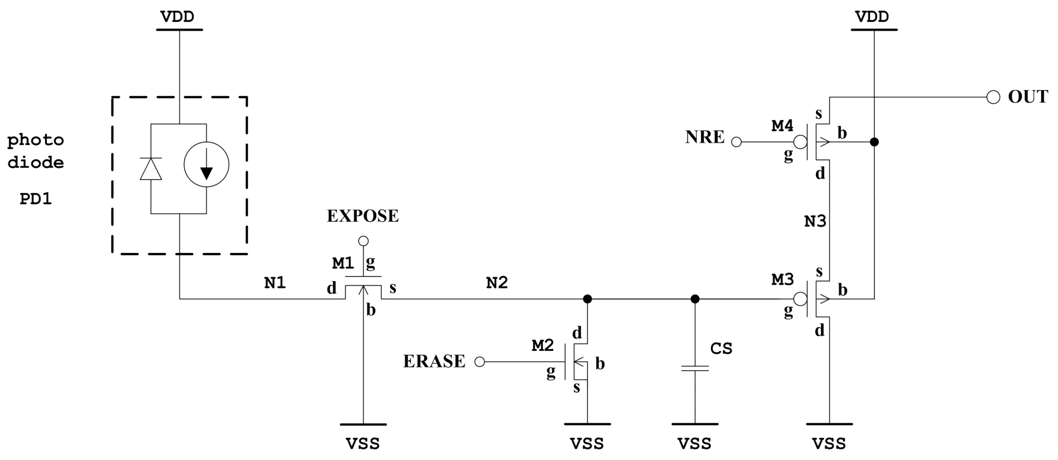


Figure 1: Schematic of one pixel with readout circuit, figure from **oppgave**

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

We will in this project assume that all voltage transient are so slow that no leakage current is present between gate and the other ports of any of the transistors, in the same way we assume there are no leakage currents through any of the capacitors.

As described in Analog integrated circuit design **AnalogBook** the current from drain to source $I_D \propto \frac{W}{L}$. This also makes sense from a geometric point of view.

In order to minimize the leakage current through a transistor that is shut off $\frac{W}{L}$ should be minimized.

3.3 Conceptual workings of a camera controller

As shown in Appendix A figure 7 the pixels depend on several digital input signals, the job of the camera controller is therefore to trigger these in the desired order. The requirements to be met by the controller are as follows:

- Pull the erase pin high except when exposing or reading the image
- Pull the expose pin high for an appropriate length of time as defined by the user
- Read out the values of all pixels in the correct order avoiding interference between different pixels on the same ADC.
- Enable the user to reset the whole system. Though the image being taken might be lost the camera should function normally afterwards.

4 Analog design

When designing the analog circuitry the topology as well as the technological limitations for production was taken from **oppgave**. This section therefore focuses on the physical dimensions of the different components as specified in figures 2 and 3.

The most important property of the analog pixel is that the charge stored over CS remains unchanged while being read, the transistors M1 and M2 must therefore be tuned for minimal leakage current as described in Section 3.2. The transistor M4 must be tuned in the same way to avoid any interference between P11 and P21 as well as between P12 and P22 during readout as shown in figure 3.

The current source transistors MC1 and MC2 must be tuned for the quickest possible response of the current source, this is in order to get the fastest possible stable output when reading from a pixel. They are therefore tuned for maximum current throughput as explained in Section 3.2 and verified in Section 6.

The capacitor CS and transistor M3 are tuned to empirically found values as shown in Section 6 in order to give the best dynamic range of the pixel as a function of lighting conditions and exposure time.

All component values are shown in table 1.

Table 1: Physical values of components

Panel A: transistors

Component	W	L
M1	1.08μ	1.08μ
M2	1.08μ	1.08μ
M3	3.00μ	0.67μ
M4	1.08μ	1.08μ
MC1	5.04μ	0.36μ
MC2	5.04μ	0.36μ

Panel B: capacitors

Component	C
CS	$2.5pF$
CC1	$3.0pF$
CC2	$3.0pF$

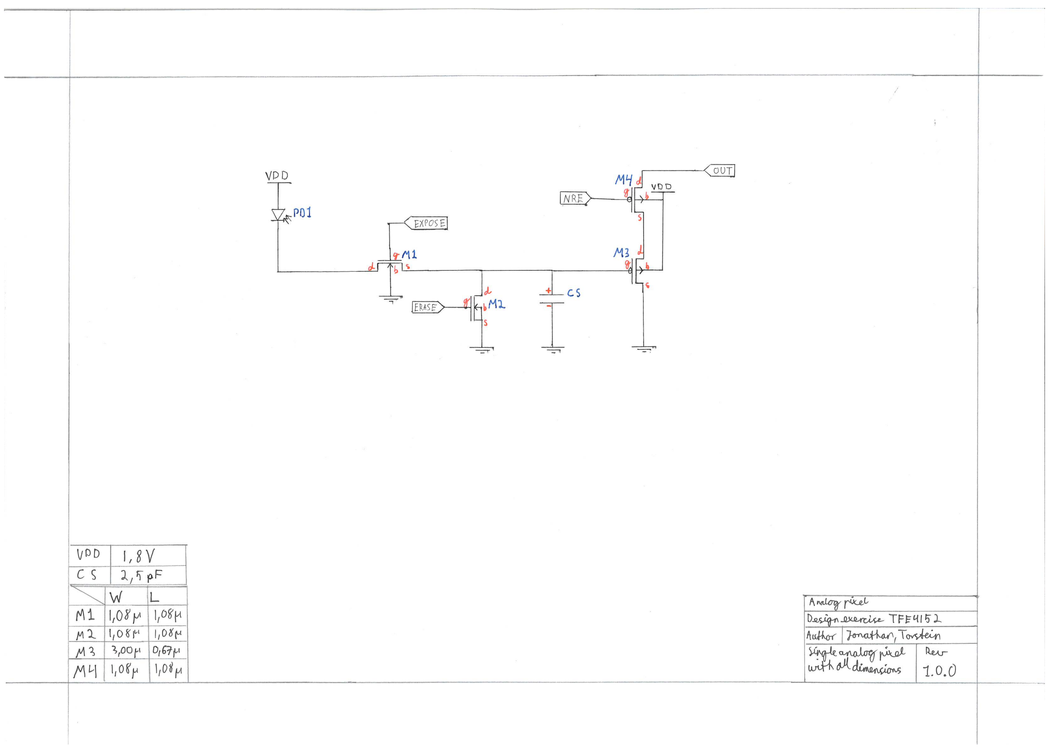


Figure 2: Implementation of one pixel, figure allso exist in Appendix A

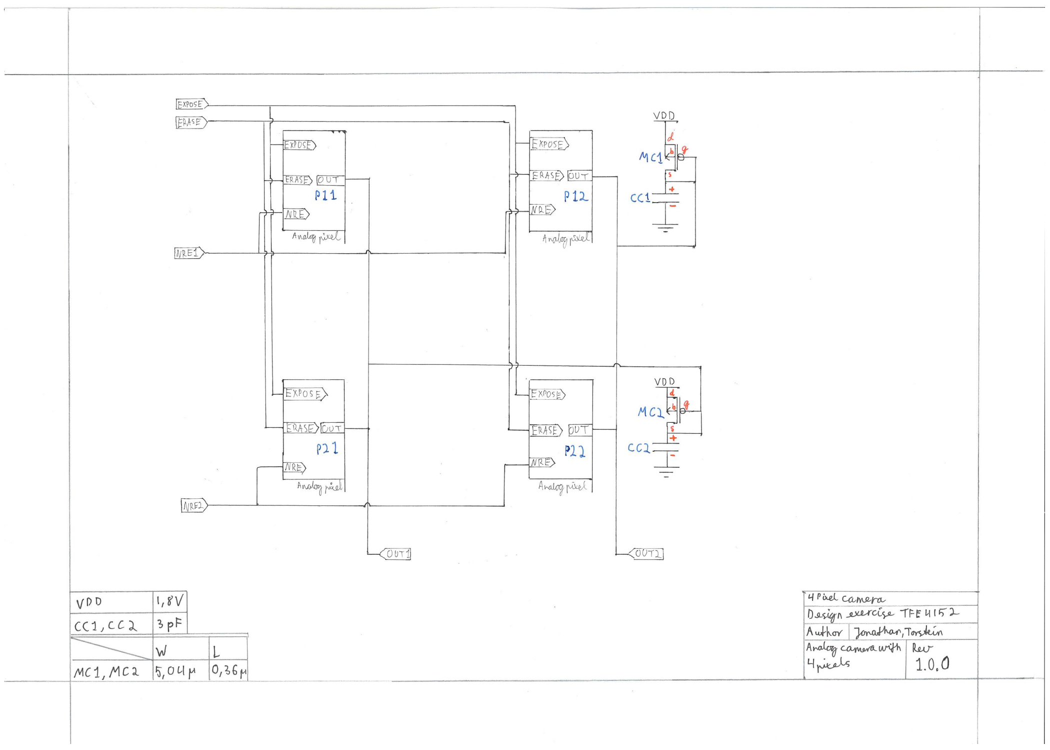


Figure 3: Implementation of the analog part of the camera, figure allso exist in Appendix A

5 Digital design

The digital control system is designed as a finite state machine with the states illegal, idle, expose and readout as shown in figure 4. The logic connecded to each state is as shown below.

- Illegal
  - This state is mainly used for resets, it resets all peripherals and set the next state to idle.
- Idle
  - This is the normal operating state, the machine allways return to this state eventually.
  - In this state exposure increase and exposure decrease are enabled, increase takes presidence over decrease.
- Expose
  - The machine normally stays in this state until the exposure time has passed.
  - Reset is the only functional input.
- Readout
  - The readout sequencer is started, the next state is set to idle afterwards.
  - Reset is the only functional input.

The hardware was implemented as shown in more detail in figure 5. The countown logic for the exposure was divided into two parts, one 5bit register to store a value between 2 and 30 as well as a countdown register to signal the end of the exposure period.

The readout cycle was also split out into its own sequencer independent of the rest of the machine.

The full logical description of the digital control system is given in SystemVerilog 2012 in Appendix C along with simple test benches for the various components. The design was heavily inspired by the FSM examples from **DigitalBook**.

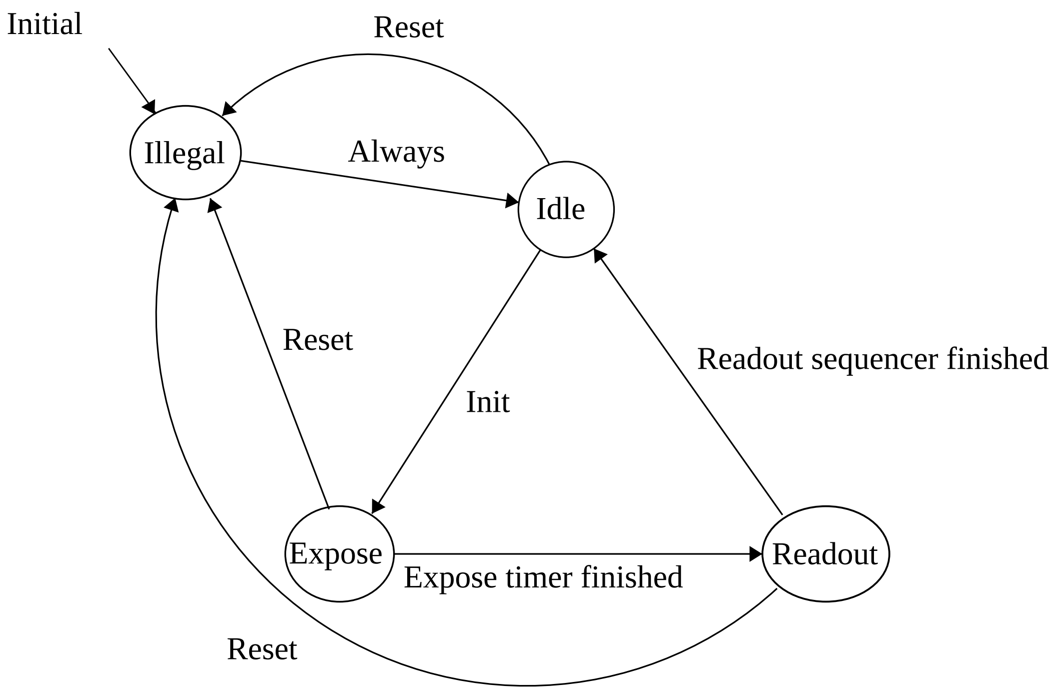


Figure 4: FSM representation of the digital control system

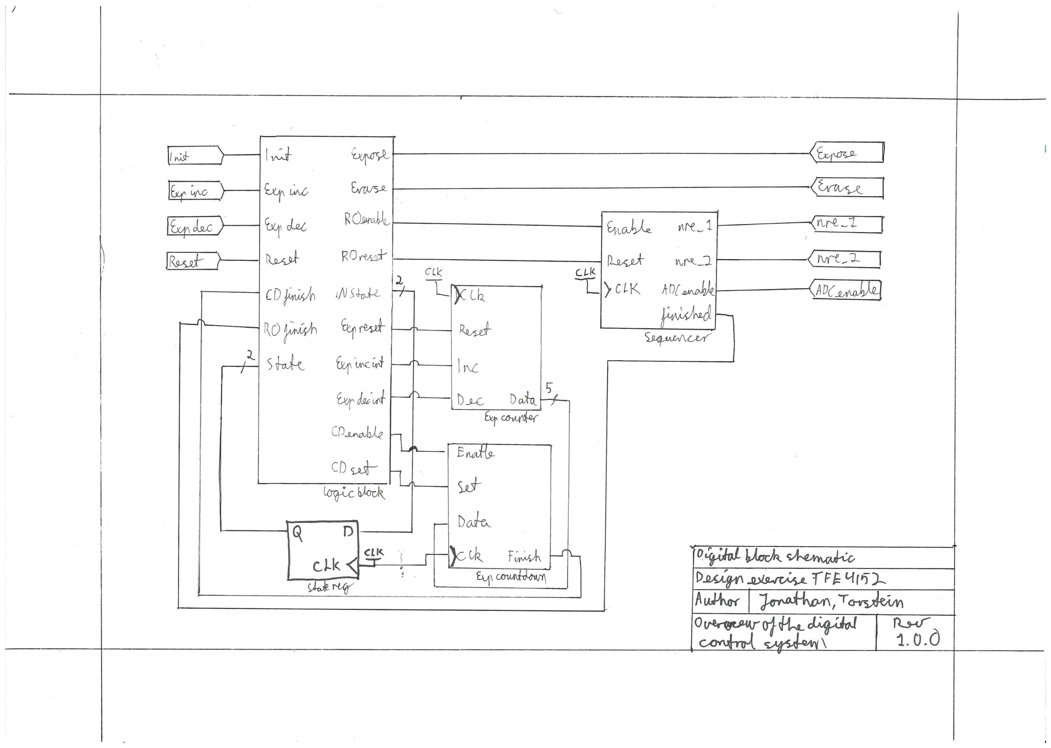


Figure 5: Schematic of the digital design, figure allso found in Appendix A

6 Simulations

6.1 Analog simulations

All simulations of the analog circuitry were done using the AimSpice SPICE backend **AIMSpice** along with the AIMPlot **aimplot** frontend.

6.2 Digital simulations

All simulation of the digital control system were run using SystemVerilog testbenches in icarus verilog **icarusVL** and shown in GTKWave **gtkwave**.

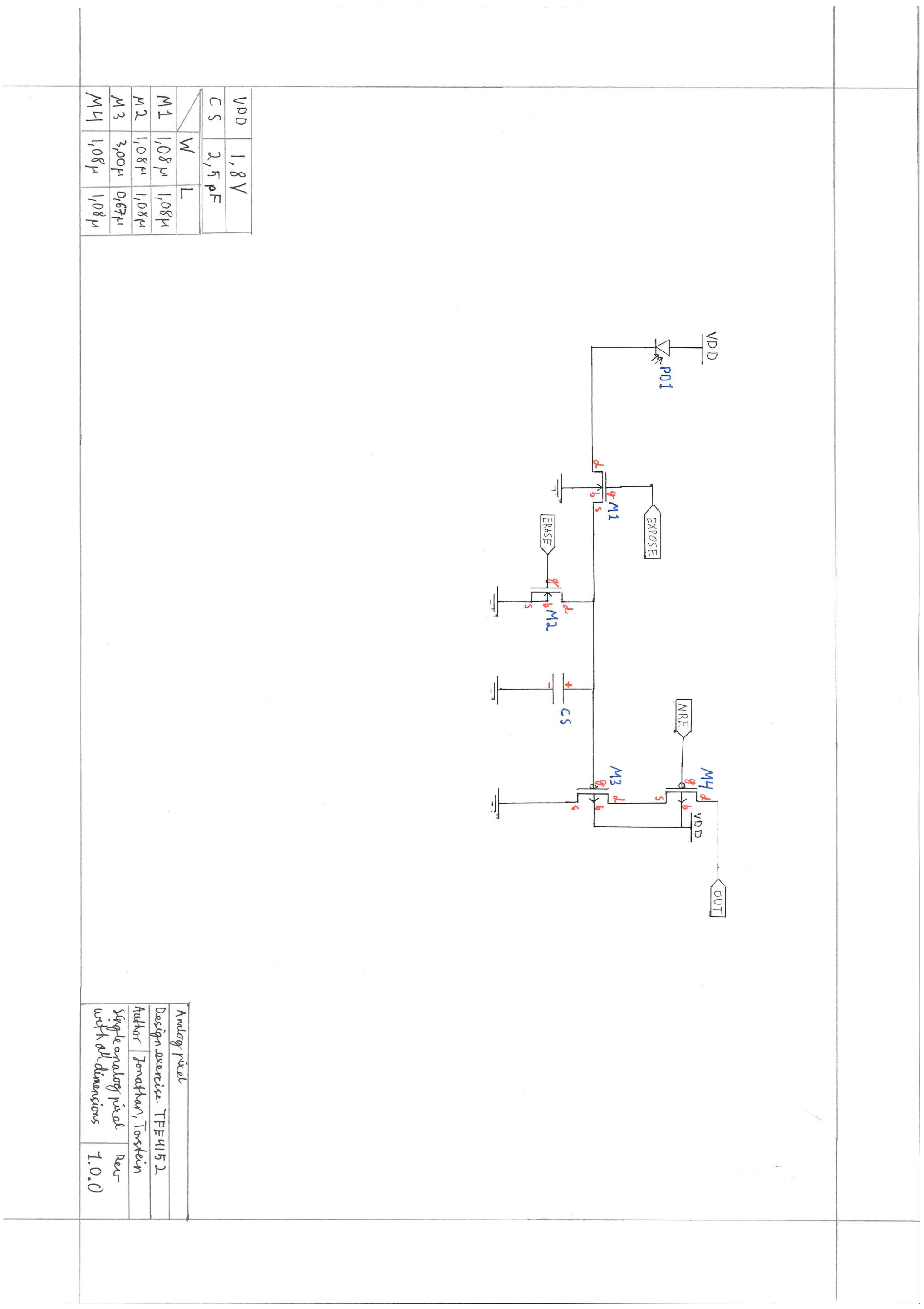
7 Conclusion

Such conclution, very concluded.

8 References

Appendices

A Schematics



VDD	1,8V
CS	2,5pF

	W	L
M1	1,08µ	1,08µ
M2	1,08µ	1,08µ
M3	3,00µ	0,67µ
M4	1,08µ	1,08µ

Analog pixel	
Design exercise TFE4152	
Author	Jonathan, Torstein
Single analog pixel with all dimensions	Rev 1.0.0

Figure 6: Analog schematic of one pixel

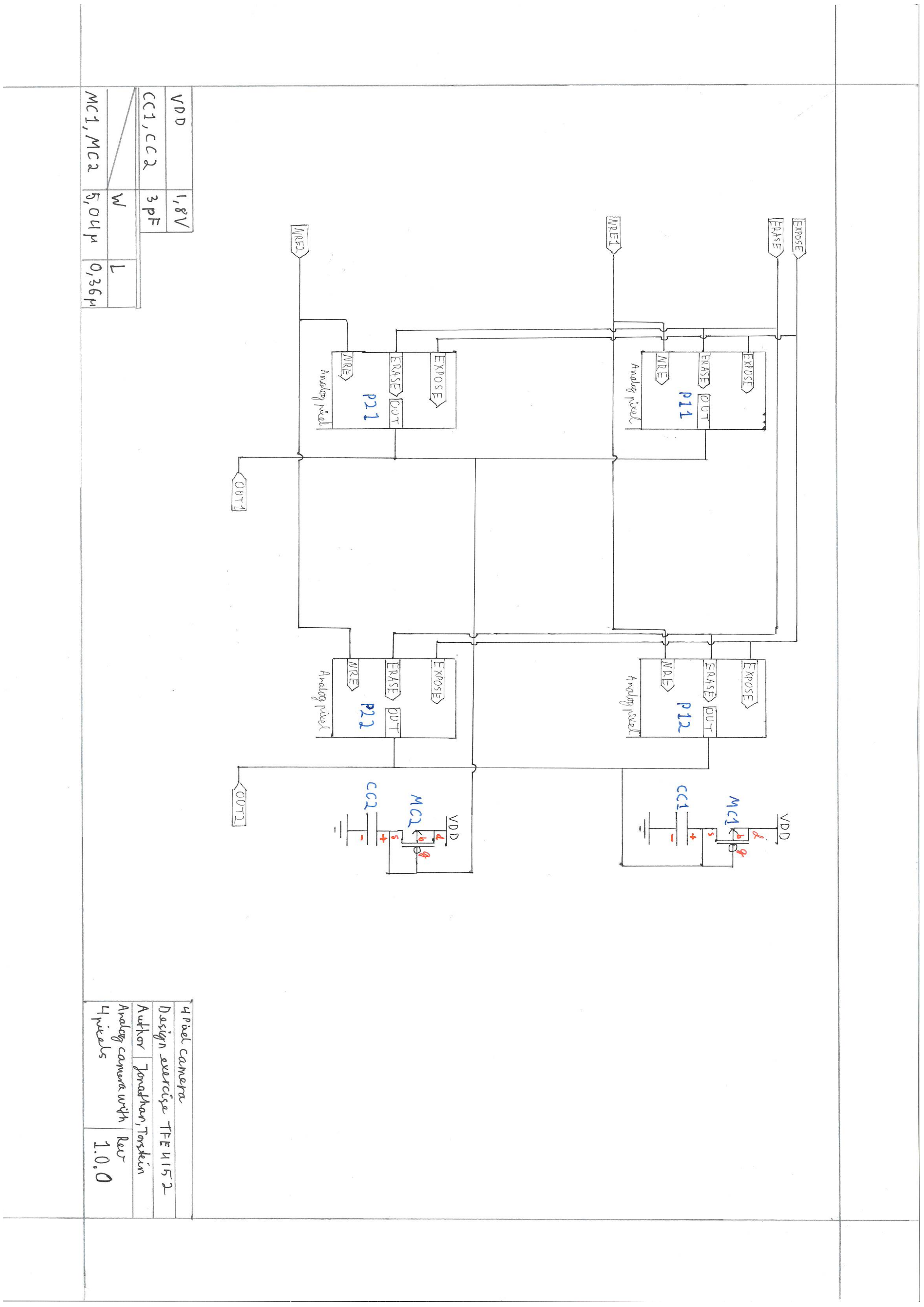


Figure 7: Analog schematic of 4 pixels in a camera

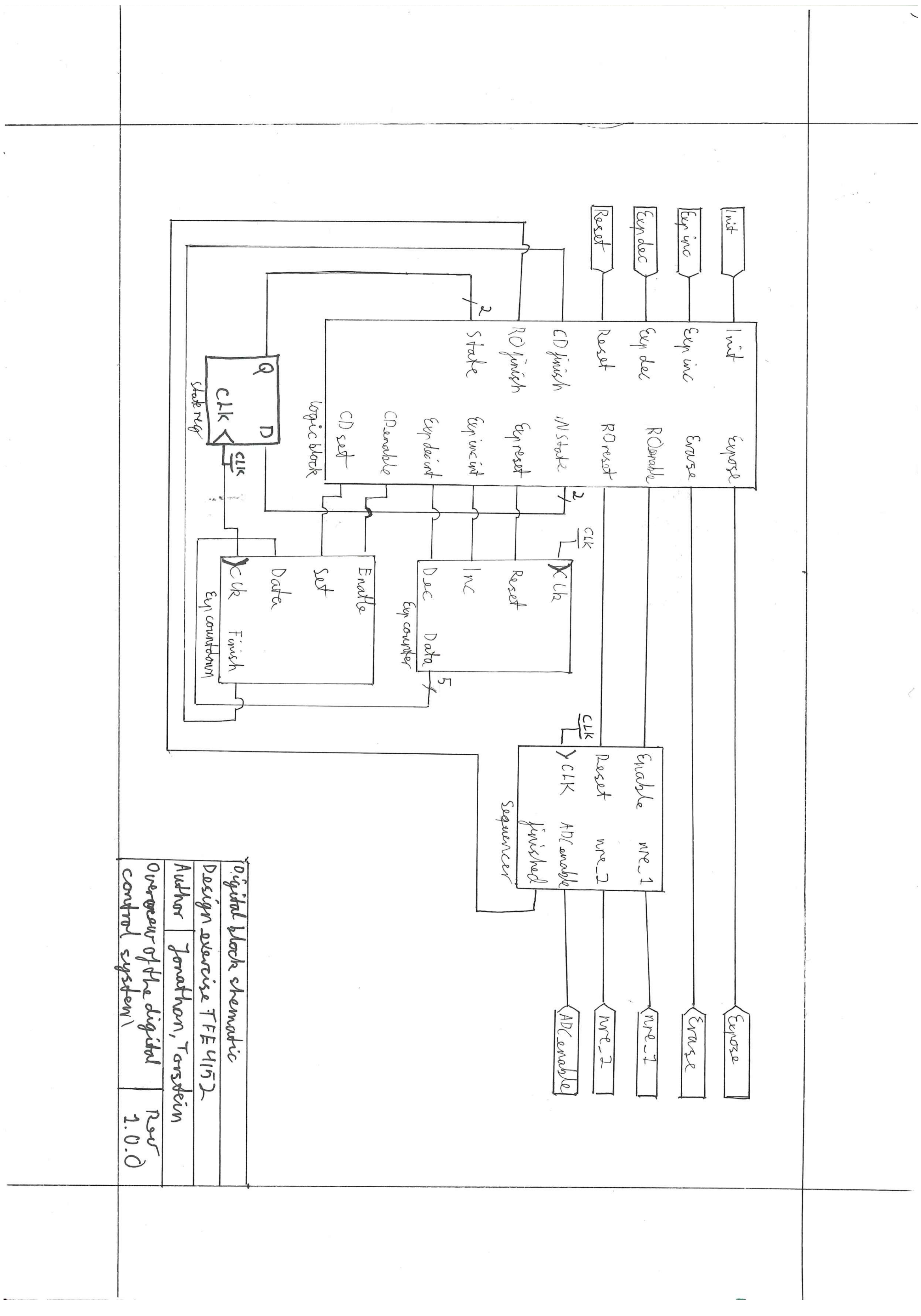


Figure 8: Digital schematic of the camera control

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=MCW 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} ! Pulsewidth 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 MCW = 5.04u
.param MCL = 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   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   =  6.88E−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

```verilog
/*
 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;

      #38 $finish;
   end

endmodule // camera_fsm_tb
```

Listing 8: Main module for camera control

```
/*
 This file contains the controll logic of the FSM of the camera.
 It depends on the fcd_reg, exp_reg and read_sequencer to function
    .
 */

// `define _no_testbench_

`ifndef _fsm_logic_v_31_
 `define _fsm_logic_v_31_

 `include "exp_reg.v"
 `include "fcd_reg.v"
 `include "readout_seq.v"


module camera_fsm(input  logic clk,
                  input logic  init, exponer_inc, exponer_dec,
                     reset,
                  output logic expose, erase, nre_1, nre_2,
                     adc_enable);

   typedef enum                 logic [1:0]
                                {idle, exposing, readout, illegal}
                                   statetypes;
   statetypes current_state, next_state;

   logic                        readout_enable, readout_reset,
      readout_finished;

   logic                        fcd_enable, fcd_dset, fcd_finished;
   logic [4:0]                  fcd_data;

   logic                        exp_reset, exp_inc, exp_dec;


   exp_reg exposure_reg(clk, exp_reset, exp_inc, exp_dec, fcd_data
      );
   fcd_reg countdown_reg(clk, fcd_enable, fcd_dset, fcd_data,
      fcd_finished);
   readout_seq readout_sequencer(clk, readout_enable,
      readout_reset, readout_finished, nre_1, nre_2, adc_enable);


   assign exp_inc = (current_state == idle) ? exponer_inc : 0;
   assign exp_dec = (current_state == idle) ? exponer_dec : 0;

   always @(posedge clk)
     case (next_state)
       idle: begin
          {expose, erase, readout_enable, readout_reset,
             fcd_enable, fcd_dset} <= 6'b010101;
          current_state <= next_state;
       end
       exposing: begin
          {expose, erase, readout_enable, readout_reset,
             fcd_enable, fcd_dset} <= 6'b100110;
          current_state <= next_state;
       end
       readout: begin
          {expose, erase, readout_enable, readout_reset,
             fcd_enable, fcd_dset} <= 6'b001001;
          current_state <= next_state;
       end
       default: current_state <= next_state;

     endcase // case (next_state)


   always @(*) begin
      if (reset)
        current_state = illegal;
      else case (current_state)
             illegal: begin
                exp_reset = 1;
                readout_reset = 1;
```

```
                exp_reset = 0;
                readout_reset = 0;
                next_state = idle;
             end
             idle: if (init) next_state = exposing;
             exposing: if (fcd_finished) next_state = readout;
             readout: if (readout_finished) next_state = idle;
             default: next_state = idle;
           endcase // case (state)
   end // always @ (*)

   initial
     current_state = illegal;

endmodule // camera_fsm


`endif
```

Listing 9: Exposure register

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

`ifndef _exp_reg_v_25_
 `define _exp_reg_v_25_


module exp_reg(input  logic clk, reset, inc, dec,
               output logic [4:0] q);

   always @(posedge clk)
     if (inc & (q < 30) & !reset)
       q <= q + 1;
     else if (dec & (q > 2) & !inc & !reset)
       q <= q - 1;

   always @(*)
     if (reset | q > 30 | q < 2)
       q <= 5'd15;

   initial
     q = 5'd15;
```

```

endmodule // exp_reg


 `ifndef _no_testbench_


module exp_reg_tb;
   logic clk, reset, inc, dec;
   logic [4:0] q;

   exp_reg testreg(clk, reset, inc, dec, q);

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

      #1 {clk, reset, inc, dec} = 4'b1100;
      #1 {clk, reset} = 2'b10;

      for (int i = 1; i <= 20; i = i+1) begin
         #1 {clk, inc, dec} = 3'b111;
         #1 {clk, inc, dec} = 3'b011;
      end

      for (int i = 1; i <= 40; i = i+1) begin
         #1 {clk, inc, dec} = 3'b101;
         #1 {clk, inc, dec} = 3'b001;
      end

      #1 reset = 1;
      #1 reset = 0;

      #1 {clk, reset, inc, dec} = 4'b1101;
      #1 {clk, reset} = 2'b00;
      #1 clk = 1;
      #1 clk = 0;

      #1 $finish;
   end

endmodule // exp_reg_tb


 `endif //  `ifndef _no_testbench_
```

```
`endif //  `ifndef _exp_reg_v_25_
```

Listing 10: Counter for exposure time

```
/*
 This file contains the register used to count down different
    values in the camera.
 It has 5 bits, can take data inn and counts down on rising clock
    edge when enabled.
 It has an output for when the internal data is 0.
 Data set has priority over enable.
 */

`ifndef _fcd_reg_v_28_
 `define _fcd_reg_v_28_


module fcd_reg(input logic clk, enable, dset,
               input logic [4:0] dread_data,
               output logic      finished);

   logic [4:0]                   data_int;

   always @(posedge clk)
     if (dset)
       data_int <= dread_data;
     else if (enable && (data_int > 'b0))
       data_int <= data_int - 1;

   assign finished = (data_int <= 5'd1) ? 1 : 0;

endmodule // fcd_reg


 `ifndef _no_testbench_


module fcd_reg_tb;
   logic clk, enable, dset, finished;
   logic [4:0] data_in;

   fcd_reg testreg(clk, enable, dset, data_in, finished);

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

      #1 {clk, enable, dset, data_in} = 7'b0000111;
      #1 {clk, dset} = 2'b11;
      #1 clk = 0;
      #1 {clk, enable, dset} = 3'b110;

      for (int i = 1; i <= 21; i = i + 1) begin
         #1 clk = !clk;
      end

      #1 {clk, enable} = 2'b10;
      #1 clk = 0;
      #1 clk = 1;

      for (int i = 1; i <= 5; i = i + 1) begin
         #1 clk = !clk;
      end

      #1 {clk, enable, dset} = 3'b111;

      for (int i = 1; i <= 5; i = i + 1) begin
         #1 clk = !clk;
      end

      #1 $finish;
   end // initial begin

endmodule // fcd_reg_tb





 `endif //  `ifndef _no_testbench_
`endif //  `ifndef _fcd_reg_v_28_
```

Listing 11: Readout sequencer

```
/*
 This file contains a sequencer for the readout of the 4 pixel
    camera.
 It can be enabled and reset.
 Note that the enable does not imply a reset.
 */

`ifndef _readout_seq_v_31_
```

```
 `define _readout_seq_v_31_


module readout_seq(input logic clk,
                   input logic  enable, reset,
                   output logic finished,
                   output logic nre1, nre2, adc);

   logic [3:0]                  step;

   assign finished = (step >= 8) ? 1 : 0;

   always @(posedge clk)
      if (!finished && enable && !reset) step <= step + 1;

   always @(*) begin
      if (reset)
        step <= 0;

      case (step)
        0: {nre1, nre2, adc} = 3'b000;
        1: {nre1, nre2, adc} = 3'b100;
        2: {nre1, nre2, adc} = 3'b101;
        3: {nre1, nre2, adc} = 3'b100;
        4: {nre1, nre2, adc} = 3'b000;
        5: {nre1, nre2, adc} = 3'b010;
        6: {nre1, nre2, adc} = 3'b011;
        7: {nre1, nre2, adc} = 3'b010;
        8: {nre1, nre2, adc} = 3'b000;
        default: {nre1, nre2, adc} = 3'b000;
      endcase // case (step)
   end // always @ (*)

   endmodule // readout_seq


 `ifndef _no_testbench_


module readout_seq_tb;
   logic clk, enable, reset, finished, nre1, nre2, adc;

   readout_seq test_seq(clk, enable, reset, finished, nre1, nre2,
      adc);

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

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

      {clk, enable, reset} = 3'b000;
      #5 reset = 1;
      #2 reset = 0;

      #4 enable = 1;
      #20 enable = 0;
      #4 enable = 1;
      #1 reset = 1; #1
      #2 reset = 0;
      #4 enable = 0;

      #4 $finish;
   end // initial begin

endmodule // readout_seq_tb


 `endif
`endif //  `ifndef _readout_seq_v_31_
```