TFE4152
Design of Integrated Circuits

J. S. Bognæs T. Nordgård-Hansen

17th 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 the square root of the number of pixels ADCs this design can be adapted to an arbitrary quadratic digital camera.

The report concists of a summary of the necessary theory, a description of the design as well as simulations.

2 Introduction

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

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.

The design was tested with AIM-Spice [1] and Icarus verilog [2] as described in Section 6 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, the transistor M2 is opened to reset the voltage stored over the capacitor CS.

It is important that the transistors 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.

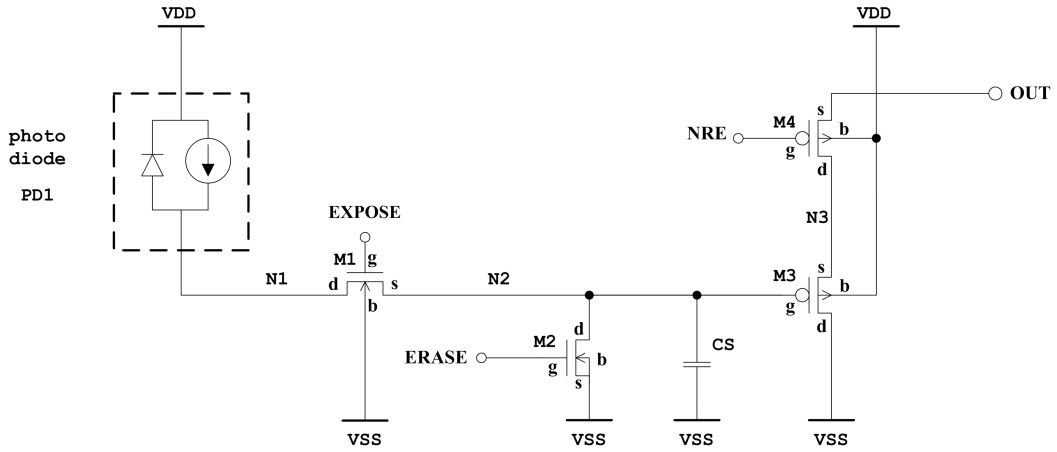

Figure 1: Schematic of one pixel with readout circuit, figure from [4]

The transistor M3 is used to convert the voltage stored over CS into a variable resistance between the nodes N3 and VSS for a nondestructive readout of the pixel, the transistor 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 [5] 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 figure 18 found in Appendix A the behavior of 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 needs to be taken into account. The analog circuitry therefore needs to be scaled for the use case and the trade-offs we are willing to make. The analog design is mainly for handling the current coming from the light sensor, and converting it to an voltage signal to be used by the ADC's. This section focuses on the physical dimensions of the different components in the camera.

4.1 Analog system overview

Each single pixel is designes with the dopology described in Section 3.1, a more detailed schematic is shown in figure 2.

The camera consists of four pixels aranged in a $2 \cdot 2$ matrix with one analog output for each column. A current source is connected to each output line made from the transistors MC1 and MC2 as well as the capacitors CC1 and CC2. This is done in order to create a stable readable output signal as demonstrated in Section 6.1.

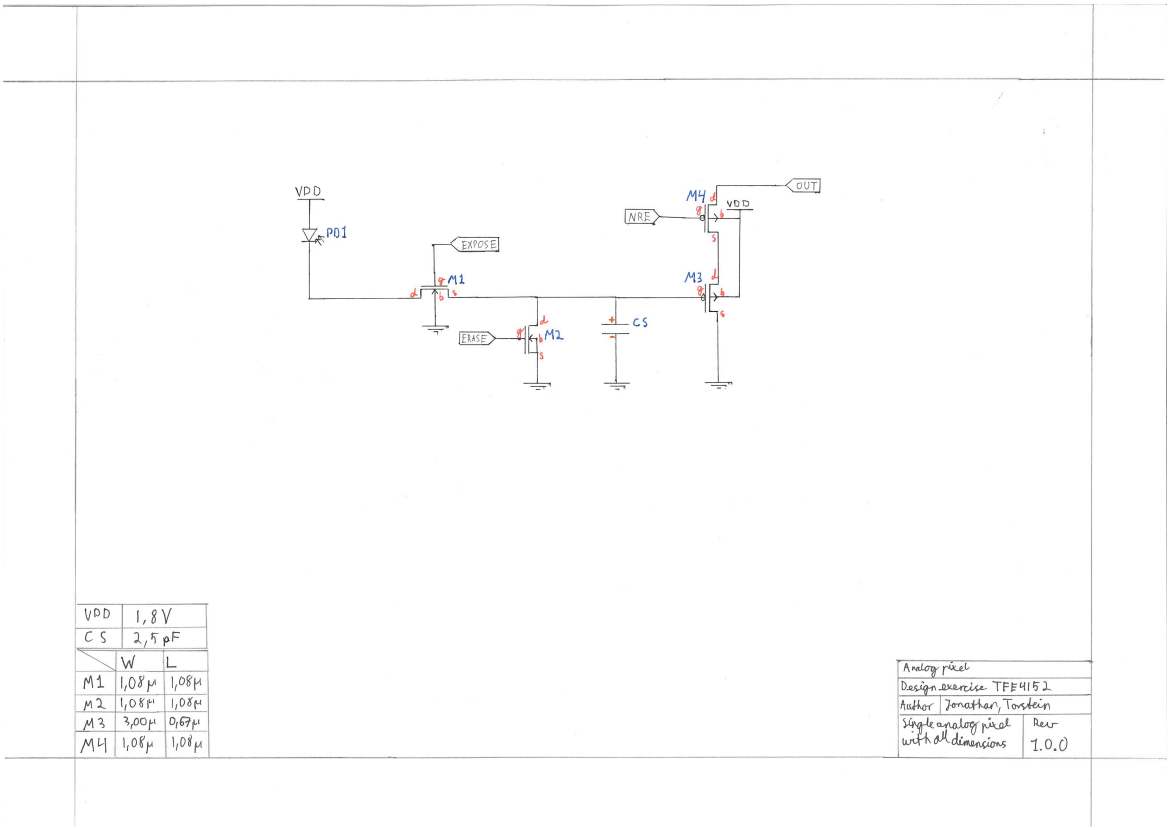

Figure 2: Implementation of one pixel, figure also exist in Appendix A

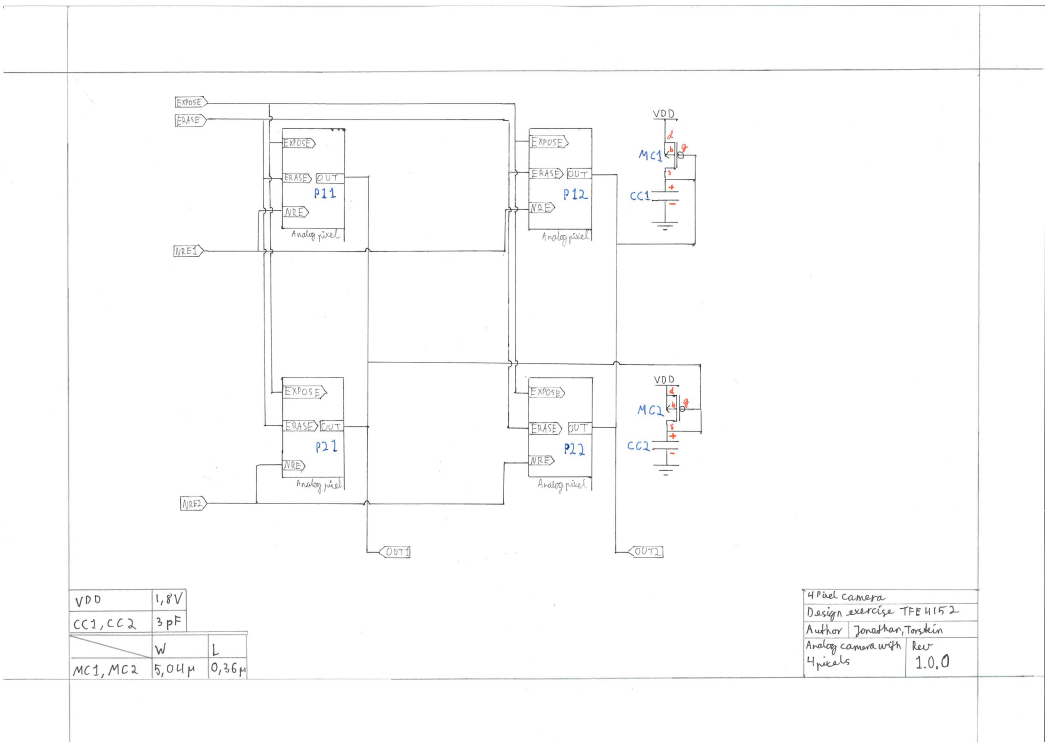

Figure 3: Implementation of the analog part of the camera, figure also exist in Appendix A

4.2 Transistor dimensions

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 transistors are limited by the technology used and must be in range: $0.3\,\mu m < L < 1.080\,\mu m$ and $1.080\,\mu m < W < 5.040\,\mu m$. This can be found in [4]. 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. Though the analog design assumes a perfect production line, it should work for typical transistors seeing as the main principles of the design would still work.

The choice of physical dimensions are showed in table 1.

4.3 Values for capacitors

The capacitors will be scaled to have maximum dynamic range. We know that the exposure time will be between $2\,ms$ and $30\,ms$, and the capacitor should be fully charged at exactly $30\,ms$. If the capacitance is too low, the capacitor will be fully charged before the full $30\,ms$ exposure time, which is not ideal. This needs to be correct when the light sensor is maxed out. The capacitance for CS was manually tuned for maximum

Table 1: Physical values of transistors

Component	W[m]	L[m]
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

bandwidth in simulation, see Section 6.

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.

Table 2: Capacitance of capacitors

Component	C[F]
CS	$2.5\,pF$
CC1	$3.0\,pF$
CC2	$3.0\,pF$

5 Digital design

The digital system as a whole consists of a few parts, mainly the control block, the ADC's and the clock pulse generator.The clock pulse generator creates a clock signal at $1\,KHz$, which is being used as a synchronizer clock signal for the control block.

The control block will activate each action on each rising clock edge.

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 connected to each state is as 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 functioning input.
- Readout
  - The readout sequencer is started, the next state is set to idle afterwards.
  - Reset is the only functioning input.

The main concern in the implementation of the hardware was maintainability since the project is meant as a proof of concept and not a production ready design. 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

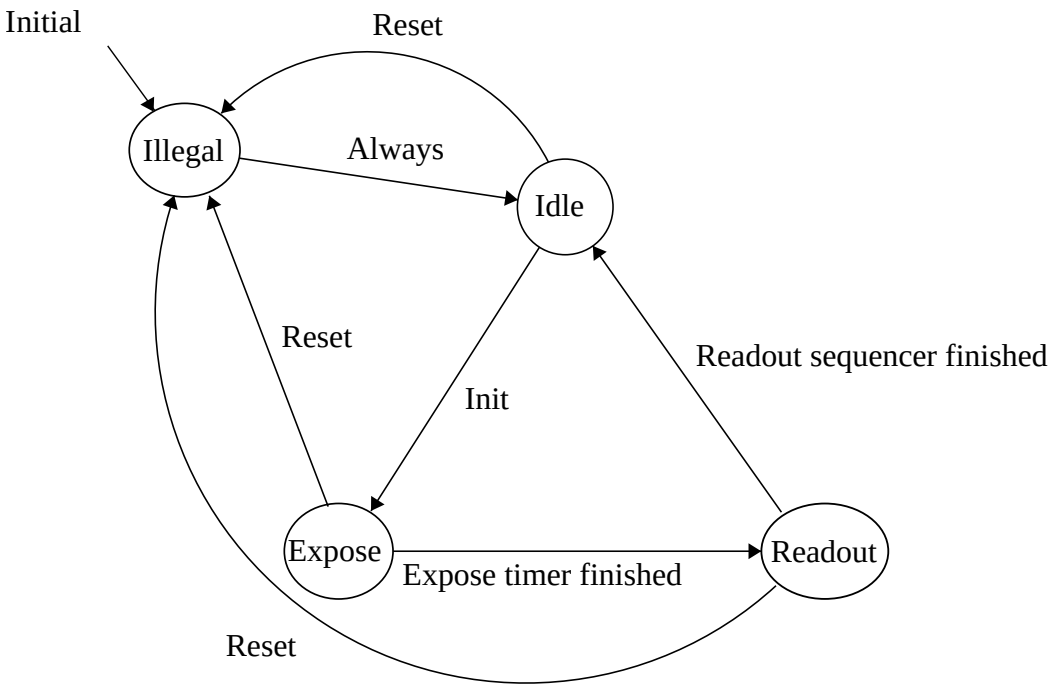

Figure 4: FSM representation of the digital control system

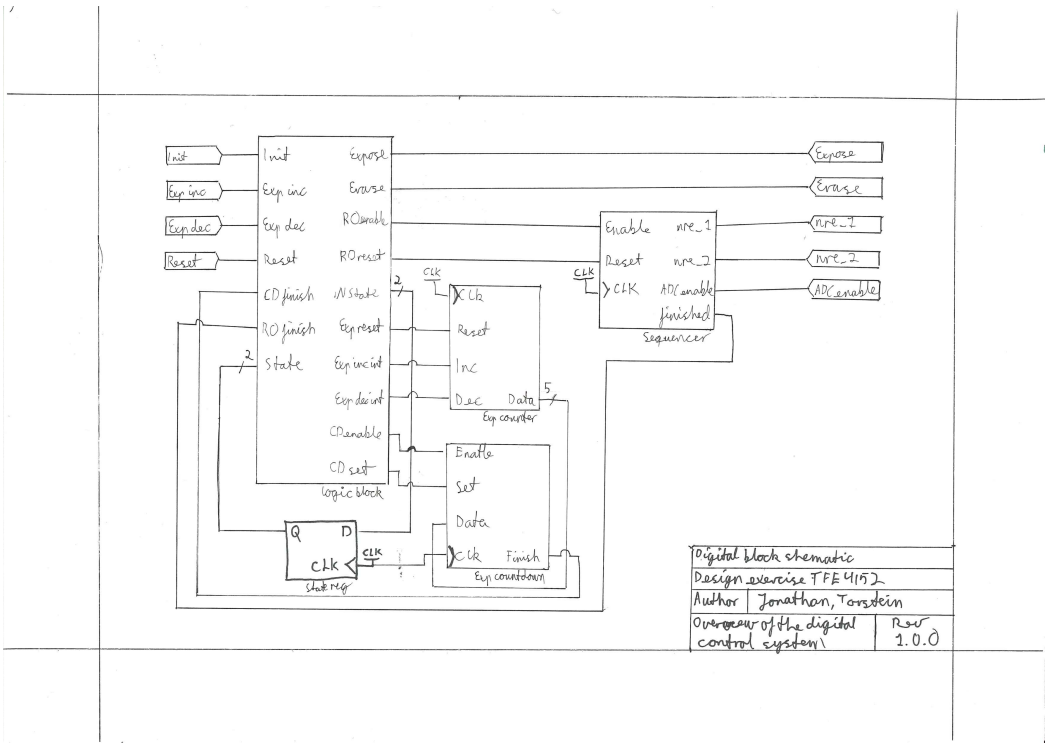

Figure 5: Schematic of the digital design, figure allso found in Appendix A

heavily inspired by the FSM examples from [6].

6 Simulations

6.1 Analog simulations

All simulations of the analog circuitry were done using the AimSpice SPICE backend [1] along with the AIMPlot [7] frontend. The resulting figures **??** through 11 display the same voltage signals plotted as a function of time. From top to bottom of the legend the signals are:

- The eraze control input
- The expose control input
- The voltage stored over the capacitor CS in pixel 11
- The readout enable control input for row 1
- The readout enable control input for row 2
- The output of column 1
- The output of column 2

Aditional simulations can be done by the reader utilizing the SPICE code in Appendix B.

The camera was simulated as a whole with the spice code described in Appendix B listing B. In order to show that the camera is working as intended simulations were performed with 750 pA current from the diodes and 2 ms exposure time as well as 50 pA diode current and 30 ms as shown in figure 6 and 7. This shows that the camera can handle both maximum and minimum lighting conditions without over or under exposure.

A more typical situation is shown in figure 8 with 400 pA diode current and 4 ms exposure.

The camera is still possible to overexpose as shown in figure 9, the user should therefore set the exposure to a sensible level.

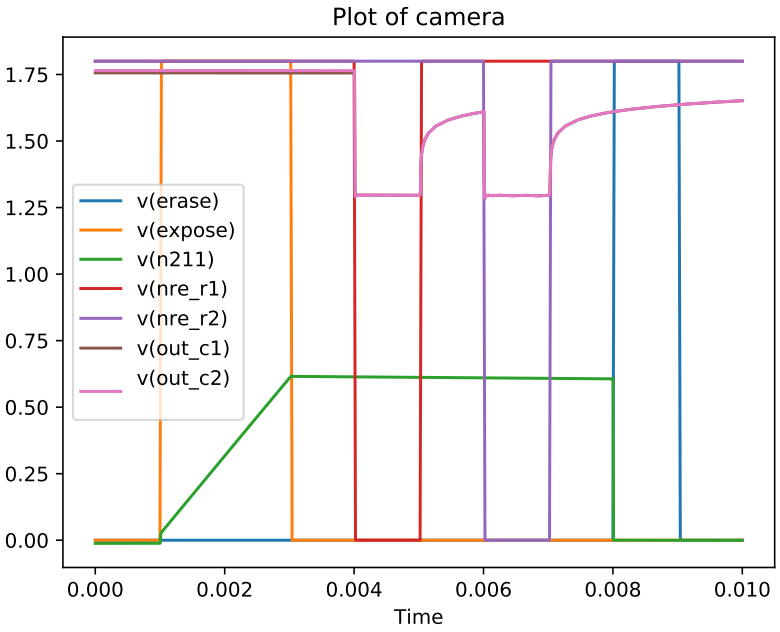

Figure 6: Analog camera tested with 750 pA and 2 ms

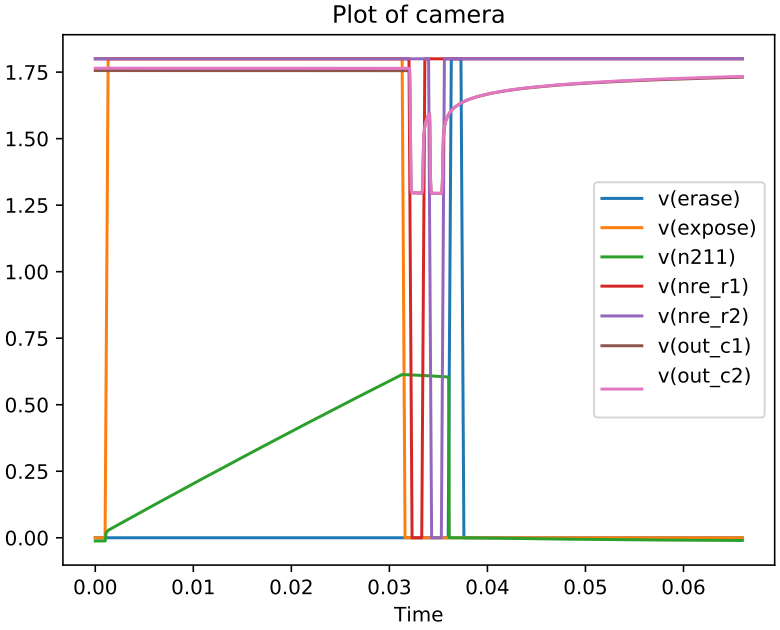

Figure 7: Analog camera tested with 50 pA and 30 ms

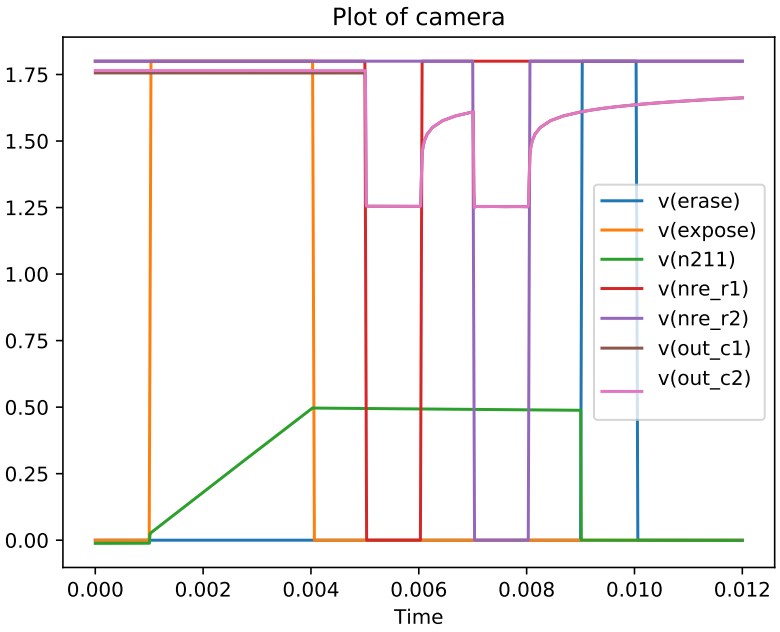

Figure 8: Analog camera tested with 400 pA and 3 ms

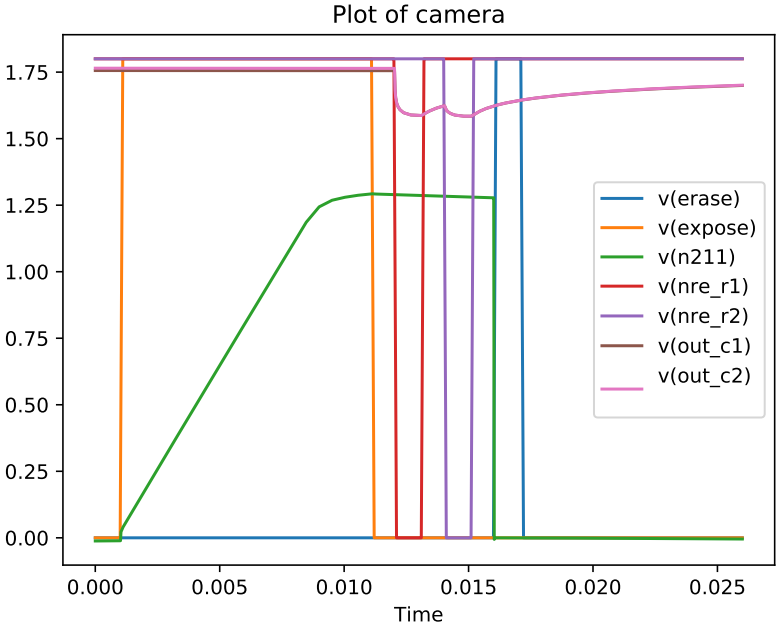

Figure 9: Analog camera tested with 400 pA and 10 ms

Both the need for the current amplifiers and the possibility of leakage through M1 or M2 were discussed in Section 4. In order to demonstrate these effects the camera was tested both without the current amplifiers and with a leaky transistor M2 as shown in figures 10 and 11.

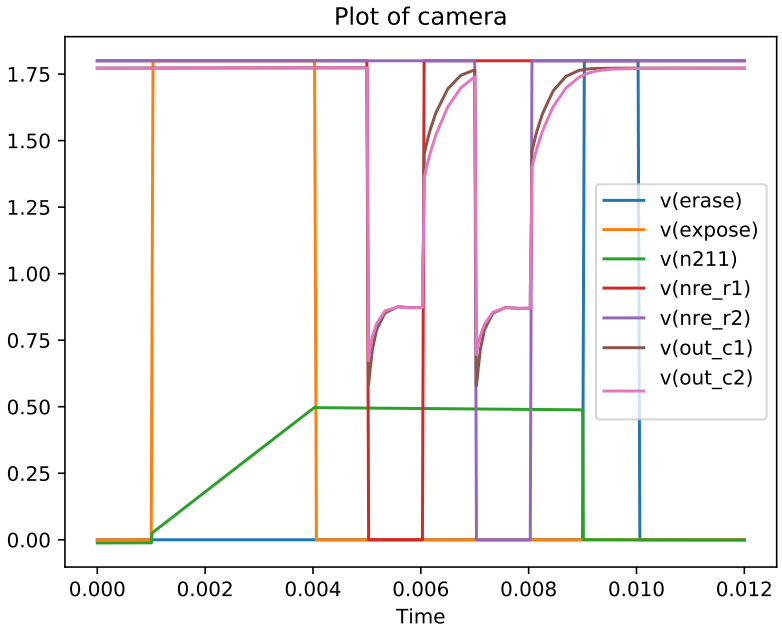

Figure 10: Analog camera tested without current amplifiers

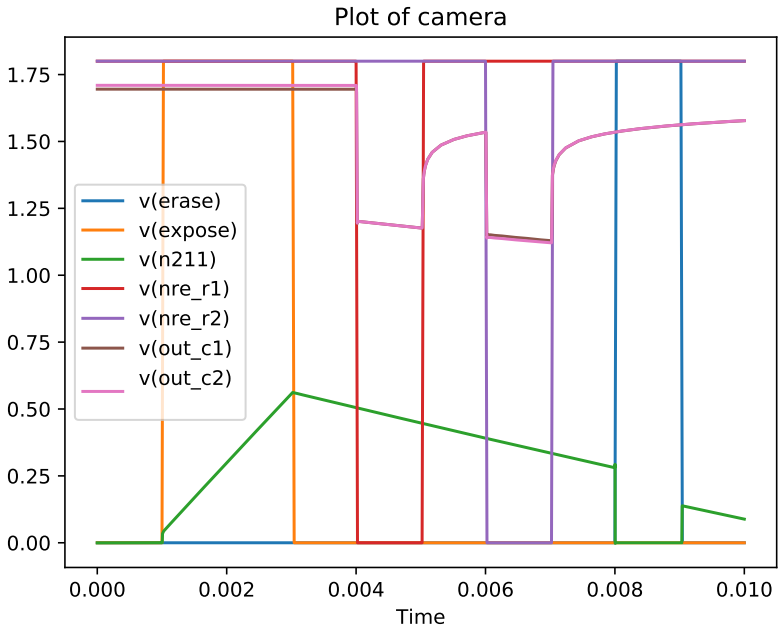

Figure 11: Analog camera tested with transistor M2 tuned for maximum leakage

6.2 Digital simulations

Each individual module of the digital design was tested with test benches written in SystemVerilog as shown in Appendix C. They were run using icarus-verilog [2] and the result visualized using GTKWave [8]. All waveform files are available from [3].

In figures 12 through 14 the color green was used for input, orange for internal state and red for output. Note that the clock signal is red for better contrast. In figures 15 and 16 the color blue is also introduced for internal signals of submodules.

Figure 12 shows the exposure counter, it demonstrates its minimum and maximum values of 2 and 30 being reached without bugs.

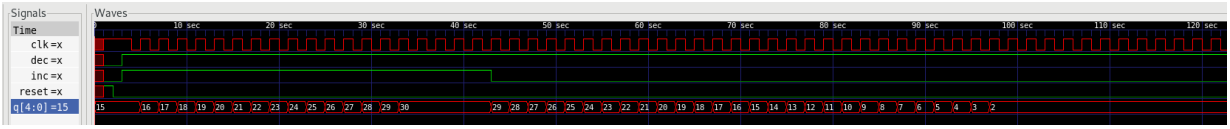

Figure 12: Digital test of exposure time counter

Figure 13 shows the countdown register for the exposure time. Its internal state *data_int* is shown in orange to indicate that it is inaccessible from the outside, it is shown purely for informational purposes.
Note that the finished signal is triggered at $data\_int \geq 1$ instead of $data\_int \geq 0$, this is in order to trigger the transition to the readout state on the same clock cycle that the register reaches 0.

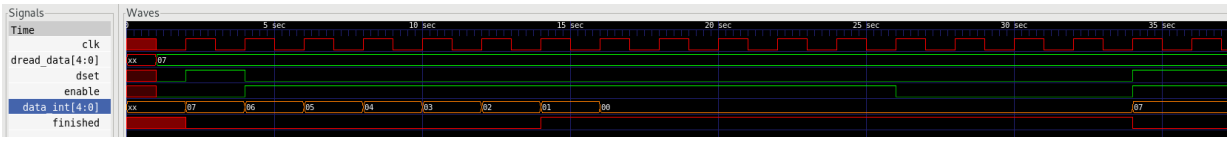

Figure 13: Digital test of exposure countown circuit

Figure 14 shows the readout sequencer, as with the countdown register the *step* signal is shown in orange to indicate that it is an internal inaccessible state.

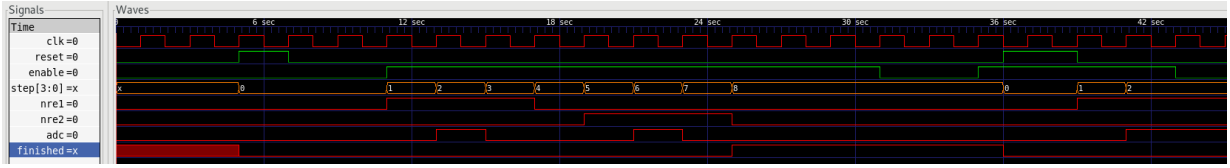

Figure 14: Digital test of readout sequencer

The camera control test-bench was to long to show on one page, its output is therefore split between figure 15 and 16. Note that the orange signals does not indicate internal states, color is instead used purely for contrast between signals for easy reading.

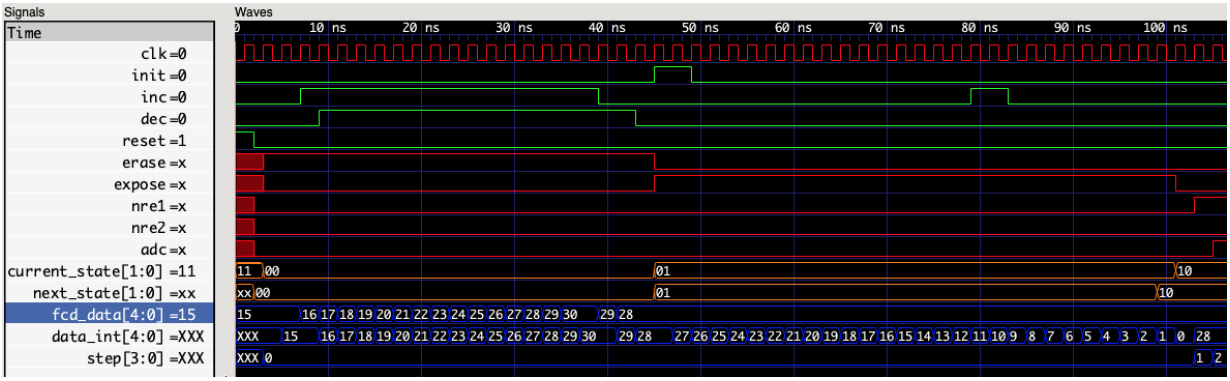

Figure 15: Digital test of the control circuitry part 1

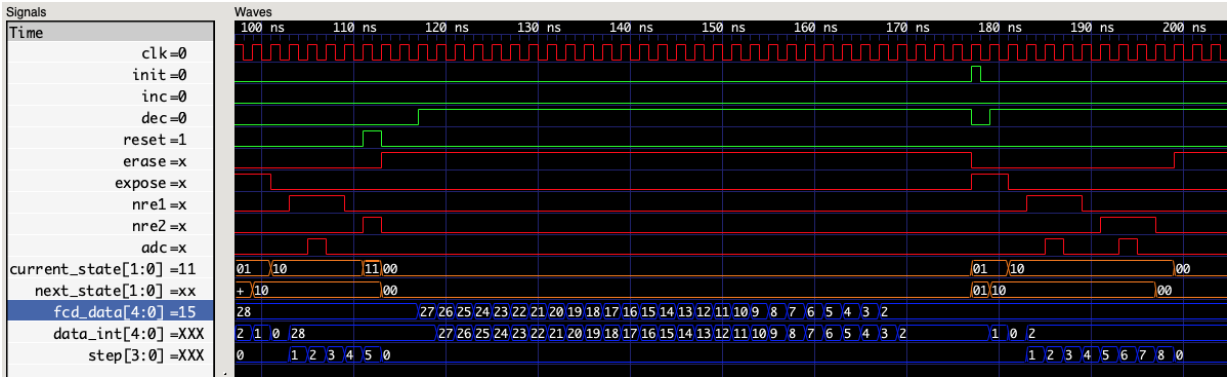

Figure 16: Digital test of the control circuitry part 2

7 Discussion

The results from the simulations shown in Section 6 show that the camera operates according to the desired behaviour described in Section 4 and 5.

It does however demonstrates a big weakness in that the amount of ADCs required scales with the square root of the number of pixels, a different design could easily yield either lower productioncost or shorter time between pictures. Other than that future improvements to this particular project would probably focus on fine tuning and scaling.

Since the analog design in Appendix B is written modularly, a natural continuation of the project would be to enable color as well as increase the resolution. The project would allso benefit from a more scalable and maintanable rewrite of the digital design in Appendix C.

8 Conclusion

This project has shown that dividing the camera into separate analog and digital parts is a good and efficient way of developing a design. It has allso been shown that a simple 4-pixel black and white camera can be implemented following this method.

The project has also developed many of the necessary buildingblocks for designing more complex digital cameras.

9 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] B. B. Larsen, 'TFE4152 - Design of integrated circuits 2019 Project description Digital camera', 2019.

[5] 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.

[6] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson, Mar. 2010, ISBN: 978-03-2154-774-3.

[7] T. Nordgård-Hansen, 'Aimplot', A frontend for AIMSpice, send a mail if you need a copy, 2019.

[8] (2019). Gtkwave, [Online]. Available: `gtkwave.sourceforge.net` (visited on 11/11/2019).

Appendices

A Schematics

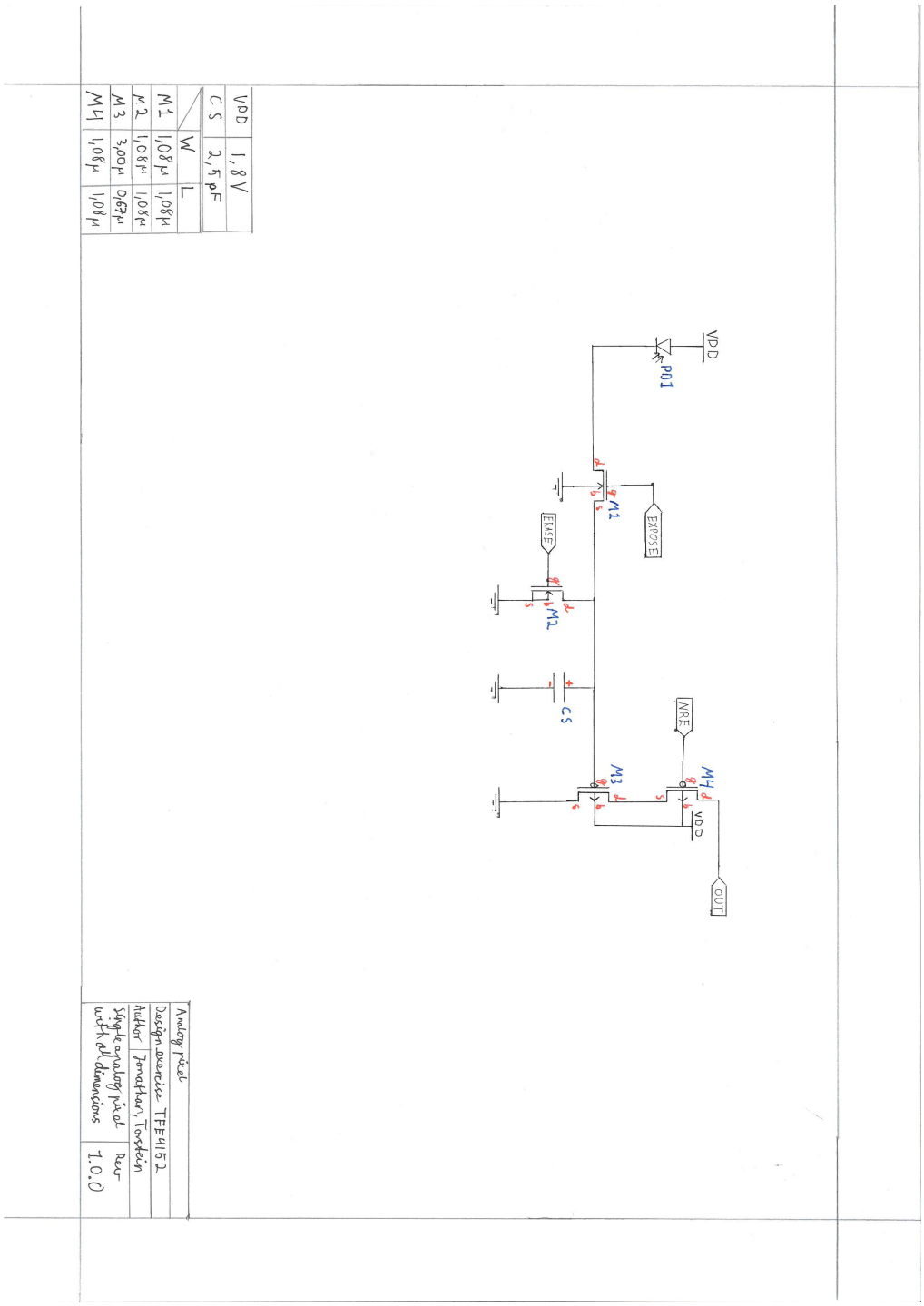

Figure 17: Analog schematic of one pixel

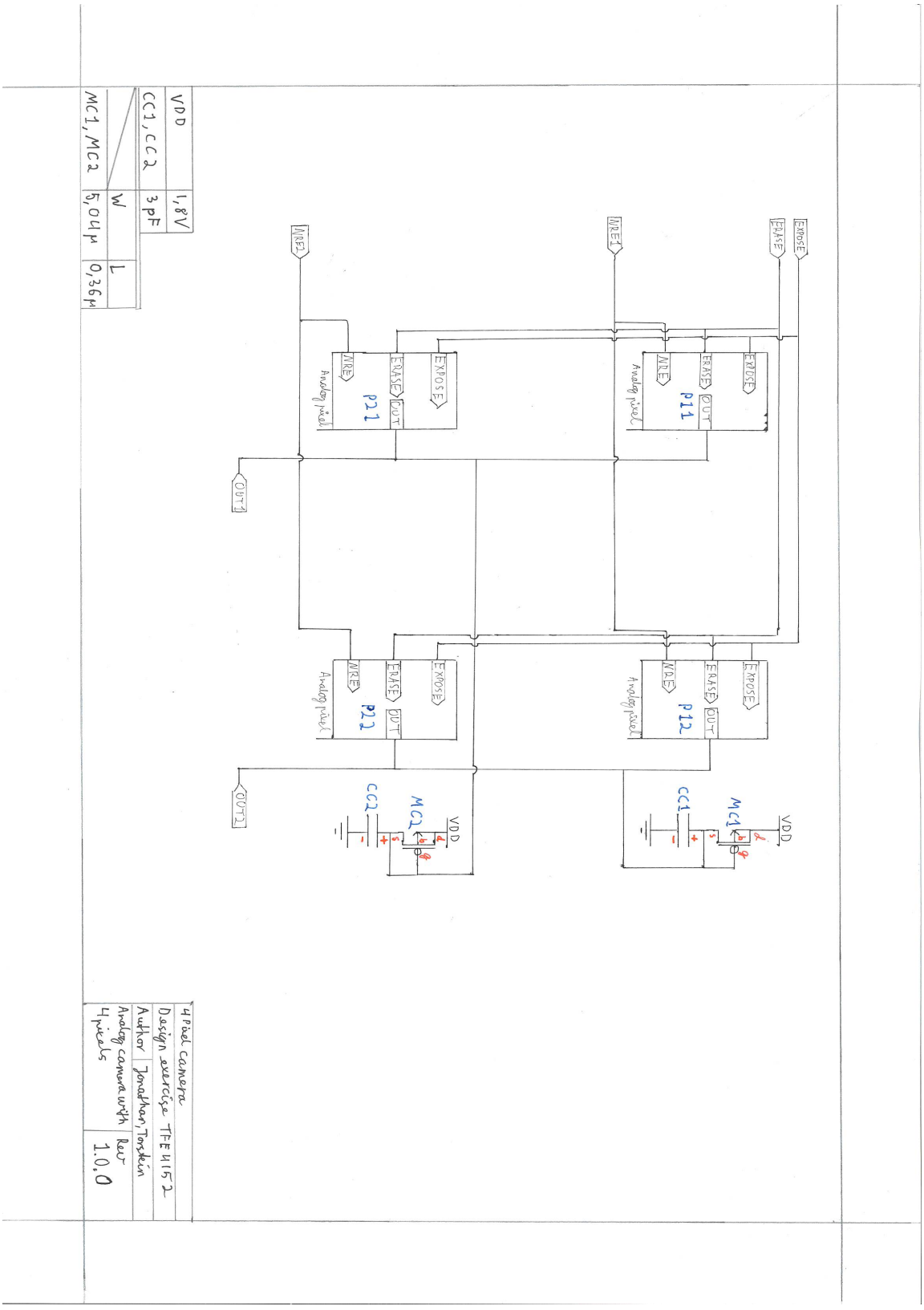

Figure 18: Analog schematic of 4 pixels in a camera

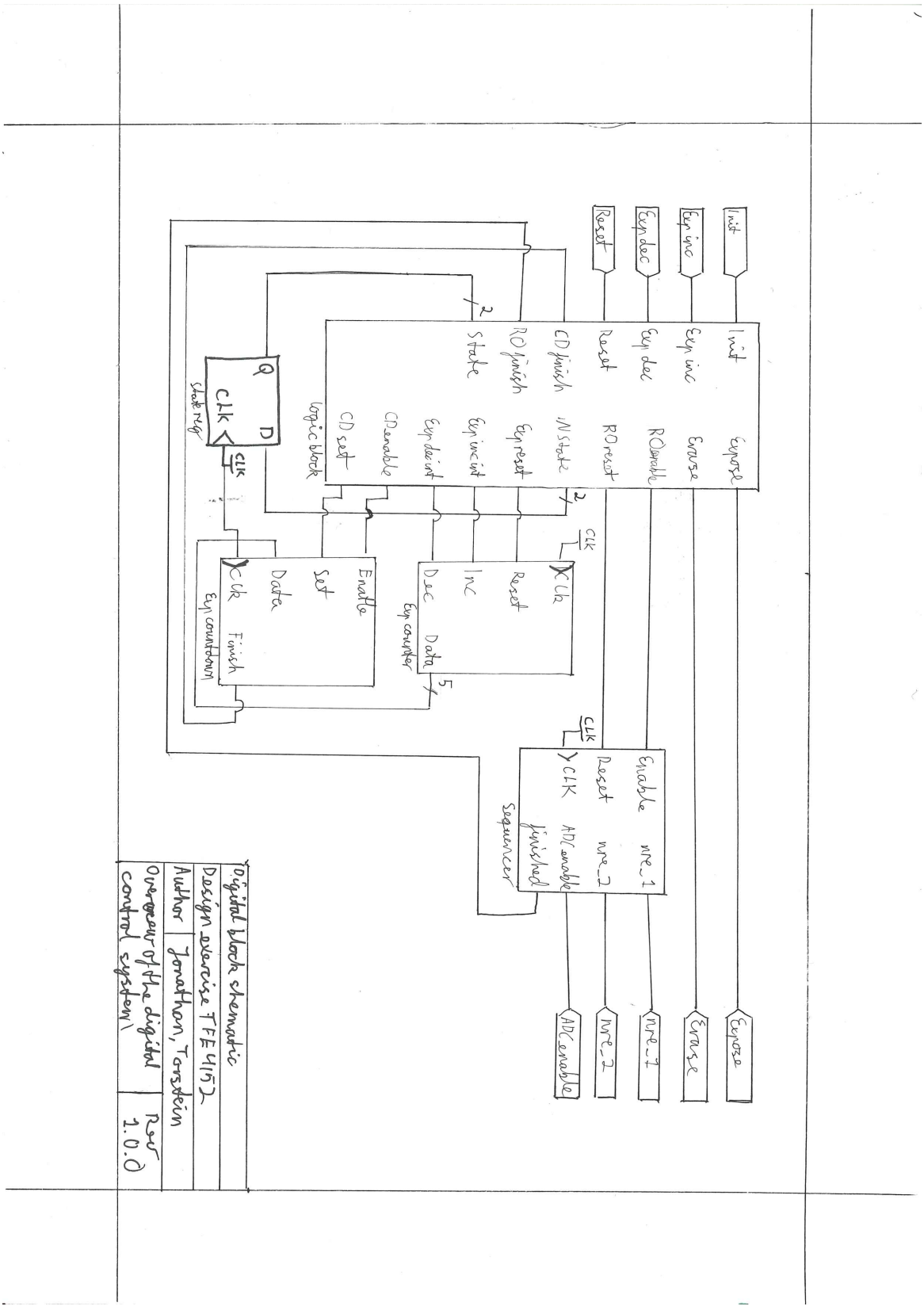

Figure 19: Digital schematic of the camera control

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 = 400p ! Photodiode current, range [50 pA, 750 pA]
.param EXPOSURETIME = 3m ! 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

```
/*
 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;

```

```
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,
                         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 9: 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 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
       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_
```