NTNU
Faculty of Information Technology
and Electrical Engineering
Department of Electronic Systems

TFE4152 - Design of integrated circuits

Digital camera module

Group 1

Authors:
Einar Advem
Øyvind Skaaden

November, 2020

Abstract

This is considered the end result of the semester project in the course TFE4152 Design of Integrated Circuits, at NTNU.
This report includes the applied theory and the design process of a CMOS implementation of the digital control circuit as well as the analog readout circuit for a four pixel digital camera.
The report is intended for readers with knowledge within electrical engineering and/or CMOS technology, but the basic concepts could be understood by anyone.
The simulation verifies the intended functionality of both the analog and digital part of the camera, but has yet to simulated together in a mixed signal simulation as well as being implemented and tested in actual hardware.

All source code and the source of this document can also be found in out GitHib reposiory.

https://github.com/oyvindskaaden/TFE4152-DIC-Project.

Contents

List of Figures

List of Tables

Abbreviations

CMOS	Complimentary metal-oxide-semiconductor
HDL	Hardware Description Language
SPICE	Simulation Program with Integrated Circuit Emphasis
NMOS	N-channel metal-oxide-semiconductor
PMOS	P-channel metal-oxide-semiconductor
FF	Fast Fast, process corner in CMOS production
SS	Slow Slow, process corner in CMOS production
SF	Slow Fast, process corner in CMOS production
FS	Fast Slow, process corner in CMOS production
TT	Typical Typical, process corner in CMOS production
ADC	Analog to Digital Converter
VLSI	Very Large Scale Integration
FSM	Finite State Machine
FET	Field Effect Transistor

1 Introduction

To capture digital representations of images digital cameras are used. As these cameras are becoming more advanced while reproducing better images, they are still required to shrink in physical size.
CMOS technology can be a solution to the challenges faced when shrinking physical size while increasing the performance.

This project report completes a theoretical CMOS implementation of a digital camera, with both an analog readout circuit and a digital control module. The analog circuit is simulated and verified using SPICE simulations, while the digital is implemented using the HDL SystemVerilog.

2 Theory

The objective of this chapter is to introduce the reader to the applied theory that is needed for the implementation of a readout- and controll circuit for a digital camera.

2.1 Field effect transistors

Field effect transistors are a fundamental building block of modern electronics design. As they are able to be produced quite small, they are by far the most used components in integrated circuit design. This is also due to the fact that they are versatile and suitable for alot of applications such as signal amplification and digital logic [1].
The most common type of FETs are NMOS and PMOS. These are transistors created by silicone dioxide, where the substrate, two heavily doped regions, and a gate electrode creates the device terminals. For NMOS the substrate is of p-type and the terminals are made up of n-doped regions. The opposite is true for PMOS, n-type substrate and p-doped regions make up the terminals. This similar structure with opposite regions, make the NMOS and PMOS complimentary as the polarities are opposite of eachother [1].

2.1.1 CMOS

Complementary MOS (CMOS) is a technology that uses both NMOS and PMOS transistors to create digital and analog integrated circuits. CMOS is the most widely used IC technology, as it has taken over many applications that for a long time, only were possible by bipolar devices. As well as the previously mentioned versatility [1].

As mentioned in section 2.1 transistors are made up of heavily doped regions and a gate electrode. To induce a channel in a PMOS transistor, a negative voltage with a higher magnitude than a threshold needs to be applied to the gate terminal. This forms a current channel between the source and drain terminal [1].
The condition can be described as:

$$|V_{gs}| \geq |V_t| \tag{1}$$

To make a current i_D flow between Drain and Source, a negative voltage v_{DS} is connected to the drain terminal [1].
The mobility of the electrons in the channel follow the following relation:

$$\mu_p C_{ox} \tag{2}$$

And the transconductance parameter of the transistor k_p, which is proportional to the aspect ratio of the transistor:

$$k_p = \mu C_{ox} \frac{W}{L} \tag{3}$$

2.2 Process variations

CMOS parameters are highly dependent on the fabrication process. This means that the device properties will vary with each fabrication, even if the design is identical. The fabrication temperature is a typical parameter which can alter the device performance. Oxide thickness may vary by 5% and dopant concentrations may vary by 10%, due to the temperature variations, although efforts are made to reduce them [2].
To compensate for process variations during design, several different device models are used. Corner analysis is used to check how "slow" or "fast" transistors affect the design. Corner types are FF (Fast Fast), SS(Slow Slow), SF(Slow Fast), FS (Fast Slow) and TT(Typical Typical) [2].

2.3 CMOS image sensors

CMOS has many powerful applications, and one of them is in image sensors [3]. Which is the main objective of the CMOS technology used in this project.

When visible light hits the surface of a doped silicon device, a number of electrons proportional to the flux density of the light and wavelength, is released into the silicon device. This phenomena is used in CMOS image sensors to collect electrical information about the light received on the silicon surface [4].

The way a CMOS image sensor is constructed is by a photodiode, i.e. the silicon surface mentioned earlier and a receiving circuit. Here the electrons are firstly stored in a potential well, before either being converted in to a voltage or sent to a metering register. After this the voltage gets passed to an analog-to-digital converter (ADC), which stores the color and intensity of the light as a discrete value, which can be presented on a digital screen [4].

2.4 SPICE

Simulation Program with Integrated Circuit Emphasis (SPICE), is a fairly old and open-source simulation language for analog circuits. It was developed in the 70s at Berkeley university in California. Today it is more commonly used for small integrated sub-circuits or discrete circuits, as Very Large Scale Integration (VLSI) requires faster simulation, and SPICE is relatively slow [5]. SPICE is a very powerful simulation tool since it is able to simulate, DC, AC, transient and noise. It is also simple to add device models for specific components. For CMOS designs simulations can be done with all process corners, as circuit performance may vary for the corner types the production has. [5].

2.5 HDL - Hardware Description Language

Since manual design of logic circuits is not feasible for large designs as the complexity becomes incomprehensible, digital designers use computer-aided tools. Hardware Description Language (HDL) is the most common way of designing, as these languages describe the hardware of digital circuits in a textual form. As simulation and verification plays a major role in HDL, the risk of producing a faulty design is significantly reduced [6].
The way HDL describes a circuit is done implicit by describing the signals and not the actual function, for example the logical output of an AND gate is described by how it is determined by the input [6].
The most common HDLs are VHDL and verilog, [7] [8]. Both of these languages provides the user with a high level description of digital design. The release of verilog 2005, provided a superset of verilog called SystemVerilog [9]. The digital design of this project was done using SystemVerilog.

3 System overview

The system consists of a analog and a digital part.

3.1 Analog

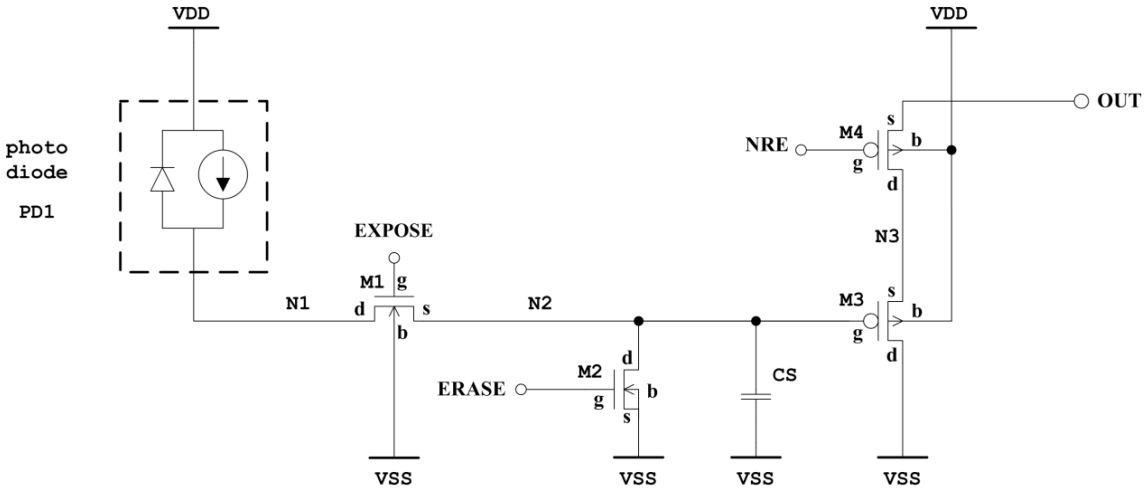


Figure 1: Schematic of the analog readout part of the digital camera

The analog part converts the electrons collected on the surface of the photodiode, in to an analog voltage. This voltage will charge up a capacitor depending on the lighting condition, for example low light condition will use longer time to charge capacitor. The expose transistor controls how how long the capacitor can be charged for. The erase transistor controls the readout circuit so it does not overexpose the pixels. The two output transistors generate a current mirror which drives the output when a pixel is stored.

To create a camera with more than one pixel, the pixels are connected in a matrix, shown in fig. 13 in appendix A.1.

3.2 Digital

The digital part controls the active pixel row, the exposure and the erase function of the analog part.

From fig. 2 the four different states in the finite state machine (FSM) are presented. The FSM describes how the control logic captures a picture. Starting with IDLE, the analog part is held stationary. When a picture is initialized, the state changes to EXPOSURE. The pixels should be exposed for a set amount of time (until `ex_done`, before continuing to the READOUT state. The READOUT consists of pulsing the pixel rows and controlling the ADC to read out the analog voltage to a digital value. After the readout is done, the state is set back to IDLE.

The ILLEGAL state is used to reset the digital and analog parts, before going back to IDLE state.

All the timing of the FSM and states are synchronous, using a clock frequency of 1 kHz.

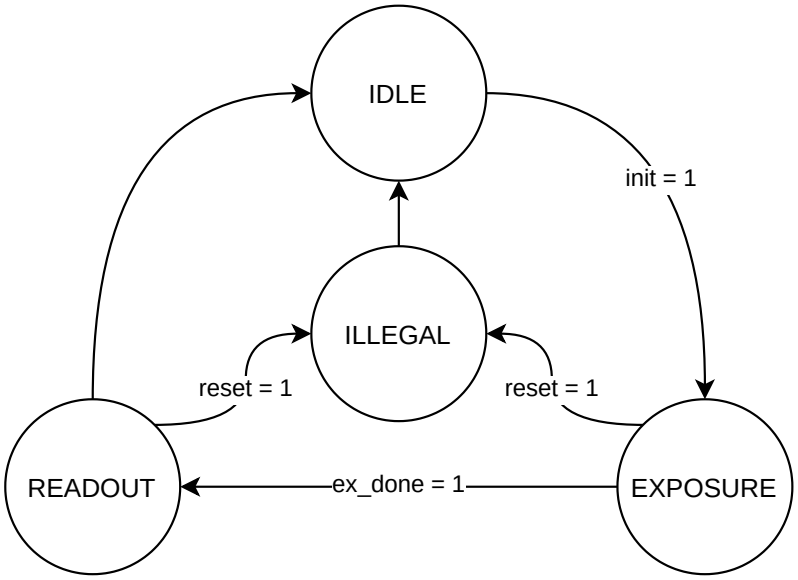


Figure 2: The different states of the control logic.

4 Design process

4.1 Analog

The analog part of the circuit was designed and verified using SPICE simulations in AIM-SPICE. As the device models where provided, the task was to develop the SPICE nets and simulation stimuli.
To allow a low voltage drop over the transistors, the length of the NMOS transistors where picked to be as short as possible, while maintaining the project specification, as shown in table 1. The width of the NMOS was therefore picked as wide possible. The PMOS transistors where picked to be equally as strong as the NMOS transistors. Therefore the PMOS lengths where picked to be equal, but the widths should be slightly larger due to the mobility relation between the transistors.

Transistor	L	W
M1	0.40 µm	1.20 µm
M2	0.40 µm	1.20 µm
M3	0.40 µm	4.63 µm
M4	0.40 µm	4.63 µm

Table 1: Transistor dimension specifications

As both CC1 and CC2 where given, the only capacitor that had to be chosen was CS. The value of CS was picked to be large enough to be charged during exposure, as well as not being to large as that would reduce the speed of operation. Capacitor values are shown in table 2.

Capacitor	Capacitance
CS	2 pF
CC1	3 pF
CC2	3 pF

Table 2: Capacitor specifications

4.1.1 Corner analysis

To demonstrate process variation, a simulation for an NMOS switch was done using AIMSPICE. The simulation covered some of the corner types mentioned in section 2.2. The following corners where simulated: TT, FF and SS.

4.2 Digital

The digital control logic is designed entirely with SystemVerilog, which is a superset of Verilog HDL. Figure 2 describes how the FSM in the control logic should behave, and fig. 3 describes the top level logic block.

There are three logic blocks inside the top level block, `FSM_ex_control`, `CTRL_ex_time` and `timer_counter`. The main control block and FSM is the `FSM_ex_control` block. The FSM is implemented with a clock triggered switch statement. Each time this is triggered, it checks the current state, and operates according to what the state should do. When the FSM is in the EXPOSURE state, the two other blocks are used. The `CTRL_ex_time` block, only has a counter with how many milliseconds the exposure should last. While `timer_counter` uses a downward counting counter, to keep track of the exposure time. When done the `ex_done` flag is set high, so that the FSM can continue to the next state which is READOUT.

In READOUT there is another smaller and simpler FSM, just to clock out the various control signals for the pixels and pixel array.

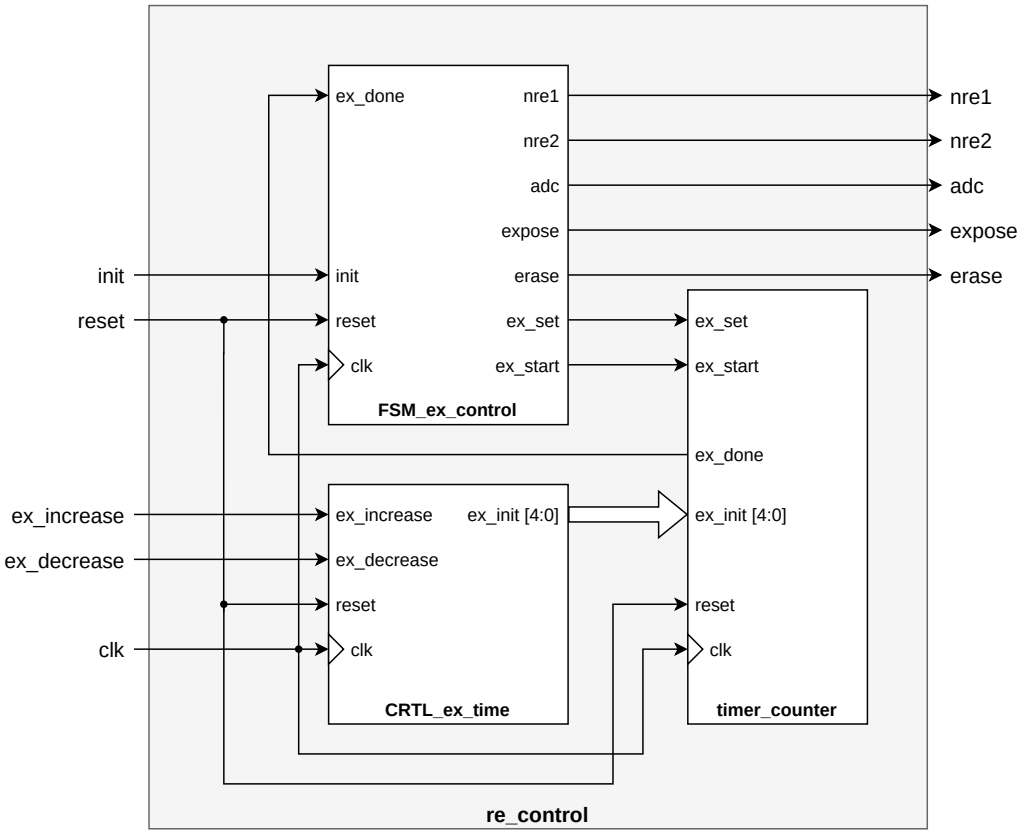


Figure 3: Block diagram for the top level digital block. Top level contain the three smaller blocks that actually contain the logic.

When readout is done, the FSM returns back to the IDLE state.

The purpose of the ILLEGAL state is to ensure that all the control signals and the two other blocks are set to their proper stationary state. For instance if you are in the middle of an exposure, and suddenly trigging `reset` to 1. The ILLEGAL state will then ensure that the camera FSM can start properly from IDLE.

5 Results

The code for analog simulation is listed in appendix B. Code for digital control logic is listed in appendix C. All the plots were created with the MATLAB code in appendix D.1.

5.1 Analog

The analog readout circuit was simulated thoroughly with different exposure voltages to simulate multiple lighting conditions. In fig. 4 the simulation result at 2ms exposure time is presented. In the first and second plot pane the erase and exposure voltages are presented. Both of these voltages influence the sampled voltages which are presented in the third plot pane. This plot pane shows the behaviour of the capacitor at different photodiode currents 200pA, 250pA, 500pA and 750pA. The Fourth plot pane shows the read enable transistor voltages in the same simulation. These are active low, meaning that they will have a voltage drop when activated. The final plot plane shows the circuit output voltages, which are thought to be sent to an ADC.

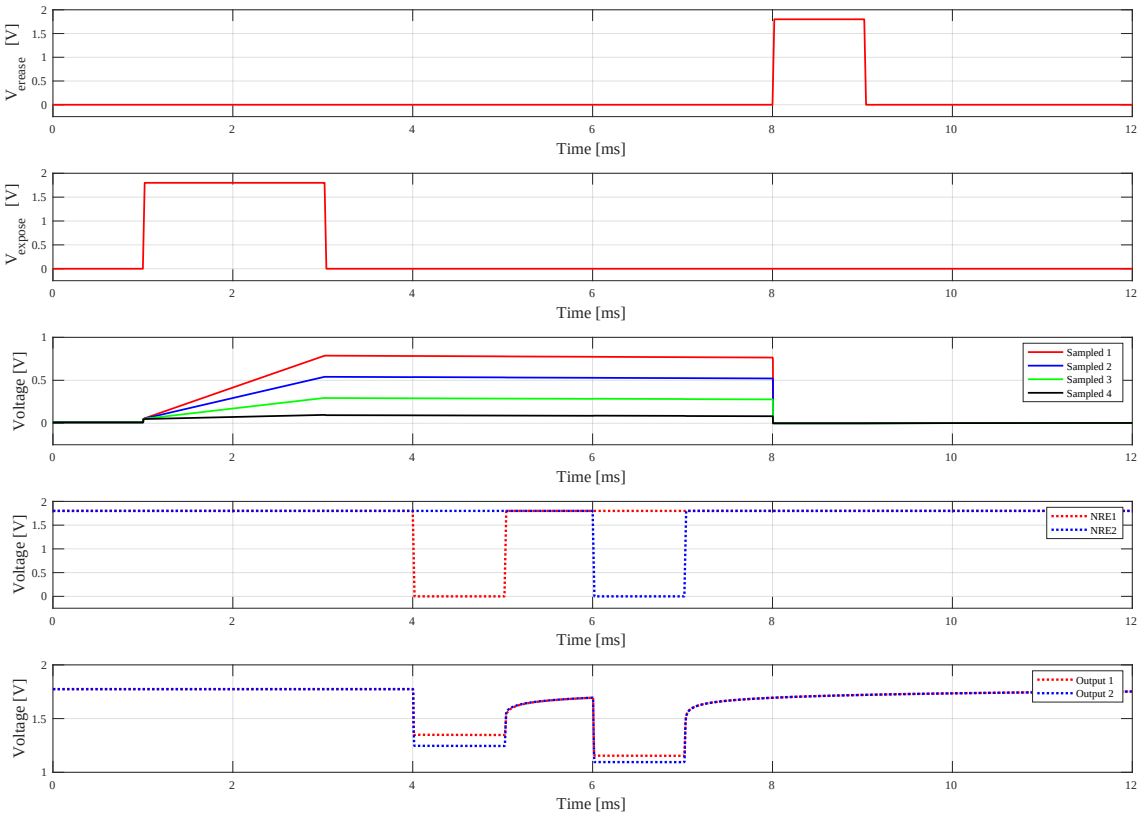


Figure 4: Node voltages at 2ms exposure time, under different photodiode currents

The same node-voltages are simulated and plotted in fig. 5 at a longer exposure time of 30ms, where it shows that the lower photo diode currents will saturate the capacitor. The same photo diode currents as before are simulated here i.e. 200pA, 250pA, 500pA and 750pA.

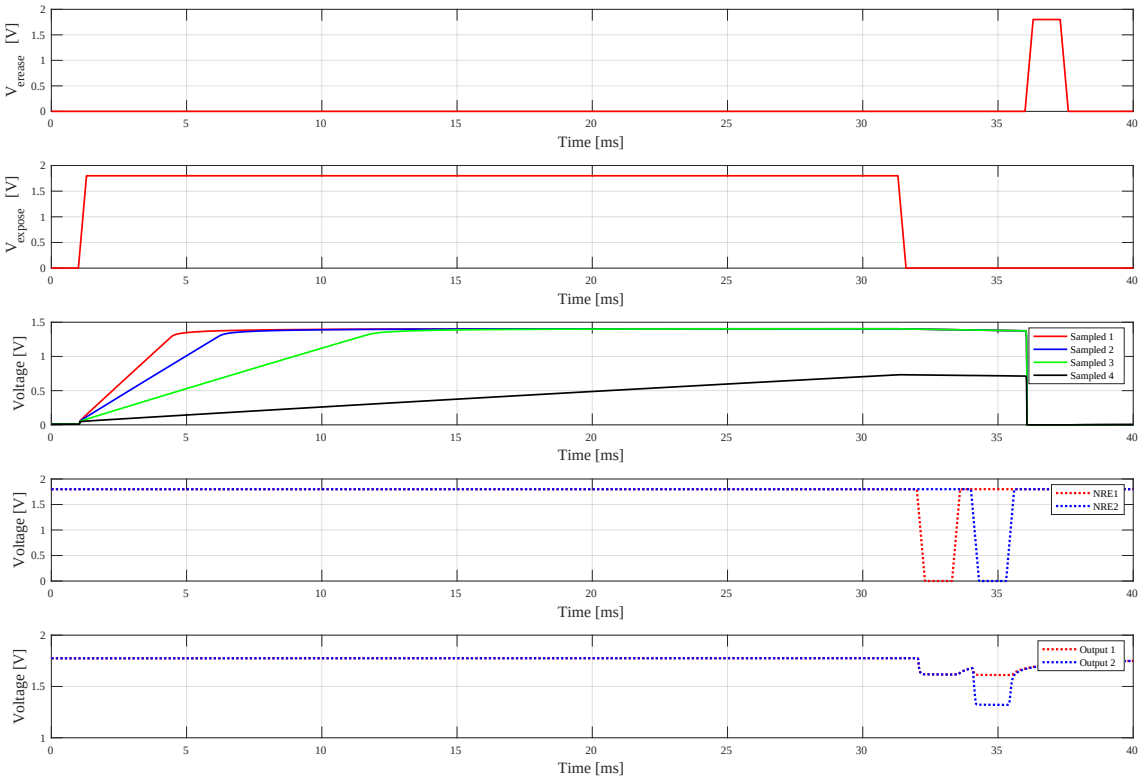


Figure 5: Node voltages at 30ms exposure time, under different photodiode currents

Figure 6 shows the same simulation as fig. 4, but a closer look at the capacitor voltage response to the exposure voltage and erase voltage. It shows that the exposure voltage charges the capacitor up to a certain voltage potential, dependent on the diode current and exposure time. It also shows that when the erase transistor receives its voltage, the capacitor discharges, and is ready for a new exposure.

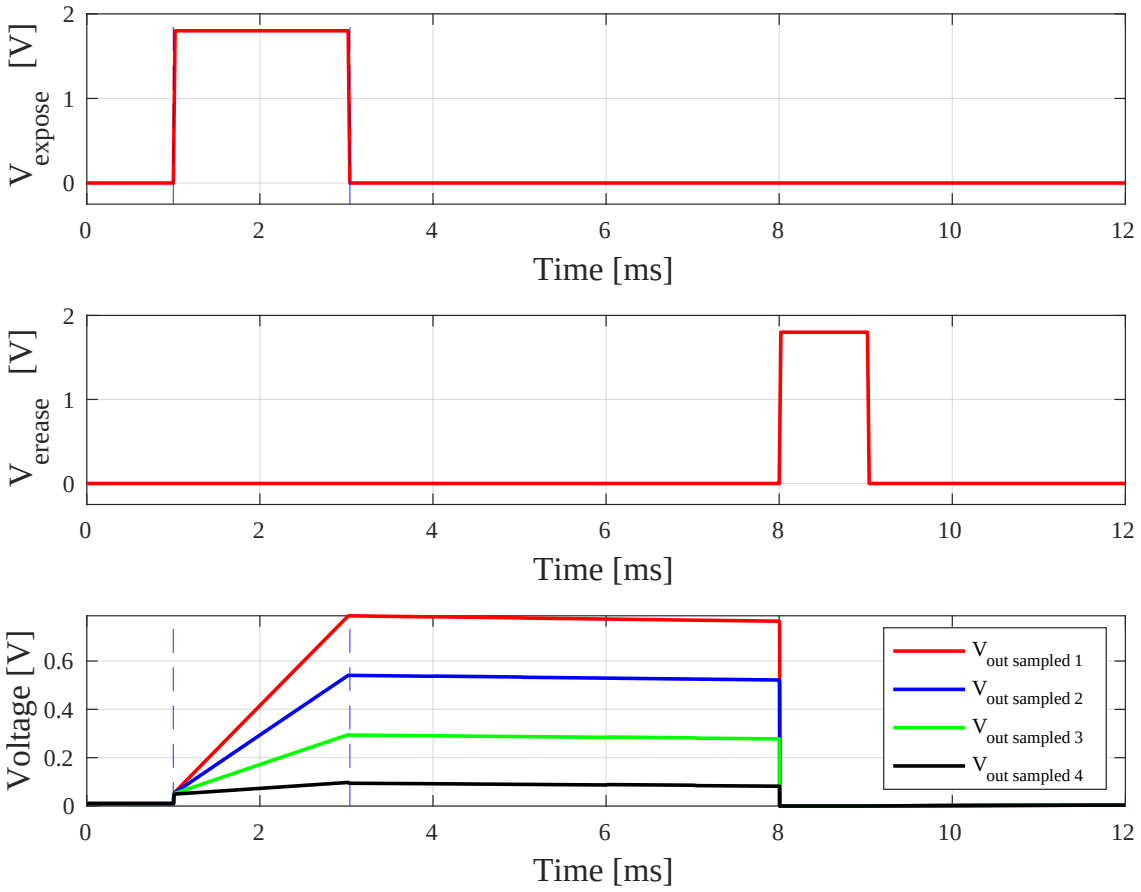


Figure 6: Capacitor voltages vs. control voltages

Figure 7 shows a closer look at the capacitor voltages at 2ms exposure under the four different lighting conditions, i.e. diode currents.

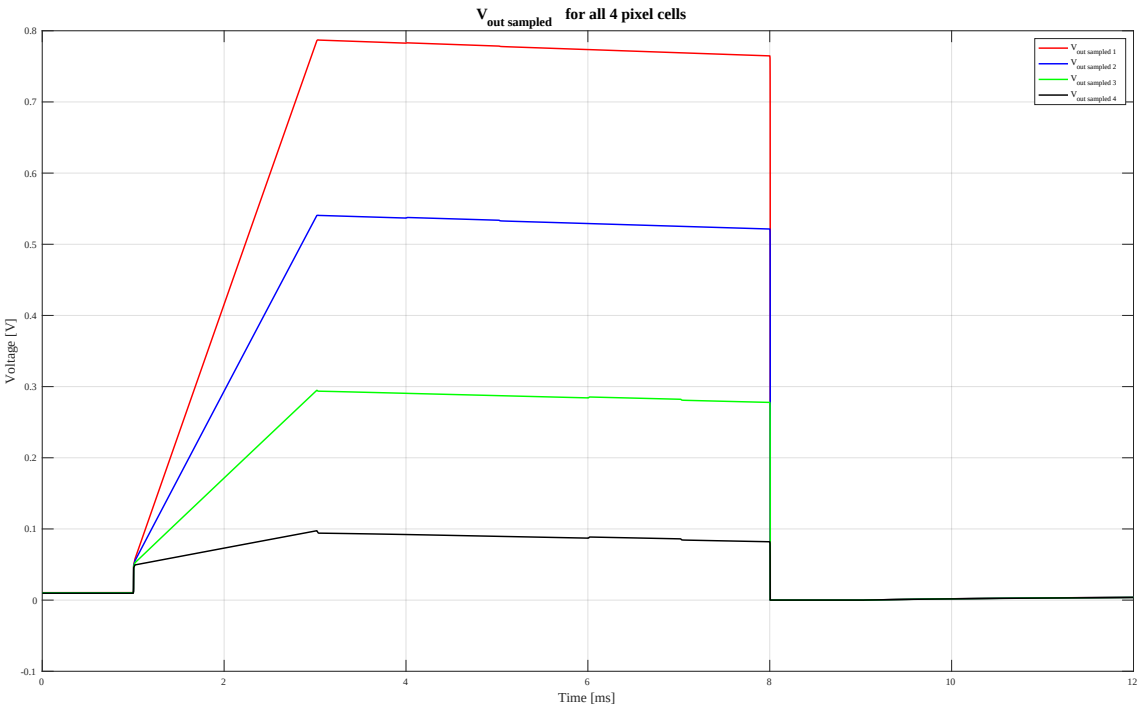


Figure 7: Capcitor voltages under different lighting conditions at 2ms exposure time

5.1.1 Corner analysis

As mentioned in section 4.1.1, a corner analysis of an NMOS switch was done. The transconductance of an NMOS where plotted using three different process corners, TT, FF and SS. The SPICE net used for this simulation is included in appendix B.5

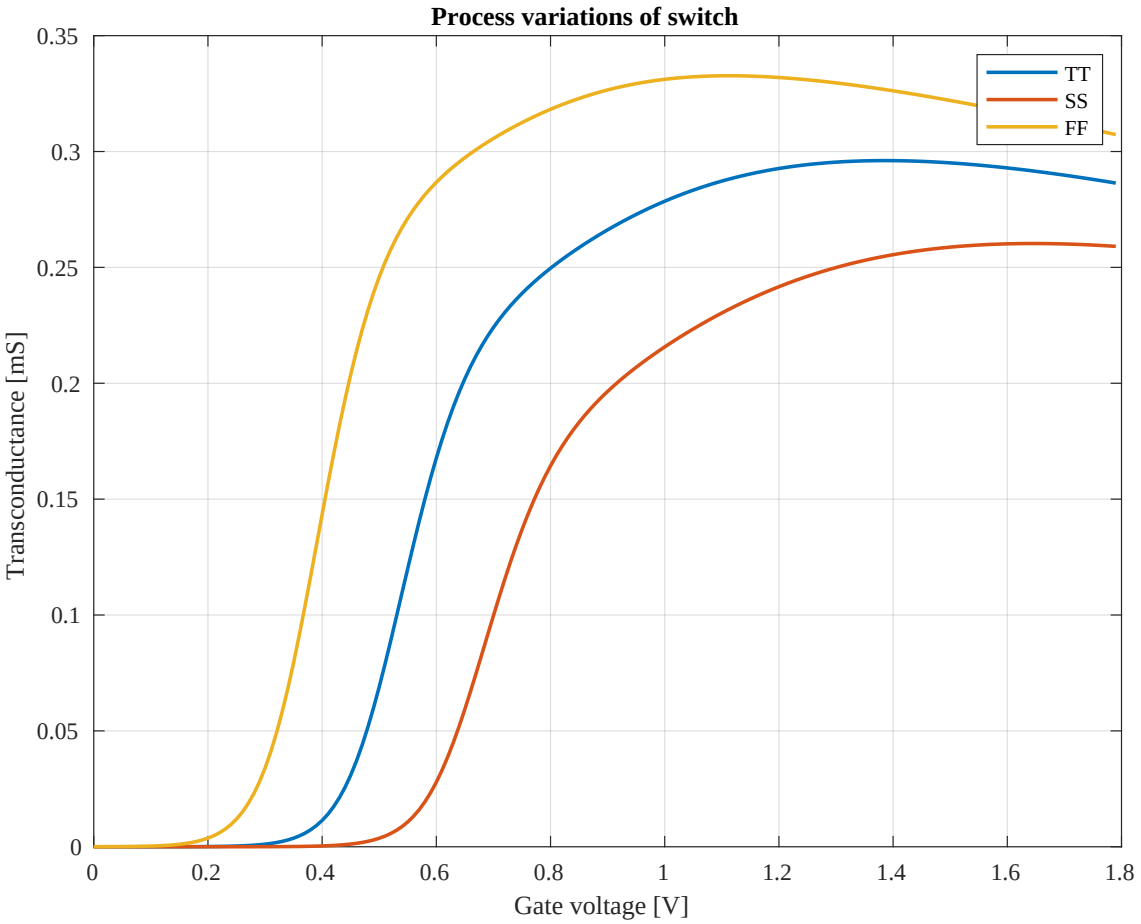


Figure 8: Corner analysis of NMOS switch

5.2 Digital

All verilog code was compiled with Icarus Verilog [10], `iverilog`. To compile with SystemVerilog 2005 use the flag `-g2005-sv`.

Figure 9 shows the timing diagram for the `timer_counter` block. To set the initial value to count from, it is shown set the `ex_set` high. When `ex_start` is set high the timer start counting down to 0. When it reaches 0 the signal `ex_done` is set high.

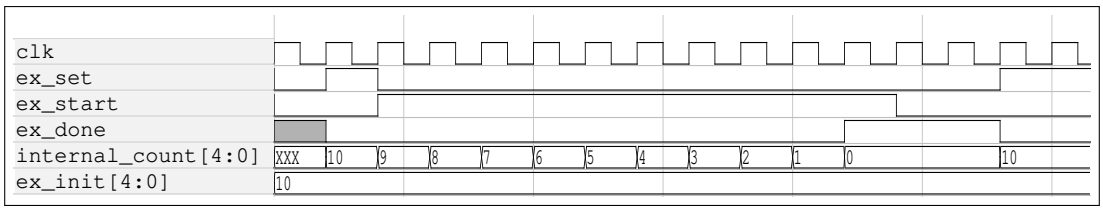


Figure 9: Timing diagram for the `timer_counter` block. Larger version in appendix A.2

Figure 10 shows the timing diagram for the `CTRL_ex_time` block. When `ex_increase` is set high,

it will increase the exposure time by one for each clock cycle. Similar for `ex_decrease`, but it decreases the exposure time.

There is a maximum and minimum value of 30 ms and 2 ms respectively. Both inc/decrease and the limits are tested and shown in fig. 10. When `reset` is set high, the initial value is set to 16, which is in the middle of 30 and 2.

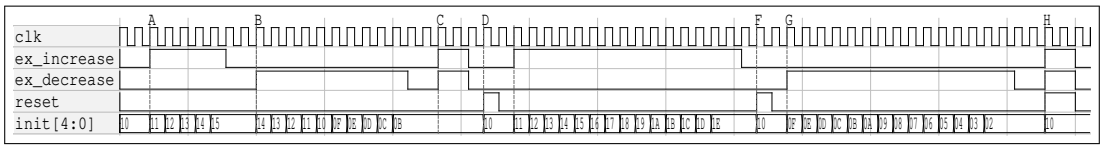


Figure 10: Timing diagram for the `CTRL_ex_time` block. Larger version in appendix A.3

In fig. 11, the timing diagram for the main logic block, `FSM_ex_control`, is shown. The FSM is reset when `reset` is high. This happens at `A`. At `B` we have a state change into EXPOSING. This is done until `ex_done` is set high. When this happens at `C`, the state is set to READOUT, and the readout sequence for the pixel array is initiated. At `D` and `E` the FSM returns to the IDLE state, finishing the sequence of capturing a picture.

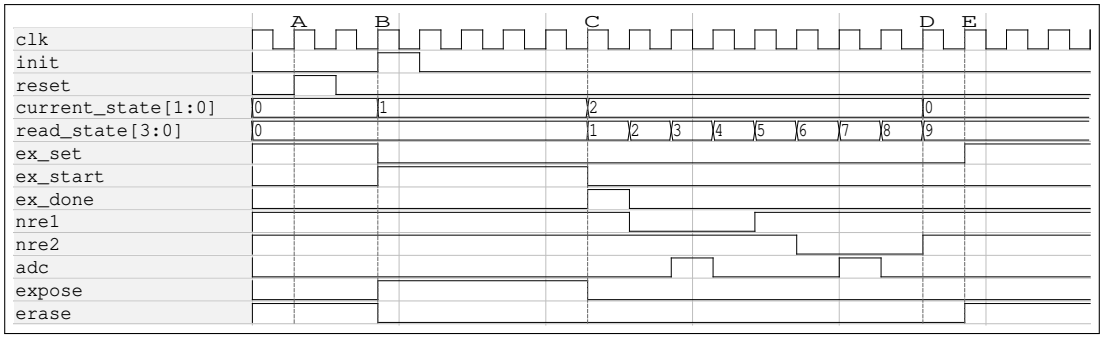


Figure 11: Timing diagram for the `FSM_ex_control` block. Larger version in appendix A.4

Figure 12 is the combined timing diagram for the top level block, `re_control`. When resetting at `A`, we can continue by decreasing the exposure time at `B`. This is done until we have 10 ms exposure. At `C`, we initiate the capture sequence by exposing and starting the exposure counter. At `D` the exposure is done, 10 clock cycles later than initialization (at 1kHz that is 10ms). Following is the readout, same as in fig. 11. At `E` the sequence is done and all values are set back to the initial values and the state is set to IDLE.

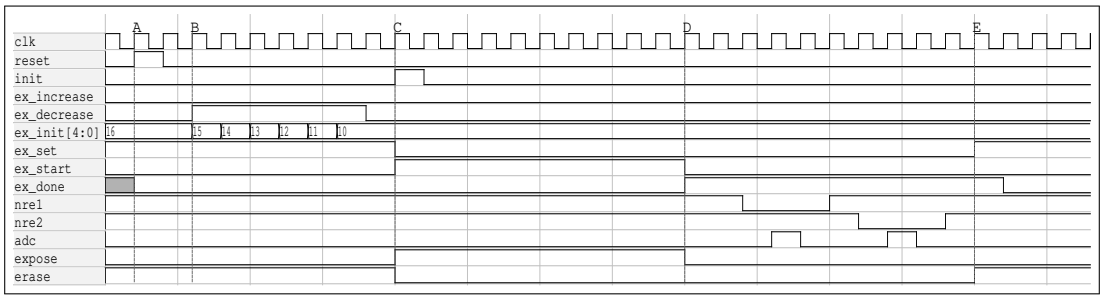


Figure 12: Timing diagram for the `re_control` block. Larger version in appendix A.5

6 Discussion

6.1 Analog

Although the simulations presented in section 5.1 seems quite promising. Seeing as the capacitor gets charged at the different voltages under the different simulated lighting conditions. As well as being discharged when the erase transistor gets activated. The circuit has yet to be tested simulated in a mixed signal simulation. The SPICE simulations have "hard" coded control signals to simulate the digital control, meaning that the interface between the two parts has yet to be tested.

6.1.1 Corner analysis

The process variations are not simulated on the complete analog circuit, only on a single NMOS switch. Therefore there is no reason to rule out that process variations may affect the function and performance of the analog circuit.

6.2 Digital

The digital circuit seems to function after the specification, this means that it works under the same timing as the analog circuit does. However since the verilog HDL is only synthesized and simulated in software and has yet to be implemented in hardware, there is no guarantee that a hardware implementation will work as simulated.

7 Conclusion

Since all simulation results function after specification, the digital camera module is at least theoretically accomplished. Both the analog and digital parts of the system works individually, but has yet to be simulated together in a mixed signal analysis. A simulation tool supporting both analog and digital mixed signal analysis should be used to test this functionality.

Seeing as the module has only been simulated, it is still only theoretically accomplished and has yet to be implemented in hardware. The process variations of the analog circuit may affect a hardware implementation. A hardware implementation of the digital circuit may also not work as the simulations shows.

Bibliography

[1] A. S. Sedra and K. C. Smith, *Microelectronic circuits*. Oxford University Press, 2016.

[2] T. C. Carusone, D. Johns, K. W. Martin, and D. Johns, *Analog integrated circuit design*, 2nd. John Wiley & Sons, 2014.

[3] elprocus, *Cmos technology : Working principle, characteristics & its applications*, Nov. 2020. [Online]. Available: https://www.elprocus.com/cmos-working-principle-and-applications/.

[4] R. Turchetta, K. R. Spring, and M. W. Davidson, *Introduction to cmos image sensors*. [Online]. Available: https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/cmosimagesensors/.

[5] L. w. Nagel, *Is it time for spice4?* [Online]. Available: https://web.archive.org/web/20060926034314/http://www.cs.sandia.gov/nacdm/talks/Nagal_Larry_NACDM2004.pdf.

[6] M. M. Mano and M. D. Ciletti, *Digital design: with an introduction to the verilog hdl*, 5th. Pearson Education Limited, 2013.

[7] S. Arar, *What is vhdl? getting started with hardware description language for digital circuit design - technical articles*, Dec. 2017. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/hardware-description-langauge-getting-started-vhdl-digital-circuit-design/.

[8] A. Steve, *Getting started with the verilog hardware description language - technical articles*, Jan. 2019. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/getting-started-with-the-verilog-hardware-description-language/.

[9] DIGILENT, *Verilog hdl background and history*. [Online]. Available: https://reference.digilentinc.com/learn/fundamentals/digital-logic/verilog-hdl-background-and-history/start.

[10] S. Williams, *Icarus verilog*, Nov. 2020. [Online]. Available: http://iverilog.icarus.com/.

Appendix

A Figures

A.1 Pixel array

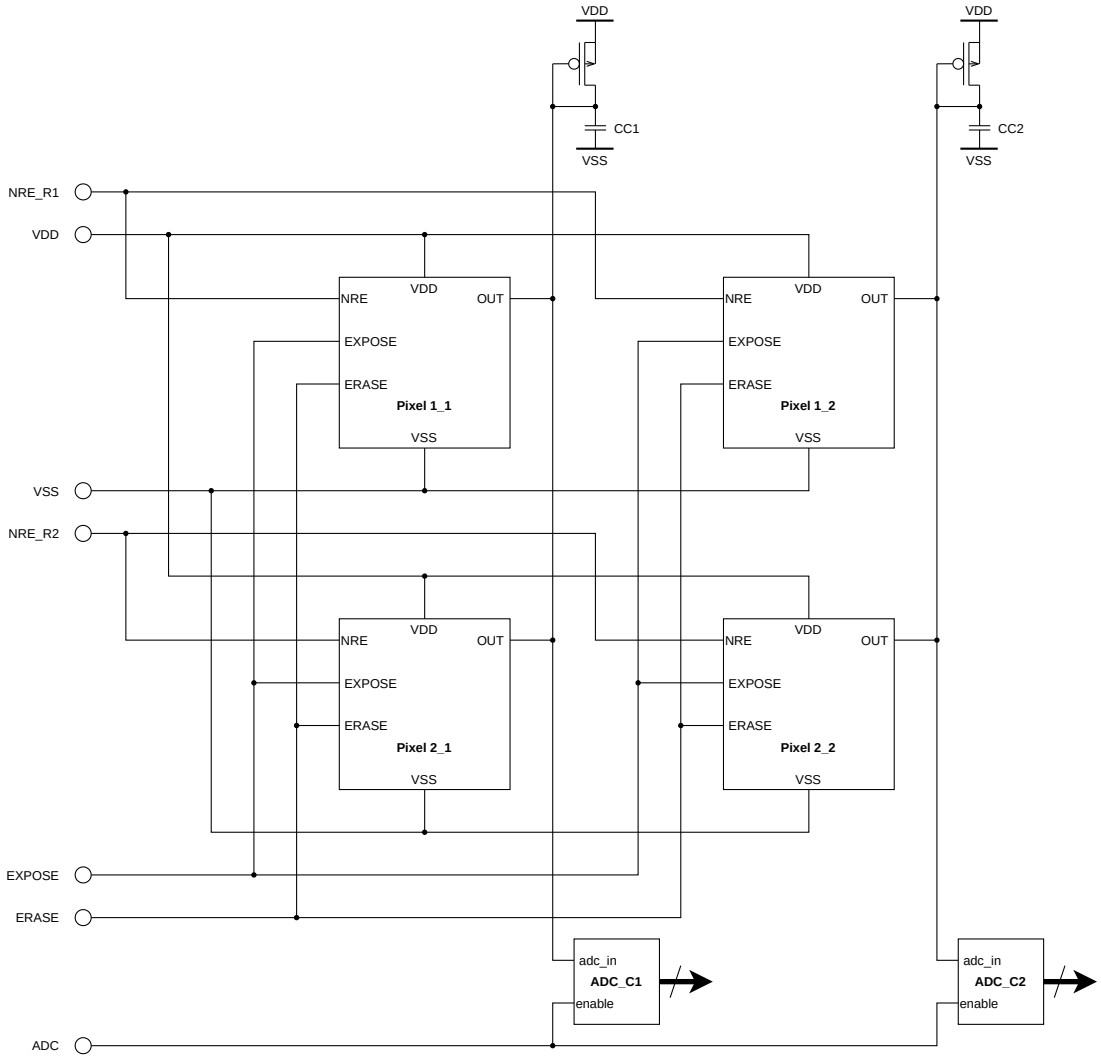
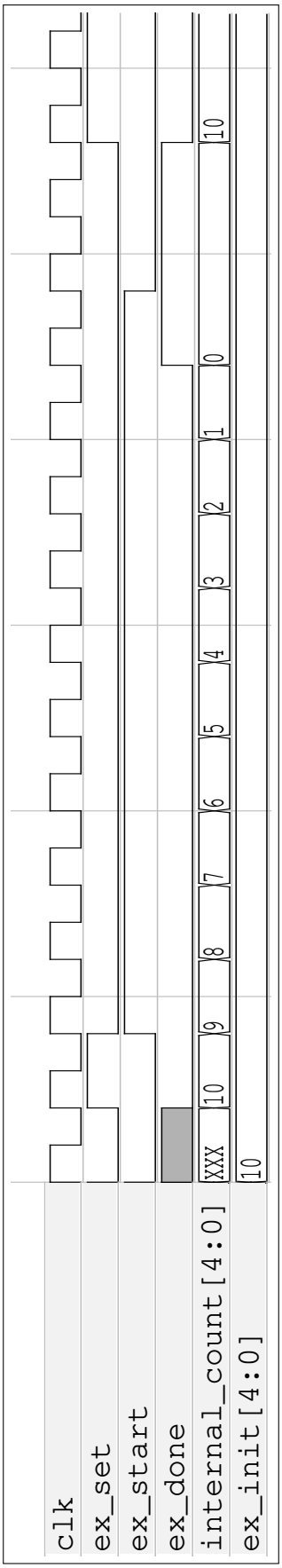
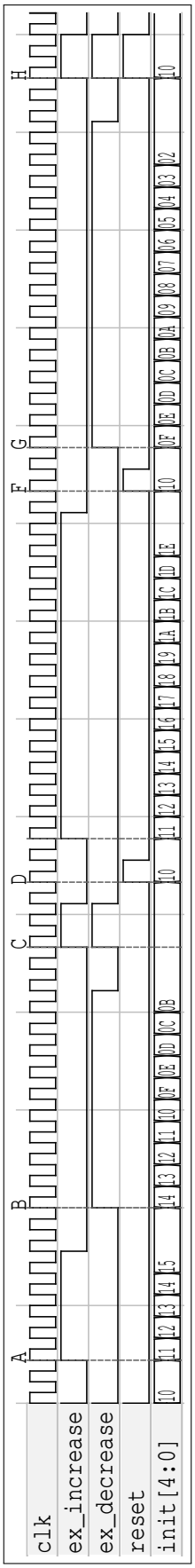


Figure 13: Connection diagram for the pixel array.

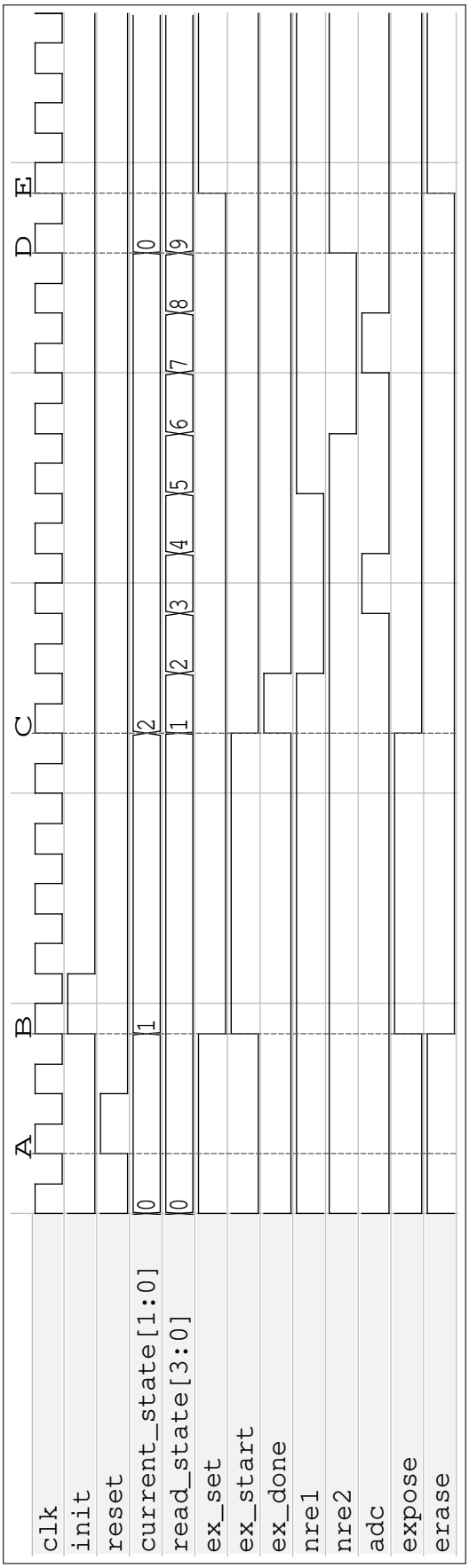
A.2 Timing diagram for `timer_counter`



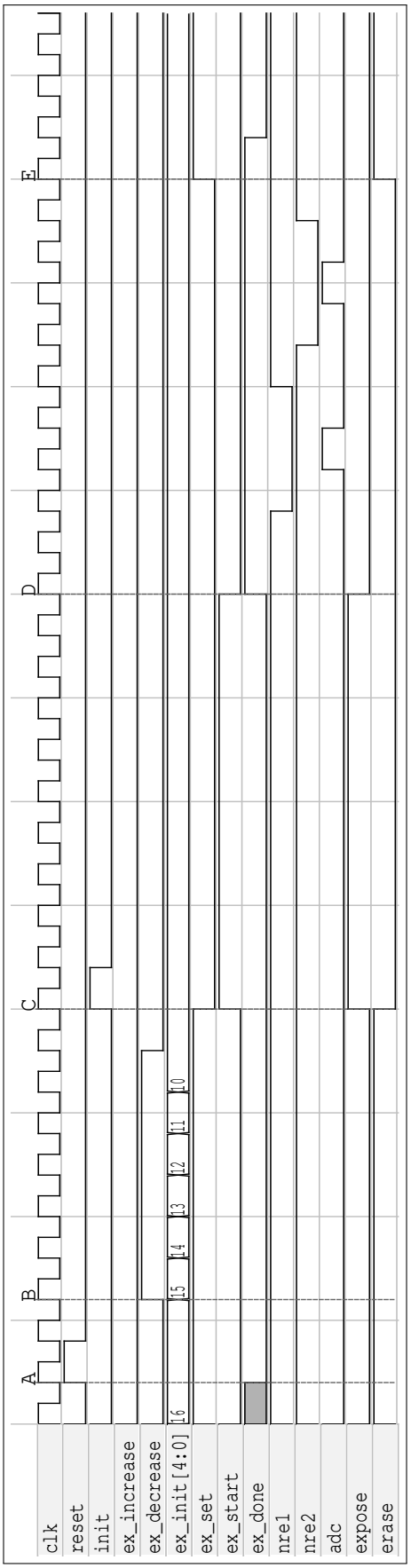
A.3 Timing diagram for CTRL_ex_timer



A.4 Timing diagram for FSM_ex_control



A.5 Timing diagram for re_control



B Analog implementation code

B.1 Testbench.cir

```
* This is a parametrized testbench for your pixel circuit array

* You should at least test your circuit with:
*        - current of 50 pA and exposure time 30 ms
*        - current of 750 pA and exposure time 2 ms

* Instructions
* Connect EXPOSE, ERASE, NRE_R1 and NRE_R2 at the right place
* Run a transient simulation with length 60 ms
* Make sure outputs of pixel circuits to ADC are called OUT1 and OUT2
* Make plots of output voltages to ADC (here called OUT1 and OUT2)
* The voltage across internal capacitor (any pixel) is also of interest (here called OUT_SAMPLED1)
* You should also plot the control signals EXPOSE, NRE_R1, NRE_R2 and ERASE

!!!!!!!!!!!!!!!!!!!!!!
!!!!! PARAMETERS !!!!!
!!!!!!!!!!!!!!!!!!!!!!
.param Ipd_11       = 750p  ! Photodiode current, range [50 pA, 750 pA]
.param Ipd_12       = 500p  ! Photodiode current, range [50 pA, 750 pA]
.param Ipd_21       = 250p  ! Photodiode current, range [50 pA, 750 pA]
.param Ipd_22       = 50p   ! Photodiode current, range [50 pA, 750 pA]
.param VDD          = 1.8   ! Supply voltage
.param EXPOSURETIME = 2m    ! Exposure time, range [2 ms, 30 ms]

! Automaticly calculate all the other parameters
.param TRF          = {EXPOSURETIME/100}             ! Risetime and falltime of EXPOSURE and ERASE signals
.param PW           = {EXPOSURETIME}                 ! Pulsewidth of EXPOSURE and ERASE signals
.param PERIOD       = {EXPOSURETIME*10}              ! Period for testbench sources
.param FS           = 1k;                            ! Sampling clock frequency
.param CLK_PERIOD   = {1/FS}                         ! Sampling clock 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   = {4*CLK_PERIOD + EXPOSURETIME}  ! Delay for NRE_R2 signal
.param ERASE_DLY    = {6*CLK_PERIOD + EXPOSURETIME}  ! Delay for ERASE signal
.param SIM_TIME     = {10*CLK_PERIOD + EXPOSURETIME} ! Delay for ERASE signal


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!! Supplies and pulse generation !!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
VDD     1       0 dc VDD                                                                                    ! General supply
VEXPOSE EXPOSE  0 dc 0  PULSE(0     VDD  EXPOSE_DLY  TRF TRF EXPOSURETIME PERIOD)   ! Expose pulse, active high
VERASE  ERASE   0 dc 0  PULSE(0     VDD  ERASE_DLY   TRF TRF CLK_PERIOD PERIOD)   ! Erase pulse, active high
VNRE_R1 NRE_R1  0 dc 0  PULSE(VDD   0    NRE_R1_DLY  TRF TRF CLK_PERIOD PERIOD)   ! NRE_R1 Pulse, active low
```

```
VNRE_R2 NRE_R2  0 dc 0  PULSE(VDD   0    NRE_R2_DLY  TRF TRF CLK_PERIOD
↪  PERIOD)   ! NRE_R2 Pulse, active low


!!!!!!!!!!!!!!!!!!!
!!!!! Includes !!!!!
!!!!!!!!!!!!!!!!!!!

! Photodiode
.include ./photodiode.cir
! Transistors
.include ./p18_model_card.inc
.include ./p18_cmos_models_tt.inc

! Pixel subcircuit
! syntax: XPixel VDD(in) VSS(in) EXPOSE(in) ERASE(in) NRE(in) OUT(out)
↪  CAP_VOLTAGE(out) PIXEL(name) PARAM: I_PD=VAL
.include ./pixel.cir
! Column capacitor
! syntax: XColCpa VDD(in) VSS(in) COLOUT(out) COLCAPS
.include ./column_caps.cir


!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!! Actual Circuit !!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!
! Pixels follow matrix format
XP11 1 0 EXPOSE ERASE NRE_R1 OUT1 OUT_SAMPLED1 PIXEL I_PD=Ipd_11
XP12 1 0 EXPOSE ERASE NRE_R1 OUT2 OUT_SAMPLED2 PIXEL I_PD=Ipd_12
XP21 1 0 EXPOSE ERASE NRE_R2 OUT1 OUT_SAMPLED3 PIXEL I_PD=Ipd_21
XP22 1 0 EXPOSE ERASE NRE_R2 OUT2 OUT_SAMPLED4 PIXEL I_PD=Ipd_22

! Column caps
XCC1 1 0 OUT1 COLCAPS
XCC2 1 0 OUT2 COLCAPS


!!!!!!!!!!!!!!!!!!!
!!!!! Analysis !!!!!
!!!!!!!!!!!!!!!!!!!
! Do a transient analysis
.tran 5u SIM_TIME 0 10u

! Plot the voltages
.plot V(OUT1) V(OUT2) ! signals going to ADC
.plot V(EXPOSE) V(NRE_R1) V(NRE_R2) V(ERASE)
.plot V(OUT_SAMPLED1) V(OUT_SAMPLED2) V(OUT_SAMPLED3) V(OUT_SAMPLED4)
```

B.2 pixel.cir

```
! Pixel cicuit
! Inputs: Vdd, Vss, EXPOSE, ERASE, NRE
! Output: OUT

.subckt PIXEL Vdd Vss EXPOSE ERASE NRE OUT N2 PARAM: I_PD=750p
!!!!!!!!!!!!!!!!!!
!!!!! Factors !!!!!
!!!!!!!!!!!!!!!!!!
.param  L       = 0.4u  ! Standard gate length
.param  WL1     = 3     ! M1 NMOS gate width factor
.param  WL2     = 3     ! M2 NMOS gate width factor
.param  WL3     = 3     ! M3 PMOS gate width factor, this is also multiplied
↪  with convertion from nmos to pmos
.param  WL4     = 3     ! M4 PMOS gate width factor, this is also multiplied
↪  with convertion from nmos to pmos

! Convertion between NMOS and PMOS
.param NtoPMOS  = {27 / 7}

! Size of sampeling capacitor
.param cs_size  = 2pf

! Actual lengths
.param W1   = {WL1 * L}             ! M1, NMOS
.param W2   = {WL2 * L}             ! M2, NMOS
.param W3   = {WL3 * L * NtoPMOS}   ! M3, PMOS
.param W4   = {WL4 * L * NtoPMOS}   ! M4, PMOS

!.param Ipd = {I_PD}

!!!!!!!!!!!!!!!!!!!!!!
!!!!! Photodiode !!!!!
!!!!!!!!!!!!!!!!!!!!!!
XPD Vdd N1 PhotoDiode Ipd={I_PD}


!!!!!!!!!!!!!!!!!!
!!!!! Expose !!!!!
!!!!!!!!!!!!!!!!!!
M1 N1 EXPOSE N2 Vss NMOS L=L W=W1


!!!!!!!!!!!!!!!!!!
!!!!! N2 Node !!!!!
!!!!!!!!!!!!!!!!!!

! Erase part
M2 N2 ERASE Vss Vss NMOS L=L W=W2

! Sampleling caps
CS N2 Vss cs_size

! M3 Transistor
M3 Vss N2 N3 Vdd PMOS L=L W=W3
```

```
53
54
55 !!!!!!!!!!!!!!!!!!
56 !!!!! N3 Node !!!!!
57 !!!!!!!!!!!!!!!!!!
58
59 ! Out transistor
60 M4 N3 NRE Out Vdd PMOS L=L W=W4
61
62 .ends
```

B.3 photodiode.cir

```
.subckt PhotoDiode  VDD N1_R1C1 PARAM: Ipd=750p
I1_R1C1  VDD   N1_R1C1   DC  {Ipd}              ! Photo current source
d1 N1_R1C1 VDD dwell 1                         ! Reverse biased Diode
.model dwell d cj0=1e-14 is=1e-12 m=0.5 bv=40 ! Diode model
Cd1 N1_R1C1 VDD 30f                            ! Photo diode capacitor
.ends
```

B.4 column_caps.cir

```
.subckt COLCAPS Vdd Vss Out

! Transistor sizes
.param  L       = 0.4u  ! Standard gate length
.param  WL      = 3     ! M1 NMOS gate width factor

! Convertion between NMOS and PMOS
.param NtoPMOS  = {27 / 7}

.param W   = {WL * L * NtoPMOS}   ! Actual width

! Actual circuit
MC1 Out Out Vdd Vdd PMOS L=L W=W
CC1 Out Vss 3pf

.ends
```

B.5 Switch.cir

```
Switch testbench

.param L = 0.18U
.param WL = 3
.param W = { WL * L}

.param Vdd = 1.8
.param Vg = Vdd

VDD 2 0 dc Vdd
VGG 3 0 dc Vg

R1 2 1 100
MN1 1 3 0 0 NMOS L=L W=W

```

```
.dc VGG 0 Vg 0.01

.plot gm(MN1)
.plot V(R1)
.plot I(R1)

.include p18_model_card.inc
.include p18_cmos_models_ss.inc
```

B.6 p18_model_card.inc

```
* 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 = -1.38E-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
```

B.7 p18_cmos_models_tt.inc

```
.param proc_delta = 1.0
.param vt_shift = 0.0
.include p18_model_card.inc
```

B.8 p18_cmos_models_ss.inc

```
.param proc_delta = 0.90
.param vt_shift = 0.15
.include p18_model_card.inc
```

B.9 p18_cmos_models_ff.inc

```
.param proc_delta = 1.10
.param vt_shift = -0.15
.include p18_model_card.inc
```

C Digital implementation code

C.1 re_control.sv

```
`include "CTRL_ex_time.sv"
`include "FSM_ex_control.sv"
`include "timer_counter.sv"

// Top level control block
module re_control(
        input wire clk,
        input wire reset,
        input wire init,

        input wire ex_increase,
        input wire ex_decrease,

        output wire nre1,
  output wire nre2,
  output wire adc,
  output wire expose,
  output wire erase
);

        // Interconnect wires
        wire ex_done;
        wire ex_start;
        wire ex_set;

        wire [4:0] ex_init;

        // Lower level blocks with actual logic
        // Interconnect with wires
        FSM_ex_control fsm_ex_control(clk, reset, init, ex_done, ex_start,
        ↪  ex_set, nre1, nre2, adc, expose, erase);
        CTRL_ex_time ctrl_ex_time (clk, reset, ex_increase, ex_decrease,
        ↪  ex_init);
        timer_counter ex_timer(clk, ex_set, ex_start, ex_init, ex_done);

endmodule // re_control
```

C.2 re_control_tb.sv

```
`include "re_control.sv"

module re_control_tb();
  logic clk;
  logic reset;
  logic init;

  logic ex_increase;
  logic ex_decrease;

  wire nre1;
  wire nre2;
  wire adc;
  wire expose;
  wire erase;

  re_control testbench(clk, reset, init, ex_increase, ex_decrease, nre1, nre2,
   ↪  adc, expose, erase);

  // Clock
  always begin
    #1 clk = !clk;
  end

  initial begin
    $dumpfile("outfiles/re_control/re_control_tb.vcd");
    $dumpvars(0, testbench);

    // Starting with clock high
    {clk, reset, init, ex_increase, ex_decrease} = 5'b10000;

    // Reset after one pulse
    #2 reset = 1;
    #2 reset = 0;

    // Decrease exposure from 16 down to 10
    #2 ex_decrease = 1;
    #12 ex_decrease = 0;

    // Start sequence with init pulse
    #2 init = 1;
    #2 init = 0;

    // Stop simulation
    #46 $finish;
  end


endmodule
```

C.3 CTRL_ex_time.sv

```
module CTRL_ex_time(
  input logic clk,
  input logic reset,
  input logic ex_increase,
  input logic  ex_decrease,
  output logic [4:0] ex_init
);

  always @(posedge clk)
    if (reset)
      ex_init <= 5'd16; // If reset, set it to the middle of range [2,30]
    else if (ex_increase && !ex_decrease && ex_init < 5'd30)
      ex_init <= ex_init + 1; // Increase if ex_increase and ex_init < 30
    else if (ex_decrease && !ex_increase && ex_init > 5'd2)
      ex_init <= ex_init - 1; // Decrease if ex_decrease and ex_init > 2

  // Initial value for ex_init
  initial begin
    ex_init <= 5'd16;
  end

endmodule
```

C.4 CTRL_ex_time_tb.sv

```
`include "CTRL_ex_time.sv"

module CTRL_ex_time_tb();

  logic clk, reset, ex_increase, ex_decrease;
  wire [4:0] ex_init;

  CTRL_ex_time testbench(clk, reset, ex_increase, ex_decrease, ex_init);

  always begin
    #1 clk = !clk;
  end

  initial begin
    $dumpfile("outfiles/CTRL_ex_time/CTRL_ex_time_tb.vcd");
    $dumpvars(0, testbench);

    {clk, reset, ex_increase, ex_decrease} = 4'b1000; // Start with clock on

    // Test counting;
    #4 ex_increase = 1;
    #10 ex_increase = 0;
    #4 ex_decrease = 1;
    #20 ex_decrease = 0;

    // Test if both buttons is high
    #4 {ex_increase, ex_decrease} = 2'b11;
    #4 {ex_increase, ex_decrease} = 2'b00;

    // Test reset
    #2 reset = 1;
    #2 reset = 0;

    // Test of the upper limit
    #2 ex_increase = 1;
    #30 ex_increase = 0;

    // Reset before testing lower limit
    #2 reset = 1;
    #2 reset = 0;

    // Test of the lower limit
    #2 ex_decrease = 1;
    #30 ex_decrease = 0;

    // Check if all is high
    #4 {reset, ex_increase, ex_decrease} = 3'b111;
    #4 {reset, ex_increase, ex_decrease} = 3'b000;
    #2 $finish;
  end


endmodule
```

C.5 FSM_ex_control.sv

```
module FSM_ex_control(
  input logic clk,
  input logic reset,
  input logic init,

  input logic        ex_done,
  output logic ex_start,
  output logic ex_set,

  output logic nre1,
  output logic nre2,
  output logic adc,
  output logic expose,
  output logic erase
);

  // Enum to keep track of current state
  enum logic [1:0] {IDLE, EXPOSURE, READOUT, ILLEGAL} current_state;

  // keep track of state in the read out state.
  logic [3:0] read_state = 0;

  always @(posedge clk) begin

    // Initialize if we are in idle and init is 1
    if (init && current_state == IDLE)begin
      current_state = EXPOSURE;
    end

    // Reset forces state to ILLEGAL
    if (reset)
      current_state = ILLEGAL;

    // FSM
    case (current_state)
      IDLE: begin
        // The IDLE values
        {ex_start, ex_set} = 2'b01;
        {nre1, nre2, adc, expose, erase} = 5'b11001;
      end
      EXPOSURE: begin
        // Start exposure by initializing timer_counter
        {ex_start, ex_set} = 2'b10;
        {nre1, nre2, adc, expose, erase} = 5'b11010;
        // Ensure read_state is 0 for next state
        read_state = 0;
      end
      READOUT: begin
        if (reset) begin
          current_state = IDLE;
          {nre1, nre2, adc, expose, erase} = 5'b11001;
```

```
        end

        // Readout FSM
        // Just a hardcoded sequence
        {expose, erase} = 2'b00;
        case (read_state)
          0: {nre1, nre2, adc} = 3'b110;
          1: {nre1, nre2, adc} = 3'b010;
          2: {nre1, nre2, adc} = 3'b011;
          3: {nre1, nre2, adc} = 3'b010;
          4: {nre1, nre2, adc} = 3'b110;
          5: {nre1, nre2, adc} = 3'b100;
          6: {nre1, nre2, adc} = 3'b101;
          7: {nre1, nre2, adc} = 3'b100;
          8: begin
            {nre1, nre2, adc} = 3'b110;
            current_state = IDLE;
          end
          default: {nre1, nre2, adc} = 3'b110;
        endcase

        read_state++;
      end
      ILLEGAL: begin
        // Reset the whole FSM
        current_state = IDLE;
        {ex_start, ex_set} = 2'b01;
        {nre1, nre2, adc, expose, erase} = 5'b11001;
      end
    endcase
  end

  // At ex_done change to READOUT state
  always @(posedge ex_done)
    if (current_state == EXPOSURE) begin
      ex_start = 0;
      expose = 0;
      current_state = READOUT;
    end

  // Inital values
  initial begin
    current_state = IDLE;
    {ex_start, ex_set} = 2'b01;
    {nre1, nre2, adc, expose, erase} = 5'b11001;
  end

endmodule
```

C.6 FSM_ex_control_tb.sv

```
`include "FSM_ex_control.sv"

module FSM_ex_control_tb();

  logic clk, reset, init;
  logic        ex_done;
  wire ex_start, ex_set;

  wire nre1, nre2, adc, expose, erase;

  FSM_ex_control testbench(clk, reset, init, ex_done, ex_start, ex_set, nre1,
   ↪  nre2, adc, expose, erase);

  // Clock
  always begin
    #1 clk = !clk;
  end

  initial begin
    $dumpfile("outfiles/FSM_ex_control/FSM_ex_control_tb.vcd");
    $dumpvars(0, testbench);

    // Start with clock high
    {clk, reset, init, ex_done} = 4'b1000;

    // Pusle reset
    #2 reset = 1;
    #2 reset = 0;

    // Pulse init to initialize
    #2 init = 1;
    #2 init = 0;

    // Simulate exposure time, set it to exposure time * 2 - 2
    #8 ex_done = 1;
    #2 ex_done = 0;

    #20 // Time of readout
    #2 $finish;


  end


endmodule
```

C.7 timer_counter.sv

```
module timer_counter (
  input logic clk,
  input logic ex_set,
  input logic ex_start,
  input logic [4:0] ex_init,
  output wire ex_done
);

  logic [4:0] internal_count;

  always @(posedge clk)
    if (ex_set)
      internal_count = ex_init; //load internal_count with new values, add one because the way it counts
    else if (ex_start && (internal_count > 0))
      internal_count = internal_count - 1; // For each clock and ex_start, decrement one

    // Set the ex_done output to 1 if the counter is 0
    assign ex_done = (internal_count == 5'b0) ? 1 : 0;

endmodule
```

C.8 timer_counter_tb.sv

```
`include "timer_counter.sv"

module timer_counter_tb();

  logic clk, ex_set, ex_start;
  wire ex_done;
  logic [4:0] ex_init;

  timer_counter testbench(clk, ex_set, ex_start, ex_init, ex_done);

  // Clock
  always begin
    #1 clk = !clk;
  end

  initial begin
    $dumpfile("outfiles/timer_counter/timer_counter_tb.vcd");
    $dumpvars(0, testbench);

    // Start with clock high
    {clk, ex_start, ex_set} = 3'b100;
    // Set start value for counter, usually controlled by CTRL_ex_time
    // Here we count down from decimal 10
    ex_init = 5'd10;
    // Pulse ex_set to set value to count down from
    #2 {ex_start, ex_set} = 2'b01;
    #2 {ex_start, ex_set} = 2'b10;
  end

  // When the ex_done is set high, stop the simulation
  always @(posedge ex_done) begin
    #2 {ex_start, ex_set} = 2'b00;
    #4 {ex_start, ex_set} = 2'b01;
    #4 $finish;
  end
endmodule
```

D Other code used in the project

D.1 plots.m

```
%Plotting script
close all;
clc;

%%
%Loading datafile and allocating variables
doc = load('./simResults/test750_500_250_50@2ms.txt');

t = doc(:,1);
v_erease = doc(:,2);
v_expose = doc(:,3);
v_nrer1 = doc(:,4);
v_nrer2 = doc(:,5);
vout1 = doc(:,6);
vout2 = doc(:,7);
vout_sampled1 = doc(:,8);
vout_sampled2 = doc(:,9);
vout_sampled3 = doc(:,10);
vout_sampled4 = doc(:,11);

%%
%First plot pane
subplot(5, 1, 1);
plot(t*1e3, v_erease, 'r', 'linewidth', 1.5);
axis([0 12 -0.25 2]);
grid;
ylabel('V_{erease} [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);

%Second plot pane
subplot(5, 1, 2);
plot(t*1e3, v_expose, 'r', 'linewidth', 1.5);
axis([0 12 -0.25 2]);
grid;
ylabel('V_{expose} [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);


%Third plot pane
subplot(5, 1, 3);
plot(t*1e3, vout_sampled1, 'r', 'linewidth', 1.5);
hold on;
plot(t*1e3, vout_sampled2, 'b', 'linewidth', 1.5);
plot(t*1e3, vout_sampled3, 'g', 'linewidth', 1.5);
plot(t*1e3, vout_sampled4, 'black', 'linewidth', 1.5);
hold off;
legend('Sampled 1', 'Sampled 2', 'Sampled 3', 'Sampled 4');
axis([0 12 -0.25 1]);
grid;
ylabel('Voltage [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);

```

```
53
54 %Fourth plot pane
55 subplot(5, 1, 4);
56 plot(t*1e3, v_nrer1, ':r', 'linewidth', 2);
57 hold on;
58 plot(t*1e3, v_nrer2, ':b', 'linewidth', 2);
59 hold off;
60 legend('NRE1','NRE2');
61 axis([0 12 -0.25 2]);
62 grid;
63 ylabel('Voltage [V]', 'fontsize', 14);
64 xlabel('Time [ms]', 'fontsize', 14);
65
66
67 %Fifth plot pane
68 subplot(5, 1, 5);
69 plot(t*1e3, vout1, ':r', 'linewidth', 2);
70 hold on;
71 plot(t*1e3, vout2, ':b',  'linewidth', 2);
72 hold off;
73 axis([0 12 1 2]);
74 legend('Output 1', 'Output 2');
75 grid;
76 ylabel('Voltage [V]', 'fontsize', 14);
77 xlabel('Time [ms]', 'fontsize', 14);
78
79
80 %%
81 %new figure
82 figure;
83 %First plot pane
84 subplot(5, 1, 1);
85 plot(t*1e3, v_erease, 'r', 'linewidth', 1.5);
86 axis([0 12 -0.25 2]);
87 grid;
88 ylabel('V_{erease} [V]', 'fontsize', 14);
89 xlabel('Time [ms]', 'fontsize', 14);
90
91 %Second plot pane
92 subplot(5, 1, 2);
93 plot(t*1e3, v_expose, 'r', 'linewidth', 1.5);
94 axis([0 12 -0.25 2]);
95 grid;
96 ylabel('V_{expose} [V]', 'fontsize', 14);
97 xlabel('Time [ms]', 'fontsize', 14);
98
99
100 %Third plot pane
101 subplot(5, 1, 3);
102 plot(t*1e3, vout_sampled1, 'r', 'linewidth', 1.5);
103 hold on;
104 plot(t*1e3, vout_sampled2, 'b', 'linewidth', 1.5);
105 plot(t*1e3, vout_sampled3, 'g', 'linewidth', 1.5);
106 plot(t*1e3, vout_sampled4, 'black', 'linewidth', 1.5);
107 hold off;
108 legend('Sampled 1', 'Sampled 2', 'Sampled 3', 'Sampled 4');
109 axis([0 12 -0.25 1]);
```

```
grid;
ylabel('Voltage [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);


%Fourth plot pane
subplot(5, 1, 4);
plot(t*1e3, v_nrer1, ':r', 'linewidth', 2);
hold on;
plot(t*1e3, v_nrer2, ':b', 'linewidth', 2);
hold off;
legend('NRE1','NRE2');
axis([0 12 -0.25 2]);
grid;
ylabel('Voltage [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);


%Fifth plot pane
subplot(5, 1, 5);
plot(t*1e3, vout1, ':r', 'linewidth', 2);
hold on;
plot(t*1e3, vout2, ':b',  'linewidth', 2);
hold off;
axis([0 12 1 2]);
legend('Output 1', 'Output 2');
grid;
ylabel('Voltage [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);



%%
%New figure
%First plot pane
figure;
subplot(3, 1, 1);
plot(t*1e3, v_expose, 'r', 'linewidth', 1.5);
axis([0 12 -0.25 2]);
grid;
ylabel('V_{expose} [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);
xline(1, '--b');
xline(3.04, '--b','HandleVisibility','off');

%Second plot pane
subplot(3, 1, 2);
plot(t*1e3, v_erease, 'r', 'linewidth', 1.5);
axis([0 12 -0.25 2]);
grid;
ylabel('V_{erease} [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);

%Third plot pane
subplot(3,1,3);
plot(t*1e3, vout_sampled1, 'r', 'linewidth', 1.5);
hold on;
```

```
grid;
plot(t*1e3, vout_sampled2, 'b', 'linewidth', 1.5);
plot(t*1e3, vout_sampled3, 'g', 'linewidth', 1.5);
plot(t*1e3, vout_sampled4, 'black', 'linewidth', 1.5);
legend('V_{out sampled 1}','V_{out sampled 2}','V_{out sampled 3}','V_{out
↪  sampled 4}')
%title('V_{out sampled} for all 4 pixel cells', 'fontsize', 16);
ylabel('Voltage [V]', 'fontsize', 14);
xlabel('Time [ms]', 'fontsize', 14);
xline(1, '--b','HandleVisibility','off');
xline(3.04, '--b', 'HandleVisibility','off');

%%
%Process variations
doc_TT = load('SwitchVdd1.8VVgSweep1.8V_TT.txt');
doc_SS = load('SwitchVdd1.8VVgSweep1.8V_SS.txt');
doc_FF = load('SwitchVdd1.8VVgSweep1.8V_FF.txt');

Vgg_TT = doc_TT(:,2);
gm_TT = doc_TT(:, 3);

Vgg_SS = doc_SS(:, 2);
gm_SS = doc_SS(:, 3);

Vgg_FF = doc_FF(:,2);
gm_FF = doc_FF(:,3);

figure;
plot(Vgg_TT, gm_TT*1e3, 'linewidth', 1.5);
hold on;
plot(Vgg_SS, gm_SS*1e3, 'linewidth', 1.5);
plot(Vgg_FF, gm_FF*1e3, 'linewidth', 1.5);
hold off;
grid;
legend('TT', 'SS', 'FF');
title('Process variations of switch');
ylabel('Transconductance [mS]');
xlabel('Gate voltage [V]');


%%
%Command for exporting vector graphics:
%exportgraphics(gcf,'title.pdf','ContentType','vector'
```