NTNU – Trondheim
Norwegian University of
Science and Technology

TFE4152 Design av integrerte kretser
Project Report

Small camera model

Group 03:

Anders Nilsen
Torgeir Leithe

November 23, 2017

Contents

1. Introduction

Electronics dominate the modern day appliance market, replacing appliances using mechanical operations with electronics to achieve the same goal. One such example is the camera. Previously, until the introduction of the digital camera in 1975, camera's worked by exposing a chemical film to light for a user-defined period of time. In digital cameras, a photo-diode is used to generate current. When the camera is instructed by the user to capture a photo, the diode is used to charge a capacitor. The voltage of the capacitor is amplified and read by an ADC. The internal mechanics of the camera, such as erasing, controlling the exposure time and the readout of data is controlled by a control circuit. Today, digital cameras dominate the market, with mechanical cameras having become a niche. Camera technology progresses each year with more advanced circuits and capture capabilities, as well as allowing for a more user-friendly experience than mechanical cameras.

As a part of the course "Design of integrated circuits", TFE4152, students are tasked with designing the analog readout circuit and digital capture control for a four-pixel digital camera.

2. Overview

The project is divided into two parts: The analogue part where the readout circuit is designed and the MOSFETs are sized to adjust the gain and offset voltage, and the digital part where the control circuitry is designed using Verilog. The circuit diagram for the analogue part was provided along with some specifications, such as capacitor sizes.

3. Analogue design

3.1. Overview

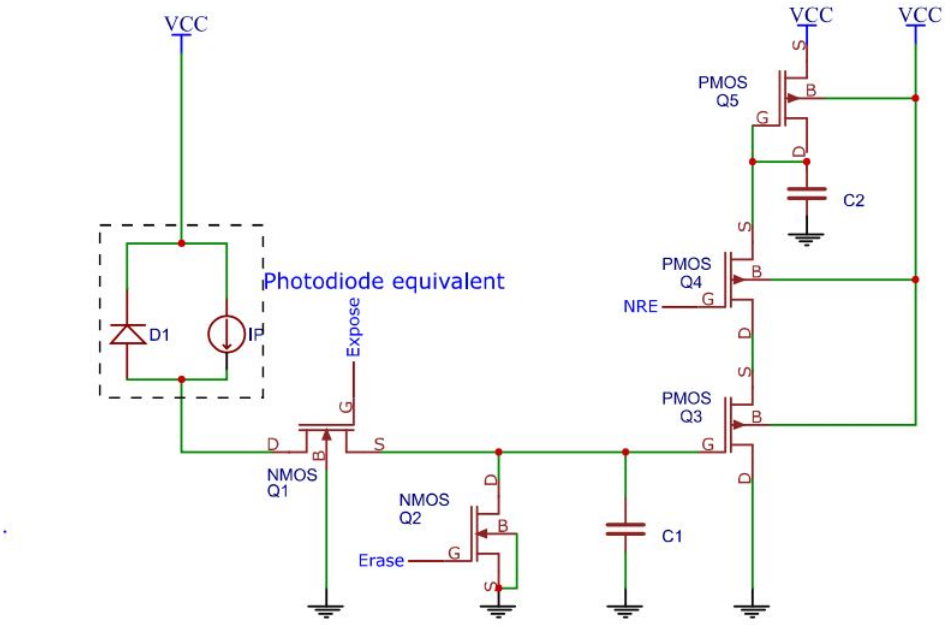


Figure 3.1.: Circuit diagram for the sensor readout

The analog circuitry uses transistors as switches in addition to a transistor for amplification and one to act as load on the output. Figure 3.1 shows the circuit. The photo-diode generates a current, between 200pA in bad lighting conditions to 3nA in good conditions, which is used to charge the capacitor C1 at the gate of PMOS Q3 when the Expose signal is high, turning on NMOS Q1, for a duration of 2ms to 30ms. Transistor Q2 is used to discharge the capacitor. Transistor Q3 is used to amplify the signal, and Q4 is used to control the voltage at the output which is between transistors Q4 and Q5. The output voltage is sent to the input of an ADC-circuit. It is assumed that it is an ideal amplifier with infinite input resistance and is therefore not included in this diagram. Q5 acts as an active load resistor as the resistance Rds can be approximated to be the inverse of the transconductance gm5. Capacitor C2 is an approximation of the parasitic capacitance of the wires connecting the components.

The transistors used are assumed to be 0.35µm-technology, as the supplied transistor model for use in AIM-SPICE simulations is the same as previously used on exercises in this subject, where the transistors were specified to be 0.35µm.

3.2. Without body effect

3.2.1. MOSFET pin sizing and offset calculation

As the circuit diagram is already provided, the main part of the analogue design is sizing the MOSFET pins. The width and length of the transistor pins are used to control the gain and resistance of transistors.

$$u_c(t) = \frac{1}{C} \int_0^t i(t)\ \mathrm{dt} \tag{3.1}$$

Transistors Q1, Q2 and Q4 can be simplified to switches as they are only used for switching the signal. The voltage across C1 can be calculated using equation 3.1, and as the main focus is sizing the pins of Q3 and Q5, all the components to the left of transistor Q3 can be replaced by a voltage source. This results in the circuit seen in figure 3.2.

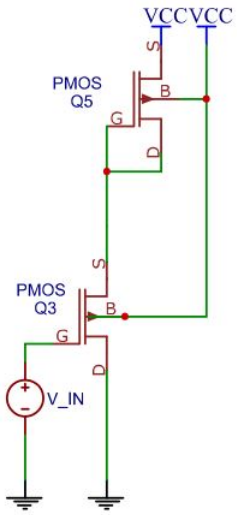


Figure 3.2.: Simplified circuit diagram for the sensor readout

The resulting circuit makes it plain to see that the drain-source current is the same for both transistors. The equation for calculating the drain-source current of a MOSFET

in the active region is shown in equation 3.2. The channel-length modulation parameter is dropped to simplify the calculations.

$$i_d = \frac{1}{2}\mu_p C_{ox}\frac{W}{L}{V_{eff}}^2[1 + \lambda(V_{DS} - V_{eff})] \tag{3.2}$$

This results in the following relationship between the two MOSFETs, which can be used to calculate the size of the pins, the gain and the offset of the circuit. The relationship is shown in equation 3.3, where the charge-carrier effective mobility, μ_p, and the gate oxide capacitance per unit area, C_{ox}, can be omitted as they are the same for both transistors. This results in equation 3.4.

$$\frac{1}{2}\mu_p C_{ox}\frac{W_3}{L_3}{V_{eff3}}^2 = \frac{1}{2}\mu_p C_{ox}\frac{W_5}{L_5}{V_{eff5}}^2 \tag{3.3}$$

$$\frac{W_3}{L_3}(V_{off} - |V_{tp}|)^2 = \frac{W_5}{L_5}(V_{DD} - V_{off} - |V_{tp}|)^2 \tag{3.4}$$

The size of the pins on Q5 can be used to scale the offset voltage, while the gain of the circuit can be scaled through the size of Q3's pins. Equation 3.4 can be rearranged to find the ratio between the transistor pins of Q3 and Q5 by choosing a desired offset voltage V_{off}. This results in equation 3.5, and by choosing an offset of 1.5V, and using $V_{DD} = 5\text{V}$ and $V_{tp} = -0.71\text{V}$ results in a ratio of 12.5.

$$\frac{W_3 \cdot L_5}{W_5 \cdot L_3} = (\frac{V_{DD} - V_{off} - |V_{tp}|}{V_{off} - |V_{tp}|})^2 \tag{3.5}$$

By choosing $W_3 = 10$ and $L_3 = 1$ results in a ratio of 1.25 between L_5 and W_5. Choosing $L_5 = 5$ and $W_5 = 4$ satisfies this criteria.

3.2.2. Gain calculation

The gain of the circuit can be calculated by using a combination of circuit diagram and the small signal model equivalent of the circuit. The small signal model is shown in figure 3.3. The voltage across V_{out} can be found by calculating the current passing through r_{ds}. This current can be found by calculating the transconductance g_{m3} of transistor Q3. To find this, the current i_d has to be calculated. This is the current running through both transistors which was used to size the transistor pins. This current is calculated using equation 3.2, and using the widths, lengths and voltages of Q3 results in a drain-current of $i_d = 171.62\mu\text{A}$.

Now, the transconductance of Q3 and Q5 can be calculated, which can be done using equation 3.6, resulting in $g_{m3} = 434\mu\text{A V}^{-1}$ and $g_{m5} = 122.9\mu\text{A V}^{-1}$.

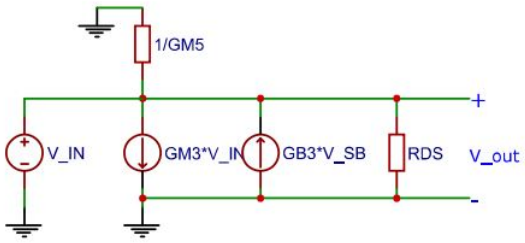


Figure 3.3.: Small signal equivalent of Q3 and Q5

$$g_m = \sqrt{2\mu_p C_{ox} \frac{W}{L} i_d} \tag{3.6}$$

The drain-source resistance of Q3 is needed to calculate the voltage of the output. This can be calculated using equation 3.7, resulting in a resistance of $r_{ds} = 2.6\text{k}\Omega$.

$$r_{ds}^{-1} = g_{ds} = \frac{\partial i_d}{\partial V_{ds}} = \mu_p C_{ox} \frac{W}{L} (V_{eff} - |V_{tp}|) \tag{3.7}$$

Using these values, the output voltage, and subsequently gain, can be calculated as described earlier in this chapter by calculating the current passing through r_{ds}. As the body effect is neglected for the time being, the current source $g_{b3}V_{SB}$ can be set to zero. The current through the resistor is then given by the current from $g_{m3}V_{in}$ divided between $1/g_{m5}$ and r_{ds}, resulting in equation 3.8. Using $V_{in} = 2\text{V}$ gives an output voltage of 1.43V, which gives the circuit a gain of $A = 0.715$.

$$V_{out} = g_{m3} V_{in} \frac{1/g_{m5}}{r_{ds} + 1/g_{m5}} r_{ds} \tag{3.8}$$

3.2.3. With body effect

When the body effect is taken into account, the calculations are changed. New threshold voltages have to be calculated, resulting in altered drain-current and gain.

The body effect is calculated based on doping concentrations and process parameters. The doping concentration for the PMOS was not supplied, but was found in the AIM-SPICE model file for the PMOS. Additional values were needed as well and is included in table 3.1.

Symbol:	Value:
q	$1.602 \cdot 10^{-19}\mathrm{C}$
k	$1.38 \cdot 10^{-23}\mathrm{J\,K^{-1}}$
n_i	$1.1 \cdot 10^{16} carriers \cdot m^{-3}$
N_A	$3.53 \cdot 10^{11} carriers \cdot m^{-3}$
T	300K
ϵ_0	$8.854 \cdot 10^{-12}\mathrm{F\,m^{-1}}$
K_{ox}	3.9
K_{si}	11.8
t_{ox}	$8 \cdot 10^{-9}\mathrm{m}$

Table 3.1.: Variables used for calculating the body effect.

$$V_{th0} + \gamma(\sqrt{V_{SB} + 2\phi_F} - 2\phi_F) \tag{3.9}$$

The threshold voltage can be calculated using equation 3.9, though the body-effect constant γ must first be calculated. This can be done using equation 3.10. This requires the fermi-potential of the body, ϕ_F, which is not supplied and can be calculated using equation 3.11, and the gate capacitance per unit area, C_{ox}, is given by equation 3.12.

$$\gamma = \frac{\sqrt{2qK_{si}\epsilon_0 N_A}}{C_{ox}} \tag{3.10}$$

$$\phi_F = \frac{kT}{q} ln(\frac{N_A}{n_i}) \tag{3.11}$$

$$C_{ox} = \frac{K_{ox}\epsilon_0}{t_{ox}} \tag{3.12}$$

Using the variables supplied and found in the AIM-SPICE-model results in $C_{ox} = 4.31 \cdot 10^{-3}\mathrm{F\,m^{-2}}$, $\phi_F = -0.267\mathrm{J\,C^{-1}} = -0.267\mathrm{V}$. Using this, the body-effect constant can be calculated to be $\gamma = 8 \cdot 10^{-7}$. The body effect constant is so small that it is negligible. As a result, the threshold voltages and gain is the same as without the body effect.

The main difference in regards to the gain of the circuit when considering the body effect is the reduction in the current going through r_{ds} on the small-signal model, as seen in figure 3.3, as a result of the current source $g_b V_{SB}$ being taken into account.

4. Digital design

4.1. State machine

To design the desired functionality of the digital circuit, a state machine was made to divide the functionality into different stages, simplifying the coding process by having a structure to base it on.

One state machine was used, with one internal state machine for the conversion-state, as seen in figure 4.1 and 4.2, respectively. As figure 4.1 shows,

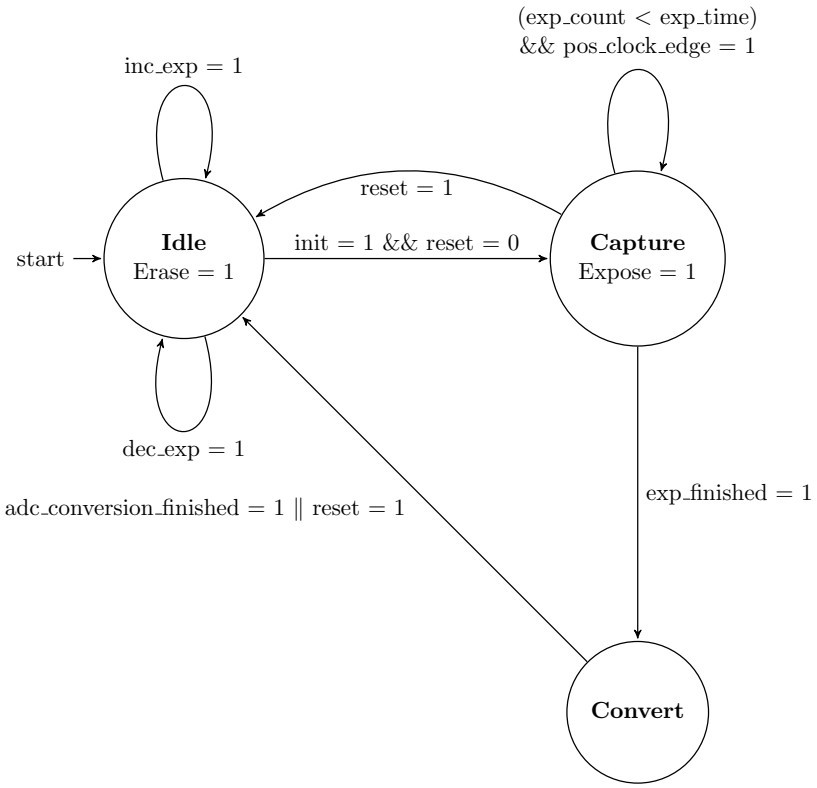


Figure 4.1.: State machine diagram for the digital control

The state machine uses counters to control exposure time and during the conversion

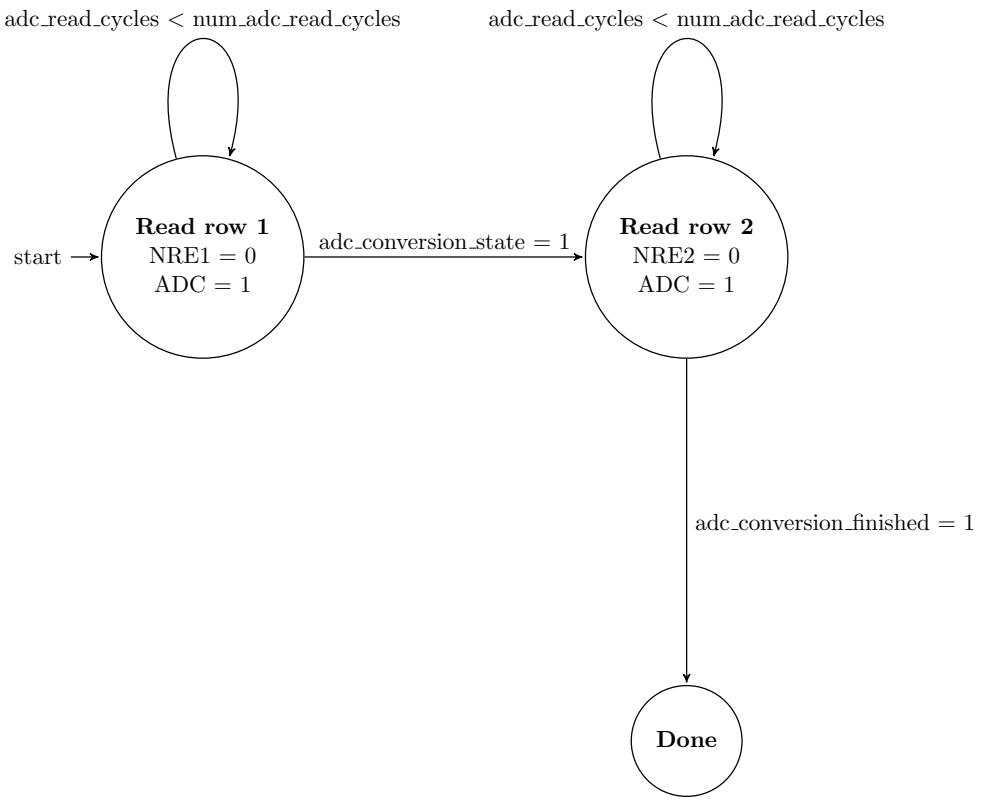


Figure 4.2.: Diagram for the internal conversion FSM

state, which is not easy to represent in a diagram.

4.2. Coding

4.2.1. Functional code

The Verilog-code was written using ActiveHDL. The code was written in one module for simplicity's sake, as no function required a separate module. Initially, the module is declared with inputs and outputs in the sensitivity list, and afterwards they are defined as inputs and outputs along with the corresponding wires and registers for the inputs and outputs, respectively. The complete code, with comments, is included as appendix B

```
1 module camera_control(init, reset, clk, exp_inc, exp_dec, nre1, nre2,
      adc, expose, erase);
2
3   input init, reset, clk, exp_inc, exp_dec;
```

```
4
5     output nre1, nre2, adc, expose, erase;
6
7     wire init, reset, clk, exp_inc, exp_dec;
8
9     reg nre1, nre2, adc, expose, erase;
10    reg exp_finished, adc_conversion_finished, adc_conversion_state;
```

Additionally the internal signals, parameters for the array sizes, and the registers themselves, are defined. As counters are being used to control the exposure time and the conversion, several internal registers are needed. Parameters for the array sizes were defined to avoid the use of "magic numbers".

```
parameter state_array_size = 2, exp_array_size = 5;
parameter num_adc_read_cycles = 3, adc_array_size = 2;
parameter idle = 2'b00, capture = 2'b01, convert = 2'b11;

reg [state_array_size-1:0] state;
reg [exp_array_size-1:0] exp_time;
reg [exp_array_size-1:0] exp_count;
reg [adc_array_size-1:0] adc_read_cycle;
```

The controller uses only one always-block which triggers on positive clock edges. When the reset-button is pressed, the state goes back to Idle, all outputs are reset and the exposure time is reset to 15ms.

```
always @(posedge clk)
    begin: FSM
      if (reset == 1'b1) begin
          $display ("Reset enabled");
          state <= idle;
          nre1 <= 1;
          nre2 <= 1;
          adc <= 0;
          expose <= 0;
          erase <= 1;
          exp_time <= 15;
      end else
          case(state)
```

After the code-snippet above, the different states of the camera controller start. The use of the case-function allows for switch-case behaviour, increasing the code readability and structure. The $display-function prints the contents of the parenthesis to the console, allowing for easier debugging and giving the user more information in regards to which state the controller is in.

```
case(state)
idle:
  if (init == 1'b1 && reset == 1'b0) begin
```

```
4         $display ("Initialising");
5         exp_finished <= 0;
6         exp_count <= 0;
7         adc_conversion_finished <= 0;
8         adc_conversion_state <= 0;
9         adc_read_cycle <= 0;
10        state <= capture;
11    end else if (exp_inc == 1'b1 && exp_dec == 1'b0) begin
12        if (exp_time < 30) begin
13            $display ("Exposure time increased by 1");
14            exp_time <= exp_time + 1;
15          end
16    end else if (exp_dec == 1'b1 && exp_inc == 1'b0) begin
17        if (exp_time > 2) begin
18            $display ("Exposure time decreased by 1");
19            exp_time <= exp_time - 1;
20          end
21    end else begin
22        $display ("State: Idle");
23        erase <= 1;
24      end
```

The idle-state erases the charge from the capacitor. In addition, the exposure time can be increased and decreased. Once the capture-button is pressed and the init-signal is generated, the controller moves into the Expose-stage after reseting all the internal counters and states.

```
1  capture:
2    if (exp_finished == 0) begin
3        $display ("Expose = 1, Erase = 0");
4        expose <= 1;
5        erase <= 0;
6        if (exp_count < exp_time) begin
7            exp_count <= exp_count + 1;
8            $display ("Exposing");
9        end else if (exp_count >= exp_time) begin
10           expose <= 0;
11           exp_finished <= 1;
12           $display("Exposure finished");
13           state <= convert;
14         end
15     end
```

Here, a loop increments until it has reached the desired exposure-value. As the clock-frequency is 1kHz, no scaling is needed for the counter values.

```
1  convert:
2    if (adc_conversion_finished == 0) begin
3        $display("Converting");
```

```
4         if (adc_conversion_state == 0) begin
5             $display("Converting row 1");
6             if (adc_read_cycle < num_adc_read_cycles) begin
7                 if (adc_read_cycle == 1) begin
8                     adc <= 1;
9                 end else begin
10                    adc <= 0;
11                end
12                nre1 <= 0;
13                adc_read_cycle <= adc_read_cycle + 1;
14            end else if (adc_read_cycle >= num_adc_read_cycles) begin
15                nre1 <= 1;
16                adc_conversion_state <= 1;
17                adc_read_cycle <= 0;
18            end
```

Once the exposure time has been reached, the controller moves into the convert-state. Here, the two "rows" of pixels are converted at a time. The NRE-signals are high for three clock cycles while the DAC-signal is high for one clock cycle. It is worth noting that these values are assumed, not specified in the project. This is repeated for the second row after the first row is finished, signalled by adc_conversion_state which is set high. After the second row is finished converting, the controller goes back to the Idle-state.

4.2.2. Testbench

To test the functionality of the camera controller and verify that it works correctly, a testbench was designed in ActiveHDL. The testbench is simple, as it only serves to stimulate the inputs of the testbench. In addition, it monitors and prints the values of the outputs and internal registers. The complete code for the testbench is included as appendix C.

```
1 `timescale 1 ms / 100 us
2
3 always
4   #0.5 clk_in = ~clk_in;
```

One important part of the testbench and functional code was to specify a timescale of 1ms. This allowed for making a simple clock by inverting the clock value every half clock cycle.

```
1 camera_control uut(
2     .init(init_in),
3     .reset(reset_in));
```

In addition to the clock, the inputs of the test bench and wires for the outputs had to be mapped to the inputs and outputs of the camera controller. Not all inputs and outputs

are included in the above code snippet.

```
initial
  begin
    clk_in = 1'b0;          //Reset clock
    reset_in = 1'b0;
    #1; reset_in = 1'b1;   //Reset is active-high
    #1; reset_in = 1'b0;
    #1; exp_dec_in = 1'b0;
    #1; exp_inc_in = 1'b1;   //Increase exposure time 1ms for each
        clock cycle
    #20; reset_in = 1'b1;    //Wait 20ms before continuing, giving an
        exposure time of 30ms.
    #1; reset_in = 1'b0;
    #1; init_in = 1'b1;      //Initialise the camera operation
    #1; init_in = 1'b0;
    #40;
    $finish;         //$finish to stop the simulation after the 40ms
        delay
  end
```

This is the code which the testbench uses to stimulate the inputs of the camera controller. This code was changed several times during the design process to test various scenarios. This code in the snippet above isu sed to test whether the exposure time is reset when the reset signal is high, one clock cycle before the circuit is initiated.

```
initial
  begin
    $monitor("time = %2d, clk_in =%b, init_in=%b, reset_in=%b,
        exp_inc_in=%b, exp_dec_in=%b, nre1_out=%b, nre2_out=%b,
        expose_out=%b, erase_out=%b, adc_out= %b, exp_time= %d,
        exp_count = %d, adc_read_cycle = %d", $time,clk_in,init_in,
        reset_in,exp_inc_in,exp_dec_in,nre1_out,nre2_out,expose_out,
        erase_out,adc_out,uut.exp_time, uut.exp_count, uut.
        adc_read_cycle);
  end
```

The code in the snippet above prints the values of the internal registers and variables to the console.

5. Results

5.1. AIM-SPICE simulation

. The analogue part of the project was simulated using AIM-SPICE, an analogue electronics simulator. To simulate a circuit the user has to write a netlist, a list of all components and connections. AIM-SPICE allows for highly detailed and accurate simulation. The netlist for the analogue part is included as appendix A.

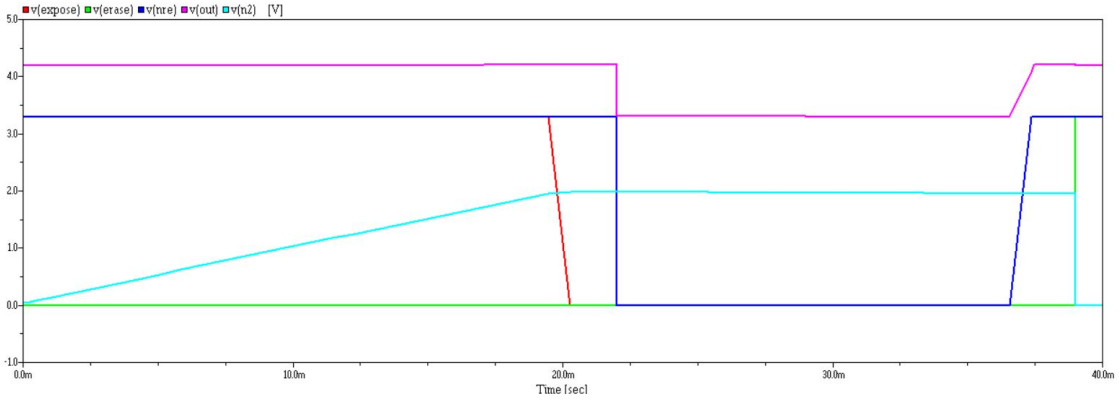


Figure 5.1.: Camera operation with a 20s exposure, with switch-signals and output voltage.

Using the transient analysis function in AIM-SPICE allows for the simulation of the amplified output-voltage, along with the voltages for the MOSFETs used as switches (NRE, Expose and Erase). Figure 5.1 shows this after a 20ms exposure time with a current of 200pA, followed by a delay of 2ms, after which a period of 15ms of reading the signal for the DAC, followed by a 2ms delay, and finally a discharge of the capacitor voltage.

Figure 5.2 shows a closer look at the output voltage and the voltage at the gate of the amplifier transistor Q3. Using a 2V signal at the gate results in an output voltage of 3V (prior to NRE activating), with an offset voltage of 1.34V. Subtracting the threshold voltage results in a gain of 0.83. This doesn't match the calculations from earlier, but is close (an error of 17%). Figure 5.3 shows the output current, and it's obvious that the current calculated earlier is incorrect, as the figure shows a current of about 13µA

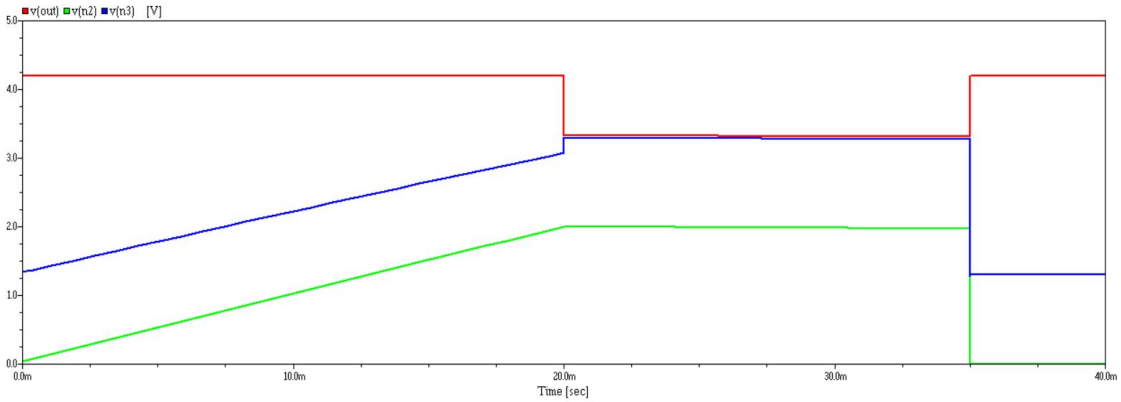


Figure 5.2.: Closer look at the output voltage (red) with the voltage at the drain of Q3.

compared to 171µA. The cause of this might be miscalculation of the body effect, or the fact that the channel-length modulation parameter was dropped from the threshold-voltage equation, resulting in an incorrect drain-source resistance.

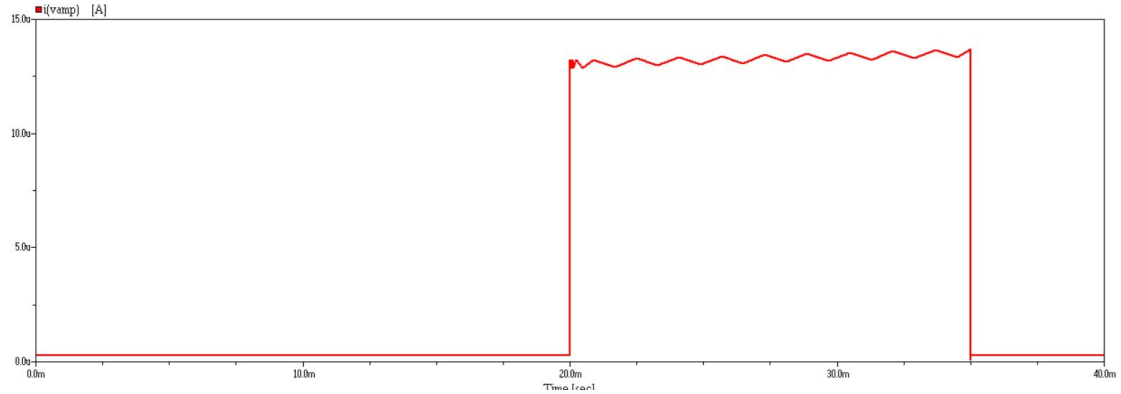


Figure 5.3.: Current through the transistors Q3, Q4 and Q5 at the output.

The body effect was most likely wrong, as the body effect constant was surprisingly small. The cause of this might be an incorrect doping concentration, or the fact that some parameters were mixed, such as C_{ox} which was calculated based on parameters from the book instead of the one used in the transistor model. If the body effect was calculated correctly and proved to be contributing to the output voltage, one solution would have been to increase the offset voltage, though this would result in a reduced voltage swing, allowing for less resolution on the exposed voltage value.

5.2. ActiveHDL simulation

The digital camera control was simulated using Active-HDL, using the testbench for input stimulation. Appendix D shows the waveform diagram for a simulation of what can be considered normal behaviour, where the exposure time is increased for ten clock cycles until it is 25ms, followed by a reset-free capture and conversion. As the figure shows, the control circuit works as expected.

To ensure that the control circuit works properly in more specific cases, several scenarios can be programmed into the testbench. Appendix E shows a reset during the exposure stage, changing the state back to Idle, as well as how the exposure time is reduced to 5ms for the first exposure, while it is increased to 20ms for the second exposure. It also shows that the circuit prioritises the Idle-signal, as the circuit moves into the Capture-state even though the exposure time is increasing. This is a prioritisation based on the priorities of a photographer: It is more important to capture a photograph than ensuring that it has the correct exposure setting.

One improvement to the digital design would be to use gray-counters for the various counters in the system. Gray-counting reduces the power consumption as only one bit is switched each count. This was used for the different states, but as the exposure counter can be up to 30 clock cycles, the effect is minuscule.

Synthesising the design using Vivado, on a Kintex7 xc7k70tfbv6761 FGPA, is possible, and estimates a power consumption of 0.44W, a temperature of 25.8°C, and using 33 look-up tables and 22 registers as flip-flops.

6. Conclusion

The analogue calculations did not match the simulation results, and as discussed in the results, might be a result of miscalculations and over-simplification of the equations used in addition to wrong parameters. The digital camera control, on the other hand, works correctly and as expected, satisfying the criteria of the project. The power consumption and the area used was, according to Vivado, low.

A. AIM-SPICE netlist

A I M - S p i c e
C:\Users\Anders\Documents\AIM-spice\DIK-project\AIM-Spice\dik-prosjekt_pmos.cir

Circuit Description

```
Transistor test
.include C:\Users\Anders\Downloads\PhotoDiodeSubCircuit.cir
.include C:\Users\Anders\Documents\AIM-spice\nmos_model.cir
.include C:\Users\Anders\Downloads\pmos_model.cir
.model nm nmos
.model pm pmos
.model dwell d

.param Ipd_1=200p-3n!Define the current range for the photo-diode

vdd vdd 0 dc 5!Define VDD
vex expose 0 dc 0 pulse(0 3.3 0ms 1ns 1ns 20ms)!Gate-signal for the Expose-switch
ver erase 0 dc 3.3 pulse(0 3.3 35ms 1ns 1ns 5ms) !Gate-signal for the Erase-switch
vnr nre 0 dc 3.3 pulse(3.3 0 20ms 1ns 1ns 15ms)!Gate-signal for the NRE-switch
vamp out outa dc 0


dphoto n1 vdd dwell!Define the photo-diode from VDD to wire N1
id vdd n1 dc pulse (0 200p 0ms 1ns 1ns 20ms)!Define the current source of the photo-diode
mexpose n1 expose n2 0 nm w=5u l=1u!Define the expose-NMOS m1 from wire N1 to N2
merase n2 erase 0 0 nm w=10u l=1u!Define the erase-NMOS m2 from wire N2 to VSS (ground)
cs n2 0 2p!Define the charge-capacitor Cs from wire N2 to VSS (ground)

mload out out vdd vdd pm w=4u l=5u!Define the load-PMOS from VDD to out-wire.
cc out 0 3p!Define the parasitic capacitance of the load-wire
mnre n3 nre outa vdd pm w=5u l=1u!Define the NRE-PMOS M4 from the load-wire to N3
m3 0 n2 n3 vdd pm w=10u l=1u !Define the amplifier-PMOS M3 from N3 to VSS (ground)
```

Transient Analysis Parameters

Stepsize ..... : 0.1
Final time ..... : 40ms
Use Initial Conditions: Off

B. Verilog FSM code

```
1  `timescale 1 ms / 100 us
2
3  module camera_control(init, reset, clk, exp_inc, exp_dec, nre1, nre2,
       adc, expose, erase);
4
5    input init, reset, clk, exp_inc, exp_dec;
6
7    output nre1, nre2, adc, expose, erase;
8
9    wire init, reset, clk, exp_inc, exp_dec;
10
11
12
13   reg nre1, nre2, adc, expose, erase;
14   reg exp_finished, adc_conversion_finished, adc_conversion_state;
       //Exp_finished = TRUE when the camera is finished exposing,
       adc_conversion_finished = TRUE when the ADC-conversion is done
15   //adc_conversion_state = TRUE when the first inputs have been read
       and translated
16
17   parameter state_array_size = 2, exp_array_size = 5;      //The
       number of states is 3, so 2 bit is designated for the state-
       coutner. The exposure-time is max. 30ms, so a 5-bit register is
       needed.
18   parameter num_adc_read_cycles = 3, adc_array_size = 2;     //The
       number of clock cycles the ADC has to read the input. Assumed to
        be 3ms. The array size for the adc-counter is 2-bit.
19   parameter idle = 2'b00, capture = 2'b01, convert = 2'b11;    //
       Defining the different states of the state-machine.
20
21   reg [state_array_size-1:0] state;                  //Array for the
       different states
22   reg [exp_array_size-1:0] exp_time;                 //Array for the
       exposure time
23   reg [exp_array_size-1:0] exp_count;                  //Array for the
       exposure time counter
24   reg [adc_array_size-1:0] adc_read_cycle;             //Array for the
       number of read-cycle counter for the ADC
25
26   always @(posedge clk)
27     begin: FSM
```

```
28          if (reset == 1'b1) begin                    //Reset is active-
                high. Resets all output variables.
29              $display ("Reset enabled");
30              state <= idle;
31              nre1 <= 1;
32              nre2 <= 1;
33              adc <= 0;
34              expose <= 0;
35              erase <= 1;
36              exp_time <= 15;
37          end else
38            case(state)
39              idle:
40                if (init == 1'b1 && reset == 1'b0) begin
41                    $display ("Initialising");              //Change
                        state to Capture when idle is high.
42                    exp_finished <= 0;                    //Reset all
                        internal counters when initiating.
43                    exp_count <= 0;
44                    adc_conversion_finished <= 0;
45                    adc_conversion_state <= 0;
46                    adc_read_cycle <= 0;
47                    state <= capture;
48                end else if (exp_inc == 1'b1 && exp_dec == 1'b0 && init
                      == 1'b0) begin    //Increase the exposure time with 1
                      ms every clock cycle the button is pressed
49                    if (exp_time < 30) begin                          //as
                        long as the exposure time is smaller than 30.
50                      $display ("Exposure time increased by 1");
51                      exp_time <= exp_time + 1;
52                    end
53                end else if (exp_dec == 1'b1 && exp_inc == 1'b0 && init
                      == 1'b0) begin      //Decrease the exposure time with
                      1ms every clock cycle the button is pressed
54                    if (exp_time > 2) begin                           //as
                        long as the exposure time is bigger than 2.
55                      $display ("Exposure time decreased by 1");
56                      exp_time <= exp_time - 1;
57                    end
58                end else begin
59                    $display ("State: Idle");             //Enable erase
                        when idle.
60                    erase <= 1;
61
62                end
63              capture:
64                if (exp_finished == 0) begin
65                    $display ("Expose = 1, Erase = 0");
```

```
66                  expose <= 1;                    //The camera is in
                        expose-mode until the counter exp_count has
                        reached the specified exposure time.
67                  erase <= 0;
68                  if (exp_count < exp_time) begin         //The count
                        can be done every clock cycle as long as the clock
                         frequency is equal to the change in time.
69                      exp_count <= exp_count + 1;
70                      $display ("Exposing");
71                  end else if (exp_count >= exp_time) begin
72                      expose <= 0;
73                      exp_finished <= 1;
74                      $display("Exposure finished");
75                      state <= convert;           //After the exposure
                            is done the state-machien goes into the
                            conversion state.
76                  end
77              end
78          convert:
79            if (adc_conversion_finished == 0) begin
                //As long as the conversion is not done, the program
                converts from the two rows of
80              $display("Converting");                         //
                    amplifiers. Which row is read is controlled by
                    adc_conversion_state which is 0 for row
81              if (adc_conversion_state == 0) begin                //1
                     and 1 for row 2.
82                  $display("Converting row 1");
83                  if (adc_read_cycle < num_adc_read_cycles) begin
                                //The ADC has 5 clock cycles to read
                            the output of the amplifiers. This is not
                            required
84                      if (adc_read_cycle == 1) begin
                            //by the assignment. When the counter has
                            not reached the limit, the ADC and NRE1/2
                            signal
85                          adc <= 1;                       //is active
                                . (The NRE gets 0 as it is a PMOS).
86                      end else begin
87                          adc <= 0;
88                      end
89                      nre1 <= 0;
90                      adc_read_cycle <= adc_read_cycle + 1;
91                  end else if (adc_read_cycle >=
                        num_adc_read_cycles) begin //After the ADC has
                         read for 5 cycles,  the ADC is turned off to
                        give it a rest between reading
92                      nre1 <= 1;                          //samples.
                            The conversion state is changed and the
```

```
                        cycle counter is reset.
93                      adc_conversion_state <= 1;
94                      adc_read_cycle <= 0;
95                    end
96                  end
97                  if (adc_conversion_state == 1) begin
                      //The same as row 1, but after it is finished
                      reading the ADC-conversion is complete
98                    $display("Converting row 2");                //and
                        the state is changed to idle.
99                    if (adc_read_cycle < num_adc_read_cycles) begin
100                     if (adc_read_cycle == 1) begin
101                       adc <= 1;
102                     end else begin
103                       adc <= 0;
104                   end
105                     nre2 <= 0;
106                     adc_read_cycle <= adc_read_cycle + 1;
107                   end else if (adc_read_cycle >=
                        num_adc_read_cycles) begin
108                     nre2 <= 1;
109                     adc_conversion_finished <= 1;
110                   end
111                 end
112               end
113             else if (adc_conversion_finished == 1) begin
114                 $display("Conversion finished");                 //
                      After the conversion is finished the state goes to
                       idle again, resetting the counters and erasing
115                 state <= idle;                                  //the
                      charge from the capacitors.
116                 erase <= 1;
117               end
118         endcase
119     end
120
121
122 endmodule
```

C. Verilog testbench code

File: c:/Users/Anders/Documents/AIM-spice/DIK-project/Verilog/DIC-project/src/camera_tb.v (/

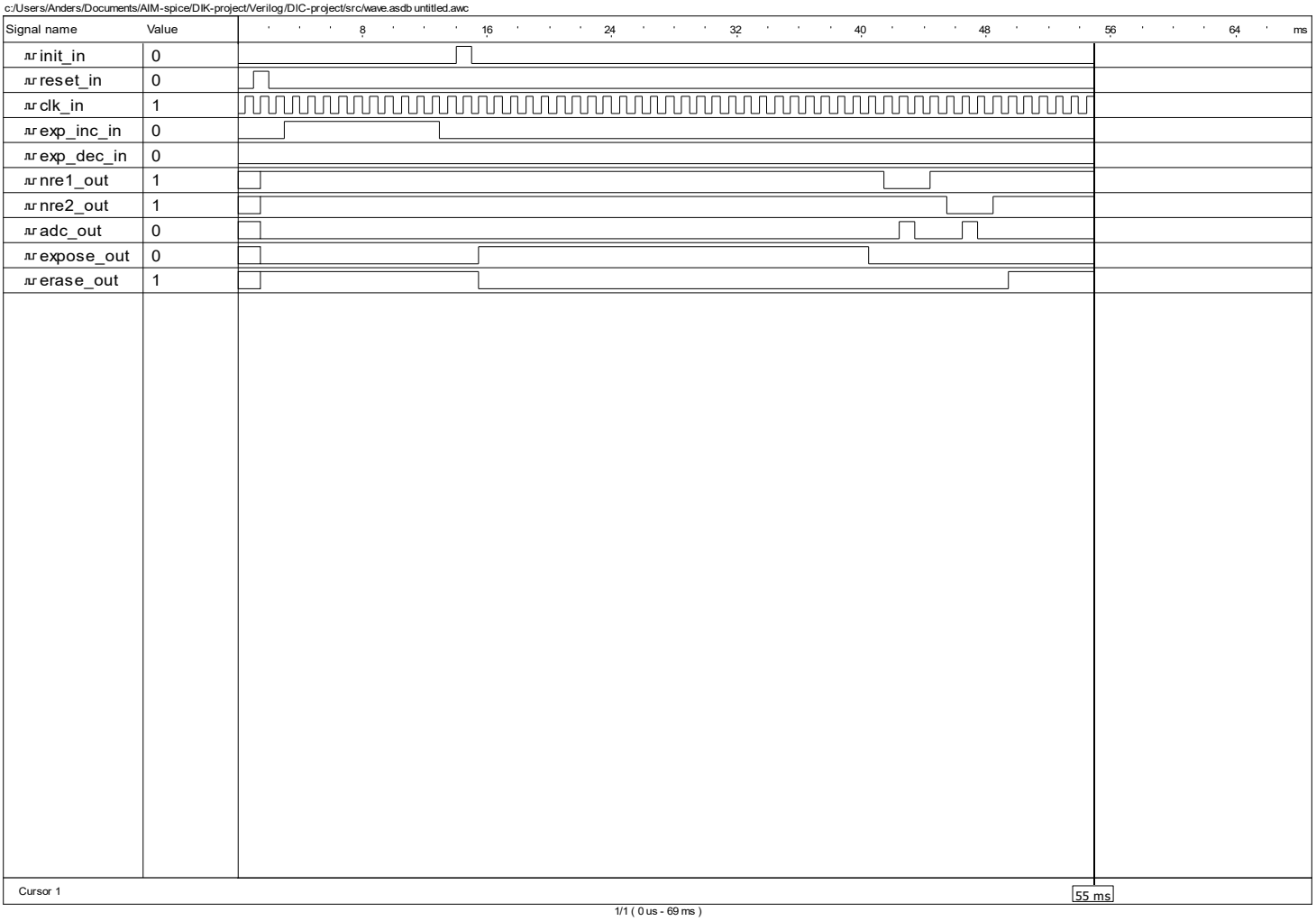
```
1   //------------------------------------------------------------------------
    ----
2   //
3   // Title        : camera_tb
4   // Design       : DIC-project
5   // Author       : Anders
6   // Company      : NTNU
7   //
8   //------------------------------------------------------------------------
    ----
9   //
10  // File         : c:\Users\Anders\Documents\AIM-spice\DIK-project\Verilog\DIC-project\src
    era_tb.v
11  // Generated    : Wed Nov  8 12:06:36 2017
12  // From         : interface description file
13  // By           : Itf2Vhdl ver. 1.22
14  //
15  //------------------------------------------------------------------------
    ----
16  //
17  // Description :
18  //
19  //------------------------------------------------------------------------
    ----
20  `timescale 1 ms / 100 us       //Timescale of 1ms as it is the clock
    frequency
21
22
23  module camera_tb;
24
25      reg init_in, reset_in, clk_in, exp_inc_in, exp_dec_in;
26
27      wire nre1_out, nre2_out, adc_out, expose_out, erase_out;
28
29      parameter size = 2, exp_size = 5;
30
31
32      camera_control uut(         //Map the inputs and outputs of the
    testbench to the module
33          .init(init_in),
34          .reset(reset_in),
35          .clk(clk_in),
36          .exp_inc(exp_inc_in),
37          .exp_dec(exp_dec_in),
38          .nre1(nre1_out),
39          .nre2(nre2_out),
40          .expose(expose_out),
41          .erase(erase_out),
42          .adc(adc_out)
43          );
44
45      always
46          #0.5 clk_in = ~clk_in; // Generate a clock signal at 1kHz
47
48      initial
49          begin
50              clk_in = 1'b0;          //Reset clock and define initial values
51              reset_in = 1'b0;
52              init_in = 1'b0;
53              exp_inc_in = 1'b0;
```

File: c:/Users/Anders/Documents/AIM-spice/DIK-project/Verilog/DIC-project/src/camera_tb.v (/

```
54              exp_dec_in = 1'b1;
55              #1; reset_in = 1'b1;     //Reset is active-high
56              #1; reset_in = 1'b0;
57              #10; exp_dec_in = 1'b0;
58              #1; init_in = 1'b1;      //Initialise the camera operation
59              #1; init_in = 1'b0;
60              #1; exp_inc_in = 1'b1;
61              #5; reset_in = 1'b1;
62              #1; reset_in = 1'b0;
63              #5; init_in = 1'b1;
64              #1; init_in = 1'b0;
65              #40;
66              $finish;                 //$finish to stop the simulation after
    the 40ms delay
67          end
68
69      initial
70          begin                        //Define the monitor outputs. Writes
    the different states to the console. Using uut.variable enables the
    monitoring of internal signals in the camera controller.
71              $monitor("time = %2d, clk_in =%b, init_in=%b, reset_in=%b,
    exp_inc_in=%b, exp_dec_in=%b, nre1_out=%b, nre2_out=%b, expose_out=%b,
    erase_out=%b, adc_out= %b, exp_time= %d, exp_count = %d, adc_read_cycle =
    %d", $time,clk_in,init_in,reset_in,exp_inc_in,exp_dec_in,nre1_out,nre2_out,
    expose_out,erase_out,adc_out,uut.exp_time, uut.exp_count, uut.
    adc_read_cycle);
72          end
73  endmodule
74
```

D. Regular FSM behaviour

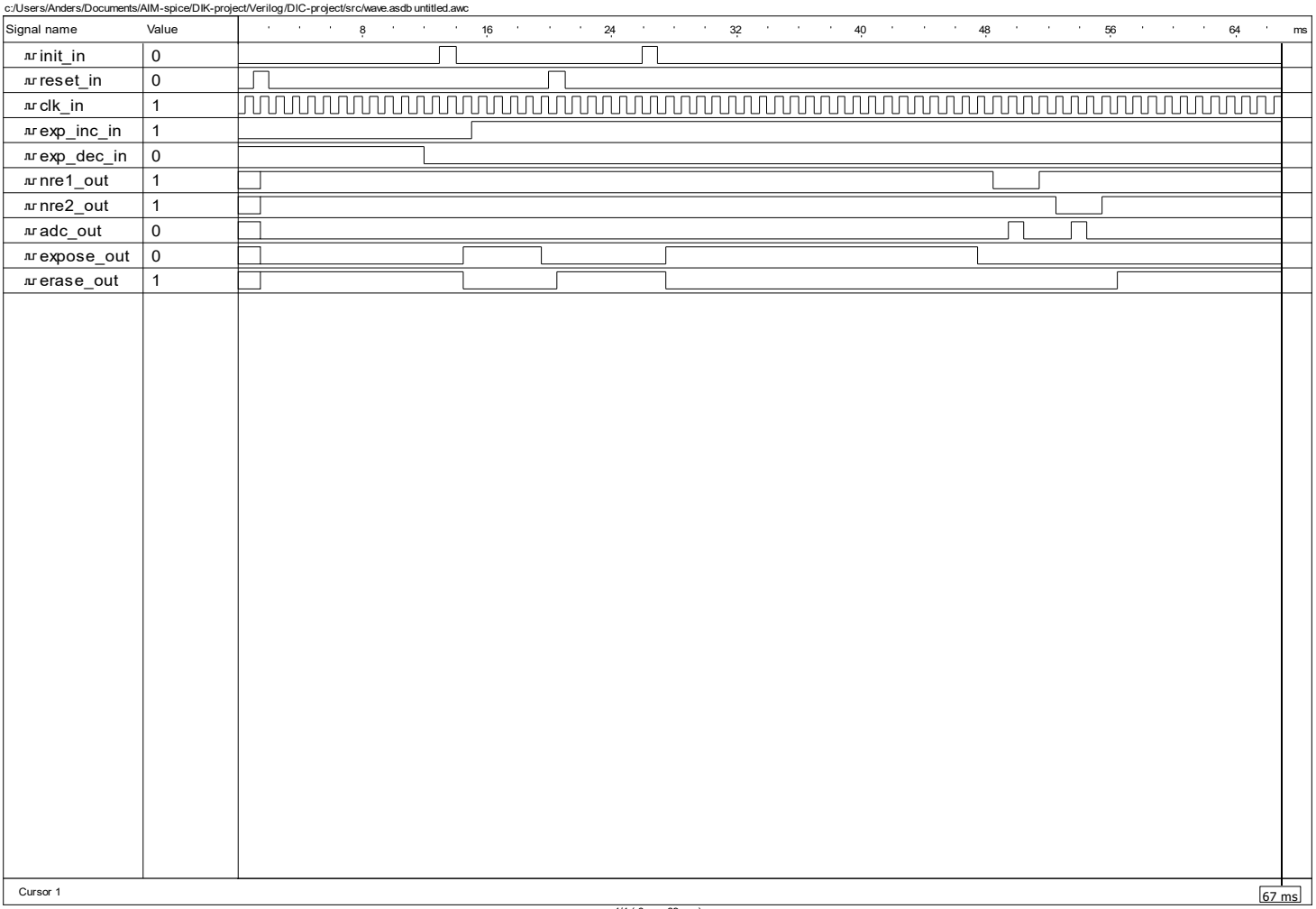
c:/Users/Anders/Documents/AIM-spice/DIK-project/Verilog/DIC-project/src/wave.asdb untitled.awc



1/1 (0 us - 69 ms)

E. Reset test and exposure change

c:/Users/Anders/Documents/AIM-spice/DIK-project/Verilog/DIC-project/src/wave.asdb untitled.awc



1/1 (0 us - 69 ms)