Project report Integrated Circuits

Andreas Bjørsvik, Gjermund Øfsti *2021-11-09*, v0.1.0

***Abstract*—When designing a digital photo sensor build up by multiple pixel sensors it is smart to simulate all the parts independent before testing the whole array. In SPICE, an analog circuit was modelled and tested by controlling the peripherals of the circuit. With a resistor used a model for the photo transistor, the results show that all though there is not actual photo transistor in this circuit, the circuit may be used on the same premise in the real world when swapping the resistor for a photo transistor. The parameter changing would then be the resistance of the photo transistor which gives different outputs for the sensor part. In SystemVerilog a 2x2 array of pixel sensors was simulated to test the control logic needed for controlling multiple pixel sensors outputting data to the same bus. The results for the pixel array shows that the control logic works as expected with expose time changing the magnitude of each sensor and the read signals driving the results on the bus in turn and order not conflicting with each other.**

I. Introduction

Most modern cameras using an on chip photo sensor. These photo sensors consist of multiple pixel sensors. Each of those pixel sensors is made by the same principals as the pixel in [1]. By the time that sensor was designed most of the pixel sensors had analog output. Analog sensors is more sensitive to noise and is harder to read. To make digital output of the sensor it needs to have an ADC on each sensor. To make the frame rate as used in [1], the analog value is saved in memory cells on each pixel.

This project is about how one of this pixel arrays can be designed and simulated. To do this it is smart to start simulating one parts of the system and add together more later. At first one pixel is tested in spice to check that the analog part works and send out digital values. When that is done an array of 2x2 pixels is tested in the digital in SystemVerilog.

II. Theory

A CMOS digital pixel sensor(DPS) as illustrated in (Figure 1), consist of a photo transistor, an analog to digital converter(ADC) and a memory cell. Firstly, the photo transistor produces a voltage increasing with the brightness of the light it is exposed to. The voltage can then be read by an ADC and converted into a digital value. In this step the exposure time comes in to play. The longer exposure time and the brighter the light is, the lower the voltage to be converted to digital is. After the ADC outputs a digital value, it is locked in place in the memory. The memory consists of 8 memory cells each holding 1 bit, which then gives $2^8 = 256$ possible values.

One digital pixel sensor is not very useful on its own but placing multiple DPSs into an array you can start to get an image with the resolution corresponding to the amount of DPSs in the array. The resolution will then be the number of sensors in the x direction times the amount in the y direction.

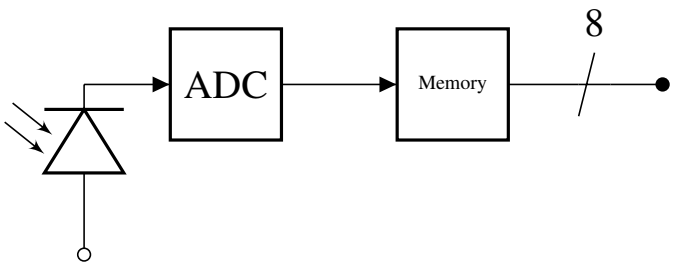


Fig. 1: Digital pixel sensor.

The quality of the image is dependent of both the resolution and the bit depth(number of bits per pixel). For this simple DPS the bit depth is 8 with no color channels, making the image black and white with relatively few gray-values.

The purpose of organizing each DPS with its own memory cell is that all sensors in the array may be instructed to expose at the same time and storing the value locally and read them in turn. This will produce a still image knowing all sensors captured at the same time, rather than exposing one pixel at a time, making a delay in which each pixel is created.

III. Implementation

A pixel sensor contains both digital parts and analog parts. To manage a good simulation for both, it is divided into two different parts. The analog part contains a photo transistor, a comparator and a memory part. In the digital simulation has a pixel part as well but it is simpler and is only there to give output to the rest of the circuit. In addition to the sensor the digital part has an array which connect four sensors. It also has a state machine to drive the order of actions and on the top a part which connects all together.

A. Analog

The analog circuit is modelled in spice like in (Figure 4). The circuit consist of three parts; a photosensor, a comparator and a memory with 8 memory cells. In this figure only one memory cell is modelled, but to get an 8-bit memory we simply copy the rightmost part(with the three transistors) 8 times.

The leftmost part, the photosensor is modelled using two transistors, a resistor and a capacitor. The resistor is a simplification to model the phototransistor. The $RESET$ input resets the voltage output V_S of the sensor to $ResetV$. The phototransistor is modelled with a transistor with input $EXPOSE$ to represent the exposure time. While $EXPOSE$ is high, the voltage over the capacitor will gradually discharge due to the RC circuit. Therefore with a higher exposure time(longer duration of high $EXPOSE$) a lower output voltage V_S will represent a brighter pixel(counter intuitive).

After $EXPOSE$ goes low, V_S is on the input of the gate of the comparator. The comparator should output a high value if the $Ramp$ input is lower than V_S. The transistor comparator circuit is a standard comparator driving the voltage low when ramp exceeds the value of V_S.

The comparator outputs a voltage to a current mirror to ensure the circuitry with the memory latches does not draw current comparator and interferes with its performance.

The output voltage of the current mirror is then fed to the 8 memory cells. While the $READ$ input pin is high, a high value is latched to the data output if the output on the comparator is over a certain value. This should be representing a digital value with depth 256 as it is 8-bits. As this is a strictly analog model, the ADC(comparator) is modelled to be a series of 1-bit ADCs in series with output doubled as we are using binary value. The output of the whole DPS is then an 8-bit digital bus.

B. Digital

The digital part of the circuit is separated into four different parts. On top of the hierarchy is the test bench which connects to the other parts of the circuit. The test bench is connected to a state machine and a pixel array, and the pixel array contains four of the same pixel sensors. A block diagram presentation of the digital part can be seen in (Figure 2). The blue rectangle represent one of the four pixels in the array.

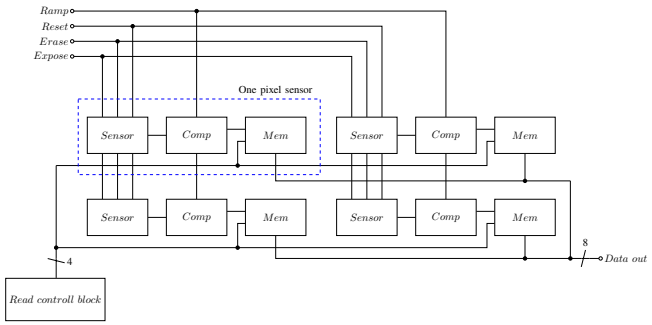


Fig. 2: Block diagram of digital circuit

The pixel sensor relates to control signals as input and data as output. The sensor is divided into sensor, comparator and memory latch. The sensor part is made to run if the expose control signal is high and then make a signal go higher and higher for every clock cycle it is high. This way of modeling the sensor is not representing the real world but it's done like this because the sensor is an analog component. A more real world representation of the sensor is done in the analog part. The comparator and memory latch are made to run one after the other. The comparator is driven by the ramp signal and the memory latch saves the value after the comparator is finished.

The pixel array is a simple part of the circuit. It only contains the logic for connecting the four sensors into an array. The different sensors have many of the control signals in common, and the only signals that is separate is the read signal and the data. To validate if the array works the different sensors have different prescaler on the light intensity value. That is because this is not a real world test so instead of making unique sensors with only one small change the value is different when the part is defined.

To drive the array, it is made a state machine. The state machine is made to drive out the control signals to drive the sensors in the right order. It is made to first erase the last read value, then expose and convert all the pixels at once. Then the out of the pixels is read one after the other, before the state machine resets, and everything goes over again. The names in the states shown in (Figure 3) is corresponding to which signals are driven high. (Figure 2) shows where the output signals from the state machine are connected.

As shown in (Figure 3) it is some wait statements between the states. The time it takes before each statement is finished is defined by parameters. The parameters are defined in the test bench. Each state has an independent time which are set by estimate on how long time the action will take.

The top of the circuit is the test bench. It contains some readout circuit creation of the ramp signal as well as some test bench stuff. The reason it is made like this is because the circuit that have the ramp circuit and the test bench in different parts did not work for an unknown reason.

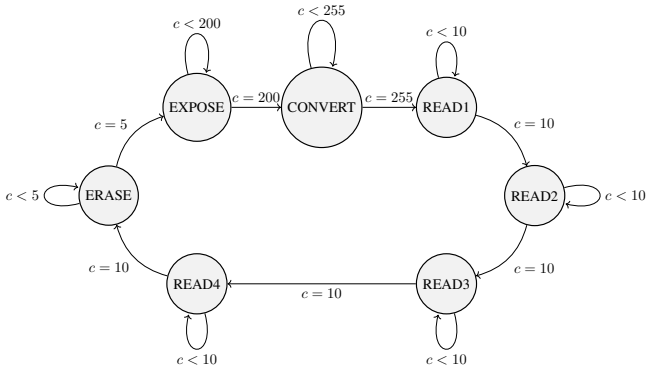


Fig. 3: Caption of the FSM

C. Analog

The analog circuit is modelled in spice like in (Figure 4). The circuit consist of three parts; a photosensor, a comparator and a memory with 8 memory cells. In this figure only one memory cell is modelled, but to get an 8-bit memory we simply copy the rightmost part(with the three transistors) 8 times.

The leftmost part, the photosensor is modelled using two transistors, a resistor and a capacitor. The resistor is a simplification to model the phototransistor. The $RESET$ input resets the voltage output V_S of the sensor to $ResetV$. The phototransistor is modelled with a transistor with input $EXPOSE$ to represent the exposure time. While $EXPOSE$ is high, the voltage over the capacitor will gradually discharge due to the RC circuit. Therefore with a higher exposure time(longer duration of high $EXPOSE$) a lower output voltage V_S will represent a brighter pixel(counter intuitive).

After $EXPOSE$ goes low, V_S is on the input of the gate of the comparator. The comparator should output a high value if the $Ramp$ input is lower than V_S. The transistor comparator

circuit is a standard comparator driving the voltage low when ramp exceeds the value of V_S.

The comparator outputs a voltage to a current mirror to ensure the circuitry with the memory latches does not draw current comparator and interferes with its performance.

The output voltage of the current mirror is then fed to the 8 memory cells. While the $READ$ input pin is high, a high value is latched to the data output if the output on the comparator is over a certain value. This should be representing a digital value with depth 256 as it is 8-bits. As this is a strictly analog model, the ADC(comparator) is modelled to be a series of 1-bit ADCs in series with output doubled as we are using binary value. The output of the whole DPS is then an 8-bit digital bus.

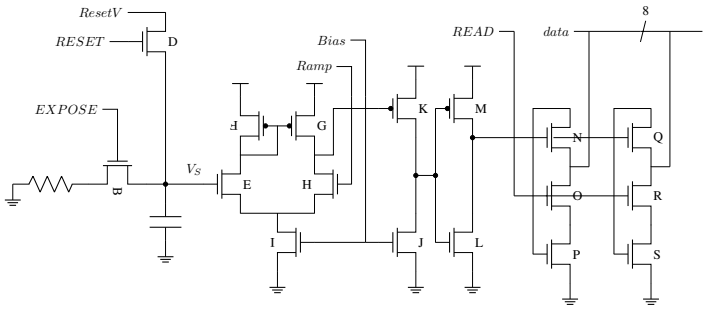


Fig. 4: Analog circuit

IV. Result

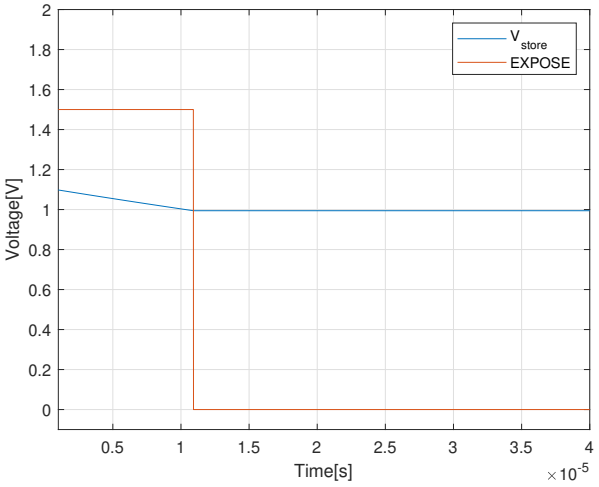
A. Analog

In the testbench for the circuit, all the different pins are modelled, and control logic is implemented. For the $ResetV$ and $Bias$ pins the value is constant. The other pins are related to the control logic. $RESET$ is the first value to be set high for duration $T_{RESET} = 0.5\mu s$. It resets the voltage on the V_S pin. Next the $EXPOSE$ pin is set high for a duration of T_{EXPOSE}. After exposing the pixel sensor, the ramp pin ramps from $0.4V$ to $1.4V$ linearly for a duration of $T_{CONVERT} = 25.5\mu s$. Lastly the signal can be read from the memory while $READ$ is high. After one cycle($T_RESET + T_{EXPOSE} + T_{CONVERT} + T_{READ}$) it all loops back around.

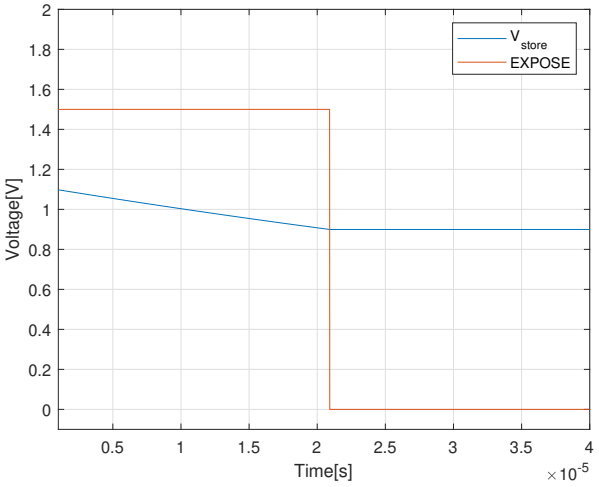
For testing the sensor part of the analog circuit, the expose time is varied to be $10\mu s$, $20\mu s$ and $30\mu s$ respectively. The voltage output of the sensor V_{store} is plotted with the expose duration in (Figure 5a), (Figure 5b) and (Figure 5c).

The output voltage of a pixel sensor is expected to have a linear relationship with the exposure time. In the results for the sensor the V_{store} voltage decreases linearly with respect to time while the expose pin is set high. When the expose pin goes low, the voltage is latched and constant.

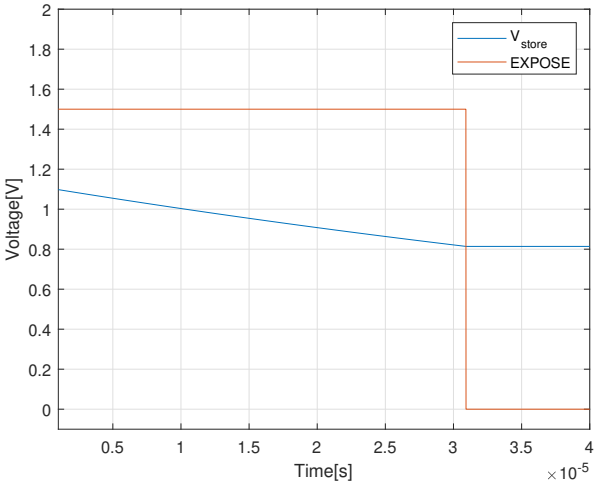
Next are the results for the comparator circuit. The expose time here is also set to be $10\mu s$, $20\mu s$ and $30\mu s$ respectively as in the test for the sensor. This gives three different voltages on the input V_{store} for the comparator. The comparator should output a high value as long as the V_{ramp} is lower than V_{store}. The results for the comparator is shown in (Figure 6a), (Figure 6b) and (Figure 6c).



(a) Test result for sensor with expose time $10\mu s$.



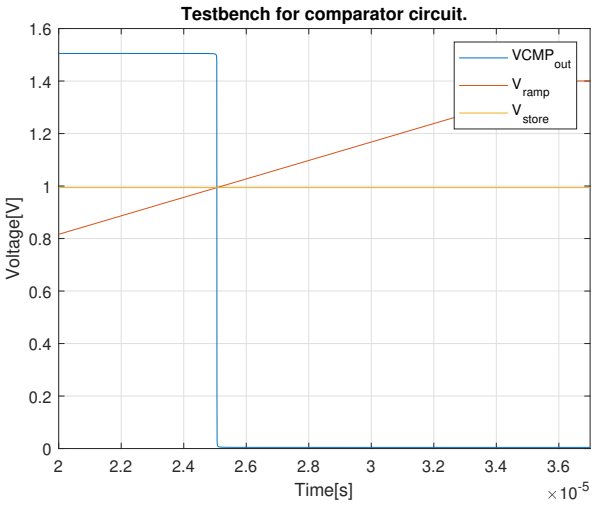
(b) Test result for sensor with expose time $20\mu s$.



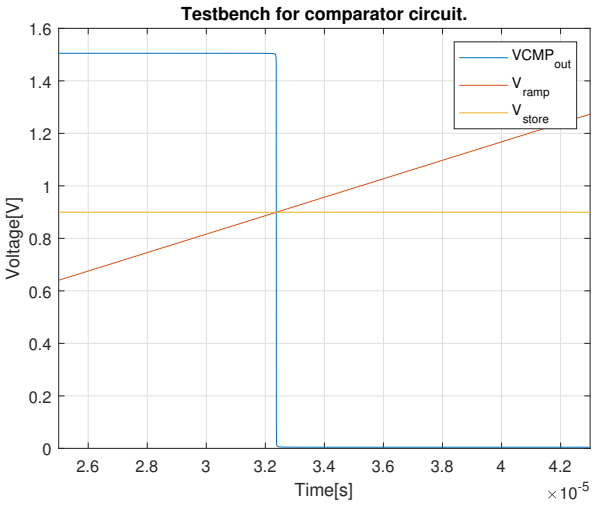
(c) Test result for sensor with expose time $30\mu s$.

Fig. 5

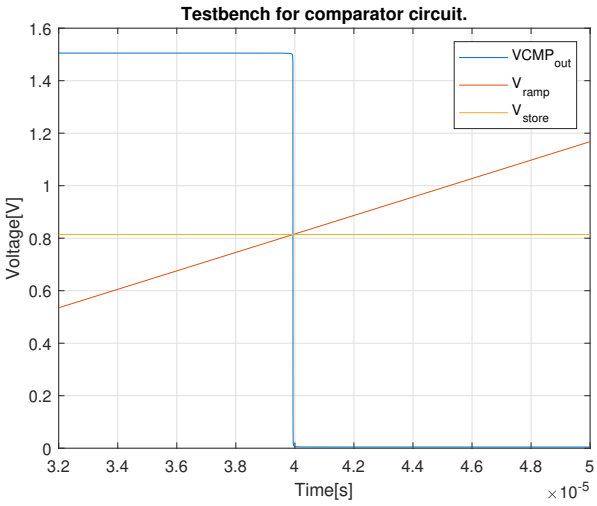
For the databus output the expose time is again varied between $10\mu s$, $20\mu s$ and $30\mu s$. The data pins for $T_{EXPOSE} =$



(a) Test result for sensor with expose time $10\mu s$.



(b) Test result for sensor with expose time $20\mu s$.



(c) Test result for sensor with expose time $30\mu s$.

Fig. 6

$10\mu s$ are plotted in (Figure 7). The pins represent a digital amplitude $Data_{out} = 128 * bit1 + 64 * bit2 + 32 * bit3 + 16 * bit4 + 8 * bit5 + 4 * bit6 + 2 * bit7 + 1 * bit8 = 45$. For the run with $T_{EXPOSE} = 20\mu s$ the output value is $Data_{out} = 0b01010101 = 85$ and for the run with $T_{EXPOSE} = 30\mu s$ the output value is $Data_{out} = 0b01111010 = 122$.

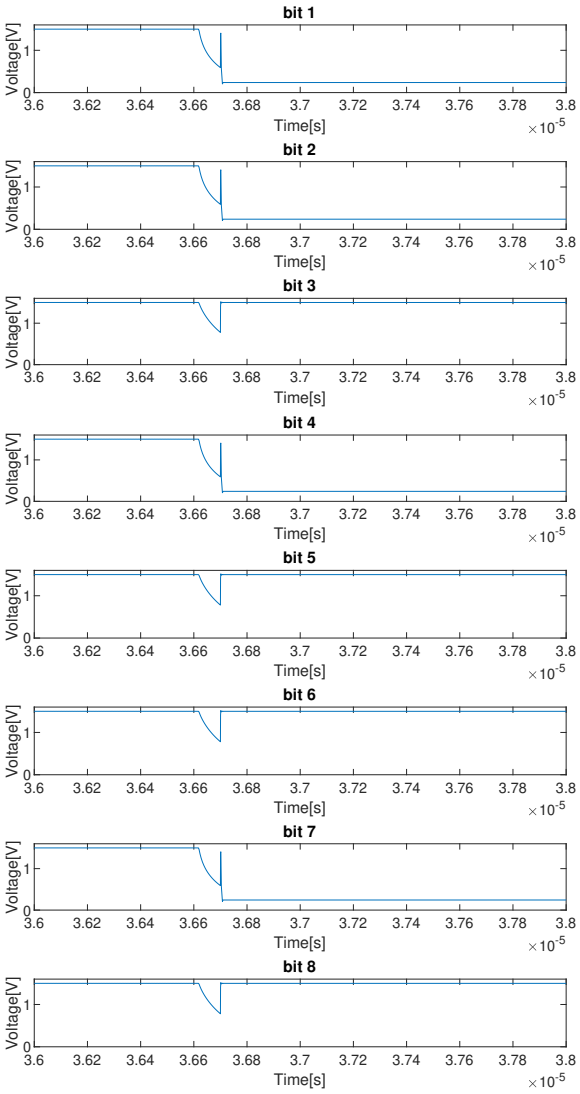


Fig. 7: Logic values for the bits on the 8 pin I/O ̈data bus ̈with $T_{EXPOSE} = 100$.

B. Digital

The test bench for the digital part contains some parts to connect the different parts of the circuit together in addition to a test bench part. The part that actually drives the circuit is small. It first sets the reset signal high for one clock cycle before it makes an output file, wait 1200 clock cycles and stops the simulation. This time is set so the simulation manages to

finish one full cycle with some margin. The output file is a wave form where all the signals in the circuit can be shown. The output made from the simulation with a expose time at 200 can be seen in (Figure 8). To easier see that the pixels give out different values it is coded inn different scaling values on how strong signal the pixels give out.

Fig. 8: multiplexing of data out

The digital part of the circuit is made to have one output signal that is time multiplexed with the different pixel outputs. The state machine has four different read states where the value that is saved on the ram on the pixels is written to the data out bus. The output from one read cycle can be seen in (Figure 8). It can be seen that the $pixelDataOut$ change value after the $read$ signal is set high. This shows that the read signals work and manage to shift the signals on the bus.

To validate that the output in the wave form (Figure 8) is correct the expected result can be calculated. The value is given by how much light the sensor detects, the more light it detects the lower the value gets. Since this is a simulation, it is a value that get lower for every clock cycle the $expose$ signa is set. The max value the sensor can gives out is 5 and on pixel 2 the constant value is $0.5 * 0.8 * (5/255)$, and when that is subtracted 200 times from the max value it gives out 4.0196. The 200 times comes from the number of clock cycles the expose is set high. When we take 4.0196 through the ADC by multiplying by $255/5$, it gives a value of 205. Converted to hex that is CD and that is that same value that can be seen between $480\mu s$ and $490\mu s$ in (Figure 8).

To test the circuit with changed values out the expose time can be changed. When the expose time goes down the light reaching the sensor goes down and the value out from the sensor goes up. (Figure 9) shows a simulation when the expose time is 50. That is 4 times shorter than the expose of 200 in (Figure 8). The readout from the first pixel can be seen between $470\mu s$ and $480\mu s$ in (Figure 8) and between $320\mu s$ and $330\mu s$. With expose of 200 the hex value is 9B and with expose of 50 it gives out the hex value of E6. Since 0 expose gives output of 255 the given output needs to be subtracted from the max to see how much it changes with the expose time. $255 - E6 = 255 - 230 = 25$ and $255 - 9B = 255 - 155 = 100$, this shows that 4 times larger expose gives 4 times larger decreasing of the output.

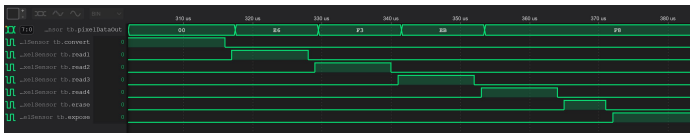


Fig. 9: multiplexing of data out with low expose time

V. Discussion

A. Analog

The test results for the sensor part of the circuit as shown in (Figure 5a), (Figure 5b) and (Figure 5c) is as mentioned expected to output a value which is linear with the exposure time. We can see that V_{store} decreases $0.1V$ for each $10\mu s$. In this circuit there is no actual photo transistor, but rather a resistor to model the light intensity. To test the circuit, only the exposure time was changed between the runs. To test the actual pixel sensor, an actual photo transistor should replace the resistor to vary the output, and the exposure time should be constant. This however is an okay way to test that the rest of the circuit works as expected. For the comparator part of the circuit the test result are shown in (Figure 6a), (Figure 6b) and (Figure 6c). We see from these results that as soon as the V_{ramp} voltage is larger than V_{store} the output value of the comparator is low as expected and intended. We also see that this is the case for all runs, regardless of what V_{store} is. The point of the comparator is to be able to convert the analog input voltage into a ”digital” 8-bit binary value. This is handled by the ADC in the test bench. The ADC is modelled using 8 1-bit ADCs in series. We can read all these binary values as shown in (Figure 7) and in the results text for the changed exposure time. From $T_{EXPOSE} = 10\mu s$ to $T_{EXPOSE} = 20\mu s$ the output value increases by 40 when converted to decimal. For the next step the output value increases by 37. This is not a linear relationship, but it is close. The reason why this may be is the way the sensor is implemented. It's hard to see, but the discharge rate of the sensor while $EXPOSE$ is high is not strictly linear. We therefore expect that the model of the sensor is not perfect, and it affects the output results.

B. Digital

The result shown in (Figure 8) shows that the state machine gives out the right signals in the right order. The wave form starts with convert and then the different read signals goes high, before the cycle start over again with erase and expose. The wave form also shows that some values are sent to the data bus after the read signals is high. The values do not give any sense without knowing the values given in the test bench. One of the result is explained and calculated in IV-B. The calculation there shows that the values are correct. The most important result from the pixel array testing is the ability to simultaneously expose and convert each pixel sensor. If we were to expose and convert the value for each pixel sensor after one another, the frame rate of the system would have to be drastically decreased. Here the task is paralleled and increases the frame rate almost four times for a 2x2 array as the read time is negligible compared to the expose and covert time.

VI. Future work

This project have simulated a 2x2 array of pixel sensors, it does not make a that much sense to make a image sensor for capturing images with only four pixels. So for making a more realistic image sensor the array should be implemented with

more pixels. It is not hard to implement a larger array, but it is time consuming.

The analog circuit is of course missing some connection to real life as the photo transistor is only modelled as a resistor. Further implementation would require a physical circuit to be able to test that we actually would be able to capture an image and recreate it from the digital values.

The result shows that the system work without any optimizing. The duration for the convert time, read and erase is just set to a value where we was sure it would work. This could be optimized to make the reading be a bit faster.

VII. Conclusion

In SPICE, an analog circuit was modelled and tested using a test bench for controlling the peripherals of the circuit. The spice simulation is done to ensure a model of a single pixel sensor is correct before working further on a larger scale project with an array of sensors. The results show that all though there is not actual photo transistor in this circuit, the circuit may be used on the same premise in the real world when swapping the resistor for a photo transistor. The parameter changing would then be the resistance of the photo transistor which gives different outputs for the sensor part. In SystemVerilog an array of pixel sensors was simulated to test the control logic needed for controlling multiple pixel sensors outputting data to the same bus. The array of pixel sensors was implemented as digital models of the analog circuit with different ”intensity” added to be able to test the digital logic. The results for the pixel array shows that the control logic works as expected with expose time changing the magnitude of each sensor and the read signals driving the results on the bus in turn and order not conflicting with each other.

VIII. Appendiks

The files used for the project can be found on https://github.com/Bjorsvik98/ProjectTFE4152delivery

Verilog code for the sensor

```
module PIXEL_SENSOR
  (
   input logic       VBN1,
   input logic       RAMP,
   input logic       RESET,
   input logic       ERASE,
   input logic       EXPOSE,
   input logic       READ,
   inout [7:0] DATA

   );
   real              v_erase = 1.2;
   real              lsb = v_erase/255;
   parameter real    dv_pixel = 0.5;

   real              tmp;
   logic             cmp;
   real              adc;

   logic [7:0]       p_data;


   // ERASE

   // Reset the pixel value on pixRst
   always @(ERASE) begin
      tmp = v_erase;
      p_data = 0;
      cmp  = 0;
      adc = 0;
   end


   // SENSOR

   // Use bias to provide a clock for
   // integration when exposing
   always @(posedge VBN1) begin
      if(EXPOSE)
        tmp = tmp - dv_pixel*lsb;
   end


   // Comparator

   // Use ramp to provide a clock for
   // ADC conversion, assume that ramp
   // and DATA are synchronous
   always @(posedge RAMP) begin
      adc = adc + lsb;
      if(adc > tmp)
        cmp <= 1;
   end


   // Memory latch

   always_comb  begin
      if(!cmp) begin
         p_data = DATA;
      end

   end


   // Readout

   // Assign data to bus when pixRead = 0
   assign DATA = READ ? p_data : 8'bZ;

endmodule // re_control
```

Verilog code for pixel array

```
module PIXEL_ARRAY
    (
 input logic anaBias1,
 input logic anaRamp,
 input logic anaReset,
 input logic erase,
 input logic expose,
 //input logic[3:0] read,
 input logic read1,read2,read3,read4,
 inout [7:0] pixData1,
 inout [7:0] pixData2,
 inout [7:0] pixData3,
 inout [7:0] pixData4
 );
 parameter real   dv_pixel = 0.5;


 PIXEL_SENSOR  #(.dv_pixel(dv_pixel))
 ps1(anaBias1, anaRamp, anaReset,
 erase,expose, read1,pixData1);
 PIXEL_SENSOR  #(.dv_pixel(dv_pixel*0.5))
 ps2(anaBias1, anaRamp, anaReset,
 erase,expose,read2,pixData2);
 PIXEL_SENSOR  #(.dv_pixel(dv_pixel*0.8))
 ps3(anaBias1, anaRamp, anaReset,
 erase,expose,read3,pixData3);
 PIXEL_SENSOR  #(.dv_pixel(dv_pixel*0.3))
 ps4(anaBias1, anaRamp, anaReset,
 erase,expose,read4,pixData4);

endmodule
```

Verilog code for FSM

```
`timescale 1 ns / 1 ps

module pixelSensorFsm(
                      input  logic clk,
                      input  logic reset,
                      output logic erase,
                      output logic expose,
```

```
                output logic read1,
                output logic read2,
                output logic read3,
                output logic read4,
                output logic convert
            );


    //State duration in clock cycles
  parameter integer c_erase = 5;
  parameter integer c_expose = 255;
  parameter integer c_convert = 25;
  parameter integer c_read = 100;

  // State Machine
parameter ERASE=0, EXPOSE=1, CONVERT=2,
READ1=3, READ2=4, READ3=5, READ4=6, IDLE=7;

  // logic               convert;
  // logic               convert_stop;
  logic [2:0]         state,next_state;
  //States
  integer           counter;
  //Delay counter in state machine

  // Control the output signals
  always_ff @(negedge clk ) begin
     case(state)
       ERASE: begin
          erase <= 1;
          read1 <= 0;
          read2 <= 0;
          read3 <= 0;
          read4 <= 0;
          expose <= 0;
          convert <= 0;
       end
       EXPOSE: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 0;
          read3 <= 0;
          read4 <= 0;
          expose <= 1;
          convert <= 0;
       end
       CONVERT: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 0;
          read3 <= 0;
          read4 <= 0;
          expose <= 0;
          convert = 1;
       end
       READ1: begin
          erase <= 0;
          read1 <= 1;
          read2 <= 0;
          read3 <= 0;
          read4 <= 0;
          expose <= 0;
          convert <= 0;
       end
       READ2: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 1;
          read3 <= 0;
          read4 <= 0;
          expose <= 0;
          convert <= 0;
       end
       READ3: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 0;
          read3 <= 1;
          read4 <= 0;
          expose <= 0;
          convert <= 0;
       end
       READ4: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 0;
          read3 <= 0;
          read4 <= 1;
          expose <= 0;
          convert <= 0;
       end
       IDLE: begin
          erase <= 0;
          read1 <= 0;
          read2 <= 0;
          read3 <= 0;
          read4 <= 0;
          expose <= 0;
          convert <= 0;
       end
     endcase // case (state)
   end // always @ (state)

    // Control the state transitions
always_ff @(posedge clk or posedge reset)
  begin
  if(reset) begin
     state = IDLE;
     next_state = ERASE;
     counter  = 0;
     convert  = 0;
  end
  else begin
     case (state)
       ERASE: begin
          if(counter == c_erase) begin
```

```
            next_state <= EXPOSE;
            state <= IDLE;
          end
        end
        EXPOSE: begin
          if(counter == c_expose) begin
            next_state <= CONVERT;
            state <= IDLE;
          end
        end
        CONVERT: begin
          if(counter == c_convert) begin
            next_state <= READ1;
            state <= IDLE;
          end
        end
        READ1:
          if(counter == c_read) begin
            state <= IDLE;
            next_state <= READ2;
          end
        READ2:
          if(counter == c_read) begin
            state <= IDLE;
            next_state <= READ3;
          end
        READ3:
          if(counter == c_read) begin
            state <= IDLE;
            next_state <= READ4;
          end
        READ4:
          if(counter == c_read) begin
            state <= IDLE;
            next_state <= ERASE;
          end
        IDLE:
          state <= next_state;
      endcase // case (state)
      if(state == IDLE)
        counter = 0;
      else
        counter = counter + 1;
  end
end // always

endmodule // test
```

Spice kode for the sensor

```
.SUBCKT PIXEL_SENSOR VBN1 VRAMP VRESET
ERASE EXPOSE READ
+ DATA_7 DATA_6 DATA_5 DATA_4 DATA_3
DATA_2 DATA_1 DATA_0 VDD VSS

XS1 VRESET VSTORE ERASE EXPOSE VDD
VSS SENSOR

XC1 VCMP_OUT VSTORE VRAMP VDD VSS COMP

XM1 READ VCMP_OUT DATA_7 DATA_6 DATA_5
DATA_4 DATA_3 DATA_2 DATA_1 DATA_0 VDD
VSS MEMORY

.ENDS

.SUBCKT MEMORY READ VCMP_OUT
+ DATA_7 DATA_6 DATA_5 DATA_4 DATA_3
DATA_2 DATA_1 DATA_0 VDD VSS

XM1 VCMP_OUT DATA_0 READ VSS MEMCELL
XM2 VCMP_OUT DATA_1 READ VSS MEMCELL
XM3 VCMP_OUT DATA_2 READ VSS MEMCELL
XM4 VCMP_OUT DATA_3 READ VSS MEMCELL
XM5 VCMP_OUT DATA_4 READ VSS MEMCELL
XM6 VCMP_OUT DATA_5 READ VSS MEMCELL
XM7 VCMP_OUT DATA_6 READ VSS MEMCELL
XM8 VCMP_OUT DATA_7 READ VSS MEMCELL

.ENDS

.SUBCKT MEMCELL CMP DATA READ VSS
M1 VG CMP DATA VSS nmos  w=0.2u  l=0.13u
M2 DATA READ DMEM VSS nmos  w=0.4u
l=0.13u
M3 DMEM VG VSS VSS nmos  w=1u  l=0.13u
C1 VG VSS 1p
.ENDS

.SUBCKT SENSOR VRESET VSTORE ERASE EXPOSE
VDD VSS

* Capacitor to model gate-source capacitance
C1 VSTORE VSS 100f
Rleak VSTORE VSS 100T

* Switch to reset voltage on capacitor
BR1 VRESET VSTORE I=V(ERASE)*V(VRESET,VSTORE)/1k

* Switch to expose pixel
BR2 VPG VSTORE I=V(EXPOSE)*V(VSTORE,VPG)/1k

* Model photocurrent
Rphoto VPG VSS 1G
.ENDS

.SUBCKT COMP VCMP_OUT VSTORE VRAMP VDD VSS

* Model comparator
BC1 VCMP_OUT VSS V = ((atan(100000*(V(VSTORE)
- V(VRAMP)))) + 1.58)/3.14*1.5


.ENDS
```

REFERENCES

[1] Kleinfelder, Lim, Liu, Gamal “A 10 000 Frames/s CMOS Digital Pixel Sensor”, JSSC, VOL 36, NO 12, 2001