Bar-Ilan University

Faculty of Engineering

Designing an Image-Sensor Array for Hardware-Implemented Neural Network Applications

Daniel Yoffe

Ben Zilberberg

Submitted as part of the requirements for the bachelor's degree in the Faculty of Engineering, Bar-Ilan University

October 2022

Dedication

This work was carried out under the supervision of

Prof. Alex Fish and Mr. Yuval Ninyo,

At the Faculty of Engineering and ENICS,

Bar-Ilan University.

Acknowledgement:

The authors would like to acknowledge the advice, guidance and support of their supervisors, Prof. Alex Fish and Mr. Yuval Ninyo, as well as the assistance of Dr. Yoav Weizman , Mrs. Noa Edri and Mr. Matan Assaf.

Table Of Contents

1. Abstract

Photodiodes are a commonly used device in multiple applications, from image sensing circuitry in electro optical devices to energy harvesting applications like solar panels.

New types and structures or photodiodes are emerging, but little research has been done on the Weight-Based Current Assisted Photonic Demodulator device and information about its properties and capabilities are scarce.

In this work, a model of a current-division-capable photo-sensing device is proposed and implemented within a photo-sensing array. Additionally, a novel modified 4T pixel design is demonstrated, made to complement the unique operation of the presented photo-sensing device. Different ways of operating the array to achieve different applications are shown and discussed. The way in which these different applications may be achieved depends on the particular unique connectivity between the photo-sensing devices and the image sensing pixels implemented. The implementation of a variety of applications at the array level of the image sensing system makes this design a candidate for smart image sensing applications including neuromorphic applications, allowing for decreased computation time, power consumption and complexity of the system.

Different connectivity configurations combined with additional logical circuits make this device a candidate for application of various image processing algorithms at the pixel level, which generally decrease the computation time, power consumption and complexity of and image sensing system. The concept of dividing photo-current among neighboring pixels is new as far as we know and it’s the main innovation of this work.

Modified variation of the 4T pixel is designed to support the operation of the WBCAPD array. The modified pixel is connected to four photodiodes belonging to four surrounding WBCAPD devices. The amount of photo-current it collects from each surrounding WBCAPD device is proportional to the reconfigurable relative weights of each photodiode.

This work will focus on the modeling of the WBACPD device and modified 4T pixel, implementation of image sensing array and utilization of the spatial connectivity. Also, an implementation of edge detection algorithm using the spatial connectivity trait of the array will be simulated in the Cadence Virtuoso environment.

2. Image Sensing - Overview

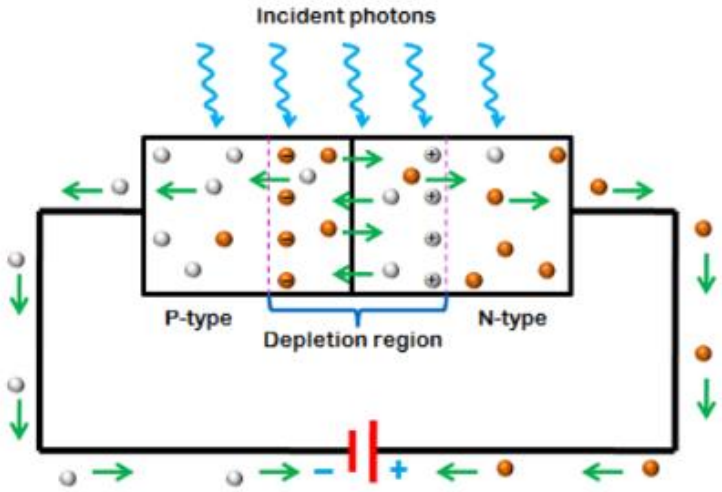
An Image Sensor is a type of sensor which transfers and processes light from a scene to form an image. It does so by converting the incoming light into an electrical signal, using a photo-sensitive device such as a photodiode, and later processing it. There are two main category of image sensor; Charge Coupled Devices and CMOS image sensors. This work will focus on the latter.

2.1 Photodiodes

A photodiode is a light sensitive semiconductor device. It produces current by generating electron-hole pairs when it absorbs light. This current can be used to transfer information about the scene, illuminating the photodiode.

2.1.1 PN Junction Photodiode

Reverse biasing a PN junction, causes its depletion region to grow larger. Applying a certain amount of volage across the reverse biased PN junction will yield the maximal depletion width. Incoming light absorbed in the depletion region will form electron-hole pairs, which will be swept by the electric field, to the cathode and another , respectively. The width of the depletion region will act as a function of the incoming light intensity. Later, an additional circuitry can be designed to sense a change in the voltage across the PN junction and transfer information about the scene.

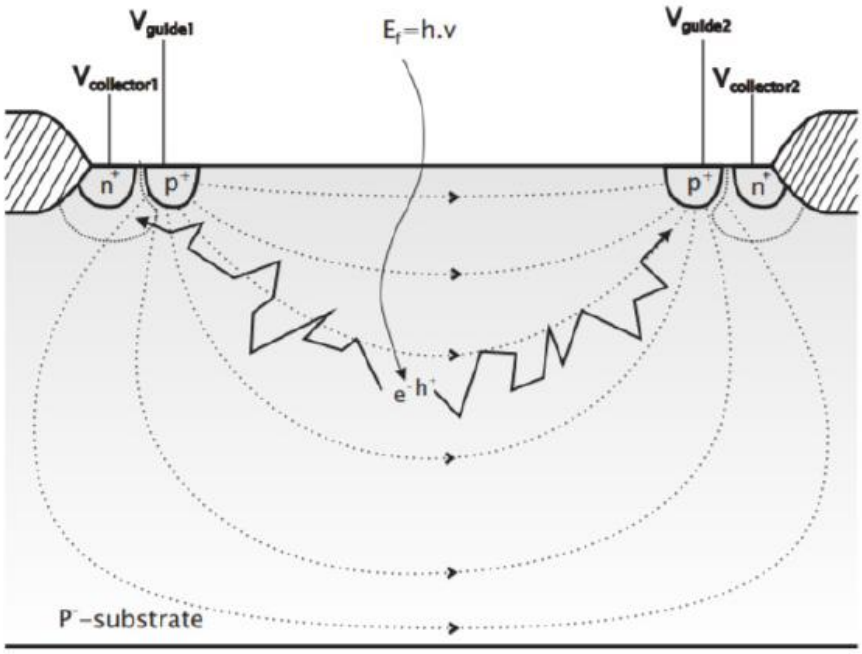


A reverse biased PN junction. The incoming light generate electron-hole pairs, which are swept by the electric field, respectively.

2.1.2 Current Assisted Photonic Demodulator (CAPD)

The CAPD is multi ported photodiode. This means it has multiple collection terminal that collect photo generated current. The first CAPD had two collection terminals, however, today there are CAPDs with four and more collection terminals.

Two ported CAPD Operation:
Each terminal has one modulation junction (P+ diffusion) and one collection junction (N+ diffusion) as seen in the image below. One modulation junction is biased with high voltage while the other modulation junction is grounded. An electric field between the terminals is created which causes a majority current (holes, if the substrate is a P type) flow between the two modulation terminals. Electrons, respectively, will flow in the opposite direction of the electric field toward the modulation junction with the highest voltage. Once electrons approached the modulation junction, the collection junction drift field (generated by the modulation-collection junction depletion region) will sweep them into the collection junction. Each collection junction can be looked at as a photodiode cathode of a PN junction between the collection and the modulation junction.

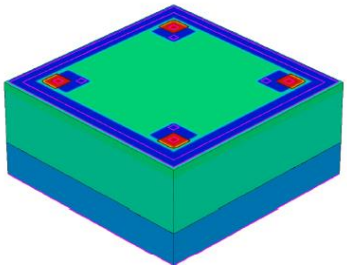


A two terminal CAPD. Each terminal has one collection (N+) and one module (P+) junction. The generation photo-current of electrons will into the junction biased with a higher voltage. [5]

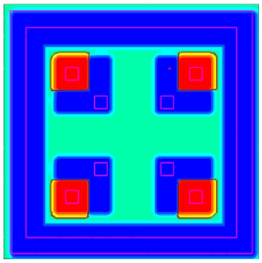
2.1.3 Weight-Based Current Assisted Photonic Demodulator (WBCAPD)

The idea behind Weight Based CAPD is being able to control the percentage of generated photo-current that flows into each collection junction as a function of the modulation junction voltage. The modulation junctions of the WBCAPD device can be biased with an external signal (e.g., applying constant voltage on certain collection junction) or with an internal signal (e.g., input from neighboring pixel). This work will use the proposed design of the WBCAPD in [1].

The proposed design of the WBCAPD to be used, is a four terminal WBCAPD. The modulation and collection junctions will be implanted on a P-type substrate as show below:



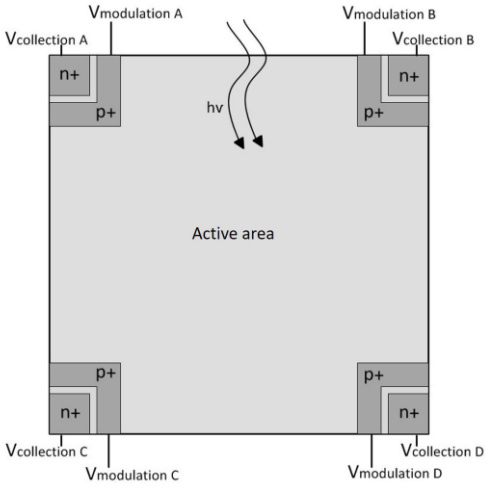
Side view of the WBCAPD. Taken from [1] – figure 4.7



Top view of the WBCAPD. Taken from [1] - figure 4.4

The P+ modulation junction as marked blue, the N+ collection junctions are marked red rectangles. The incoming light will generate photo-current. Depending on the voltage applied on each modulation junction (i.e., "**Weight**") each junction will draw current as a function of the relative voltage across that junction.

This can be shown by the equation: $I_A = \frac{V_A}{V_A+V_B+V_C+V_D} \cdot i_{photo}$.



Top view of four port WBCAPD. Taken from [1] – figure 3.3.

2.2 Pixel Physics

2.2.1 Overview

A pixel is the smallest addressable unit in an image sensing array. Each pixel performs control over a photodiode and can be addressed to perform readout. Each pixel output will correspond to a fraction of the scene it's photodiode captures.

2.2.2 Charge Mode pixel operation mode

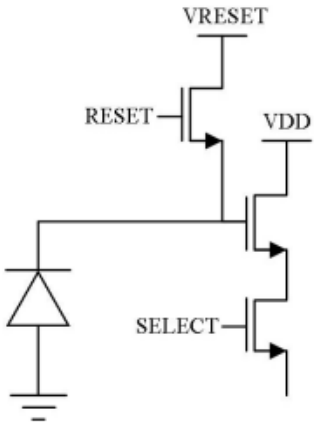
A charge-mode pixel is made of a photodiode, a reset transistor a source follower transistor and a row select transistor. The charge mode operation has three stages: rest, photodiode integration and readout. During the integration time the photo-current collected by the photodiode is accumulated by discharging the voltage set onto the photodiode at reset.

2.2.3 Active Pixel Sensor (APS)

An active pixel sensor consists of an array of pixels, each containing one or more MOSFET amplifier which convert the photo generated charge to voltage, amplify the signal voltage and reduce noise.

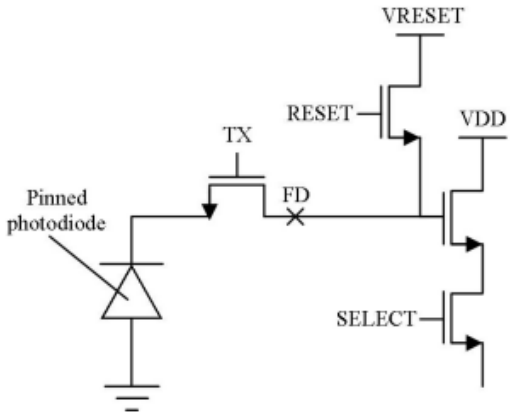
- 3T APS

  The 3T active pixel sensor operates in charge operation mode. The illuminated light generates electron-hole pairs which discharge the photodiode pre-reset voltage. The latter value is amplified using a source follower and read out through the Row Select transistor using external circuitry [6].



Common 3T APS (from [6] – figure 3)

- 4T APS
  The 4T active pixel sensor architecture uses additional three elements to the common 3T design: a transfer gate transistor (**'TX'**), a floating diffusion (**'FD'**) node and a pinned photodiode instead of normal one photodiode. The PPD is reset to V_{rst} by opening the RESET transistor. After the reset is complete, **TX** transfer gate transistor is closed, and the integration of the photodiode begins. After said integration time, the transfer gate is opened again, transferring the accumulated charge to **FD** for amplification at the source follower and later readout. The implementation of the floating diffusion allows for the sampling of the reset value, for later noise reduction from the reset transistor.



4T pixel architecture (from [6] – figure 4)

2.3 Neuromorphic engineering

Neuromorphic engineering is an interdisciplinary field which tries to mimic the functionality of the brain. In this field, the brain's architecture of neuron connection, which is very wide and complex, is implemented onto ICs []. Though computers are faster than individual neuron, they are not match to a net of neurons when it comes to image sensing and pattern recognition processes. Thus, a high interest in ICs that imitate the brain's wide connectivity and computation ability, emerges.

2.3.1 Edge Detection

Vision systems rely on the transformation of flux of photons into an array intensity values. Physical boundaries of an object captured in an image, or its Edges, are one of its most important properties. An edge of an image can be defined numerically as the intensity change in the array, capturing the information of the image.

The goal of an edge detection process is the characterization of intensity changes of an image in terms of the physical events that created them [3] (e.g., generated photo-current from a captured scene).

In modern devices, determination of physical edges is highly applicable in many computational vision applications.

This work will use a simple edge detection algorithm, based on a spatial averaging of the scene. To achieve to resulted edges, the image is first captured using traditional image sensing operation and its illumination intensity values are stored in an array. Later, the intensity values are averaged spatially. Subtraction of the values that originated directly from the scene from the spatially averaged values will result in edges [4].

3. Objectives and goals:

This work will try to achieve the following objectives using TSMC18 (CMOS 180nm) technology, using Cadence Virtuoso environment:

A. Designing a Verilog-A Current Division model capable of dividing input current as a function of its input voltage according to $I_j = \frac{V_j}{V_j+V_k+V_l+V_m} \cdot i\_in$.

B. Designing a WBCAPD model capable of emulating a four-port WBCAPD unit.

C. Designing a Pixel circuit model to support the operation of the WBCAPD array and allowing the spatial connectivity of neighboring WBCAPDs and pixels.

D. Designing a WBCAPD array capable of performing image sensing and feedback operation.

E. Implementation of Edge Detection algorithm using the designed WBCAPD image sensing array.

4. WBCAPD Modeling

4.1 WBCAPD Verilog-A – Ideal Mathematical Model

In order to achieve the desired operation of weight-based current division, a mathematical model was created by the implementation of a Verilog-A module:

```
`include "constants.vams"
`include "disciplines.vams"

module current_divider(vdd, vss, v_in[0:3], i_in, i_p[0:3], i_m[0:3]); //Symbol Port Decleration

//I/O decleration for symbol ports
inout vdd, vss, i_p[0:3], i_m[0:3];
input v_in[0:3], i_in;
//Port Type (Voltage / Current)
voltage vdd, vss, v_in[0:3];
current i_in, i_p[0:3], i_m[0:3];

//Main
analog begin

    if (V(v_in[0]) + V(v_in[1]) + V(v_in[2]) + V(v_in[3]) == 0) begin
        I(i_p[0]) <+ (0.25*I(i_in));
        I(i_p[1]) <+ (0.25*I(i_in));
        I(i_p[2]) <+ (0.25*I(i_in));
        I(i_p[3]) <+ (0.25*I(i_in));

        I(i_m[0]) <+ (-I(i_p[0]));
        I(i_m[1]) <+ (-I(i_p[1]));
        I(i_m[2]) <+ (-I(i_p[2]));
        I(i_m[3]) <+ (-I(i_p[3]));
        end
    else begin
        I(i_p[0]) <+ ( V(v_in[0]) / ( V(v_in[0]) + V(v_in[1]) + V(v_in[2]) + V(v_in[3]) ) ) * (I(i_in));
        I(i_p[1]) <+ ( V(v_in[1]) / ( V(v_in[0]) + V(v_in[1]) + V(v_in[2]) + V(v_in[3]) ) ) * (I(i_in));
        I(i_p[2]) <+ ( V(v_in[2]) / ( V(v_in[0]) + V(v_in[1]) + V(v_in[2]) + V(v_in[3]) ) ) * (I(i_in));
        I(i_p[3]) <+ ( V(v_in[3]) / ( V(v_in[0]) + V(v_in[1]) + V(v_in[2]) + V(v_in[3]) ) ) * (I(i_in));

        I(i_m[0]) <+ (-I(i_p[0]));
        I(i_m[1]) <+ (-I(i_p[1]));
        I(i_m[2]) <+ (-I(i_p[2]));
        I(i_m[3]) <+ (-I(i_p[3]));
        end
end
endmodule
```

This module has a total of 15 ports:

- $\boldsymbol{vdd}, \boldsymbol{vss}$ - general voltage sources (where: $vdd = 1.8[V], vss = 0$).
- $\boldsymbol{i\_in}$ – the generated photo-current (proportional to image brightness).
  Disclaimer: input current **i_in** is proportional to photo-current generated in the active area of the WBCAPD, it is used due to the limitation of the Virtuoso environment.
- $\boldsymbol{v\_in}[\mathbf{0}:\mathbf{3}]$ – input voltage (“weights”)
- $\boldsymbol{i\_p}[\mathbf{0}:\mathbf{3}]$ – current input to device from current supply.
- $\boldsymbol{i\_m}[\mathbf{0}:\mathbf{3}]$ – current output from device.

The WBCAPD Module has a current (photo-current) input and four voltage inputs (weights). The module performs a mathematical weight ratio computation by: $I_j = \frac{\dot{V_j}}{V_j+V_k+V_l+V_m} I_{in}$.

Module Operation:

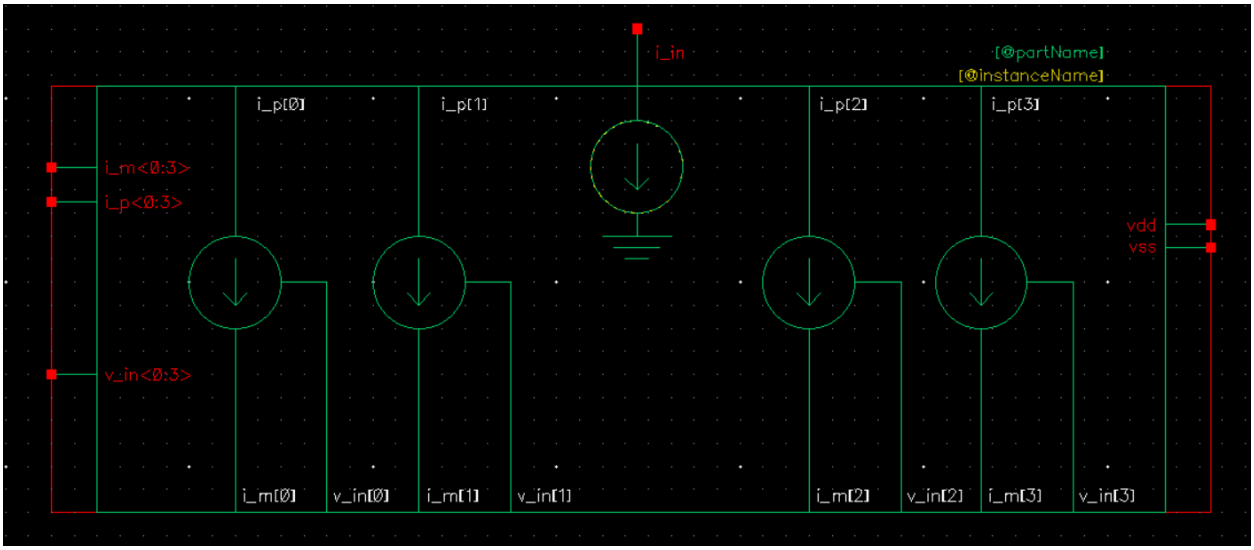
The Verilog-A Module' purpose is to compute the current it will output through $\boldsymbol{i\_m[0:3]}$ using the value of the input weights $\boldsymbol{v\_in[0:3]}$ and the generated photo-current in the WBCAPD photo sensing device $\boldsymbol{i\_in}$ using the equation $i\_p\,[j] = I_j = \frac{V_j}{V_j+V_k+V_l+V_m} \cdot i\_in$.

The "positive" terminals $\boldsymbol{i\_p[0:3]}$ are drawing current from the source (V_{DD}), while the "negative" terminals $\boldsymbol{i\_m[0:3]}$ are supplying the current from the device as an output.

For equal weights, the photo-current collected **i_in**, will be divided symmetrically (i.e., $\boldsymbol{i\_m}$**[0:3]=** $\frac{i\_in}{4}$).

On the other hand, if only **node j** is biased with positive weight and the rest of the weights are significantly smaller (i.e., $v\_in[j] \gg v\_in[k], v\_in[l], v\_in[m]$) , then this node will draw all the photo-current: $\boldsymbol{i_{m[j]} \approx i\_in}$, $\boldsymbol{i\_m[k] = i\_m[l] = i\_m[m] \approx 0}$.

In essence, the device acts as four dependent current sources.

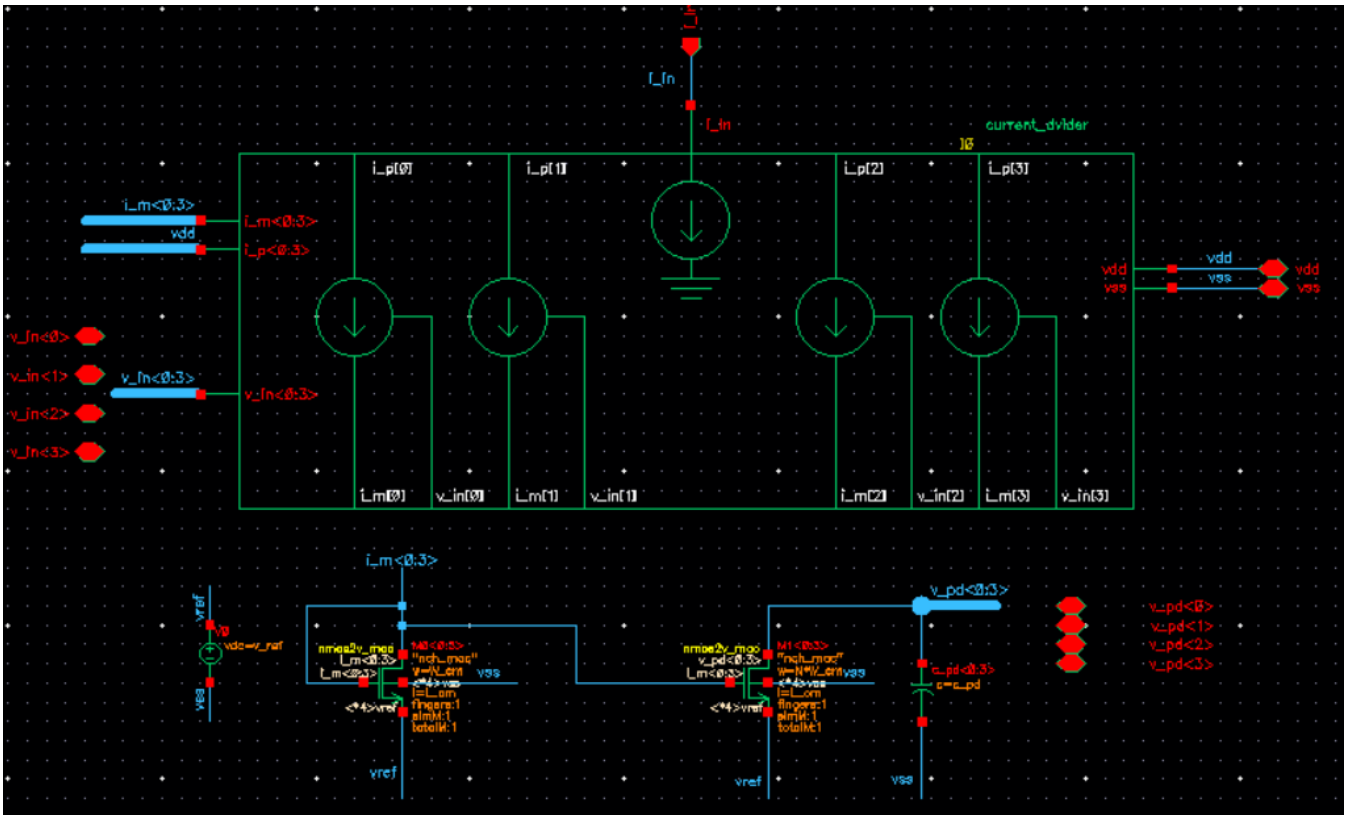


4.2 WBCAPD Complete Module

To emulate a PN Junction photodiode, a current mirror circuit connected to the Verilog-A Module's output ($i\_m[j]$) and to a capacitor **c_pd** (simulating the depletion region capacitance in a reverse biased PN junction). In addition, both transistors of the current mirror circuit are biased with a **vref** voltage to emulate the voltage of a photodiode at saturation.

The combined module and circuit emulates a PN junction photodiode, whom voltage is limited by the reset voltage from above and **vref** voltage from below. The current mirror circuit is used as an approximated current source. The M0 Transistor's gate and drain are shortened to force the transistor into saturation operation mode, making the current flowing through M1 transistor directly related to the current flowing through M0 transistor, thus discharging the **c_pd** capacitor similarly to a generated photo-current discharging a reverse biased PN junction's depletion region.

To choose the capacitance **c_pd**, various diodes in TSMC18 library were compared in the capacitance per area criteria (section 6.1.1).

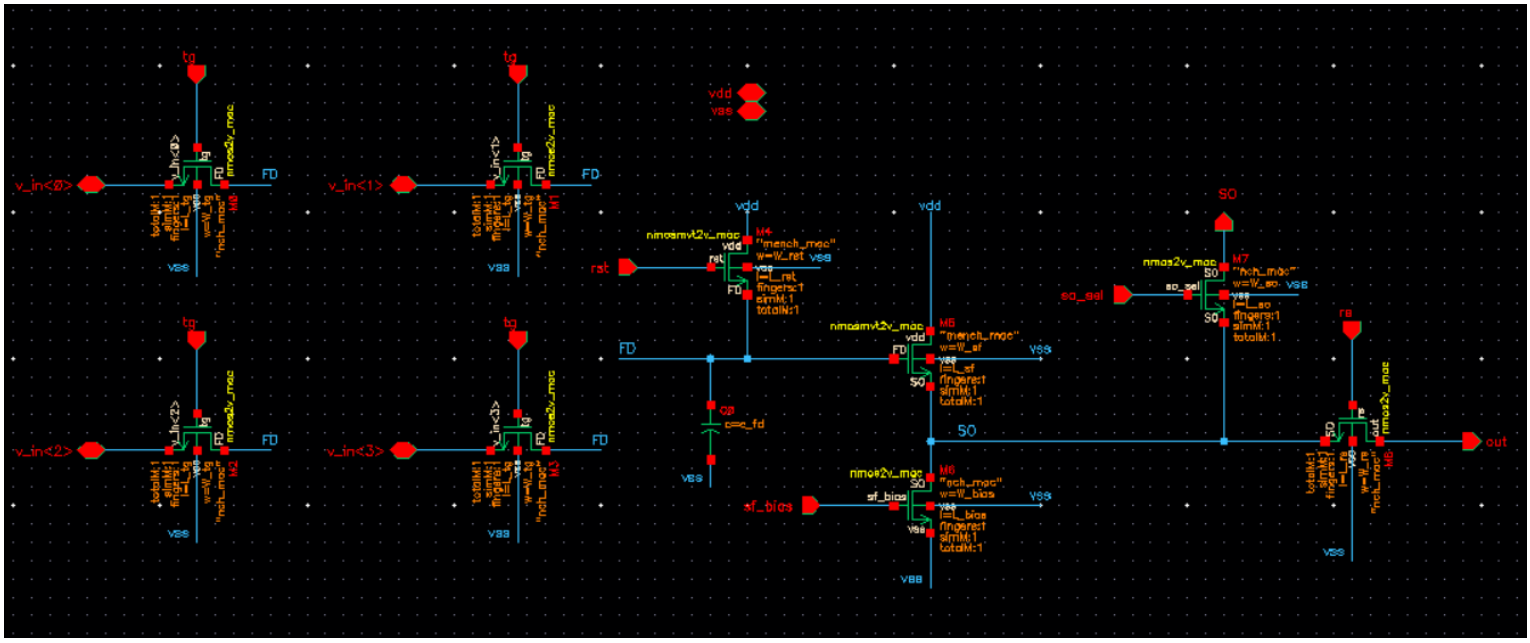


We will relate to **c_pd** capacitor as "the photodiode" from now on.

4.3 Pixel Model

4.3.1 Pixel Model Design

For this work, a common 4T APS was modified to support the WBCAPD array. For the chosen connectivity configuration of the WBCAPD array each pixel has four input nodes connected to four “Transfer Gate” (‘tg’) transistors, sharing a common control signal. A “Floating Diffusion” (‘FD’) capacitance is added to store the analog value of the reset voltage and prevent leakage currents during sampling. To support feedback operation mode, an addition “Source Follower Bias” (‘sf_bias’) and “Secondary Output Select” (‘so_sel’) transistors were added. Thus, said pixel structure will support two operation modes: image sensing and real time feedback. The following design was chosen:



Low V_{th} transistors were selected for the Reset and Source Follower transistors to reduce voltage drop.

Thus, the photodiode voltage is limited to $\boldsymbol{V_{DD} - V_{th(rst)}}$ from above and by **vref** from below.

Similarly, the Output is limited $\boldsymbol{V_{DD} - V_{th(rst)} - V_{th(sf)}}$ from above and by $\boldsymbol{vref - V_{th(sf)}}$.

Low V_{th} transistors should be used to minimize voltage drop.

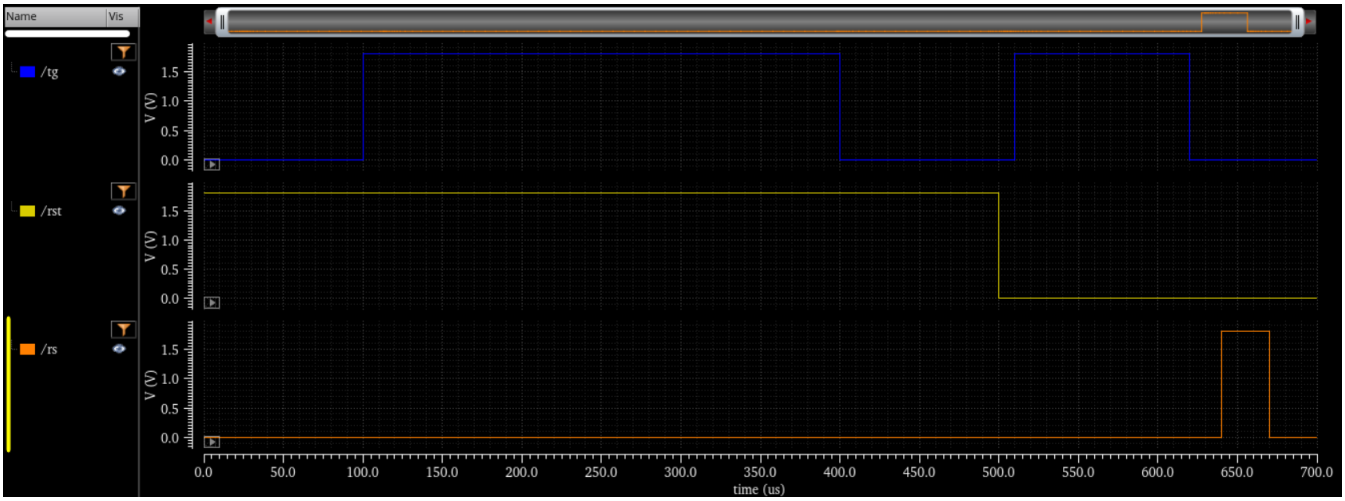
4.3.2 Image Sensing Mode - Pixel Operation

During the image sensing operation mode, the pixel operates in charge mode:

A. **rst** transistor is opened to begin Reset stage and charge the Floating Diffusion capacitance.
B. **tg** transistor is opened to reset the photodiode to a value of $V_{DD} - V_{th(rst)}$.
C. After said pre-charge, the **tg** transistor closes and the photodiode begins integration. During this time period photo-generated current discharges the photodiode.
D. **rst** transistor is closed. In the time period between stage c and this stage, the reset voltage can be sampled since it's stored in the floating diffusion.
E. **tg** transistor is opened to perform sample & hold. The stored charge passes from the photodiode to the floating diffusion.
F. **tg** transistor is closed. (The integrated photodiode value is stored at the floating diffusion).
G. Pre-readout delay (20 µs).
H. **rs** transistor is opened to perform row readout.
I. **rs** transistor is closed.

During the image sensing operation mode, the Source Follower Bias and Secondary Output Select transistor are cutoff.

4.3.3 Image Sensing Mode - Pixel Control Signal Waveform



During the Image Sensing operation mode Source Follower Bias and Secondary Output Select signals are set to V_{ss}.

Note: **Floating Diffusion Reset** and **Photodiode Reset** times can be greatly reduced.

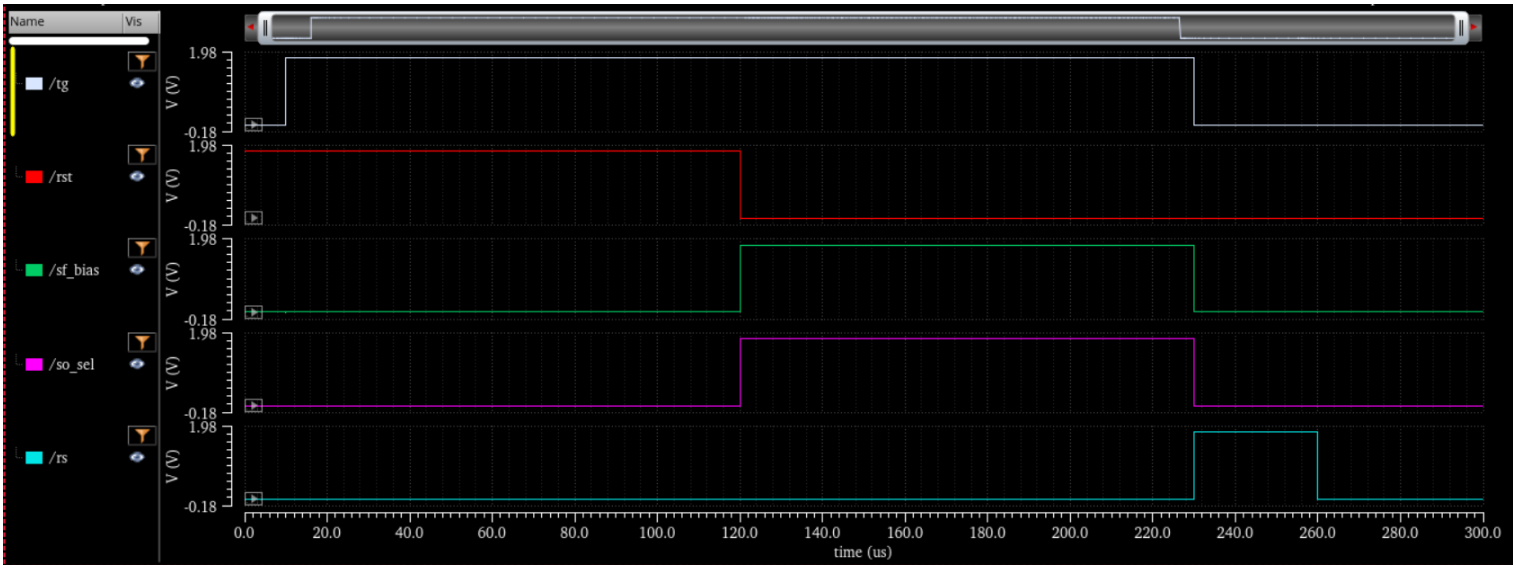
4.3.4 Feedback Mode - Pixel Operation

During the feedback operation mode, the pixel's output is the real-time integrated value as a function of its inputs. The Pixel operates in the following manner:

A. **rst** transistor is opened to reset the Floating Diffusion capacitance to reset value. During this time the reset value can be read out for additional implementations.
B. **tg** transistor is opened to perform reset of the photodiode to $V_{DD} - V_{th(rst)}$.
C. **rst** transistor is closed and **sf_bias** ,**so_sel** transistors are opened and allow the real time integrated pixel value to be outputted via **SO** pin.
D. **Tg, sf_bias, so_sel** transistors are closed, while **rs** transistor is openes to perform readout through the **rs** transistor.
E. **rs** transistor is closed.

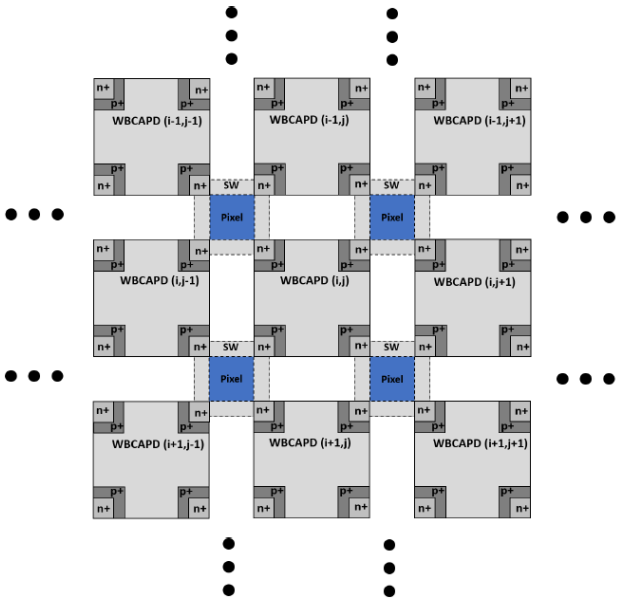
In this operation mode, the pixel's real time integrated value (Secondary Output) is continuously changing as a function of its input.

4.3.5 Feedback Mode - Pixel Control Signal Waveform



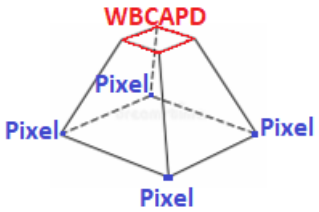
5. WBCAPD Array Design

In this work we propose the following connectivity: connecting each collection junction of the WBCAPD module to neighboring pixel inputs (Storage Wells). Thus, a wide spatial connectivity is achieved. By connecting the Secondary output of a pixel to neighboring WBCAPD's modulation junction a simple feedback loop is created: $Pixel \rightarrow Weight \rightarrow Photodiode \rightarrow Pixel$.
A visual representation of the array connectivity can be seen below:



Proposed array connectivity capable edge detection implementation

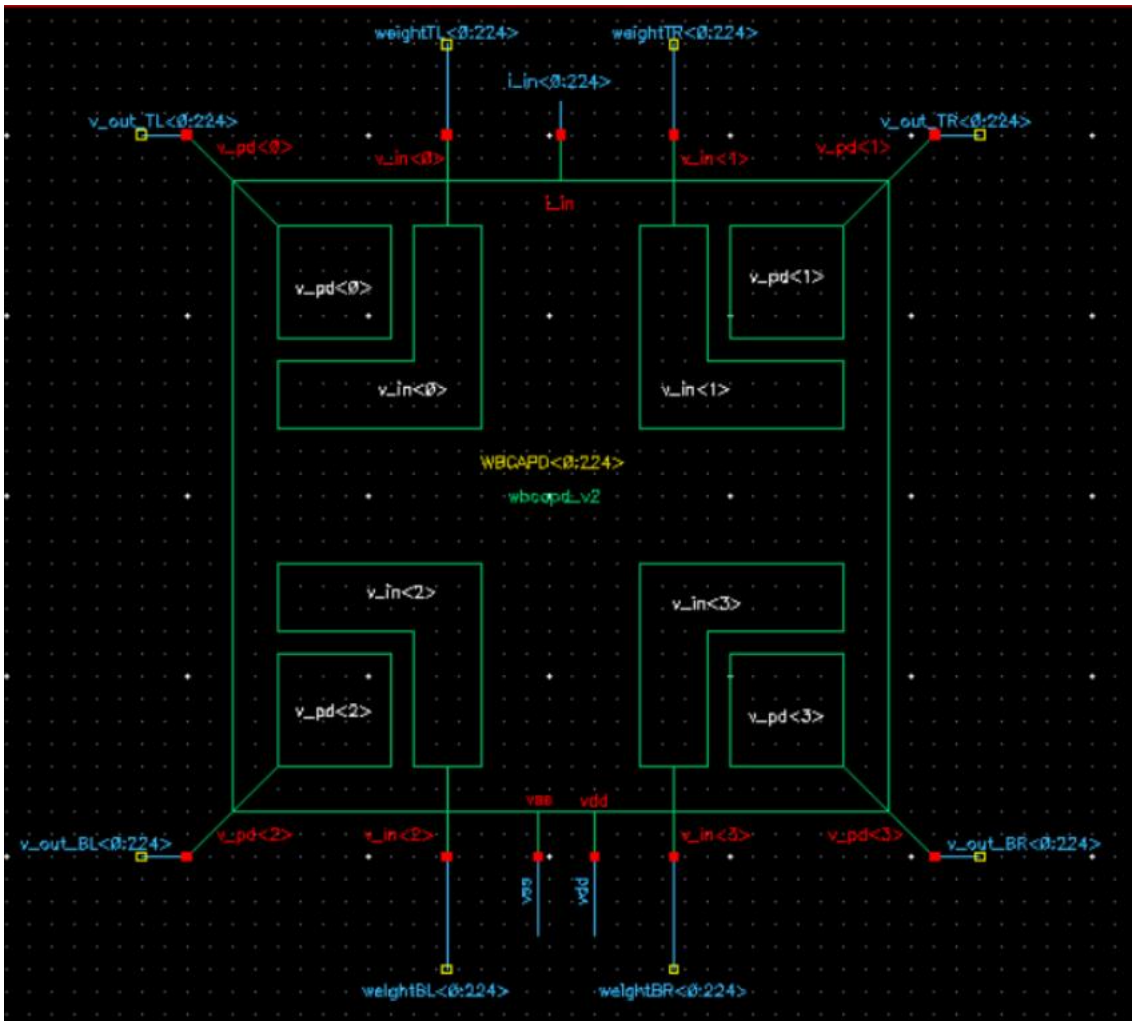
The connectivity seen in fi. 3.1 is a 2-layer truncated rectangular pyramid; Layer 1 are the Pixels and Layer 2 are the WBCAPDs.
The number of pixels elements is $(m \times n)$.
The number of WBCAPD elements is $(m-1) \times (n-1)$.
This work will focus on square arrays $(m = n)$.



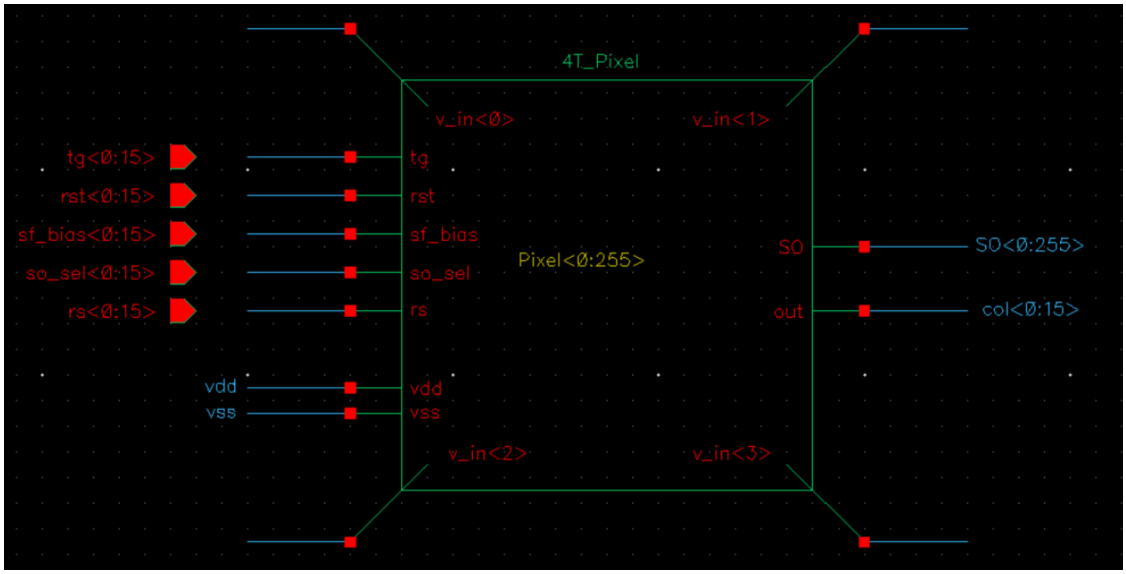
One unit of WBCAPD surrounded by four pixels. This cell will construct the WBCAPD array.

Implementation of such spatial connectivity for a (16×16) Pixel Array and (15×15) WBCAPD Array was achieved using Cadence Virtuoso environment.

WBCAPD Array instances:

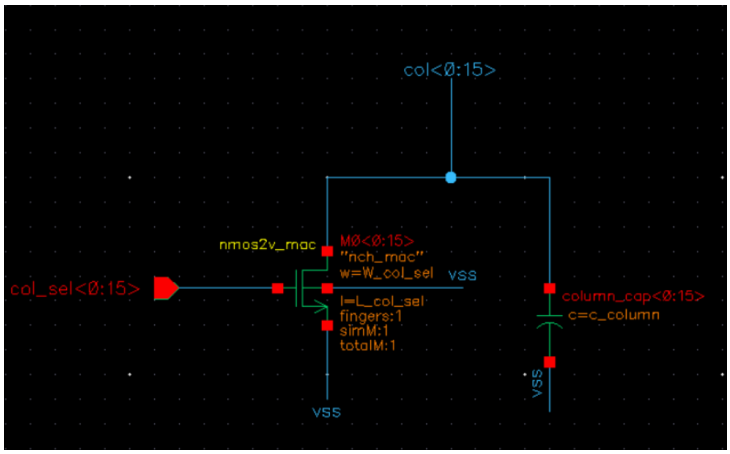


225 Instances of WBCAPD module are simulated. Each WBCAPD has four inputs which can be biased with an external signal or an internal signal (feedback), and four outputs (collection junctions) which connect to the inputs of four surrounding Pixels.



Pixel Array instances

256 instances of the modified 4T Pixel. Each of the Pixel's inputs connects to one of four neighboring WBCAPD modules, and its output is connected to a column readout circuit. To perform feedback operation, the Secondary Outputs of the Pixels are connected to the modulation junction of each of the 4 surrounding WBCAPD modules. The array is connected to control signals which are common for each row.



Column Readout circuit

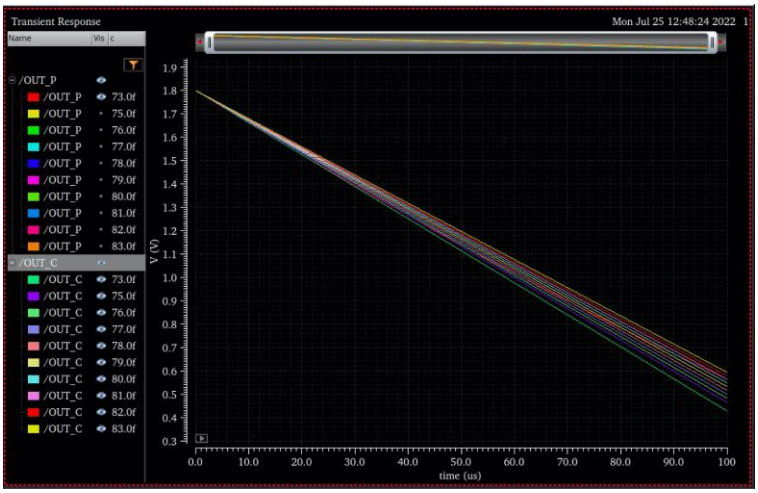
To perform readout, each array column is connected to a column bias transistor ('**col_sel**') parallel to a capacitor for voltage sampling. During the readout stage, **rs** transistor passes the voltage value down the column to the column circuit. To sample the column value **col_sel** transistors opens for a period of time.

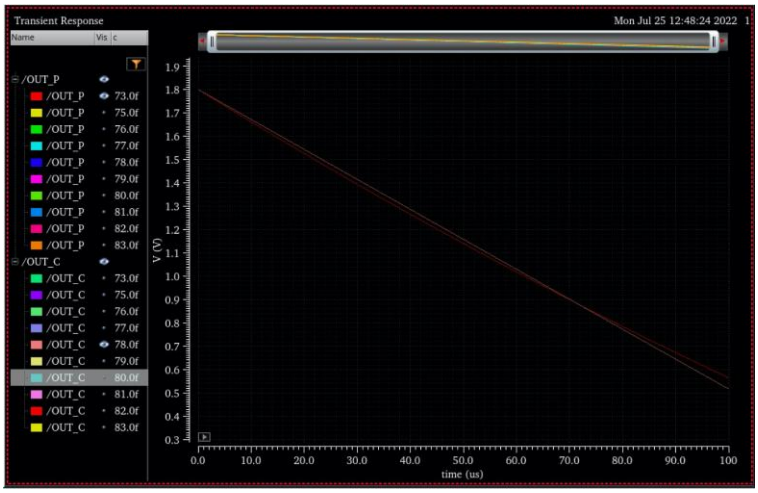
6. Simulations

6.1 Photodiode simulations

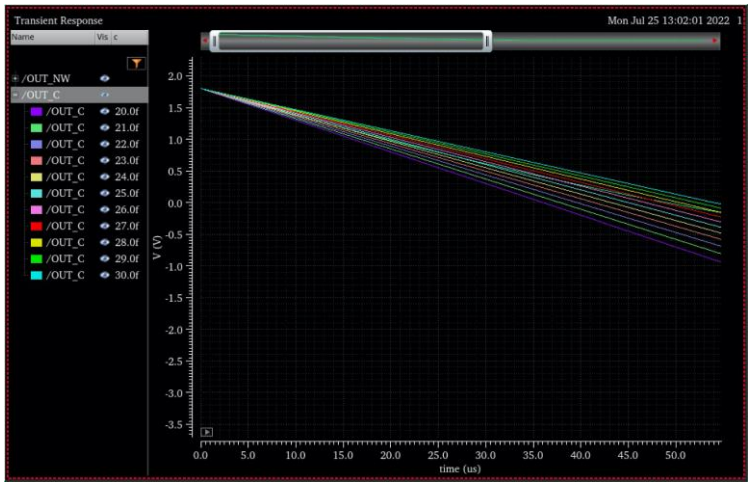
TSMC18 Photodiode comparison simulation

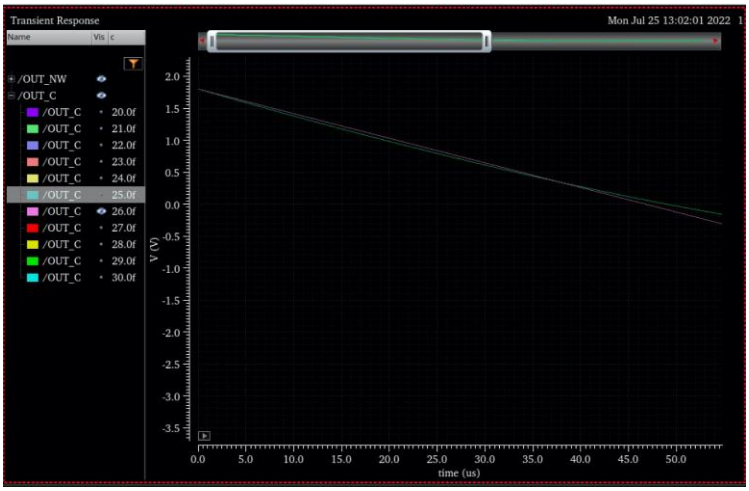
A test bench circuit (see Appendix 7.2.1) was created to test the capacity density of the diodes, available in TSMC18 (180nm) technology. Three diode cells were selected: “PDiode”, “NWDiode” and “NDiode” and connected in reverse bias. The selected diodes area is chosen to be $10\mu m \times 10\mu m$. The capacitor and the diodes were set to an initial condition of $V_{DD} = 1.8\ [V]$. Using a current source and the reference capacitor, the voltage curves of the diodes were matched to a certain capacitance value:



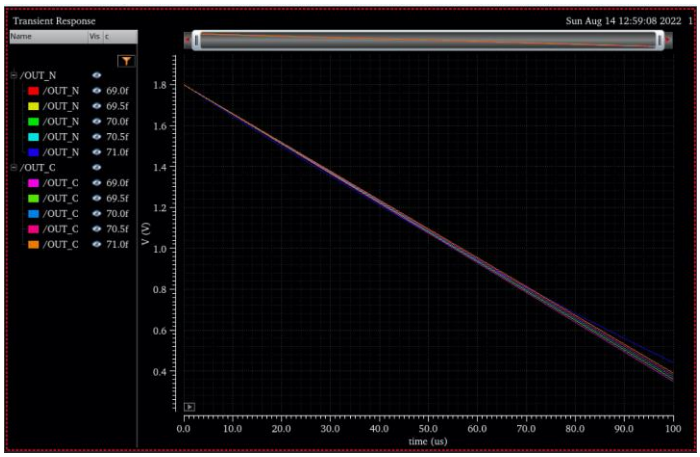


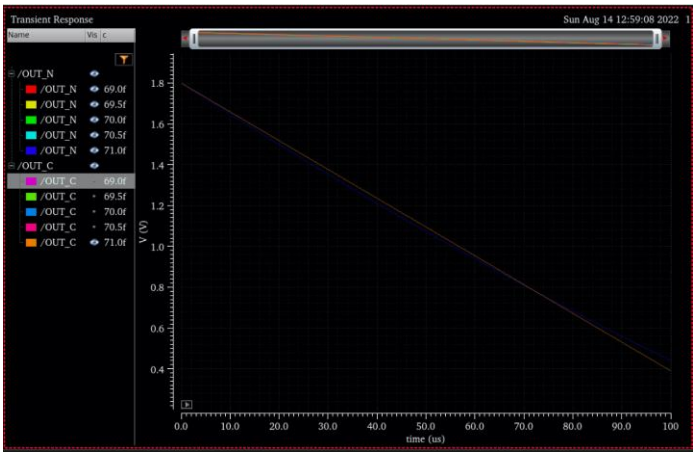
“PDiode” capacitance matching: $C_{PDiode} = 78\ [fF]$





"NWDiode" Capacitance matching: $C_{NWDiode} = 26\ [fF]$





"NDiode" Capacitance matching. $C_{NDiode} = 71\ [fF]$

The calculated capacitance density of the diodes:

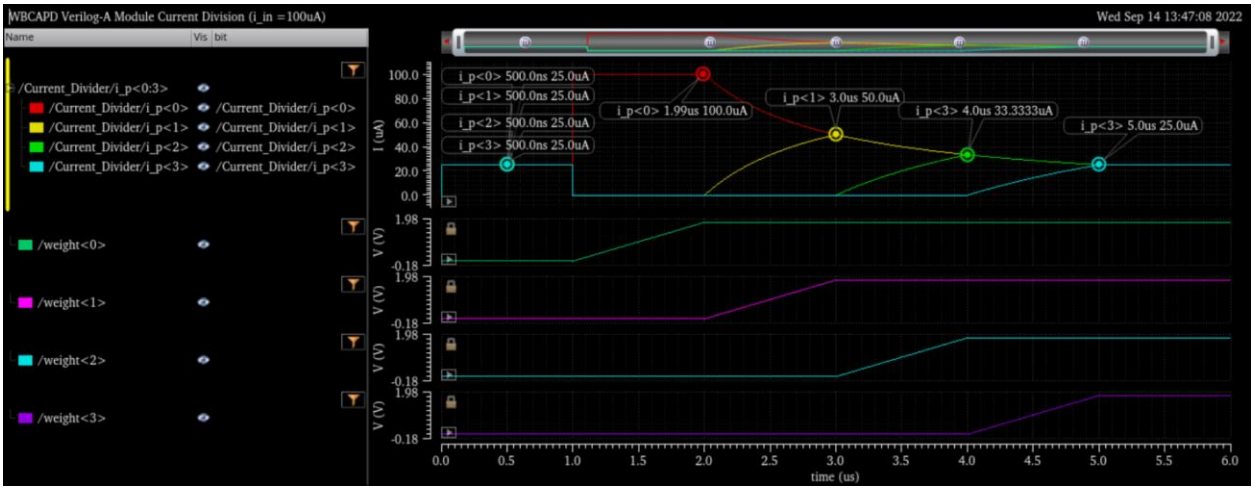
Diode	Area: $10\mu m \times 10\mu m$	Area: $1\mu m \times 1\mu m$
PDiode	78 [fF]	0.78 [fF]
NWDiode	26 [fF]	0.26 [fF]
NDiode	71 [fF]	0.71 [fF]

An "NDiode" capacitance density of $0.71[fF]$ will be used. The size of the chosen diode will be $6.28\mu m \times 6.28\mu m$ resulting in a capacitance to be used: $c_{pd} = 28.08\ [fF]$.

6.2 WBCAPD Module Simulations

6.2.1 Verilog A Current division (ramp inputs)

To demonstrate the current division capabilities of the designed WBCAPD Module. The weights are ramped up from V_{SS} to V_{DD} with a delay of $1\mu s$ between consecutive weights. For simplicity, a current of $100\mu A$ was used.

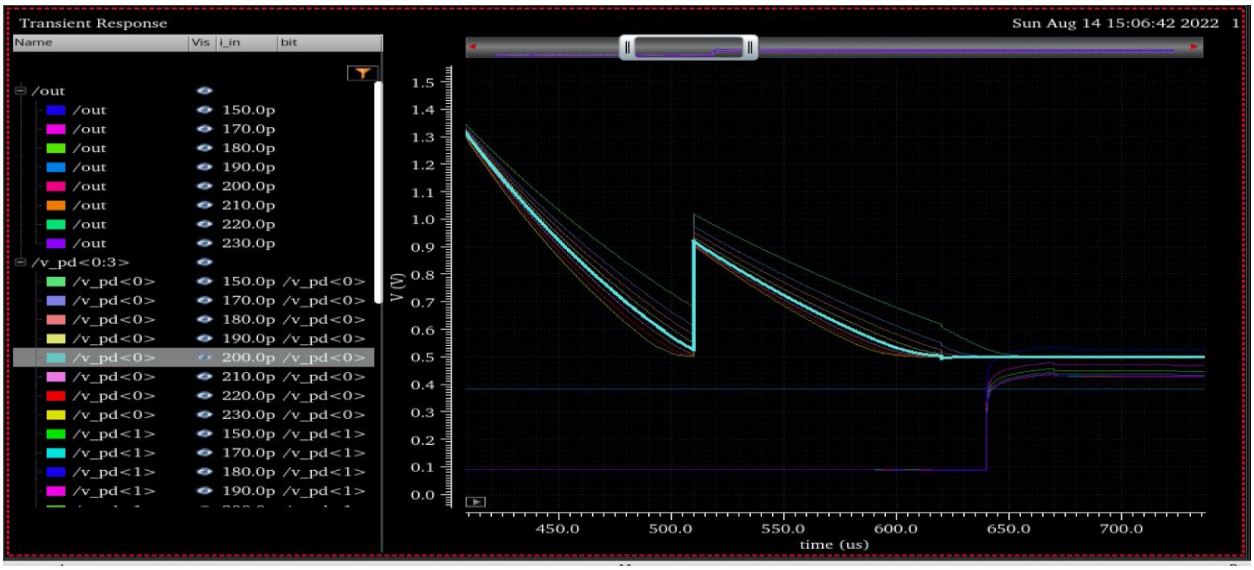


The simulation results demonstrate that the WBCAPD's current division Verilog-A model is capable of ideal (mathematically) current division. The limitation of this this model is that it does not consider leakage currents.

Test setup and parameters are shown in Appendix (7.1).

6.2.2 Determination of maximal photo-current

To determine the maximum allowable photo-current into the device, without preemptively (before integration time ends) discharging the capacitance and potentially losing information of the scene captured by the photodiode. For a chosen integration time of $110\mu s$ the allowable maximal photo-current is $I_{max} = 200\ [pA]$.



Voltage Curve of $28.08\ [fF]$ capacitor vs Time $[\mu s]$ - comparison over a range of input photo-currents

The chosen maximal current allows for:

A. The photodiode capacitance won't discharge before the integration time end, so no information will be lost.
B. The chosen capacitance discharge profile is similar to one of an actual photodiodes.
C. At maximum current, $I_{photo-current} = I_{max} = 200\ [pA]$, the capacitance is discharged to its lowest voltage $\boldsymbol{vref = 500\ [mV]}$.

For simulation simplicity, a zero photo-current was chosen to be $I_{min} = 1\ [aA] = 10^{-18}[A]$ instead of numerical zero.

6.3 Single Pixel Level Simulations

To put the operation of the designed pixel to the test, a test bench circuit was created (Appendix 7.3). A pixel's single input was connected to a single WBCAPD's collection junction. This configuration enabled the integration of the Pixel and WBCAPD for later simulations.

During this simulation all WBCAPD weights were set to V_{SS} except the modulation junction of the collection junction connected to the pixel, which was set to V_{DD}, so that the node connected to the pixel will draw 100% of the current.

6.3.1 Image Sensing simulation (determination of output headroom)

Using an operation mode described in 4.3.2, the output of the pixel was sampled twice: once in the case of high light intensity using $I_{max} = 200[pA]$, and once in the case of a dark scene $I_{min} = 1[aA]$.

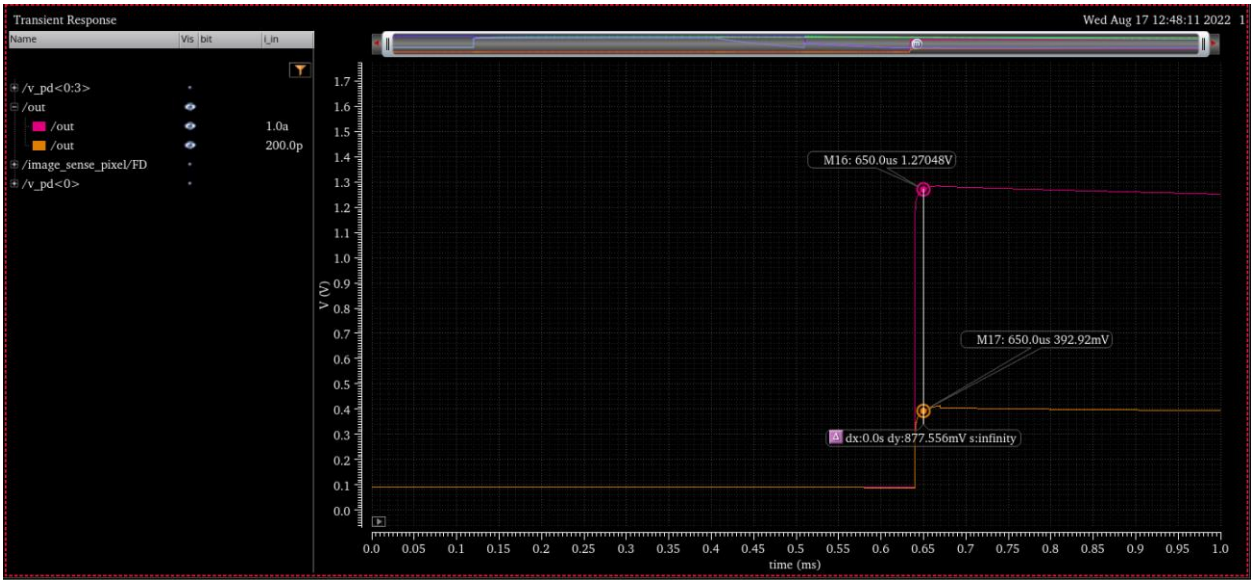
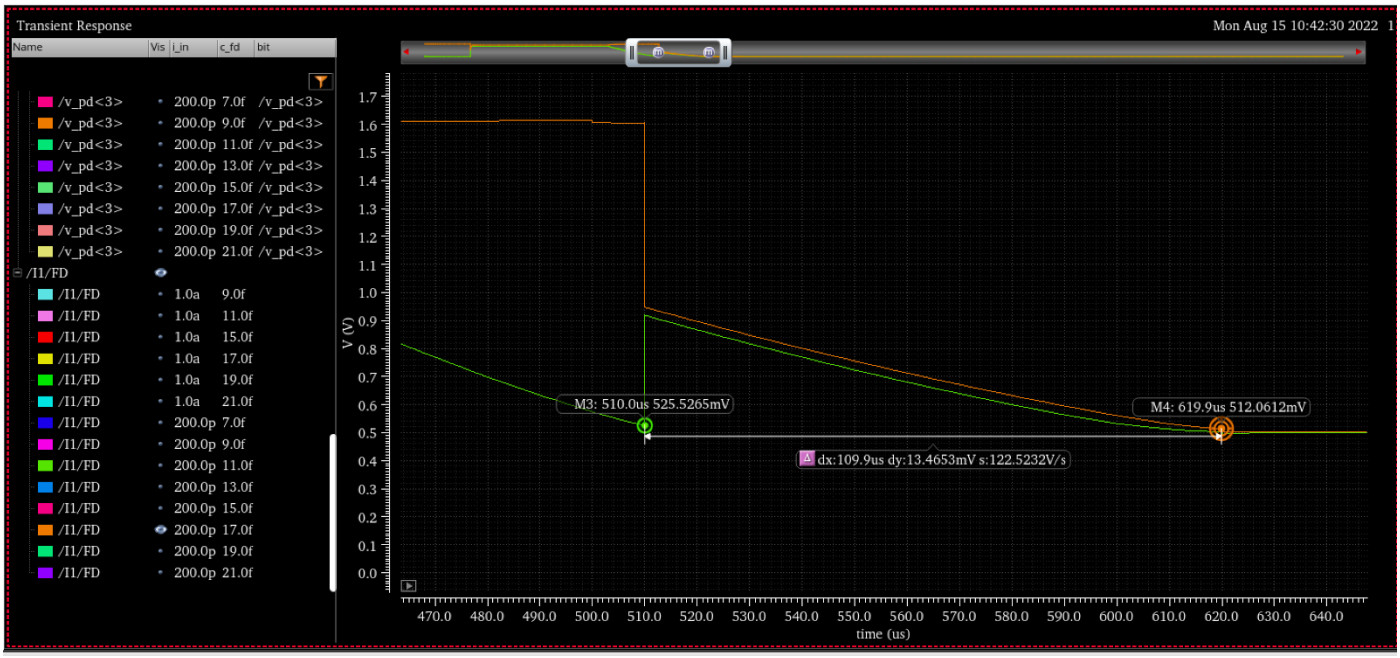


Image Sensing operation – sampling output

The resulted headroom is $877.56\ [mV]$. The floating diffusion capacitance and column capacitance were accounted for. See following simulations **6.3.2** and **6.3.3**.

6.3.2 Floating Diffusion Capacitance

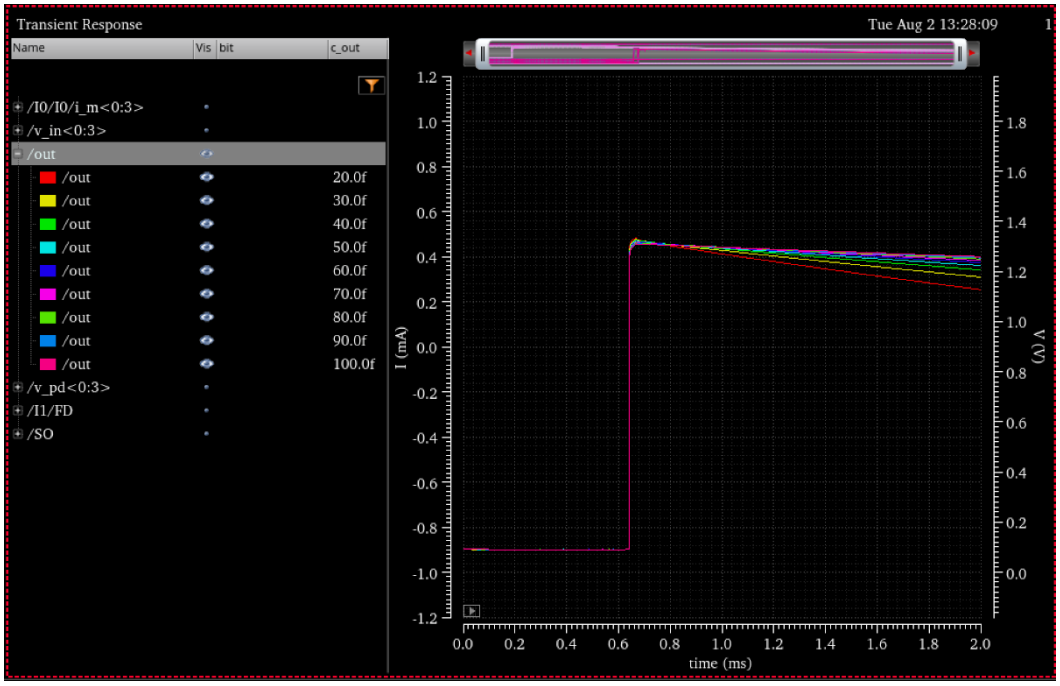
To assess the loss of charge during readout, a set of c_fd capacitance values was swept over.



The main parameter for choosing a capacitance value was the accuracy of the sample, which can be denoted by $\frac{V_{PD}@sample_{start} - V_{FD}@sample_{end}}{V_{PD}@sample_{start}} \times 100\%$. For the chosen capacitance $C_{fd} = 17\ [fF]$, the error is 2.56%.

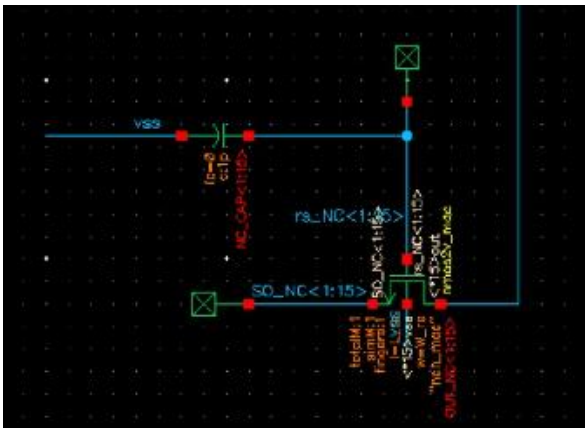
6.3.3 Column Capacitance Simulations

Since they primitive cells of the WBCAPD and pixel will be integrated into a 16×16 pixel array, a capacitance should be accounted at the column level, simulating the capacitance each pixel output "sees" at column level, after the Row select transistor. For this task, the same test bench mentioned in 6.3 was used, with the exception of the output connected in parallel to additional 15 Row Select transistors.



A column capacitance of $c_{column} = 40\ [fF]$ was chosen for its ability to maintain the charge over time without loosing much of it's stored value and yet saving space on the chip.

The following circuitry simulates the output of a single pixel connected in parallel to another fifteen Row Select NMOS transistors:



6.4 Array Simulations

The models were simulated using Cadence Virtuoso environment. Current sources were implemented to simulate the individual pixels of the frame we were testing. (i.e., $I_{photo-current} \equiv i\_in$).

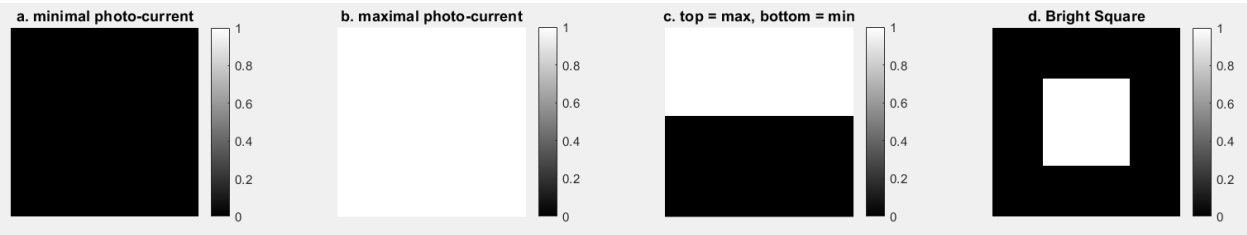
In order to perform the simulations, using a supportive MATLAB code, external variable and control files were generated and imported for the simulations. Respectively, outputted results from the simulations were exported and later processes by MATLAB.

To test the ability of the array to perform image sensing operations, current variables were imported using a supportive MATLAB code (Appendix 8.1) .Photo-current variables (i_in<0:224>) represent the input - a (15×15) frame.

To control the pixels control signals, a supportive MATLAB code was used (Appendix 8.2 and 8.3), achieving pixel control using a VEC file.

The output value is extracted using Virtuoso Calculator expression that samples the voltage value at the end of column read operation, resulting in a (16×16) output. Using the supportive MATLAB code, the analog voltage readout results were converted to an image and used to determine edges by subtracting the spatially averaged readout values from the image sensing readout values (Appendix 8.5).

Four inputs were used in the following simulations:

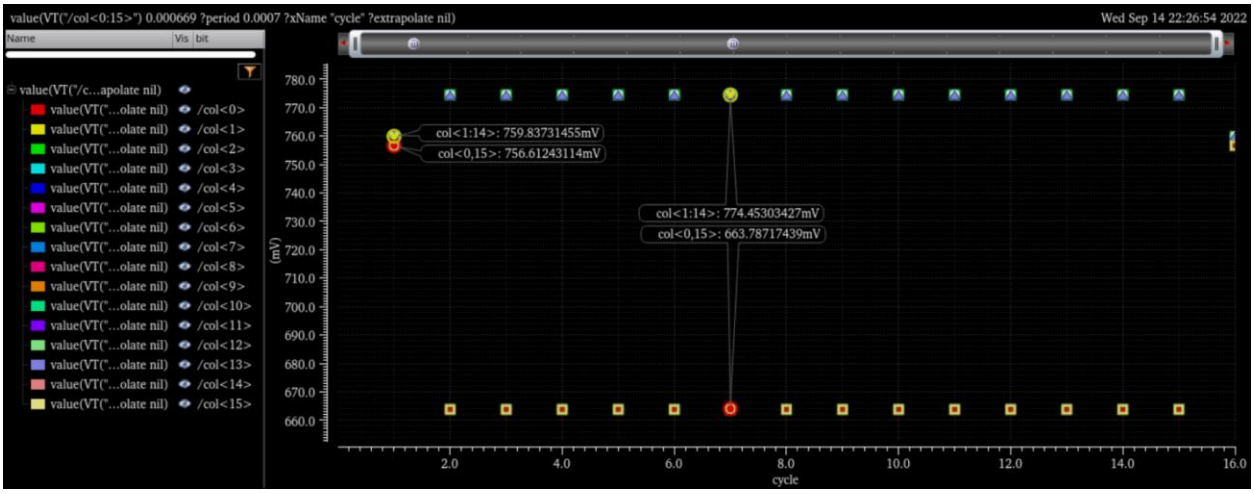


A. Dark image $-i\_in < 0{:}\,224 >= minimal\ photo - current = 1[aA]$
B. Bright image $-i\_in < 0{:}\,224 >= maximal\ photo - current = 200[pA]$
C. Two halves: Top half is bright (200pA), bottom half is dark (1aA).
D. Bright Square (200pA) in dark scene (1aA).

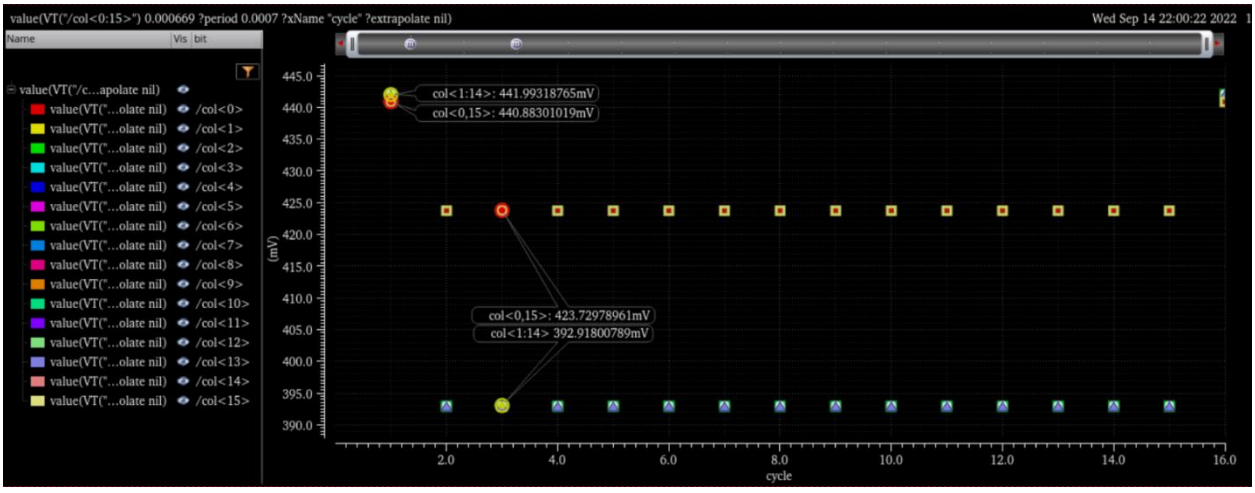
Since anomalies at the perimeter of the output array were observed, they will be neglected in the following discussions of the simulated results. This matter will be addressed later in the end of this section.

6.4.1 Image Sensing Operation Simulation

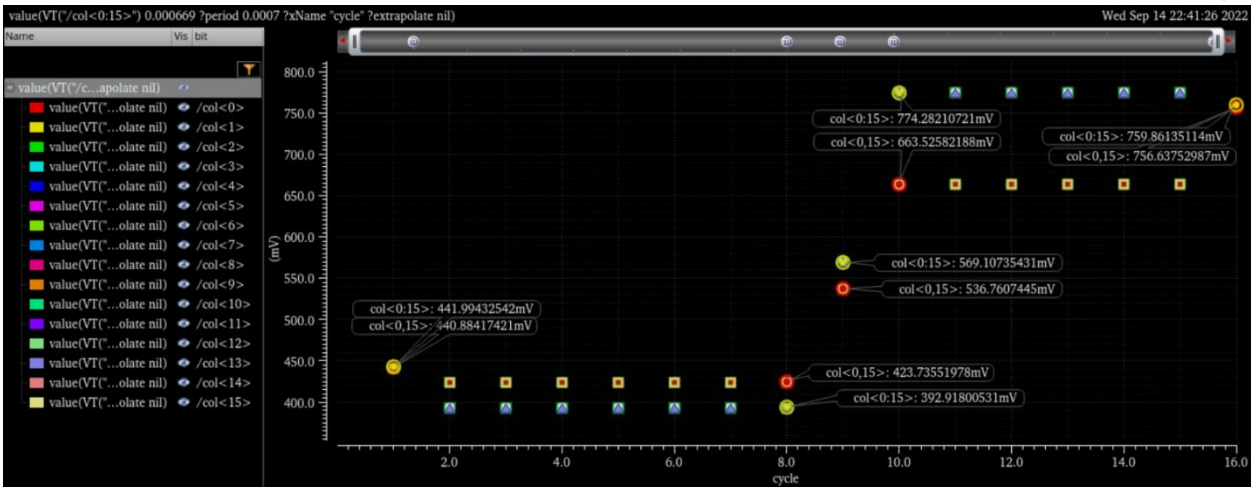
(All weights are equally biased by V_{DD})



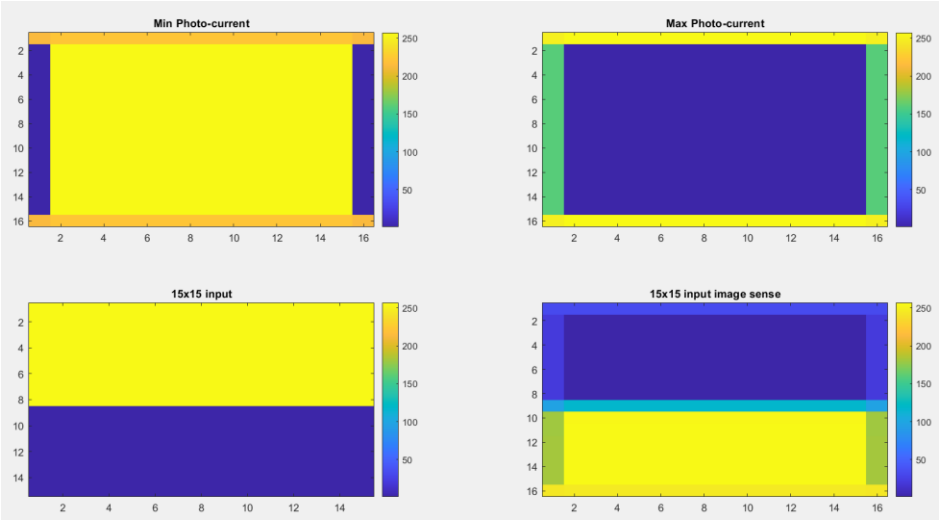
A. When the simulated image sensing array is introduced minimal current input (1aA), simulating minimal illumination of the array, the WBCAPD module divides the current equally between its collection junctions. Since the inputted current is minimal the photodiodes barely discharge, resulting in a “high output”.
The maximal voltage output is $V_{max} = 774.45\ [mV]$.



B. When the simulated image sensing array is introduced maximal current input (200pA) , simulating maximal illumination of the array, the WBCAPD module divides the current equally between its collection junctions. Since the inputted current is maximal the photodiodes discharge, resulting in a “low output”.
The minimal voltage output is $V_{min} = 423.73\ [mV]$.

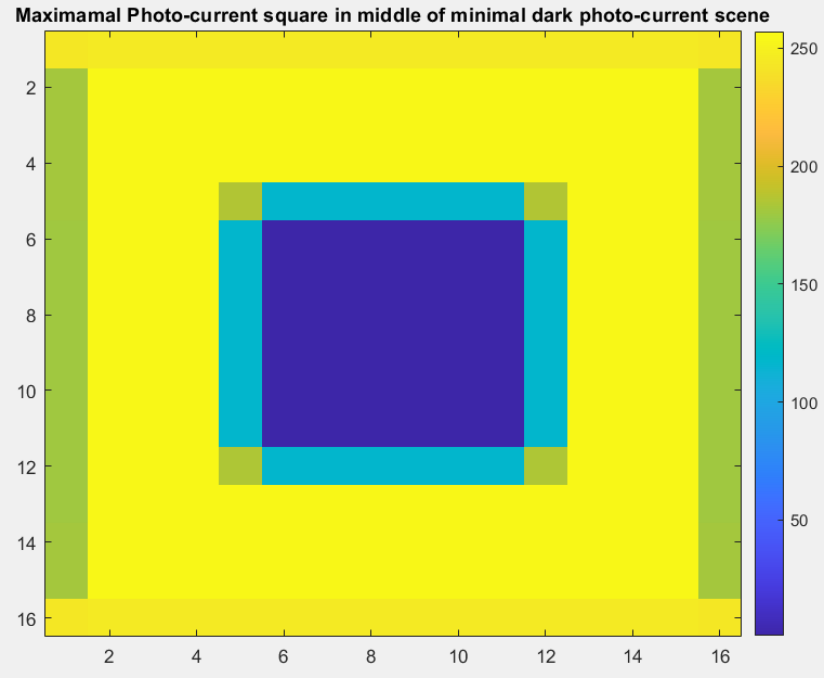


C. When the simulated image sensing array is introduced to a non-uniform input array, the array performs a slight spatial averaging. “cycle” represents the number of row being sampled.
The “top half” of the inputted image was maximal photo-current, resulting in a “low” output (as seen in cycles 1-8). The “bottom half” of the inputted image was minimal photo-current, resulting in a “high” output (as seen in cycles 10-16).
At the interface (row 9, represented by cycle 9) between the maximal and minimal photocurrent input, slight spatial averaging occurs. The observed output voltage is the average of the photo-current resulted output of said interface WBCAPDs’.



Visual representation of the resulted image sensing outputs (a-c) using MATLAB (Appendix 8.5).

The conclusion we draw from these simulations is that the WBCAPD image sensing array can perform image sensing indeed. The resulted output will be a slight averaged of the scene.

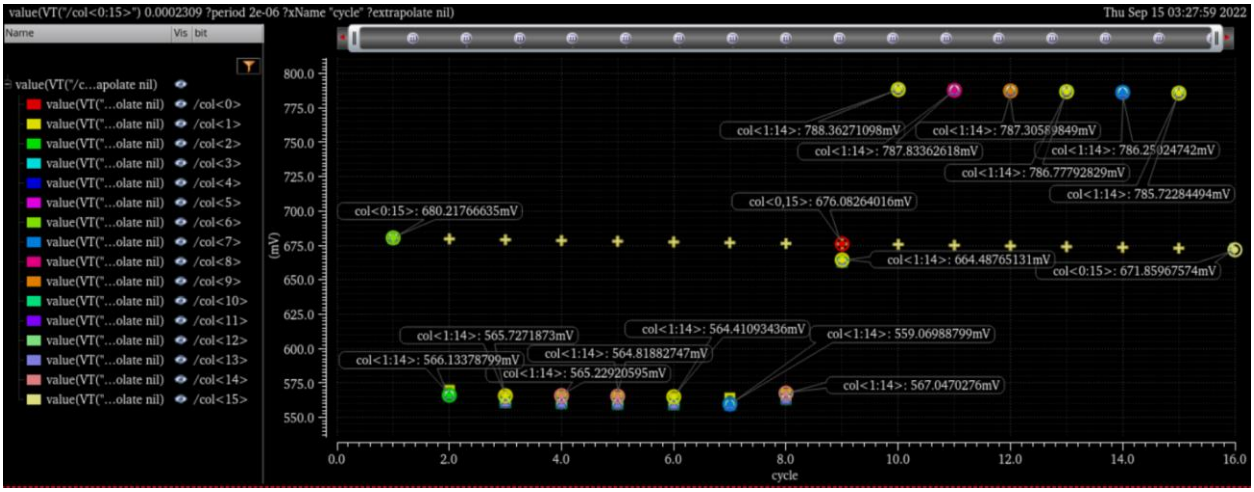


Visual representation of the resulted image sensing output (D) using MATLAB (Appendix 8.5).

D. To observe a better effect of the slight spatial averaging at two-dimensional space, input D was introduced. A conclusion can be drawn that the resulting output is a slight spatial averaging of the scene, limited to the interface of the different “brightness” (current values that differ highly).

6.4.2 **Feedback Operation Array Simulations**

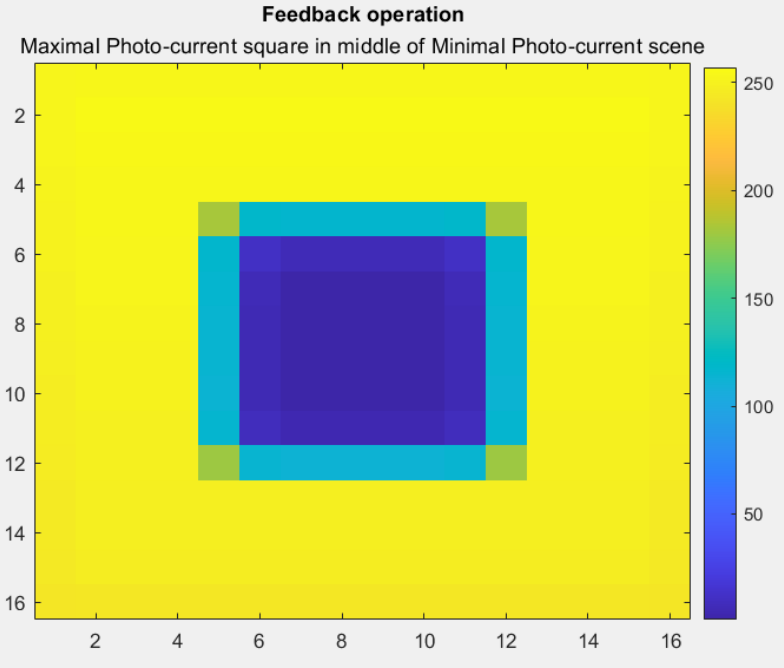
In the following simulations, all WBCAPDs' weights are connected to their neighboring pixels' secondary outputs, dynamically changing over time .The **feedback time** is equal to $110[\mu s]$ similar to the integration time of the image sensing operation.



Voltage vs Row ("cycle") calculated output at column sampling

Visual representation of the resulted feedback operation output (for input C) using MATLAB (Appendix 8.5).

Image (C) from section 4 was inputted. The resulting image is a spatial averaging of the scene. The interface of the low and high output is the direct result of the dynamic averaging the surrounding WBCAPDs' collection junctions; half of which were at high voltage (resulted from not discharging the photodiode at low photo-current input) and half which were at low volage (resulted from discharge of the photodiode at high photo-current input). Additionally, spatial averaging occurs not only at the interface, but deeper into the array. Eventually, each pixel in the array, "senses" all other pixels.

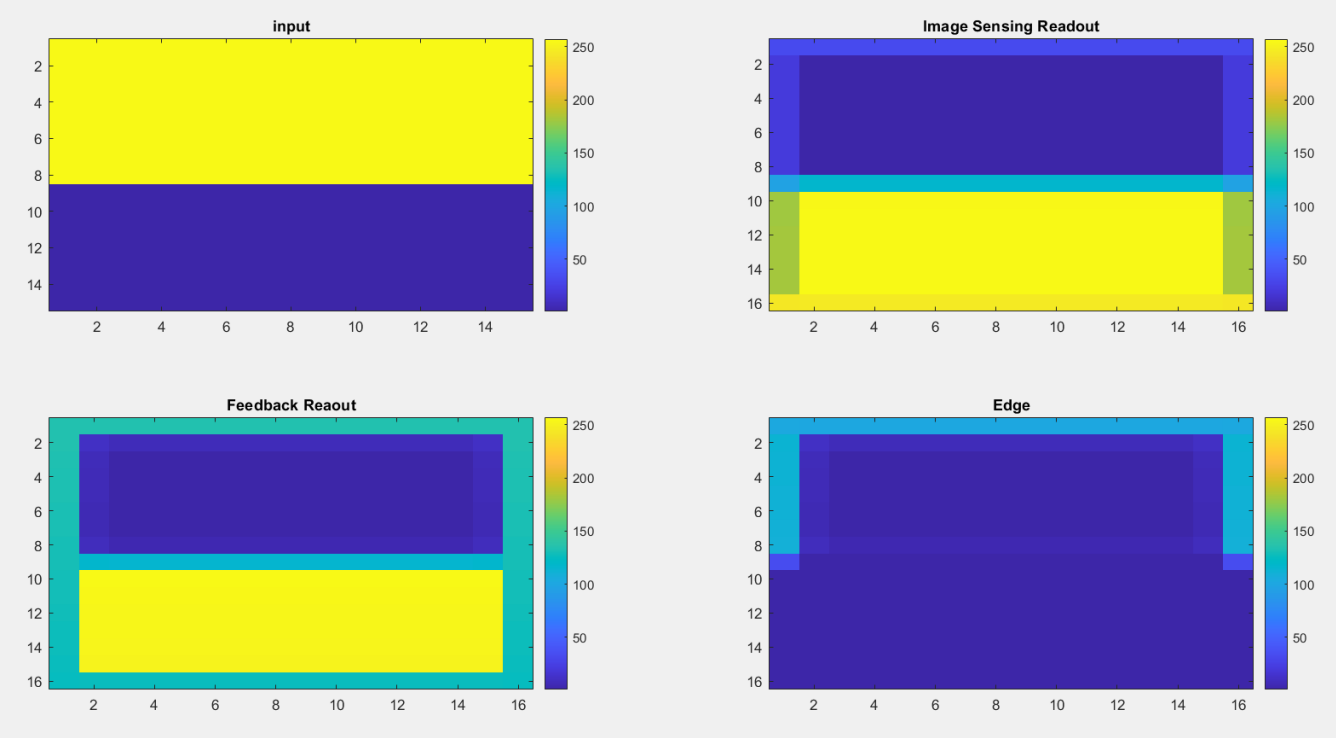


Visual representation of the resulted feedback operation output (for input D) using MATLAB (Appendix 8.5).

To observe a better representation of the spatial averaging throughout the array, image (D) was inputted. The resulting image is a spatial averaging of the scene. It can be observed that the value of captured intensity is changing at the depth of the array and not only restricted to the interface defined by high change in intensity of WBCAPD's illumination.

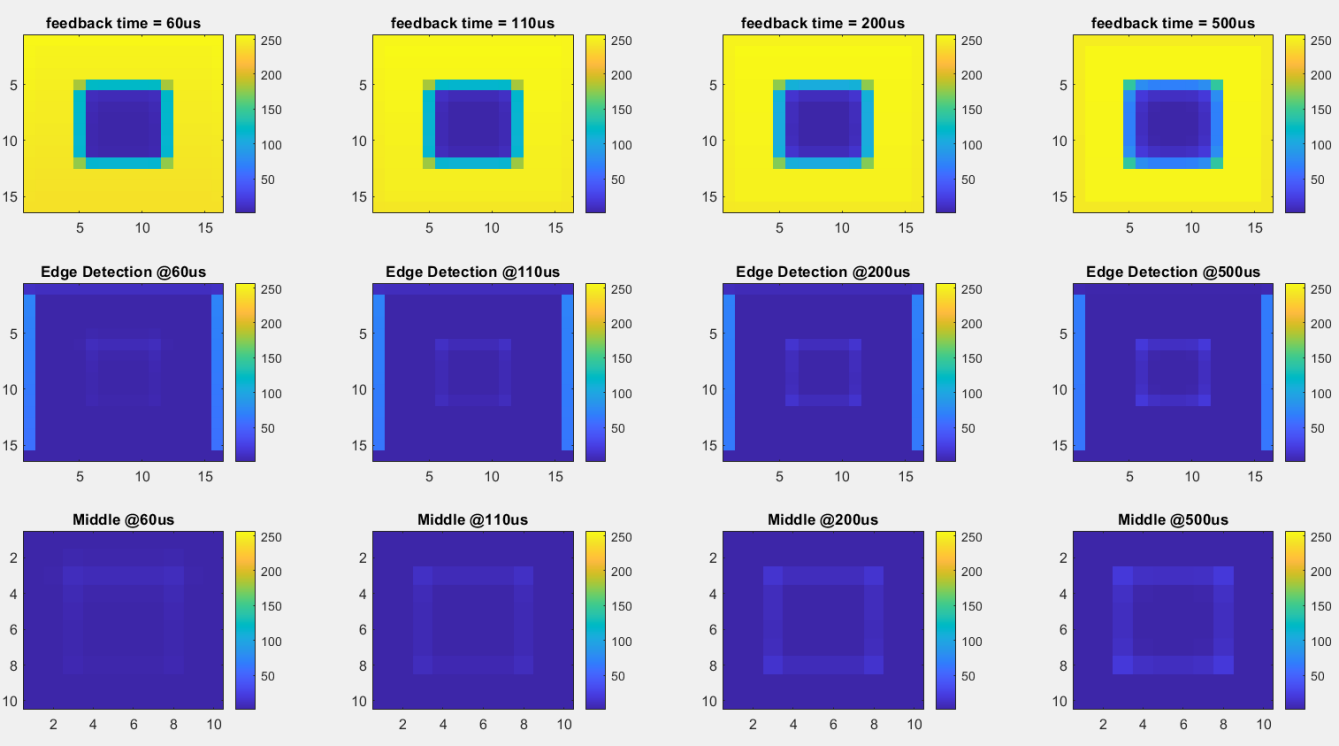
6.5 Implementation of Edge Detection

Using the algorithm mentioned in 2.2.1 (subtraction of the captured scene illumination values from the spatially averaged illumination values), edges will be calculated and shown for two input images:



Visual representation of the resulted edge detection process (for input C) using MATLAB (Appendix 8.5).

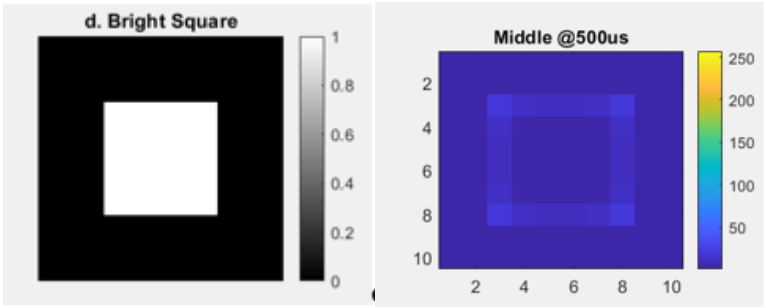
A single edge can be observed after the resulting edge detection process, confirming the applicability of the used edge detection algorithm to this work's image sensing array. It can be seen that the edge does not distinct highly from its scene and the scene's boundaries.



Visual representation of the resulted edge detection process (for input D) using MATLAB (Appendix 8.5).

To improve the resulting edges, the process was repeated for input D (which represent a more realistic input, and not only limited to a straight-line interface) with different feedback times at the feedback operation phase of the process.

The resulting edges are bolder proportionally to the length of feedback time. The longer we allow the array to perform feedback, the more spatially averaged the scene is, resulting in "sharper" edges.



Illustrated above is the input (left) and the output (right) of the designed system

7. Summary and conclusions

The defined goals and objectives in section 3 were fully achieved; a WBCAPD model capable of dividing current as function of its input weights, while achieving the behavior of a photodiode, and later integrated into an image sensor, was designed for future hardware-implemented neural network applications.

7.1 Challenges and Solutions

Challenges and solutions during the design of the model:

- Modeling a photodiode using a capacitor is incomplete since it does not consider for photodiode saturation – a current mirror circuit, biased with $\boldsymbol{vref} = \mathbf{500}[\boldsymbol{mV}]$ was integrated into the WBCAPD model.
- Leakage current during sampling – during the readout process of the pixel, the charge accumulated at the source follower's gate is leaking through the 'tg' transistors - a Floating Diffusion Capacitance was introduced and adjusted to minimize said current.

Challenges and solutions while performing simulations on the model:

- Simulating a large array requires multiple variables for multiple currents sources that represent the scene. Manual assignment of the photo-current representing the image is rigorous and time consuming. To address this matter and input Virtuoso a large quantity of variable efficiently, a supportive MATLAB code was written, which converts an input image to a CSV file which can be imported to Virtuoso testing environment (Appendix 8.1).
- To perform control of the operation of the entire WBCAPD and pixel array, VEC file input was used. In order to generate a large number of control signals, a supportive MATLAB code was written to:
    - Perform a Rolling Shutter operation image sensing simulation control signal file (Appendix 8.2).
    - Perform a Global Shutter during feedback operation and Rolling Shutter readout operation simulation control signal file (Appendix 8.3).
- Construction of an interconnected 16×16 $Pixel$ array with 15×15 $WBCAPD$ array and a readout circuitry in Virtuoso environment requires multiple labeling of wires in a precise order. Performing this task manually is time consuming and requires labeling hundreds of wires. A MATLAB supportive code was written to perform an automatic generation of said labels which only need to be copied into the virtuoso environment as a string (Appendix 8.4).
- Implementation of edge detection algorithm was used by exporting the voltage values sampled from the array and subtraction of the normalized values between the spatially averaged image of the scene and the original scene, resulting in a matrix representing edges. This operation was achieved using a supportive MATLAB code (Appendix 8.5).

7.2 Future work

- Control time optimization – in this work long reset times were used for simplicity's sake. These times can be greatly reduced to increase frame rate of the entire array.
- Transistor sizing – this worked used the TSCM18 technology to model the WBCAPD, Pixel and their array with minimal sizing of the selected transistors.
- Peripheral pixels' output is different from the "deeper" pixels in the array likely due to smaller extrinsic capacitance from the neighboring WBCAPDs. While the middle pixels have four inputs from four WBCAPDs, the peripheral pixels "see" only two WBCAPD, while the pixels in the corners are connected to only one. Additional research on the matter, with consideration of active area dimensions, should be considered to remove the voltage difference at the perimeter of the array.
- Implementation of additional connectivity and control variations to test different image processing algorithms at the array hardware level.

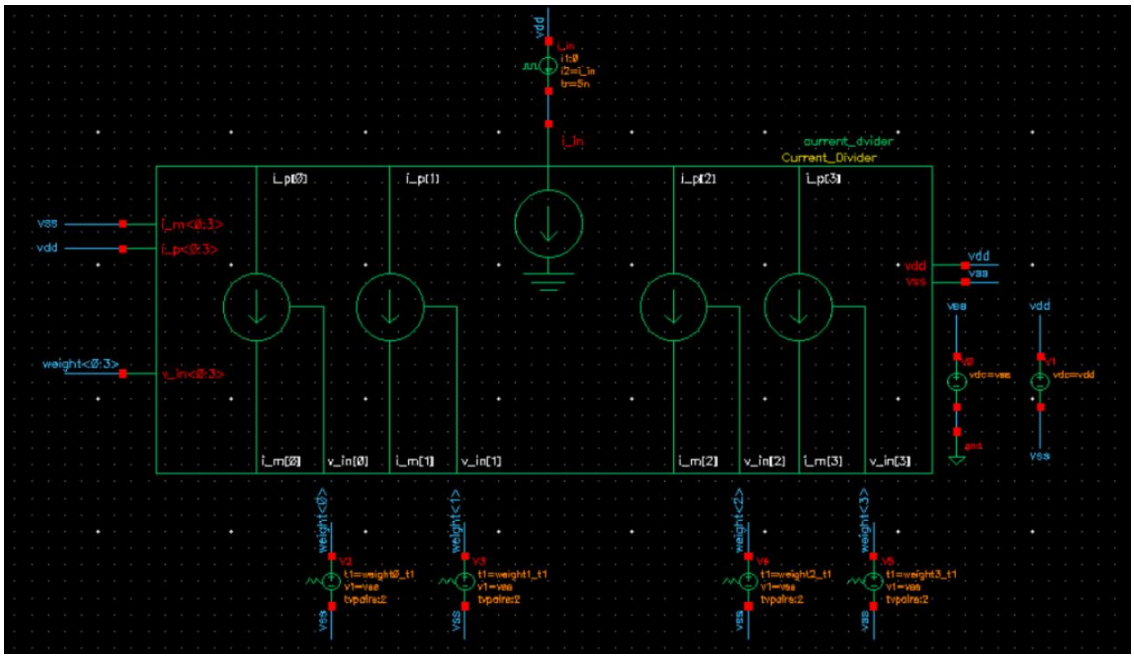
7.3 Conclusions

This work has shown that a WBCAPD array can be implemented in hardware-neural networks applications. During this work, the WBCAPD device was modeled and tested. The results of these tests show that the WBCAPD array is capable of performing image sensing operation and allows for wide connectivity applications. The designed array was used to implement an edge detection algorithm and showed promising results. The WBCAPD array operation depends on the connectivity configuration and pixel control. We think that additional research on the WBCAPD array should be done.

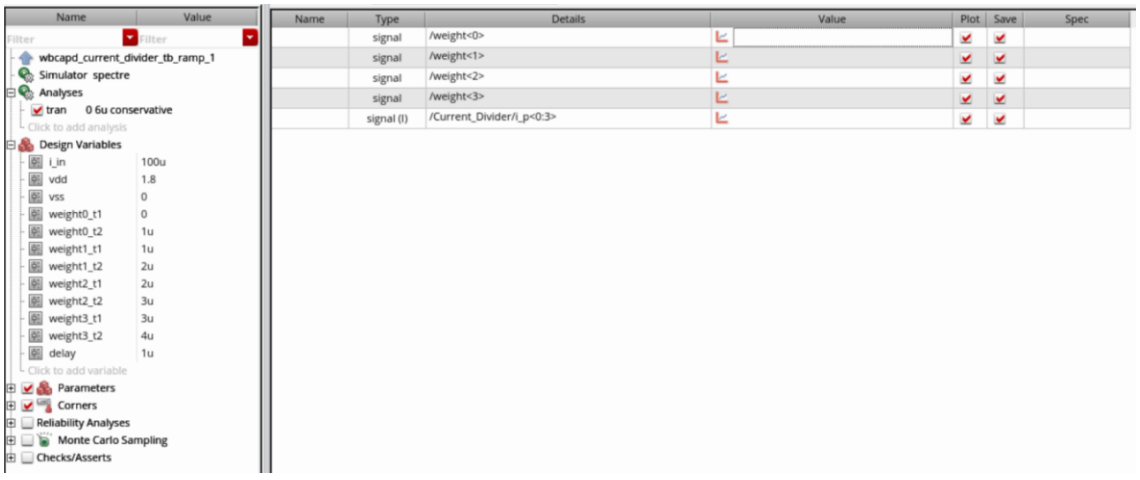
7. Appendix

7.1 WBCAPD Verilog-A device current division test setup (cell:“wbcapd_current_divider_ramp”):

Schematic:



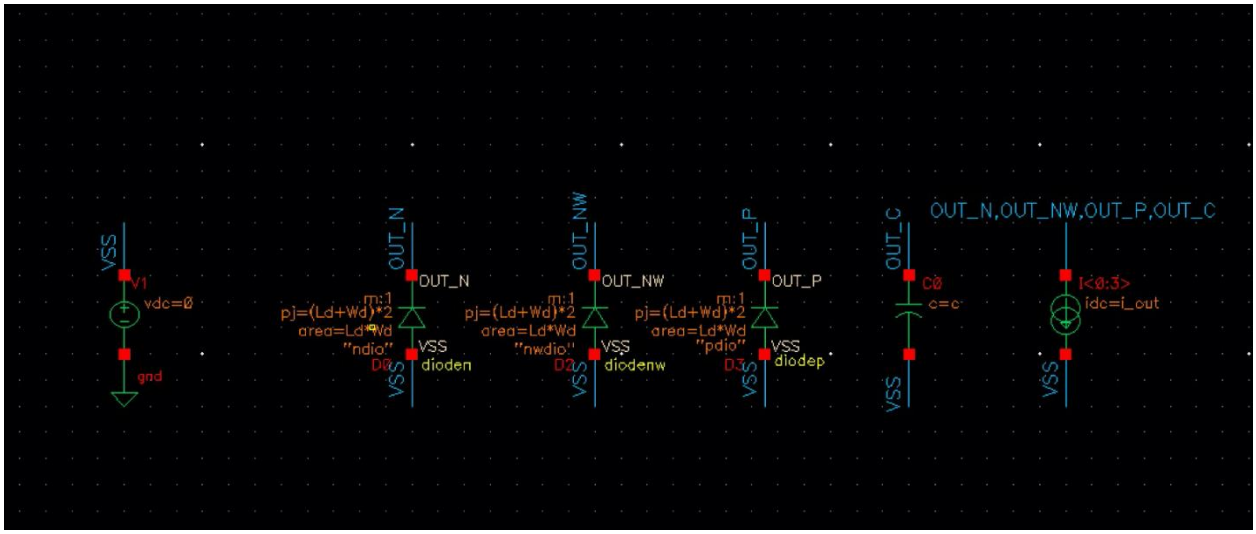
Simulation Variables:



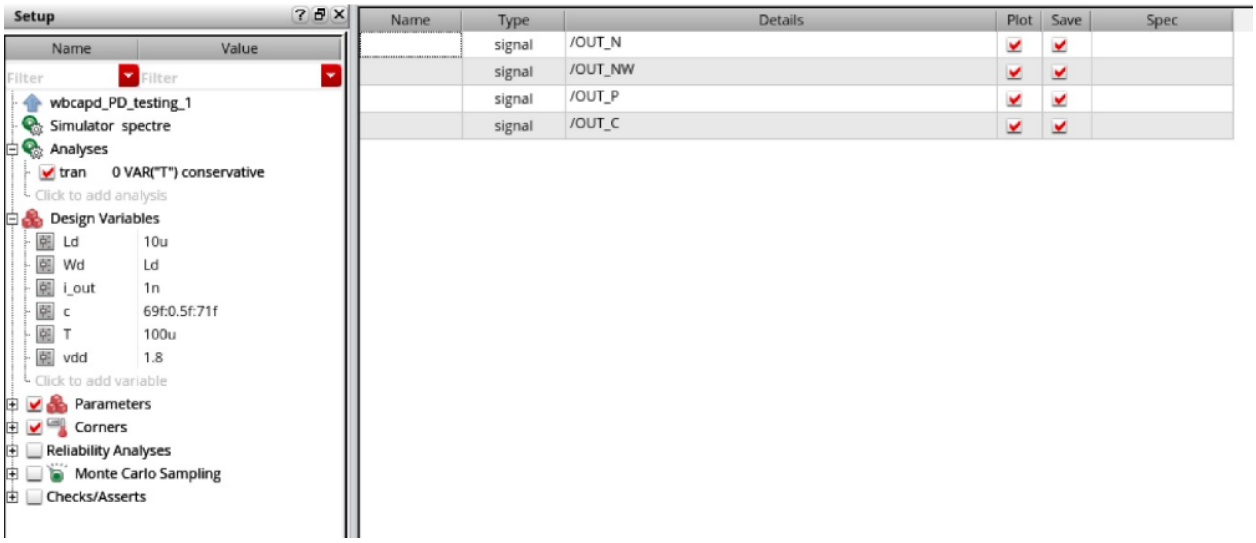
7.2WBCAPD Complete Model (Photodiode) Simulations:

7.2.1 TSMC18 photodiode comparison simulation:

Simulation Schematic:

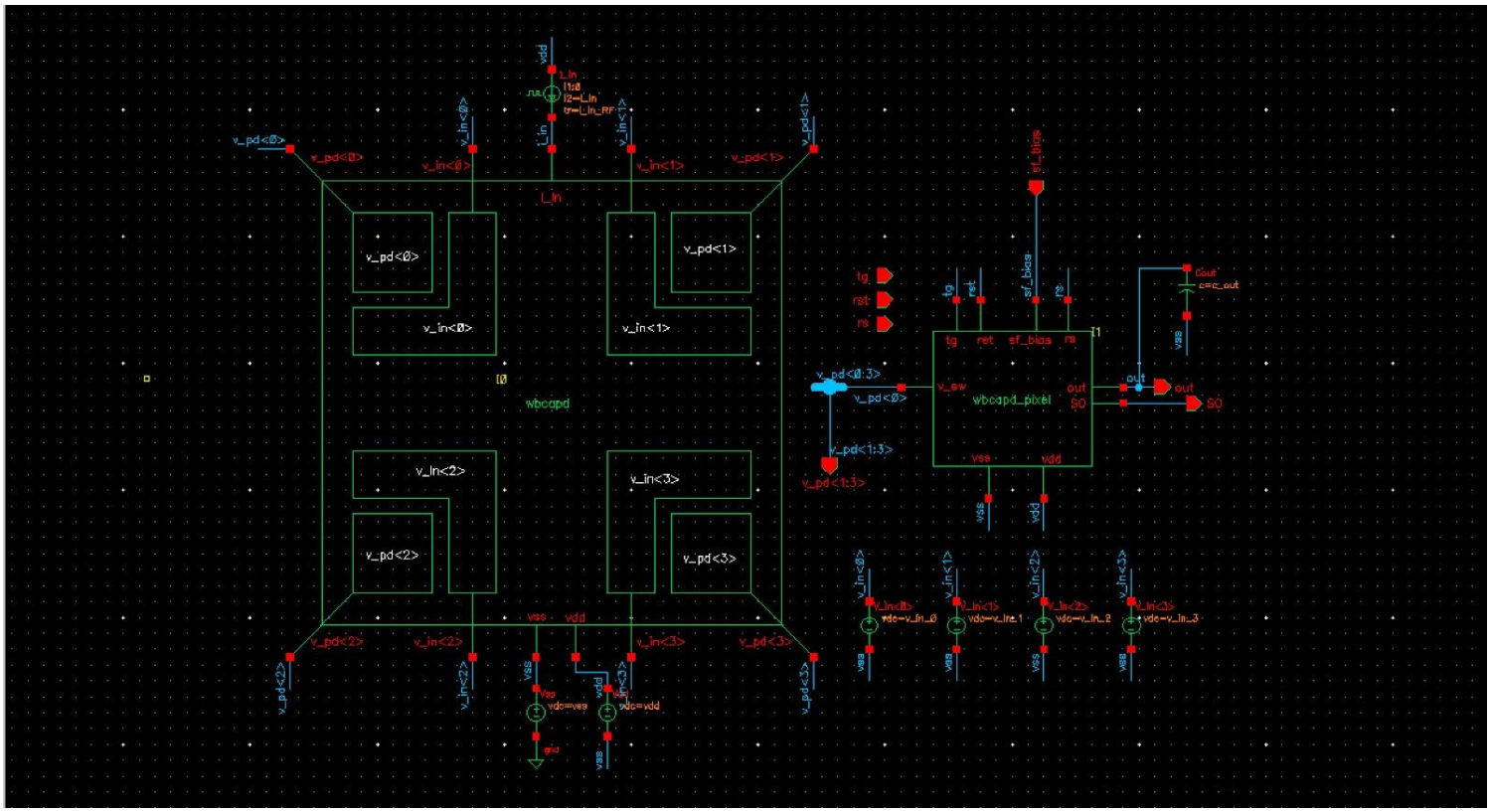


Variables:



7.3 Single Pixel Level Simulations:

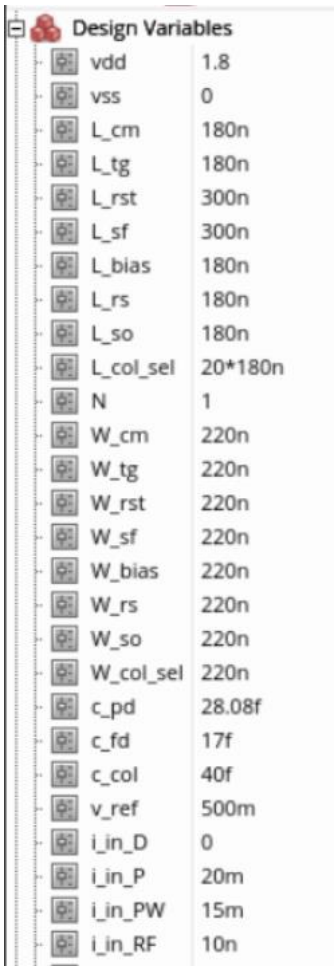
Single Pixel Test bench schematic:



7.4 Array Simulations:

Image Sensing and feedback Simulation:

Simulation Variables:



8. MATLAB Supportive Code appendix:

This section will provide documentation of the used supportive MATLAB code, for repeatability purposes of this work and it's future expansion.

8.1 Automated Generation of 'csv' file for simulations variables from a given 15×15 image.

```
%   1. Initial Values for Simulation
i_min = 1e-18; % [A]
max_photo_current = 200e-12; % [A]
i_max = max_photo_current;
sizing_factor = 1/255;
%            wbcapd array dimension (mxn)
m = 15;
n = m;
%   2.  Matrix initialization
A = imread('image.bmp');
img_in = rgb2gray(A);
%   5.  Calculate Current Values from greyscale
img_in_t=img_in';
val = reshape(img_in_t, 1, []);
i_in = i_min + (i_max-i_min) * sizing_factor * val;
%   6.   Variable Setting
title_row = {'Type', 'Name', 'Value', 'Enabled', 'Tags', 'Notes'};
%Simulation Variables
variable_block = {'variable','vdd', '1.8', 'enabled','' ,'' ;
                  'variable','vss', '0', 'enabled','' ,'' ;
                  'variable','L_cm',            '180n', 'enabled','' ,'';
                  'variable','L_tg',            '180n', 'enabled','' ,'';
                  'variable','L_rst',           '300n', 'enabled','' ,'';
                  'variable','L_sf',            '300n', 'enabled','' ,'';
                  'variable','L_bias',          '180n', 'enabled','' ,'';
                  'variable','L_rs',            '180n', 'enabled','' ,'';
                  'variable','L_so',            '180n', 'enabled','' ,'';
                  'variable','L_col_sel',       '30*180n', 'enabled','' ,'';
                  'variable','N',                '1',      'enabled','' ,'';
                  'variable','W_cm',            '220n', 'enabled','' ,'';
                  'variable','W_tg',            '220n', 'enabled','' ,'';
                  'variable','W_rst',           '220n', 'enabled','' ,'';
                  'variable','W_sf',            '3*220n', 'enabled','' ,'';
                  'variable','W_bias',          '220n', 'enabled','' ,'';
                  'variable','W_rs',            '3*220n', 'enabled','' ,'';
                  'variable','W_so',            '220n', 'enabled','' ,'';
                  'variable','W_col_sel',       '220n', 'enabled','' ,'';
                  'variable','c_pd',            '28.08f', 'enabled','' ,'';
                  'variable','c_fd',            '17f', 'enabled','' ,'';
                  'variable','c_column',        '40f', 'enabled','' ,'';
                  'variable','v_ref',           '500m', 'enabled','' ,'';
                  'variable','i_in_D',  '0', 'enabled','' ,'';
                  'variable','i_in_P',  '20m', 'enabled','' ,'';
                  'variable','i_in_PW', '15m', 'enabled','' ,'';
                  'variable','i_in_RF', '10n', 'enabled','' ,'';
                  };
```

```
%   7.  Image-Current Values data for csv table file
str = 'i_in%d';
current_block = cell(n*n, 6);
for i = 1:(m*n)
    current_block(i,:) = {'variable',sprintf(str,i-1),i_in(i), 'enabled','',''};
end
image(img_in);
colorbar;


%   8.  write data to cvs file
data = title_row;
data = vertcat (data, variable_block);
data = vertcat (data, current_block);
writecell (data,'simulation_parameters.csv');
```

8.2Automatically generated control signals for 16×16 Image Sensing Pixel array (Rolling Shutter operation and readout):

```
m=16; %rows
n=m; %columns
                              num_of_pixels = m*n;
%                           (number of WBCAPD = (m-1)*(n-1) )

% times periods for transistor control (depends on "is_tb" simulation time values)
%time is in microsecond (us  = 10^-6 second)
t_init = 0;                       % initial time
% time periods of waveform
t_pre_reset_hold = 100;             % time to discharge FD node before opening TG for
reset (per row) - open tg
tg_reset_period  = 300;             % period of time TG is open for reset stage (per
row) - close tg
t_post_reset_hold = 100;            % time to discharge FD node after closing TG
after reset (per row) - close rst
t_integration = 110;                % time for photodiode discharge (integration
time)
t_sample_period = 110;              % time for sample and hold (tg open second time)
t_pre_read_hold = 20;               % time hold pre readout
t_row_read_period = 30;             % time for readout (rs open)

pixel_time = t_pre_reset_hold + tg_reset_period + t_integration + t_sample_period +
t_pre_read_hold + t_row_read_period;
t_row_start = t_init;
time_stage = zeros(1,8) ;

time_stage = ...
    [t_row_start...
    t_row_start + t_pre_reset_hold...
    t_row_start + t_pre_reset_hold + tg_reset_period...
    t_row_start + t_pre_reset_hold + tg_reset_period + t_post_reset_hold...
    t_row_start + t_pre_reset_hold + tg_reset_period + t_integration...
    t_row_start + t_pre_reset_hold + tg_reset_period + t_integration +
      t_sample_period...
    t_row_start + t_pre_reset_hold + tg_reset_period + t_integration +
t_sample_period + t_pre_read_hold...
    t_row_start + t_pre_reset_hold + tg_reset_period + t_integration +
t_sample_period + t_pre_read_hold + t_row_read_period...


    ];

%time and control signal cell initialization

pixel_control = cell(8 * m +1, 1 + 5 * m);
for i=1:8*m
    for j=1:(5*m+1)
        pixel_control{i,j}=0;
    end
end
```

```
%                       TIME VECTOR
k=0;
t_stage = 0;
row_delay = 30;
for i=1:m
    for j=1:8
        pixel_control{i+k+j-1,1} = time_stage(j) + t_stage;
    end
    t_stage =  i*pixel_time + row_delay*i;
    k=k+7;
end

                                        % PER ROW CONTROL SIGNALS
           %tg      rst  sf_bias     rs   col_out
signals = [ 0       1       0       0       0     ;...
            1       1       0       0       0     ;...
            0       1       0       0       0     ;...
            0       0       0       0       0     ;...
            1       0       0       0       0     ;...
            0       0       0       0       0     ;...
            0       0       0       1       1     ;...
            0       0       0       0       1    ];

%                   ASSIGN CONTROL SIGNALS PER ROW OF PIXELS

for i =1:m
    for k=1:5
        for j=1:8
            pixel_control{(i-1)*8+j,(i-1)*5+1+k}= signals(j,k);
        end
    end
end

%                   COLUMN OUT CONTROL

for i =1:8*m
    for j=1:5*m+1
        %open all  col transistor during readout
        pixel_control{(i-1)*8+8,(j-1)*5+6}=1; % signal(8,5) = 1
        pixel_control{(i-1)*8+7,(j-1)*5+6}=1; % signal(7,5) = 1
        %simultanious row reset
        pixel_control{(i-1)*8+3,(j-1)*5+3}=1; % signal(1,2) = 1
        pixel_control{(i-1)*8+2,(j-1)*5+3}=1; % signal(2,2) = 1
        pixel_control{(i-1)*8+1,(j-1)*5+3}=1; % signal(3,2) = 1
    end
end
```

```
%                    EXPORT TO VEC
% VEC File Structure
% radix ( 1 bit )
% io (inputs only)
% vname (tg<0> .... rst<0> .... sf_bias<0> ... rs<0> ........)
% trise
% tfall
% vih , vil, voh, vol
%time vector + signal matrix
fid = fopen('image_sense_vec_v1.vec','w');
fprintf(fid, 'radix\t');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       radix
for i=1:m*5
    fprintf(fid,'1 \t');
end
fprintf(fid, '\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       i/o
fprintf(fid, 'io \t');
for i=1:m*5
    fprintf(fid,'i \t');
end
fprintf(fid, '\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       vname
fprintf(fid, 'vname \t');
for i=1:m
    fprintf(fid,'tg<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> \t');

    fprintf(fid,'rst<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> \t');

    fprintf(fid,'sf_bias<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> \t');

    fprintf(fid,'rs<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> \t');

    fprintf(fid,'col_sel<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> \t');

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       unit step parameters
fprintf(fid,'\n');
fprintf(fid,'tunit \t us \n');
fprintf(fid,'trise \t 0.005 \n');
fprintf(fid,'tfall \t 0.005 \n');
fprintf(fid,'vih \t 1.8 \n');
fprintf(fid,'vil \t 0 \n');
fprintf(fid,'voh \t 1.8 \n');
fprintf(fid,'vol \t 0 \n');
```

```
% PIXEL CONTROL SIGNALS FINAL MAT

for i=1:8*m
    for j=1:(5*m+1)
        fprintf(fid, num2str(pixel_control{i,j}));
        fprintf(fid,'\t');
        if j==1
            fprintf(fid,'\t');
        end
    end
    fprintf(fid,'\n');
end
% END SIM

fprintf(fid,num2str(pixel_control{m*8,1} + row_delay));
fprintf(fid,'\t \t');
for i=2:(5*m+1)
    fprintf(fid,'0 \t');
end
fclose (fid);
```

8.3Automatically generated control signals for 16×16 Pixel Feedback array (Global feedback operation and Rolling Shutter readout):

```
m=16; %rows
n=m; %columns
num_of_pixels = m*n;
%                          (number of WBCAPD = (m-1)*(n-1) )
%time and control signal cell initialization
num_of_gs_stages = 3; % rst fd, rst pd, feedback
gshutter_signals = 4; %number of elements in the vector time_stage used globally (tg,
rst, sf_bias, so_sel)
rshutter_signals = 2*m; % rs and col_sel - (readout)
control_row = num_of_gs_stages + rshutter_signals;
control_col = 1+ gshutter_signals +rshutter_signals;
pixel_control = cell(control_row, control_col);

for i=1:control_row
    for j=1:control_col
        pixel_control{i,j}=0;
    end
end

t_fd_rst = 10;
t_pd_rst = 110;
%t_feedback = 110; %equal to image sensing integration time
%t_feedback = 60;   % short feedback time
%t_feedback = 200;   % long feedback time
t_feedback = 500;   % longer feedback time

t_gs = t_fd_rst + t_pd_rst + t_feedback;

pixel_control{1,3}=1;
pixel_control{2,1}=t_fd_rst;
pixel_control{2,2}=1;
pixel_control{2,3}=1;
pixel_control{3,1}=t_fd_rst + t_pd_rst;
pixel_control{3,2}=1;
pixel_control{3,4}=1;
pixel_control{3,5}=1;

% add time values for readout
read_period = 1;

k=0;
for i=4:control_row
    pixel_control{i,1}= t_gs + k*read_period;
    k=k+1;

end


%   constant open col_sel during readout for all columns
```

```
for i=4:control_row
    for j=7:2:control_col
        pixel_control{i,j}= 1;
    end
end
%       Open rs for each row readout
k=6;
for i=4:2:control_row
        pixel_control{i,k}= 1;
        k=k+2;
end

%                   EXPORT TO VEC

% VEC File Structure
% radix ( 1 bit )
% io (inputs only)
% vname (tg<0> .... rst<0> .... sf_bias<0> ... rs<0> ........)
% trise
% tfall
% vih , vil, voh, vol
%time vector + signal matrix

fid = fopen('feedback_vec.vec','w');

fprintf(fid, 'radix\t');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       radix
for i=1:gshutter_signals + m*2
    fprintf(fid,'1 \t');
end
fprintf(fid, '\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       i/o
fprintf(fid, 'io \t');
for i=1:gshutter_signals + m*2
    fprintf(fid,'i \t');
end
fprintf(fid, '\n');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%       vname
fprintf(fid, 'vname \t');
    fprintf(fid,'tg ');
    fprintf(fid,'rst ');
    fprintf(fid,'sf_bias ');
    fprintf(fid,'so_sel ');
for i=1:m
    fprintf(fid,'rs<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> ');

    fprintf(fid,'col_sel<');
    fprintf(fid,num2str(i-1));
    fprintf(fid,'> ');
end
```

```
% unit step parameters
fprintf(fid,'\n');
fprintf(fid,'tunit \t us \n');
fprintf(fid,'trise \t 0.005 \n');
fprintf(fid,'tfall \t 0.005 \n');
fprintf(fid,'vih \t 1.8 \n');
fprintf(fid,'vil \t 0 \n');
fprintf(fid,'voh \t 1.8 \n');
fprintf(fid,'vol \t 0 \n');

% PIXEL CONTROL SIGNALS FINAL MAT

for i=1:control_row
    for j=1:control_col
        fprintf(fid, num2str(pixel_control{i,j}));
        fprintf(fid,'\t');
        if j==1
            fprintf(fid,'\t');
        end
    end
    fprintf(fid,'\n');
end

fclose (fid);
```

8.4Automatically generated wiring labels:

```
wbcapd_arr = cell(m,n);
for i = 1:m
    for j = 1:n
        wbcapd_arr{i,j} = zeros(2,4); % 4 modulation + 4 colletcion
    end
end
% connectivity
k = 0;
for i = 1:m
    for j = 1:n
        wbcapd_arr{i,j} = {strcat('SO<',int2str(n*(i-1)+(j-1)+k),'>')...    %wbcapd
TL modulation to adjacent pixel SO
                           strcat('SO<',int2str(n*(i-1)+(j)+k),'>')...      %wbcapd
TR modulation to adjacent pixel SO
                           strcat('SO<',int2str(n*(i)+(j)+k),'>')...        %wbcapd
BL modulation to adjacent pixel SO
                           strcat('SO<',int2str(n*(i)+(j+1)+k),'>');        %wbcapd
BR modulation to adjacent pixel SO
                           'v_in<0>'...   %wbcapd TL collection to adjacent pixel BR
Storage Well
                           'v_in<1>'...   %wbcapd TR collection to  adjacentpixel  BL
Storage Well
                           'v_in<2>'...   %wbcapd BL collection to adjacent pixel TR
Storage Well
                           'v_in<3>'...   %wbcapd BR collection to adjacent pixel TL
Storage Well
                           };
    end
    k = k +1 ;
end
 %                                  4-Input-Pixel array initialization & connectivity
m = m+1; %pixel rows
n = n+1; %pixel columns
% 4 Storage-Well inputs + SO
% TL-1, TR-2, BL-3, BR-4, SO-5
pix_arr = cell(m,n);
for i=1:m
    for j=1:n
        pix_arr{i,j}=zeros(1,5);
    end
end
%corner connectivity
pix_arr{1,1} = {'NC','NC','NC','v_out_TL<0>','SO<0>'};
pix_arr{1,n} = {'NC','NC',strcat('v_out_TR<',int2str(n-2),'>'),'NC',
strcat('SO<',int2str(n),'>')};
pix_arr{m,1} = {'NC',strcat('v_out_BL<',int2str((m-3)*n+2),'>'),'NC','NC',
strcat('SO<',int2str(n*(m-1)),'>')};
pix_arr{m,n} = {strcat('v_out_BR<',int2str((m-1)*(n-1) - 1),'>'),'NC','NC','NC',
strcat('SO<',int2str((m*n)-1),'>')};
```

```
%top edge connectivity
for i=2:n-1
   pix_arr{1,i} = {'NC','NC',...
       strcat('v_out_TR<',int2str(i-2),'>'),...
       strcat('v_out_TL<',int2str(i-1),'>'),...
       strcat('SO<',int2str(i-2),'>')}; % 4 storage-well inputs+ SO  in each pixel
end
%bottom edge connectivity
for i = 2:n-1
    pix_arr{n,i} = {strcat('v_out_BR<',int2str((m-2)*(n-1)+i-2),'>'),...
        strcat('v_out_BL<',int2str((m-2)*(n-1)+i-1),'>'),...
        'NC', 'NC',...
        strcat('SO<',int2str((m-1)*n+1),'>')};
end
%left edge connectivity
for i= 2 : m-1
    pix_arr{i,1} = {'NC',...
        strcat('v_out_BL<',int2str((i-2)*(n-1)),'>'),...
        'NC',...
        strcat('v_out_TL<',int2str((i-1)*(n-1)),'>'),...
        strcat('SO<',int2str((i-1)*n),'>')};
end
%right edge connectivity
k=1;
for i= 2 : m-1
    pix_arr{i,n} = {strcat('v_out_BR<',int2str((i-1)*(n-2)+k-1),'>'),...
        'NC',...
        strcat('v_out_TR<',int2str((i)*(n-2)+k),'>'),...
        'NC',...
        strcat('SO<',int2str(i*n-1),'>')};
    k=k+1;
end
%middle connectivity
for i = 2:m-1
    for j = 2:n-1
        pix_arr{i,j} = {strcat('v_out_BR<',int2str((i-2)*(n-1)+(j-2)),'>'),...
            strcat('v_out_BL<',int2str((i-2)*(n-1)+(j-1)),'>'),...
            strcat('v_out_TR<',int2str((i-1)*(n-1)+(j-2)),'>'),...
            strcat('v_out_TL<',int2str((i-1)*(n-1)+(j-1)),'>'),...
            strcat('SO<',int2str(n*(i-1)+j-1),'>')};
    end
end
% PIXEL input (output from WBCAPD) connectivity strings
pixel_connectivity_string = cell(4,1);
    for i = 1:m
        for j=1:n
            pixel_connectivity_string{1,1} =
strcat(pixel_connectivity_string{1,1},pix_arr{i,j}(1)) ;   %TL
            pixel_connectivity_string{1,1} =
strcat(pixel_connectivity_string{1,1},',');                %TL

            pixel_connectivity_string{2,1} =
strcat(pixel_connectivity_string{2,1},pix_arr{i,j}(2)) ;   %TR
```

```
            pixel_connectivity_string{2,1} =
strcat(pixel_connectivity_string{2,1},',');                 %TR

            pixel_connectivity_string{3,1} =
strcat(pixel_connectivity_string{3,1},pix_arr{i,j}(3)) ;   %BL
            pixel_connectivity_string{3,1} =
strcat(pixel_connectivity_string{3,1},',');                 %BL

            pixel_connectivity_string{4,1} =
strcat(pixel_connectivity_string{4,1},pix_arr{i,j}(4)) ;   %BR
            pixel_connectivity_string{4,1} =
strcat(pixel_connectivity_string{4,1},',');                 %BR

        end
    end
% WBCAPD input (output from PIXEL SO [for FEEDBACK ])  Connectivity Strings
m=m-1;
n=n-1;
wbcapd_connectivity_string = cell(4,1);
    for i = 1:m
        for j=1:n
            wbcapd_connectivity_string{1,1} = strcat(
wbcapd_connectivity_string{1,1},wbcapd_arr{i,j}{1,1},','); % TL
            wbcapd_connectivity_string{2,1} = strcat(
wbcapd_connectivity_string{2,1},wbcapd_arr{i,j}{1,2},','); % TR
            wbcapd_connectivity_string{3,1} = strcat(
wbcapd_connectivity_string{3,1},wbcapd_arr{i,j}{1,3},','); % BL
            wbcapd_connectivity_string{4,1} = strcat(
wbcapd_connectivity_string{4,1},wbcapd_arr{i,j}{1,4},','); % BR
        end
    end
connections = cell(2,1);
 connections {1,1} = pixel_connectivity_string;      %cell {1,1} - pixel input strings
(TL,TR,BL,BR)
 connections {2,1} = wbcapd_connectivity_string;     %cell {2,1} - wbcapd input
strings (TL,TR,BL,BR)
 pix_in = strings (4,1);
 pix_in(1,1) = connections{1,1}{1,1}(1);
 pix_in(2,1) = connections{1,1}{2,1}(1);
 pix_in(3,1) = connections{1,1}{3,1}(1);
 pix_in(4,1) = connections{1,1}{4,1}(1);

 wbcapd_in = strings (4,1);
 wbcapd_in(1,1) = connections{2,1}{1,1};
 wbcapd_in(2,1) = connections{2,1}{2,1};
 wbcapd_in(3,1) = connections{2,1}{3,1};
 wbcapd_in(4,1) = connections{2,1}{4,1};
%               Virtuoso Labels (remove: " ' " in the begining and end or string and
" , " at the end of the string)
 total = [pix_in ; ' ' ; wbcapd_in];
 writematrix(total, 'WBCAPD_input_edge_detection_label_names.xls');
% final file save to edge_detection_label_names.xls

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                       control signal strings for pixel (remove ',' at the
%                       end of the string)
m=m+1;
n=n+1;
%control strings (pixel inputs)
tg_str = strings(1,m*n);
rst_str = strings(1,m*n);
sf_bias_str = strings(1,m*n);
rs_str = strings(1,m*n);
col_str = strings(1,m*n);
%   ROW control signals strings mat
k=0;
for i=1:m
    for j=1:n
        tg_str(1,i+j+k-1) = strcat('tg<',int2str(i-1),'>',',');
        rst_str(1,i+j+k-1) = strcat('rst<',int2str(i-1),'>',',');
        sf_bias_str(1,i+j+k-1) = strcat('sf_bias<',int2str(i-1),'>',',');
        rs_str(1,i+j+k-1) = strcat('rs<',int2str(i-1),'>',',');
    end
    k=k+n-1;
end
%  COLUMN control signals strings mat (For readout operation)
count =0;
j=1;
for i=1:m*n
        col_str(1,i) = strcat('col<',num2str(j-1),'>',',');
        count = count +1;
        if count==n
            j=0;
            count =0;
        end

        j = j+1;
end
```

8.5Automatic analog voltage value to image conversion from Cadence Virtuoso array simulation results for edge detection implementation:

```
is_voltage_raw = readmatrix ('is_results.csv'); %impoted file from Virtuoso
is_voltage(:,:) = is_voltage_raw(:,2:17); %removes "cycle" column
min_val = min(is_voltage(:));
is_voltage = is_voltage - min_val;
max_val = max(is_voltage(:));
is_voltage = ((is_voltage)*255);
is_voltage = is_voltage/max_val;
image(is_voltage);

fb_voltage_raw = readmatrix ('fb_results.csv'); %impoted file from Virtuoso
fb_voltage(:,:) = fb_voltage_raw(:,2:17); %removes "cycle" column
min_val = min(fb_voltage(:));
fb_voltage = fb_voltage - min_val;
max_val = max(fb_voltage(:));
fb_voltage = ((fb_voltage)*255);
fb_voltage = fb_voltage/max_val;
image(fb_voltage);


edge_mat = fb_voltage - is_voltage;
image(edge_mat);
```

9. Bibliography

1. This work is based on “Weight-Based Current Assisted Photonic Demodulator (WBCAPD) – Analysis, Design and Applications”, submitted in 2021, Bar-Ilan University, Ramat Gan, Israel.
2. M. R. Ahmed and B. K. Sujatha, "A review on methods, issues and challenges in neuromorphic engineering," *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015, pp. 0899-0903, doi: 10.1109/ICCSP.2015.7322626.
3. V. Torre and T. A. Poggio, "On Edge Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 2, pp. 147-163, March 1986, doi: 10.1109/TPAMI.1986.4767769.
4. C. A. Mead and M. A. Mahowald, _A silicon model of early visual processing,_ Neural networks, vol. 1, no. 1, pp. 91_97, 1988.
5. D. Van Nieuwenhove, W. van der Tempel, and M. Kujik, _Novel standard cmos detector using majority current for guiding photo-generated electrons towards detecting junctions, in Proceedings Symposium IEEE/LEOS Benelux, 2005, pp. 229_232.
6. R. Coath, J. Crooks, A. Godbeer, M. Wilson, R. Turchetta and the SPiFeR Collaboration, Advanced Pixel Architectures for Scientific Image Sensors, Rutherford Appleton Laboratory, Science and Technology Facilities Council, UK. https://heplnm061.pp.rl.ac.uk/display/spider/