# Design of Combinatorial APS Pixel Addressing Circuitry using the Sky130 Process Design Kit

Carolina Vieira Souza
*Faculdade de Engenharia*
*Universidade Federal de Juiz de Fora*
Juiz de Fora, Brazil
carolina.vieira@engenharia.ufjf.br

Guilherme Ferrara Jorge Pereira
*Faculdade de Engenharia*
*Universidade Federal de Juiz de Fora*
Juiz de Fora, Brazil
guilherme.pereira2020@engenharia.ufjf.br

Estêvão Coelho Teixeira
*Faculdade de Engenharia*
*Universidade Federal de Juiz de Fora*
Juiz de Fora, Brazil
estevao.teixeira@ufjf.br

***Abstract*—Currently, CMOS imagers are found in a plethora of applications, ranging from digital cameras in cell phones to security applications. The Active Pixel Sensor (APS) is the core of the CMOS imagers, consisting of a photosensitive element and three or more transistors that implement a readout circuit. Each pixel of the APS can be accessed and read by a combination of generated digital signals for Reset and Pixel Selection. This work carries out the design of digital combinatorial circuitry, composed by two decoders, used for addressing and reading an APS matrix, using the open Sky130 Process Design Kit (PDK). The circuits are described, in the context of the proposed matrix, and the open source tools used for the design are explained. Finally, the generated blocks layouts and simulation results are presented.**

***Index Terms*—APS, digital design, OpenLane, open PDK, Sky130, open source tools.**

## I. Introduction

CMOS imagers based on Active Pixel Sensor (APS) technology became commonly used since its launching in the 1990s [1]. These devices capture the luminosity onto them and create images based on the principle of photocurrent. Operating based on this principle, photons are integrated into a reverse-biased p-n junction. Digital cameras have become easily accessible and are integrated into notebooks, cell phones, tablets, and security systems. The aforementioned characteristics have made CMOS sensors attractive for these applications, due to the posibility of integrating the sensors and all the readout electronics into a same integrating circuit, leading to the concept of ”camera on chip” [2].

According to [2], the basic APS pixel is consisted of a photosensitive element (usually a photodiode) and a readout circuit, consisted by at less three transistors (the 3T pixel): a Reset transistor, a readout transistor (which acts as a buffer and a charge amplifier) and a selection transistor, which ties the selected pixel to the column biased by a current source, composing a source follower configuration that that undergoes a process of reading or analog signal processing.

In addition to the pixel array itself and the readout and signal processing circuitry, the APS imager is also composed by a row and column selection logic, implemented through digital combinatorial pixel selection circuits composed of two decoders that address a particular pixel, or a group of pixels at the same line, depending upon the configuration adopted. This highlights another interesting feature of the APS: the possibility of readout of an individual pixel, or an area of pixel, instead of the entire matrix.

The release of the SkyWater Open Source Process Design Kit (PDK) in 2020 enabled undergraduate students to access chip design without restrictions like non-disclosure agreements (NDAs). With the use of free and open source tools, digital, analog and mixed-signal circuits can be designed using the process called Sky130, a mature CMOS process using the hybrid 180nm-130nm, 1.8-Volt technology node [3].

This paper presents the design of two selection circuits using the Sky130 PDK. The open source tool OpenLane, suited for the PDK, was used, which is a platform supporting other open source tools, integrating various stages of the silicon implementation process into a process flow; converting a digital design described in Verilog, along with configuration files into an integrated circuit layout database file format (GDSII). Other open source tools were used for layout verification and simulation of results.

The work is organized as follows: in Section II, the decoders used in the pixel selection matrix are explained in the context of the APS. Section III describes how OpenLane was used and operated to achieve the final result, along with some preliminary simulations. In Section IV, the verifications and simulations of the obtained results are shown, along with the tools used for this purpose. Finally, conclusions are presented in Section V.

## II. Decoders

Decoders are logic circuits that receive a set of input signals representing a binary number and usually activate only one output, corresponding to the received number [4]. Typically, the maximum number of outputs is $2^n$ (where n is the number of inputs). Their importance is reflected in their widespread use in digital designs, ranging from precise memory selection to flawless code conversion (such as binary to decimal) and efficient data routing.

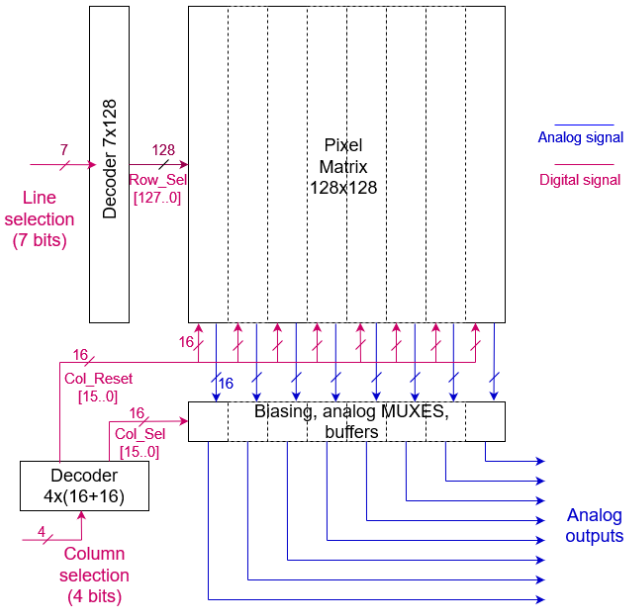



Fig. 1: Block diagram of the APS matrix.

This work presents the development of two CMOS decoders for addressing a 128x128 APS matrix. It is important to contextualize the design of the decoders within the overall project. Given that the current stage of the project aims to enable individual access to pixels and read the analog signal for characterizing the pixels in Sky130 technology, the imager was designed in a simplistic manner, devoid of signal processing stages and analog-to-digital converters. The digital circuitry is also planned for a simple operation, without sequential circuits for pixel addressing.

A block diagram illustrating the imager structure is shown in Fig. 1, showing both the digital and analog signals paths. The 128x128 matrix is divided into eight blocks of 128 lines and 16 columns. Indeed, eight pixel outputs can be read in parallel. This arrangement requires two decoders, one with seven inputs and 128 outputs, and another with four inputs and duplicated 16 outputs.

The row addressing logic is implemented by a 7x128 decoder, structured through the cascading of smaller decoders. It operates as follows: in the same row, there is a bus that enables all its pixels through a row selection transistor present in each pixel, with a HIGH logic signal at the decoder output. The seven input lines of the decoder are provided externally.

Another decoder, smaller than the row decoder, is required in order to address one of the 16 columns of each block, which will be done by an analog multiplexer, out of the scope of this paper. This decoder is also responsible for generating the Reset signals, which are brief signals common to all the pixels of a same column, applied to the Reset transistor of each column pixel. Therefore, a 4x32 decoder is needed, where the last 16 outputs will be the same as the first 16, but only activated when an Enable input is put into at a HIGH logic level. The four inputs will be externally accessible, as well as the Enable signal, which will receive an external pixel Reset signal. Due to this logic, only the pixels of the addressed columns will be reset.

## III. The OpenLane 1

The OpenLane 1 is configured as a robust open source system that facilitates the construction of physical implementation flows for digital application-specific integrated circuits (ASICs) [5]. This platform combines open source Electronic Design Automation (EDA) tools with advanced features to automate the design process. As described in [6], the tool offers a complete flow that ranges from register-transfer level (RTL) representation to the generation of the GDSII format used for chip manufacturing. Some benefits to mention include the possibility of having free and open source software (FOSS) that can be as efficient as commercial ones. One of the main advantages of OpenLane is its flexibility, allowing the design flow to be customized to meet the specific needs of each project by simply configuring its *config.json* input file.

In the context of VLSI design, a macro is a block that implements a function to be integrated into the chip, simplifying the design process, enabling its reuse, and increasing reliability since they can be tested separately [7]. Hardening a design macro is a process that can be executed by OpenLane1 starting from a Hardware Description Language (HDL) Verilog, resulting in the manufacturable layout file of the chip (GDSII format).

The flow of the OpenLane environment primarily uses tools such as OpenRoad [8], Yosys [9], and Open Circuit Design [10], and operates as follows: Yosys synthesizes the design into a gate-level netlist, and OpenSTA [11] performs static timing analysis, verifying that the design does not have setup and hold violations. Next, Design For Testability is executed by inserting scan chains and IO ports for post-fabrication testing. Then, floorplanning and placement are executed, where the user can make changes through a configuration file, as well as the positioning of the I/O pins. Every time a tool alters the netlist, a Logic Equivalence Check (LEC) is performed using Yosys, ensuring it maintains the same logic and functionality. Optimization scripts and global routing are then performed [12].

Finally, Design Rule Checking (DRC) and Layout versus Schematics (LVS) are executed by Magic, generating GDSII and LEF files used to implement the created macro into a larger design. In Fig. 2, the complete project flow is described.

### A. Design example

For the design of the decoders, it was necessary, in the case of the row decoder, to meet some physical properties, since it will be directly connected to the pixels of each row. Therefore, each output pin should have a distance of 15 $\mu$m, which will be planned pixel pitch. Through the configuration file of the flow (*config.json*), PDK support configuration files, and by analyzing the documentation [13], the following configuration was made. The configurations were implemented as follows:

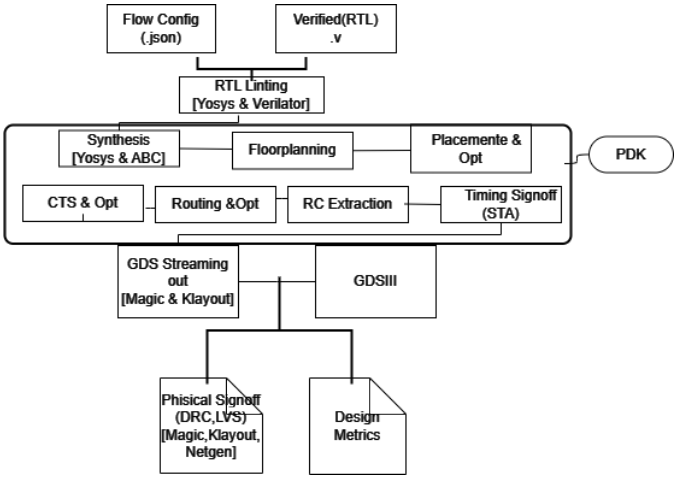



Fig. 2: OpenLane flow.

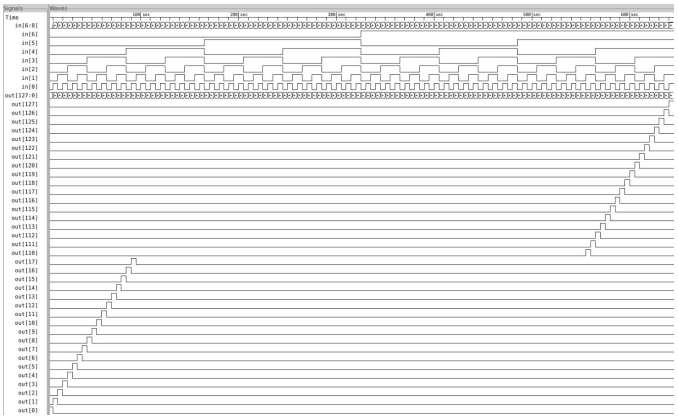


Fig. 3: Row Decoder IVerilog simulation.

- ”FP_SIZING”: ”absolute”, ensuring that the area of the decoder is precisely established;
- ”DIE_AREA”:”0.0 0.0 2000.0 100.0”, specifying this area to have a length of 2000$\mu$m and fit alongside the pixel array;
- ”FP_IO_MODE”:false, ensuring that the allocation of the I/O pins is not done automatically, along with ”FP_PIN_ORDER _CFG”, establishing an auxiliary configuration file “pin_order.cfg”, in which the input and output pins can be ordered in specific directions;
- ”RT_MAX_LAYER”: allowing the highest layer to be used in the routing process. In this case, the maximum layer defined is ”met4”. This means that OpenLane will have access to all metal layers up to ”met4” to make the interconnections between the project components.

## IV. Verification and Simulation

Before running the OpenLane1 flow, the hardware developed in Verilog must be verified. For this, the tool Icarus Verilog [14], suited for the Sky130 PDK, was used. It is a Verilog HDL compiler, as described in the IEEE-1364 Standard [15]. Therefore, after creating the design, a testbench

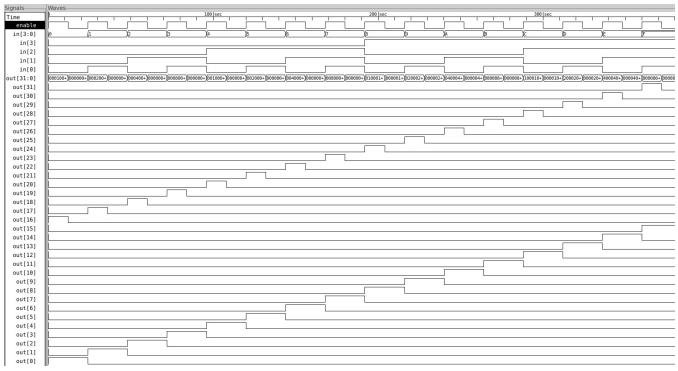
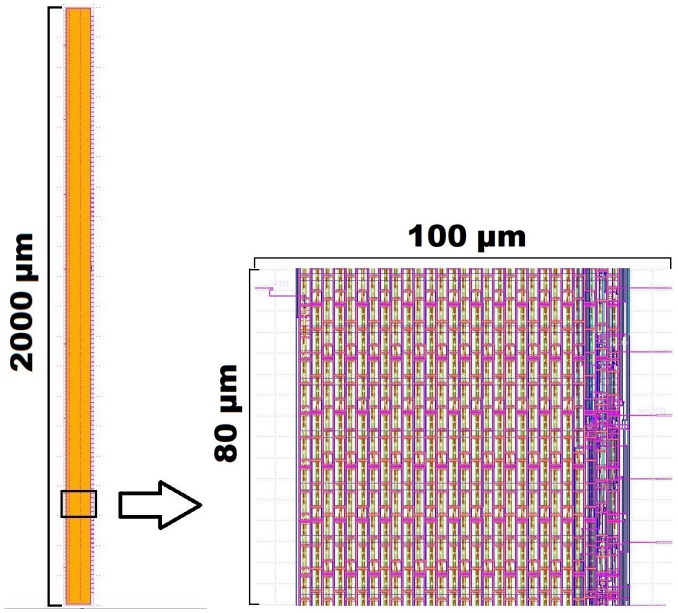


Fig. 4: Column Decoder IVerilog simulation.




(a) Full decoder. (b) Detail.

Fig. 5: Row Selection decoder layout.

file was created to evaluate if the created logic corresponds to the expected behavior. This way, it was observed that both decoders operated as expected, responding to the input signal and enabling only one output. In Fig. 3, it can be observed, besides the input signals, the first 18 and last 18 outputs of the row decoder. The other outputs were omitted for the purpose of simplifying visualization.

In the case of the column decoder, the testbench was designed to simulate the operation of the reset signal in the APS, where the ’enable’ signal, which enables half of the outputs, has a period lower than the period of the least significant input bit (LSB) of the decoder. This results in an output with a shorter period than the outputs that are not affected by this control signal, as seen in Fig. 4. Here, only for didactic purposes, the period of Enable is half of the period os the LSB. The figure shows, in sequence, the Enable (Reset) signal, the four input bits, the 16 Reset signals and the 16 Column Select signals.

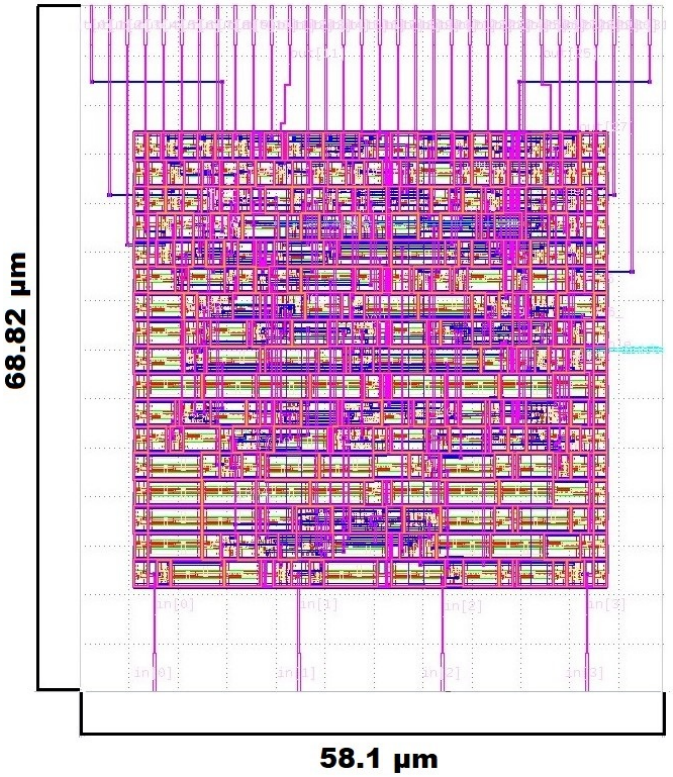



Fig. 6: Column Selection decoder layout.

After obtaining the integrated circuit layout (GDSII file) through the OpenLane flow, the tool Klayout [16], compatible with the PDK, was used. It is a viewer and editor of GDSII layouts. It allowed to check that the specifications were met, such as the distance between the output pins (15 $\mu$m), size, and number of metal layers.

The visualization of the generated layouts is depicted in Fig. 5 and Fig. 6. The obtained areas of the row selection decoder and the column selection decoder are, respectively, $200000\,\mu\mathrm{m}^2$ ($0.2\,\mathrm{mm}^2$) and $3997.3\,\mu\mathrm{m}^2$ ($\sim 0.004\,\mathrm{mm}^2$).

## V. Conclusion

This paper presented the design of two decoders in the Sky130 technology, conceived in order to address a 128x128 APS Matrix. According to the planned readout scheme, it was necessary a 7x128 decoder for row selection, while a 4x(16+16) decoder was designed both for column selecion and for Reset of the pixels of a given group of columns. The open source tools used were suited for use with the open Sky130 PDK.

The constraints for the design of the row decoder (with specific physical dimensions) were described in the text. The layouts of the decoders were shown, as well as pre-layout simulation results.

In this case, both of the designed circuits are purely combinatorial, given the simplified purpose of pixel testing for the matrix. However, the tools have proven to be useful in the design of more complex circuits, which incorporate both combinatorial and sequential features.

This project demonstrates that open source tools are viable and effective for novice designers, including undergraduate students and independent users. These tools lead to satisfactory results even in complex projects. Additionally, the open PDK associated with open source tools can contribute to democratizing integrated circuit design.

## VI. Acknowledgment

This work was supported by an Academic Professional Training Program (TPA 2023/2024), by the PROGRAD/UFJF.

## References

[1] E. R. Fossum, “CMOS Image Sensors: Electronic Camera on a Chip”, IEEE Transactions on Electron Devices, v. 44, n. 10, pp. 1689-1698, Oct. 1997.

[2] E. R. Fossum. “Digital Camera System on a Chip”, IEEE Micro, pp. 8-15, May-Jun. 1998.

[3] Skywater Open Source PDK. Available at: https://github.com/google/skywater-pdk (access May 31, 2024).

[4] D. M. Harris and S. L. Harris, Digital Design and Computer Architecture. Morgan Kaufmann, 2nd edition, 2012.

[5] M. Shalan and T. Edwards, ”Building OpenLANE: A 130nm OpenROAD-based Tapeout- Proven Flow : Invited Paper”. In: 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-6.

[6] ”OpenLane documentation” Available at: https://openlane.readthedocs.io/en/latest/ (accessed May 31, 2024).

[7] S. A. Maurya, Structured design methodology for high performance VLSI arrays. Avaiable at : https://core.ac.uk/download/pdf/79563898.pdf (accessed May 31, 2024).

[8] The OpenROAD project – foundations and realization of open and accessible design. Avaiable at: https://theopenroadproject.org (accessed May 31, 2024).

[9] yosys: Yosys Open SYnthesis Suite. [s.l: s.n.]. Available at: https://yosyshq.net/yosys/ (accessed in May 31,2024).

[10] Open Circuit Design. Avaiable at: http://opencircuitdesign.com (accessed in May 31, 2024).

[11] OpenSTA: OpenSTA engine. Avaiable at: https://github.com/The-OpenROAD-Project/OpenSTA (accessed in May 31, 2024).

[12] A. A. Ghazy and M. Shalan, ”OpenLANE: the open-source digital ASIC implementation flow”. Proceedings of the Workshop on Open-Source EDA Technology - WOSET 2020. Virtual, 2020. Available in:: https://woset-workshop.github.io/PDFs/2020/a21.pdf (accessed in May 30, 2024).

[13] OpenLane’s Flow Configuration Variables. Available at: https://github.com/The-OpenROAD-Project/OpenLane/blob/master/docs/source/reference/configuration.md (accessed in May 31, 2024).

[14] S. Williams, Iverilog: Icarus verilog. Available at: https://github.com/steveicarus/iverilog (accessed in May 31, 2024).

[15] Institute of Electrical and Electronics Engineers, ”IEEE Standard for Verilog Hardware Description Language”. IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001), pp.1-590, April 7, 2006, doi: 10.1109/IEEESTD.2006.99495.

[16] M. Koefferlein, KLayout Layout Viewer And Editor. Available at: https://www.klayout.de/intro.html (accessed in May 31, 2024).