TR-0003

OptoPC
Operations with Colors

Internal Usage Only

Abstract

The following document provides an introduction to the data flow of the OPTOPC prototypes.

Index

1. OptoPC Hybrid-Optoelectronic Processors ..... 2 2. The OptoPC Hybrid Processor ..... 3 3. The Control Board 4

3.1. Workloads ..... 5 3.2. Scheduling the Workflows ..... 6 3.3. The Control Logic ..... 7 4. The Optical Matrix 8

4.1. Example of Computation Operations ..... 11 4.2. Using Colors as a Control Line 14

5. Conclusion 18

6. Document Revision History 18

1. OptoPC Hybrid-Optoelectronic Processors

OPTOPC is working on a novel optical computer design tailored for arithmetic computation, matrix manipulation, and hyperdimensional computation, utilizing light as information carriers resulting in hybrid optoelectronic processors.

Hybrid optoelectronic processors represent a cutting-edge approach to computing that leverages the strengths of both optical and electronic technologies. By combining the highspeed, parallel processing capabilities of optics with the precision and flexibility of electronics, these systems aim to overcome the limitations of traditional electronic computing architectures.

- Optical Computing: Utilizes light to perform calculations, offering massive parallelism and high-speed data transfer.
- Electronic Computing: Relies on electrical signals to process information, providing flexibility and precision.
- Hybrid Opto-Electronic Processing: Integrates optical and electronic components to achieve a balance of speed, capacity, and precision.

Internal Usage Only

Hybrid optoelectronic processors typically consist of the following key components:

- Optical Interconnects: These components use light to transmit data between different parts of the processor, significantly reducing latency and power consumption compared to traditional electrical interconnects.
- Optical Signal Processors: These devices manipulate and process optical signals, enabling parallel operations and high-speed data processing.
- Electronic Components: These traditional electronic components are responsible for control, memory, and input/output operations.
- Optoelectronic Converters: These components bridge the gap between the optical and electronic domains, converting light signals into electrical signals and vice versa.

Advantages of Hybrid Opto-Electronic processors:

- High-Speed Parallel Processing: Optical interconnects enable massive parallelism, allowing for simultaneous processing of multiple tasks.
- Low Power Consumption: Optical interconnects consume less power than electrical interconnects, leading to energy-efficient systems.
- Large Bandwidth: Optical signals can carry much more data than electrical signals, enabling high-speed data transfer and processing.
- Reduced Latency: The speed of light allows for faster data transmission and processing compared to electrical signals.
- Potential for Novel Computing Paradigms: Hybrid systems can explore new computing paradigms, such as hyperdimensional computing, neuromorphic computing, and quantum computing.

2. The OptoPC Hybrid Processor

At a very high level, the OPTOPC hybrid processor approach is summarized by the following Figure 1:

Internal Usage Only

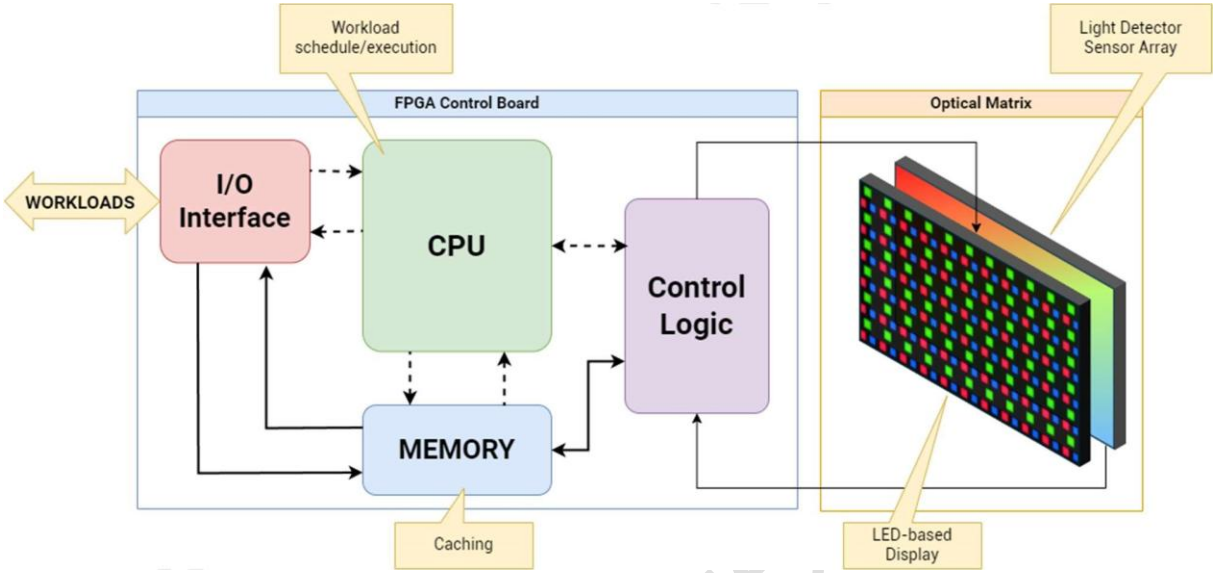


Figure 1. High-level view of the OptoPC hybrid processor approach.

The system is composed of the following key components:

- Control Board. This component is responsible for controlling one or more optical matrices, scheduling the execution of the workloads, managing the memory, and handling input/output operations.
- Optical Matrix. This component is responsible for manipulating and processing optical signals, enabling parallel operations and high-speed data processing. The optical matrix comprises an LED-based light-emitting layer, and light detectors layer. Many

  solutions for implementing the light matrix exploit different layers assembled in many different configurations.

At high-level, the operation of the system is designed as follows:

1. Data is sent to the OPTOPC processor together with the code to execute on such data (workloads).
2. The workloads (data and code) are stored in the cache memory.
3. The workload scheduler (CPU) decides which operations to execute.
4. The control logic loads the data and the related operations into the optical matrix.
5. The control logic receives computed results from the optical matrix. Data loading should be massively parallel and deterministic in time.

Internal Usage

3. The Control Board

The Control Board component is responsible for managing the workloads that need to be executed. In particular, it:

- handles input/output operations, receives the required workloads in input, and outputs the workload execution results.
- manages the memory, storing and retrieving workloads data and results.
- schedules the execution of the workloads.
- controls the optical components (SLMs) in order to set up optical computations.

During each work cycle, the control board must perform the following operations:

- receive workloads information if any, and store it in the cache memory;
- read the information of the workloads from the cache memory;
- encode the data and the related operators;
- load everything into the optical matrix, properly configuring the SLMs;
- read the result of the previous operations from the light detectors, if any.

The current implementation of the component is based on FPGA boards in order to exploit their intrinsic parallel nature regarding the execution of operations.

3.1. Workloads

Concerning the OPTOPC processor, a workload can be defined as the set composed of data required for the computation together with the operations to execute on such data. Workloads can range from simple tasks to complex procedures like processing large-scale data analytics or artificial-intelligence application data.

The operations that can be performed in a single work cycle (clock cycle) are named “atomic operations.” Examples of atomic operations are the logical operators: AND, OR, NAND, NOR, XOR, XNOR, and NOT.

Any atomic operation must be massively parallelizable. Therefore, atomic operations are designed to work on arrays of homogeneous data.

On the other hand, operations that require more than one clock cycle are named “composite operations.”

Internal Usage Only

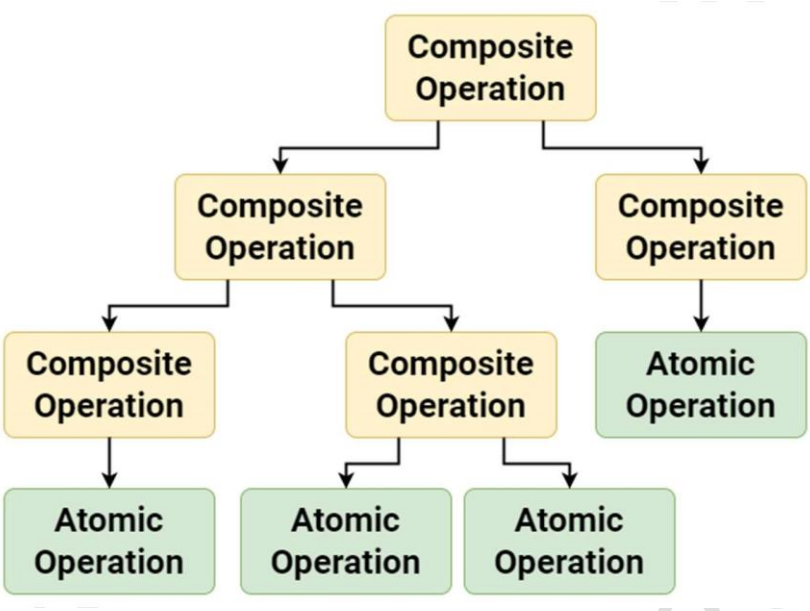


Figure 2. Hierarchical definition of the operations.

As show in Figure 2, composite operations are defined hierarchically as combinations of atomic and other composite operations. We can represent a composite operation as a tree where every leaf is an atomic operation while parent nodes are composite operations. Some atomic operations can be executed in parallel, while others must be executed sequentially.

Every composite operation contains information related to the dependencies among its child operations.

Custom external toolchains, which are under development, are required for translating highlevel source code into workloads that can be interpreted by the processor.

Internal Usage Only

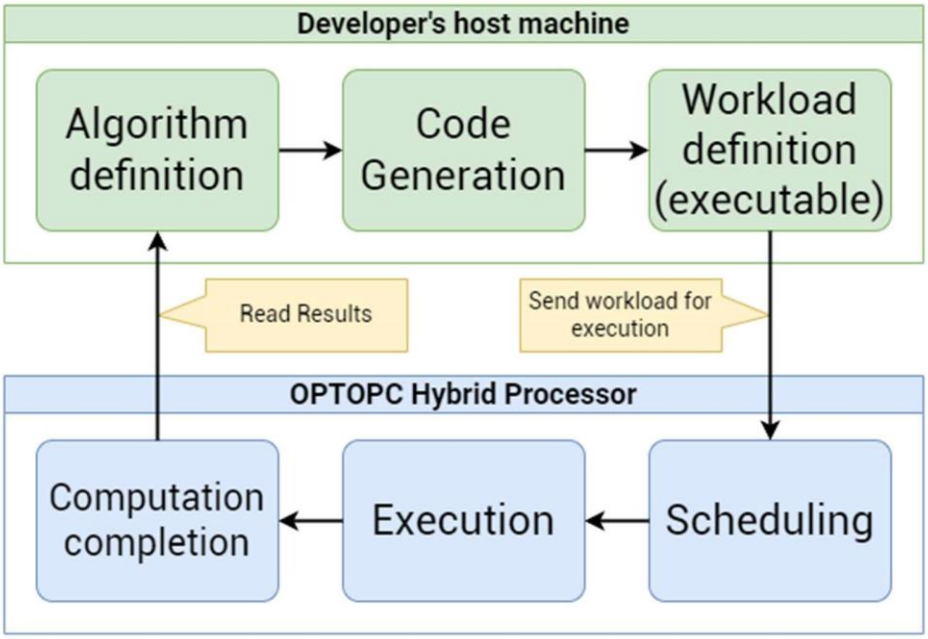


Figure 3. Development and execution workflow.

Final users and developers are not required to know the details on how atomic and composite operations are defined. They will simply code in a high-level language, and they will use the OPTOPC tools in order to generate an executable workflow as described in Figure 3.

3.2. Scheduling the Workflows

A dedicated CPU oversees the scheduling of the execution of the workflows.

The scheduler is the component responsible for analyzing composite operations and deciding which atomic operations to load, using the control logic, into the optical matrix based on their dependencies.

Effectively harnessing the potential parallel execution of the operations requires meticulous task scheduling. This process involves distributing and executing operations across the optical matrix in a way that optimizes resource utilization, minimizes execution time, and maximizes system throughput.

One of the primary challenges in task scheduling is managing task dependencies and data dependencies. Tasks often rely on the completion of other tasks or the availability of specific data. Workloads are designed in order to maximize tasks that can be executed in parallel, optimize task granularity, and manage fault tolerance.

Internal Usage Only

Moreover, the dedicated CPU provides efficient resource allocation in order to avoid bottlenecks and maximize performance and guarantees synchronization and communication between operations.

3.3. The Control Logic

The control logic component is responsible for loading the information of the operations in the optical matrix and reading the results from the light-detector sensor array.

It is known that the light ranges on a vast number of frequencies. Therefore, every light ray can carry a significant amount of information. The sensibility of the light emitting devices and light detectors determines how many values can be stored in a single pixel.

The data must be encoded into its pixel representation before loading into the optical matrix. Conversely, the detected pixels must be decoded to read back the results.

The control logic component performs the conversion, and it is responsible for enabling and disabling the proper pixels by “drawing” the information on the light emitting devices.

There are many ways for encoding and decoding data depending on the amount of information carried by every pixel and the type of operation to be performed.

Starting with the simplest approach, we can assume that every pixel is used for representing binary information: “light detected” or “light not detected.” Unfortunately, with this simple approach representing the operations to be performed is not straightforward.

Dual rail is a slightly better encoding, requiring four pixels for representing each bit, it can be used for performing a complete set of operations.

Depending on the specific application, several other encodings are available or can be devised. E.g., a far more complex setting can be achieved if we could represent and detect 256 different values from the same light ray, then we could represent an entire byte on a single pixel.

4 he Opt ca Matr x

The OPTOPC optical matrix comprises an LED-based light-emitting layer (an OLED display), and light detectors array (a CMOS sensor). Many solutions for implementing the optical matrix exploit different layers assembled in many different configurations.

Internal Usage Only

At its core, an OLED display is a type of display technology where each pixel in the screen is made up of organic compounds that emit light when an electric current is passed through them. This is the fundamental difference compared to LCDs, which require a backlight to illuminate liquid crystals that then filter light to create the image.

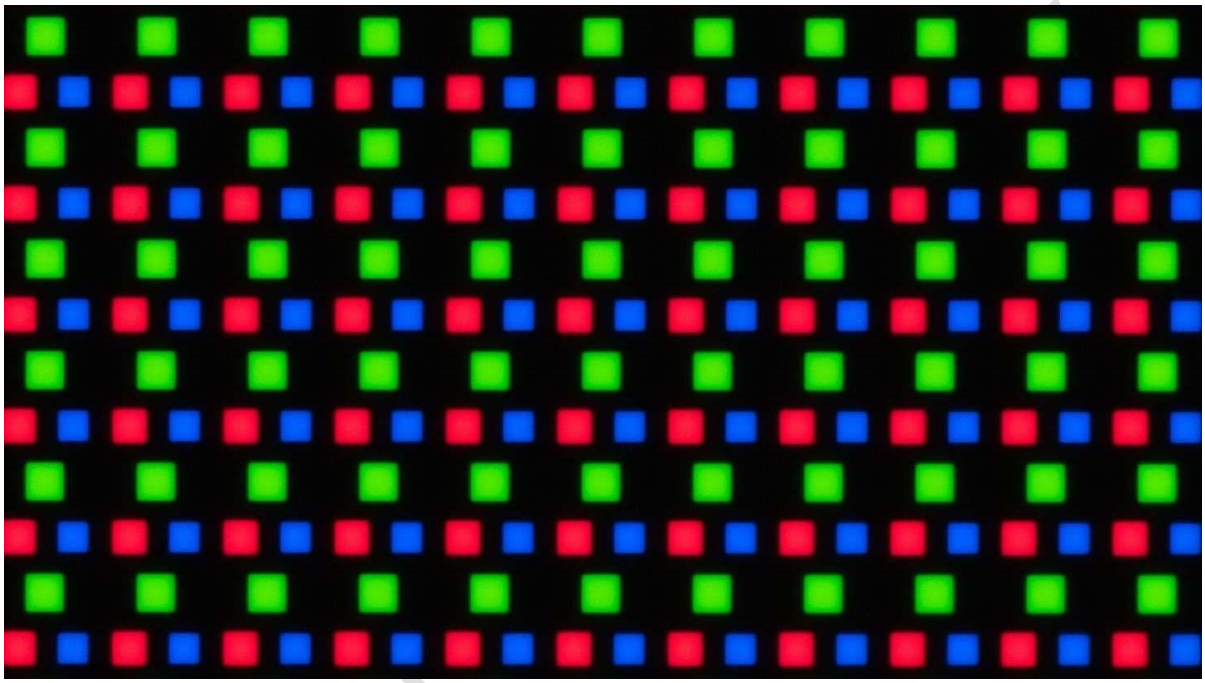


Figure 4. The subpixel structure of a Samsung OLED display.

In LED displays, as also shown in Figure 4, subpixels are typically created using individual LED chips that emit red, green, and blue light. The individual LEDs of different colors form a single pixel and allow color mixing.

Each subpixel in an LED display is controlled by driver Integrated Circuits (ICs). These chips regulate the current flowing through each red, green, and blue LED (or LED chip within an SMD package) allowing the regulation of each pixel color.

A Complementary Metal-Oxide-Semiconductor (CMOS) sensor is a semiconductor device that acts as the "eye" of a digital camera or imaging system. Its primary function is to convert light into electrical signals. These electrical signals are then processed to create a digital image.

A CMOS sensor is not just a single light detector, but rather a massive array of pixels arranged in a grid. Each pixel is a miniature photo-detector capable of sensing light intensity. Think of it as a grid of tiny light-

Internal

sensitive buckets. CMOS are fundamentally grayscale sensors. They measure the intensity of light hitting them, but they do not inherently “see” color. To capture color images with a single sensor, we need a way to separate and record the red, green, and blue components of the light.

Instead of needing three separate sensors (one for each color), we use one sensor and selectively filter the light reaching different pixels to make them primarily sensitive to red, green, or blue light.

The Bayer pattern, also known as the Bayer filter array (Color Filter Array - CFA), is a mosaic of tiny color filters arranged over the pixels of an image sensor. Specifically, it is a repeating pattern of red (R), green (G), and blue (B) filters. This pattern is placed directly on top of the sensor's pixel grid. It is a filter, not part of the sensor itself, but positioned precisely above it during manufacturing.

The three primary colors of light: Red, Green, and Blue are chosen because they approximate how the human eye perceives color (trichromatic vision).

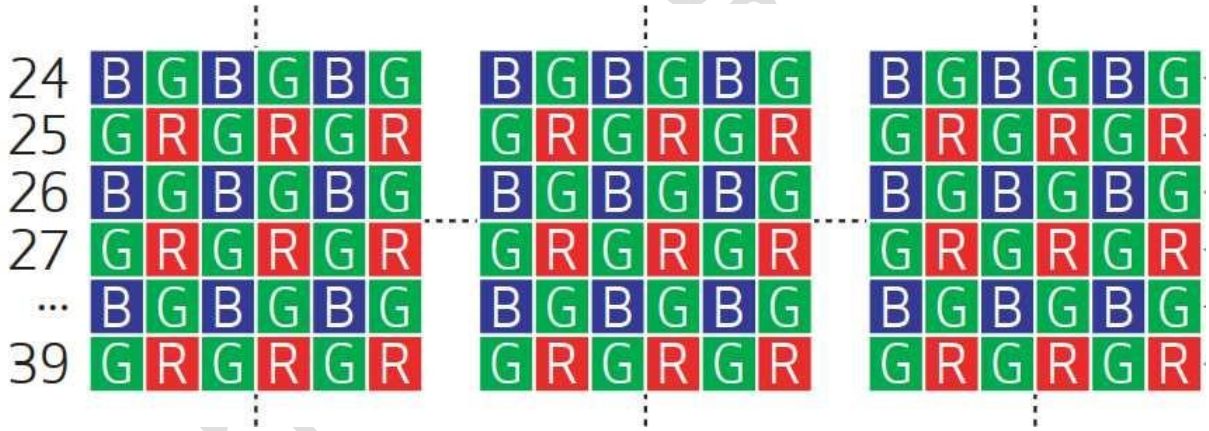


Figure 5. One of the most common CFA patterns, organized in a 2x2 repeating unit.

As also shown in Figure 5, the most common Bayer pattern is a 2x2 repeating unit.

Within this 2x2 block, there are:

- One Blue (B) filter
- Two Green (G) filters
- One Red (R) filter

Notice there are twice as many green filters as red or blue. This is intentional and important for a few reasons:

Internal Use Only

- Human Eye Sensitivity: The human eye is most sensitive to green light. Having more green sensors helps capture more luminance (brightness) information, which is crucial for perceived image sharpness and detail.
- Luminance Information: Green light contributes most significantly to the overall brightness (luminance) of an image.
- Noise Reduction: Having more green samples helps reduce noise in the luminance channel, which is generally more noticeable than noise in the color channels.

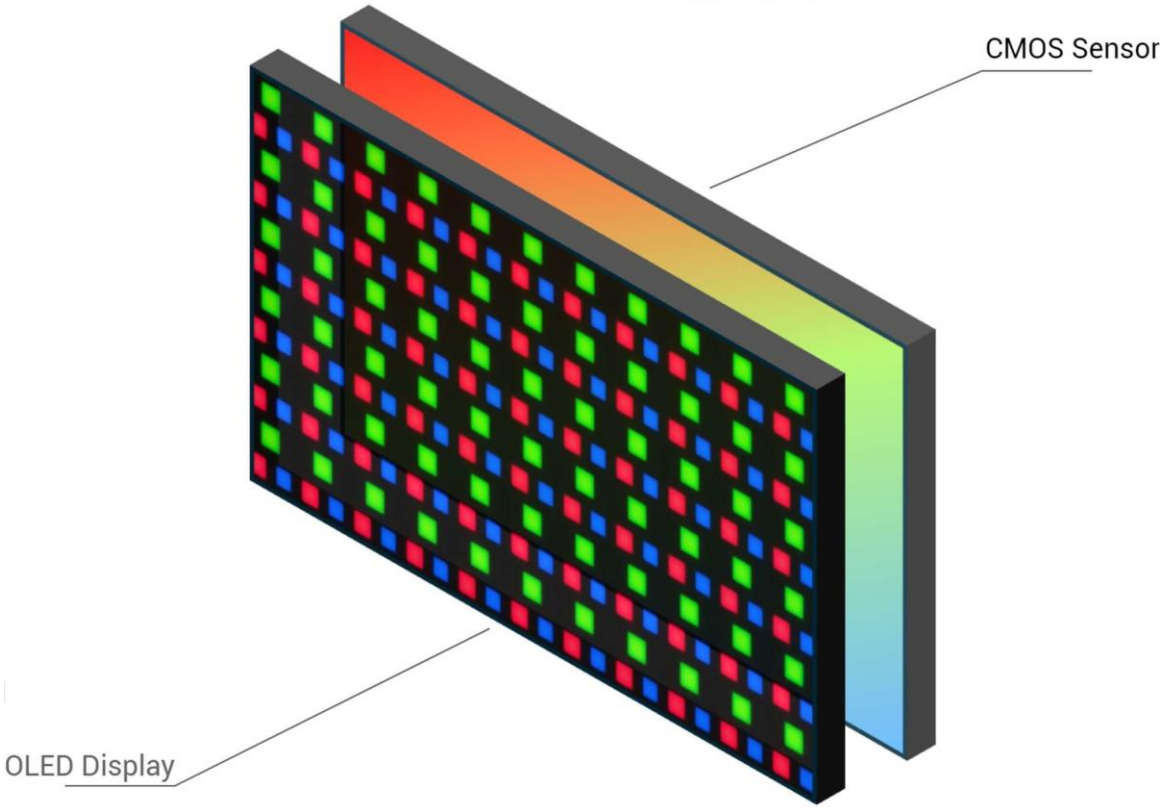


Figure 6. The proposed configuration for the optical matrix.

The current implementation of the OPTOPC optical matrix, as also shown in Figure 6, is based on an OLED display tightly coupled with a CMOS based sensor.

The OLED display is used for encoding operands and operations, while the CMOS sensor is used for computing the results of the intended operations.

The next paragraphs describe the intended behavior of the optical matrix when performing computations.

4.1. Example of Computation Operations

As a simple example of computation operations, let us outline the process of computing the AND, OR, XOR bitwise logical operations between two numbers using the OPTOPC optical matrix.

Let us start with the representation of the operands in the OLED display and let us choose two random 16-binary numbers, e.g., a=39818 and b=53086. The 16-bit size was chosen in this example for ease of representation.

The main idea of this procedure consists of encoding both the operands a and b in the OLED display, in the same display line and pixel positions, however using different colors for representing the different numbers. The binary representation of each number is obtained by activating or deactivating the corresponding color for each pixel used.

Taking in mind the subpixel pattern shown in Figure 4, in the following example we will use the full intensity of the green color for representing operand a, and the full intensity of the red color for representing b. For the moment, the blue color is not used.

The decimal value 39818 of the operand a corresponds to the binary value 0b1001 1011 1000 1010.

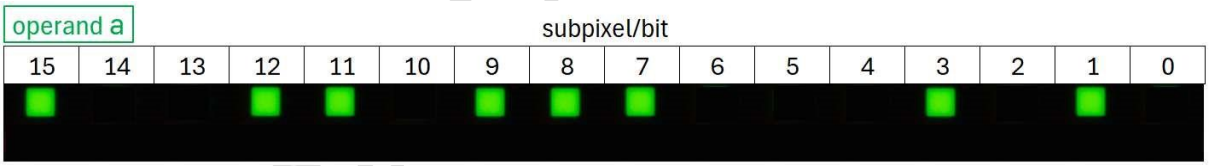


Figure 7. Binary representation of operand a, using the green color.

The Figure 7 shows the binary representation of a using 16 pixels of one line of the OLED display.

At the same way, the decimal value 53086 of b corresponds to the binary value 0b1100 1111 0101 1110.

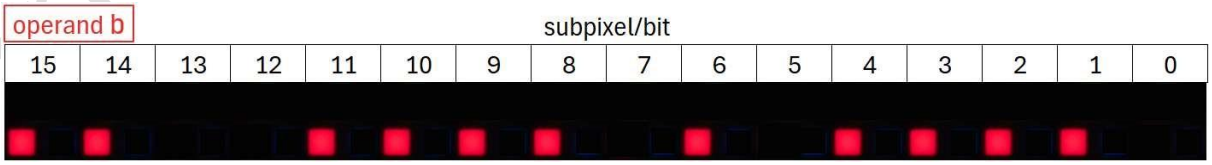


Figure 8. Binary representation of operand b, using the red color.

The Figure 8 shows the binary representation of b using 16 pixels of one line of the OLED display.

Internal Usage Only

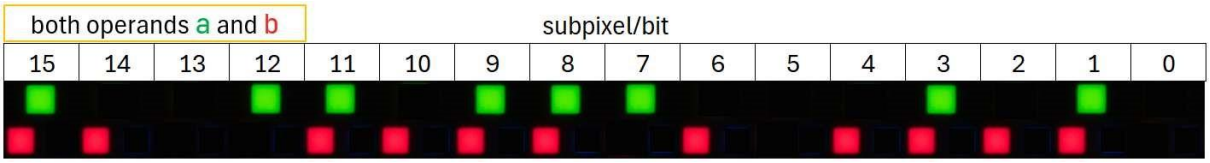


Figure 9. Binary representation of both the operands a and b, using green and red colors.

Putting all together, Figure 9 shows the binary representation of both the operands a and b using different colors subpixels in the same pixel array.

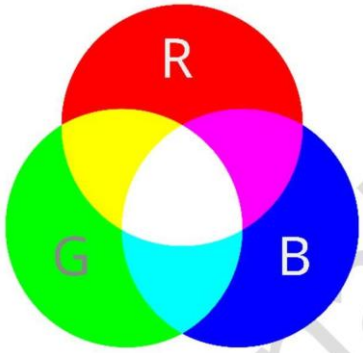


Figure 10. A diagram demonstrating additive color with RGB.

The RGB color model, as also shown in Figure 10, is additive in the sense that if light beams of differing color (frequency) are superposed in space their light spectra add up, wavelength for wavelength, to make up a resulting, total spectrum.



Figure 11. The resulting RGB pixel configuration for the OLED display.

Therefore, the Figure 11 shows the resulting light emitted by the OLED display given the specified configuration of the two operands.

Let us now describe the sensing component behavior. As we have established, a CMOS sensor is a semiconductor-based image sensor. Its primary role is to convert light into electrical signals, and it is built using an array (millions) of light-sensitive elements called pixels, arranged in a grid.

Internal Usage Only

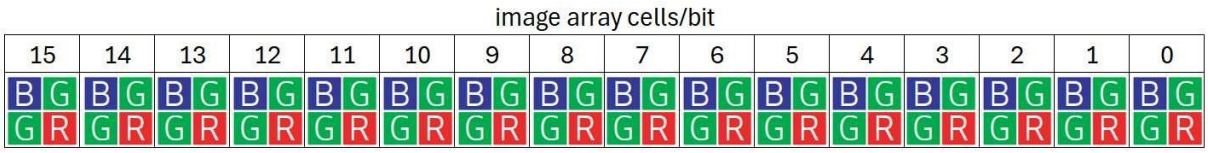


Figure 12. Details of the CMOS sensor pixel organization.

Figure 5 and Figure 12 show the organization of the pixel array with a specific CFA.

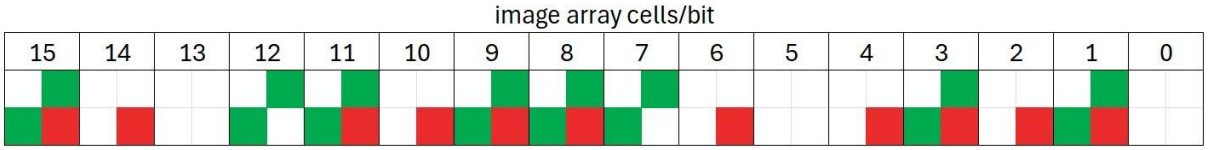


Figure 13. The CMOS detection results from a logical point of view.

Figure 13 shows the detection results from the CMOS device. Recalling the Figure 11, the bit 0 corresponded to the black color (no light emitted), and the CMOS correctly does not show any value. For the bit 1, the original color was yellow, and therefore the CMOS detected light presence in the grid pixels corresponding to green and red positions. As shown in Figure 13, the process is repeated for all the other pixels.

A	B	A AND B	A OR B	A XOR B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Table 1 Output comparison of 2-input logic operations.

The results of the logical AND, OR, XOR operations are determined using the truth tables presented in Table 1.

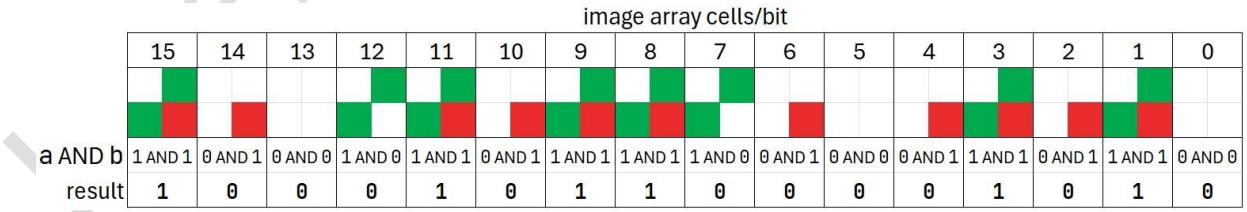


Figure 14. Determining the result for the “a AND b” operation.

Given that a=0d39818 (0b1001 1011 1000 1010) and b=0d11009 (0b0010 1011 0000 0001), Figure 14 gives the details for determining the result of the c=a AND b operation. In this case c=0d35594 (0b1000 1011 0000 1010).

Internal Use Only

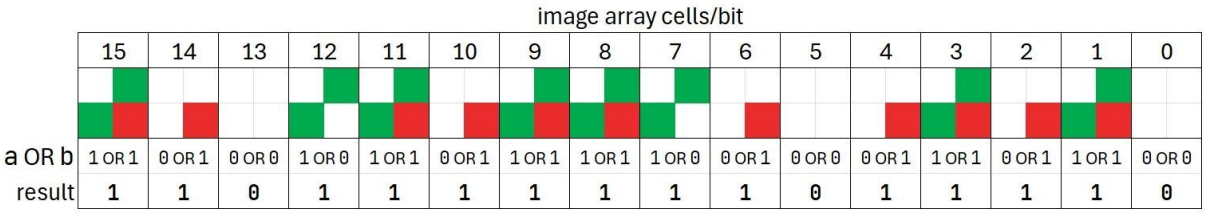


Figure 15. Determining the result for the "a OR b" operation.

Figure 15 gives the details for determining the result of the c=a OR b operation. In this case c=0d57310 (0b1101 1111 1101 1110).

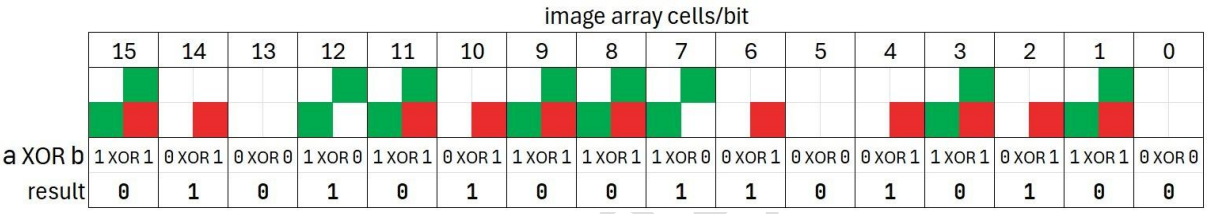


Figure 16. Determining the result for the "a XOR b" operation.

Figure 16 gives the details for determining the result of the c=a XOR b operation. In this case c=0d21716 (0b0101 0100 1101 0100).

4.2. Using Colors as a Control Line

The OPTOPC optical matrix exploits another important idea: using the colors for encoding control information (instruction set) regarding the operations to be computed.

Traditionally, from the early days of computing with limited instruction sets and varied word sizes dictated by hardware, the evolution of opcodes and architecture bit-widths dramatically accelerated with the microprocessor revolution.

The 8-bit era brought a standardization and expansion of opcodes, enabling more complex operations and addressing modes, but still constrained by cost and hardware limitations.

The subsequent move to 16-bit architectures, driven by the need for larger address spaces and more powerful software, witnessed a significant opcode explosion, incorporating richer data types, improved addressing modes, and marking the beginning of Complex Instruction Set Computing (CISC) trends aimed at providing powerful single instructions.

Internal Usage Only

The 32-bit era cemented the dominance of general-purpose computing, with architectures further diverging into CISC (like x86, continuing opcode complexity and backward compatibility) and RISC (prioritizing simpler, fixed-length instructions and compiler optimization).

The transition to 64-bit architectures was driven by the demand for even larger address spaces and performance, leading to further opcode extensions and specializations, particularly for multimedia, vector processing, and emerging domains like AI.

Despite the significant evolution in bit-widths from 8 to 64 bits and the accompanying changes in instruction set complexity and architectural paradigms, the fundamental concept of opcodes remains a constant and unifying element across all these architectures. Whether in the constrained 8-bit world or the expansive 64-bit landscape, opcodes serve as the essential machine-level encoding for instructions, dictating the operations a processor must perform. Think of opcodes as direct commands. When you want a computer to perform a task, whether it's adding two numbers, moving data in memory, or making a decision based on a condition, you need to give it precise instructions.

OpName	OpCode	Format	Notes
SEQ	00000	SEQ Rd Op1 Op2	(SetIfEq) Set register Rd to 1 if the values in Op1 and Op2 are equal, else set to 0.
ADD	00001	ADD Rd Op1 Op2	Add registers Op1 and Op2 and store the result in Rd.
SGT	00010	SGT Rd Op1 Op2	(SetIfGtr) Set register Rd to 1 if Operand1 > Operand2, else set to 0.
ADDI	00011	ADDI Rd Op #	Store in register Rd the value Op + the 5-bit signextended immediate.
OpName	OpCode	Format	Notes
SUB	00100	SUB Rd Op1 Op2	Compute Op1 - Op2 and store the result in Rd.
SLT	00101	SLT Rd Op1 Op2	(SetIfLes) Set register Rd to 1 if Op1 < Op2, else set to 0.
OR	00111	OR Rd Op1 Op2	Store in register Rd the bitwise OR of Op1 and Op2.
ORI	01000	ORI Rd Op #	Store in register Rd the bitwise OR of Op and the 5bit zero-extended immediate.

Internal Use Only

AND	01001	AND Rd Op1 Op2	Store in register Rd the bitwise AND of Op1 and Op2.
ANDI	01010	ANDI Rd Op #	Store in register Rd the bitwise AND of Op and the 5bit zero-extended immediate.
XOR	01011	XOR Rd Op1 Op2	Store in register Rd the bitwise XOR of Op1 and Op2.
XNOR	01100	XNOR Rd Op1 Op2	Store in register Rd the bitwise XNOR of Op1 and Op2.
NOT	01101	NOT Rd Op	Store in register Rd the bitwise complement Op.
MUL	10010	MUL Rd Op1 Op2	Multiply the lower 8 bits of Op1 and Op2 and store the result in Rd.
NOP	11100	NOP	No Operation. Do nothing

Table 2 Example instruction set for a 16-bit RISC microprocessor.

Table 2 gives a list of example instruction and opcodes for a 16-bit RISC microprocessor.

Given that, as described in Paragraph 4.1, we are using green and red colors for representing the operands in the computations, in the following examples we will use the blue color for representing the operation to be performed on the operands.

Let us use the opcodes described in Table 2 as a reference, therefore the AND operation is represented by the 16-bit sequence 0b0000 0000 0000 1001. In the same way, OR is 0b0000 0000 0000 0111 and XOR is 0b0000 0000 0000 1011.

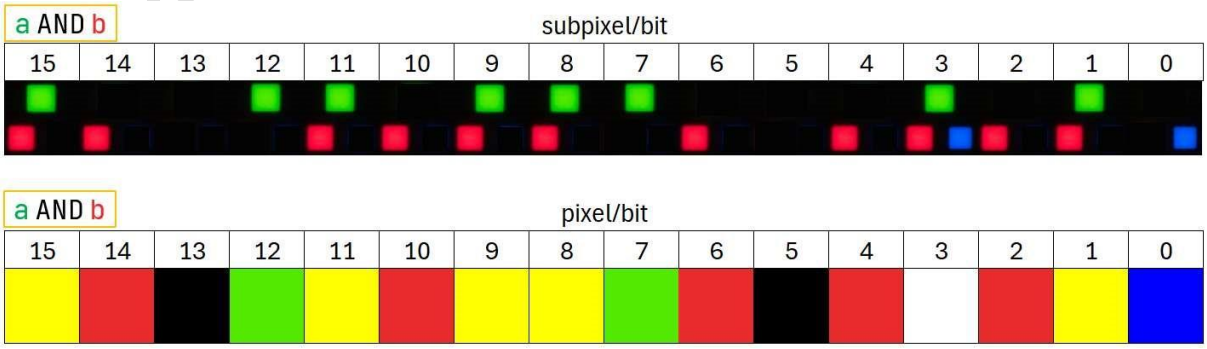


Figure 17. Encoding the AND operation with the blue color.

Internal Usage Only

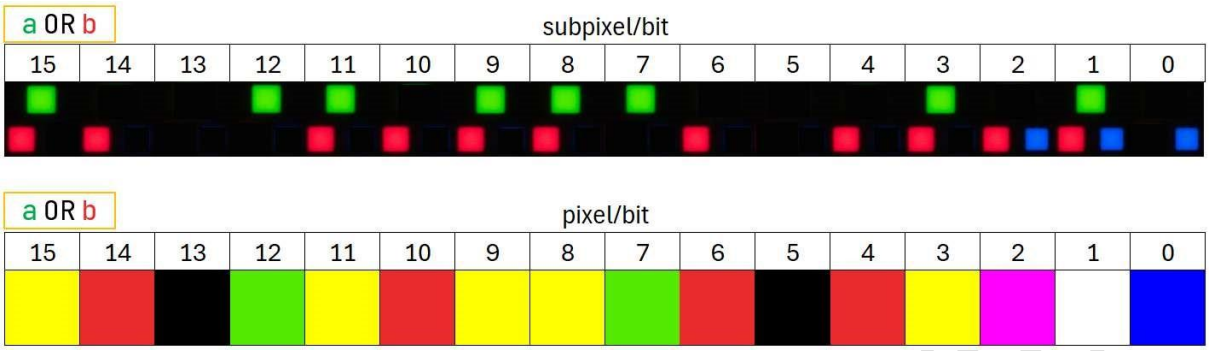


Figure 18. Encoding the OR operation with the blue color.

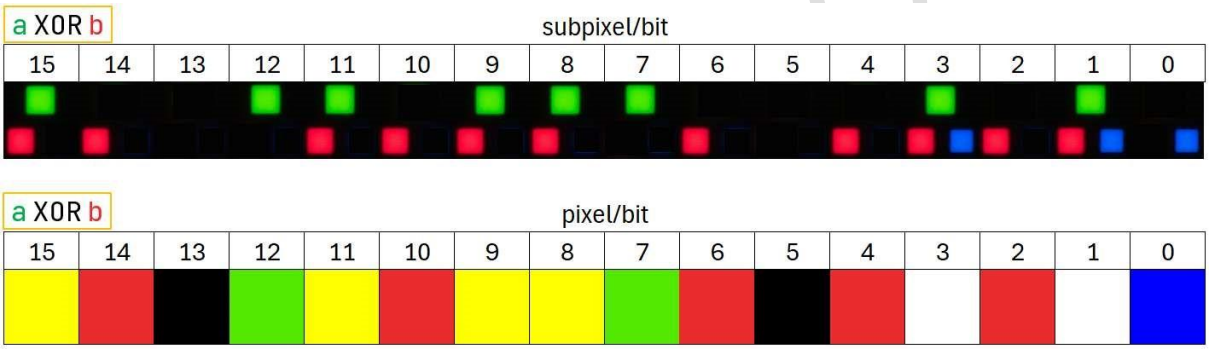


Figure 19. Encoding the XOR operation with the blue color.

Figure 17, Figure 18, and Figure 19 represent the encoding of the operands and the operations AND, OR, and XOR respectively.

The sensing component behavior remains the same as what describe in Paragraph 4.1, enabling to decode the operation together with operands.

5 Conclusion

This document has laid the groundwork for understanding the OPTOPC prototypes' data flow and components.

The OPTOPC Hybrid-Optoelectronic Processors have been described, giving an architectural overview and details on the main components proposed.

By elucidating the behavior of the optical matrix, we have gained valuable insights into its operational mechanisms, providing a detailed explanation of a computation operation.

Internal Only

6. Document Revision History

Document version	Changes
2025-02-11	Initial release.

OPTOPC SA

Piazza Indipendenza 1, 6830 Chiasso, Svizzera

Tel +41 91 682 0330 info@optopc.com

www.optopc.com

Internal Usage Only