*electronics*

MDPI

*Article*

# FPGA Implementation of Image Ordering and Packing Algorithm for TuMag Camera

**Eduardo Magdaleno [1,*], Manuel Rodríguez Valido [1], David Hernández [2,3], María Balaguer [4], Basilio Ruiz Cobo [2,3] and David Díaz [1]**

[1] Department of Industrial Engineering, University of La Laguna, 38203 San Cristóbal de La Laguna, Spain; mrvalido@ull.edu.es (M.R.V.); ddiazmar@ull.edu.es (D.D.)
[2] Institute of Astrophysics of Canary Islands, 38205 San Cristóbal de La Laguna, Spain; dhdez@iac.es (D.H.); brc@iac.es (B.R.C.)
[3] Department of Astrophysics, University of La Laguna, 38203 San Cristóbal de La Laguna, Spain
[4] Institute of Astrophysics of Andalusia, 18008 Granada, Spain; balaguer@iaa.es
* Correspondence: emagcas@ull.edu.es

**Abstract:** The TuMag instrument is a Tunable Magnetograph that has been designed to measure the magnetic field of the sun. This instrument and others will be connected to a telescope that will be sent into the stratosphere using a balloon for an uninterrupted observation of the sun for four days in the summer of 2022. The TuMag camera is a new development for implementing the image detector of the instrument. It is based on the GPIXEL GSENSE400-BSI scientific CMOS image sensor and an FPGA device in charge of controlling the image sensor, configuring it and grabbing images. FPGA device consists of an array of Configurable Logic Blocks. However, the sensor does not supply the image data in a row-by-column format. This task has to be done in the FPGA that controls the sensor because the frame grabber has a significant workload with the control of all the instruments, the telescope, the refrigeration, the navigation, and so on. This work describes the FPGA implementation of Image Ordering and Packing algorithm for TuMag Camera concerning the real-time ordering of the images before grabbing and sending to the Data Processing Unit.

**Keywords:** FPGA; firmware; TuMag camera; Sunrise 3 mission; image processing; VHDL



**Citation:** Magdaleno, E.; Rodríguez Valido, M.; Hernández, D.; Balaguer, M.; Ruiz Cobo, B.; Díaz, D. FPGA Implementation of Image Ordering and Packing Algorithm for TuMag Camera. *Electronics* **2021**, *10*, 1706. https://doi.org/10.3390/electronics10141706

Academic Editors: Akash Kumar and Manohar Das

Received: 27 May 2021
Accepted: 14 July 2021
Published: 16 July 2021



## 1. Introduction

Sunrise 3 is the third missions where a telescope will be launched into the stratosphere on a balloon from the Arctic Circle, and reaching an altitude of about forty kilometers, which avoids the day and night cycles and the image degradation caused by the terrestrial atmosphere. On this basis, it observes the Sun continuously for a few days [1]. The IMaX/Sunrise project was originally approved in 2002 by the National Program as a strategic step towards a technological demonstrator for the Solar Orbiter magnetograph (PHI, Polarimetric and Helioseismic Imager) [2]. The IMax (Imaging Magnetograph eXperiment) was one of the post-focus instruments of the one-meter solar telescope aboard the first Sunrise mission in 2009 [3]. This mission studied the solar magnetism in a period of minimum solar activity [4,5].

The first Sunrise mission obtained excellent results and led to a second mission that took place in 2013 in a period of maximum solar activity [6]. The success of the previous mission in 2009 meant that the design of the 2013 mission was largely unchanged. The gondola and the telescope were the same; the IMaX on the 2013 flight was also very similar to the version flown on Sunrise I, although a number of smaller changes and updates had been made. The parts replaced included the Field Programmable Gate Array (FPGA) in the proximity electronics and other components (filter system, retarders and so on). Most of these parts were replaced by nearly identical ones. Only the filter system of one of the instruments on board was modified [7]. The scientific results of the Sunrise

missions can be consulted at [6,7]. This second science flight allowed this balloon-borne solar observatory to obtain the first seeing-free observations of an active region close to the diffraction limit of the 1 m diameter telescope. These data are rich in information about a variety of solar phenomena at very high spatial resolution and at wavelengths that partly cannot be accessed from the ground [7–11].

Rather than being a transient thought for a given phase of solar activity, Sunrise 3 is meant to observe the evolution of the magnetically coupled solar atmosphere at high spatial resolution. A new Tunable Magnetograph (TuMag) has been implemented for this mission. The TuMag instrument is a new, and improved, version of the IMaX instrument [12]. Besides changing the cameras that were not very efficient and refurbish most of the optics and opto-mechanics accordingly, a major conceptual change in the optics has been realized due to the tunable capability of the instrument filter. TuMag provides $50 \times 50$ arcmin$^2$ images of the Sun in various narrow bands (~8 pm wide). From these images, the information of the four polarization states can be extracted [13–15].

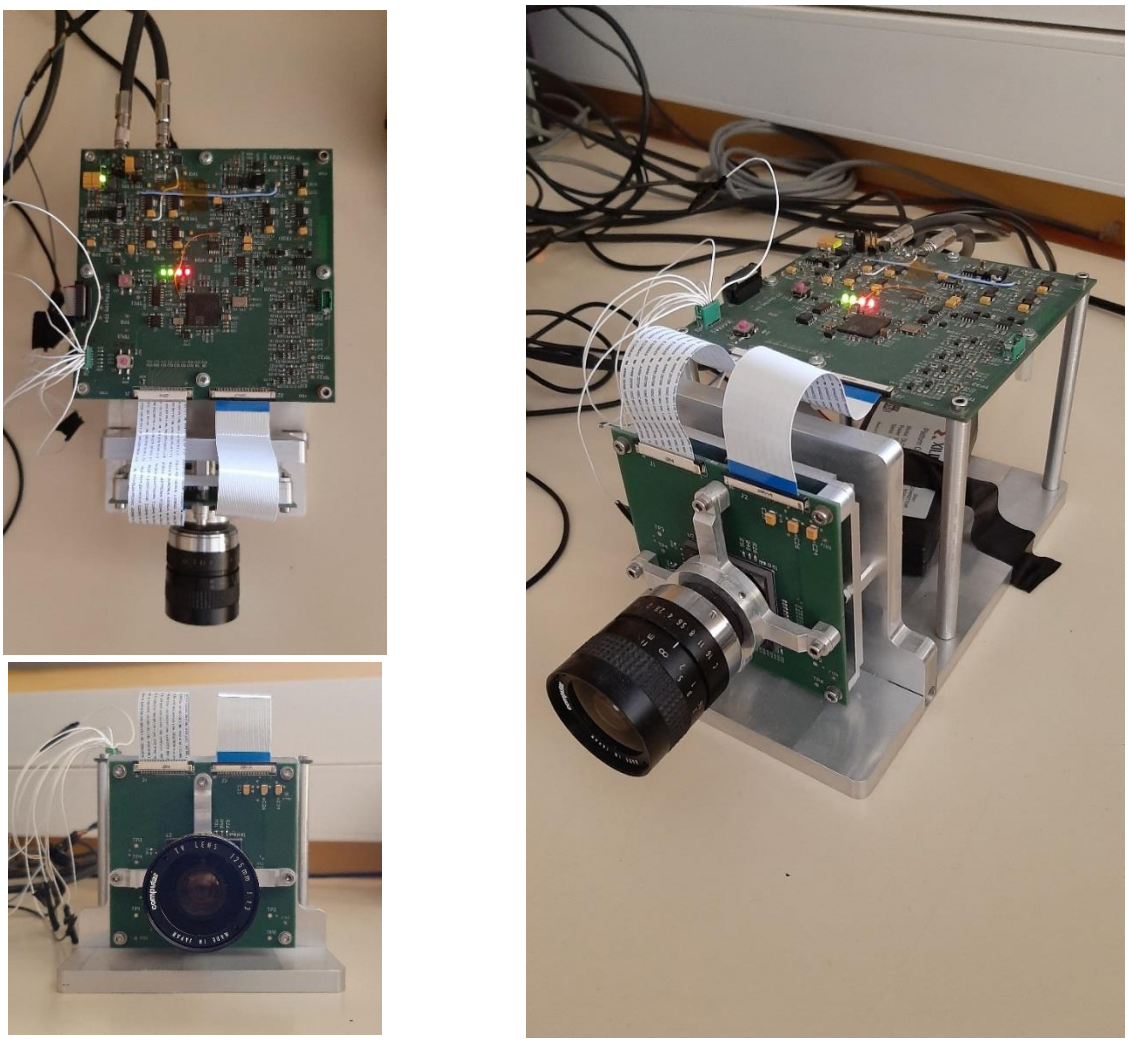
The electronic unit of TuMag instrument has been completely redesigned respect of the old IMaX. On the one hand, most of the former components were damaged after the second flight. On the other hand, the big old unit is inefficiently old-fashioned. New technologies have been easy to incorporate into a much smaller box. Therefore, current technological components have been used that will result in a much smaller system, leaving more physical space for the other instruments on board.

In relation to the TuMag camera sensor, hardware has been developed that includes two Printed Circuit Boards (PCB). The first of them contains a Complementary Metal-Oxide Semiconductor (CMOS) image sensor, the GPIXEL GSENSE400-BSI [15]. The second one includes an Artix-7 XC7A50T-2CSG325C FPGA that is responsible for controlling the sensor and communicating with the Data Processing Unit (DPU) of the instrument through a CoaxPress interface [16,17]. Artix-7 FPGA device consists of an array of Configurable Logic Blocks (CLBs) which are composed of two slices. These elements are connected to other similar blocks via programmable interconnects and switch matrices. Inside of a slice there are four Lookup Tables (6-LUT) capable of implementing a logic function up to six inputs, a small memory (named distributed-memory) or a shift register. The Artix-7 XC7A50T-2CSG325C FPGA has 32,600 6-LUT and 16,300 slice registers. The FPGA implementation makes the designed algorithm, flexible, customizable, reconfigurable or reprogrammable with advantages of well-customized, integration, accessibility and expandability. The system can be resized according to its needs taking advantages of the VHDL configurability. Figure 1 shows pictures of the first implemented prototype of the TuMag camera. The sensor PCB is located vertically and the FPGA PCB is located horizontally.

The GSENSE400 is a 4-megapixel ($2048 \times 2048$) resolution CMOS image sensor with 11 μm photodiode pixels and it has been chosen to incorporate it into the detection system as it has a set of characteristics that meet the requirements of the scientific commissions of the Sunrise 3 mission, such as extremely low reading noise, high sensitivity, and high dynamic range features [16].

The sensor has a 12-bit AD converter, a temperature sensor, a PLL and an SPI interface for control tasks. In order to configure the detector and start it up properly, we have to write to memory banks of the detector through this SPI interface.

The sensor has eight differential channels, which provide 12-bit pixel data at 300 MHz, in order to send the image to the FPGA in serial format. The rows of the CMOS sensor are read or reset in time slots of 513-pixel clock cycles (25 MHz) using control signals that the Artix-7 FPGA must send to the sensor. These channels have to be calibrated so that the data can be correctly read. This task is performed by the FPGA [18].



(**a**) (**b**)

**Figure 1.** Pictures of the first prototype of the TuMag instrument camera: (**a**) Frontal and overhead views; (**b**) Isometric view.

In addition, the format of the reading data is different of the row-by-column format, so an image ordering must be implemented in the FPGA firmware. This task has to be done in the FPGA that controls the sensor as the DPU has a high workload with the control of all the instruments, the telescope, the refrigeration, the navigation, communications and so on. Furthermore, the acquisition of the images must be carried out in very precise moments of time using a hardware trigger. For this reason, the time delays that any operating system or software program can cause should be avoided. The GPIXEL sensor supplies image data in eight-channel format. Therefore, we have implemented in firmware the image ordering. Due to the eight-channel format, the sensor only allows the acquired images to be cropped in height (rows), and not in width (columns). Therefore, the Region of Interest (RoI) that can be configured into the sensor is limited. The implemented VHDL module to order the image considers this limitation, and cuts the received data into columns as well, so that only the RoI is sent to the DPU. This prevents the CoaXPress communication channel from being overloaded with unnecessary data.

This work describes the FPGA implementation of Image Ordering and Packing algorithm for TuMag Camera concerning the real-time ordering of the images before grabbing and sending to the DPU. FPGA devices have proven to be ideal for implementing real-time algorithms for images coming from cameras for astrophysical applications [19–26]. In this case, the improved VHDL module allows cropping of the image in rows and columns, and

is included in the driver that controls the TuMag camera. The TuMag instrument is part of the Sunrise 3 mission. The flight is scheduled for the northern summer of 2022.

The rest of the present work is organized as follows: The second section briefly describes the image sensor configuration, communication and operation. The third section details the firmware implemented in the FPGA with an emphasis on image ordering detailed in the fourth section. Then, in section five, we present the results and conclusions. Finally, the future work is presented.

## 2. Main Features of the Image Sensor

The GSENSE400 sensor is a 4-megapixel (2048 × 2048) resolution CMOS image sensor with 11 µm photodiode pixels. The sensor has an extremely low reading noise [16]. It has two operation modes: In STD mode, the sensor works at 48 frames per second and in HDR mode, the sensor is optimized for high dynamic range applications and it works at 24 frames per second. The high sensitivity, low read noise, and high dynamic range feature makes it perfect for a variety of scientific applications, such as the one at hand.

The sensor has a 12-bit AD converter, a temperature sensor, a PLL and an SPI interface for control tasks. The interface with the main ports of GSENSE400 sensor is depicted in Figure 2. In order to configure the detector and start it up properly, we have to write to memory banks of the detector through an SPI interface (*spi* signals). The sensor has eight differential channels that provide 12-bit pixel data at 300 MHz in order to send the image to the FPGA in serial format (*sensor_data_ser* signals). The synchronism of these signals is supplied using a 25 MHz pixel clock (*clk_pix*). User can calibrate these signals using a training word that is written in the register banks using the SPI interface. When *train* signal is set to high, GSENSE400 sends the training word continuously through *sensor_data_ser* signals. Finally, the firmware in FPGA supplies the timing control signals and the address of pixel rows directly to the sensor (*pixel_read_timing* and *decoder*). The values of these signals in time are described in [16,19].


**Figure 2.** Signal from/to FPGA used to drive the image sensor.

The image sensor has two operation modes: In Standard (STD) mode, the sensor works at 48 frames per second and in high-dynamic range (HDR) mode, the sensor is optimized for high dynamic range applications and it works at 24 frames per second. In both modes, the rows of the CMOS sensor are read or reset in time slots of 513-pixel clock cycles (at 25 MHz) through control signals that the Artix-7 FPGA must send. In HDR mode, a time slot consists of 2 phases: One row read phase and one row reset phase. Using this mode, two readings are obtained for each direction, one in high gain and the other in low gain. This mode is not configured.

In STD mode, a time slot consists of 4 phases: Single row read phase, single row reset phase, and read and reset phases of subsequent rows, respectively. Figure 3 shows how rows M and following are read and rows N and following are reset in two time slots [16]. As mentioned previously, the implemented driver in FPGA sends decoder and *pixel_read_timing* values for a right STD mode operation.

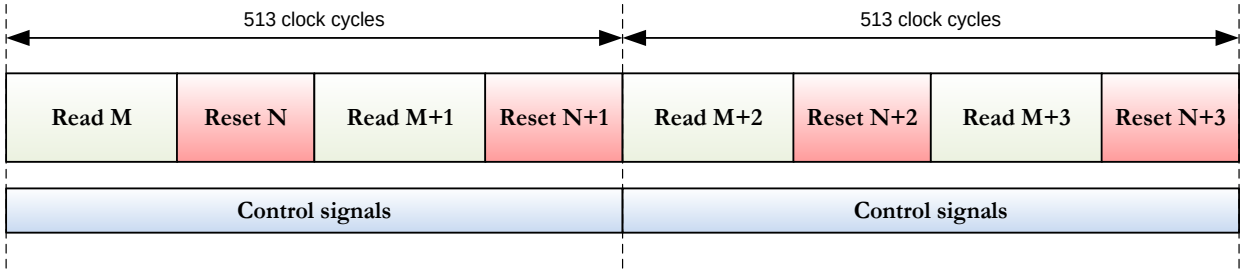



**Figure 3.** Relationship between decoder signal (image row address) and Control signals in STD mode. Red slot indicates the row to be reset and green slot indicates the row to be read.

## 3. TuMag Camera Firmware Architecture

FPGA PBC contains the XC7A50T-2CSG325C Artix-7 FPGA that configures and communicates with the sensor. This one also features a CoaXPress interface for communication with the host/DPU system or with the Frame Grabber in the prototype [18,27,28]. An overview of the camera FW architecture is depicted in Figure 4.

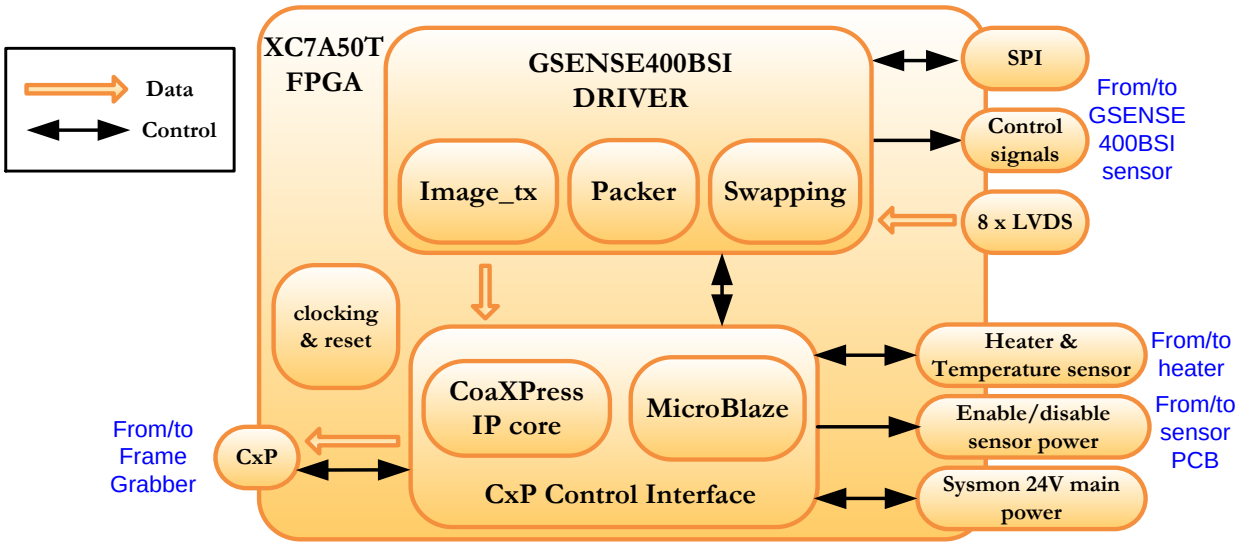



**Figure 4.** Camera FW device architecture.

Two main blocks can be distinguished, namely, GSENSE400-BSI driver and CoaXPress (CxP) Control Interface. The main function of CxP Control Interface module is the control of the CoaXPress device IP core, which implements the communication functionalities between the GSENSE400-BSI sensor and the Frame Grabber. The firmware implemented in the FPGA is in charge of controlling the image sensor, configuring it and grabbing images from it through a 3.125 Gbps CoaXPress interface [18,29,30]. The camera is triggered through the same interface. For such a purpose, the architecture is based on an embedded Microblaze soft processor [31]. This soft-core also manages a system monitor of the main power (24 V), a heater and controls the sensor on/off power.

The FPGA firmware allows the modification of several parameters as the Region of Interest (ROI), the exposure time, the number of frames to be acquired, the sensor gain and the black level offset to obtain the camera required functionality. The host sends these parameters to MicroBlaze through control signals of CxP interface and then MicroBlaze writes to a memory bank in the GSENSE driver. Finally, most of these parameters require writing to the sensor register bank. This is done through the SPI interface in Figure 4.

Once the sensor is configured and calibrated, the driver waits for a read command. When this happens, the driver sends to the sensor the control signals and the read and reset rows (see Figure 2) according to Figure 3. Control signals are described in [16]. The driver

stores these values in a 513 × 19 memory and using a simple counter sends values correctly to the sensor. Decoder signal in Figure 2 are calculated using row values of RoI (the beginning row and the size in rows, named *reg_window_row_start* and *reg_window_row_length*) and the exposure time (named *reg_integration*).

Figure 5 shows the signals that the driver sends to the sensor once it has been calibrated. When the driver receives a read image command (*frame_req* set to high), it generates the reset and read addresses and the internal signal named *action*, which decodes if a reset or a read is made of an odd or even row according Table 1. The driver composes decoder signal with these internal values and send it with *pixel_read_timing* signal through the corresponding image sensor ports (see Figure 2). The values, 4000 and 4001 of the decoder are dummy addresses. Therefore, in the first time slot no read is realized and there are only rows reset operation. In the second time slot, the driver resets rows number 2 and 3 and reads rows number 0 and 1. The internal *sync* signal (in red) specifies the end of a time slot and it is used to preserve synchronism with the data that the driver will receive from the sensor.

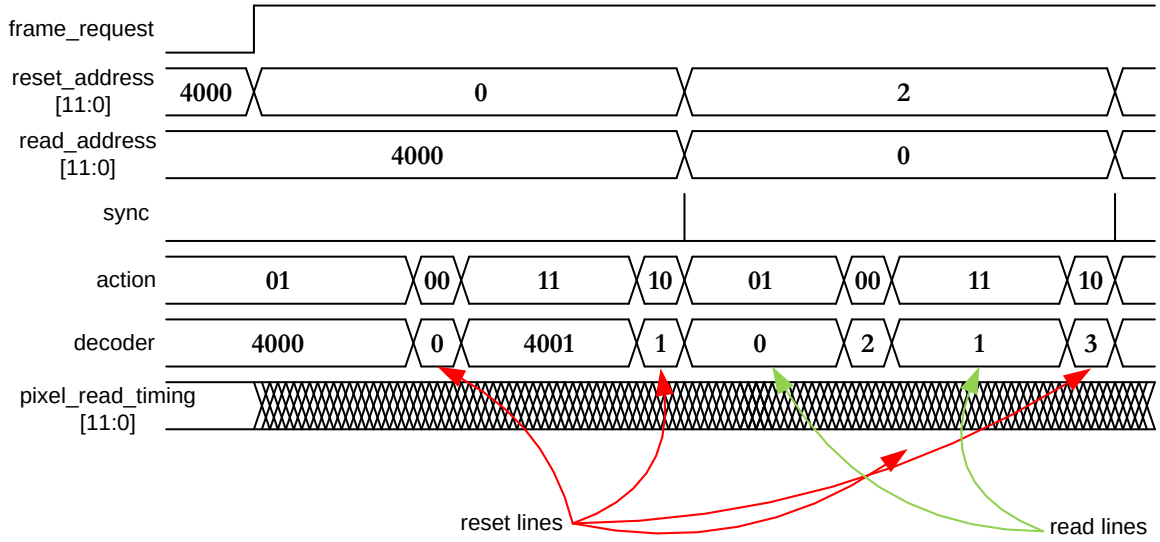



**Figure 5.** Sequence of the beginning of a frame request in STD mode.

**Table 1.** Operations in STD mode according Action value.

| Action | Operation |
|---|---|
| 00 | address reset |
| 01 | address read |
| 10 | address + 1 reset |
| 11 | address + 1 read |

## 4. Image Ordering Algorithm

As mentioned in Section 2, the rows of the CMOS sensor are read or reset in time slots of 513-pixel clock cycles (at 25 MHz) through control signals that the Artix-7 FPGA must send, according Figure 3 in STD mode. This reset and read mode is performed simultaneously for the 8 channels, which distribute the 2048 rows of the sensor as is shown in Table 2.

Figure 6 depicts the data output format in STD mode considering the time slots. For this example, the firmware resets rows from 0 to 13 in 7 time slots and reads consecutive rows starting from the time slot number 3. This number depends on the configured exposure time, which in this case is equivalent to 3 time slots. Before that time, the firmware sends dummy addresses (4000 and 4001 for even and odd rows). The sensor takes 2 time slots to send the data since the firmware sends the read address to the sensor. Then,

and as can be seen in Figure 6, every two rows simultaneously (even and odd), the sensor supplies the data as follows: channel 0, pixels from 0 to 255 of those two rows; channel 1, pixels 256 to 511, and so on. Therefore, the data must be sorted on two levels: by row parity and by channels.

**Table 2.** Data output in STD mode.

| Channel | Even Rows (Address) | | Odd Rows (Address + 1) | |
|---|---|---|---|---|
| | Pixel Init. | Pixel End | Pixel Init. | Pixel End |
| 0 | 0 | 255 | 0 | 255 |
| 1 | 256 | 511 | 256 | 511 |
| 2 | 512 | 767 | 512 | 767 |
| 3 | 768 | 1023 | 768 | 1023 |
| 4 | 1024 | 1279 | 1024 | 1279 |
| 5 | 1280 | 1535 | 1280 | 1535 |
| 6 | 1536 | 1791 | 1536 | 1791 |
| 7 | 1792 | 2047 | 1792 | 2047 |

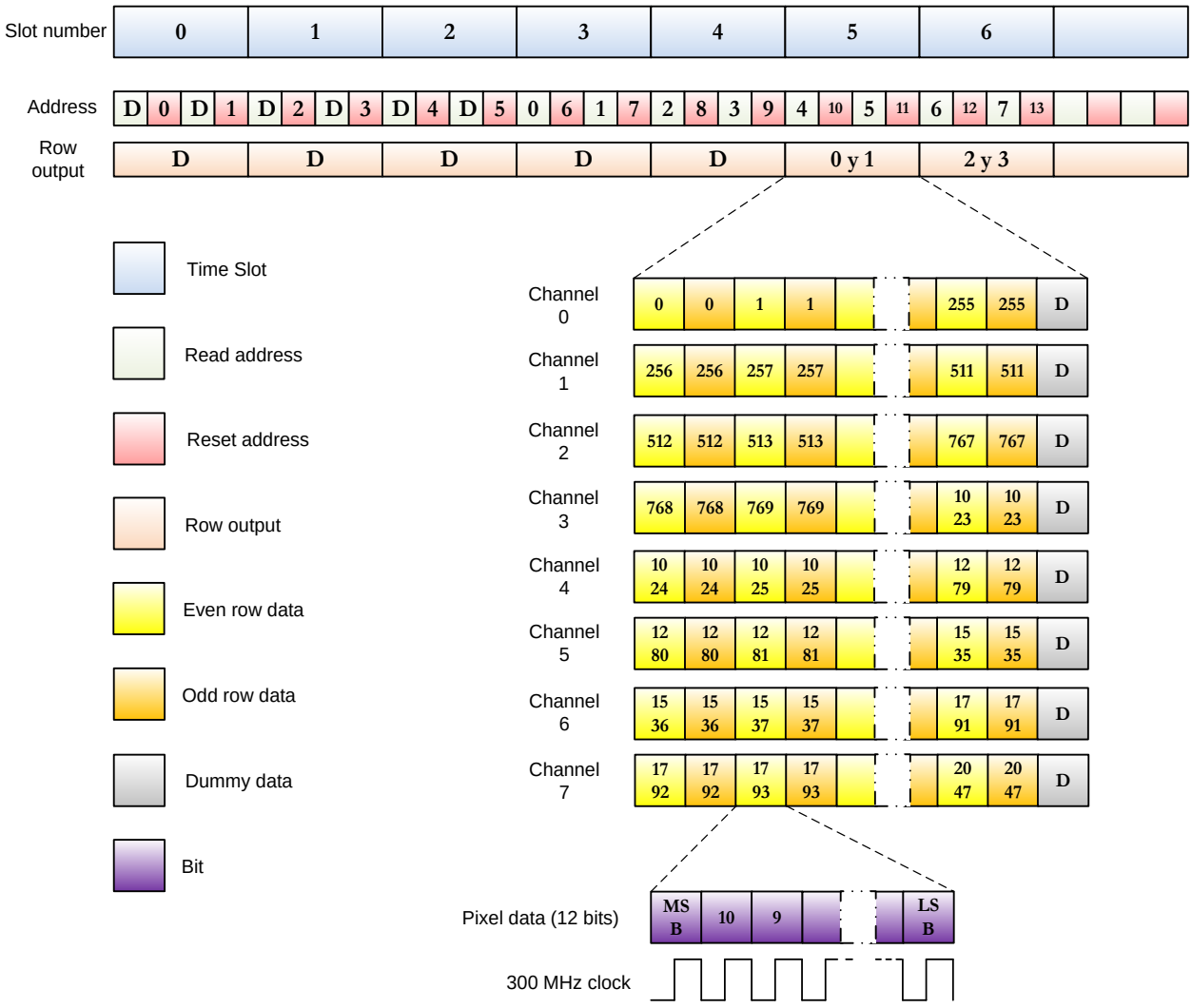



**Figure 6.** Data output of the sensor in STD mode (standard).

### *4.1. Swapping Module*

In order to implement the ordering of the incoming image, a module named swapping has been designed (see Figure 4). This module implements FIFOs (First-In First-Out

components) and multiplexers in order to arrange the incoming 8-channel Low-voltage differential signalling (LVDS) data. There is two-level ordering: Parity arrange (odd and even rows) and channel arrange. It has been described using VHDL language and its interface is shown in Figure 7. A brief description of each module port is also shown in Table 3.

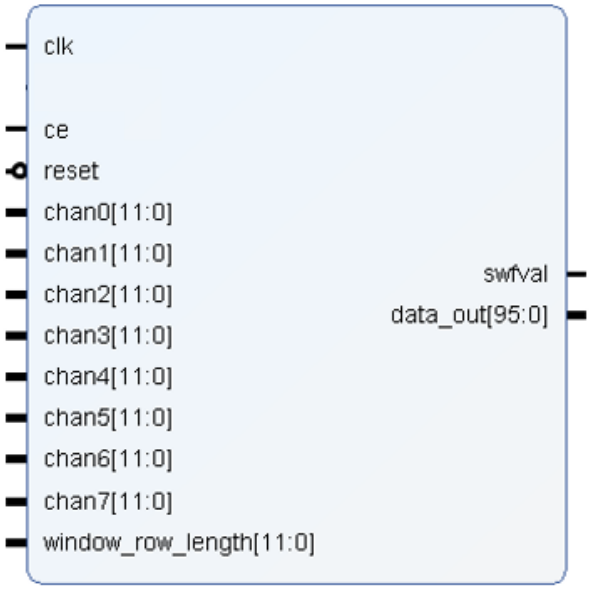



**Figure 7.** Interface of the implemented swapping.vhd module.

**Table 3.** Port description of the swapping module.

| Name | I/O Port | Description |
|---|---|---|
| clk | Input | 25 MHz clock |
| reset | Input | Asynchronous high level clear |
| window_row_length [11:0] | Input | Selection of the Region of Interest (RoI) |
| ce | Input | Enable the submodule when there are incoming data valid in channel ports |
| chan0 to chan7 [11:0] | Input | Pixel data for each channel |
| swfval | Output | Swapped frame valid. Go to high when there are arranged pixel data for data_out |
| data_out [95:0] | Output | Arranged data. Data are supplied in 8-pixel format |

The detailed diagram of this module is shown in Figure 8. For each channel, there is a channel swap submodule. Each submodule includes an even FIFO and an odd FIFO that are continuously swapping for classifying odd and even pixels. In order to implement the image ordering, the incoming data for each channel are stored into the FIFOs inside the *channel_swap* submodules. An FSM that controls *con_rd* counter, which in turn controls the read-out of *channel_swap* and a 16-to-1 multiplexer, selects which data are sent to *data_out* port. Each of the channel_swap submodules are made up of two 7-depth shift-registers and two 96-bit (named SRL7) and 512-depth FIFOs, as shown in Figure 9.

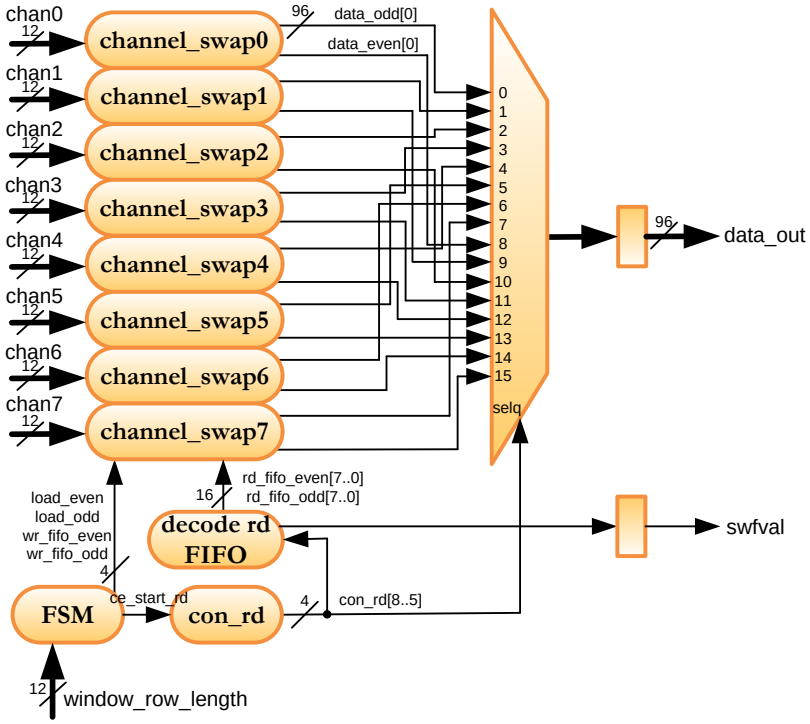



**Figure 8.** Block diagram of the implemented swapping.vhd module.

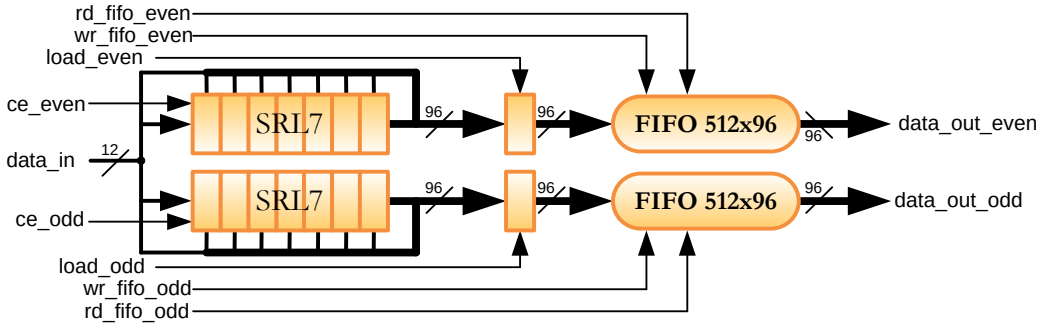



**Figure 9.** Block diagram of the implemented channel_swap.vhd module.

At the input of each FIFO there is a concatenation module in order to group every 8 pixels. The reason for that is that data are divided into 32-bit groups in the packer module in order to use CoaXPress IP. Therefore, the swapping submodule has 16,512 × 96 FIFOs. FSM of the swapping module is basically a 16-module counter that controls *load_even*, *load_odd*, *wr_fifo_even* and *wr_fifo_odd* signals. These signals are common to the 8 submodules. When the counter is 14, the *load_even* signal is set to one. When it reaches 15, the *load_odd* signal is set to one. So, concatenated two 8-pixel data (96 bits) are written into each *channel_swap* with *wr_fifo_even* and *wr_fifo_odd* signals.

Figure 10 shows a simulation of incoming data to the swapping module for channel 0. The pixel data, in pink colour, are set to correlative numbers in the channel simulation in order to better follow the propagation of the signal through the different components. In this way, pixels with even values belong to row number 0 and pixels with odd values belong to row number 1 and so on. The yellow signals in the simulation belong to the upper SRL7 and FIFO system of Figure 9 and the red signals belong to the lower system. The *ce_even* and *ce_odd* signals flip according to the parity of the concatenation counter, *con_count* signal, performing a 1-to-2 demultiplexer, and thus distributing the odd and even row data between the two shift-registers for each channel. When the 7-SRLs are full, *load_even* and *load_odd* signals are set to high to write the concatenated data to the corresponding register. These data are stored in even and odd FIFOs in the next clock cycle

(*wr_fifo_even* and *wr_fifo_odd* set to high), which are the first 8 pixels for rows 0 and 1, of the 256 that it supplies in channel 0 according to Table 2 and Figure 6.

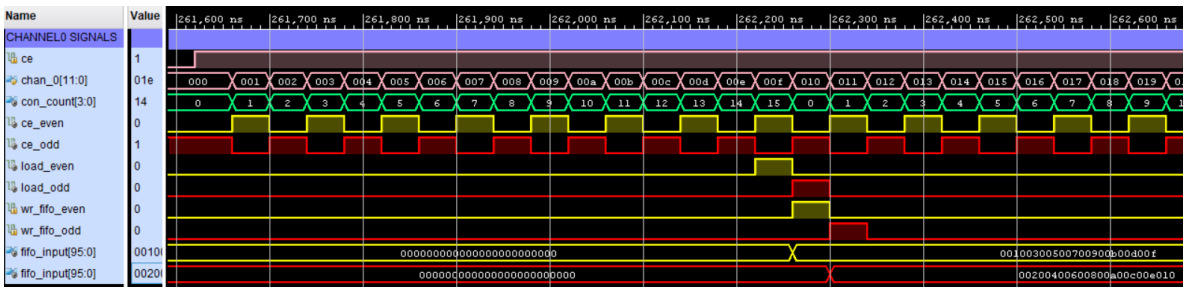



**Figure 10.** Simulation of incoming data to the swapping module for channel 0.

When *swfal* signal is equal to one, *data_out* (see Figure 8) provides the first 8-pixel data for row 0, then the following 8-pixel data and so on until 256 times:

$$256 \times 8 = 2048 \text{ pixels.} \quad (1)$$

Then, the submodule consecutively provides rows 1, 2, and so on until it reaches the *window_row_length* defined by the selected RoI.

Figure 11 shows a simulation of the sequential reading of all FIFOs to form the correctly ordered data. The swapping module waits for enough data to be stored in the FIFOs to avoid any empty FIFO situation. This is ensured when the internal *con_row_chan* counter reaches the value of 127 stored rows and then the ce_start_rd signal is set to high. This trigger activates the *con_rd* counter which is used to generate *rd_fifo* signals and the selection signal of the 16-to-1 multiplexer in Figure 8. The decoding of the *con_rd* counter bits is shown in Table 4. Each *read_fifo* signal in Figure 11 remains high for 32 clock cycles because the 8-pixel data format takes that time to read the 512-pixel section of the row corresponding to its channel.

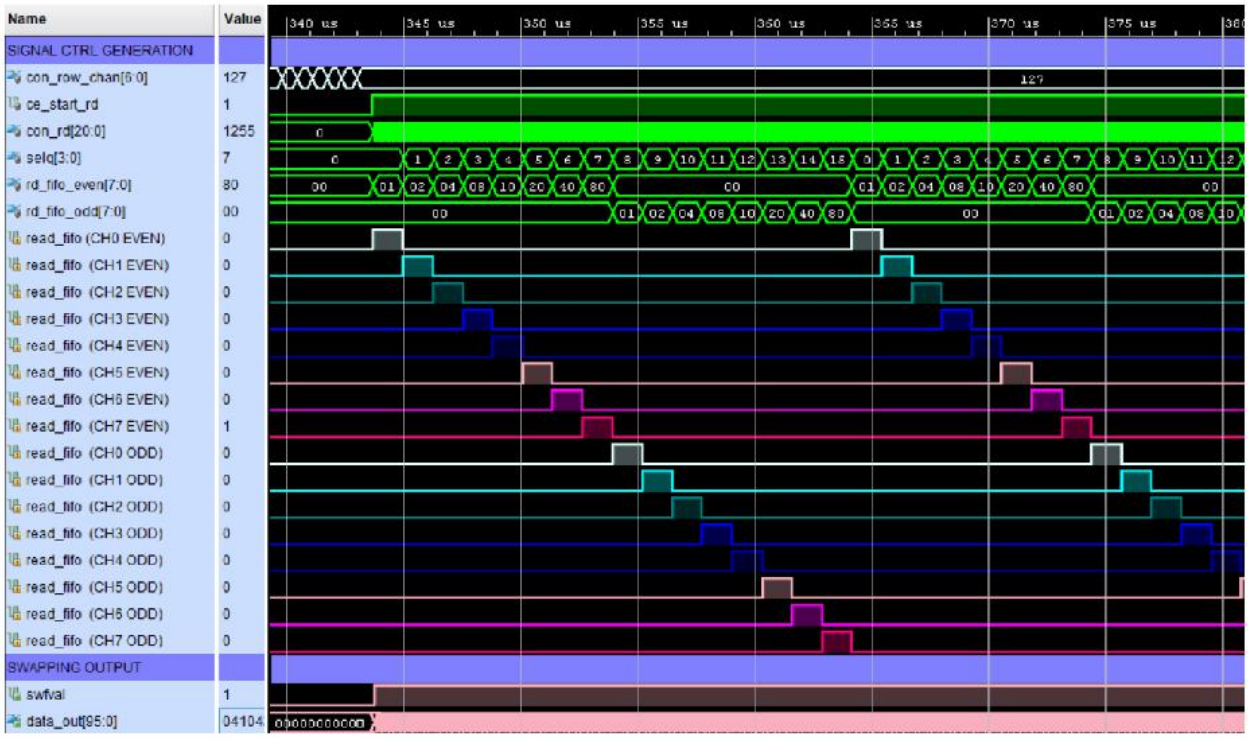



**Figure 11.** Simulation of the FIFOs reading and data output of the swapping module.

**Table 4.** Bit description of con_rd signal.

| con_rd Counter Bits | Description |
|---|---|
| con_rd [20:9] | Number of rows (limited by reg_window_length of RoI) |
| con_rd [8:5] | Selection signal for the 16-to-1 multiplexer (sel signal). Also, it generates read_fifo signals with a decoder |
| con_rd [4:0] | Number of cycles to read data of the section of the row of each channel (32) |

When *swfal* is equal to one, *data_out* provides the first 8-pixel data for row 0, then the following 8-pixel data and so on until 256 (in 32 clock cycles) and then the same with the other channels:

$$256 \times 8 \text{ channels} = 2048 \text{ pixels}. \quad (2)$$

These signals are in pink colour in Figure 11. Then, the submodule consecutively provides rows 1, 2, 3 . . . until it reaches the *window_row_length* defined by the selected RoI. The *data_out* port supplies the image data arranged in rows-by-columns but with the data grouped in packets of 8 pixels (96 bits) as mentioned.

### 4.2. Packer Module

The packer module adapts the incoming 96-bit data (8 pixels) to 32-bit in order to send them via CoaXPress IF. It has been described using VHDL language and its interface is shown in Figure 12. A brief description of each module port is also shown in Table 5.

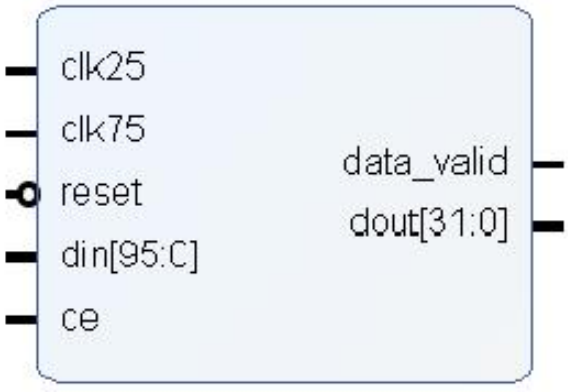



**Figure 12.** Interface of the implemented packer.vhd module.

**Table 5.** Port description of the swapping module.

| Name | I/O Port | Description |
|---|---|---|
| clk25 | Input | 25 MHz clock |
| clk75 | Input | 75 MHz clock |
| reset | Input | Asynchronous low level clear |
| din [95:0] | Input | Incoming data (8-pixel data each clock cycle) |
| ce | Input | Chip enable |
| data_valid | Output | Data valid signal |
| dout [31:0] | Output | Packed data output (2.66-pixel each clock cycle) |

This module implements three 32 × 1024 FIFO to change the clock domain from 25 MHz to 75 MHz and a 3-to-1 multiplexer. This multiplexer selects 31 to 0 bits, 63 to 32 bits and 95 to 64 bits for each incoming data. A system control module generates read and write FIFO signals and the selection signal for the multiplexer. The detailed diagram of this module is shown in Figure 13.

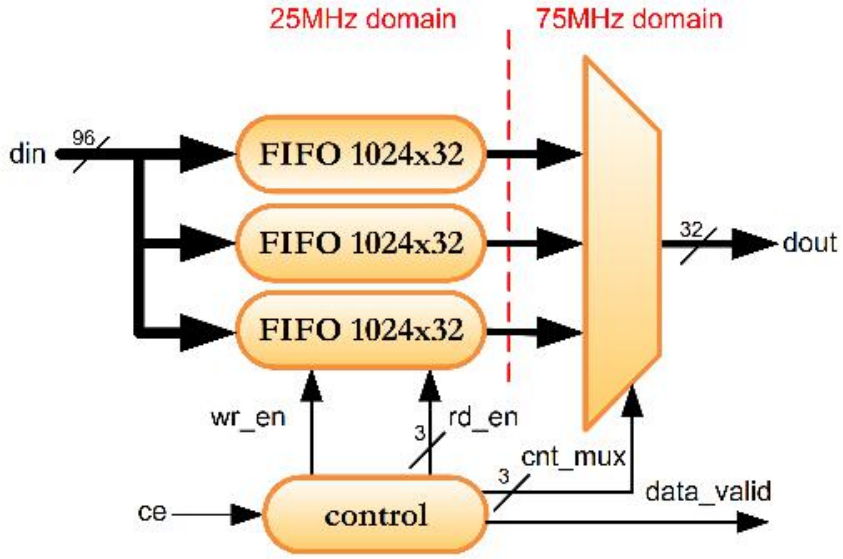



**Figure 13.** Block diagram of the implemented packer.vhd module.

Figure 14 depicts a simulation of the packer operation. When *ce* signal is set to high, *wr_en* signal is set to high and the 8-pixel incoming data are distributed to the 3 FIFOs divided in a 32-pixel format as mentioned. These data are then sequentially read at 75 MHz in a totally sequential operation using *rd_en* signals (in red). The data for the output port is generated by concatenating the outputs of the FIFOs using the *cnt_mux* signal (in yellow). Thus, it can be seen that every 8 pixels are divided into 3 packets of $2^{2/3}$ pixels each (see values the *din* and *dout* orange signals).


**Figure 14.** Simulation of the packer module.

### *4.3. Image_tx Module*

It adapts the data for sending them via CoaXPress IF. The port interface of this module is depicted in Figure 15. *Swfval* data acts as chip enable and *reg_window_row_length*, *reg_window_column_start* and *reg_window_column_length* parameters are used to generate *mas_group* signals. A brief description of each module port is also shown in Table 6.

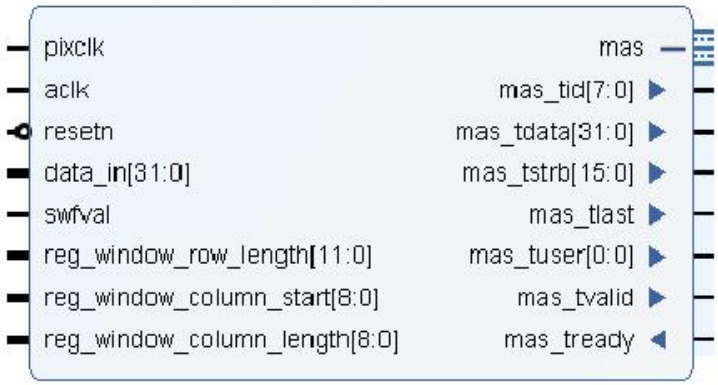



**Figure 15.** Interface of the implemented image_tx.vhd module.

**Table 6.** Port description of the image_tx module.

| Name | I/O Port | Description |
|---|---|---|
| pixclk | Input | 75 MHz clock |
| din [31:0] | Input | 32-bit incoming data (8-pixel each clock cycle) |
| swfval | Input | Chip enable signal |
| dout [31:0] | Output | Data output (8-pixel each clock cycle) |
| reg_window_row_length [11:0] | Input | Data valid signal |
| reg_window_column_start [8:0] | Input | Data valid signal |
| reg_window_column_length [8:0] | Input | Data valid signal |
| mas group | Output | AXI signals |

In this case, the *reg_window_column_start* and *reg_window_column_length* registers have only 9 bits as trimming the RoI by columns only allows multiples of 8, due to the limitation imposed by the 32-bit packet transmission channel. Thus, this module uses dividers to convert these registers into equivalent transmission packets. The *image_tx* module also implements two counters, named *row_counter* and *swfval_count*, which keep track of the rows and the packets of each row that are transmitted.

Figure 16 depicts a simulation of this module for a 16-rows image. When *swfal* is equal to one, it generates a trigger (*mas_tuser* signal), enable a strobe signal (*mas_tvalid*) and the module sends packets to the CoaXPress IP through *mas_tdata* signal (see green signals in Figure 16). In this case, the *reg_window_column_length* register is equal to 256. This is:

$$256 \times 8 = 2048 \text{ columns}, \tag{3}$$

that is, all the columns in the image. Therefore, each row is made up of:

$$2048 \times 12 \text{ bits} = 768 \text{ 32-bit packets} \tag{4}$$

(see *top_row_std* signal in cyan colour). When *row_counter* counter reaches the value 16 and the *swfval_count* counter reaches the value 768, all the image data has been transmitted, and the *mas_tvalid* signal is disabled.

Figure 17 depicts a simulation of an image using a RoI with 16 rows and 512 columns. In this case, *reg_window_column_start* is equal to 2 and *reg_window_column_length* is 64. Performing the same calculations as in the previous case, now *pack_start* signal is 16 and *pack_end* is 6 + 192 = 198 (orange signals in Figure 17). The module only sends packets that are within this range, in such a way that if the counter is less than 6 or greater than 198, the *mas_tvalid* signal is disabled (see green signals).

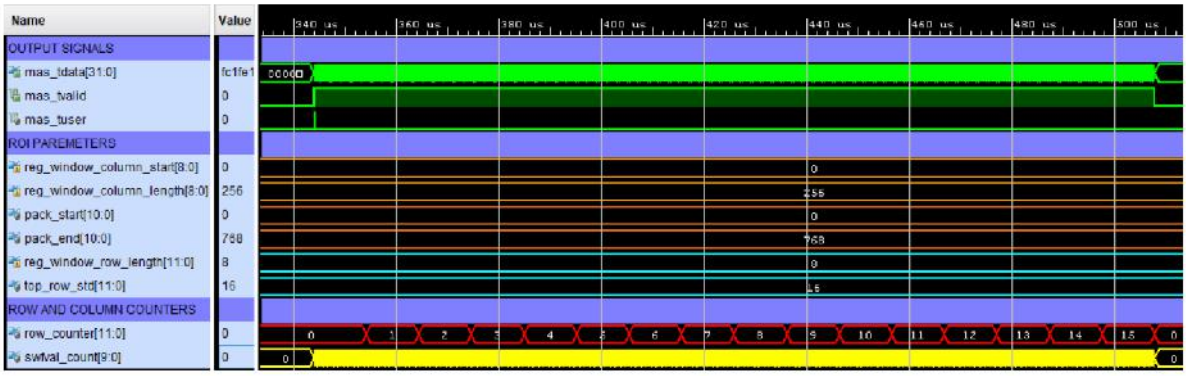



**Figure 16.** Simulation of the image_tx module for a 16 × 2048 image.


**Figure 17.** Simulation the image_tx module for a 16 × 512 image.

## 5. Results

Firmware Image Ordering and Packing for TuMag Camera has been implemented using VHDL language. For this, the Artix-7 XC7A50T-2CSG325C device has been used. The code has been developed using Vivado 2017.4 for simulation, debugging and implementation. This module has been successfully integrated into the sensor driver and is now fully operational. The FPGA resources used by each of the three sub-modules are detailed in Table 7.

**Table 7.** XC7A50T FPGA Resources.

| Name | Slice LUTs (32,600) | Slice Registers (16,300) | Block RAM Tile (75) |
|---|---|---|---|
| Image System | 1517 | 1901 | 28 |
| swapping | 1090 | 1164 | 24 |
| packer | 265 | 524 | 3 |
| image_tx | 162 | 213 | 1 |

It can be seen that the swapping submodule is the one that uses the most resources, mainly due to the relevant number and size of the modules it uses for ordering. In general, the most committed resource is memory (BRAM), used to configure the 16 FIFOs necessary to carry out the ordering operation (37.3%). However, the rest of the resources used by the sensor driver are used in control and configuration tasks and little additional memory is used.

Table 8 shows the latency of the imaging system. It can be seen that the greater latency of the system is due to the image ordering module. *Packer* and *image_tx* modules only handle data grouping and CoaXPress control signals, so the latency is much lower compared to the *swapping* module.

**Table 8.** Latency of the imaging system.

| Name | Clock Cycles | Time |
|---|---|---|
| Image System | 2057 | 82.28 µs |
| swapping | 2049 | 81.96 µs |
| packer | 4 | 0.16 µs |
| image_tx | 4 | 0.16 µs |

The image ordering algorithm was implemented using software tools, in order to obtain the improvement achieved in terms of computational time. For the computational time analysis in Matlab and C++, we used a Dell Vostro PC with the following characteristics: Windows 7 Professional 64 bits, Intel Core i7 860/2.8 GHz, 20 Gb RAM DDR3. The execution time is 37.469 ms using Matlab and 277.57 µs using C++.

The results show the improvement in the computational time of the FPGA implementation over the Matlab and C++ simulations. The speed-up is 455 in comparison with the Matlab implementation and 3.37 in comparison with the C++ implementation.

As we have mentioned previously, GSENSE400 is a relatively new image sensor with high dynamic range, high sensitivity and low noise. The few developments that they have currently been implemented for this sensor do not include the Image Ordering and Packing algorithm in the read-out electronic and this task is performed by software at the host [32–35]. Also, the FPGA devices in these implementations are older: Spartan-6, Virtex-4 and Virtex-5 respectively. Our implementation uses an Artix-7, a FPGA of the 7-series family, with more resources and new components. This increase in performance allows the integration of more functionalities, such as the swapping module. Spartan-6, Virtex-4, and Virtex-5 implementations are developed using the deprecated ISE development environment. Our implementation using an Artix-7 FPGA has been developed using Vivado, the current Xilinx tool. This makes it easy to upgrade the FPGA firmware with new IP libraries.

Figure 18 shows the results of acquiring an image with the prototype in which the USAF (United States Air Force) resolution test chart has been used. The image on the left has been taken without including the swapping module in the firmware (unordered image). In the image on the left, the swapping module has been included in the firmware and the image obtained in the frame grabber is correctly arranged in the desired rows by columns format (ordered image).

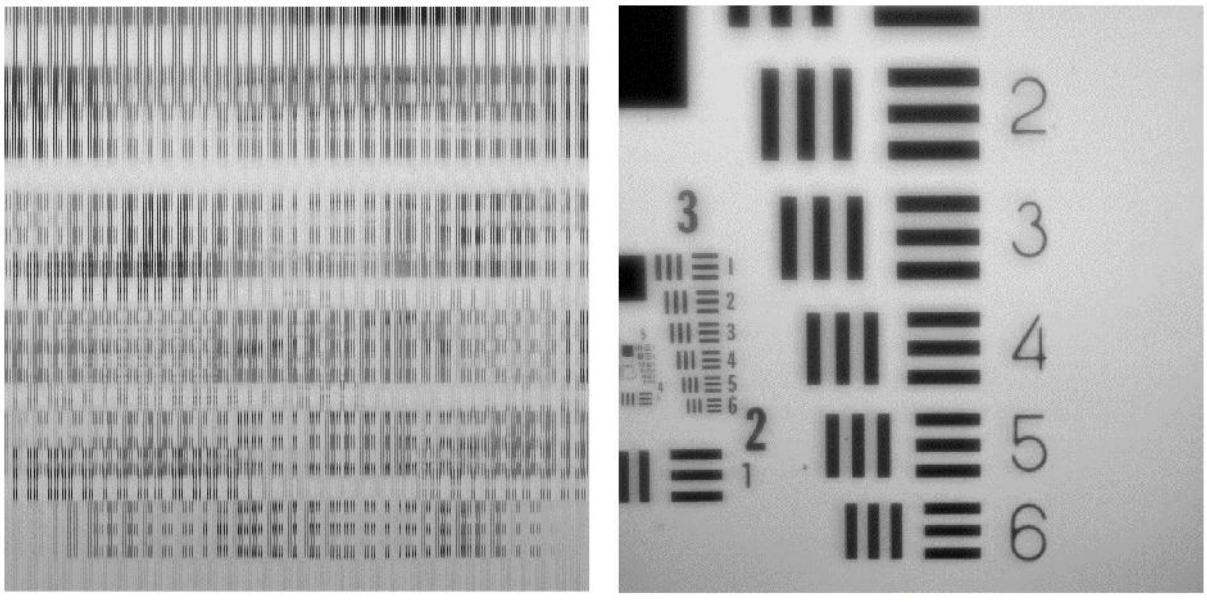



**Figure 18.** 2048 × 2048 image using TuMag camera. (**a**) Unordered image; (**b**) ordered image.

Figure 19 depicts the result of selecting a smaller size in image rows using *reg_window_row_start* equal to 512 and *reg_window_row_length* equal to 512. At left, unordered image is depicted and at centre, ordered image is depicted. As mentioned, the sensor does not allow the selection of a range of columns, so the image at right in Figure 19 comes from the same captured image as for image at centre. In this case, *reg_window_column_start* and *reg_window_column_length* registers have been used to select the range of columns to be sent through the CoaXPress interface in the *image_tx* module, as explained in Section 4.3. In this case, *reg_window_column_start* equal to 128 and *reg_window_column_length* equal to 64 has been used to obtain the sub-image at right.

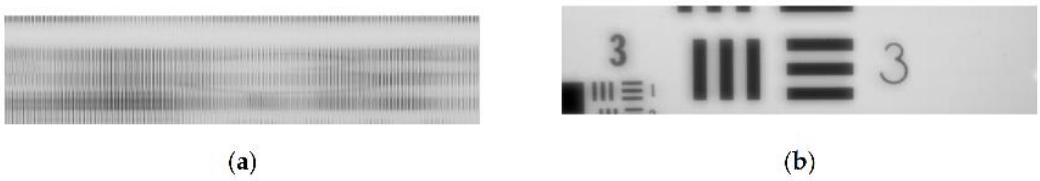



**Figure 19.** 256 × 2048 image using TuMag camera. (**a**) Unordered 256 × 2048 image; (**b**) Ordered 256 × 1024 image; (**c**) Ordered 256 × 256 image.

In the development of the firmware, special attention has been paid to the synchronism of the signals between the modules. Due to the characteristics of the sensor output format, the loss of a pixel causes a misalignment of a pixel, but also the disorder between odd and even rows. This effect can be seen in the images in Figure 20 that were obtained during the development of the firmware. The detail in the image on the left shows unwanted line clutter. In the image on the right the timing has been corrected and is displayed correctly.

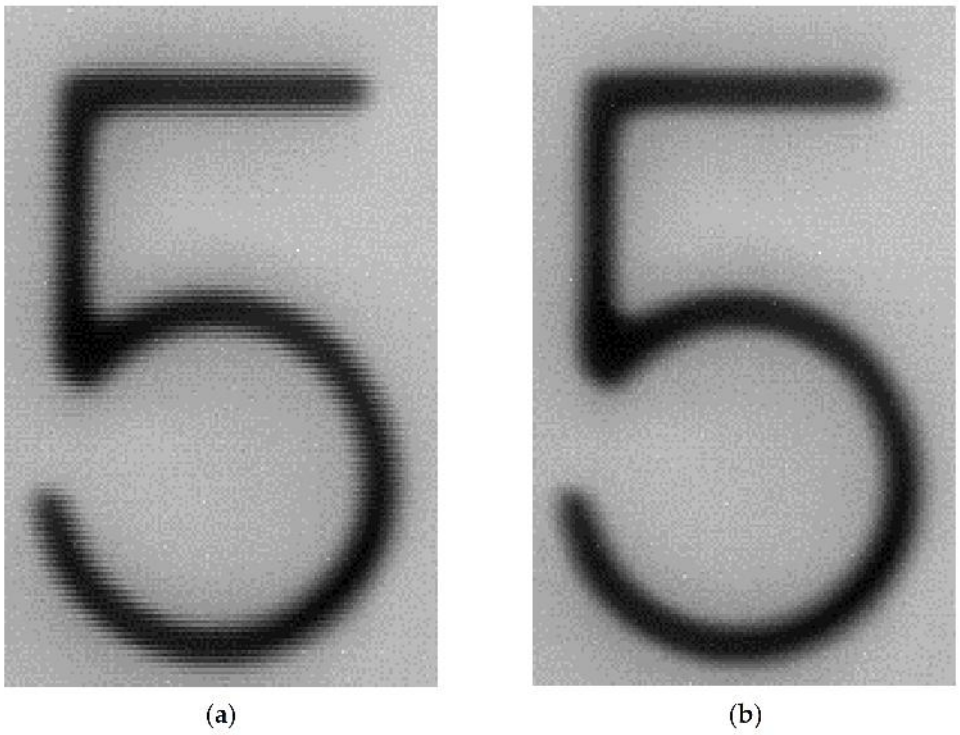



**Figure 20.** (**a**) Detail of an image with undesired line clutter.; (**b**) Detail of an image correctly synchronized.

## 6. Conclusions and Future Work

The camera firmware has been successfully debugged. This firmware includes image ordering and image packing. The image ordering is done correctly without synchronism or line spacing problems. It also adapts the data for sending them via CoaXPress IF. This

uses relatively few FPGA resources, so the rest of the camera control firmware can be implemented without resource issues.

Efforts are currently focused on the development of DPU control firmware. The camera prototype must also be tested in a vacuum chamber in which the temperature will be modified in the range of −20 to 40-Celsius degrees.

Before the flight, rigorous and demanding processes will be carried out to validate the performance of TuMag camera in the laboratory. For this, the extreme conditions of the space environment are simulated on multiple testbenches, which are located into different research center of TuMag project. Consequently, some instruments including the camera and the host/tester must be transported to each center for mount different experimental setups. The transport of this equipment involves risks in its integrity, and requires human and economic project resources to execute the necessary logistics. It is possible to simplify the number of equipment to be transported and reduce the mounting time of experimental setups if each research center has the same host/tester installed on its testbench. In this case, we assume that transportation will only be required for the camera. So, another step consists in the design and implementation of a TuMag camera tester using Eurasys Coaxlink Quad frame grabber and EGenTL/EGrabber libraries for use in multiple testbenches.

**Author Contributions:** Conceptualization, E.M., M.R.V. and D.H.; methodology, E.M., M.R.V. and D.H.; software, E.M., M.R.V., D.H. and D.D.; validation, M.B.; investigation, B.R.C.; writing—original draft preparation, E.M.; writing—review and editing, E.M., M.R.V.; supervision, M.B. and B.R.C.; project administration, B.R.C.; funding acquisition, B.R.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the RTI2018-096886-B-C53 Project of the Ministry of Science and Innovation.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Quintero Noda, C.; Katsukawa, Y.; Shimizu, T.; Kubo, M.; Oba, T.; Ichimoto, K. Sunrise-3: Science target and mission capabilities for understanding the solar atmosphere. In Proceedings of the 19th Space Science Symposium, Sagamihara, Japan, 9 January 2019.
2. Gandorfer, A.; Solanki, S.K.; Woch, J.; Pillet, V.M.; Herrero, A.A.; Appourchaux, T. The Solar Orbiter Mission and its Polarimetric and Helioseismic Imager (SO/PHI). *J. Phys. Conf. Ser.* **2011**, *271*, 8. [CrossRef]
3. Pillet, V.M.; Del Toro Iniesta, J.C.; Álvarez-Herrero, A.; Domingo, V.; Bonet, J.A.; Fernández, L.G.; Jiménez, A.L.; Pastor, C.; Blesa, J.G.; Mellado, P.; et al. The Imaging Magnetograph eXperiment (IMaX) for the Sunrise Balloon-Borne Observatory. *Sol. Phys.* **2011**, *268*, 57–102.
4. Barthol, P. *The Sunrise Balloon-Borne Stratospheric Solar Observatory*, 1st ed.; Springer: New York, NY, USA, 2011; pp. 1–123.
5. Barthol, P.; Gandorfer, A.; Solanki, S.K.; Schüssler, M.; Chares, B.; Curdt, W.; Deutsch, W.; Feller, A.; Germerott, D.; Grauf, B.; et al. The Sunrise Mission. *Sol. Phys.* **2011**, *268*, 1–34. [CrossRef]
6. Solanki, S.K.; Barthol, P.; Danilovic, S.; Feller, A.; Gandorfer, A.; Hirzberger, J.; Riethmüller, T.L.; Schüssler, M.; Bonet, J.A.; Pillet, V.M.; et al. Sunrise: Instrument, Mission, Data, and First Results. *Astrophys. J. Lett.* **2010**, *723*, L127–L133. [CrossRef]
7. Solanki, S.K.; Riethmüller, T.L.; Barthol, P.; Danilovic, S.; Deutsch, W.; Doerr, H.P.; Feller, A.; Gandorfer, A.; Germerott, D.; Gizon, L.; et al. The Second Flight of SUNRISE Balloon-borne Solar Observatory: Overview of Instrument Updates, the Flight, the Data, and First Results. *Astrophys. J. Suppl. Ser.* **2017**, *229*, 16. [CrossRef]
8. Kaithakkal, A.J.; Riethmüller, T.L.; Solanki, S.K.; Lagg, A.; Barthol, P.; Gandorfer, A.; Gizon, L.; Hirzberger, J.; Rodríguez, J.B.; Iniesta, J.D.T.; et al. Moving Magnetic Features Around a Pore. *Astrophys. J. Suppl. Ser.* **2017**, *229*, 13. [CrossRef]
9. Centeno, R.; Rodríguez, J.B.; Iniesta, J.D.T.; Solanki, S.K.; Barthol, P.; Gandorfer, A.; Gizon, L.; Hirzberger, J.; Riethmüller, T.L.; van Noort, M.; et al. A Tale of Two Emergences: SUNRISE II Observations of Emergence Sites in a Solar Active Region. *Astrophys. J. Suppl. Ser.* **2017**, *229*, 3. [CrossRef]
10. Requerey, I.S.; Iniesta, J.C.D.T.; Rubio, L.R.B.; Bonet, J.A.; Pillet, V.M.; Solanki, S.K.; Schmidt, W. The History of a Quiet-Sun Magnetic Element Revealed by IMaX/SUNRISE. *Astrophys. J. Suppl. Ser.* **2014**, *789*, 12. [CrossRef]
11. Requerey, I.S.; Cobo, B.R.; Iniesta, J.D.T.; Suárez, D.O.; Rodríguez, J.B.; Solanki, S.K.; Barthol, P.; Gandorfer, A.; Gizon, L.; Hirzberger, J.; et al. Spectropolarimetric Evidence for a Siphon Flow along an Emerging Magnetic Flux Tube. *Astrophys. J. Suppl. Ser.* **2017**, *229*, 5. [CrossRef]

12. Castelló, E.M.; Valido, M.R.; Expósito, D.H.; Cobo, B.R.; Balaguer, M.; Suárez, D.O.; Jiménez, A.L. IMAX+ camera prototype as a teaching resource for calibration and image processing using FPGA devices. In Proceedings of the 2020 XIV Technologies Applied to Electronics Teaching Conference (TAEE), Porto, Portugal, 8–10 July 2020.
13. Berkefeld, T.; Schmidt, W.; Soltau, D.; Bell, A.; Doerr, H.P.; Feger, B.; Friedlein, R.; Gerber, K.; Heidecke, F.; Kentischer, T.; et al. The Wave-Front Correction System for the Sunrise Balloon-Borne Solar Observatory. *Sol. Phys.* **2011**, *268*, 103–123. [CrossRef]
14. Riethmüller, T.L.; Solanki, S.K.; Barthol, P.; Gandorfer, A.; Gizon, L.; Hirzberger, J.; Van Noort, M.; Rodríguez, J.B.; Iniesta, J.D.T.; Suárez, D.O.; et al. A New MHD-assisted Stokes Inversion Technique. *Astrophys. J. Suppl. Ser.* **2017**, *229*, 14. [CrossRef]
15. Orozco Suárez, D.; Bellot Rubio, L.R.; del Toro Iniesta, J.C.; Tsuneta, S.; Lites, B.W.; Ichimoto, K.; Katsukawa, Y.; Nagata, S.; Shimizu, T.; Shine, R.A.; et al. Quiet-Sun Internetwork Magnetic Fields from the Inversion of Hinode Measurements. *Astrophys. J.* **2007**, *670*, L61–L64. [CrossRef]
16. Gpixel. *GSENSE400 4 Megapixels Scientific CMOS Image Sensor. Datasheet V1.5*; Gpixel Inc.: Changchum, China, 2017.
17. Xilinx, 7 Series FPGAs Data Sheet: Overview. Product Specification. DS180 (v2.6.1) 2020. Available online: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (accessed on 2 December 2020).
18. EASii IC. *CoaxPress Device IP Specification. Hard Soft Interface Document. IC/130206*; EASii IC SAS: Grenoble, France, 2017.
19. Magdaleno, E.; Rodríguez, M.; Rodríguez-Ramos, J.M. An Efficient Pipeline Wavefront Phase Recovery for the CAFADIS Camera for Extremely Large Telescopes. *Sensors* **2010**, *10*, 1–15. [CrossRef] [PubMed]
20. Magdaleno, E.; Lüke, J.P.; Rodríguez, M.; Rodríguez-Ramos, J.M. Design of Belief Propagation Based on FPGA for the Multistereo CAFADIS Camera. *Sensors* **2010**, *10*, 9194–9210. [CrossRef] [PubMed]
21. Pérez, J.; Magdaleno, E.; Pérez, F.; Rodríguez, M.; Hernández, D.; Corrales, J. Super-Resolution in Plenoptic Cameras Using FPGAs. *Sensors* **2014**, *14*, 8669–8685. [CrossRef] [PubMed]
22. Rodríguez, M.; Magdaleno, E.; Pérez, F.; García, C. Automated Software Acceleration in Programmable Logic for an Efficient NFFT Algorithm Implementation: A Case Study. *Sensors* **2017**, *17*, 694. [CrossRef] [PubMed]
23. Alvear, A.; Finger, R.; Fuentes, R.; Sapunar, R.; Geelen, T.; Curotto, F.; Rodríguez, R.; Monasterio, D.; Reyes, N.; Mena, P.; et al. FPGA-based digital signal processing for the next generation radio astronomy instruments: Ultra-pure sideband separation and polarization detection, in Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy VIII. In Proceedings of the SPIE, Edinburgh, UK, 26 June–1 July 2016; Volume 9914. [CrossRef]
24. Schwanke, U.; Shayduk, M.; Sulanke, K.-H.; Vorobiov, S.; Wischnewski, R. A Versatile Digital Camera Trigger for Telescopes in the Cherenkov Telescope Array. In *Nuclear Instruments and Methods in Physics Research Section A: Accelerators Spectrometers Detectors and Associated Equipment*; Elsevier: Amsterdam, The Netherlands, 2015; Volume 782, pp. 92–103. [CrossRef]
25. Lecerf, A.; Ouellet, D.; Arias-Estrada, M. Computer vision camera with embedded FPGA processing. In Proceedings of the SPIE 3966, Machine Vision Applications in Industrial Inspection VIII, San Jose, CA, USA, 21 March 2000. [CrossRef]
26. Heller, M.; Schioppa, E.; Jr Porcelli, A.; Pujadas, I.T.; Ziętara, K.; Della Volpe, D.; Montaruli, T.; Cadoux, F.; Favre, Y.; Aguilar, J.A.; et al. An innovative silicon photomultiplier digitizing camera for gamma-ray astronomy. *Eur. Phys. J. C* **2017**, *77*, 47. [CrossRef]
27. Sawyer, N. LVDS Source Synchronous 7:1 Serialization and Deserialization Using Clock Multiplication. Xilinx XAPP585 (v.1.1.2) July 8, 2018. Available online: https://www.xilinx.com/support/documentation/application_notes/xapp585-lvds-source-synch-serdes-clock-multiplication.pdf (accessed on 2 December 2020).
28. Magdaleno, E.; Rodríguez, M. TuMag Camera Firmware design and implementation report. SR3-IMAXP-RP-SW730-001. Revision A. Release date 2020-08-01, La Laguna, Spain, September 2021.
29. CoaxPress Host IP Specification. *Hard Soft Interface Document. IC/130249*; EASii IC SAS: Grenoble, France, 2017.
30. GenICam Standard. Generic Interface for Cameras. Version 2.1.1. Emva. Available online: https://www.emva.org/wp-content/uploads/GenICam_Standard_v2_1_1.pdf (accessed on 2 December 2020).
31. MicroBlaze Processor Reference Guide. Xilinx UG984 (v2018.2) June 21, 2018. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug984-vivado-microblaze-ref.pdf (accessed on 2 December 2020).
32. Tang, X.; Liu, G.; Qian, Y.; Cheng, H.; Qiao, K. A Handheld High-Resolution Low-Light Camera. In Proceedings of the SPIE 11023, Fifth Symposium on Novel Optoelectronic Detection Technology and Application, Xi'an, China, 12 March 2019; Volume 110230X. [CrossRef]
33. Heng, Z.; Qing-jun, M.; Shu-rong, W. High speed CMOS imaging electronics system for ultraviolet remote sensing instrument. *Opt. Precis. Eng.* **2018**, *26*, 471–479. [CrossRef]
34. Ma, C.; Liu, Y.; Li, J.; Zhou, Q.; Chang, Y.; Wang, X. A 4MP high-dynamic-range, low-noise CMOS image sensor. In Proceedings of the SPIE 9403, Image Sensors and Imaging Systems 2015, San Francisco, CA, USA, 13 March 2015; Volume 940305. [CrossRef]
35. Wang, F.; Dai, M.; Sun, Q.; Ai, L. Design and implementation of CMOS-based low-light level night-vision imaging system. In Proceedings of the SPIE 11763, Seventh Symposium on Novel Photoelectronic Detection Technology and Applications, San Francisco, CA, USA, 12 March 2021; Volume 117635O. [CrossRef]