SUDE Experiment

Camera Task Report

Supervisor	Johnny Öberg
Student	Ahmadmunthar Zaklouta

Abstract:

This document contains the documentation for the camera task within SEUD experiment in MIST project. The task is to experiment interfacing a low-cost camera to MIST. Two cameras from OmniVision have been investigated for this experiment; ov5647 and ov7670. ov7670 has been chosen for several reasons such as availability, readiness, datasheet clarity and quality, ease of connectivity, and example availability.

Contents

Table of Figures:

1. Introduction:

This section contains brief introduction and explanation about imaging principles and sensors.

1.1 Basic Imaging Principle:

"Incoming light hits the image sensor in form of photons accordingly generating electrons. These are transported pixel by pixel to the serial register and subsequently run through an amplifier. The generated voltage can be converted by an analog-digital converter (not shown) to a digital image signal". (2)

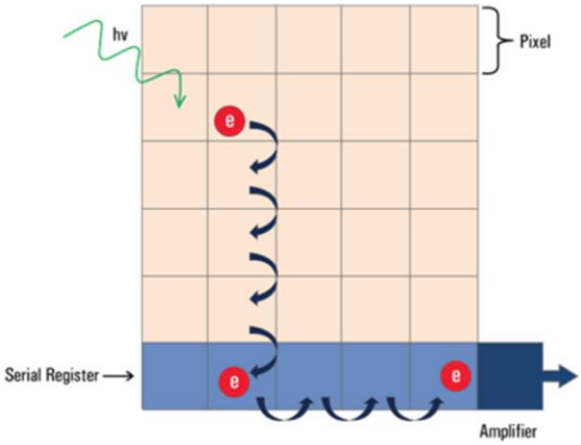


Figure 1: Basic imaging principle.

1.2 Image Sensor Types:

Basically, there are two categories of sensors in the machine: Charge-Coupled Device (CCD) and Complementary Metal Oxide Semiconductor (CMOS) imagers.

1.2.1 Charge-Coupled Device (CCD): "Capture light on the small photo-sites on their surface. To begin, the charges on the first row are transferred to a readout register. From there, the signals are then fed to an amplifier *one pixel at a time* and then on to an analog-to-digital converter. Once a row has been read, its charges on the read-out register row are deleted. The next row then enters the readout register, and all of the rows above move down one row. The charges on each row are "coupled" to those on the row above so when one moves down, the next moves down to fill its old space. In this way, each row can be read—one row at a time". (3)

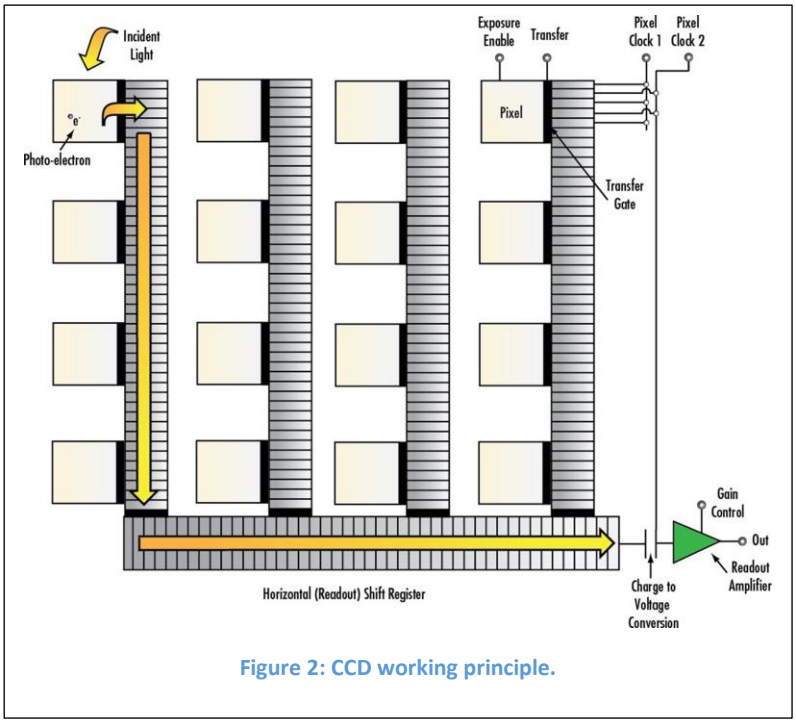


Figure 2: CCD working principle.

1.2.2 Metal Oxide Semiconductor (CMOS): "The charge from the photosensitive pixel is converted to a voltage at the pixel site and the signal is multiplexed by row and column to multiple on chip digital-to-analog converters (DACs). Inherent to its design, CMOS is a digital device. Each site is essentially a photodiode and three transistors, performing the functions of resetting or activating the pixel, amplification and charge conversion, and selection or multiplexing. This leads to the high speed of CMOS sensors". (4)

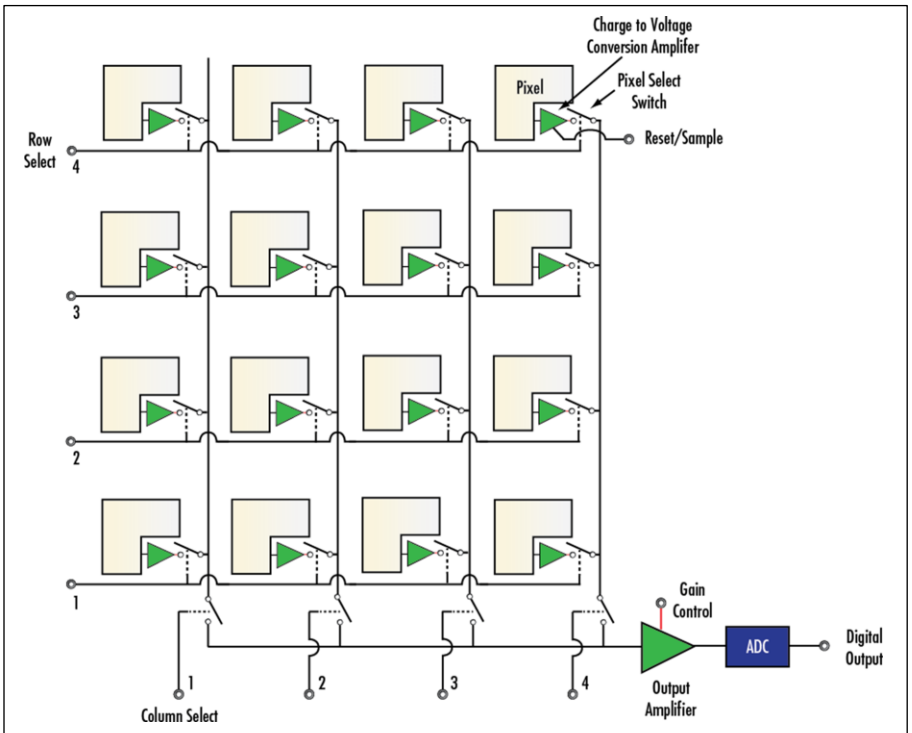
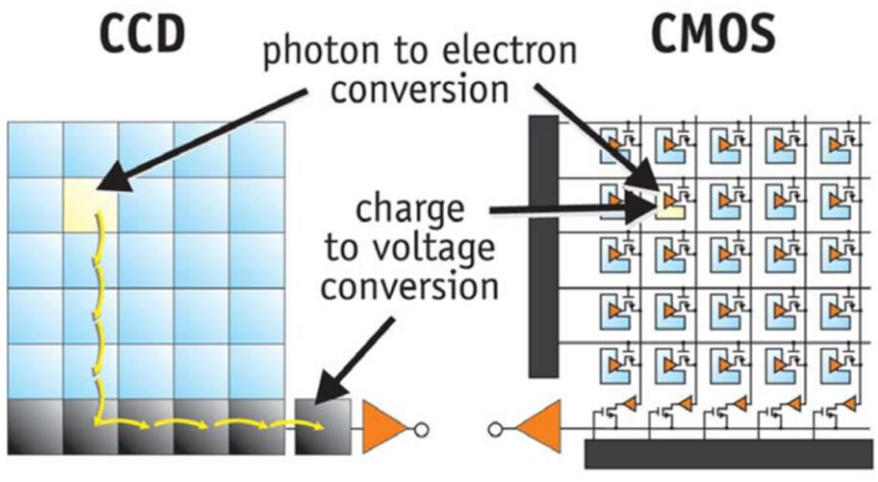


Figure 3: CMOS image sensor working principle.

The main difference between CCD and CMOS in term of working principle is depicted in *figure4*:



CCDs move photogenerated charge from pixel to pixel and convert it to voltage at an output node. CMOS imagers convert charge to voltage inside each pixel.

Figure 4: CCD VS CMOS.

2. Ov7670 camera:

"The OV7670 CAMERACHIPTM is a low voltage CMOS image sensor that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. The OV7670/OV7171 provides full-frame, sub-sampled or windowed 8-bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface". (5)

Main Specifications	
Array Element (VGA)	640 x 480
Output Formats (8-bit)	• YUV/YCbCr 4:2:2 • RGB565/555 • GRB 4:2:2 • Raw RGB Data
Maximum Image Transfer Rate	30 fps for VGA
Electronics Exposure	Up to 510:1 (for selected fps)
Pixel Size	3.6 µm x 3.6 µm
Automatic Exposure Control (AEC).	
Automatic Gain Control (AGC).	
Automatic White Balance (AWB).	
Automatic Band Filter (ABF).	
Automatic Black-Level Calibration (ABLC).	

2.1 Functional Block Diagram:

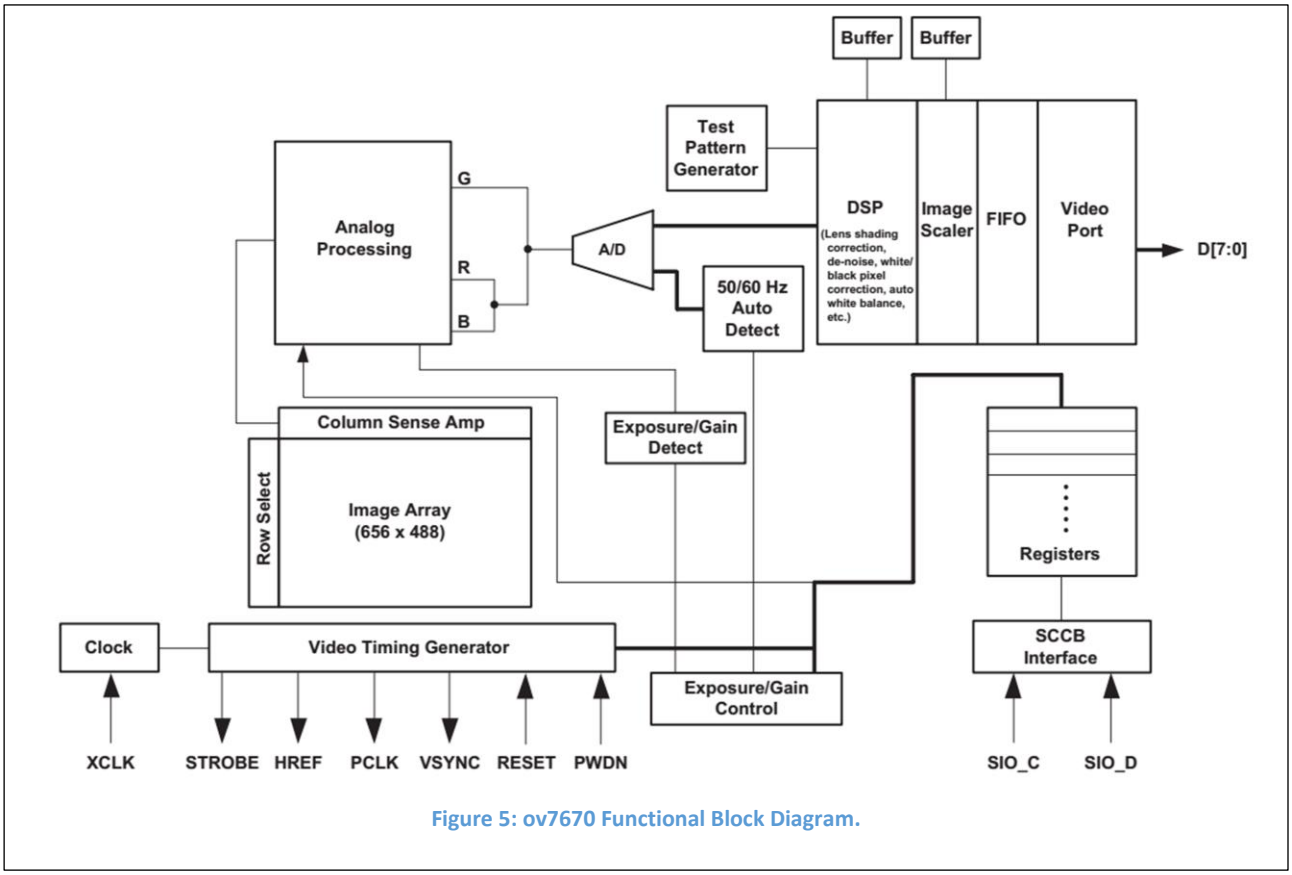


Figure 5: ov7670 Functional Block Diagram.

2.2 Image's Frame Specifications:

Format	Raw Bayer RGB (8-bit R or 8-bit G or 8-bit B)
Resolution	VGA (640x480)

This specification can be achieved by implementing this configuration:

Format	**Pixel Data Output**	**Register Settings**			
		COM7[2]	**COM7[0]**	**COM15[5]**	**COM15[4]**
Raw Bayer RGB	8-bit R or 8-bit G or 8-bit B	0	1	X	0

Resolution	**Register**	**Address**	**Value**	**Description (24 MHz Input Clock)**
VGA (640x480)	CLKRC	0x11	0x81	30 fps VGA Raw Bayer RGB mode
	COM7	0x12	0x01	
	COM3	0x0C	0x00	
	COM14	0x3E	0x00	
	SCALING_XSC	0x70	0x3A	
	SCALING_YSC	0x71	0x35	
	SCALING_DCWCTR	0x72	0x00	
	SCALING_PCLK_DIV	0x73	0xF0	
	SCALING_PCLK_ DELAY	0xA2	0x02	

Each image point takes three clock cycles to send the full pixel data to the FPGA as depicted in the table below:

MODE	BYTE	D7	D6	D5	D4	D3	D2	D1	D0	
RGB888	First (RED)	R7	R6	R5	R4	R3	R2	R1	R0	Pixel
	Second (GREEN)	G7	G6	G5	G4	G3	G2	G1	G0	
	Third (BLUE)	B7	B6	B5	B4	B3	B2	B1	B0	

Related configuration parameters are presented in the table below:

Automatic Gain Control	AGC Enable
Automatic White Balance	AWB Enable
Automatic Exposure Control	AEC Enable
Automatic Exposure/Gain speed	Enable fast AGC/AEC algorithm

2.3 Clocking Specifications:

The camera is clocked by PLL from the FPGA. The input clock signal is pumped up by a PLL multiplier first and then divided the clock by Pre-scalar.

Function	Register	Address	Description
PLL Multiplier	DBLV[7:6]	0x6B	**PLL control Bit[7:6]:** • 00: Bypass PLL. • 01: PLL times the input clock by 4. • 10: PLL times the input clock by 6. • 11: PLL times the input clock by 8.
Clock pre-scalar	CLKRC	0x11	• **Bit[6]:** Use external clock directly (no clock pre-scale available). • **Bit[5:0]:** Internal clock pre-scalar F(internal clock) = F(clk_in)/(Bit[5:0]+1) Range: [0 0000] to [1 1111]

The internal clock can be calculated using the following equation:

$$f_{int\_clk} = \frac{f_{clk} * PLL_{Multiplier}}{2 * (CLKRC[5:0] + 1)}$$

By default, the *PCLK* will have the same frequency of *XCLK*; however, pre-scalars and PPLs can be configured using the I2C to produce a *PCLK* of different frequency.

The time for one frame can be calculated using the following equation:

$$frame\ size\ (rows * columns) * bytes\ per\ pixel\ = PCLK\ cycles\ per\ frame$$

$$656\ (columns) * 488\ (rows) * \frac{3\ bytes}{pixel}(RGB888) = \ 960384\ PCLK\ cycles\ per\ frame$$

AS the task is not to stream video but to capture one frame (picture), a single scan of a frame is needed and *PCLK* frequency can be low.

A proposed clocking configuration is presented in the following table:

System Clock	PLL Multiplier DBLV	Clock pre-scalar CLKRC	Internal Clock	PCLK COM14
12 MHz	1 (int_clk By 4)	1 (Divided by 4)	12 MHz	12 MHz (Normal)

2.4 Synchronization Signals:

There are three synchronization signals generated by camera listed in the below table:

Signal	Description
PCLK	Pixel clock: PCLK signal is free running by default but can be gated by HREF.
HREF	Horizontal reference: The HREF signal is only valid when there is output data (during sending each row). The rising edge of HREF signals the start of a row, and the falling edge of HREF signals the end of the row. If there is no output data, the HREF signal will remain at either high or low, depending on the polarity selection.
VSYNC	Vertical synchronization: default value is low and a pulse determines the start or the end of the frame.

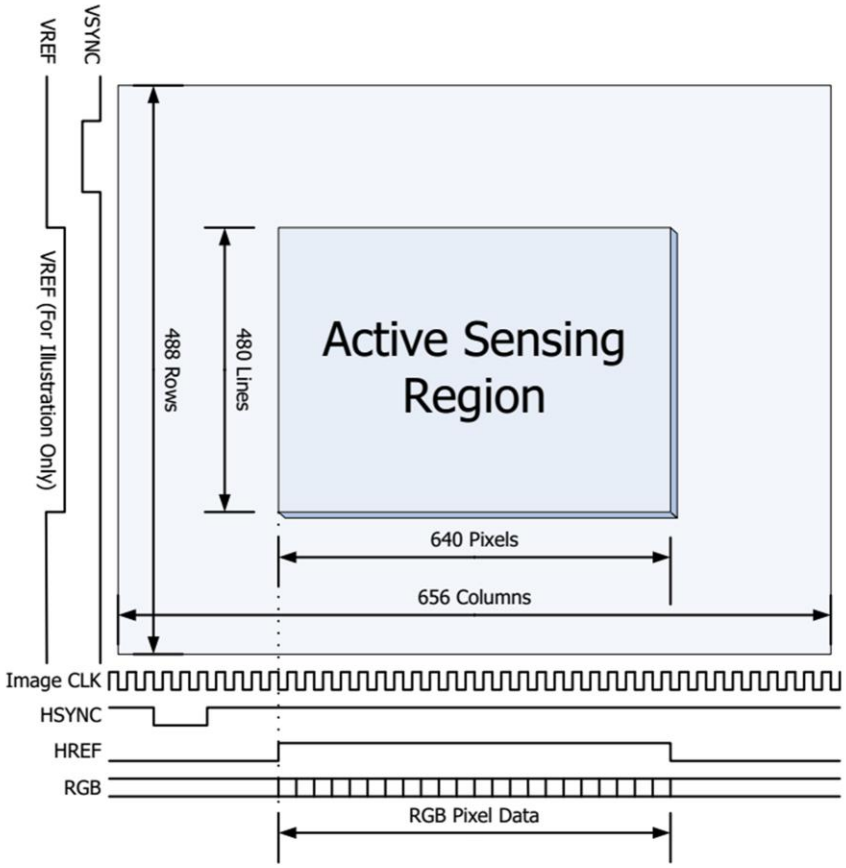


Figure 6: Typical Parallel Interface Timing.

3. Implementation:

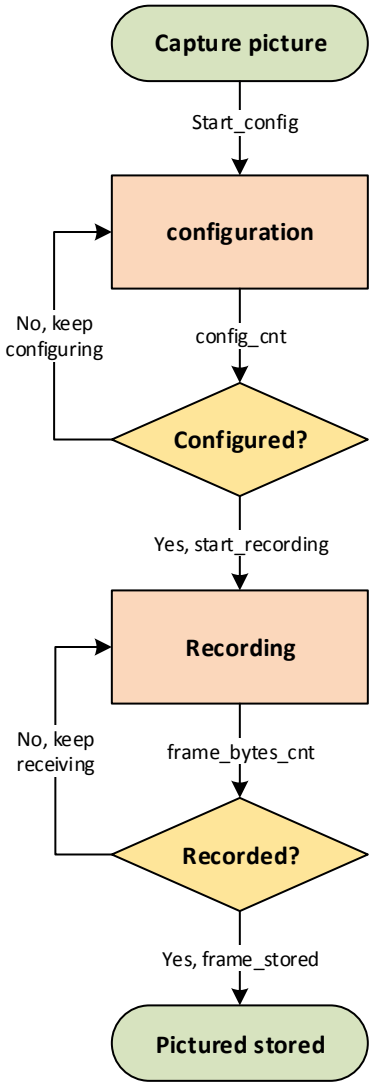
This section contains the implementation and design of the interface.

3.1 Design's General Flow Chart:

There are two stages for taking a picture:

1- **Configuration**: configuring the camera registers.
2- **Recording**:
    1. **Receiving**: receiving the picture data from the camera.
    2. **Transmitting**: sending the picture data to SDRAM controller.

The process starts by receiving *capture* signal from the main controller (ARM), and then the design asserts a start_config signal in order to start the process of configuring the camera. After all required registers have been configured, the design asserts a start_recording signal which indicates that the design is ready for recording process which is receiving the streamed picture from the camera and sending it to SDRAM controller in order to store it. After all picture's frame bytes have been received and sent to SDRAM controller, the design asserts a frame_stored signal which indicates the end of the process of taking the picture and the design go back to its idle state and wait for next capture signal.



3.2 Main block diagram:

The camera interface design is built from three main units:

- Camera_top:
  1. Sensor_Ctrl:
     1. I2C.
     2. Register_map.
  2. PCI (Parallel Camera Interface).
  3. Frame_Ctrl:
     1. FIFO.

The connection and communication between the units are depicted in *figure7*:

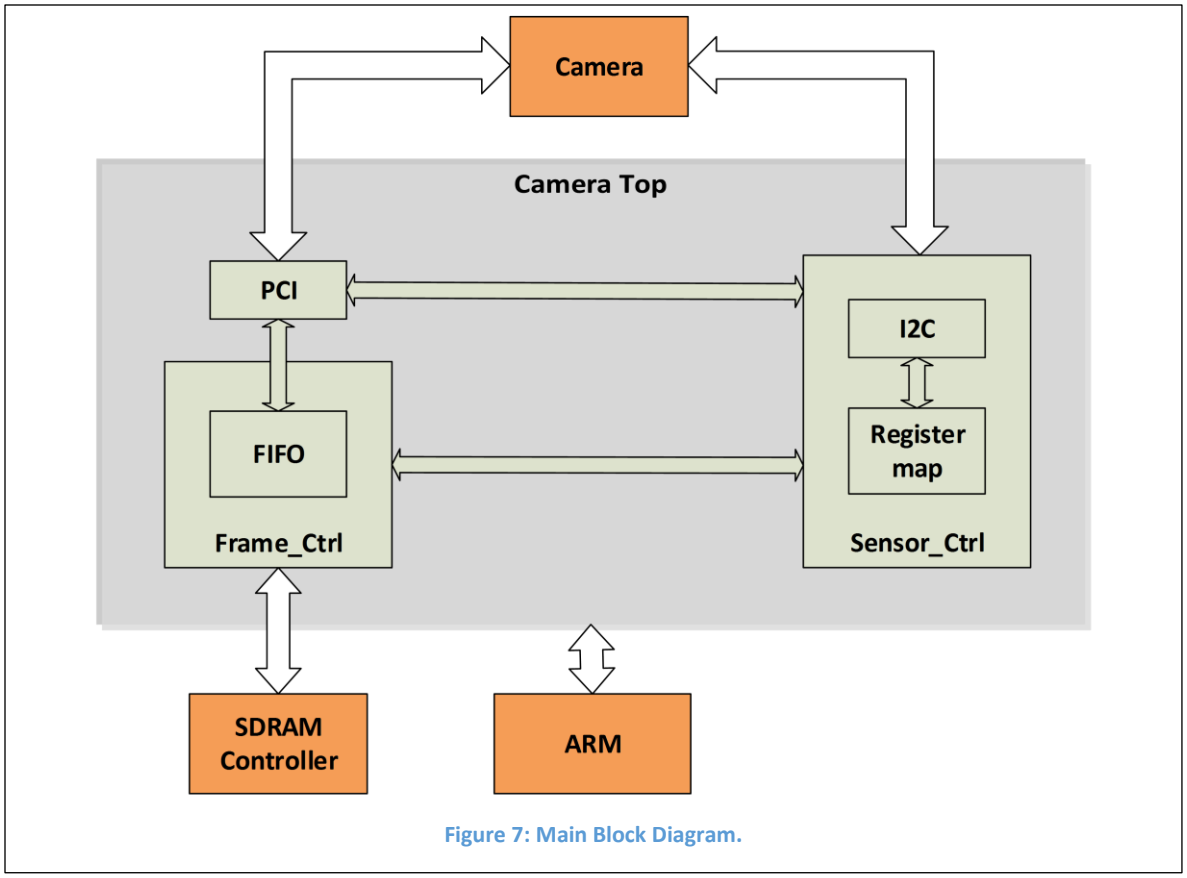


Figure 7: Main Block Diagram.

The schematic is depicted in figure8:

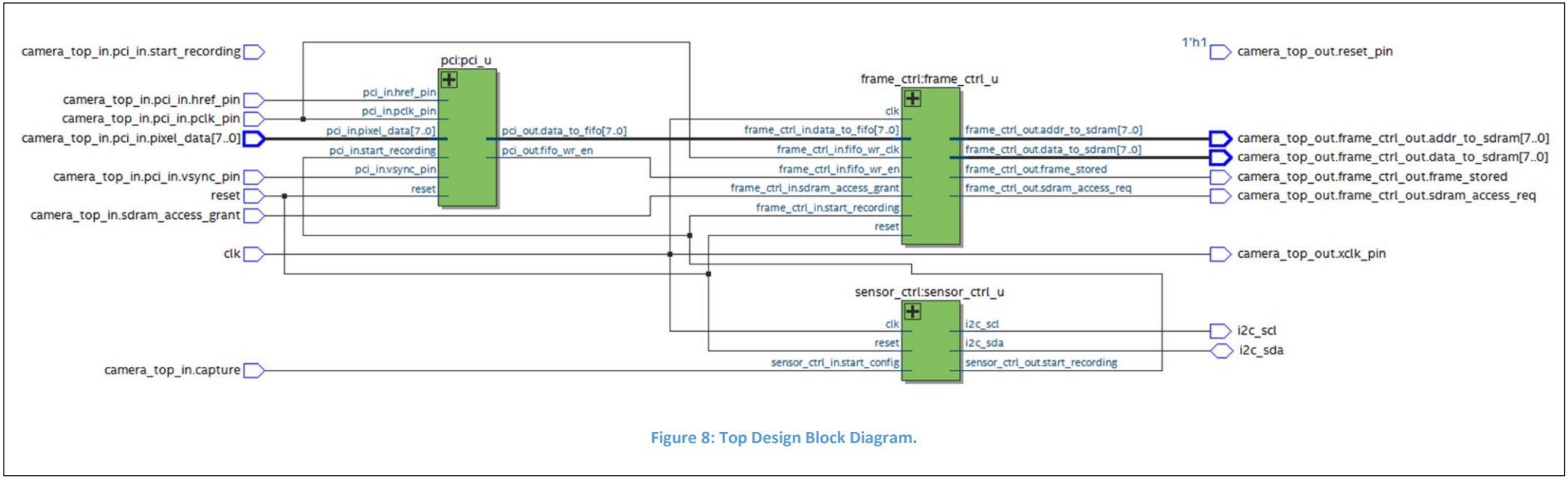


Figure 8: Top Design Block Diagram.

3.3 Units Design and Implementation:

This section contains a description of the implementation and detailed specifications for each unit of the design.

Note: the programming methodology used is structured-VHDL methodology by Jiri Gaisler. (6)

3.3.1 Sensor_ctrl unit:

This unit contains two subunits:

- I2C.
- register_map.

It is responsible for controlling I2C unit and makes sure that I2C unit requests all configuration data stored in the register_map unit and the camera interface is configured and ready to receive the picture data from the camera.

The block diagram for sensor_ctrl is depicted in *figure9*.

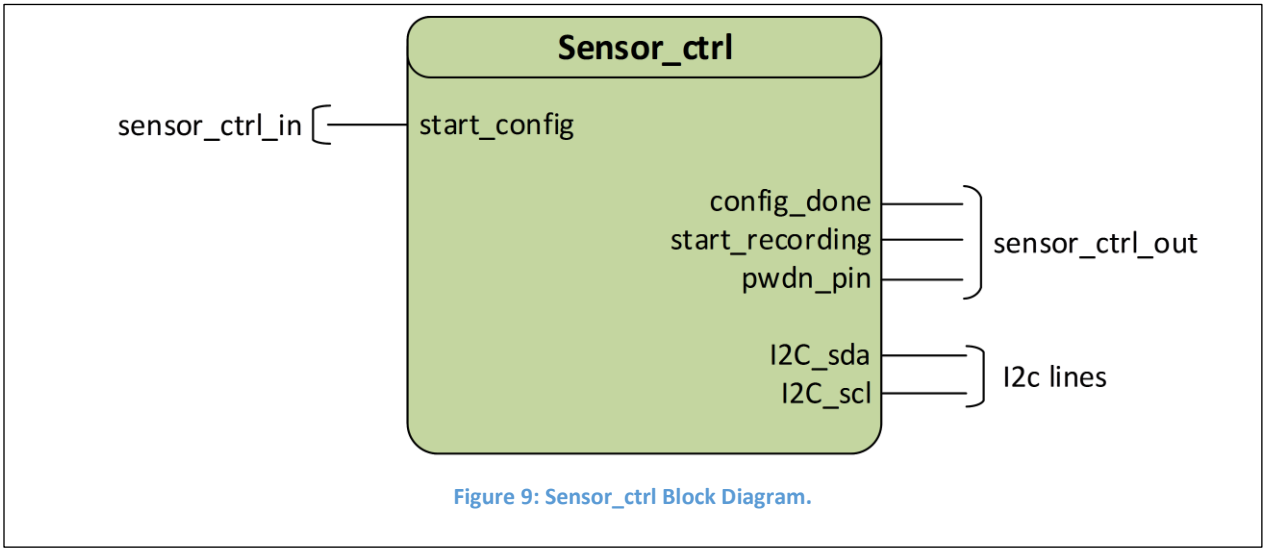


Figure 9: Sensor_ctrl Block Diagram.

3.3.1.1 **Unit Specifications:**

- **Ports and Internal Signals:**

sensor_ctrl - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
sensor_ctrl_in	start_config	Camera_top	sensor_ctrl	---	Spike to start the configuration of the sensor. This signal is asserted each time a picture wanted to be taken.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
sensor_ctrl_out	config_done	sensor_ctrl	---	'0'	Signal that indicates the configuration process has finished and the sensor is ready to capture a picture and stream it.
	start_recording	sensor_ctrl	frame_ctrl	'0'	Spike for pci and frame_ctrl to start receiving and recording streamed picture from the camera.
	pwdn_pin	sensor_ctrl	camera	'0'	Power down pin for the camera. ('0') ➔ power-up, ('1') ➔ power-down
I2C_line	I2C_sda	I2C	Camera	'Z'	I2C data line to transmit data serially to camera.
	I2C_scl	I2C	Camera	'1'	I2C clock line. Generated by I2C unit. Baud rate ➔ 200 kB.

sensor_ctrl - Internal Record Signals	
Signal Name	**Description**
sensor_ctrl_machine	FSM to configure the sensor.
i2c_in	Same as i2c_in in I2C unit.
register_map_in	Same as register_map_in in register_map unit.
start_recording	Same as start_recording in sensor_ctrl_out.
config_done	Same as config_done in sensor_ctrl_out.
pwdn_pin	Same as pwdn_pin in sensor_ctrl_out.
config_cnt	A counter for the number of the registers to be configured within the camera.
access_done	Same as access_done in i2c_out.

3.3.3.2 FSM:

Sensor_ctrl_machine is responsible for controlling the configuration process flow.

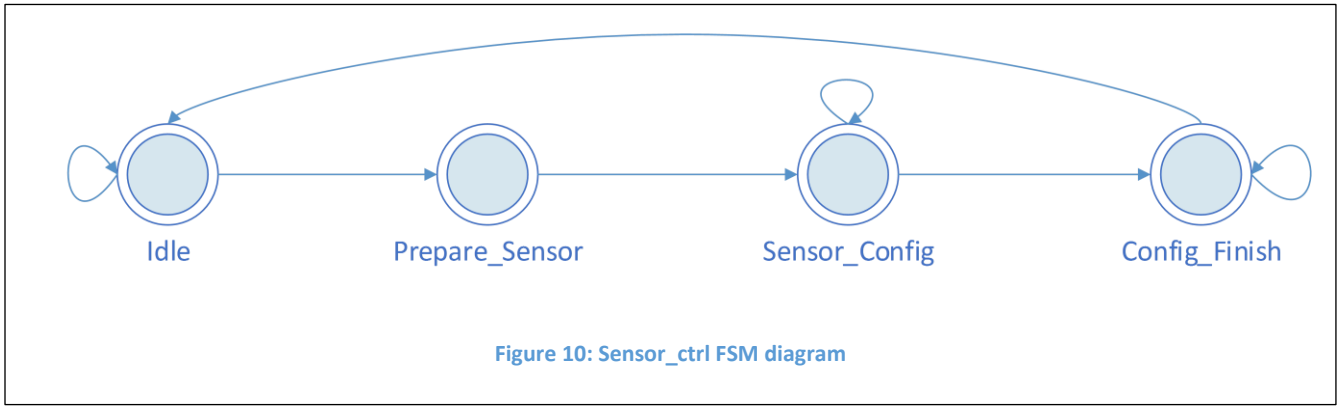


Figure 10: Sensor_ctrl FSM diagram

sensor_ctrl - sensor_ctrl_machine		
state	**Next state**	**Description**
idle	prepare_sensor	The machine waits for the start_config trigger to prepare the sensor for configuration.
prepare_sensor	sensor_config	Prepare the sensor by assert access_start and rw_n and reset the configuration data counter.
sensor_config	config_finished	Map the configuration data from register_map to I2C and re-accessing by asserting access_start until all configuration data in register_map have been read.
config_finished	idle	This state indicates that the configuration process has finished and the camera is ready to take the picture and stream it and a trigger to pci and frame_ctrl (start_recording) is asserted here to wake them up. Move back to idle state and wait for the next configuration request.

3.3.2 I2C Unit:

This unit is responsible to handle the control communication between the camera and the FPGA. It has only one task which is configuring the camera registers each time a picture wanted to be captured. The communication is done through SCCB protocol which is an I2C-like protocol (7).
Each time this unit receives access_start from the sensor_ctrl unit; it requests two values (register address and value) from the register_map unit and transmits them to the camera. This unit doesn't provide the possibility to read from the camera.

The block diagram for I2C is depicted in *figure11*.

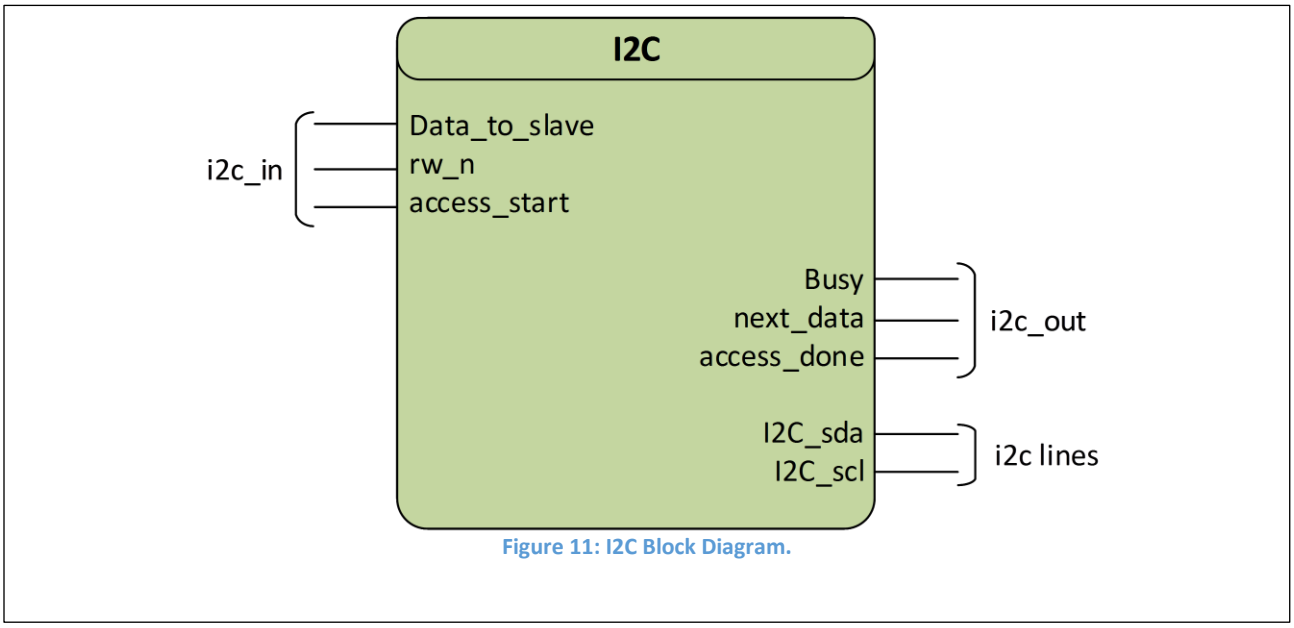


Figure 11: I2C Block Diagram.

3.3.2.1 **Unit Specifications:**

- **Ports and Internal Signals:**

I2C - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
i2c_in	data_to_slave	Register_map	Camera	---	Data to be sent to the camera through i2c_sda. This data (8-bits) is either the address of the register to be configured or the actual configuration value to be written into the addressed register.
	rw_n	Sensor_ctrl	Camera	'0'	Determine whether the operation is Write ('0') or Read ('1').

	access_start	Sensor_ctrl	I2C	'0'	Trigger to start accessing the camera for configuration through i2c.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
i2c_out	busy	I2C	---	'1'	Signal that indicates the status of the unit. ('0') ➔ ready, ('1') ➔ busy
	next_data	I2C	Register_map	'0'	Spike to enable next data byte from register_map.
	access_done	I2C	Sensor_ctrl	'0'	Spike that indicates the end of the access to the camera.
I2C_line	I2C_sda	I2C	Camera	'Z'	I2C data line to transmit data serially to camera.
	I2C_scl	I2C	Camera	'1'	I2C clock line. Generated by I2C unit. Baud rate ➔ 200 kB.

I2C - Internal Record Signals	
Signal Name	**Description**
bit_machine	FSM to handle bit transmitting and receiving bits from the camera through I2C.
phase_machine	FSM to handle the preparation of each transmission phase and feed the necessary data to bit_machine.
busy	Same as busy in i2c_out.
next_data	Same as next_data in i2c_out.
rw_n	Same as rw_n in i2c_in.
scl	Same as i2c_scl in i2c_line.
divisor_cnt	A counter to generate the i2c_scl clock.
trigger_bit_machine	A trigger to start bit_machine.
byte_done	Spike that indicates that the byte transmission has been done. Used to communicate between bit_machine and phase_machine.
access_done	Same as access_done in i2c_out.
trx_bit_cnt	A counter for transmitted bits within a byte.
trx_byte	Transmission byte register. A register to hold the data to be transmitted to the camera. Transmission is done by shifting the register to the left and feeding the i2c_sda line by the MSB of this register.
scl_en	Enable signal to control i2c_scl clock line.
Transmission	Output enable for i2c_sda line.
addr_cnt	A counter for register address. Used if the address for the register is more than 1 byte.

3.3.2.2 **FSM:**

There are two state machines in this unit:

- bit_machine state machine.
- phase_machine state machine.

Those two state machines together implement SCCB protocol.

The specification for SCCB is depicted in *figures12-13-14-15*. (7)

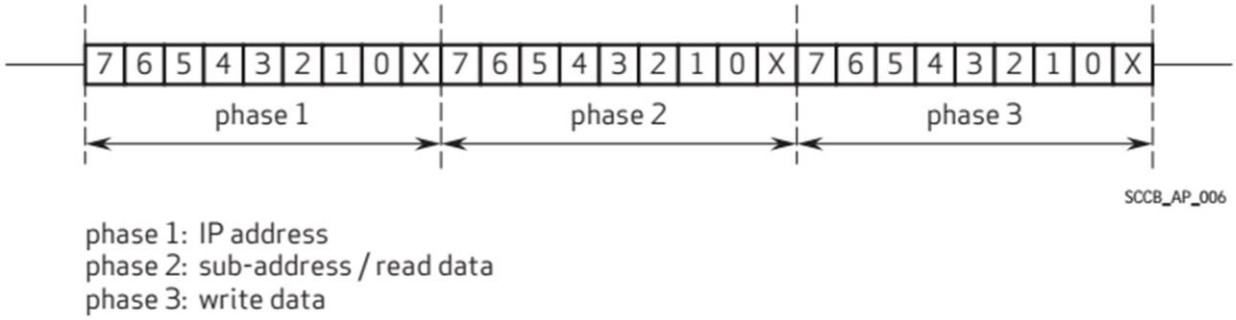


Figure 12: 3-Phase Write Transmission Cycle for SCCB.

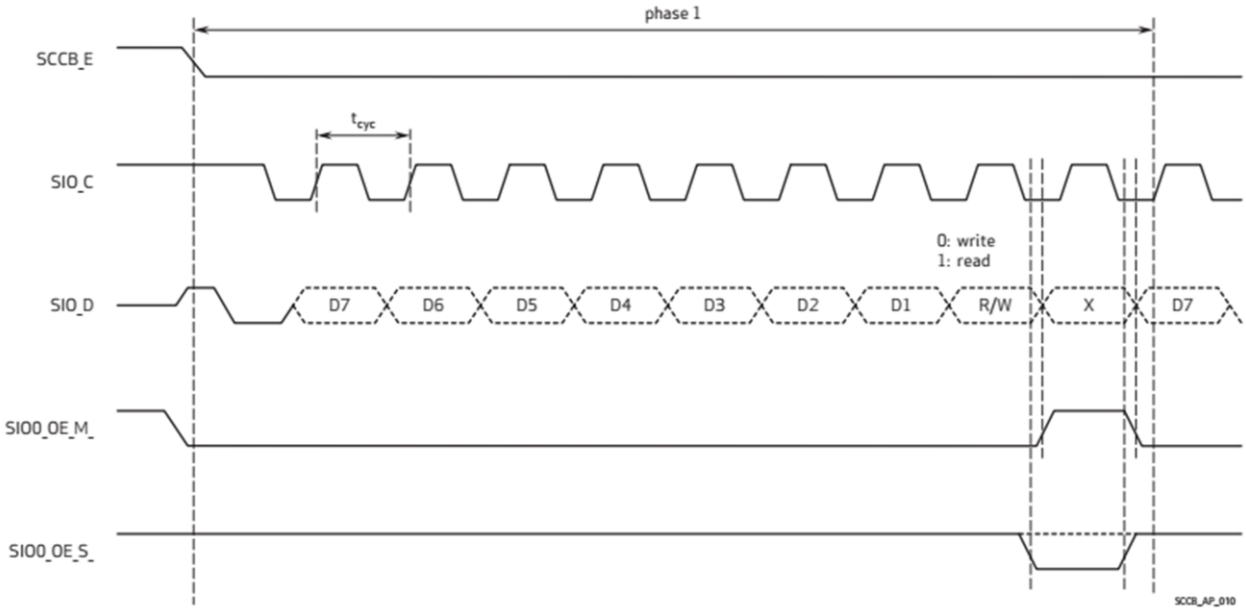


Figure 13: SCCB Phase-1 ID Address.

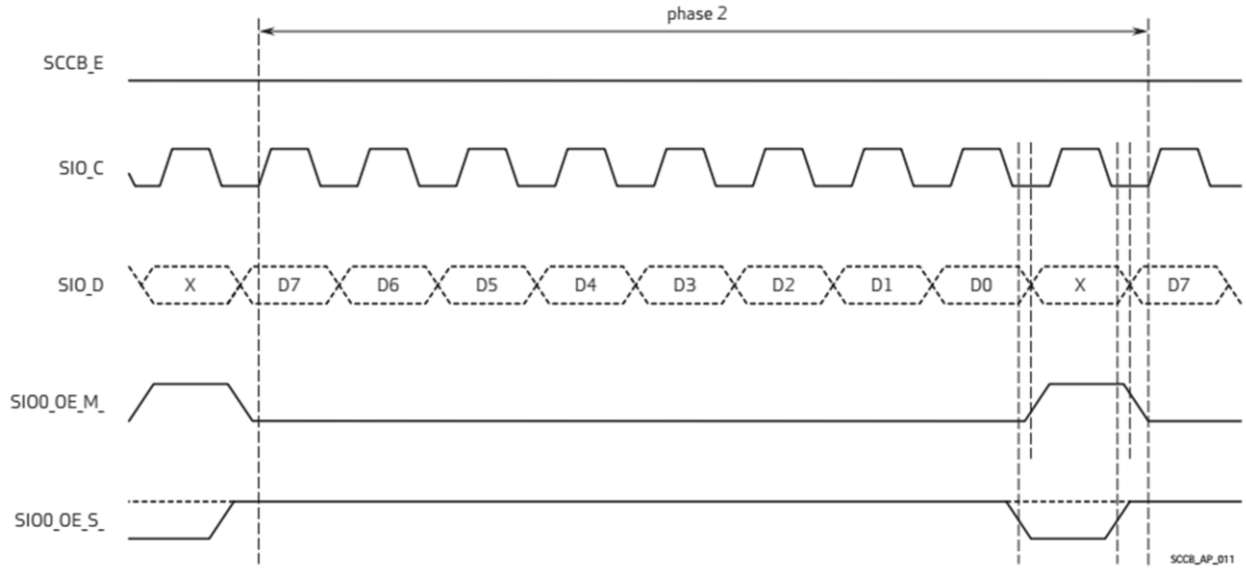


Figure 14: SCCB Phase-2 Sub Address.

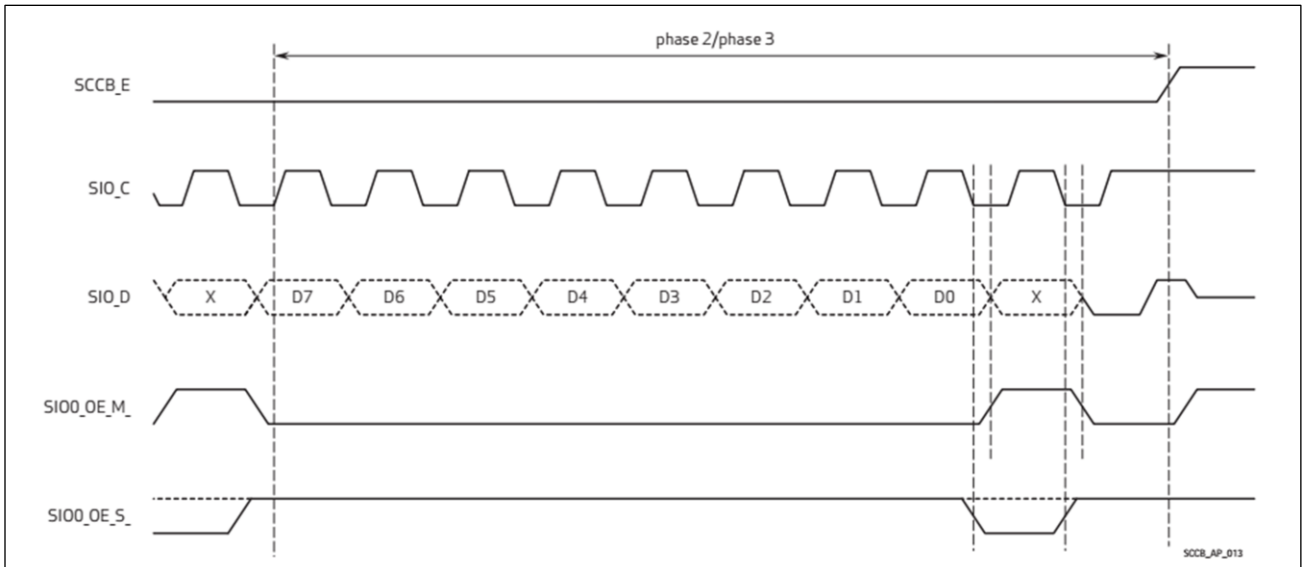


Figure 15: SCCB Phase-3 Write Data.

- **bit_machine FSM:** responsible for transmitting the data serially bit by bit to the camera according to SCCB protocol.

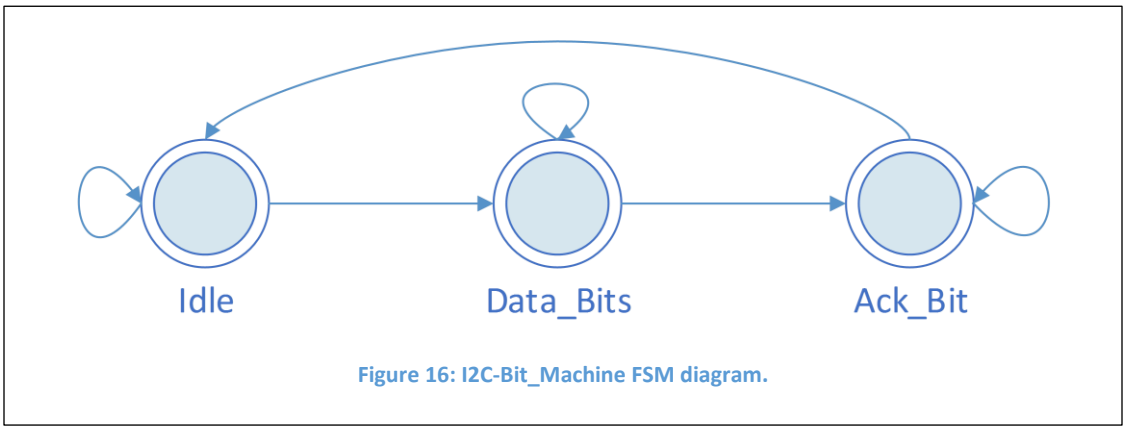


Figure 16: I2C-Bit_Machine FSM diagram.

I2C - bit_machine		
state	**Next state**	**Description**
idle	data_bits	In this state, the machine waits for the trigger to move to the data_bit state and start transmission.
data_bits	ack_bit	Transmit the data bit by bit by shifting the transmission byte register to the left in the middle of the low part of the i2c_scl clock. After transmitting 8 bits, move to the ack_bit state and assert next_data.
ack_bit	idle	Transmit the acknowledgment bit (‘1’) to the camera. Go back to idle state and assert byte_done to indicate that the transmission has done.

- **phase_machine FSM:** responsible for controlling bit_machine and implementing SCCB protocol.

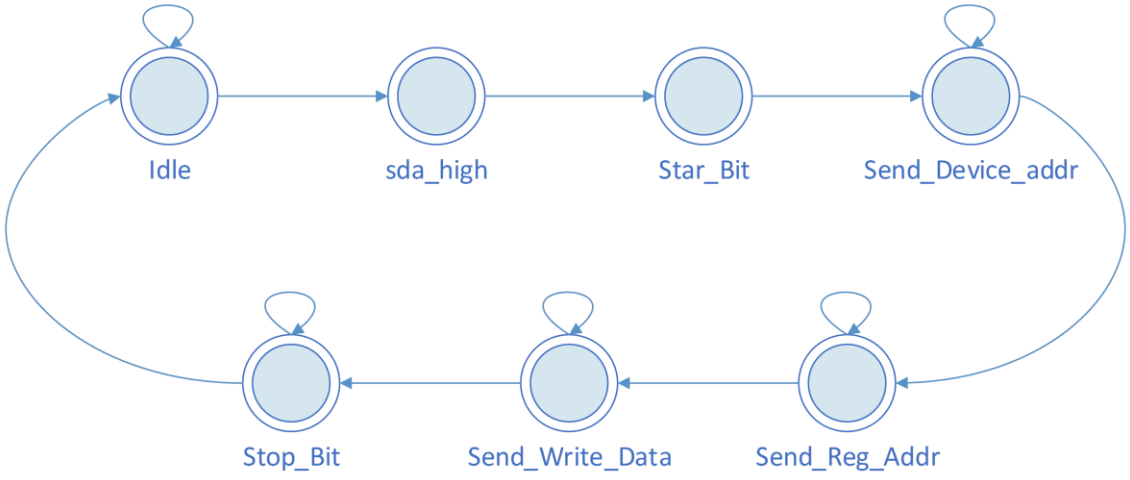


Figure 17: I2C-Phase_Machine FSM diagram.

I2C - phase_machine		
state	**Next state**	**Description**
idle	sda_high	The machine waits for the access_start trigger to start access to camera and Initialize transmission byte register by Sensor address and move to sda_high.
sda_high	start_bit	Assert i2c data line (i2c_sda) to '1', enable i2c_scl clock, Initialize the MSB bit of transmission byte register by '1' Start bit (High to low), and move to start_bit.
start_bit	send_device_addr	Transmit start_bit, trigger bit machine to transmit device address, and move to send_device_addr.
send_device_addr	send_reg_addr	Wait until device_address has been transmitted, Initialize transmission byte register by register address byte, trigger bit machine to transmit register address, and move to send_reg_addr.
send_reg_addr	send_write_data	Wait until register address has been transmitted, Initialize transmission byte register by register configuration value, trigger bit machine to transmit register configuration value, and move to send_write_data.
send_write_data	stop_bit	Wait until register configuration data has been transmitted, initialize the MSB bit of transmission byte register by '0' to drive i2c_sda to '0' after acknowledgment bit, and move to stop_bit.
stop_bit	idle	Release i2c_scl clock, initialize the MSB bit of transmission byte register by '0' Stop bit (Low to High), assert access_done, and move back to idle state and wait for next access request.

3.3.3 Register_map unit:

This unit is a configuration ROM. The ROM holds the camera's registers addresses and the configuration value for them. When the data (address or value) is requested by I2C unit, this unit responds by providing the data. It doesn't provide the possibility to read a specific address but the internal address counter starts from the beginning and is increased each time a data requested by I2C unit.
Note: Refer to the datasheet and implementation note to see a full description of the register set.

The block diagram for register_map is depicted in *figure18*.

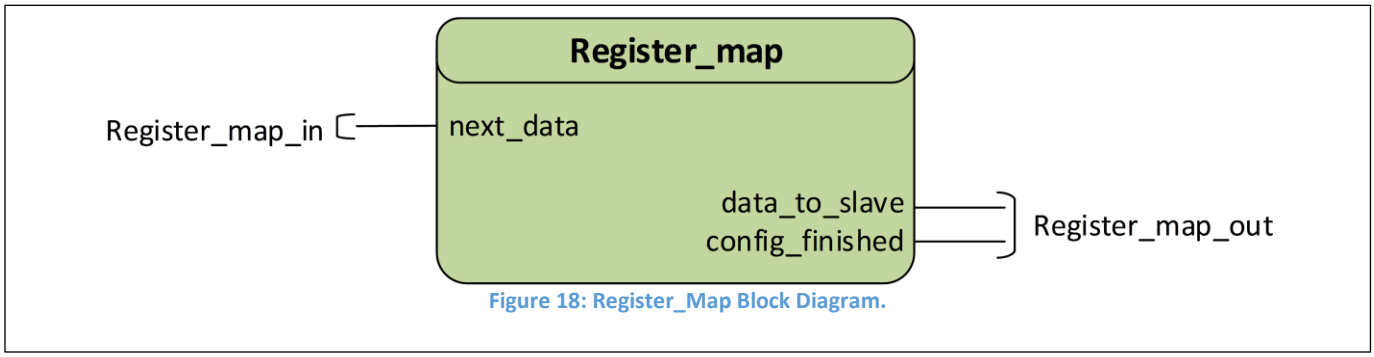


Figure 18: Register_Map Block Diagram.

3.3.3.1 **Unit Specifications:**

- **Ports:**

register_map - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
register_map_in	next_data	I2C	Register_map	'0'	Spike comes from I2C to enable next data byte from register_map.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
register_map_out	data_to_slave	Register_map	I2C	---	Configuration data to be sent to the camera through i2c_sda. This data (8-bits) is either the address of the register to be configured or the actual configuration value to be written into the addressed register.
	config_finished	Register_map	sensor_ctrl	'0'	Signal that indicates that all configuration data from register_map have been read.

3.3.4 Pci unit (Parallel Camera Interface):

This unit is responsible for receiving the picture data (pixel) byte by byte from the camera and storing it in the synchronization FIFO buffer. This unit is driven by the pixel clock *PCLK* from the camera, not by the system clock.

The block diagram for pci is depicted in *figure19*.

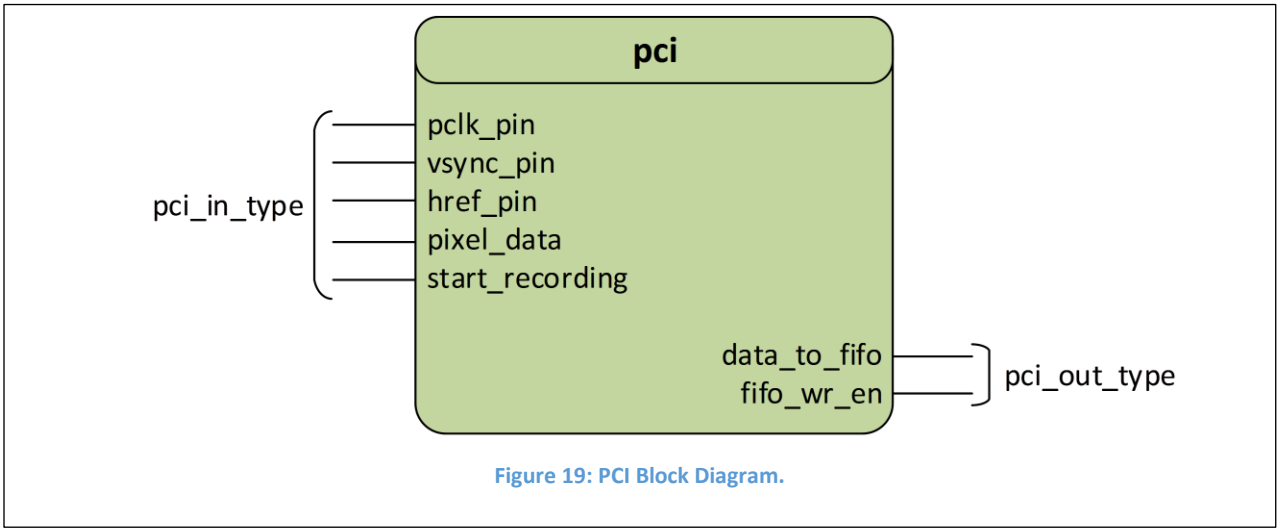


Figure 19: PCI Block Diagram.

3.3.4.1 **Unit Specifications:**

- **Ports and Internal Signals:**

pci_ctrl - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
	start_recording	sensor_ctrl	pci_ctrl	'0'	Spike Trigger generated by sensor_ctrl to start receiving the pixel data from the camera. It is asserted for one clock cycle after the camera has been configured and it is ready to stream the picture.
pci_in	pclk_pin	Camera	pci_ctrl	---	Pixel clock generated by the camera. The pixel data must be latched according to this clock. The frequency and sample edge can be set by configuring the related registers in the camera through i2c. See datasheet for more details.

	vsync_pin	Camera	pci_ctrl	‘0’	Vertical synchronization generated by the camera. This signal is asserted while streaming the frame. See datasheet for more details.
	href_pin	Camera	pci_ctrl	‘0’	Horizontal synchronization generated by the camera. This signal is asserted while streaming each row of the frame. See datasheet for more details.
	pixel_data	Camera	pci_ctrl	---	Streamed pixels’ byte by the camera. See datasheet for more details.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
pci_out	data_to_fifo	pci_ctrl	frame_ctrl	x“00”	Each received byte coming from the camera is buffered into the FIFO within frame_ctrl.
	fifo_wr_en	pci_ctrl	frame_ctrl	‘0’	Write enable signal for FIFO. This signal is asserted for each row of the frame and de-asserted between while moving from row to row within the frame and when there is no streaming.

pci - Internal Record Signals	
Signal Name	**Description**
frame_machine	FSM to control receiving the pictures’ frame from the camera.
row_machine	FSM to receive bytes within one row of the frame.
trigger_row_machine	Trigger from frame_machine to start row machine.
data_to_fifo	Same as data_to_fifo in pci_out.
fifo_wr_en	Same as fifo_wr_en in pci_out.
byte_received	Spike to indicate that the byte has successfully received.
row_received	Spike to indicate that the row has successfully received.
frame_received	Spike to indicate that the frame has successfully received.
row_bytes_cnt	A counter for bytes within one row of the frame.
rows_cnt	A counter of rows within one frame.

3.3.4.2 **FSM:**

There are two state machines in this unit. First, the row_machine state machine which is responsible for receiving the row byte by byte from the camera and storing them in the synchronization FIFO buffer. Second, frame_machine state machine which is responsible for controlling row_machine, receiving the picture frame row by row, and making sure that the frame has been received properly

- **Horizontal Timing specification:** represent the timing for the row within the frame.

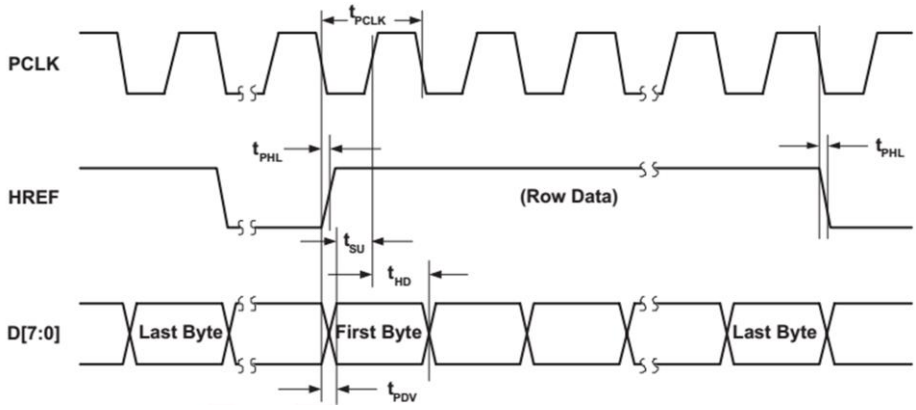


Figure 20: Horizontal Timing (Row Timing).

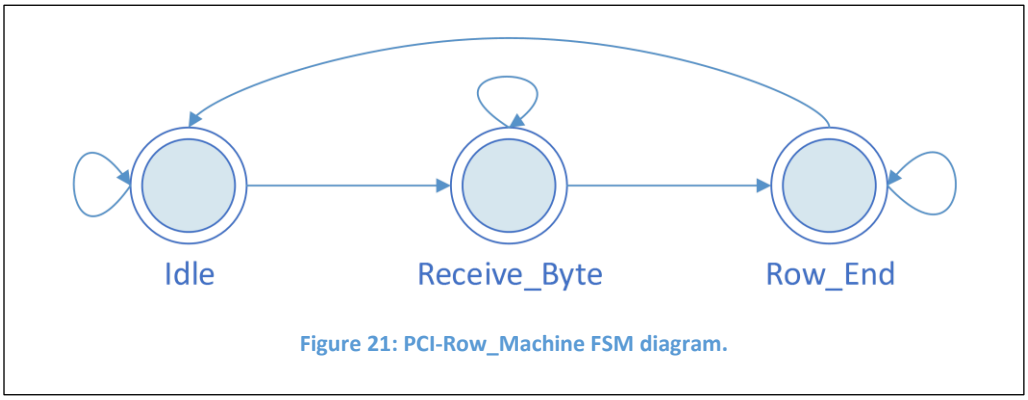


Figure 21: PCI-Row_Machine FSM diagram.

pci - row_machine		
state	**Next state**	**Description**
idle	receive_byte	The machine waits for a the trigger_row_machine trigger from frame_machine to start receiving rows' bytes.
receive_byte	row_end	Receive bytes and increase bytes' counter until all bytes within one row of the frame have been received and enable FIFO writing.
row_end	idle	Receiving one row has finished. Increase row counter and assert row_received. Move back to idle state and wait for next trigger.

- **Vertical timing specification:**

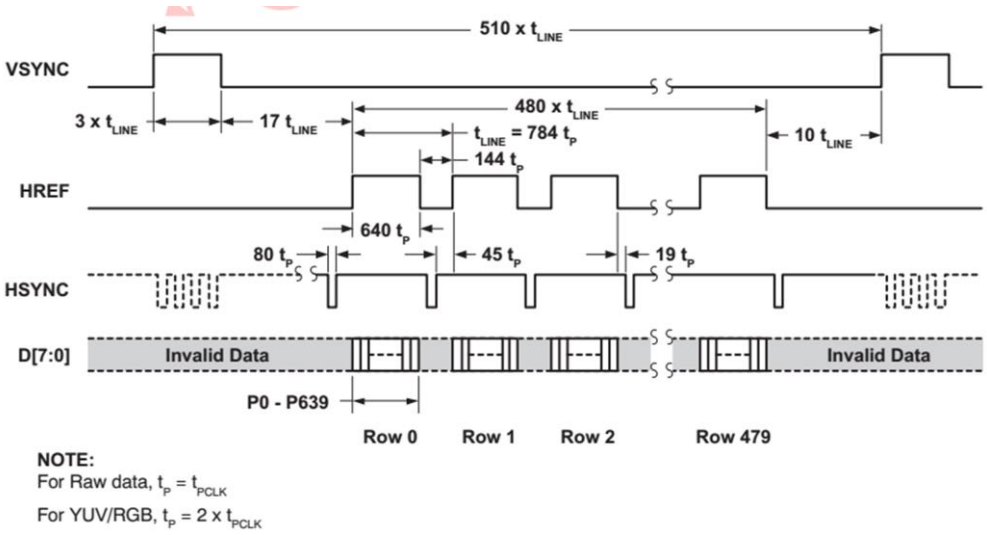


Figure 23: Vertical Timing (Frame Timing).

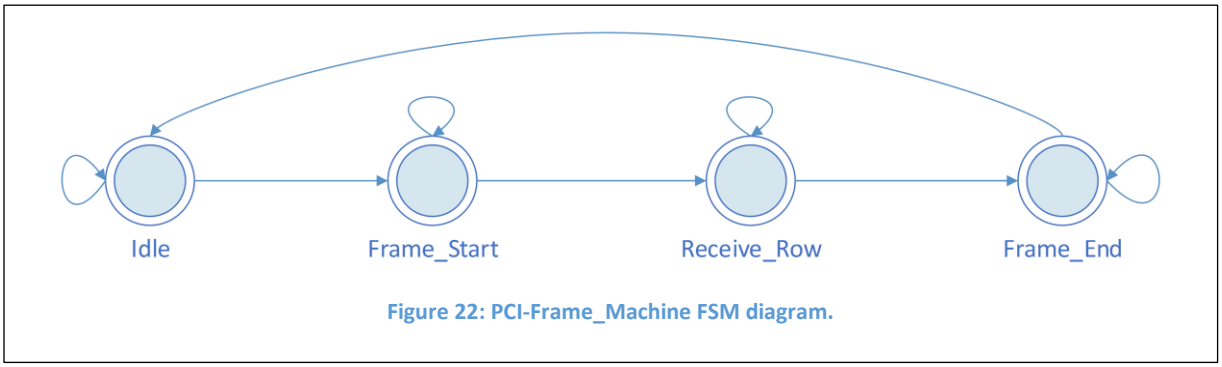


Figure 22: PCI-Frame_Machine FSM diagram.

pci - frame_machine		
state	**Next state**	**Description**
idle	frame_start	The machine waits for a start_recording trigger from sensor_ctrl to start receiving the pictures' frame.
frame_start	receive_row	The machine waits for the frame beginning and then triggers row_machine to start receiving the row.
receive_row	frame_end	The machine keeps receiving rows by keep triggering row_machine until all rows within one frame have been received.
frame_end	idle	The machine waits for the frame end and asserts frame_received. Move back to idle state and wait for next frame.

3.3.5 Frame_ctrl unit:

This unit is responsible for reading the picture data (pixel) byte by byte from the synchronization FIFO buffer and sending it to SDRAM controller. It makes sure that all picture bytes have been read from FIFO and sent to SDRAM controller. The FIFO IP is instantiated inside this unit. The block diagram for frame_ctrl and FIFO is depicted in *figure24* and *figure25* respectively.

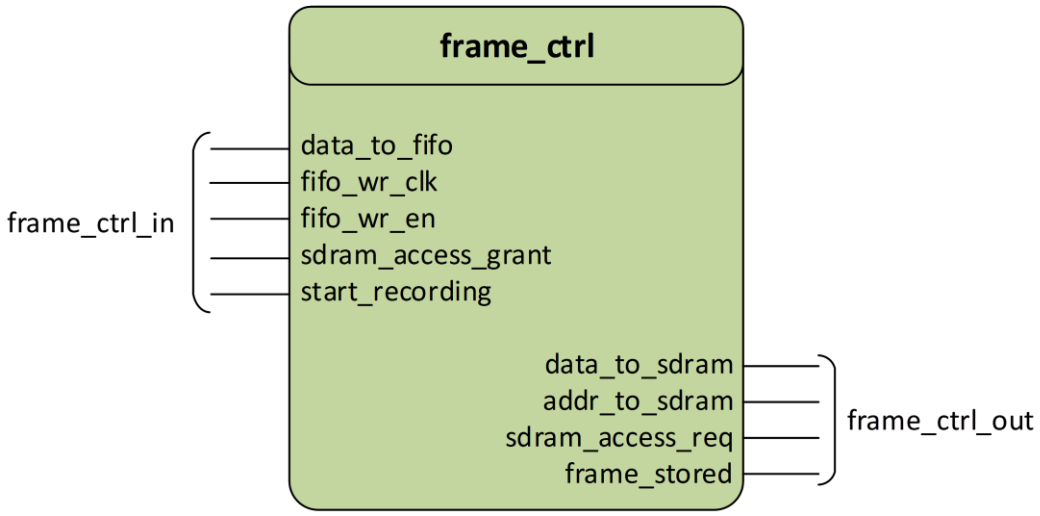


Figure 25: Frame_ctrl Block Diagram.

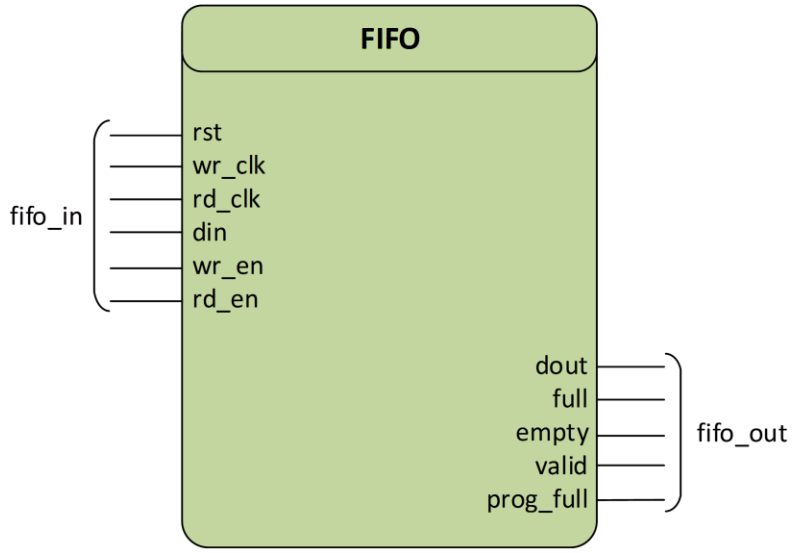


Figure 24: FIFO Block Diagram.

3.3.5.1 **Unit Specifications:**

- **Ports and Internal Signals:**

frame_ctrl - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
Frame_ctrl_in	data_to_fifo	pci	FIFO	---	Received byte coming from pci.
	fifo_wr_clk	Camera	FIFO	---	Writing clock for FIFO. This is the same as PCLK (pixel clock) generated by the camera.
	fifo_wr_en	pci	FIFO	'0'	Write enable signal for FIFO. See pci_out.
	sdram_access_grant	sdram_ctrl	frame_ctrl	'0'	Access grant signal received from SDRAM in response for SDRAM access request sdram_access_req. this signal indicates that SDRAM is ready to receive data.
	start_recording	sensor_ctrl	frame_ctrl	'0'	Spike Trigger generated by sensor_ctrl to start receiving the pixel data from the camera. It is asserted for one clock cycle after the camera has been configured and it is ready to stream the picture.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
Frame_ctrl_out	data_to_sdram	frame_ctrl	sdram_ctrl	x"00"	Pixel byte read from FIFO and to be written in SDRAM.
	addr_to_sdram	frame_ctrl	sdram_ctrl	start_addr - 1	Address for SDRAM to store the pixel byte. Incremented automatically after each writes to SDRAM.
	sdram_access_req	pci	frame_ctrl	'0'	Access request signal to SDRAM. This signal is asserted for each byte need to be stored in SDRAM.
	frame_stored	frame_ctrl	Camera_top	'0'	Signal to indicate that the frame has been completely stored in SDRAM.

FIFO - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
FIFO Inputs	rst	frame_ctrl (Internal)	FIFO	'0'	Reset signal to FIFO.
	wr_clk	Pci → frame_ctrl	FIFO	---	Same as fifo_wr_clk in frame_ctrl_in.
	rd_clk	PLL	FIFO	---	System clock.
	din	Pci → frame_ctrl	FIFO	X"00"	Same as data_to_fifo in frame_ctrl_in.
	wr_en	Pci → frame_ctrl	FIFO	'0'	Same as fifo_wr_en in (Internal)
	rd_en	frame_ctrl (Internal)	FIFO	'0'	Same as fifo_rd_en in internal record in frame_ctrl_in.
Signal Name		**Source**	**Sink**	**Reset Value**	**Description**
FIFO Outputs	dout	frame_ctrl	sdram_ctrl	x"00"	Same as data_to_sdram in frame_ctrl_out.
	full	FIFO	frame_ctrl	'0'	FIFO Full flag (Not used).
	empty	FIFO	frame_ctrl	'0'	FIFO Empty flag (Not used).
	valid	FIFO	frame_ctrl	'0'	Valid flag, data on *dout* is ready.
	prog_full	FIFO	frame_ctrl	x"03"	Programmable full flag. (Read from FIFO cannot be done unless there are 3 bytes in the FIFO)

frame_ctrl - Internal Record Signals	
Signal Name	**Description**
frame_ctrl_machine	FSM to control storing the frame in SDRAM.
data_to_sdram	Same as data_to_sdram in frame_ctrl_out.
addr_to_sdram	Same as addr_to_sdram in frame_ctrl_out.
sdram_access_req	Same as sdram_access_req in frame_ctrl_out.
frame_stored	Same as frame_stored in frame_ctrl_out.
fifo_rd_en	Read enable signal for FIFO.
fifo_rst	FIFO reset signal.
frame_bytes_cnt	A counter for bytes within one frame.
FIFO - Internal Signals	
Signal Name	**Description**
data_from_fifo	Same as *dout* in FIFO outputs.
fifo_valid_flag	Same as valid in FIFO outputs.
fifo_pfull_flag	Same as prog_full in FIFO outputs.
fifo_full_flag	Same as full in FIFO outputs.
fifo_empty_flag	Same as empty in FIFO outputs.

3.3.5.2 **FSM:**

frame_machine is responsible for handling the communication between FIFO and SDRAM controller.

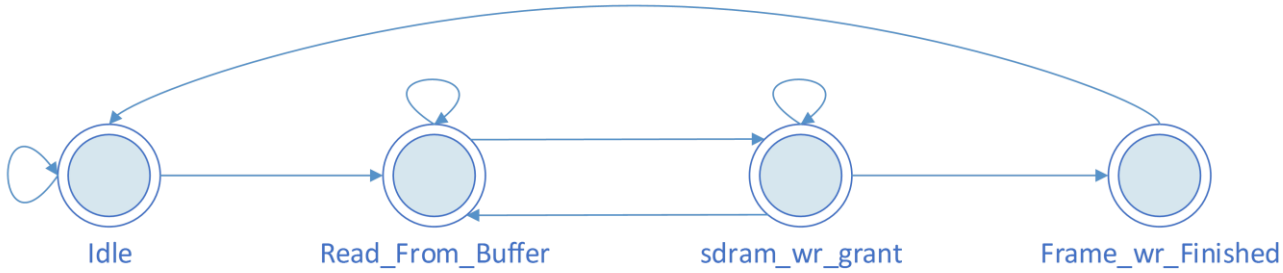


Figure 26: Frame_ctrl_Machine FSM diagram.

frame_ctrl - frame_machine		
state	**Next state**	**Description**
idle	read_from_buffer	The machine waits for a start_recording trigger from sensor_ctrl to start reading from the FIFO and storing in the SDRAM. Initialize addr_to_sdram.
read_from_buffer	sdram_wr_grant	Read a byte from FIFO if it has enough bytes and send a request to access from SDRAM.
sdram_wr_grant	frame_wr_finished	The machine waits for SDRAM grant and a valid data signal from FIFO to write the read byte to the SDRAM. If not all bytes have been stored in SDRAM the machine goes back to the read_from_buffer state, otherwise, the frame storing has been finished.
frame_wr_finished	idle	Assert frame_stored signal to indicate that the frame has been stored and go back to idle state and wait for next frame.

3.3.5.3 **FIFO:**

The FIFO serves as a synchronization buffer. Since the *pci* unit is clocked by *PCLK* and *frame_ctrl* is clocked by the system clock, this FIFO buffer makes sure that the synchronization is achieved. The size of FIFO depends on the worst case of SDRAM response to a request signal.

This FIFO is implemented using FIFO Generator in Vivado 2016.4 with the features in the table below:

Feature	**Option**
Clocking Scheme	Independent Clocks
Memory Type	Block RAM
Model Generated	Behavioral Model
Write Width	8
Write Depth	63
Read Width	8
Read Depth	63
Almost Full/Empty Flags	Not Selected/Not Selected
Programmable Full/Empty Flags	Selected/Not Selected
Data Count Outputs	Not Selected
Handshaking	Selected
Read Mode / Reset	Standard FIFO / Asynchronous
Read Latency (From Rising Edge of Read Clock)	1

Refer to FIFO product guide for more details about the FIFO specifications. (8)

Note: the size of FIFO and the programmable full flag must be adjusted to a different value if the clock frequency used or the worst case response of SDRAM controller have modified.

3.3.6 Camera_top unit:

This unit is the top design which all other units are instantiated and mapped together inside it.

The block diagram for camera_top and FIFO is depicted in *figure27*

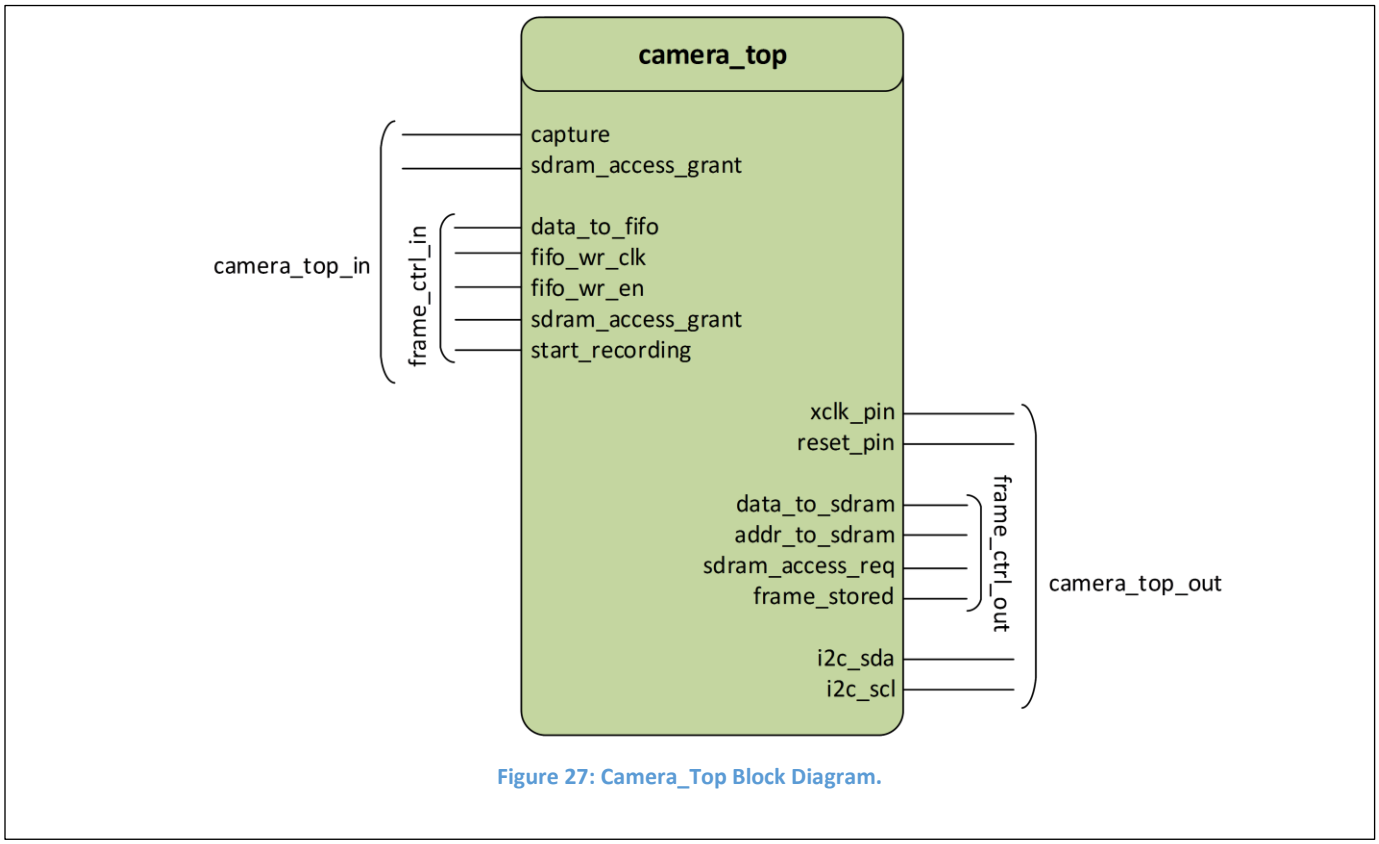


Figure 27: Camera_Top Block Diagram.

Unit Specifications:

- **Ports:**

Camera_top - Ports					
Signal Name		**Source**	**Sink**	**Default Value**	**Description**
Camera_top_in	capture	main controller (ARM)	Sensor_ctrl	---	Signal from main controller to start taking a picture.
	sdram_access_grant	sdram_ctrl	frame_ctrl	---	Same as sdram_access_grant frame_ctrl_in.
	pci_in	Camera and sensor_ctrl	pci	---	Same as pci_in in pci unit.

Signal Name		Source	Sink	Reset Value	Description
Camera_top_out	frame_ctrl_out	frame_ctrl	sdram_ctrl and main controller	---	Same as frame_ctrl_out in frame_ctrl unit.
	xclk_pin	PLL	camera	---	Clock to the camera.
	reset_pin	camera_top	camera	always '1'	Camera reset pin. Must be driven by '1' so the camera run in the normal mode.

3.3.7 Design Files:

File Name	Description
Camera_top.vhd	Top design which includes all components instantiations.
camera_pkg.vhd	Package which includes type's declarations and constants.
sensor_ctrl.vhd	Sensor control interface which include i2c and register_map.
I2c.vhd	SCCB protocol which is I2c_like.
Register_map.vhd	Configuration ROM that contains registers addresses and configurations values.
Pci.vhd	Parallel Camera Interface to receive the data from camera.
Frame_ctrl.vhd	Frame control to control receiving the frame and transmit it to SDRAM controller.

4- Testing:

- Configuring the camera has been tested and verified. This includes ***i2c***, ***register_map***, and ***sensor_ctrl*** units.
- The rest of the units ***(pci, frame_ctrl, FIFO, and top_camera)*** have not been tested yet.
- The current configuration ROM contains 19 value (configure 19 registers). The rest of the registers are kept on their default value. Refer to datasheet and application note for more details. (5) (9)

References

1. Xilinx. *Xilinx.* [Online] Xilinx. https://www.xilinx.com/.

2. **Christoph Greb, Peter Laskey, Karin Schwab.** Introduction to Digital Camera Technology. *www.leica-microsystems.com.* [Online] February 02, 2016. [Cited: July 19, 2017.] http://www.leica-microsystems.com/science-lab/microscope-cameras/introduction-to-digital-camera-technology/.

3. CMOS Fundamentals. *www.siliconimaging.com.* [Online] [Cited: July 19, 2017.] http://www.siliconimaging.com/cmos_fundamentals.htm.

4. Imaging Electronics 101: Understanding Camera Sensors for Machine Vision Applications. *www.edmundoptics.com.* [Online] [Cited: July 19, 2017.] https://www.edmundoptics.com/resources/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/.

5. **OmniVision.** *ov7670 datasheet.* s.l. : OmniVision.

6. **Gaisler, Jiri.** Cobham Gaisler AB. *www.gaisler.com.* [Online] [Cited: July 26, 2017.] www.gaisler.com/j25/doc/vhdl2proc.pdf , www.gaisler.com/doc/structdesign.pdf , ens.ewi.tudelft.nl/Education/courses/et4351/structured_vhdl.pdf.

7. **OmniVision.** *OmniVision Serial Camera Control Bus (SCCB) Functional Specification.* 2007.

8. **Xilinx.** *FIFO Generator v13.1.* 2017.

9. **OmniVision.** *OV7670/OV7171 CMOS VGA (640x480) CameraChip™ Implementation Guide.* 2005.

10. —. OmniVision. *OmniVision.* [Online] OmniVision. http://www.ovt.com/.

11. **(FTDI), Future Technology Devices International Limited.** *What is the Camera Parallel?* s.l. : http://www.ftdichip.com/, 2015.

12. **Xilinx.** *Xilinx 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide for HDL Designs.* 2013. UG768 (v14.7).