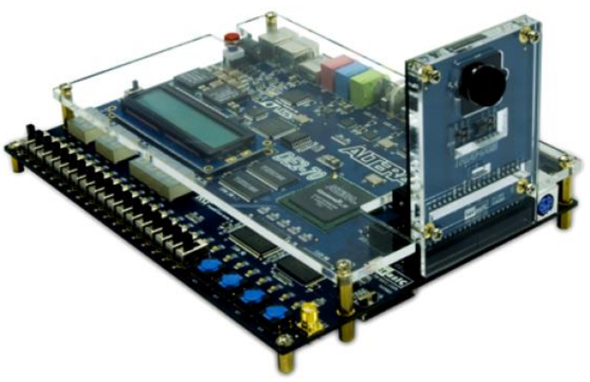
User Manual

CMOS TRDB-D5M Camera

and

8MB SDRAM A2V4S40CTP



Index

1. Contextualization

This document has the objective of describing the fundamental aspects of operation of the TRDB-D5M camera from Terasic and of the SDRAM A2V4S40CTP present in the FPGA Cyclone II device. The specific design of the SDRAM 4-Port Controller provided in the TRDB-D5M camera's demo will also be explored.

Firstly we will talk about the internal operations of the TRDB-D5M camera and the main configuration parameters/registers defining its operation. The main features of the demo provided by Terasic will be explained in detail.

The SDRAM operation explanation will follow an identic structure. There is a first approach to the internal operation of the SDRAM and its control signals, followed by an explanation of the specific SDRAM controller available in the demo.

It should be noted that the reading of this document doesn't exempt from checking the datasheets and manuals of the two components.

This document was produced alongside with the development of a digital systems' project which used the TRDB-D5M and this SDRAM model [5].

2. TRDB-D5M CMOS Camera

This section will introduce the key concepts of TRDB-D5M operation. In a first phase, the main concepts related to the structure of pixel array generated by the CMOS sensor will be introduced. Then, specific information about the most relevant timing and formatting parameters will be provided.

After this contextualization some functions provided by the camera will be approached, focusing on pixel clock adjustment and possible readout modes (skipping, binning, mirroring). These features allow frame rate variation and resolution reduction without decreasing the field of view (FOV), respectively.

Finally, the actual change of cameras register configurations in the demo's Verilog files will be explained, with a specific example on pixel clock adjustment through the use of the PLL.

2.1. Pixel Array Format – Key Concepts

Active Region, Dark Rows and Dark Columns

The pixel array generated by the TRDB-D5M CMOS sensor has 2752 columns and 2004 rows. The address (column 0, row 0) corresponds to the upper right corner of the array. The **active region** (central region) has 2592 columns and 1944 rows by default, and is surrounded by an **active boundary**, which in its turn is surrounded by a boundary of **dark pixels**. The values of the composing pixels are generated according to the Bayer pattern – Green 1, Green 2, Red and Blue (G1,G2,R,B). The default reading order is specified on the rightmost image of figure 2.1.

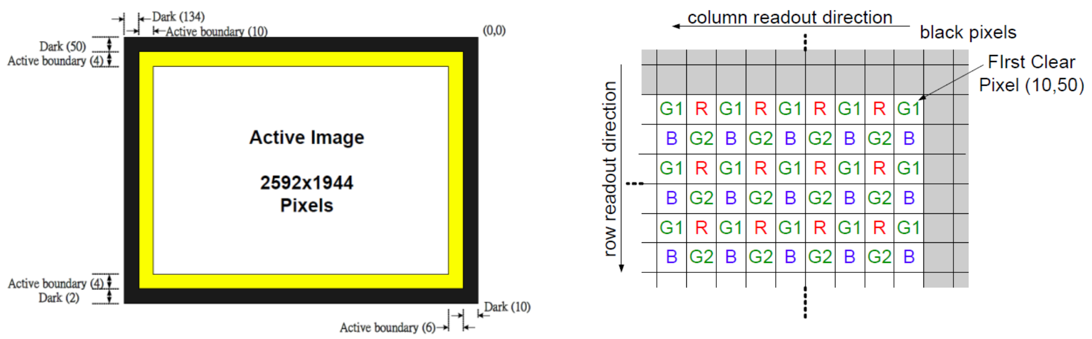


Figure 2.1. In the left image , the pixel array structure is illustrated; In the right image, the readout directions and Bayer patterns are represented.

Valid Region, Vertical Blanking and Horizontal Blanking and the signals LINE_VALID and FRAME_VALID

The frame pixels are read in a progressive scan. Valid data from an image are surrounded by regions of horizontal and vertical blanking , as shown in figure 2.2.

- The signal LINE_VALID, which defines the boundaries between rows of the active region, is HIGH on the Valid Region of the image.
- The signal FRAME_VALID defines the vertical active region of a frame.
- A pixel is valid when LINE_VALID and FRAME_VALID are simultaneously active. If FRAME_VALID is LOW, there's vertical blanking occurring and if LINE_VALID is LOW, there's horizontal blanking occurring – see figure 2.3.

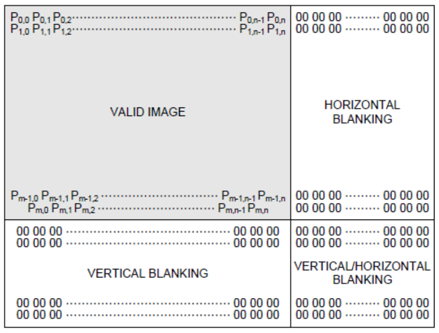
Additional considerations must be taken if the reading of dark rows/columns is active. It is also possible to change the relative operation of the signals FRAME_VALID and LINE_VALID. There are three different available formats; in the default behavior, LINE_VALID is deactivated whenever FRAME_VALID is inactive – see figure 1.5.

The pixel clock - PIXCLK

The clock signal PIXCLK dictates the sampling rate of pixel data – for each cycle a new 12 bit value is generated on the data pins (DOUT) – see figure 1.4. It is generally equal to the period of the external clock, EXTCLK.

Readout Window

The readout window is generally defined in a way so that it contains only active pixels. The user can also read the dark regions, but for that purpose one must take additional considerations regarding the readout order of these regions.



$P_{0,0}$ $P_{0,1}$ $P_{0,2}$ ······ $P_{0,n-1}$ $P_{0,n}$
$P_{1,0}$ $P_{1,1}$ $P_{1,2}$ ······ $P_{1,n-1}$ $P_{1,n}$
VALID IMAGE
$P_{m-1,0}$ $P_{m-1,1}$ $P_{m-1,2}$ ······ $P_{m-1,n-1}$ $P_{m-1,n}$
$P_{m,0}$ $P_{m,1}$ $P_{m,2}$ ······ $P_{m,n-1}$ $P_{m,n}$
HORIZONTAL BLANKING
VERTICAL BLANKING
VERTICAL/HORIZONTAL BLANKING
00 00 00 ······ 00 00 00

Figure 2.2. Spatial illustration of image readout.

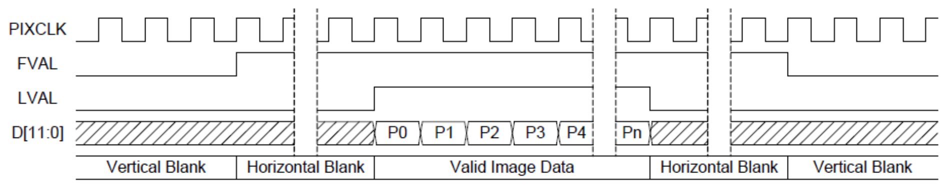


Figure 2.3. Default Pixel Output Timing.

Timing and Formatting Parameters

Table 1 presents the essential formatting and timing related parameters for the camera's generated images, and how their values can be computed. Some of the values referred to in their calculation (such as *Column_Size, Row_Size, Column_Skip, Row_Skip, etc.)* correspond to values of camera's internal registers which are directly configurable by editing the respective value on demo files, which can be consulted in section 2.3. *Camera Registers and Register Description* . The graphic meaning of some of these parameters is depicted in figure 2.4.

Parameter	Value
Output image Width (W)	$W = \frac{2.ceil(Column\_Size + 1)}{(2\ (Column\_Skip + 1))}$
Output image Height (H)	$H = \frac{2.ceil(Row\_Size + 1)}{(2\ (Row\_Skip + 1))}$
Horizontal Blanking (HB)	$HB = Horizontal\_Blank + 1$
Vertical Blanking (VB)	$VB = Vertical\_Blank + 1$
Minimum Horizontal Blanking (HBMIN):	$HBMIN = 208 \times (Row\_Bin + 1) + 64 + \frac{WDC}{2}$
Minimum vertical Blanking (VBMIN)	$VBMIN = max(8, SW - H) + 1$
Shutter Width (SW)	$SW = max(1, (2 * 16 \times Shutter\_Width\_Upper) + Shutter\_Width\_Lower)$
Row time (t_ROW)	$t_{ROW} = 2 \times t_{PIXCLK} \times max(((W/2) + max(HB, HBMIN)), (41 + 208\,.\,(Row\_Bin + 1) + 99))$
Frame Time (tFRAME)	$t_{FRAME} = (H + max(VB, VBMIN)) \times tROW$

Table 1. Timing and Formatting Parameters

Additional observations:

- The HBMIN values are dependent of the binning configurations for rows and columns. *Width Dark Columns* (WDC) consists of the width occupied by the dark columns. It is important to stress out that the *Column_bin* value has an impact on the WDC value, according to the following table:

Column_Bin	WDC (after binning)
0	80
1	40
3	20

Table 2. WDC values for different values of column binning.

The Height of Dark Rows (HDR) is 8 for any value of *Row_Bin*.

- t_{ROW} is the period between the sampling of the first pixel of a row and the first pixel of the following row.

- t_{FRAME} corresponds to the duration of a frame.

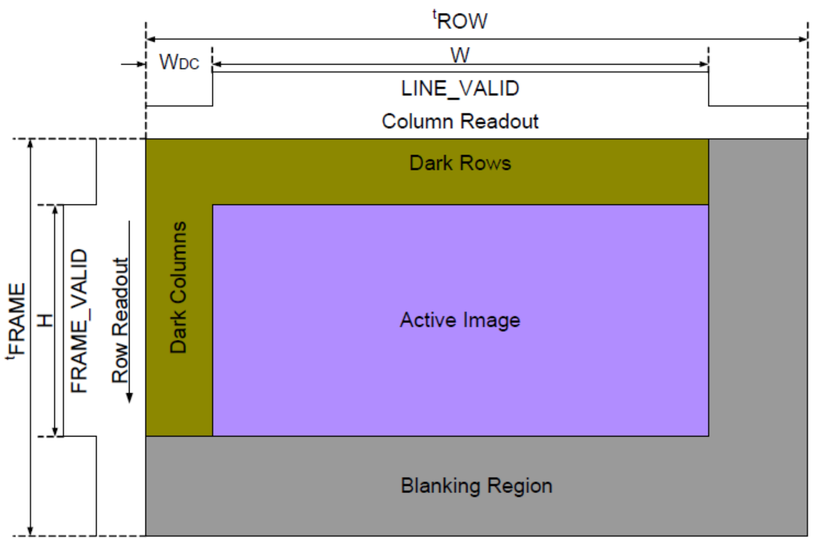


Figure 2.4. Frame Timing Parameters

2.2. Additional Features – Clock Division, Skipping, Binning and Mirroring

The TRDB-D5M camera offers pixel clock adjustment and different readout modes. In this section, the means of clock adjustment and the different available readout modes will be explored. Details on camera's internal registers associated with this features can be consulted on section *2.3. Camera Registers and Register Description*.

2.2.1. Clock Division

It is possible to change the clock signal's frequency by directly altering the value of the register *Divide_Pixel_Clock* , or by using the PLL for scaling:

- **Clock division using *Divide_Pixel_Clock:***

The setting of a non-null value to the *Divide_Pixel_Clock* register results in the internal division of the external clock input (XCLKIN).

$$f_{PIXCLK}\,(\mathrm{Hz}) = \begin{cases} f_{XCLKIN} & se\ \mathrm{Divide\_Pixel\_Clock} = 0 \\ \dfrac{f_{XCLKIN}}{(2 \times \mathrm{Divide\_Pixel\_Clock})} & \text{caso contrário} \end{cases}$$

- **Clock division using PLL:**

The PLL contains:

- Pre-scaler: divides the clock input signal XCLKIN.
- VCO: applied gain to the output of the pre-scaler.
- An additional stage to further divide the output from VCO. This stage outputs the final clock.

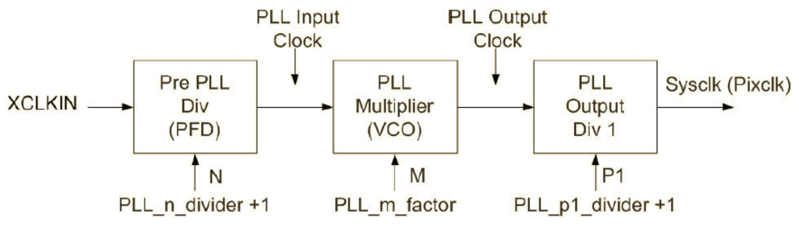


Figure 2.5. PLL Generated Master Clock.

The PLL control registers can be modified to obtain the desired clock frequency.

PLL utilization rules:

- XCLKIN $\in [\,6\,,27\,]\ MHz$
- The PLL must be activated by activating the *Power_PLL* bit (Reg0x10[0] = 1 <-> See section 2.3. Camera Registers and Register Description)
- Assign the values to the *PLL_m_Factor*, *PLL_n_Divider* and *PLL_p1_Divider* registers to get the desired PIXCLK value from XCLKIN, through the following equation:

$$f_{PIXCLK} = (\,f_{XCLKIN} \times M\,)\,/\,(\,N \times P1\,)$$

where

$$\begin{aligned} M &= PLL\_m\_Factor \\ N &= PLL\_n\_Divider + 1 \\ P1 &= PLL\_p1\_Divider + 1 \end{aligned}$$

Note: If P1 is odd (*PLL_P1_Divider* is even), the internal clock duty cycle of the system won't be 50:50.

2.2.2. Reading modes – Skipping, Binning and Mirroring

Subsampling – Skipping and Binning

The valid image window, which defines the FOV, is defined by four register fields:

- *Column_start* and *Row_start* define the X and Y pixel coordinates of the upper left corner of the FOV. They should be even numbers.
- *Column_size* defines the FOV width in pixels. Should be an odd value.
- *Row_Size* defines the FOV height in pixels. Should be an odd value.

The *row skip* and *column skip* readout modes use subsampling in order to reduce the resolution of the generated images without FOV reduction. There are also binning modes to reduce the aliasing impact of skipping through the use of averaging of pixels in adjacent columns/rows.

If skipping is active (different than 0) full lines and full rows aren't sampled, resulting in a lower resolution image. The rows and columns are always read in pairs. Figure 2.6 shows an example of a 2X column skip.

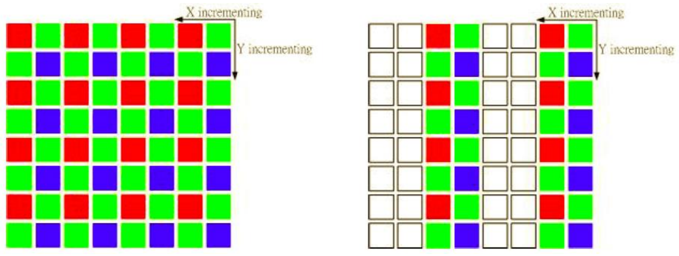


Figure 2.6. In the left image, pixel readout with no skipping; In the right image, skipping readout modefor Column Skip 2X.

The size of the resulting image after skipping is thus given by:

$$W = 2.ceil((Column\_Size + 1) / (2.(Column\_Skip + 1)))$$

$$H = 2.ceil((Row\_Size + 1) / (2.(Row\_Skip + 1)))$$

Binning is used in cooperation with skipping. Pixels that with only skipping would be ignored can be used for averaging and have a weight in the output pixel value. *Row_Bin* and *Column_Bin* values define the number of neighbor pixels to enter the averaging process. The below figure illustrates the case where Colum_*skip* and *Column_bin* are set to 1, which produces a 2X bin.

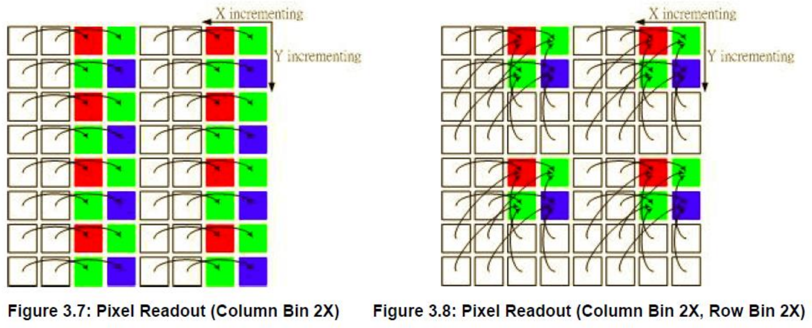

X incrementing
Y incrementing
Figure 3.7: Pixel Readout (Column Bin 2X)
X incrementing
Y incrementing
Figure 3.8: Pixel Readout (Column Bin 2X, Row Bin 2X)

Figure 2.7. In the left image, pixel readout with column bin 2X; In the right image, pixel readout mode for column bin 2X and row bin 2X.

Note: *Column_Start* must be an even value and multiple of $2\ (\mathrm{Column\_Bin} + 1)$; likewise, *Row_Start* must be a multiple of $2\ (\mathrm{Row\_Bin} + 1)$.

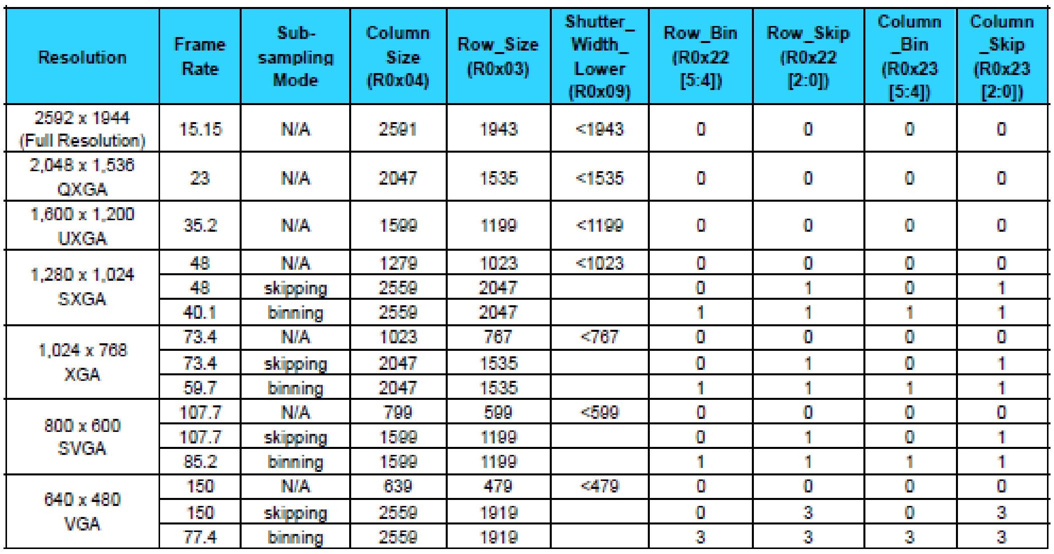
One must have in mind that for each *Column_bin* value there is a different legal range of skipping values. The *Column_start* value is also dependent of the bin and of the Mirror Column Mode (see below). This values can be consulted in detail on the camera hardware specification manual.

Mirror Readout Modes

It is also possible to reverse the column and line readout orders through the use of mirror modes. If the register bit Mirror Column is high (Reg0x20[14] = 1) the column scan order is reversed, and if the register bit Mirror row is high (Reg0x20[15] = 1) the row scan order is reversed.

Standard Resolutions

Table 3 lists some standard resolutions and provides the values to assign to each intervening configuration register to achieve each resolution, assuming that HBMIN and VBMIN values respect the above exposed criteria and that the remaining registers hold their default values.



Resolution	Frame Rate	Sub-sampling Mode	Column Size (R0x04)	Row_Size (R0x03)	Shutter_Width_Lower (R0x09)	Row_Bin (R0x22 [5:4])	Row_Skip (R0x22 [2:0])	Column _Bin (R0x23 [5:4])	Column _Skip (R0x23 [2:0])
2592 x 1944 (Full Resolution)	15.15	N/A	2591	1943	<1943	0	0	0	0
2,048 x 1,536 QXGA	23	N/A	2047	1535	<1535	0	0	0	0
1,600 x 1,200 UXGA	35.2	N/A	1599	1199	<1199	0	0	0	0
1,280 x 1,024 SXGA	48	N/A	1279	1023	<1023	0	0	0	0
	48	skipping	2559	2047		0	1	0	1
	40.1	binning	2559	2047		1	1	1	1
1,024 x 768 XGA	73.4	N/A	1023	767	<767	0	0	0	0
	73.4	skipping	2047	1535		0	1	0	1
	59.7	binning	2047	1535		1	1	1	1
800 x 600 SVGA	107.7	N/A	799	599	<599	0	0	0	0
	107.7	skipping	1599	1199		0	1	0	1
	85.2	binning	1599	1199		1	1	1	1
640 x 480 VGA	150	N/A	639	479	<479	0	0	0	0
	150	skipping	2559	1919		0	3	0	3
	77.4	binning	2559	1919		3	3	3	3

Table 3. Standard Resolutions and respective register configurations.

2.3. Camera Registers and Register Description

The list of all the camera's internal registers as well as their respective functions and default values can be consulted on the hardware specification manual in chapter 2[2]. The Table 2.2 from that chapter gives important additional information by describing the purpose of each register and some criteria to have in mind. The description of each register in that table contains its identification, register number, the meaning of the information on each of its bits and its default value. Below we list some of the most used registers (from a total of 256) for the purposes of the developed project.

Register	Bits	Default Value	Function and considerations	Legal Values
R0x001 Row Start	15:0	0x0036	The Y coordinate of the upper-left corner of the Field of View. If this register is set to an odd value, the next lower even value will be used. Causes a Bad Frame if written.	[0, 2004] even
R0x002 Column Start	15:0	0x0010	The X coordinate of the upper-left corner of the Field of View. The value will be rounded down to the nearest multiple of 2 times the column bin factor Note: Set Column_Start such that it is in the form shown below, where n is an integer: Mirror_Column = 0 / Mirror_Column = 1 no bin: 4n / 4n + 2 Bin 2x: 8n / 8n + 4 Bin 4x: 16n / 16n + 8	[0, 2750] even.
R0x003 Row Size	15:0	0x0797	The height of the field of view minus one. If this register is set to an even value, the next higher odd value will be used Causes a Bad Frame if written.	[1, 2005] odd.
R0x004 Column Size	15:0	0x0A1F	The width of the field of view minus one. If this register is set to an even value, the next higher odd value will be used. In other words, it should be (2*n*(Column_Bin + 1) - 1) for some integer n. Causes a Bad Frame if written.	[1, 2751] odd.
R0x005 Horizontal Blank	15:0	0x0000	Extra time added to the end of each row, in pixel clocks. Incrementing this register will increase exposure and decrease frame rate. Setting a value less than the minimum will use the minumum horizontal blank. The minimum horizontal blank depends on the mode of the sensor. Causes a Bad Frame if written..	[0, 4095]
R0x006 Vertical Blank	15:0	0x0019	Extra time added to the end of each frame in rows minus one. Incrementing this register will decrease frame rate, but not affect exposure. Setting a value less than the minimum will use the minimum vertical blank.	[8, 2047]
R0x009 Shutter Width Upper	15:0	0x0000	The most significant bits of the shutter width, which are combined with Shutter Width Lower (R9).	
R0x009 Shutter Width Lower	**15:0**	**0x0797**	The least significant bits of the shutter width. This is combined with Shutter_Width_Upper and Shutter_Delay for the effective shutter width. If set to zero, a value of "1" will be used.	
R10:0 Pixel Clock Control	6:0	0x0000	**Divide Pixel Clock** Produces a PIXCLK that is divided by the value times two. The value must be zero or a power of 2. This will slow down the internal clock in the array control and datapath blocks, including	{0, 1, 2, 4, 8, 16, 32, 64}

			pixel readout. It will not affect the two-wire serial interface clock. A value of "0" corresponds to a PIXCLK with the same frequency as XCLKIN. A value of 1 means f_PIXCLK = (f_XCLKIN / 2); 2 means f_PIXCLK = (f_XCLKIN / 4); 64 means f_PIXCLK = (f_XCLKIN / 128); etc. NOTE: This field is not reset by the soft Reset (R13). This field should not be written while in streaming mode. Instead, Pause_Restart should be used to suspend output while the divider is being changed.	
R0x00C Shutter Delay	15:0	0x0000	A negative adjustment to the effective shutter width in ACLKs. See Shutter_Width_Lower.	[0, 8191]
R0x010 PLL Control	1	0x0000	Use PLL. When set, use the PLL output as the system clock. When clear, use XCLKIN as the system clock.	
	0		Power PLL. When set, the PLL is powered. When clear, it is not powered.	
R0x011 PLL Config 1	15:8	0x0064	PLL m Factor PLL output frequency multiplier.	[16, 255]
	5:0	0x0004	PLL n Divider PLL output frequency divider minus 1.	[0, 63]
R0x012 PLL Config 2	4:0	0x0000	PLL p1 Divider PLL system clock divider minus 1. If this is set to an even number, the system clock duty cycle will not be 50:50. In this case, set all bits in R101 or slow down XCLKIN.	[0, 127]
R0x022 Row Address Mode	5:4	0x0000	Row Bin The number of rows to be read and averaged per row output minus one. The rows will be read independently into sampling capacitors, then averaged together before column readout. For normal readout, this should be 0. For Bin 2X, it should be 1; for Bin 4X, it should be 3. Causes a Bad Frame if written.	[0, 3]
	2:0		Row Skip The number of row-pairs to skip for every row-pair output. A value of zero means to read every row. For Skip 2X, this should be 1; for Skip 3X, it should be 2, and so on. This value should be no less than Row_Bin. For full binning, Row_Skip should equal Row_Bin. Causes a Bad Frame if written.	[0, 7]
R0x023 Column Address Mode	5:4	0x0000	Column Bin The number of columns to be read and averaged per column output minus one. For normal readout, this should be zero. For Bin 2X, it shoud be 1; for Bin 4X, it should be 3. Causes a Bad Frame if written.	{0, 1, 3}
	2:0		Column Skip The number of column-pairs to skip for every column-pair output. A value of zero means to read every column in the active image. For Skip 2X, this should be 1; for Skip 3X, this should be 2, and so on. This value should be no less than Column_Bin. For full binning, Column_Skip should equal Column_Bin. Causes a Bad Frame if written.	[0, 6]

Table 4. Some Register fields, their functions and default values.

Additional notes concerning the interpretation of this table:

- The default values in case of sub-sets of bits in a register concern only that sub-set and not the register value as a whole.
- Most of the values are affected by another register value (*Synchronize Changes*) and their writings are synchronized with the frame boundaries – the effects of the change of a register value will only take place in the next frame.
- In the table from the hardware specification manual one can also find information regarding to registers with functions concerning the reading modes, the color filter gains and test patterns.

2.4. Implementation Details – Principal Camera Modules and their functions

The Terasic's demo contains several files related to numerous sub-functions of the interactions between the camera and the FPGA board and between the FPGA board and the VGA displayer. By having this set of files, the problem was divided into modules which are assembled to produce the final solution. Next, we present some of the camera modules and their functions. More detailed information about these modules can be found in available documentation of developed projects using this camera[3].

Data capture – CCD Capture

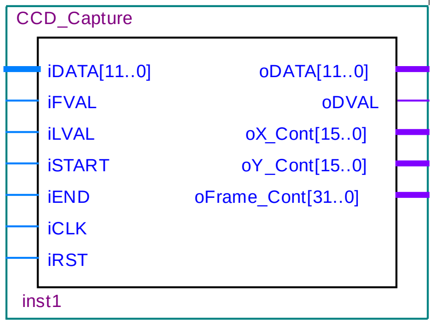


Figure 2.8. CCD_Capture module.

The essential function of this block is to extract the data of valid pixels by taking into account the information of the signals FVAL (Frame Valid) and LVAL (Line Valid) from the CMOS camera, as explained before. The clock frequency of this module is PIXCLK. The output signal oDVAL indicates whether the pixel data under analysis is valid or not.

This module counts the number of frames taking into account the signal FVAL and shows the counting on seven segments displays. The coordinates X and Y of each valid pixel of an image are used as inputs for the RAW2RGB module. The CCD Capture module is defined in *CCD_Capture.v*.

RAW2RGB Module

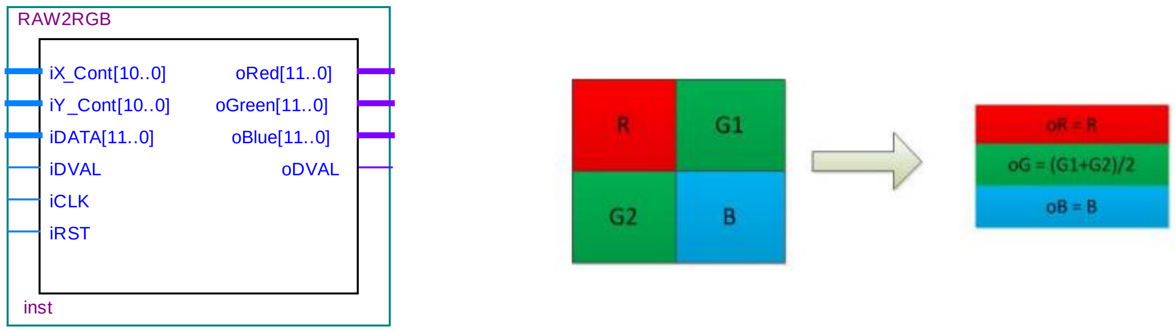


Figure 2.10. In the left image, the RAW2RGB module; In the right image, the way of obtaining RGB components from the Bayer pattern.

This module converts the Bayer Pattern data from the CMOS sensor to RGB. The adopted method consists in extracting four adjacent pixels in a square (R, G1, G2, B) and reproduce a pixel with:

$$\begin{aligned} oR &= R \\ oG &= (G1 + G2)/2 \\ oB &= B \end{aligned}$$

The module is defined in the file *RAW2RGB.v*.

I2C_CCD_Config - I2C Sensor Configuration Module

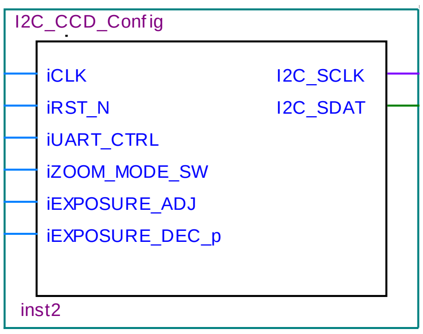


Figure 2.9. I2C_CCD_Config

The CMOS image sensor supports a two-wired serial interface based on the I2C bus protocol. This module uses a serial interface to define the sensor's registers values. The I2C bus is composed of two SDA (Serial Data) bidirectional lines and SCL (Serial Clock). Each device on the I2C bus has a unique address and can operate either as an emitter or as a receiver. One device acts as *master* and the others act as *slaves*. The *master* device generates the clock signal on SCL and initiates/terminates the data transference. The data transference occurs through the SDA.

On the Terasic's demo, the bus connects the FPGA chip (*master*) and the CMOS sensor (*slave*). Aside from the sending the slave address and a directional bit, the configuration of the internal registers implies sending the register ID and 16 data bits. Thus, the writing into registers is composed of several phases: start, slave address + direction bit, register address, one data byte, another data byte, stop. The module is implemented by using a finite state machine (FSM) that generates the right SCL e SDA signals. This module is defined in *I2C_CCD_Config.v*.

VGA Controller

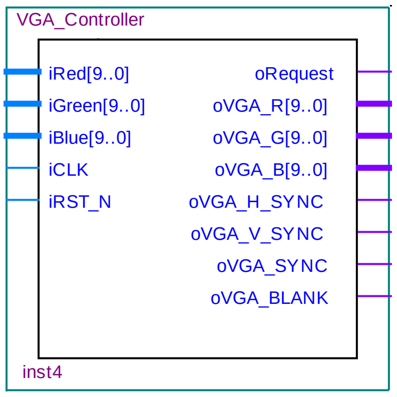


Figure 2.11. VGA_Controller module.

The information in RGB, output from the RAW2RGB module, will be later used as input for the VGA controller module. The outputs of this module correspond to the RGB values to be sent to the VGA and the necessary synchronization signals.
The module is defined in the file *VGA_Controller.v*.

SDRAM Multi-Port Controller

The SDRAM controller will be explained in detail in the section *3. Communication with the SDRAM*.

2.4.1. Interaction among the different modules

The system's architecture is described, in a high level, in figure 2.12[3]. The above explained modules are represented, as well as the principal flows of information flowing among them.

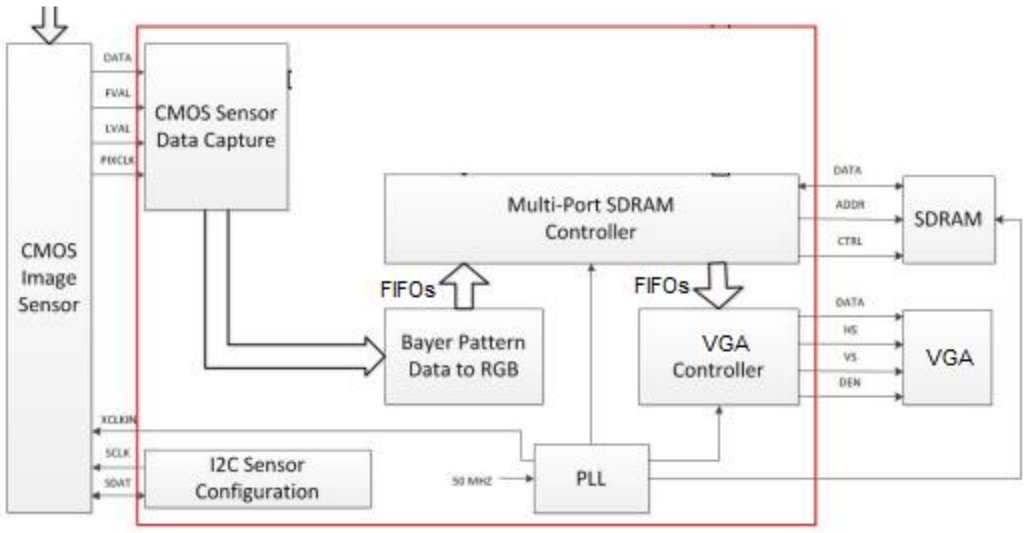


Figure 2.12. System's architecture.

The CMOS sensor captures the image and sends the data to the CMOS Sensor Data Capture module (*CCD_Capture*) which, as already explained, is responsible to extract the data related to the valid pixels (FRAME_VALID and LINE_VALID dependent). The data are generated in Bayer pattern and then sent to the RAW2RGB module which will convert them to RGB.

Then, the RGB data is sent to the SDRAM multi-port controller which will write them to the SDRAM memory with the help of buffering FIFOs. The SDRAM is thus used as a frame buffer. By using FIFOs again the RGB data stored in the SDRAM is read out and sent to the VGA controller which will in its turn send them to the VGA. For more information on the SDRAM and buffering FIFOs, read section *3.3. SDRAM 4-Port Controller module*.

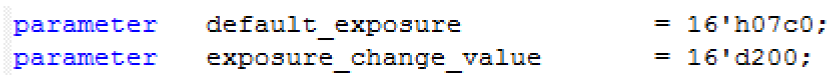
The *I2C_CCD_Config* module defines the CMOS sensor register values using the I2C interface, and the PLL module supplies the clock signals.

2.5. Changing the Camera’s Configurations

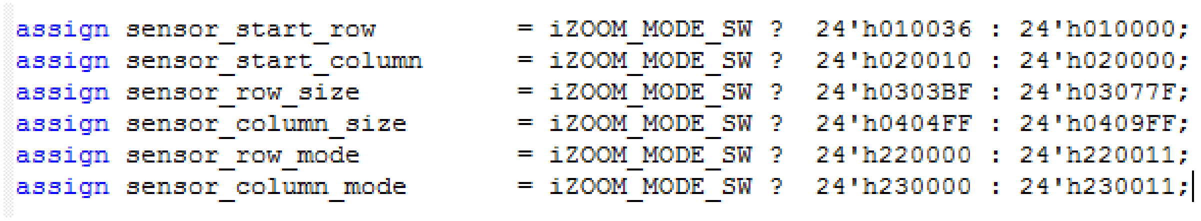
One of the advantages of using the TRDB-D5M camera is the possibility of changing its configurations, such as image resolution, frame rate, etc.

To alter the camera configuration one must change the register values in the file *I2C_CCD_Config.v* and should be aware of the register value criteria listed in table 4.

In the CMOS sensor’s register settings on the demo, some attributes are already defined:



```
parameter   default_exposure            = 16'h07c0;
parameter   exposure_change_value       = 16'd200;
```



```
assign sensor_start_row         = iZOOM_MODE_SW ?  24'h010036 : 24'h010000;
assign sensor_start_column      = iZOOM_MODE_SW ?  24'h020010 : 24'h020000;
assign sensor_row_size          = iZOOM_MODE_SW ?  24'h0303BF : 24'h03077F;
assign sensor_column_size       = iZOOM_MODE_SW ?  24'h0404FF : 24'h0409FF;
assign sensor_row_mode          = iZOOM_MODE_SW ?  24'h220000 : 24'h220011;
assign sensor_column_mode       = iZOOM_MODE_SW ?  24'h230000 : 24'h230011;
```

The variables (wires, to be precise) containing the register data to send to the CMOS sensor can take one of two values depending on the IZOOM_MODE_SW (zoom mode). On the demo the default selected data is that on the right of the ‘:’ operator. By not changing IZOOM_MODE_SW, one can change only the rightmost values to make the desired changes.

The data structure to send (by the I2C interface) is the following:

24’h010036

- 24 bits in total
- Hexadecimal notation
- Register Address/ID – Consult register table
- Data to write to the register (15 bits) – Consult register table

2.5.1. Changing the Frame Rate

To change the frame rate, the PLL module can be used to divide the clock signal XCLKIN, as explained before in section *2.2.1. Clock Division* .

Example:

```
10: LUT_DATA  <= 24'h100051; //      set up PLL power on
11 : LUT_DATA <= 24'h112007; //      PLL_m_Factor<<8+PLL_n_Divider
12 : LUT_DATA <= 24'h120002; //      PLL_p1_Divider
13: LUT_DATA  <= 24'h100053;//       set USE PLL
```

Which means:

$$M = PLL\_m\_Factor = 32$$
$$N = PLL\_n\_Divider + 1 = 8$$
$$P1 = PLL\_p1\_Divider + 1 = 3$$
$$f_{XCLKIN} = 27MHz$$

Resulting clock frequency:

$$f_{PIXCLK} = (f_{XCLKIN} \times M) / (N \times P1) = 36MHz$$

3. Communication with the SDRAM

For communication with the SDRAM there is the need of a SDRAM controller responsible for the generation of the appropriate control signals. Figure 3.1 shows a top level design of a generic SDRAM control circuit. The controller circuit implements the control logic and datapath for generation of adequate control signals, commands and SDRAM addresses.

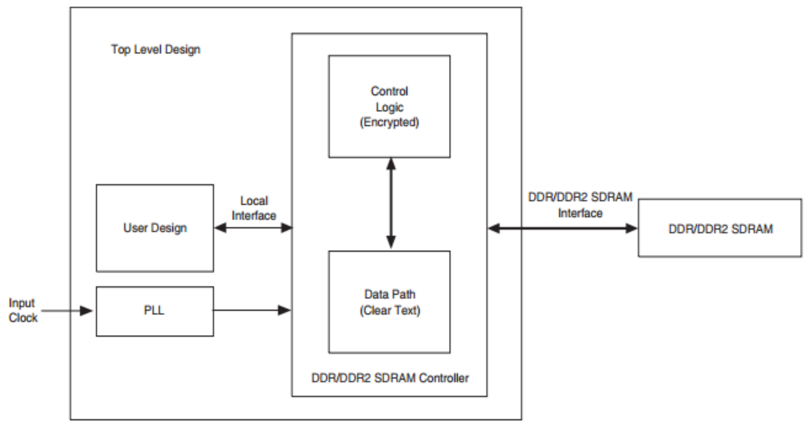


Figure 3.1. High level design of an SDRAM control circuit.

The FPGA Cyclone II board from Altera uses the A2V64S40CTP SDRAM model from Powerchip Semiconductor Corporation.

This section begins by introducing explains the internal structure of the SDRAM circuit and its control signals, as well as some key concepts concerning its operation. These signals include command inputs and addressing information.

Then, the structure of some of the most common commands (Active, Read, Write, Precharge) will be briefly explored.

The gathered information on signals and command structure will be combined in a high level Finite State Machine (FSM) description of the SDRAM control.

Finally, the specific structure and functioning of the TRDB-D5M SDRAM 4-Port controller with buffering FIFOs will be addressed, with relevant information concerning input data structure to obtain the desired result of the controller, and thus correctly implement SDRAM reading and writing.

3.1. SDRAM control circuit structure and basic operation principles

In a very summarized description, the SDRAM circuitry is composed by three main entities:

- 4 banks – Each bank is an array structure of rows and columns (8192 rows x 512 columns x 16 bits). Each memory localization is specified by its bank, row and column.
- A command decoder which decodes the commands received by the circuit.
- A mode register which defines the current operation mode of the SDRAM.

The described structure is depicted in figure 3.2. Note that this figure concerns another SDRAM circuit model from Micron, but presents an identical structure.

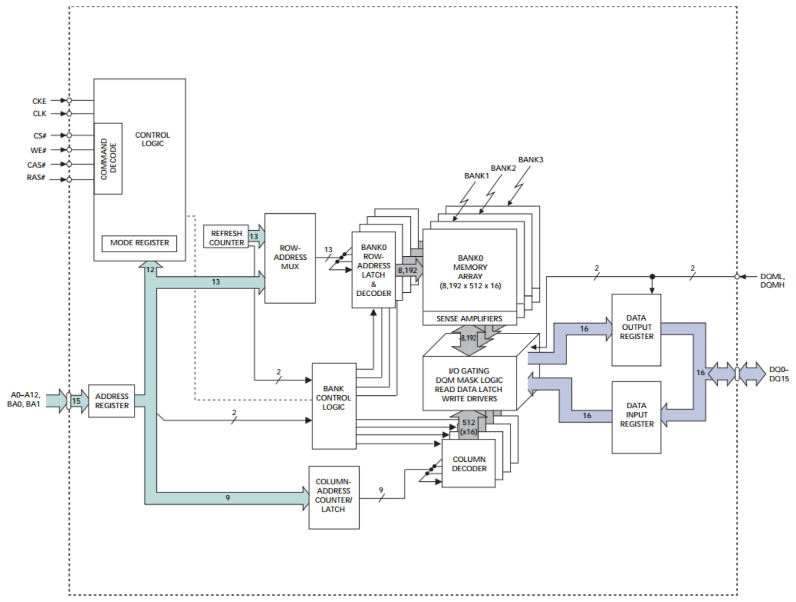


Figure 3.2. SDRAM functional block diagram

The access for data reading and writing from and to the SDRAM are made through bursts. An access burst (reading or writing burst) has its start in a specific column of an open row of an active bank – there is the need to activate a line from a specific bank in order to write in its columns. Only one line can be open at a time in each bank – if the bank has an open line, that line must be closed before the activation of a new line.

The columns' scan order in a line and the length of the access burst are defined in the mode register. Possible burst lengths are 1,2,4 and full page. The burst reading/writing order of the columns can be sequential or interleaved. Reading data are available after 2 or 3 clock cycles after each read access of a read burst.

When a burst request occurs, a block of consecutive columns is selected in the current opened row. The starting column is specified in the address of the command that gave place to the burst. The address of the first column is consists of zeros in the least significant bits, with the number of zeros being equal to the number of bits necessary to address the set of columns of the block (depending on the burst length). In the case of sequential access, the burst starts on the start column and proceeds sequentially until the end column that defines the block, returning to the starting column when the end column is reached (circular writing).

3.2. Implementation details – SDRAM Control Signals, Commands and Controller FSM

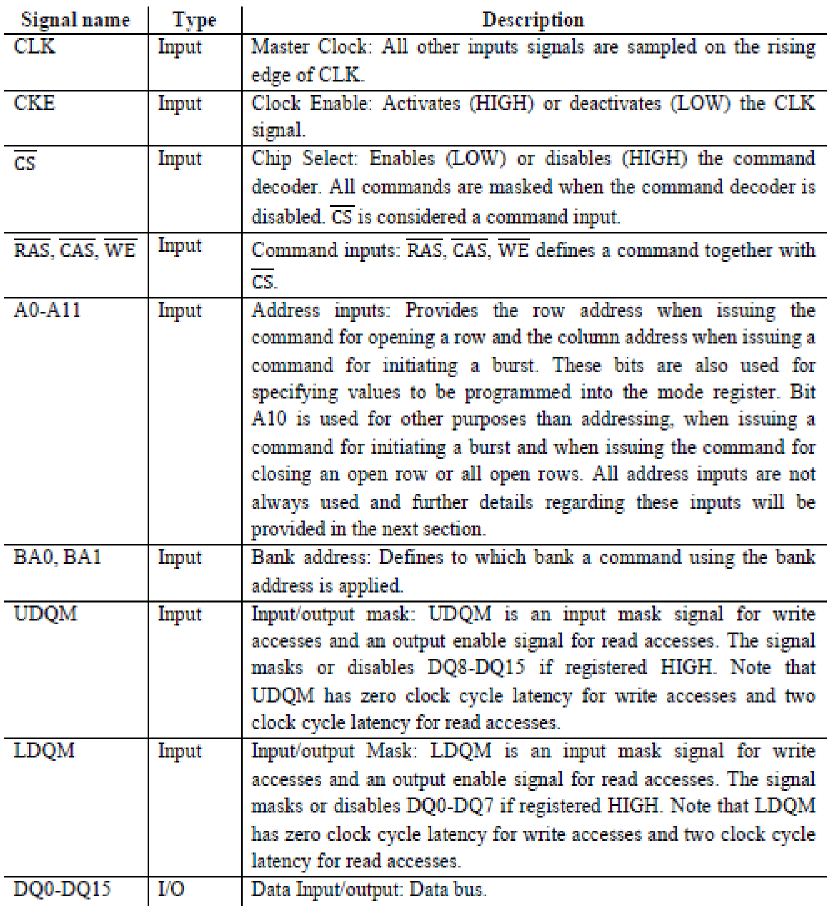
Here, SDRAM control details are addressed. In a first approach, the most common SDRAM control signals and respective functions will be introduced. These include command inputs, clock signals and SDRAM addresses (bank, line and column addressing), and data masks.

In a second phase, these signals will be merged to form control commands. The structure of a few commands will be briefly explained. A high level description of the SDRAM control model through a FSM will also be briefly introduced to further complement the information gathered so far.

In a final phase, implementation details concerning the SDRAM 4-Port controller with buffering FIFOs provided in the camera's demo will be explored.

3.2.1. Signals

Table 5[4] lists some relevant control signals of the SDRAM circuit used in Altera's Cyclone-II DE2 board, explaining their function. Figure 3.3. shows signal and data flow directions between SDRAM controller and the SDRAM chip.



Signal name	Type	Description
CLK	Input	Master Clock: All other inputs signals are sampled on the rising edge of CLK.
CKE	Input	Clock Enable: Activates (HIGH) or deactivates (LOW) the CLK signal.
$\overline{CS}$	Input	Chip Select: Enables (LOW) or disables (HIGH) the command decoder. All commands are masked when the command decoder is disabled. $\overline{CS}$ is considered a command input.
$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	Input	Command inputs: $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ defines a command together with $\overline{CS}$.
A0-A11	Input	Address inputs: Provides the row address when issuing the command for opening a row and the column address when issuing a command for initiating a burst. These bits are also used for specifying values to be programmed into the mode register. Bit A10 is used for other purposes than addressing, when issuing a command for initiating a burst and when issuing the command for closing an open row or all open rows. All address inputs are not always used and further details regarding these inputs will be provided in the next section.
BA0, BA1	Input	Bank address: Defines to which bank a command using the bank address is applied.
UDQM	Input	Input/output mask: UDQM is an input mask signal for write accesses and an output enable signal for read accesses. The signal masks or disables DQ8-DQ15 if registered HIGH. Note that UDQM has zero clock cycle latency for write accesses and two clock cycle latency for read accesses.
LDQM	Input	Input/output Mask: LDQM is an input mask signal for write accesses and an output enable signal for read accesses. The signal masks or disables DQ0-DQ7 if registered HIGH. Note that LDQM has zero clock cycle latency for write accesses and two clock cycle latency for read accesses.
DQ0-DQ15	I/O	Data Input/output: Data bus.

Table 5. Main SDRAM control signals used in DE2 board.

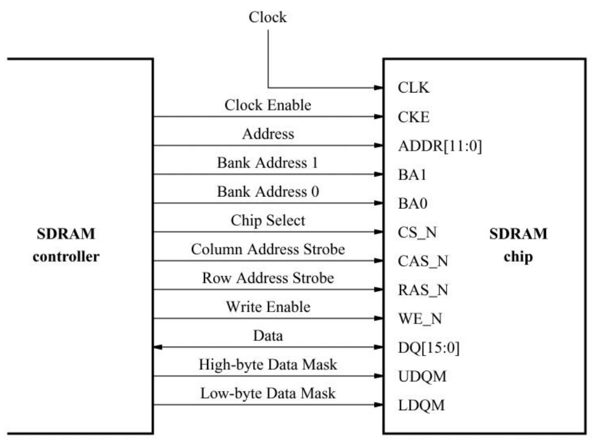


Figure 3.3. SDRAM control signals.

3.2.2. Commands

Table 6 [4] lists some relevant SDRAM commands and command inputs associated to each of them.

Command	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$
COMMAND INHIBIT(NOP)	H	X	X	X
NO OPERATON (NOP)	L	H	H	H
ACTIVE	L	L	H	H
PRECHARGE	L	L	H	L
READ	L	H	L	H
WRITE	L	H	L	L
LOAD MODE REGISTER	L	L	L	L

Table 6. List of the most relevant commands.

Herein, the structure of some of the above listed commands – Active, Precharge, Read and Write commands – will be addressed, in terms of the signals previously introduced in Table 5. The remaining commands' structure and extra details on the selected ones can be further consulted in the SDRAM's datasheet.

Active Command

Command used to open a row in a specific bank:

- BA0 and BA1 specify the bank (0 to 3)
- A0-A11 specify the bank line to open

Note: It is not allowed to open a line in a bank with another active line. It is necessary to close the latter in order to open the one.

Precharge Command

Command used to close an open line of a specific bank or to close the open lines of all banks.

- A10 active implies the closure of open lines on all banks independently of the values in BA0 and BA1.

The precharge command can truncate writing and reading accesses if the automatic precharge is not selected. With automatic precharge, after each data read or write, the open line is closed automatically.

Read Command

Command that starts a reading access burst in an **open line** of a given bank.

- BA0 and BA1 specify the bank
- A0-A7 specify the starting access column
- A10 determines if the precharge is automatic or not – HIGH means automatic precharge.
- The data from the starting column will be put in DQ0-DQ15 two or three clock cycles after the READ command is registered, depending on the *column access strobe* (CAS) latency value specified on the mode register.
- Data from the following columns will also be put in DQ0-DQ15 in the subsequent clock cycles. If signals **DQM** (LDQM or UDQM) are active, the data won't show in the correspondent byte lane until two clocks cycles later.

Write Command

Command that starts a writing access burst in an open line of a given bank.

- BA0 and BA1 specify the bank
- A0-A7 specify the starting access column
- A10 determines if the precharge is automatic or not – HIGH means automatic precharge.
- LDQM, UDQM and DQ0-DQ15 define the data writing in the starting column and on the next ones in each consecutive clock cycle. If a DQM signal is HIGH, the correspondent byte lane isn't written in the column.

3.2.3. SDRAM control Finite State Machine

Figure 3.4[4] describes the different states of control of the SDRAM and specifies the commands and signals associated to each state transition.

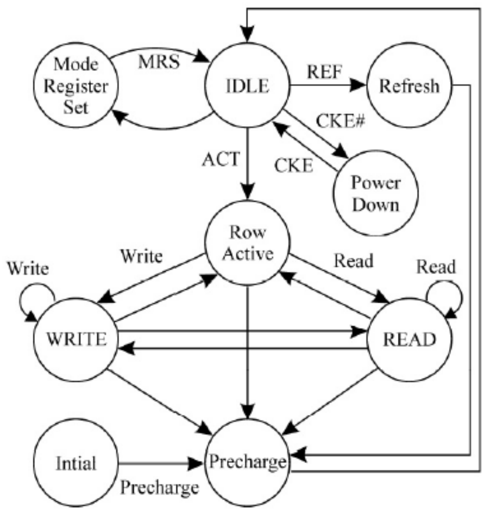


Figure 3.4. SDRAM control FSM.

From the FSM, we can draw some observations that reaffirm and summarize the given information so far:

For reading and writing operations, three main actions should be taken:

1. Open a line with the command Active – minimum break time between active commands is 7 clock cycles.
2. Call the read or write procedural for the open line (SC RCD = 3 clock cycles).
3. Delay between the registry of a read command and the first valid received data is CAS (2 or 3 cycles).

To open a new line, it is necessary that the controller returns to the IDLE state, which is accomplished by the precharge command.

Additionally, as SDRAM is a volatile memory, it needs automatic refresh to maintain the data. This refresh is executed over 8192 times in each 64 ms.

Note: The initialization is a rather complex process. The details of the initialization process should be consulted in the SDRAM's datasheet if needed.

3.3. SDRAM 4-Port Controller module

In this specific project, the data from the RAW2RGB module are output at PIXCLK frequency. The SDRAM and the SDRAM controller operate by default at 100MHz. To overcome clock frequency mismatches, FIFOs are used for buffering. In Terasic's demo there is a total of 4 FIFOs in the SDRAM 4-Port Controller.

The RGB information of each pixel has a total of 30bits (10 bits per color). The camera, with the default configurations given in the demo, uses only two of the four banks in the SDRAM. For each one of the banks, it uses two FIFOs – one for reading and one for writing. The RGB information of a pixel is divided by the two FIFOS and banks: The 10 bits of red and 5 bits of green are kept in one bank, and the 10 bits of blue and the remaining 5 green bits are kept on the other bank.

The SDRAM controller generates the control signals (CAS_N, RAS_N, WE_N, etc.), memory addresses (BA[1..0], SA[11..0]) and commands for the SDRAM from the info received from the reading and writing FIFOs. The data read from the SDRAM are sent to RD1_DATA and RD2_DATA outputs, and the data to write in the SDRAM are given by WR1_DATA and WR2_DATA inputs. At this level of abstraction the user may only care about the sending and receiving signals to and from the FIFOs.

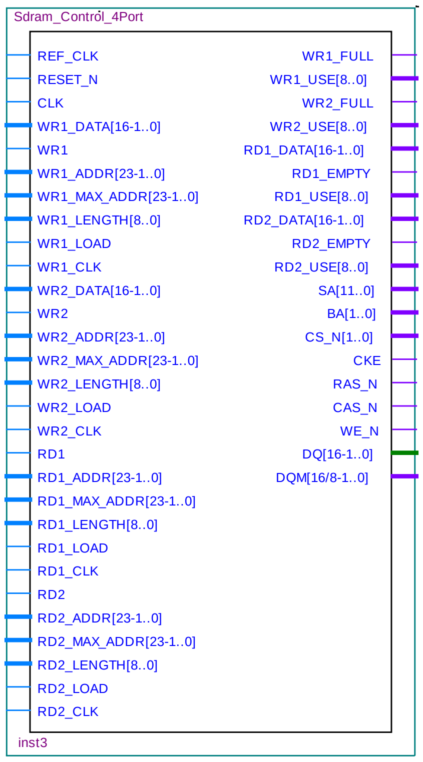


Figure 3.5. SDRAM 4-Port Controller Block Diagram

The input and output information for each FIFO is represented by the numbers one and two (FIFOs from the bank 1 or from the bank 2) and by the indicative RD if it is a read FIFO or WR if it is a write FIFO.

So, for bank 1 writing FIFO, the following info must be provided:

- WR1_DATA [15..0] – Data of a pixel to write in the first bank (the first bit is 0 + 5 bits of green + 10 bits of red).
- WR1_ADDR[22..0] – SDRAM memory address. It provides information for the SDRAM controller to generate appropriate memory addresses for writing to the SDRAM.
    - Bit 22 – CS_N – Chip selects.
    - Bits 21..20 – Bank address (0 to 3) – selects a bank from the 4 available. In this case would be the first bank (00).
    - Bits 19..8 – Row address – Selects the line from the bank to be written.
    - Bits 7..0 – Column address – Starting column of the write.
- WR1_LOAD – when set to 1, it causes state registers to reset for writing a new image in the SDRAM, always with the same starting memory address.
- WR1_CLK – writing clock , the PIXCLK is used for this purpose.
- WR1_LENGTH – increment value to add to the writing address for the next writing. In the demo, the default value is 256, corresponding to a full line.
- WR1 – It should be set to 1 when a writing command is desired.
- WR1_MAX_ADDR – Maximum address in the SDRAM bank 1 to use in the image storing. In the demo, it is the address 640*480 for bank 1.

For the reading FIFO of bank 1:

- RD1_DATA[15..0] – data read from the address generated by the SDRAM controller from the given RD1-ADDR SDRAM address.
- RD1_ADDR[22..0] – Same structure as in the writing case.
- RD1_LOAD – Same as for writing.
- RD1_CLK – clock used for reading. In the demo, the VGA clock is used.
- RD1_LENGTH – Same as for writing.
- RD1 – It should be set to 1 to send a read request.
- RD1_MAX_ADDR – Maximum address to read from the SDRAM by this FIFO. In case of the demo, this address is 640*480 (Reads the data in the same address they're written by the Write FIFO).

For the FIFOs designated for bank 2, the necessary follows an identical structure.

4. Camera - SDRAM-VGA Communication

In short, after the detailed description of each element, it is now possible to establish the relations between the principal intervening entities in the system:

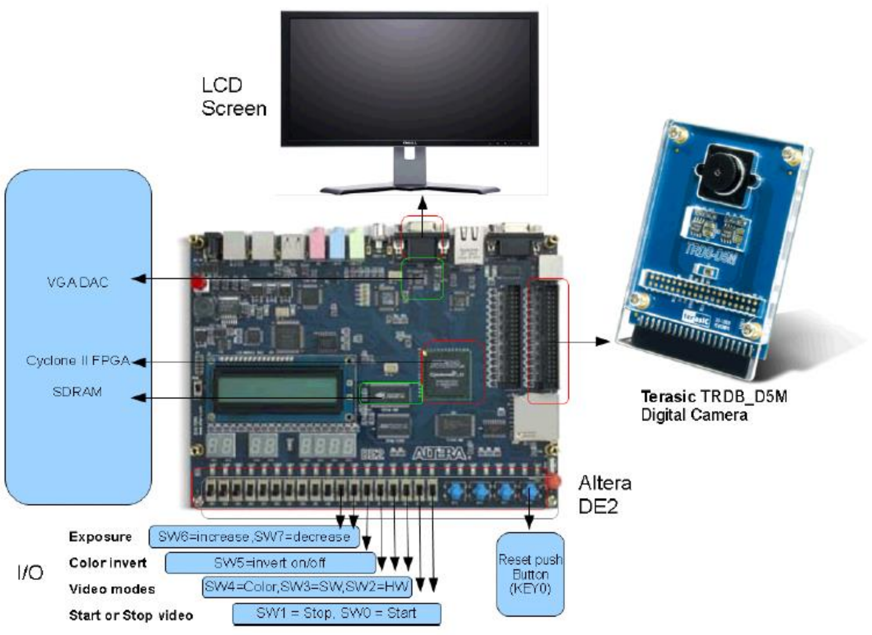


Figure 3.5. SDRAM control FSM

The file responsible for the instantiation and communication between modules is the file *DE2_D5M.v*. This *Verilog* file uses wires to transport the signals and data between the components. One can use this file to create a symbol file to be used as a schematic block.

In summary, the main components interact the in the following way:

Camera-SDRAM interface

As mentioned, the camera sends the pixel RGB data to the SDRAM-Controller, using the two write FIFOs – one for each of the two banks used. The SDRAM controller handles the writing of the buffered information to the SDRAM.

SDRAM – SDRAM controller interface

The SDRAM controller generates the control signals, memory addresses and the commands for the SDRAM from the information received from the write FIFOs and a few other inputs.

SDRAM controller – VGA interface

The read FIFOs buffer the RGB pixel data stored in the SDRAM, which are passed as inputs to the VGA controller.

5. References

[1] Terasic Technologies, TRDB-D5M 5 Mega Pixel Digital Camera Development Kit, User Manual

[2] Terasic Technologies, TRDB-D5M, Terasic TRDB-D5M Hardware specification, Document Version 0.2, June 10, 2009.

[3] Ruiwen Zhen, Enhanced Raw Image Capture And Deblurring, Graduate Program in Electrical Engineering, Notre Dame, Indiana, April 2013.

[4] Daniel Bengtsson, Richard Fång, Developing a LEON3 template design for the Altera Cyclone-II DE2 board, *Master of Science Thesis in Integrated Electronic System Design,* Chalmers University of Technology University of Gothenburg Department of Computer Science and Engineering Göteborg, Sweden, August 2011.

[5] Bruna Nogueira, Ricardo Mendes, Object Tracking and Detection on FPGA Board Cyclone II, Digital Systems Project, Electrical and Computer Engineering Department, University of Coimbra, 2015.

6. Other Sources of Information

Alexander Bochem, Rainer Herpers and Kenneth B. Kent, Acceleration of Blob Detection in a Video Stream using Hardware, Faculty of Computer Science University of New Brunswick Fredericton, Canada, May 26, 2010.

Cato Marwell Jonassen, Embedded Demonstrator for Video Presentation and Manipulation, Master of Science in Electronics, Norwegian University of Science and Technology, June 2010.

V. B. Jagdale, R. J. Vaidya, High Definition Surveillance System Using Motion Detection Method based on FPGA DE-II 70 Board, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-2, December-2012.

Muhammad Shahzad , Object Tracking Using FPGA, (An application to a mobile robot), Master Thesis, School of Information Science, Computer and Electrical Engineering, Halmstad University, May 2012

M. Surumbar Khuzhali, *Back Ground Subtraction Algorithm For Moving Object Detection In FPGA,* Department of ETC, Bharath University, India, Middle-East Journal of Scientific Research 20 (2): 198-204, 2014.

Bing Han, William Roberts, Dapeng Wu, Jian Li*, Robust Feature-based Object Tracking*, Department of Electrical and Computer Engineering University of Florida Gainesville

G. Shrikanth, Kaushik Subramanian, Implementation of FPGA-based Object Tracking Algorithm, Electronics and Communication Engineering Sri Venkateswara College of engineering, Sriperumbudur, April 2008.