CDA 4203 Sec 001 Spring 2022 Computer System Design

Mini-Project: Simple Digital Camera Design
Due: 11:59PM, Wednesday, 6th April 2022
Submission only by Canvas

Today's Date:	4/9/2022
Team Member Names: Your U Numbers: (Up to 3 members per team)	Ahmed El Maliki- U18019330 *Stella Kariuki - U19511653* *Ivan Gonzalez- U 15904968*
Work distribution	Ahmed El Maliki: Part A Stella Kariuki: Part B: 3,4,5,6 Ivan Gonzalez Part B: 1,2,7,8

Feedback: Your feedback is extremely important to improve the mini-project for future course offerings.

Total number of person hours spent:	*Ahmed El Maliki: 20H* *Stella Kariuki: 15H* *Ivan Gonzalez: 15H*
Exercise difficulty: (Easy, Average, Hard)	
Issues you ran into:	● *Budget is too tight, project requires an external clock which makes it hard when it comes to finding the right FPGA board* ● *Does not specify if we need to actually assemble this in a box to be a functional project*
Any suggestions to improve this project:	*Boring lengthy precise instructions*
Any other feedback:	

Part A
Front End Interface Design

A.1**(1 pt.)** Specification Analysis: Analyze the design specification and identify all requirements. What additional features would you like to see in the camera? (Maximum 1 Page)

Requirements :

As a user I want the camera to be able to focus on a subject or scene of interest and press a shutter button to take a snapshot.

As a user I want the resulting image to be stored in the camera's local memory.

As a user I want the resolution of the image be at-least 1,024H x 1,024V (~1 million pixels)

As a user I want to be able to take several images and store them in the memory.

As a user I want be able to interface the camera with a PC to upload the images

As a designer I cannot exceed 200$ as the budget for this project.

As a designer, I want to maximize the number of pictures that can be stored in the memory of the camera, and maximize the rate at which snapshots can be taken successively, all while not exceeding the 200$ budget.

As a designer I want a Pixel Array component where each pixel when exposed to the light can register the light intensity in a specific range of 0-255 (0 for complete darkness and 255 maximum brightness).

As a designer I must implement the pixel array using the following IC from Micron Inc, MT9M001C12STM

As a designer I want a Camera controller that would be in charge of three main functions: initialize the block, the image capture process, and read out the pixel data.

Additional Features if Possible:

Video capability.

Zoom in and out

Image compression

Shooting modes

Image Stabilization

Autofocus

Eye detection

A.2 **(1 pt.)** Read the datasheet and analyze the sensor array features. Summarize the **features of the sensor array relevant to your design**. *(Maximum 1 page)*

The sensor can be operated in its default mode or programmed by the user for frame size, exposure, gain setting, and other parameters. The default mode outputs a QXGA image at 12 frames per second (fps).

The sensor has an on-chip analog-to-digital converter (ADC) that provides 10 bits per pixel.

Control signals FRAME_ VALID and LINE_ VALID are output on dedicated pins, along with a pixel clock that is synchronous with valid data.

The MT9T031 pixel array is configured as 2,112 columns by 1,568 rows.

Columns from 0 through 27 and from 2,085 through 2,111 are optically black.

Rows from 0 through 15 and from 1,561 through 1,567 are optically black.

These optical black columns and rows can be used to monitor the black level.

The black rows and columns can also be read out by setting specific Data to some registers(R0x20 (11) and R0x1E (7))

There are 2,057 columns by 1,545 rows of optically active pixels, which provides a four-pixel boundary around the QXGA (2,048 x 1,536) which serve to avoid boundary effects during color interpolation and correction.

The Sensor uses a Bayer color pattern:

even-numbered rows contain green and red color pixels

odd-numbered rows contain blue and green color pixel

The even-numbered columns contain green and blue color pixels

odd-numbered columns contain red and green color pixels.

Bulb Mode.

A.3 **(1 pt.)** Define the port interface of the Camera controller block. Briefly describe the purpose of each port. *(Maximum 1 page)*

Capture : input - user capture picture button // added for design

GSHT_CTL : Input - Global shutter control // guessing it de-bounces the added external Capture button

// Clocks

PIXCLK: Output - Pixel clock: pixel data outputs are valid during the falling edge of this clock. Frequency = (master clock)

EXTCLK: input – Clock in: master clock into sensor (48 MHz maximum)

SCLK: input – serial clock: clock for serial interface

// Validation signals

LINE_VALID: Output - Line valid: output is pulsed HIGH during line of selectable valid pixel data (see R0x20 for options).

FRAME_VAILD: Output - Frame valid: output is pulsed HIGH during frame of valid pixel data.

// Others

Standby: Input - activates (HIGH) standby mode, disables analog bias circuitry for power saving mode.

Trigger: Input - activates (HIGH) snapshot sequence.

Reset: Input - activates (LOW) asynchronous reset of sensor. All registers assume factory defaults.

OE: Input - When HIGH, places outputs DOUT[9:0], FRAME_VALID, LINE_VALIDPIXCLK, and STROBE into a tri-state configuration.

SDATA: I / O -serial data bus, requires 1.5KΩ resistor to3.3V for pull-up.

DOUT[9:0]: Output - Data out: pixel data output bit 0, DOUT[9] (MSB), DOUT[0] (LSB).

STROBE: Output - output is pulsed HIGH to indicate sensor reset operation of pixel array has completed

A.4 **(2.5 pts.)** Analyze and define the timing interface required between the Pixel Array and Camera Controller blocks. *(Use as many pages as needed)*

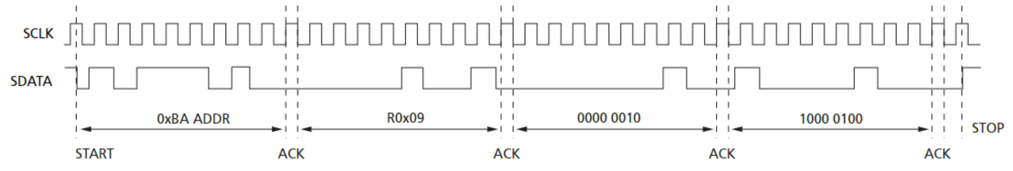
The process starts with initializing the sensor through SDATA and SCLOCK. and setting up the register using a defined protocol.

The protocol is defined as follows :
The two-wire serial defines several different transmission codes, as follows:
- *a start bit*
- *the slave device 8-bit address*
- *a(n) (no) acknowledge bit*
- *an 8-bit message*
- *a stop bit*

Below is an example of write to register 9 with value 0x0284:

next register. The master stops writing by sending a start or stop bit.

Figure 19: Timing Diagram Showing a Write to R0x09 with the Value 0x0284



We do not need to adjust the image size, by default; it satisfies the requirement.

All we need to is enable strobe signal and assert a trigger signal to get the ball rolling:

R0B (0x0B) = 0x0001 // assert trigger

R30(0x1E) [bit 9] = 1 // enable strobe signal

El Maliki- Stella Kariuki- Ivan Gonzalez

Once the user hits the external button to take a picture, the trigger signal kicks in and the value of that register is then reset to 0, the image sensor is now collecting the required data from the pixel array and immediately checking FRAME_VALID afterwards, The LINE_VALID signal will then be checked, when data is being acquired from active columns of the pixel. If LINE_VAILD is valid then every negative edge of the PIXCLK will produce 10 bits of data for each pixel. The pixel data needs to be captured on the following rising edge of the PIXCLK; Hence,the camera controller operates at 100Mhz and the image sensor goes for 48 MHZ. Once all the lines are captured, FRAME_VALID will be set to 0

Figure 7: Timing Example of Pixel Data

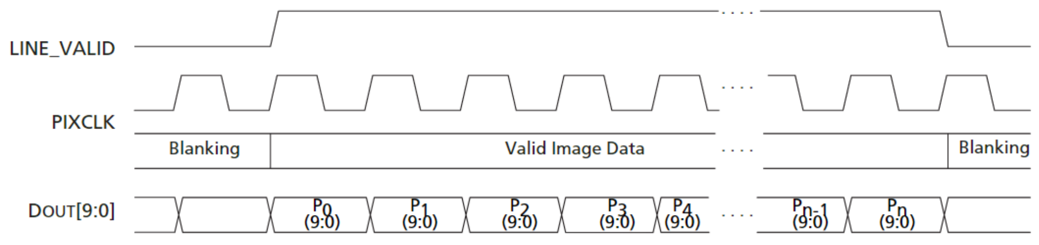
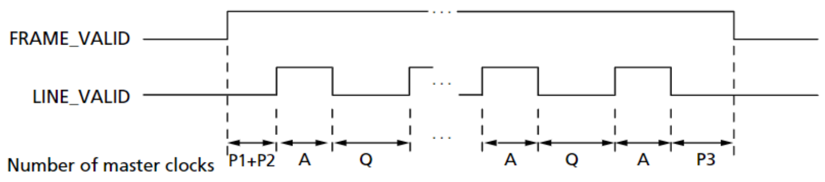
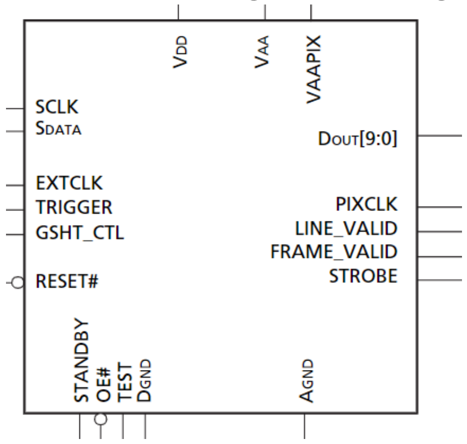


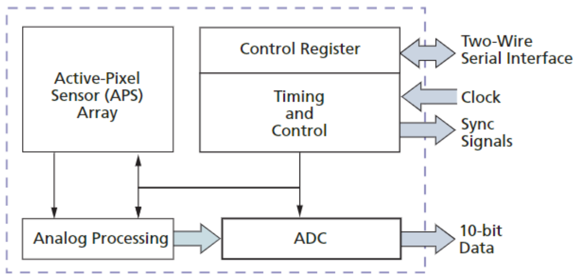
Figure 8: Row Timing and FRAME_VALID/LINE_VALID Signals



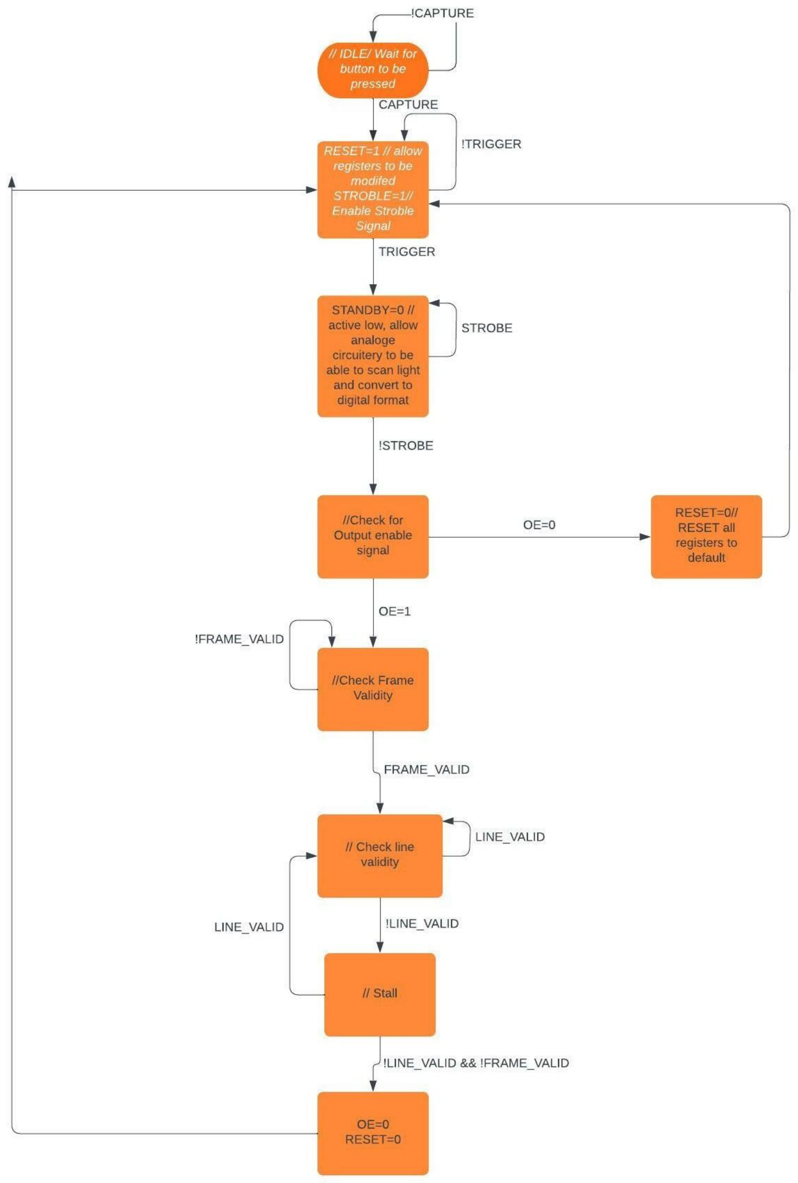
Black box and high Level Diagram Block:



Block Diagram



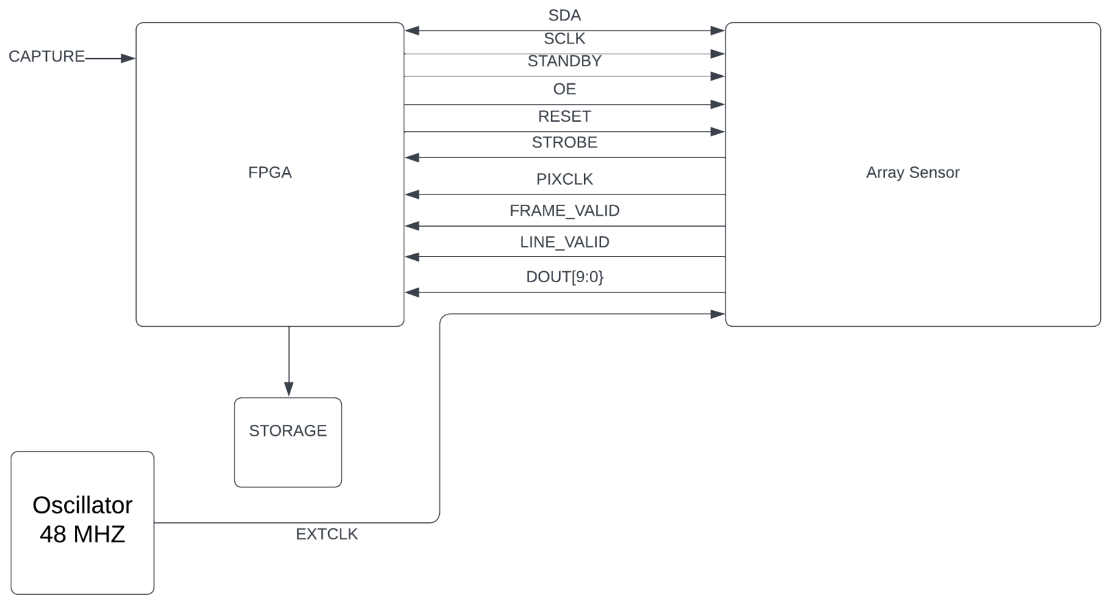
A.5 (**2.5 pts.**) Implement an RTL design satisfying the port and timing interfaces determined in Questions (3) and (4). For the controller, you can stop at the state diagram.
(Use as many pages as needed)


!CAPTURE
// IDLE/ Wait for button to be pressed
CAPTURE
!TRIGGER
RESET=1 // allow registers to be modifed STROBLE=1// Enable Strobe Signal
TRIGGER
STANDBY=0 // active low, allow analoge circuitery to be able to scan light and convert to digital format
STROBE
!STROBE
//Check for Output enable signal
OE=0
RESET=0// RESET all registers to default
OE=1
!FRAME_VALID
//Check Frame Validity
FRAME_VALID
// Check line validity
LINE_VALID
LINE_VALID
!LINE_VALID
// Stall
!LINE_VALID && !FRAME_VALID
OE=0 RESET=0

El Maliki- Stella Kariuki- Ivan Gonzalez

A.6 **(1 pt.)** Draw a detailed schematic of the partial design of the front-end as well as user interfaces. Identify any other components that are required (for example, crystal-controlled oscillator). Show these components as well in the schematic. *(Maximum 1 page)*

// Trigger gets initialized by internal SDA register protocols



A.7 **(1 pt.)** Estimate: (a) how long it will take for one image capture; and (b) the approximate dollar cost to implement the front-end interface. *(Use as many pages as needed)*

We did not edit the resolution of the image and we are using a master clock of 48Mhz, the approximate time it will take to capture a picture is 83.27 seconds. (Please refer to the table 4 attached Below)

As for the Cost:

FPGA board: Mimas V2 Spartan 6 FPGA Development Board with DDR SDRAM SKU: FPGA006 — Cost: $ 71.99

Crystal Oscillator 48 MHZ: NX2016SA-48M-EXS00A-CS08718 —Cost:$0.74

Image sensor: MT9T031- Cost: $16.9

Button:Momentary Pushbutton Switch - 12mm Square —Cost: $0.55

Total Cost:$90.18

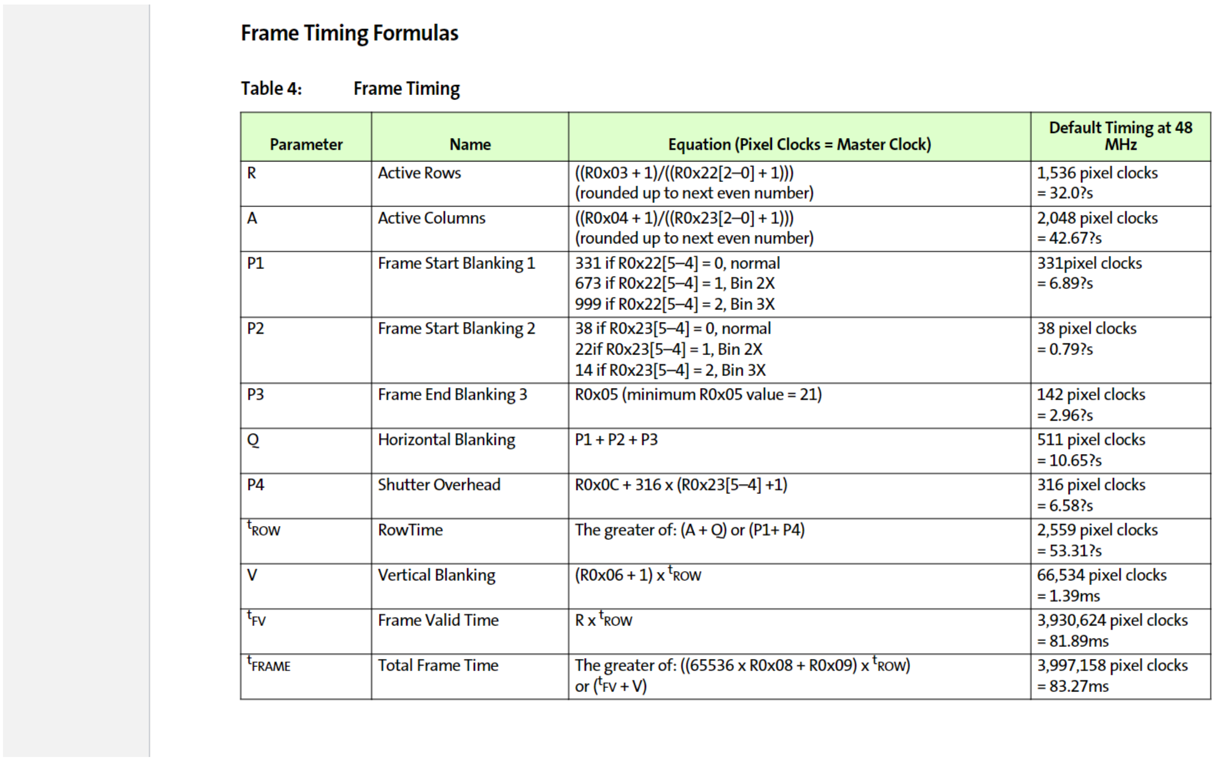
References:

https://numato.com/product/mimas-v2-spartan-6-fpga-development-board-with-ddr-sdram/

https://www.digikey.com/en/products/detail/ndk-america,-inc./NX2016SA-48M-EXS00A-CS08718/7645321

https://www.findchips.com/detail/MT9T031C12STC/1937066-Aptina%20Imaging

https://www.sparkfun.com/products/9190



Frame Timing Formulas

Table 4: Frame Timing

Parameter	Name	Equation (Pixel Clocks = Master Clock)	Default Timing at 48 MHz
R	Active Rows	((R0x03 + 1)/((R0x22[2–0] + 1))) (rounded up to next even number)	1,536 pixel clocks = 32.0?s
A	Active Columns	((R0x04 + 1)/((R0x23[2–0] + 1))) (rounded up to next even number)	2,048 pixel clocks = 42.67?s
P1	Frame Start Blanking 1	331 if R0x22[5–4] = 0, normal 673 if R0x22[5–4] = 1, Bin 2X 999 if R0x22[5–4] = 2, Bin 3X	331pixel clocks = 6.89?s
P2	Frame Start Blanking 2	38 if R0x23[5–4] = 0, normal 22if R0x23[5–4] = 1, Bin 2X 14 if R0x23[5–4] = 2, Bin 3X	38 pixel clocks = 0.79?s
P3	Frame End Blanking 3	R0x05 (minimum R0x05 value = 21)	142 pixel clocks = 2.96?s
Q	Horizontal Blanking	P1 + P2 + P3	511 pixel clocks = 10.65?s
P4	Shutter Overhead	R0x0C + 316 x (R0x23[5–4] +1)	316 pixel clocks = 6.58?s
^{t}ROW	RowTime	The greater of: (A + Q) or (P1+ P4)	2,559 pixel clocks = 53.31?s
V	Vertical Blanking	(R0x06 + 1) x ^{t}ROW	66,534 pixel clocks = 1.39ms
^{t}FV	Frame Valid Time	R x ^{t}ROW	3,930,624 pixel clocks = 81.89ms
^{t}FRAME	Total Frame Time	The greater of: ((65536 x R0x08 + R0x09) x ^{t}ROW) or (^{t}FV + V)	3,997,158 pixel clocks = 83.27ms

Part B
Memory & PC Interface Design

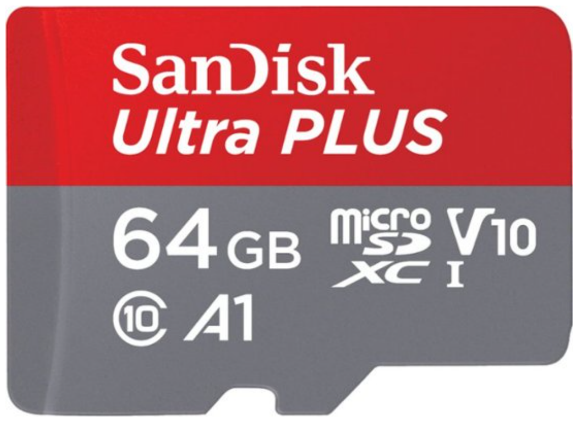
B.1(1 pt.) Memory Component:
Choose an off-the-shelf memory component that can be used as internal memory for the camera. List the memory components that you have researched and provide arguments for your memory choice.

In this part of the project, we use two main components, an FPGA that would be used as the camera controller and a micro SD card with the capacity necessary to support the image resolution and the number of pictures to be stored.

After analyzing different options we concluded that the best component for our camera was the Mimas V2 Spartan 6 FPGA Board as the Camera controller and since the component comes with a built-in micro SD card slot, we use the SanDisk Ultra PLUS 64GB microSDXC UHS-I, Memory Card.

When looking at the different types of memories the main features that we focused on were being able to keep up with the speed of the camera controller and having enough space to be able to store the pictures. The built-in memory in our FPGA was not a good option since it is a 16 Mb SPI flash memory and with that amount of storage we would not be able to save a good amount of pictures. If we were looking for any FPGA with a higher built-in memory the cost would increase and exceed our budget. Therefore, our best option was to use the built-in micro SD adapter.

The SanDisk Ultra PLUS microSDXC card was a good option since it is compatible with most devices, offers 64GB of storage, and it is a Class 10 rating memory, meaning the card can sustain at least 10MB/s continuous recording; the maximum read speed of 130 megabytes per second (transfers up to 1,000 pictures in a minute).



B.2 **(1 pt.) Memory Component Features:** Read the datasheet of the selected memory component and briefly summarize its features.

SanDisk Ultra Plus 64GB microSDXC UHS-I Memory Card

Model: SDSQUB3-064G-AN6TN

Storage Capacity: 64 gigabytes

Maximum Read Speed: 130 megabytes per second

Memory Card Format: microSD

Speed Class Rating: 10

SD Bus Mode: UHS-I

UHS Speed Class: U1

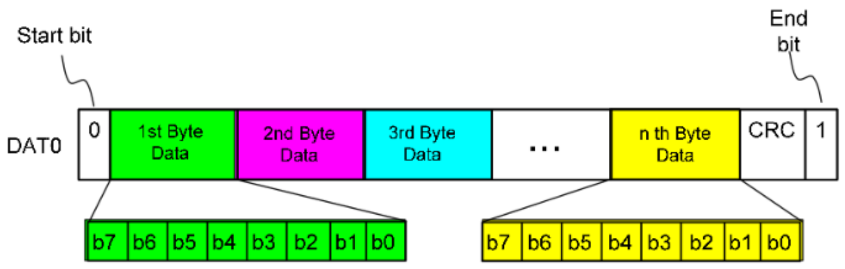
Video Speed Class: V10

Overall the memory is a good balance between cost and performance, it is inexpensive for our budget and it is an optimized memory that has fast and good performance when used in our FPGA.

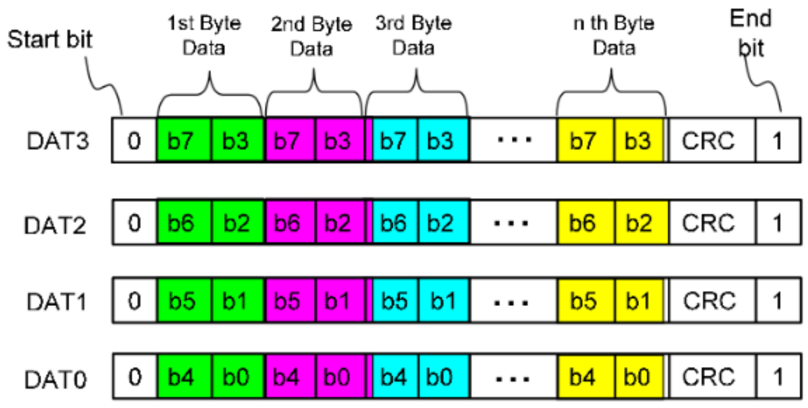
B.3 **(1 pt.) Port Interface:** Define the port interface of the memory with the camera controller. Briefly describe the purpose of each port.

This section takes a closer look at how the memory card communicates with the camera controller for storing and retrieval purposes. Here we will examine the ports on the memory card and what they do, the data they send/receive to/from the camera controller. Note that in this section we will only look at the working of SD Bus Mode.
We did not use SPI bus mode for the following reasons:
- *The SPI mode is slow and there is loss of performance since SPI mode uses a single data line*
- *The current versions of SD cards are not compatible with SPI mode although they are built with the option. Since we have a fairly new model, it would be suitable to use SD bus mode.*

How does SD Mode work?
Communication over the SD bus is based on command and data bit streams that are initiated by a start and stop bit. There are two data packet formats in SD mode:
1. *Usual Data*
    - *Data is sent in 8-bit width with LSByte first and the MSByte last sequence, but in individual bytes, the MSBit goes first and the LSBit goes last.*
    - *The data packet format for Usual Data (8 bit width) is shown below.*

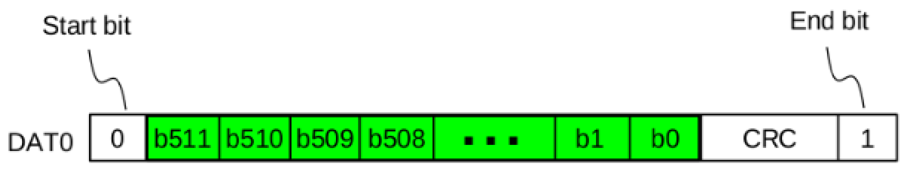


Data Packet Format for Standard Bus (only DAT0 used)

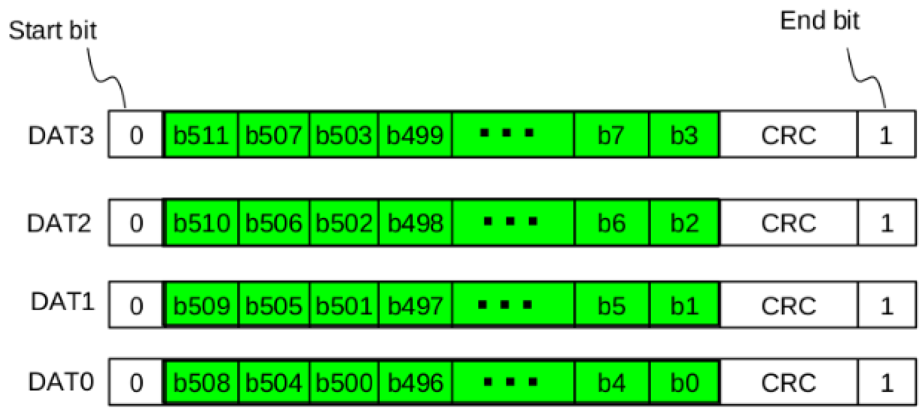

Start bit
1st Byte Data
2nd Byte Data
3rd Byte Data
n th Byte Data
End bit
DAT3
DAT2
DAT1
DAT0
0
b7 b3 b7 b3 b7 b3 ··· b7 b3
b6 b2 b6 b2 b6 b2 ··· b6 b2
b5 b1 b5 b1 b5 b1 ··· b5 b1
b4 b0 b4 b0 b4 b0 ··· b4 b0
CRC
1

Data Packet Format for Wide Bus (all four lines used)

2. *Wide width data (512 bits wide)*
   - *Wide width data is usually shifted from the MSB bit as shown below:*



Data Packet Format for Standard Bus (only DAT0 used)


Start bit
End bit
DAT3
DAT2
DAT1
DAT0
0
b511 b507 b503 b499 ··· b7 b3
b510 b506 b502 b498 ··· b6 b2
b509 b505 b501 b497 ··· b5 b1
b508 b504 b500 b496 ··· b4 b0
CRC
1

Data Packet Format for Wide Bus (all four lines used)

In SD Mode the following Pins are used:

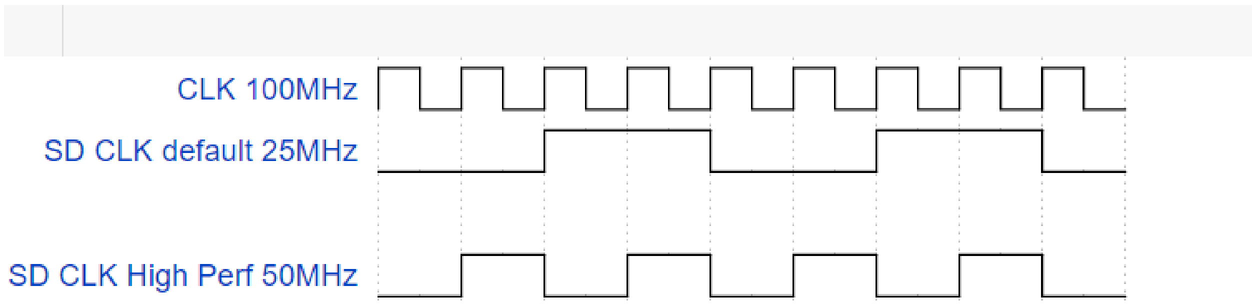
Pin No.	SD Mode		
	Name	**Type**[1]	**Description**
1	DAT2[2,5]	I/O/PP	Data Line [bit 2]
2	CD/DAT3[2]	I/O/PP[3]	Card Detect/Data Line [bit 3]
3	CMD	PP	Command/Response
4	V_{DD}	S	Supply Voltage
5	CLK	I	Clock
6	V_{SS}	S	Supply voltage ground
7	DAT0	I/O/PP	Data Line [bit 1]
8	DAT1[2,4]	I/O/PP	Data Line [bit 2]

1. ***CMD*** *- command. A command is a token that starts an operation. A command is from the host to either a single card (address command) or multiple connected cards (broadcast command). A command is transferred serially on the CMD line.*
   A response is also sent either from an addressed card or synchronously from all connected cards to the host as an answer to a previously received command. A response is also serially transferred on the CMD line.

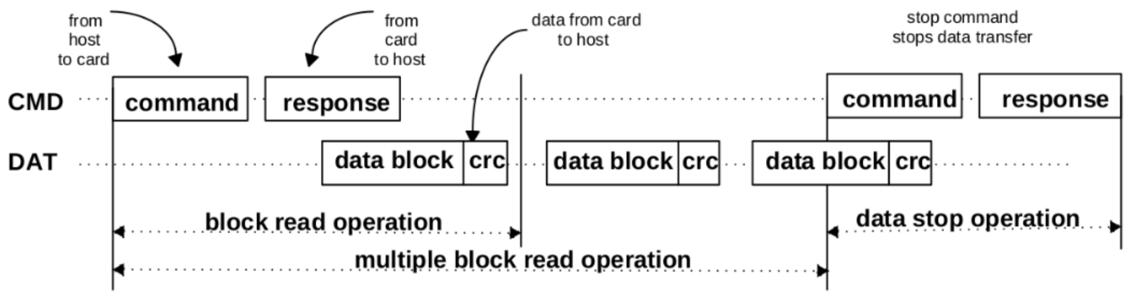
2. ***DAT0*** *- Data is transferred serially on data lines DAT0 -DAT3, and can be transferred to and from the host. CRC (Cyclic Redundancy Check) response and Busy indication can only be sent by the card to the host using this line only. At this time DAT1-DAT3 are don't care values.*
3. ***DAT1, DAT2*** *- Refer to 2 above*
4. ***DAT3/CD*** *- This serves as a card detect line that communicates with the host information on whether the card is detected or not.*
5. ***CLK*** *- Clock input controls the speed of read or write between the SD card and the host. This clock will be fed as input from the PIXCLK*
6. ***VDD*** *- Supply voltage*
7. ***VSS*** *- Supply voltage ground*

B.4 **(1 pt.) Timing Interface:**
Analyze and define the timing interface required between the memory and the rest of the system.

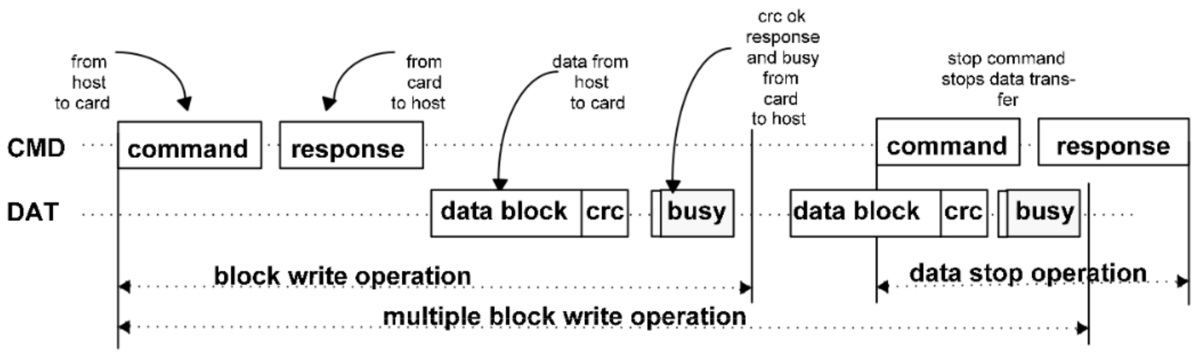
The SD card maximum clock frequency of operation is 50MHz. The CLK input from the camera controller is 100 MHZ which does not support this particular SD card. For default performance mode, the SD card requires a clock frequency of up to 25 MHz i.e read/write speeds of up to 12.5 MB/s. To achieve this mode, the SD CLK would need to pulse on every 4th period of the CLK input. In high performance mode, the SD card requires a clock frequency >25MHz and up to 50MHz. This enables read/write speeds of up to 25MB/s. This can also be achieved by SD CLK pulsing on every 2nd period of the CLK input.



Read Operation of SD card:
The read operation of the SD cards is initialized by receiving a command from the host controller via the CMD line. Depending on the status of the SD card, the SD will send a response to the host over the same line. After which, the data will start being received on the DAT lines. For single address mode, only the DAT0 line will be used for transfer of data. For multiple address mode (broadcast) read, multiple lines may be used. CRC bits are attached to the data as they come in from the host. These are usually 7 bits that work to detect data errors. Messages pertaining to CRC status and BUSY status, as we shall see in the write operation, are sent through the DAT0 line only. Data can be read in a single block, or multiple blocks. It is faster to use multiple blocks. When reading is done, a stop command is issued on the CMD line to stop the reading operation, and a response is sent from the SD to the host.

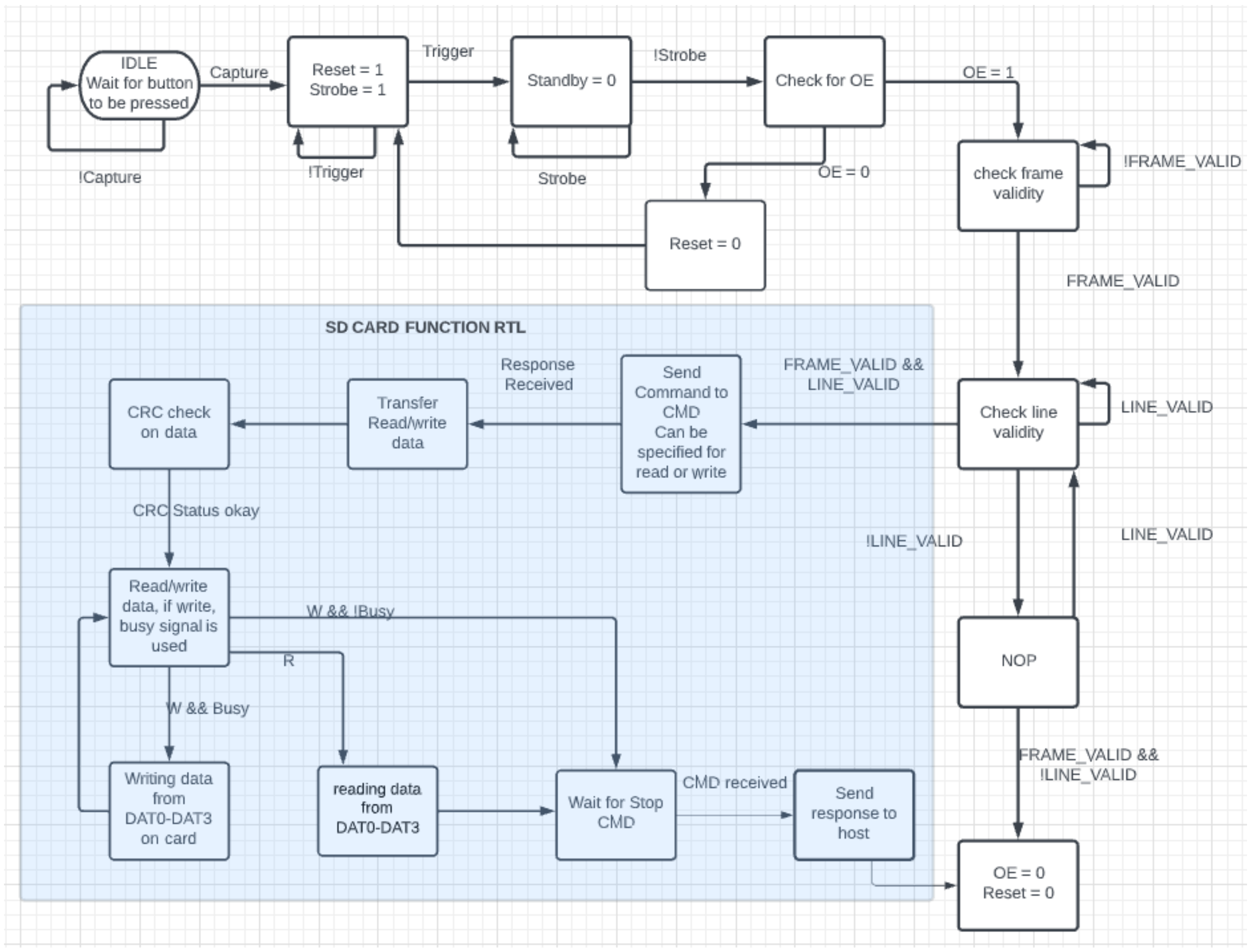


Write Operation of SD card:
The write operation of the SD card is initialized in the same way using a command and response on the CMD line. The data to be written on the SD card is received via the data lines and CRC status, busy status are sent back to the host via DAT0. After data has been checked and CRC okays the data, a busy signal is asserted on DAT0 to signal the writing operation. It is not clear why this signal is asserted but I would assume it works to prevent the error of reading while writing to the SD. To stop or terminate a writing operation, a command token is sent to the SD card, data transfer is stopped and writing completed, then a response sent back to host.



El Maliki- Stella Kariuki- Ivan Gonzalez

B.5 **(2 pts) Port and Timing Interfaces:**
Extend your design (developed in Part A) to implement the port and timing interfaces determined in Questions B.3 and B.4. For the controller, you can stop at the state diagram.



When the FRAME_VALID && LINE_VALID is true, then the image should be stored in the SD card. The above boolean value should therefore be used as a trigger to send a command token to the SD card. When the SD card receives the command token, it evaluates the command and sends a response to the host. The above RTL diagram does not show the SD card state diagram but only shows an over the top functioning of the SD card read and store operation

When it comes to the state diagram of the SD card and the Command tokens sent on the CMD line, the following diagram illustrates.

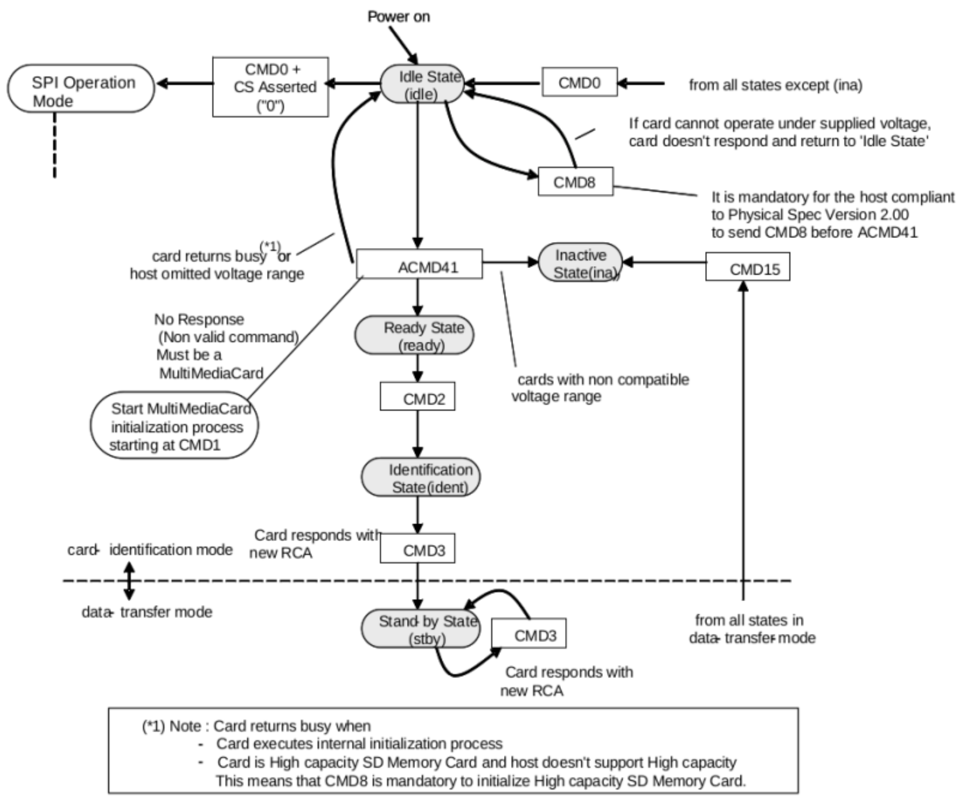

Power on
SPI Operation Mode
CMD0 + CS Asserted ("0")
Idle State (idle)
CMD0
from all states except (ina)
If card cannot operate under supplied voltage, card doesn't respond and return to 'Idle State'
CMD8
It is mandatory for the host compliant to Physical Spec Version 2.00 to send CMD8 before ACMD41
card returns busy (*1) or host omitted voltage range
ACMD41
Inactive State(ina)
CMD15
No Response (Non valid command) Must be a MultiMediaCard
Start MultiMediaCard initialization process starting at CMD1
Ready State (ready)
cards with non compatible voltage range
CMD2
Identification State(ident)
Card responds with new RCA
CMD3
card- identification mode
data- transfer mode
Stand-by State (stby)
CMD3
Card responds with new RCA
from all states in data- transfer mode
(*1) Note : Card returns busy when
- Card executes internal initialization process
- Card is High capacity SD Memory Card and host doesn't support High capacity
This means that CMD8 is mandatory to initialize High capacity SD Memory Card.

Figure 4-1: SD Memory Card State Diagram (card identification mode)

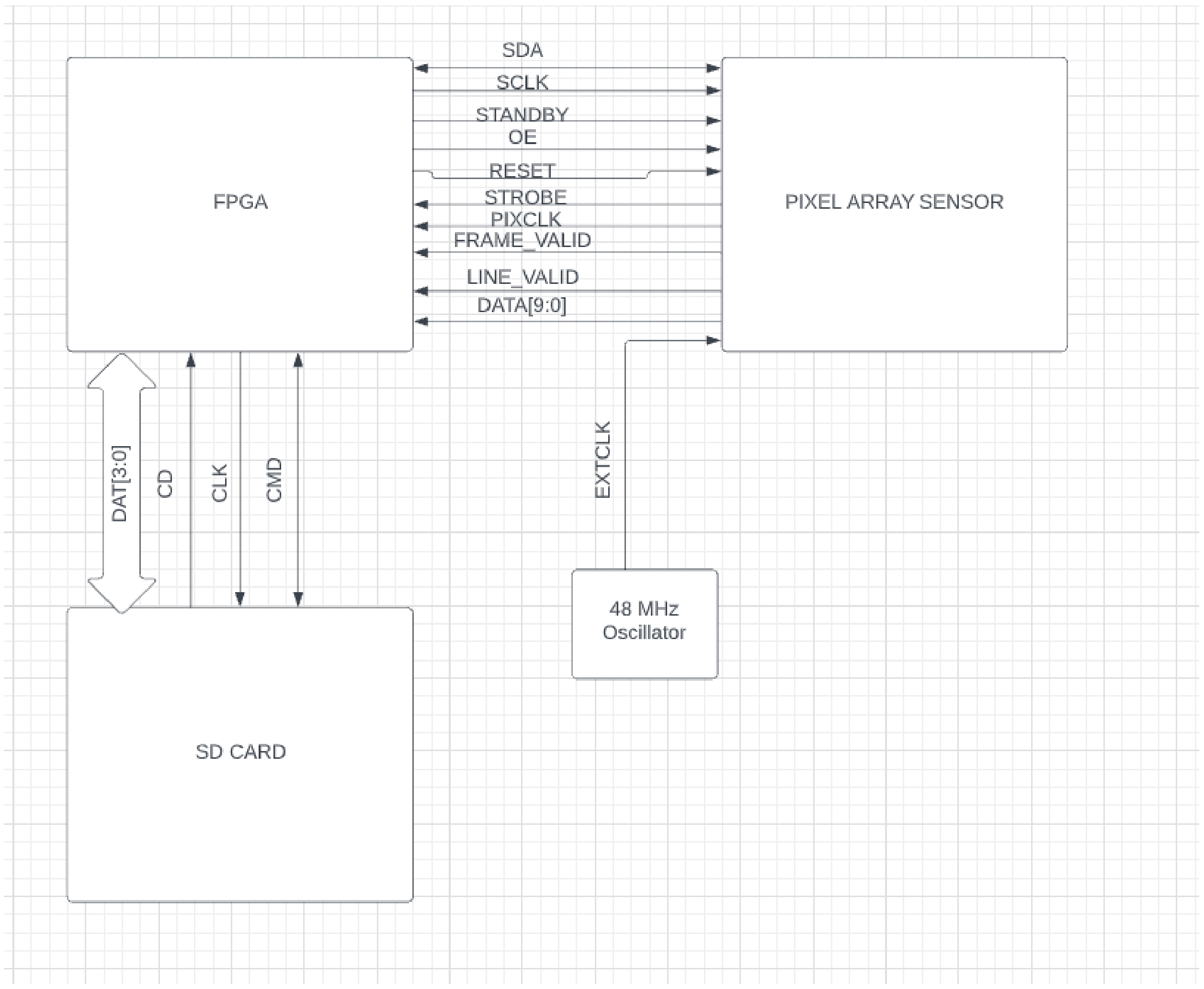
The card has 3 main operation modes:

1. Inactive (Ina): the card is in an inactive state
2. Card Identification (CID) mode: The card can be in an idle, Ready or identification state.
3. Data transfer mode:Include Standby state, Transfer state, Sending-date state, receiving-data state, Programming state and Disconnect state.

The following command tokens are specified:

CMD0 - reset all of the cards to IDLE mode
CMD2 - this asks the cards to send its CID numbers back on the CMD lines
CMD3 - this asks the card to issue a new corresponding address
CMD8 - if supplied voltage is not sufficient for card operation, then the card will not respond and return to the IDLE
CMD15 - forms all states in the data transfer mode.
ACMD41 - starts the multimediacard initialization process and requests the card to send the operating response registers.

B.6 **(1 pt.) Detailed Schematic:**
Extend the detailed schematic of your partial design (developed in Part A) to include the memory. Identify any other components that are required. Show these components as well in the schematic.



El Maliki- Stella Kariuki- Ivan Gonzalez

B.7 **(1.5 pts) PC Interface**

Choose a suitable interface (serial/parallel/wireless) between the camera and PC such as USB, Bluetooth, etc. Suggest an off-the-shelf solution to implement this interface. You can "drop in" an existing design provided by the interface vendor. **You need not extend the camera controller for this interface. However, you should include the interface cost in your final cost estimation.**

To have a stable connection between the camera and the PC we would be using a USB interface. An USB Host Controller manages the communication between the camera controller and a PC or any device that is connected.

This hardware device is responsible for:

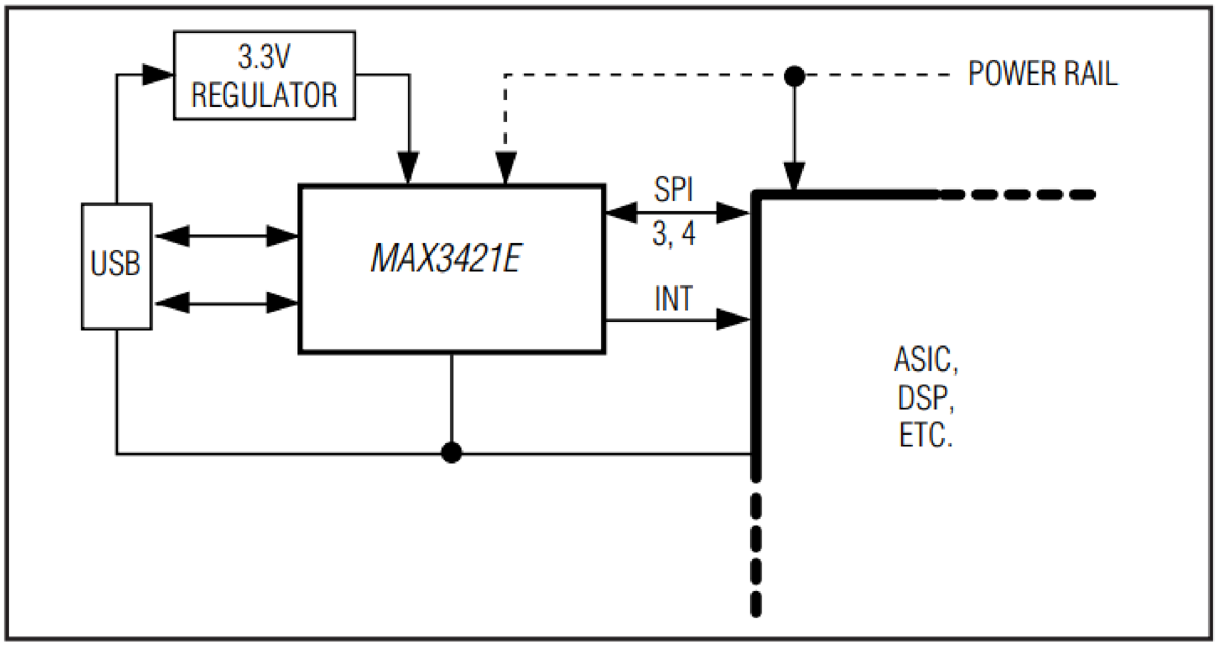
- ➔ *Detecting when a USB device is connected*
- ➔ *Monitoring device status*
- ➔ *Providing power to attached USB devices*
- ➔ *Managing control and data flow between the camera and the PC*
- ➔ *Checking the basic validity of bus transactions*

The Host Controller that we chose is the USB Peripheral/Host Controller with SPI Interface by Maxim Integrated. The USB peripheral/host controller is designed with a digital logic and analog circuitry that is able to implement a full-speed USB peripheral or a full-/lowspeed host compliant to USB specification rev 2.0.

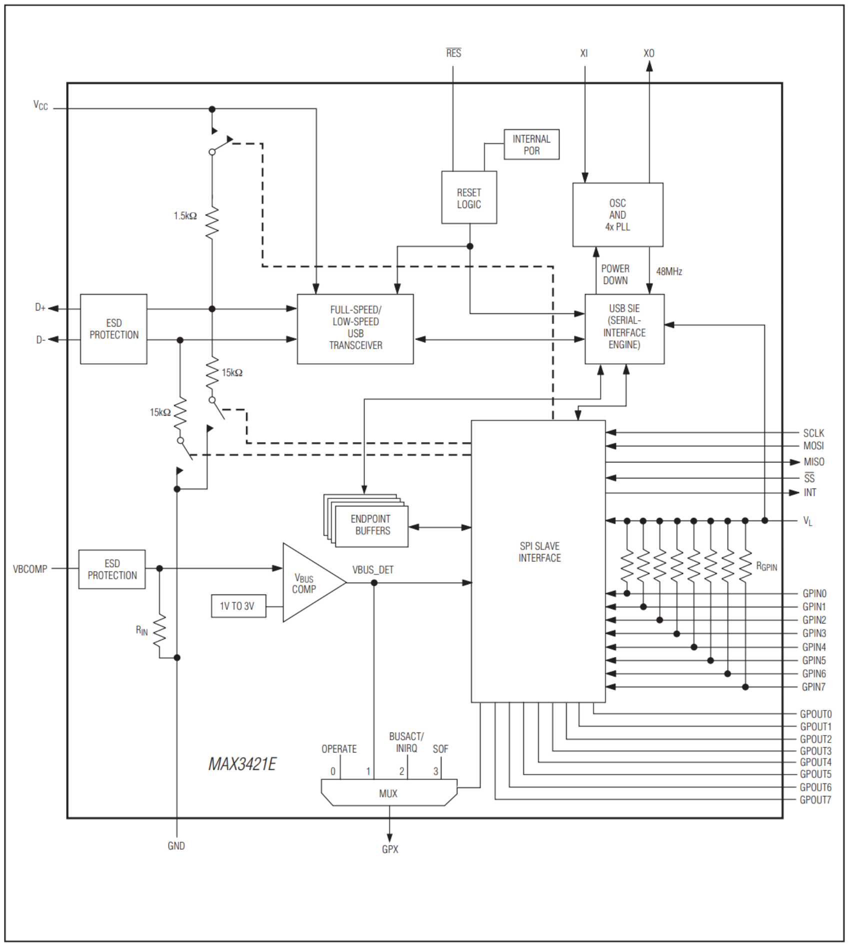
Main features:

- ➔ *Complies with USB Specification Revision 2.0*
- ➔ *Integrated USB Transceiver*
- ➔ *Programmable 3- or 4-Wire, 26MHz SPI Interface*
- ➔ *Intelligent USB SIE*
- ➔ *Automatically Handles USB Flow Control and Double Buffering*
- ➔ *Firmware/Hardware Control of an Internal D+ Pullup Resistor (Peripheral Mode) and D+/DPulldown Resistors (Host Mode)*
- ➔ *Host Controller Operates at Full Speed or Low Speed*

General diagram of the Host Controller connected to a larger chip:



Functional Diagram of the controller:



El Maliki- Stella Kariuki- Ivan Gonzalez

B.8 **(1.5 pt.) Estimations**
Estimate: (a) the maximum number of images we can store in the memory; (b) the time required to store/retrieve one image; and (c) the approximate dollar cost to prototype the camera (excluding costs for PCB design and manufacturing, component soldering, and testing).

A.

Image Resolution: 2,048 x 1,536 pixels = 3145,728 pixels
*Total number of bits: 2048 * 1538 * 10= 31,498,240bit= 3.93728MB*
Total size per picture: 3.93728MB => 0.00393728 GB
Total memory: 64GB
Total pictures that can be stored in the memory: 64GB / 0.00393728 GB = 16,254 Pictures

B.

The camera is 3.1 MP (total pixels / 1,000,000), 3.93728MB per image.
The size of the image do not get compressed and the memory that we selected has a max reading speed of 100MB/sec and a writing speed of 62 MB/sec therefore:
Writing time: 63.5045 ms
Reading time: 39.3728 ms

C.

FPGA board: Mimas V2 Spartan 6 FPGA Development Board with DDR SDRAM SKU: FPGA006 — Cost: $ 71.99

Crystal Oscillator 48 MHZ: NX2016SA-48M-EXS00A-CS08718 —Cost:$0.74

Image sensor: MT9T031- Cost: $16.9

Button:Momentary Pushbutton Switch - 12mm Square —Cost: $0.55

MicroSD memory: $19.99

USB Host Controller: $13.87

Total Cost:$124.04