CDA 4203 Sec 001
Spring 2015
Computer System Design

Mini-Project: Digital Camera Design

By *Automagically*

Sean Murphy
U49850246
(33%)

Jesse Walton
U89823440
(33%)

Austin Matthew
U89904116
(33%)

Mini-Project: Simple Digital Camera Design
Due: 11:59PM, Monday, 30th March 2015
Submission only by Canvas

Today's Date:	3/25/2015
Team Member Names: Your U Numbers: (Up to 3 members per team)	Sean Murphy Austin Matthew Jesse Walton
Work distribution	*Sean: State diagrams generation, math for total cost, capture time, max image storage, PC interface.* *Jesse: Schematics, timing diagrams, FSM, compact flash interface.* *Austin: diagram and schematic prettifying. windowing function, description of features for the data sheet problems, port descriptions.*

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

Total number of person hours spent:	5 meetings of 2 hour length, and one meeting of 5 hours. 15 “group” hours total. **45 man hours total**
Exercise difficulty: (Easy, Average, Hard)	Average (long)
Issues you ran into:	FInding data sheets for certain parts(we couldn't use are first choice of CF card because the data sheet was no where to be found) Intent of questions was a bit vague and confusing
Any suggestions to improve this project:	Either combine the two parts completely, or assign them separately.
Any other feedback:	Having multiple systems to choose from rather than just a camera.

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:

- Take pictures in < 1 second (as fast as possible)
- Take as many pictures as possible
- Pictures must be at least 1MP
- Sub $200
- Be controlled by external controls

Additional future features:

- Image compression
- USB flash drive compatible
- Battery
- Quick start time
- Adjustable picture size
- Video mode

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)*

Features:

- Windowing: Used to drop our capture resolution to 1024x1024 of a possible 2048x1536. Set a Start and “Run” register to truncate image.
- High resolution (2048x1536 pixels)
- Fast capture.
- Bulb mode and Exposure modes - not going to be used, but they would allow for external shutters.
- Global reset release, horizontal and vertical binning, column and row skip modes is also not required.
- Simple two-wire serial interface will be used to communicate settings of the image sensor, but not the actual image data itself.

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

Camera Controller Block (connecting to):

- **Image Sensor:** (page 7-9) [Broken into categories]
  - I2C - used to modify registers on image sensor, which affect its settings. Can also take picture.
    - **SCLK** (duplicated in Clock section)
    - **SDA**
  - Clocks(synced with sync signals)
    - **EXTCLK**: This is the master clock going into the sensor. It has a maximum of 48 MHz.
    - **SCLK**: Serial interface clock.
    - **PIXCLK**: Output Pixel clock, validates pixel data outputs on falling edge. Slaved to master clock.
  - Sync Signals (synced with clock)
    - **LINE_VALID**: Timing signal used to start/stop blanking per line.
    - **FRAME_VALID**: Timing signal used to start counting the blanking before valid data of each frame.
  - Additional
    - **TRIGGER** - initiates image capture
    - **GSHT_CTL** - dunno, sounds fancy tho
    - **STROBE** - used for dance parties
    - **DOUT**[9:0] - 10 bit pixel data, valid on falling edge of **PIXCLK**. We only need 8 bit data, so the two LSB lines are ignored.
- **[Included in part B]** **CF Memory Interface**[1] **Using "PC Card Memory" Mode**
  - D[15:0] - data bus
  - A[10:0] - address bus
  - nCS0, nCS0 - chipselect lines
  - nCSEL - cable select
  - nINPACK - input acknowledge
  - nIORD - IO read strobe

[1] http://cache.freescale.com/files/32bit/doc/app_note/AN2293.pdf also http://rumkin.com/reference/aquapad/media/cfspc3_0.pdf

    - nIOWR - IO write strobe
    - nIORDY - IO ready
    - nWE - write enable
    - nOE - output enable
    - nREG - register select
    - RESET - active high reset
    - nIOIS16 - 16 bit mode
    - RDY - ready

- **User Controls**
    - Power Button
    - Capture Button
- **PC Interface (UART)**
    - TX
        - din [7:0]
        - write
        - (serial_out)
        - buffer_full
        - en_16_X_baud
        - reset_buffer
        - clk
    - RX
        - dout [7:0]
        - read
        - (serial_in)
        - reset_buffer
        - buffer_full
        - data_present
        - clk

*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)*

Setup

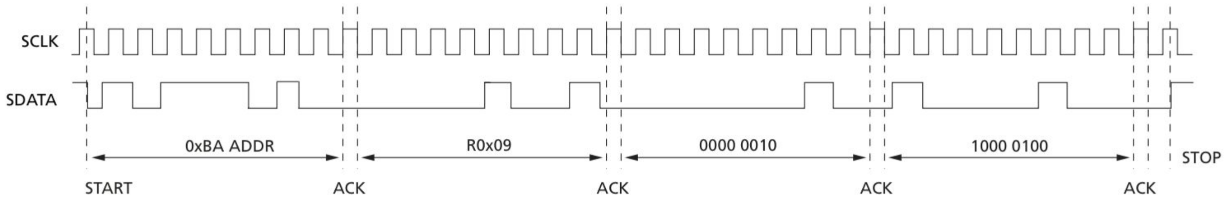
The image sensor is initialized by setting register values via the I2C protocol. The following describes how the I2C protocol timing works and then specifies which registers are to be set in this way.

The pertinent information needed to perform I2C communication with the MT9t031 is as follows:

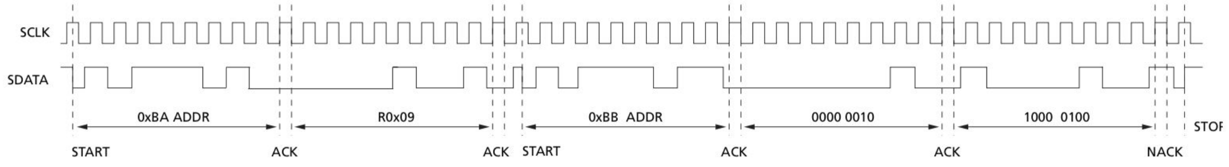
- Device Address - this 8 bit value contains both the address of the device as well as the intended mode (read/ write). The image sensor has an address of 0b1011101X (7 addr. + 1 mode). The last bit is specified as follows:
    - Write: 0b10111011
    - Read: 0b10111010

I2C Timing Diagram Examples[2]

Example: write 0x284 to register 0x09



Example: read 0x284 from register 0x09



2 MT9T031 Datasheet

- <u>Example Read/ Write Protocol</u> -
    - **Write to Register:**

	INFO:	SDATA:
1.	Start Bit:	High to Low (1 bit)
2.	Slv Addr & mode:	0xBA (7 bit addr + 1 bit mode)
3.	ACK:	Low (1 bit)
4.	Set Register:	0x?? (8 bits)
5.	ACK:	Low (1 bit)
6.	Write[15:8]:	0x?? (8 bits)
7.	ACK:	Low (1 bits)
8.	Write[7:0]:	0x?? (8 bits)
9.	ACK:	Low (1 bit)
10.	Stop bit:	High (1 bit)

    - **Read from Register:**

	INFO:	SDATA:
1.	Start Bit:	High to Low
2.	Slv Addr & mode:	0xBA (last bit is write)
3.	ACK:	Low
4.	Set Register:	0x??
5.	ACK:	Low
6.	Start bit:	High to Low
7.	Slv Addr & mode:	0xBB (last bit is read)
8.	ACK:	Low
9.	Write[15:8]:	0x??
10.	ACK:	Low
11.	Write[7:0]:	0x??
12.	ACK:	Low
13.	Stop bit:	High

- <u>Registers to set:</u>
    - windowing:
        - R01 (0x01)=0x0200, // Row start: 512
        - R02 (0x02)=0x0200, // Column start: 512
        - R03 (0x03)=0x03FF, // Row size: 1023
        - R04 (0x04)=0x03FF, // Column size: 1023
    - integration time (exposure time) - [Leave as default. This should be set by the “optical engineer” on the design team or by trial and error dependent on the lens.]

- triggering mode
    - R30(0x1E)=0x0740, // Extend strobe, enable strobe, snapshot mode, noise suppression
    - R32(0x20)=0x2001, // Enable output of bad frames

Trigger

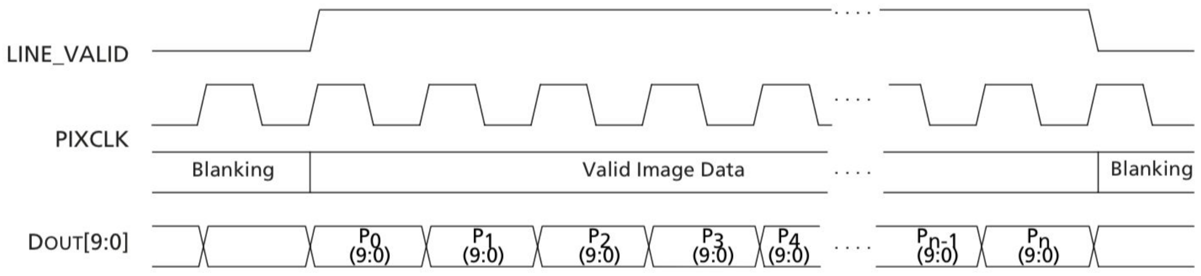
After setup, the only I2C communication is used to initiate the taking of the picture:

- trigger:
    - R0B (0x0B) = 0x0001 // take picture

Image Capture

Once trigger is initiated by setting R0B to 0x0001, the image capture process is started and R0B is automatically set to 0x0000. The image capture process is as follows:

Example: Row Read[3]



- **PIXCLK**: Clock signal from external source controls timing of FRAME_VALID, LINE_VALID and DOUT.

- **FRAME_VALID**: Indicates that there is data to be transmitted from the image sensor.
    a. High - image sensor has lines to transmit
    b. Low - no data to transmit

- **LINE_VALID**: Indicates that a row of pixels is to be transmitted.

[3] MT9T031 Datasheet

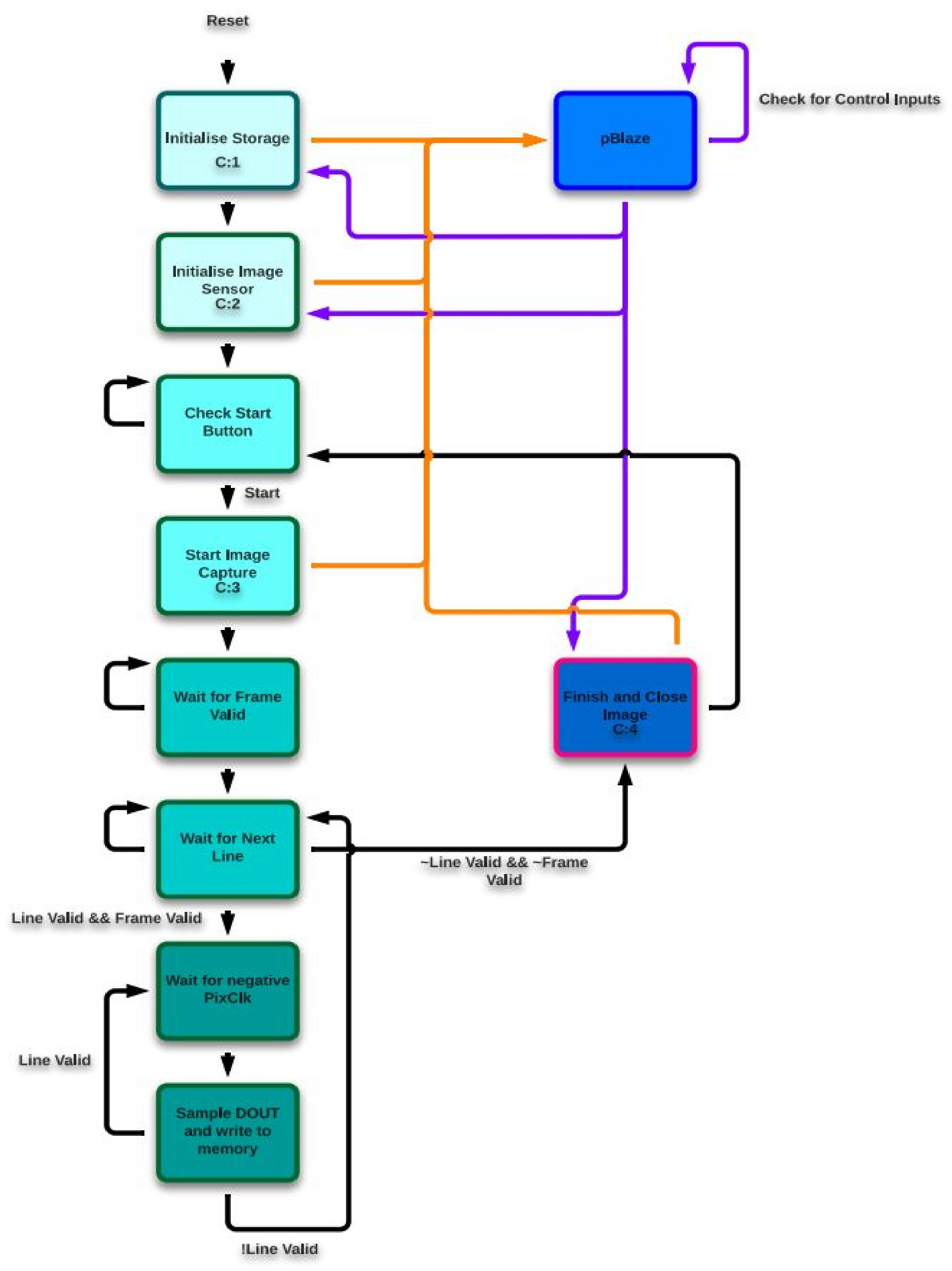
    a. High - image sensor is currently transmitting pixel data.
    b. Low - blanking time, no data is transmitted during this time

- **DOUT [9:0]:** Data output containing 10 bits of pixel data, transmitted on the falling edge of PIXCLK when FRAME_VALID and LINE_VALID are high.

Typical Operation (image sensor):

1. After setup, the image sensor waits until the 'trigger' register (0x0B) is set to 0x0001. When 0x0B is set, the image sensor automatically resets it to 0x0000.
2. The image sensor sets FRAME_VALID high when it is ready to begin outputting the image. FRAME_VALID will remain high for the remainder of the image capture cycle.
3. For each row (1024 total), the sensor will set LINE_VALID high and transmit pixel data for each column (1024 total) then will set LINE_VALID low during the blanking period.
4. Once all lines have been transmitted, FRAME_VALID will be set low.

*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)*



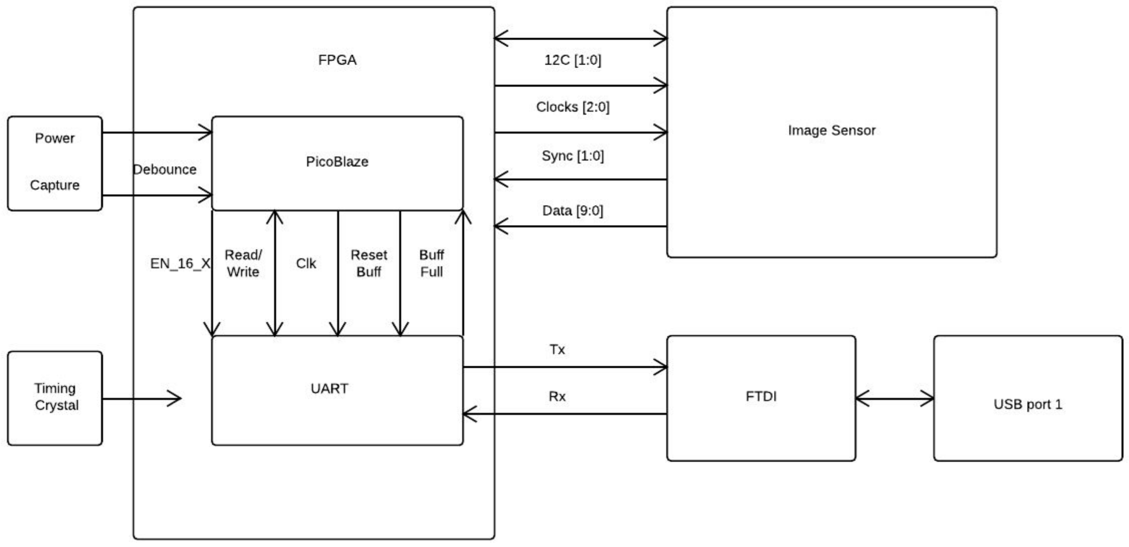
C1:Check CF card to find next file.
C2:Writes to registers that hold window dimensions, and trigger modes
C3:trigger.
C4:Tells pico to finish file and move to next file.

Included in the FSM is communication with a pBlaze microcontroller to handle the I2C communications and some of the CF card setup. The FSM signals the pBlaze to complete a task and waits for the pBlaze to signal that it has finished before jumping to the next state.

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)*

Required Components:

1. Image sensor
2. FPGA
3. CMOS oscillator
4. Buttons



*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)*

Image Capture Time

How long will image capture take? For the most part, this will be a measurement of how long our device takes from when the user presses the "capture" button to the device being ready to capture another image. Time to startup is something to consider as well, but with the typical use case, that shouldn't matter as much. Either way, startup time should be under one second. There is no operating system to initialize. Just about all that there is to do is run the initial configuration for the pixel array and to check the filesystem of the CF memory card. The initial configuration is only a handful of I2C writes, and the memory configuration is out of the scope of the first half of this project.

Image capture and memory writing is performed in an FSM. If this were managed by a PicoBlaze, then that could be a bottleneck. As is, the bottleneck will be either the memory device or the pixel array pixel clock speed. In the coming sections, we will be examining Compact Flash cards as our memory device. CF is selected because, in theory, it has high enough bandwidth to directly write at the maximum data speed of the pixel array. Assuming this is the case, the question about the time for image capture is a question about how quickly image data is clocked from the pixel array.

Image capture is started when a user presses the capture button. Because the FSM is waiting for this button press, the latency here is negligible (~6 master clock cycles for ~3 pBlaze instructions). From here, the pBlaze initializes an image capture by writing 0x0001 to R0x0B. This will be an 8-bit device address, an 8-bit register address, and a 16-bit register value. There are also about 8 bits of start/stop/ack. In total, this is 40 bits. Serial clock has a maximum speed of 1.2MHz. Slightly more standard I2C data rates are 100kHz for low speed and 400kHz for high speed. We want high speed, but we want leniency in timing, so we will assume that 500kHz is our data rate. It's plenty fast while still being quite within speed bounds. This gives us a value of 100us for the total transfer.

After this, the pixel array resets all of its rows in sequence. The mode we are using does not list a time for this, but other modes specify 18,820 pixel clocks for the minimum global shutter (used as reset) time, so this value is assumed. After this, the strobe line is asserted for one row time. Strobe then goes low, and the sensor waits 16 row times more before starting to output the data.

We have selected 1024 rows of image data. Each row has 1024 pixels of usable data then 511 pixel clocks (P1+P2+P3 from the datasheet Table 4). After this, we no longer need to capture image data, so Frame Restart R0x0B could be reasserted to start the process again instead of sitting through the vertical blanking.

This totals to 18820+(1+1024)*(1024+511) = 1592195. Assuming a relaxed pixel clock speed of 25MHz, this process takes 63.7ms from reset to finishing getting the image data. This entirely totals to 63.8ms per image capture from the user pressing the button to the image being finished.

Total Part Cost

What is the total dollar cost for the front end interface?

MT9T031 Image Sensor	$15.86[4]
Spartan 6 FPGA	$16.52[5]
Cheap momentary buttons	$0.10
Pinhole camera lens[6]	$1.00
100MHz CMOS oscillator	$1.13[7]
Total:	$33.48

[4] https://www.verical.com/pd/aptina-imaging-image-sensor-MT9T031C12STC-314896
[5] http://www.digikey.com/product-detail/en/XC6SLX9-2TQG144C/122-1745-ND/2339919 chosen because it is the most popular Spartan 6 on Digikey and fulfills our hardware needs
[6] Chosen because of cost, an impressive depth of field, and no need for focusing hardware
[7] http://www.digikey.com/product-detail/en/501ACA100M000DAG/336-2993-ND/4582103

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.

We need ~60MB/s write speed for the max rate of data from the pixel controller. Options are as follows:

- RAM - super fast but small and expensive-ish; if we were not limited to a single memory device we could get some due to fast storage before we offload to slower external storage. The RAM in the Atlys board can hold 128 1MP 8bit images internally.
- SD card - probably best for our budget, but they aren't fast enough for raw write speed from the pixels.
- SSD - ~8 times faster than raw pixel data rates but expensive, and we really won't gain anything from this much speed.
- HDD - too slow for raw writes, but great storage per dollar.
- Tape drives - best price per gigabyte, but the reader/writer bay is ~$400 which is out of budget.
- **Compact Flash (CF) card** - Fast enough for raw pixel data, and $73 will get us 64GB (61,000 images). We will choose this option as its speed is greater than that of our other components so it will not cause a bottleneck, yet has plenty of memory per dollar.

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

Memory Component: Transcend Industrial CF Card

Model #: TS16GCF200I[8]

Features:

- Single Power Supply: (approx. read/ write mA)

3.3V	(140 / 143)	or
5V	(173 / 153)	

- Operation Modes: PC Card Memory Mode
- Durability of Connector: 10,000 times
- Support for S.M.A.R.T.
- Read/ Write Speed:

Rand. Read:	~46.8 MB/s
Rand. Write:	~11.7 MB/s
Seq. Read:	~50.4 MB/s
Seq. Write:	~39.7 MB/s

We have chosen to use the Transcend CF card to act as our main memory as it is able to store image data as fast as it is output from the image sensor. The CF card also provides for robust data preservation and security features for preserving data to the maximum extent possible.

[8] http://www.farnell.com/datasheets/1683451.pdf

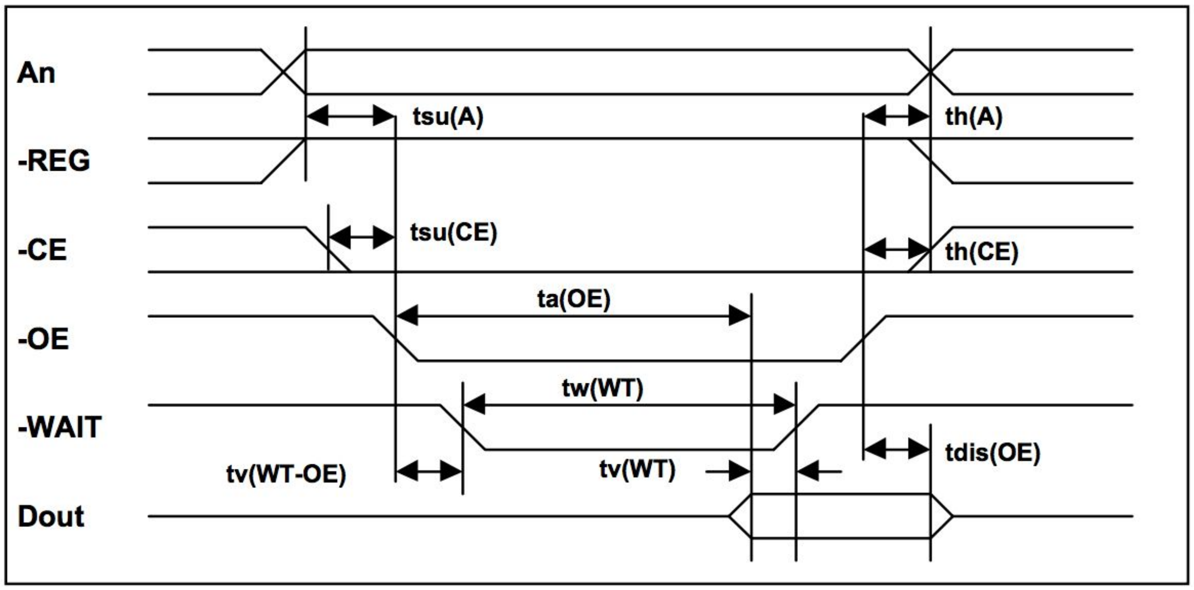
B.3 ***(1 pt.) Port Interface:*** *Define the port interface of the memory with the camera controller. Briefly describe the purpose of each port.*

CF Memory Interface

- D[15:0] - data bus
- A[10:0] - address bus
- nCE0, nCE0 - chip enable lines
- nCSEL - cable select
- nINPACK - input acknowledge
- nIORD - IO read strobe
- nIOWR - IO write strobe
- nIORDY - IO ready
- nWE - write enable
- nOE - output enable
- nREG - register select
- RESET - active high reset
- nIOIS16 - 16 bit mode
- RDY - ready

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

Read

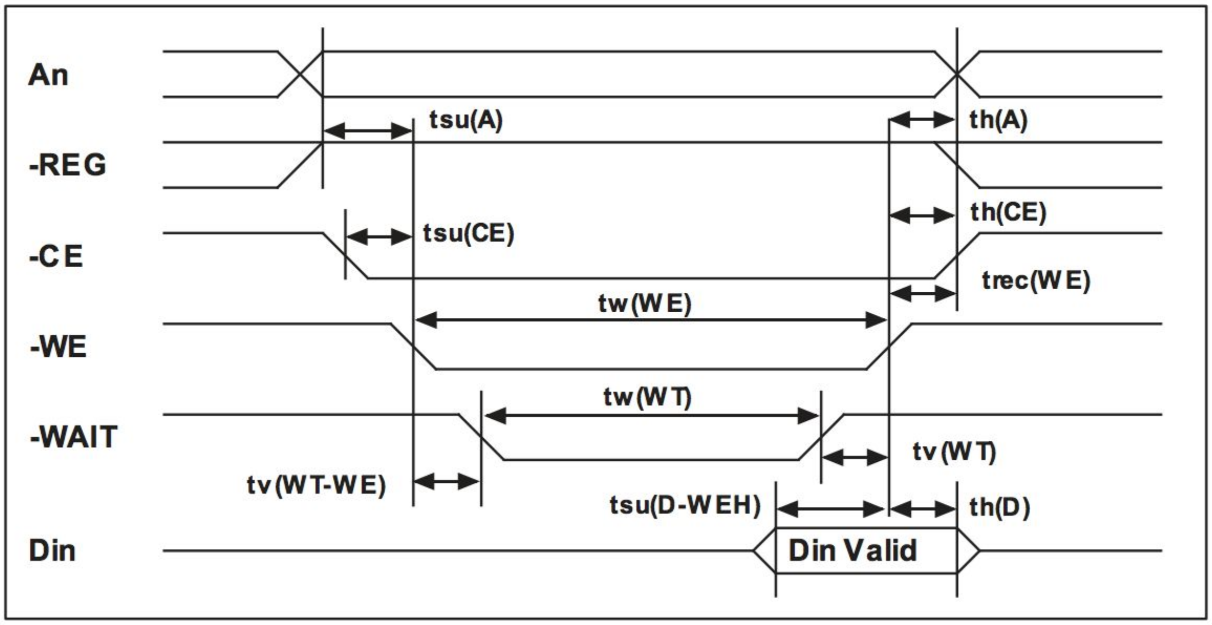


Memory Read Timing Diagram[9]

For the process of memory read, the address lines are set, and the nREG line is asserted to indicate to the CF card that memory is being accessed instead of registers. nCE is deasserted to enable the chip, and nOE is deasserted to signal the CF card to enable its output buffers on the bidirectional data bus. The CF card deasserts nWAIT until the data is ready. Once nWAIT is asserted, the controller samples the data from the CF card and asserts nOE. The CF card now disables its output buffers.

[9] http://people.cis.ksu.edu/~dominic/mse/resources/compact_flash_spec_2.0.pdf page 37

Write

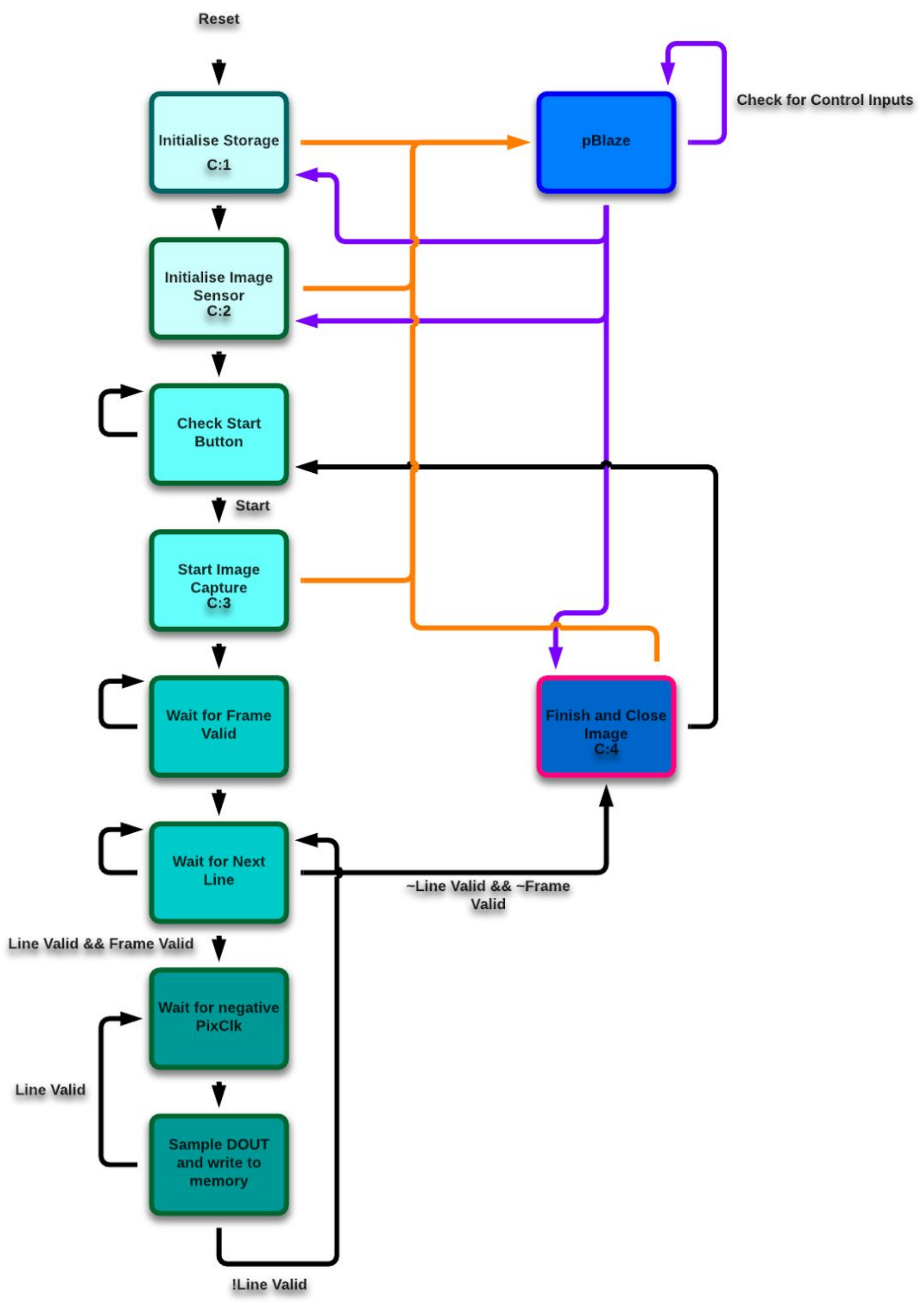


Memory Write Timing Diagram[10]

For the write process, as before, the address lines are set to the address of memory for the write, and the nREG line is asserted to signify memory access. The controller deasserts nCE to enable the CF card and deasserts nWE to enable write. The controller also outputs data on the data bus. The CF card will deassert nWAIT once it sees the write request then reassert once the data has been sampled and stored internally. Once finished writing its data, the CF card asserts nWAIT. The controller now asserts nWE to finish the transaction. The controller can now assert nCE to disable the CF card.

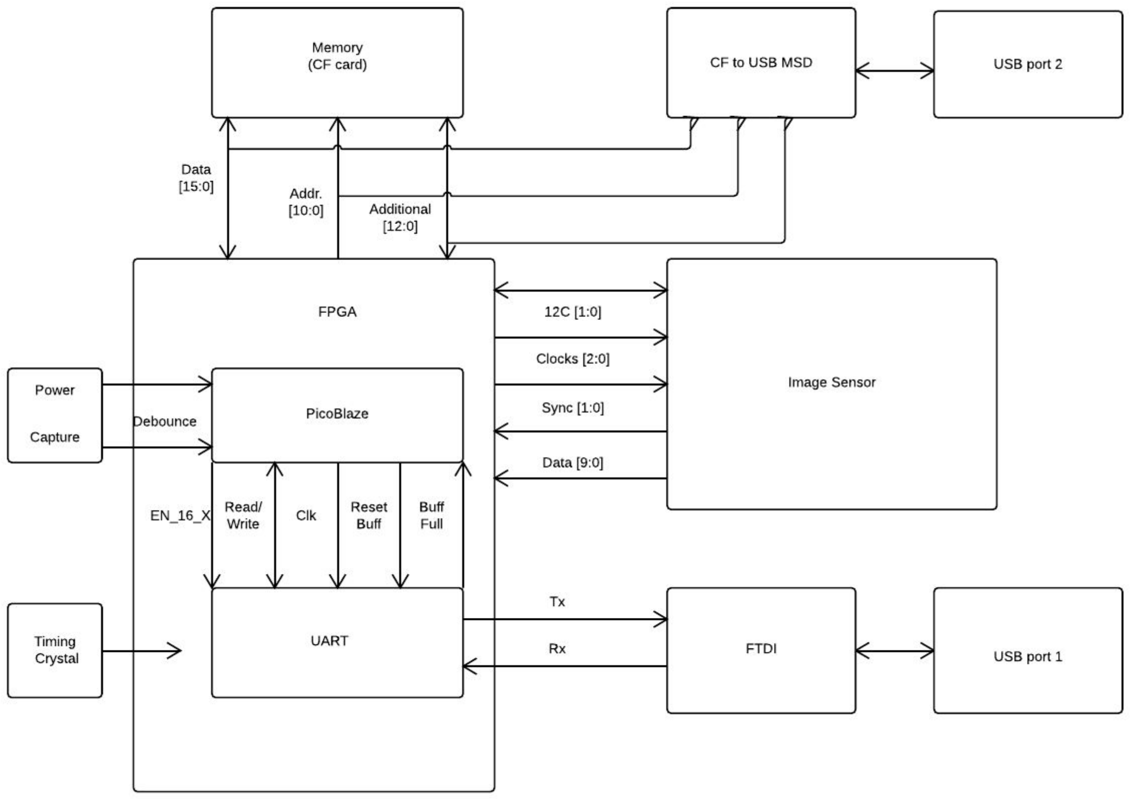
[10] http://people.cis.ksu.edu/~dominic/mse/resources/compact_flash_spec_2.0.pdf page 38

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.*



C1:Check CF card to find next file.
C2:Writes to registers that hold window dimensions, and trigger modes
C3:trigger.
C4:Tells pico to finish file and move to next file.

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.



B.7 ***(1.5 pts) PC Interface***
Choose a suitable interface (serial/parallel/wireless) between the camera and PC such as USB, Firewire, 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.***

The USB2601[11] by Microchip is a chip that, among other things, provides a bridge between USB and a CF memory card. This chip will allow the CF card in the camera to be directly accessed by a computer as a USB memory card. It has internal timing generation and internal voltage regulation which provides an extraordinarily "drop in" solution to our problem.

[11] http://www.microchip.com/wwwproducts/Devices.aspx?product=USB2601

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).

Images are 1024x1024 pixels. We are truncating the data to only 8-bits. For simplicity, we are implementing our own file format that is just raw pixel data without any metadata. Also, while compression would be useful in practice, we are keeping our data raw. In total, each image is 1MiB == 1.05MB. Assuming a 16GB CF card, this will give us slightly above 15 thousand images.

With our selected CF card[12], the given write speed is 39,689kB/s sequential and 10,739kB/s random. Our access is sequential, but for real world approximations, assume that we only get about half of rated sequential performance. This puts the total time to write an image at 53ms (which is less than the capture time of 64ms). Because capture time is more than write time, write time is dominated by the capture time, so effective write time is 64ms.

Reading an image is handled by the USB-CF bridge chip and a computer. The camera itself never reads an image. USB 2.0 is specified to have a transfer rate of 60MB/s[13]. A more important number would be the read speed of the USB-CF bridge, but the datasheet was sadly lacking in transfer speed details. The CF card has read speeds of 50,381kB/s sequential and 46,757kB/s random. If we assume that the USB-CF bridge chip operates at mostly the full speed of USB 2.0, then performance is dominated by the CF card. As before, the data is sequential, but for a real world approximation, 50% performance is assumed. This yields an image retrieval time of 42ms.

[12] http://www.farnell.com/datasheets/1683451.pdf

[13] http://www.pcworld.com/article/2360306/usb-3-0-speed-real-and-imagined.html

Approximate cost:

Camera Front End[14]	$33.48
USB-CF Bridge Chip	$3.79[15]
16GB Transcend CF Card[16]	$27.50[17]
Li+ Battery	$10.00
Case	$10.00
Other components[18]	$20.00
Total	$104.77

[14] Question A.7
[15] https://www.verical.com/pd/microchip-technology-hubs-USB2601-NU-05-805803
[16] Equivalent to but not the same the card as previously discussed
[17] http://www.amazon.com/Transcend-400X-Compact-Memory-TS16GCF400/dp/B002WE2PW8/
[18] USB ports, power supply chips, peripheral electrical components