Lens controller for Biomedical cameras

TA: Zhicong Fan
Professor : Viktor Gruev

Siddharth Sharma (sharma62)
Kevin Sha (ksha3)
Jihun Kim (jihunhk2)

OVERVIEW

- Introduction & Objective
- Design
- Project Build & Functional Test Results
- Successes & Challenges
- Conclusion & Areas of Improvement

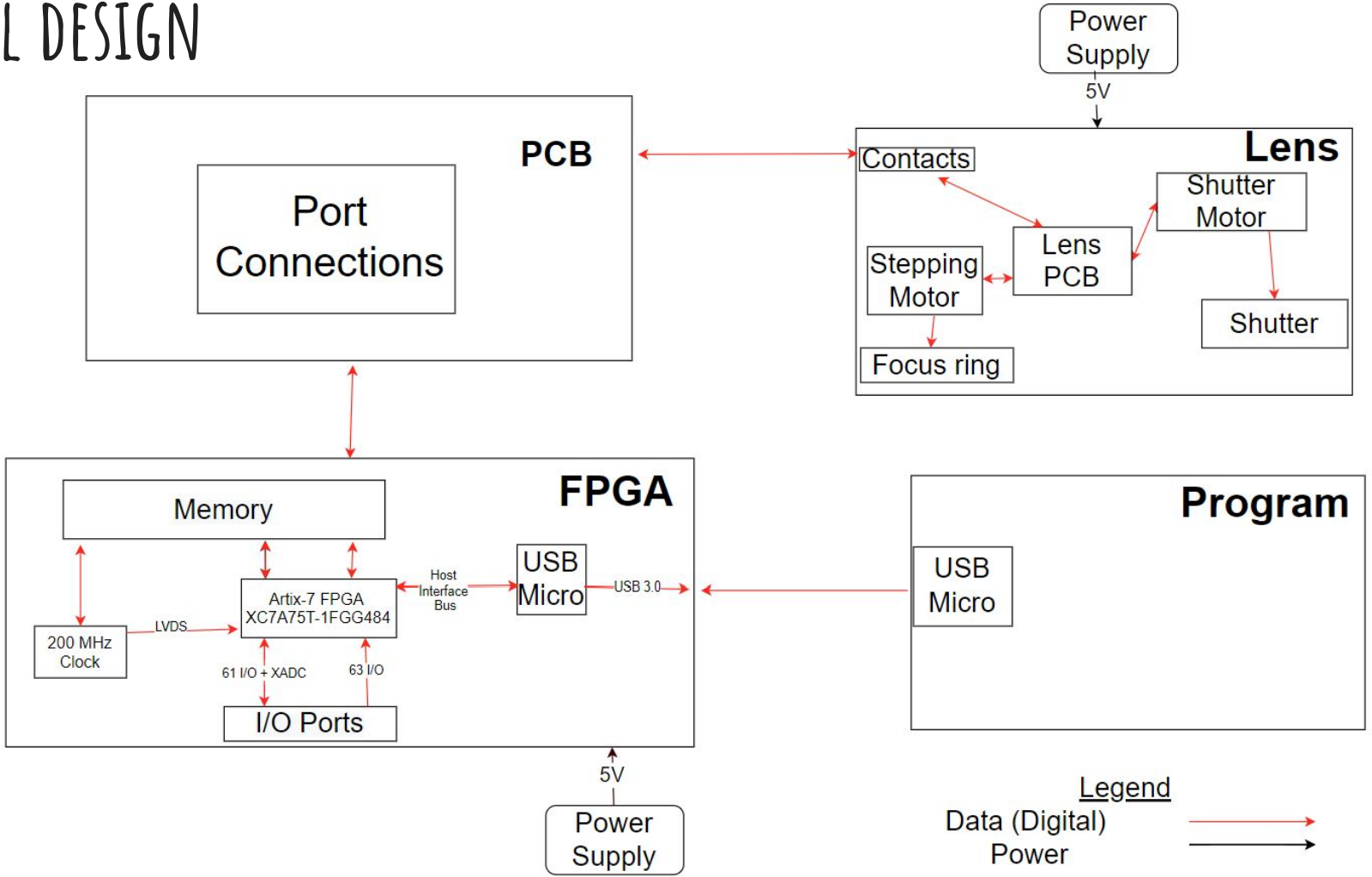
Introduction to project

- Margin of error in many medical operations is very slim
- One prime example is cancer treatment :
  - It requires high degree of accuracy
  - 25% of breast cancer patients, 35% of colon cancer patients, and 40% of head and neck cancer patients suffer from incomplete tumor removal
- Hence, this is a significant problem and requires solution.

OBJECTIVE

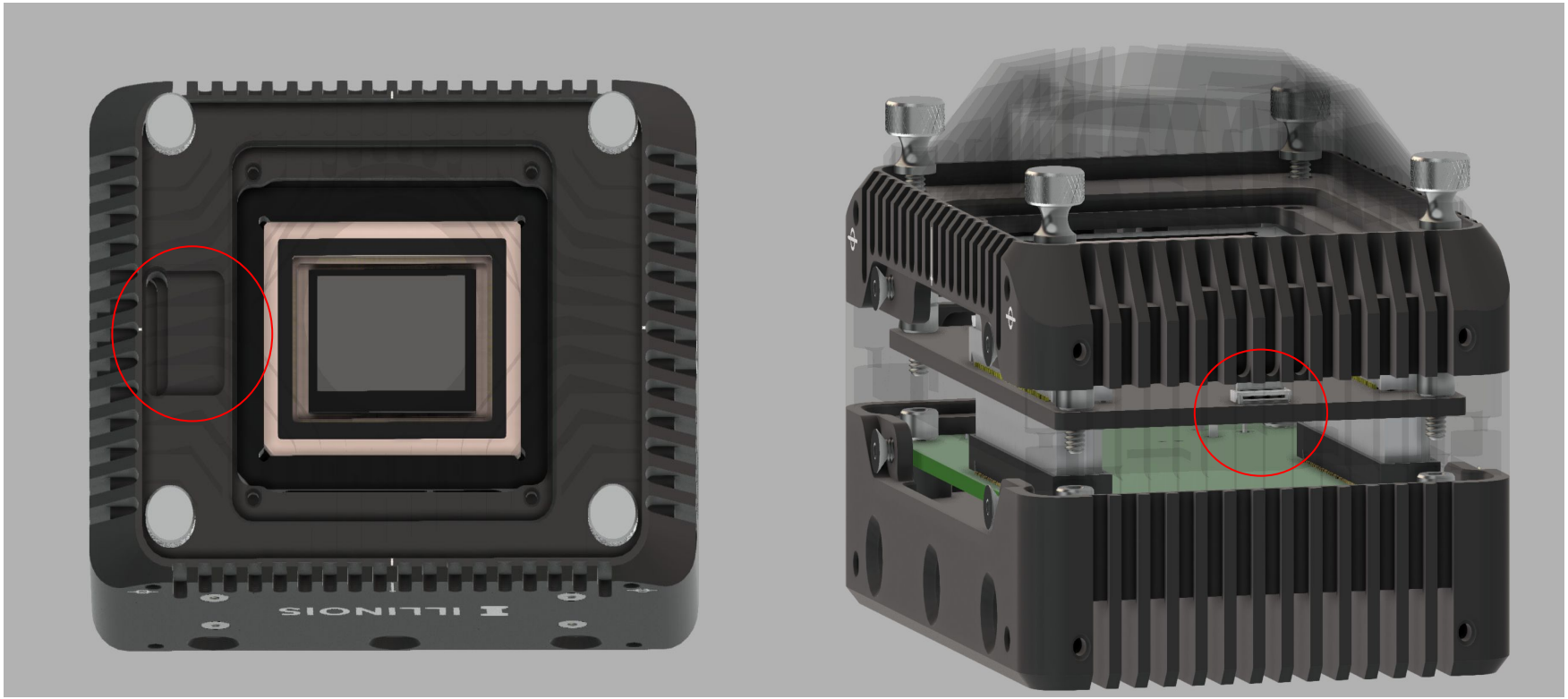
- To remotely control the camera’s focus and shutter for use in the operating room
- The desired goal is that the camera’s specifications can be adjusted to make sure that the entire tumor(s) is removed

Original design



Changes made since then: PCB Sub-section

- Redesigned sub-section completely since Design Document
- Originally thought that one flexible PCB would be enough to connect the FPGA and the lens mount
- Found that new system would have to be developed

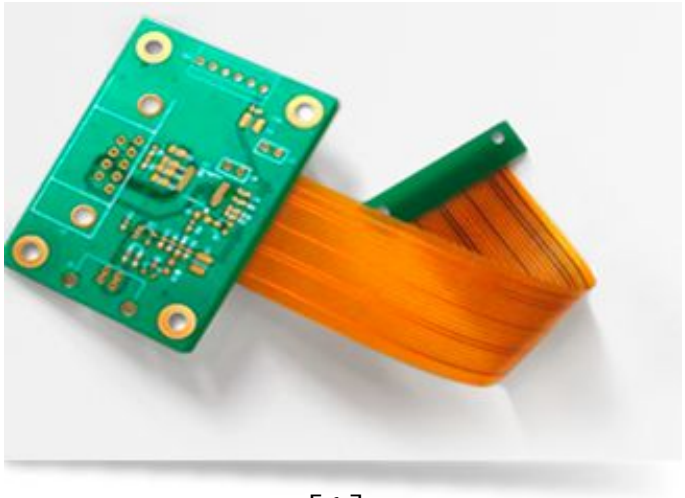

ILLINOIS

Project build: PCB Sub-section

- 2 PCBs:
  - A rigid-flex-rigid which would transport the signals through the tight constraints of the lens mount
  - A flex PCB, which will connect the signals from the FPGA to the the bottom of the lens mount

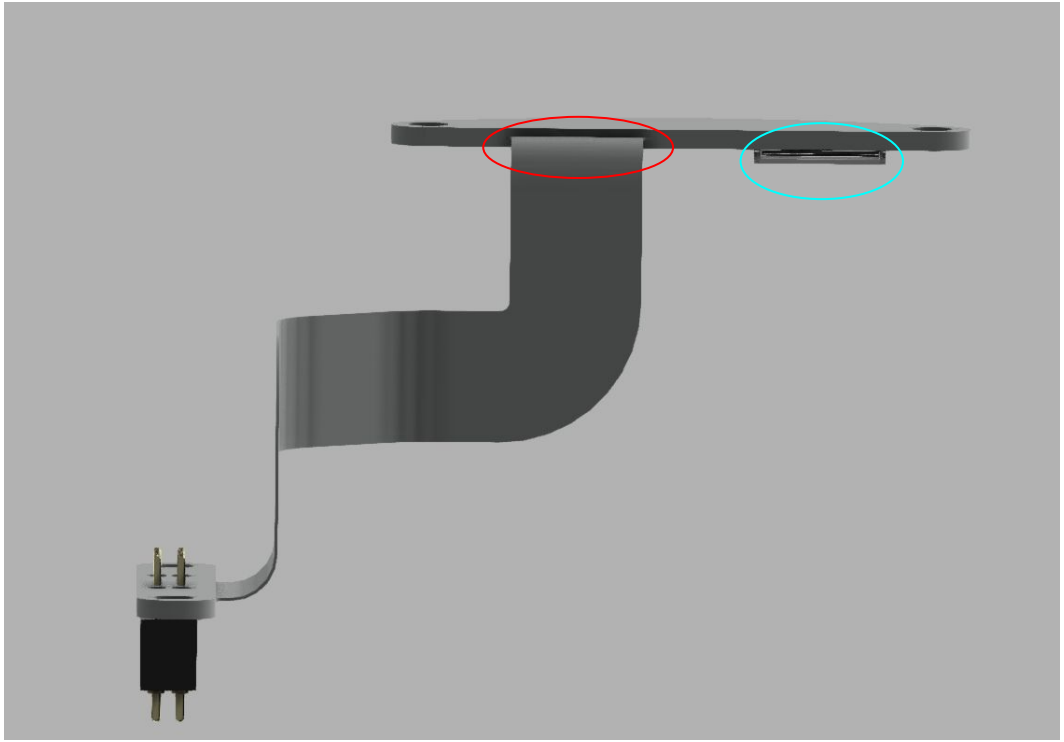
Rigid-flex PCBs

- Rigid section with flex section emerging from the middle
- Allows for rigid PCBs to be placed in tight regions



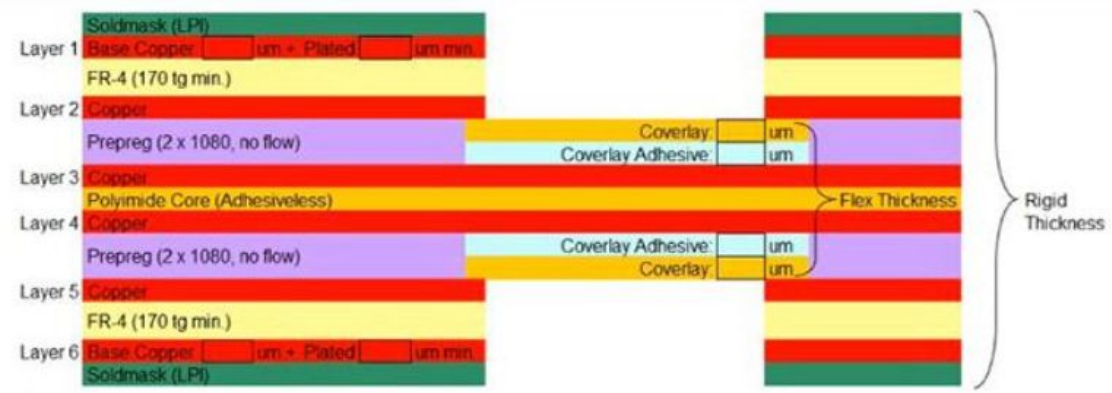
[1]

Emerging flex section connects two rigid sections

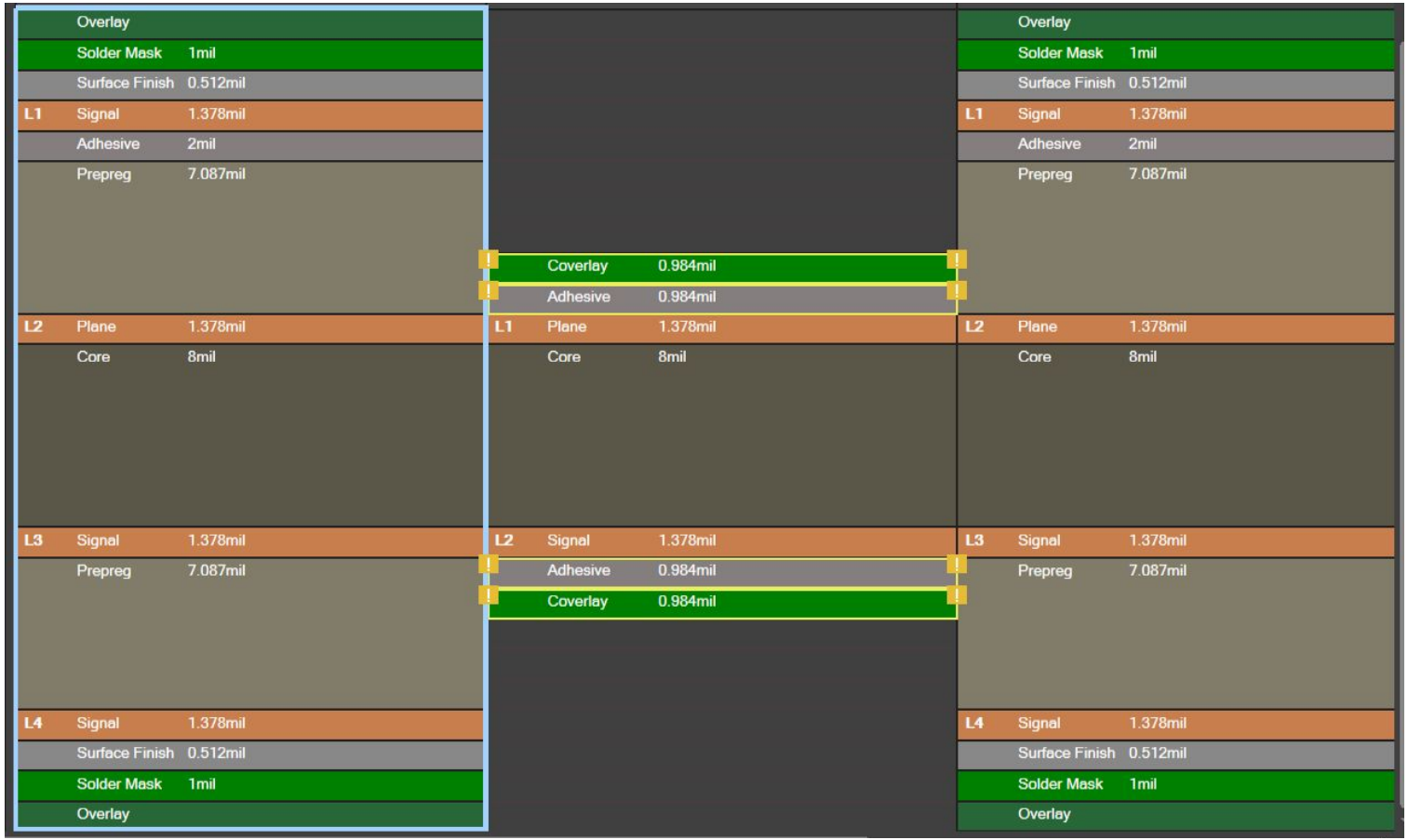


Layer stackup

- Important to correctly incorporate layers for optimum signal integrity
- Flex signal layers need to be properly extended

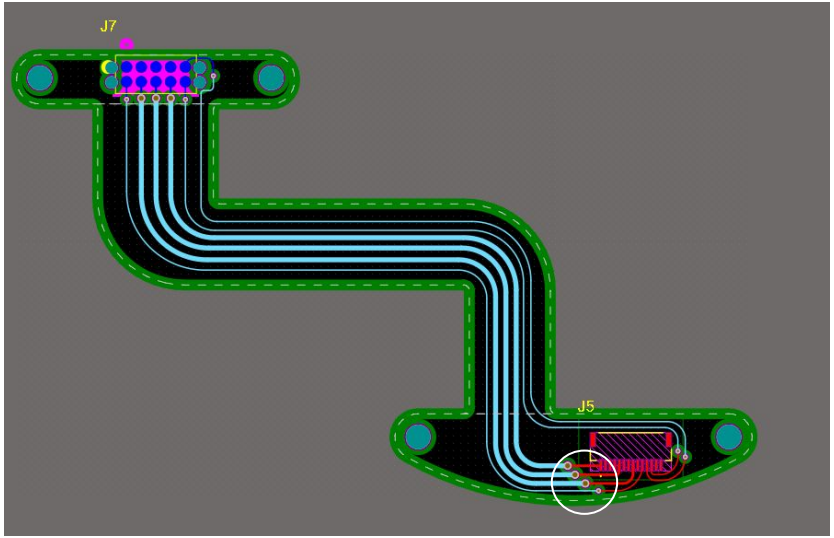


[2]


Overlay
Solder Mask 1mil
Surface Finish 0.512mil
L1 Signal 1.378mil
Adhesive 2mil
Prepreg 7.087mil
L2 Plane 1.378mil
Core 8mil
L3 Signal 1.378mil
Prepreg 7.087mil
L4 Signal 1.378mil
Surface Finish 0.512mil
Solder Mask 1mil
Overlay
Coverlay 0.984mil
Adhesive 0.984mil
L1 Plane 1.378mil
Core 8mil
L2 Signal 1.378mil
Adhesive 0.984mil
Coverlay 0.984mil
Overlay
Solder Mask 1mil
Surface Finish 0.512mil
L1 Signal 1.378mil
Adhesive 2mil
Prepreg 7.087mil
L2 Plane 1.378mil
Core 8mil
L3 Signal 1.378mil
Prepreg 7.087mil
L4 Signal 1.378mil
Surface Finish 0.512mil
Solder Mask 1mil
Overlay

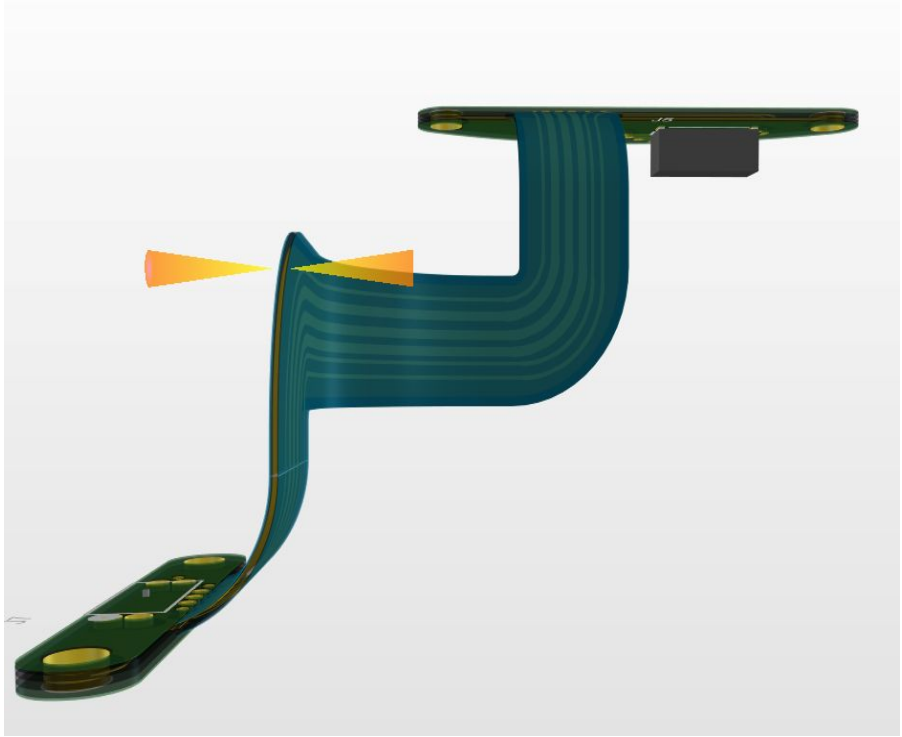
Routing and electrical connections

- Rigid and flex sections are on different layers
- Signals transported between different layers using vias



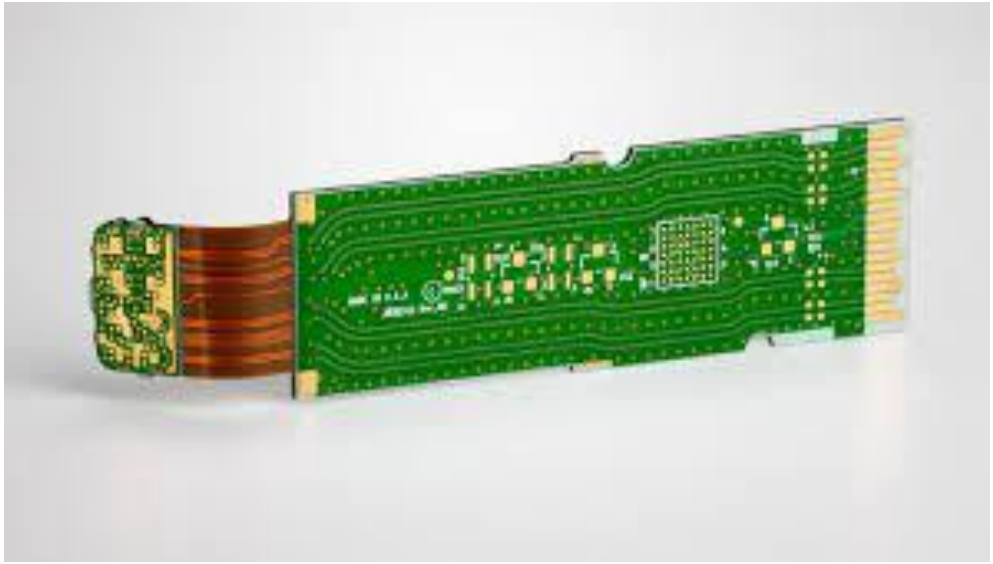
Bend angle/bend radius of flex region

- Bend angles are all 90°, experimented with bend direction
- Bend radius was dependent on bend angle



Flex PCBs

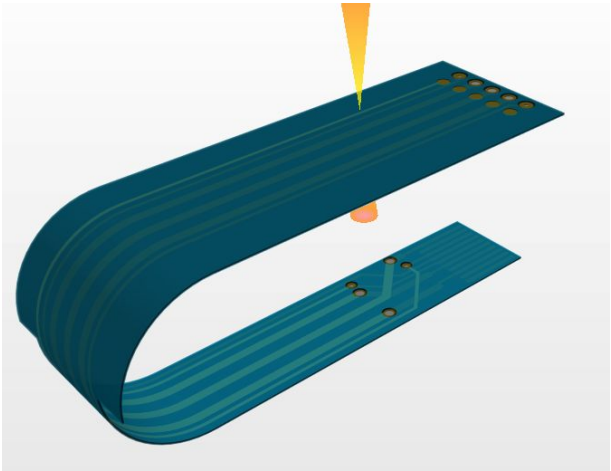
- Used to navigate tight mechanical constraints



[3]

Flex PCB Specs

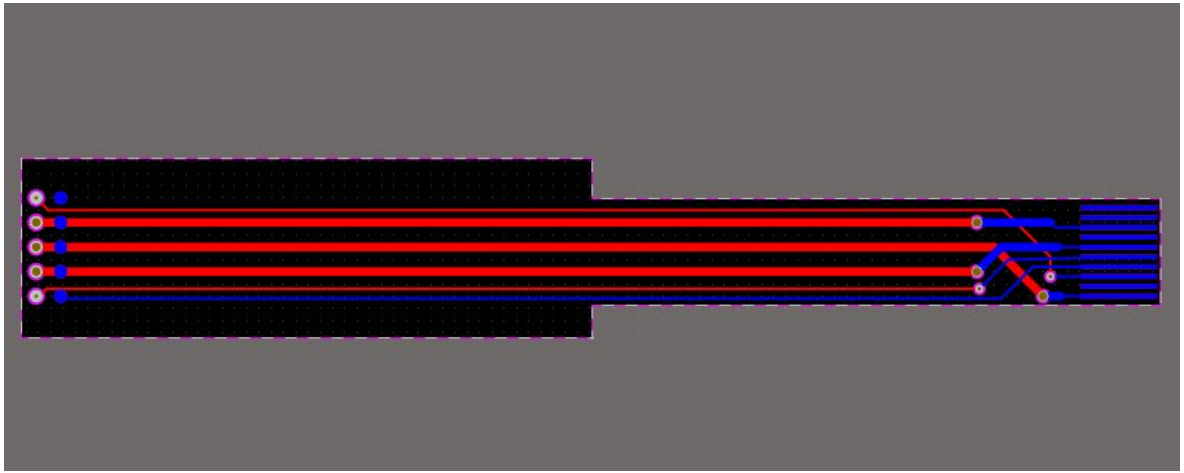
- Flex PCB is used to connect the FPGA connector and the camera body via contact pads and an exposed beveled edge



Flex pcb layer stackup

Routing and wire connections

- Beveled edge and contact pads are on the same layer



Successes and failures: PCB Sub-section

Successes

- Un-bending the CAD model & making it our board shape
- Using vias to ensure optimal signal routing
- Implementing different layer stackups
- Application of rigid-flex and flex PCB knowledge

Failures

- Biggest setback was that we believed that the port mapping of the FPGA connector was the port mapping of the lens connector
- Did not anticipate the process of actually ordering the PCB

Project build: FPGA +PC

FPGA

- XEM7310-A75 (testing purpose)

- XEM7310-A200(in actual design).

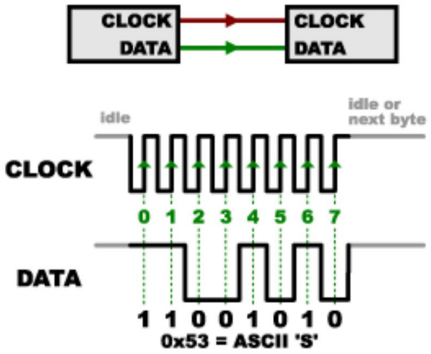
- FPGA is for communication between the computer and the lens.
  - Lens - SPI protocol
  - Computer/PC - OK modules & python codes

PC

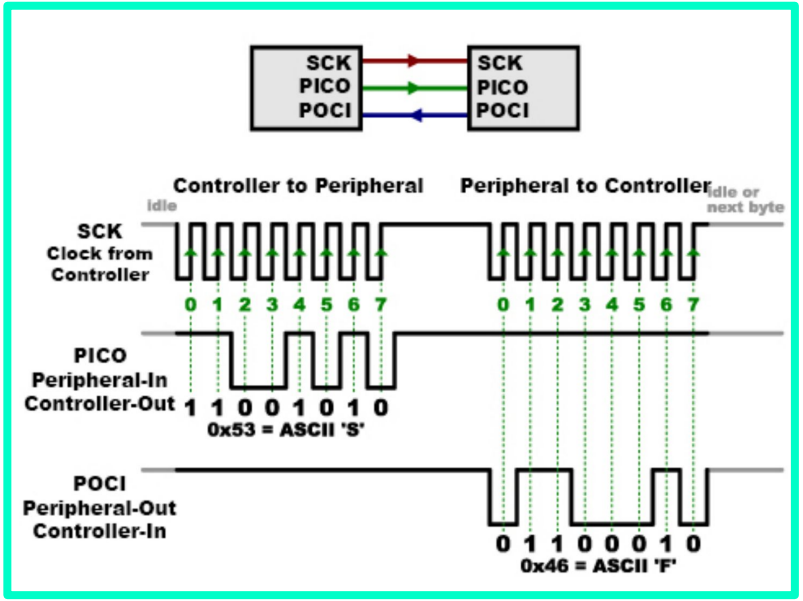
- Users will be sending the commands for the lens to the FPGA using the python code on their PC.

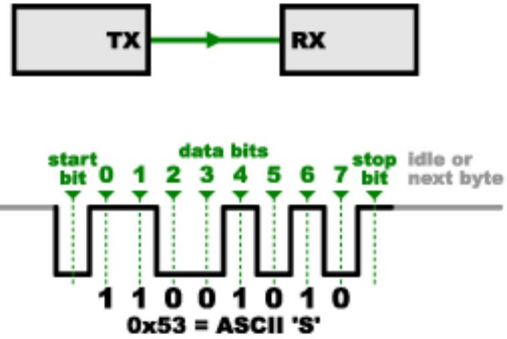
Serial Peripheral interface (SPI) protocol

- It is a common interface used to send data between a microcontroller and small peripheral devices such as sensors.
- It consists of clock, data lines, and select line.



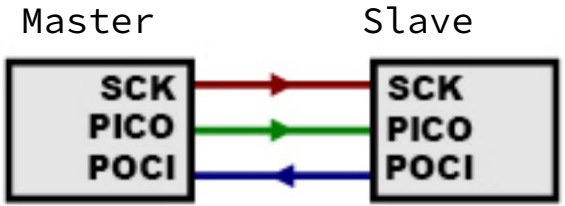
Synchronous

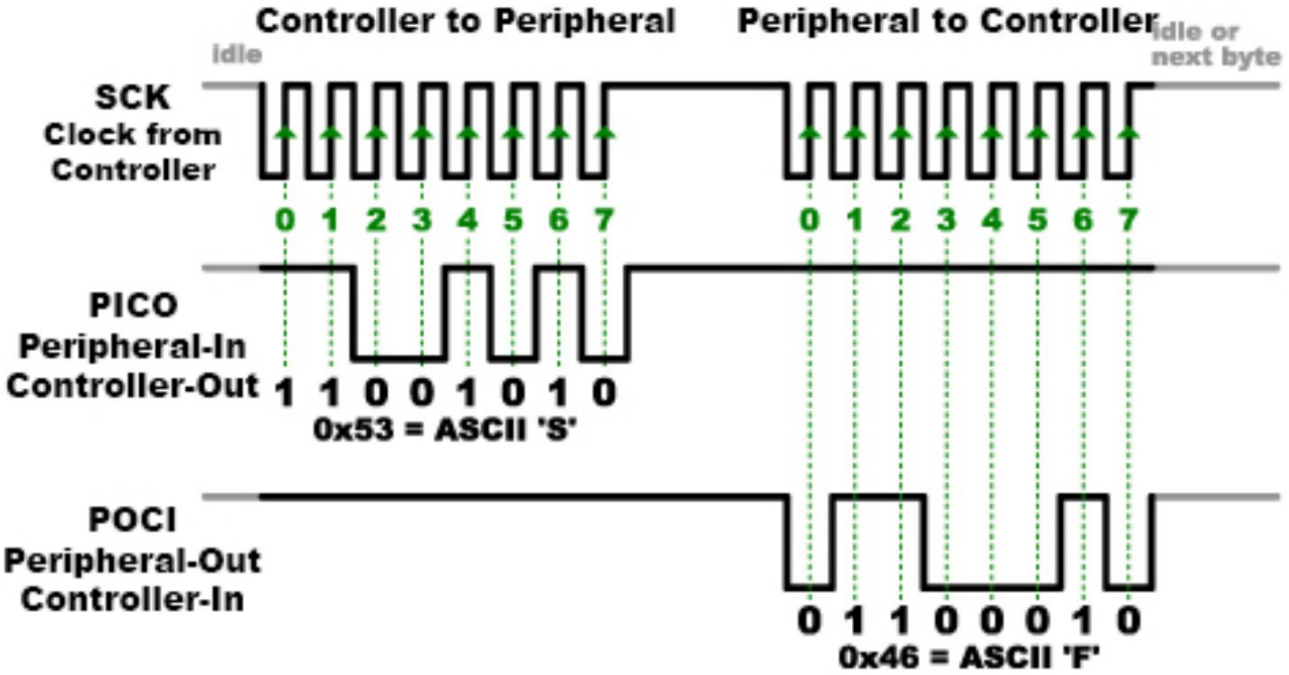




Asynchronous

SPI Protocol





Project build: FPGA +PC

- **Commands**
  - **0x12** : Change aperture
    - + one 8-bit argument
  - **0x44** : Change focus
    - + two 8-bit arguments
  - **0x05** : Focus to max
  - **0x06** : Focus to min
  - **0x0A** : Read/sync

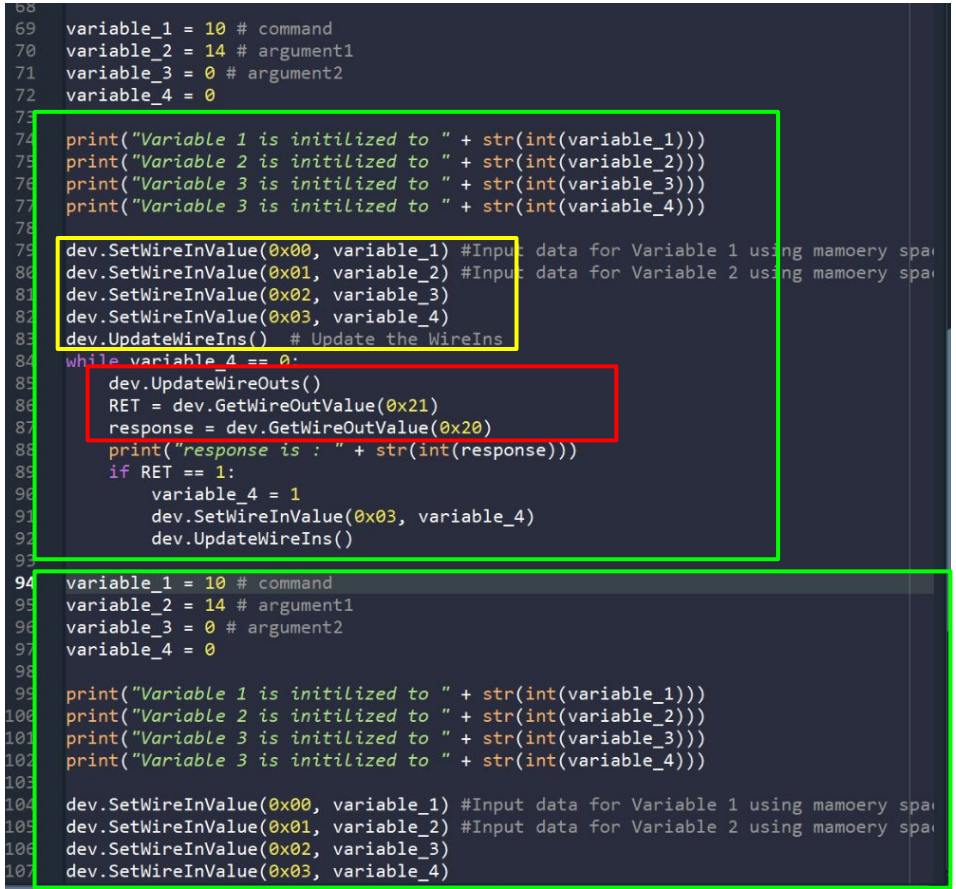
Project build: FPGA +PC

- **Python**
  - Spyder IDE
    - Enables us to make PC communicate FPGA.

- **Verilog**
  - Modules to transfer data between FPGA and PC.
    - OkWireIn
    - OkWireOut

  - State Machine
    - Transfer and receive data in between the FPGA and the lens using SPI protocol

PYTHON



```
variable_1 = 10 # command
variable_2 = 14 # argument1
variable_3 = 0 # argument2
variable_4 = 0

print("Variable 1 is initilized to " + str(int(variable_1)))
print("Variable 2 is initilized to " + str(int(variable_2)))
print("Variable 3 is initilized to " + str(int(variable_3)))
print("Variable 3 is initilized to " + str(int(variable_4)))

dev.SetWireInValue(0x00, variable_1) #Input data for Variable 1 using mamoery spa
dev.SetWireInValue(0x01, variable_2) #Input data for Variable 2 using mamoery spa
dev.SetWireInValue(0x02, variable_3)
dev.SetWireInValue(0x03, variable_4)
dev.UpdateWireIns()  # Update the WireIns
while variable_4 == 0:
    dev.UpdateWireOuts()
    RET = dev.GetWireOutValue(0x21)
    response = dev.GetWireOutValue(0x20)
    print("response is : " + str(int(response)))
    if RET == 1:
        variable_4 = 1
        dev.SetWireInValue(0x03, variable_4)
        dev.UpdateWireIns()

variable_1 = 10 # command
variable_2 = 14 # argument1
variable_3 = 0 # argument2
variable_4 = 0

print("Variable 1 is initilized to " + str(int(variable_1)))
print("Variable 2 is initilized to " + str(int(variable_2)))
print("Variable 3 is initilized to " + str(int(variable_3)))
print("Variable 3 is initilized to " + str(int(variable_4)))

dev.SetWireInValue(0x00, variable_1) #Input data for Variable 1 using mamoery spa
dev.SetWireInValue(0x01, variable_2) #Input data for Variable 2 using mamoery spa
dev.SetWireInValue(0x02, variable_3)
dev.SetWireInValue(0x03, variable_4)
```

Verilog

```
okWireIn wire10 (    .okHE(okHE),
                     .ep_addr(8'h00),
                     .ep_dataout(variable_1));

okWireIn wire11 (    .okHE(okHE),
                     .ep_addr(8'h01),
                     .ep_dataout(variable_2));

okWireIn wire12 (    .okHE(okHE),
                     .ep_addr(8'h02),
                     .ep_dataout(variable_3));

okWireIn wire13 (    .okHE(okHE),
                     .ep_addr(8'h03),
                     .ep_dataout(variable_4));
```

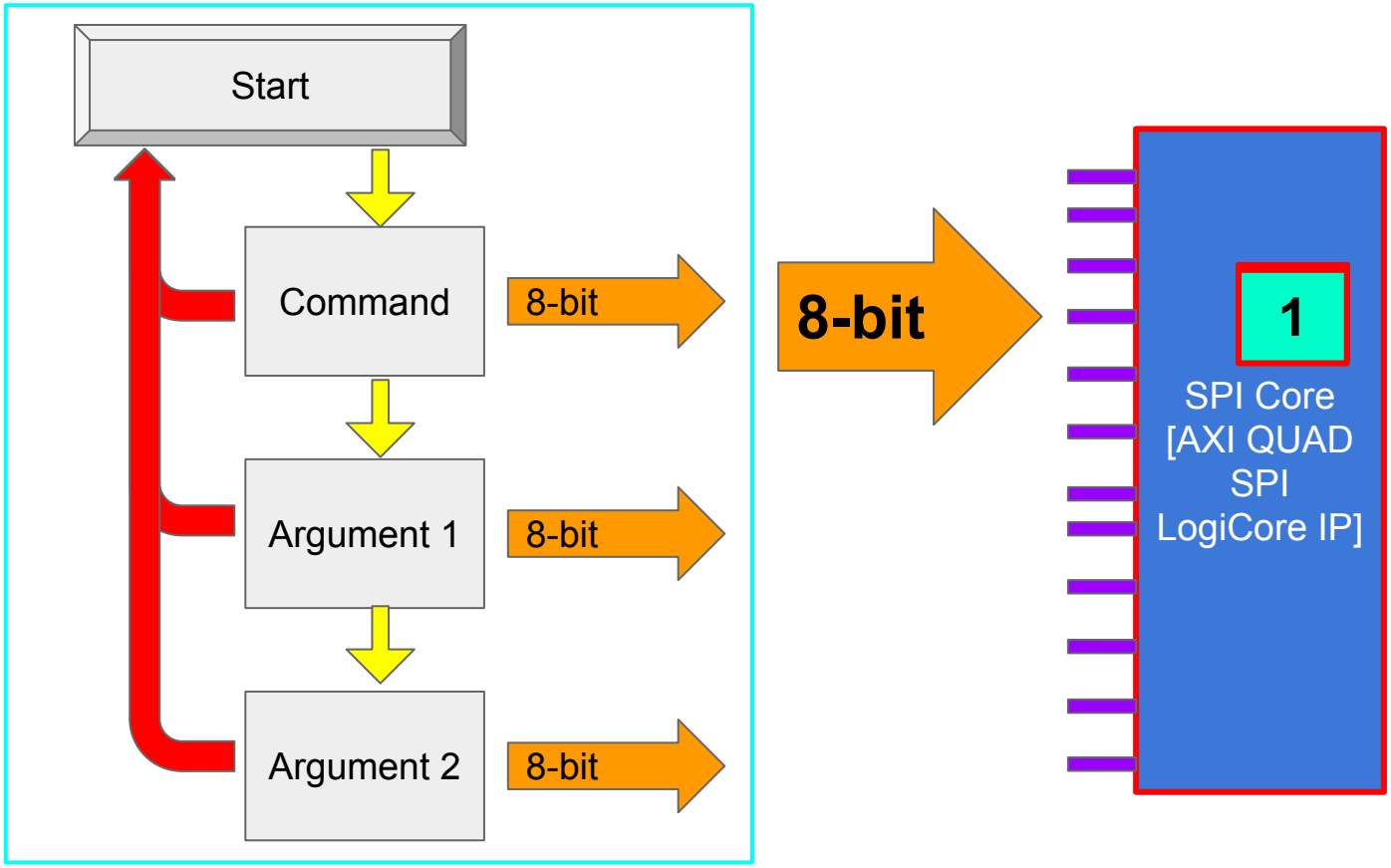
```
okWireOut wire20 (   .okHE(okHE),
                     .okEH(okEHx[ 0*65 +: 65 ]),
                     .ep_addr(8'h20),
                     .ep_datain( {24'b0, response} )
                     );

okWireOut wire21 (   .okHE(okHE),
                     .okEH(okEHx[ 1*65 +: 65 ]),
                     .ep_addr(8'h21),
                     .ep_datain( {31'b0, FIN} ));
```

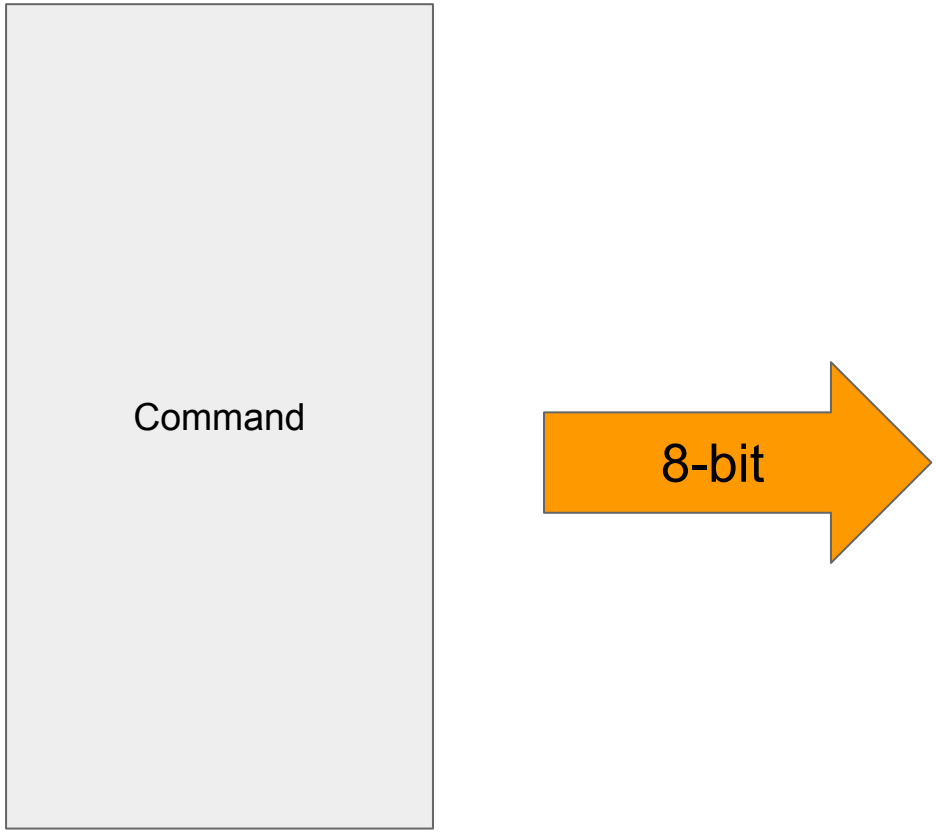
```
dev.SetWireInValue(0x00, variable_1)
dev.SetWireInValue(0x01, variable_2)
dev.SetWireInValue(0x02, variable_3)
dev.SetWireInValue(0x03, variable_4)
```

```
RET = dev.GetWireOutValue(0x21)
response = dev.GetWireOutValue(0x20)
```

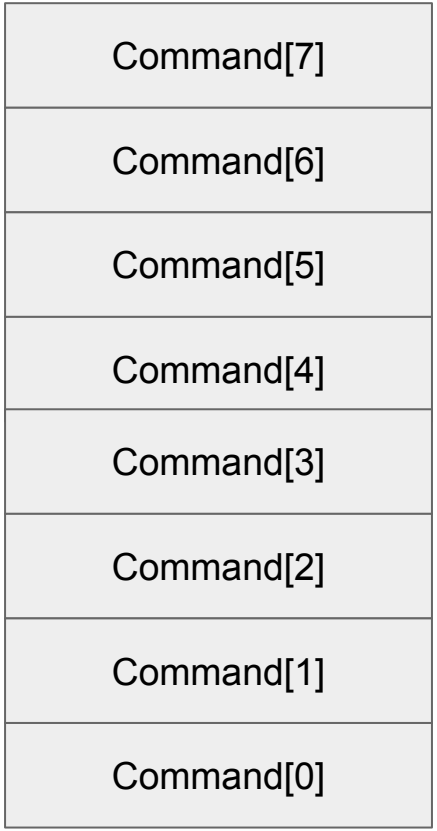
Verilog (State machine) - Initial version



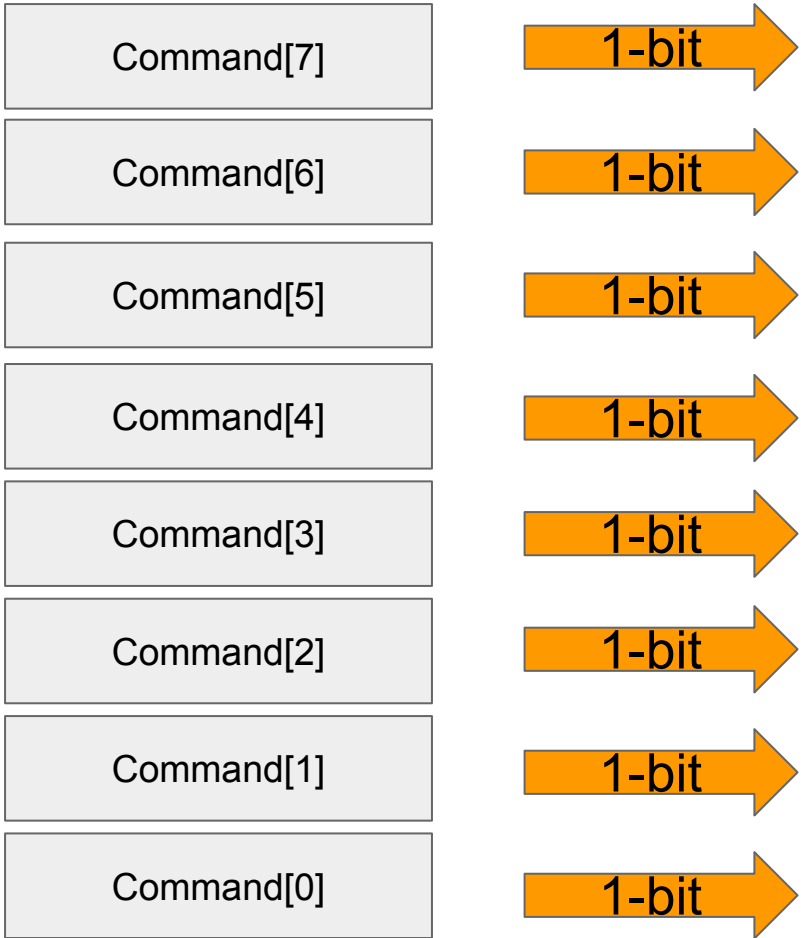
Verilog (State machine) - second version



Verilog (State machine) - second version

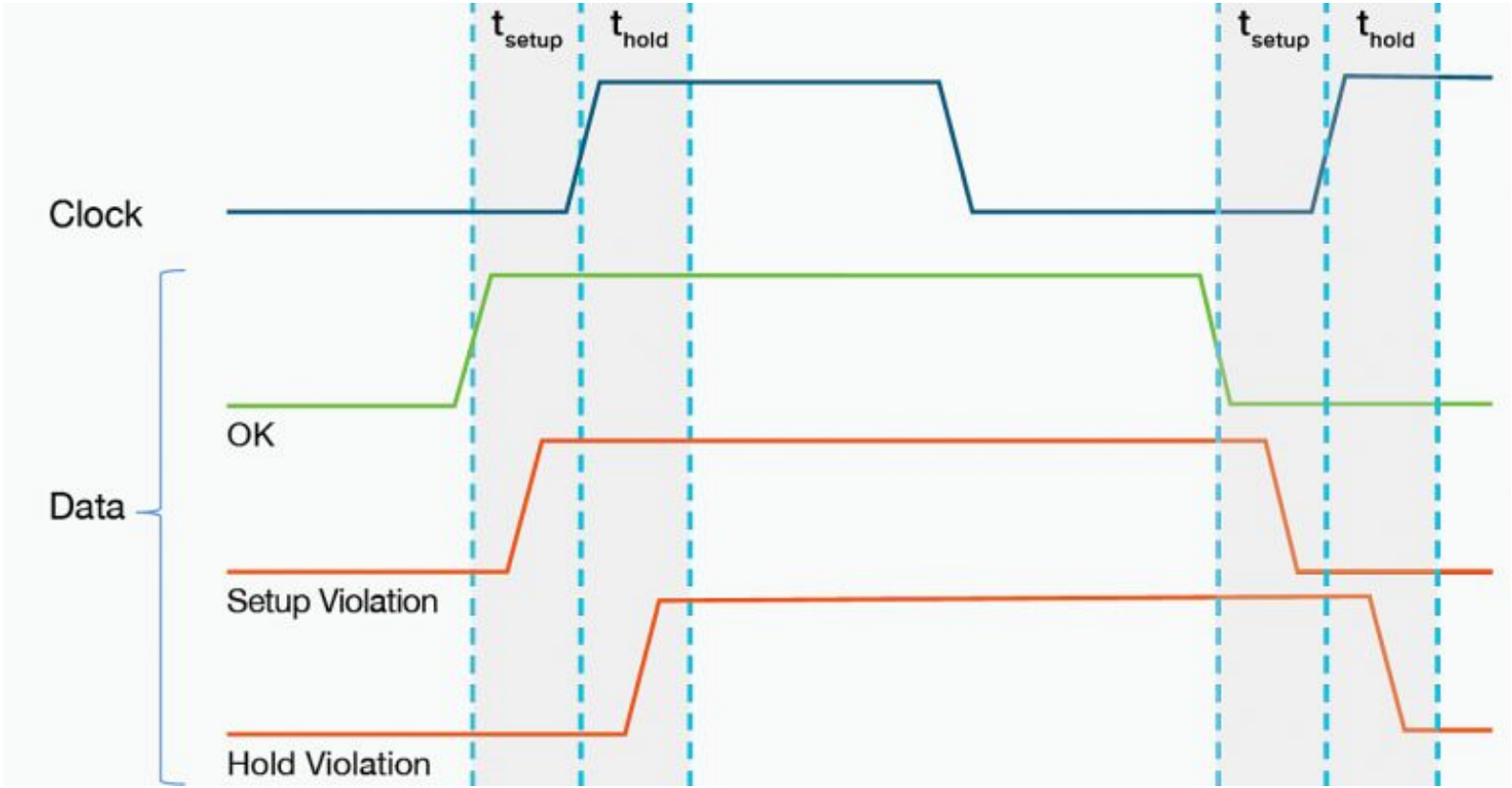


Verilog (State machine) - second version

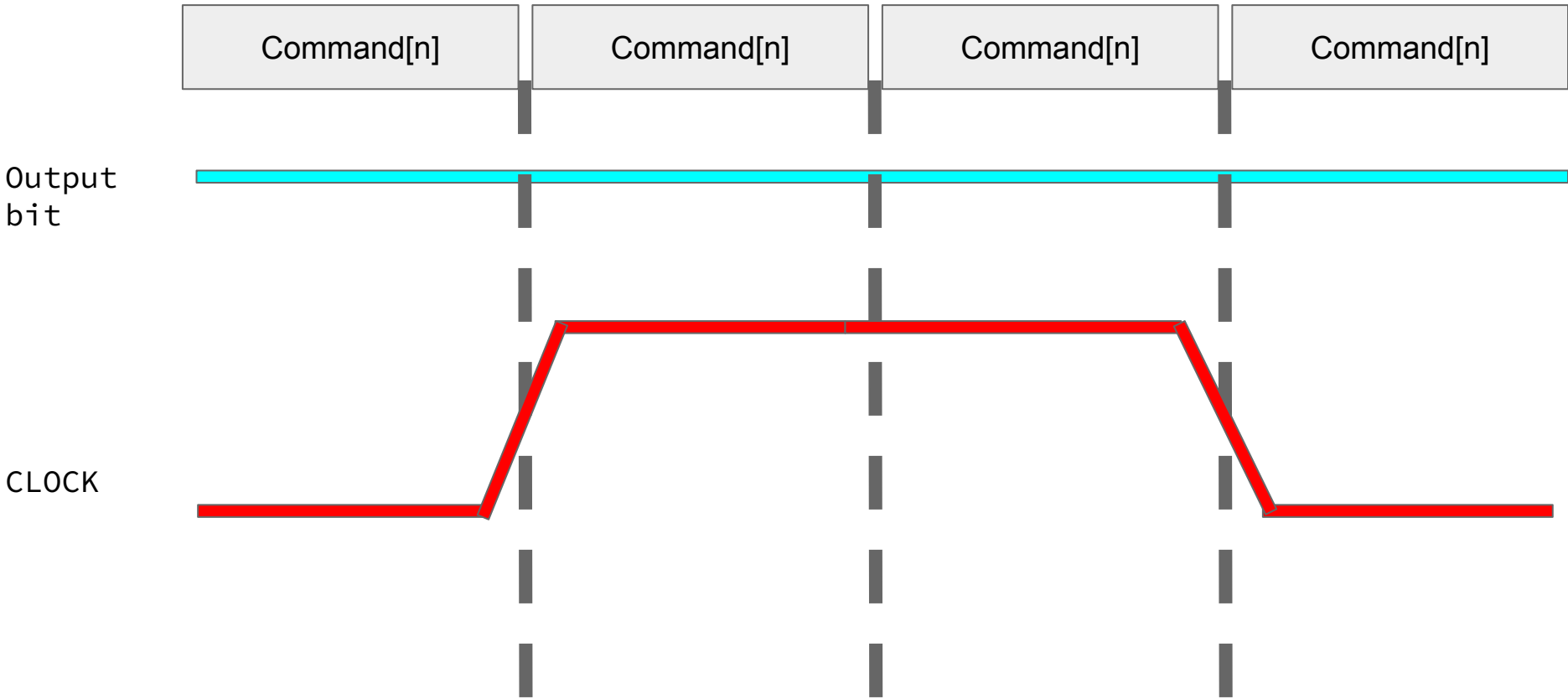


Verilog (State machine) - Third version

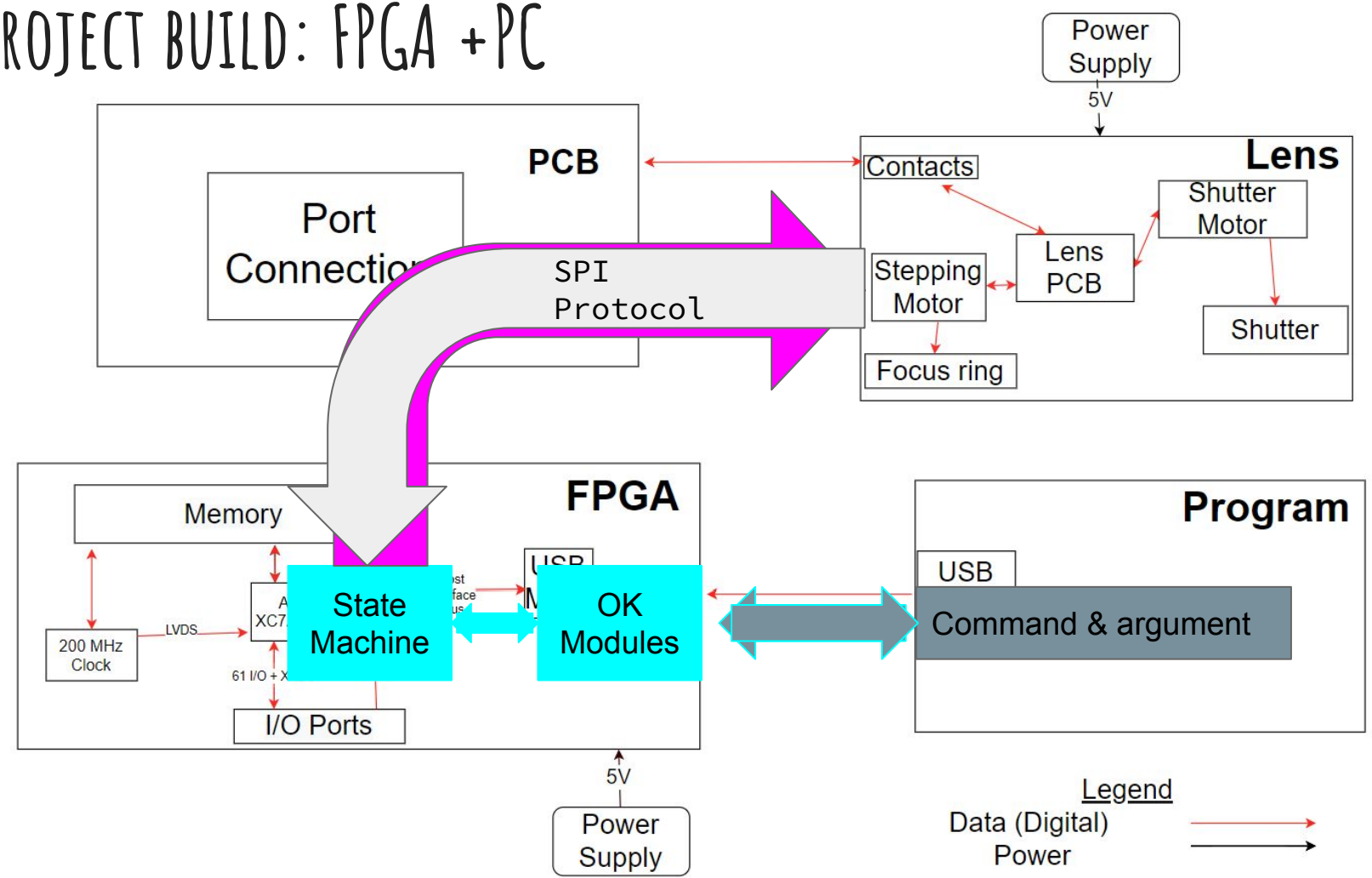
- **Setup and Hold Time Violation**



Verilog (State machine) - Third version

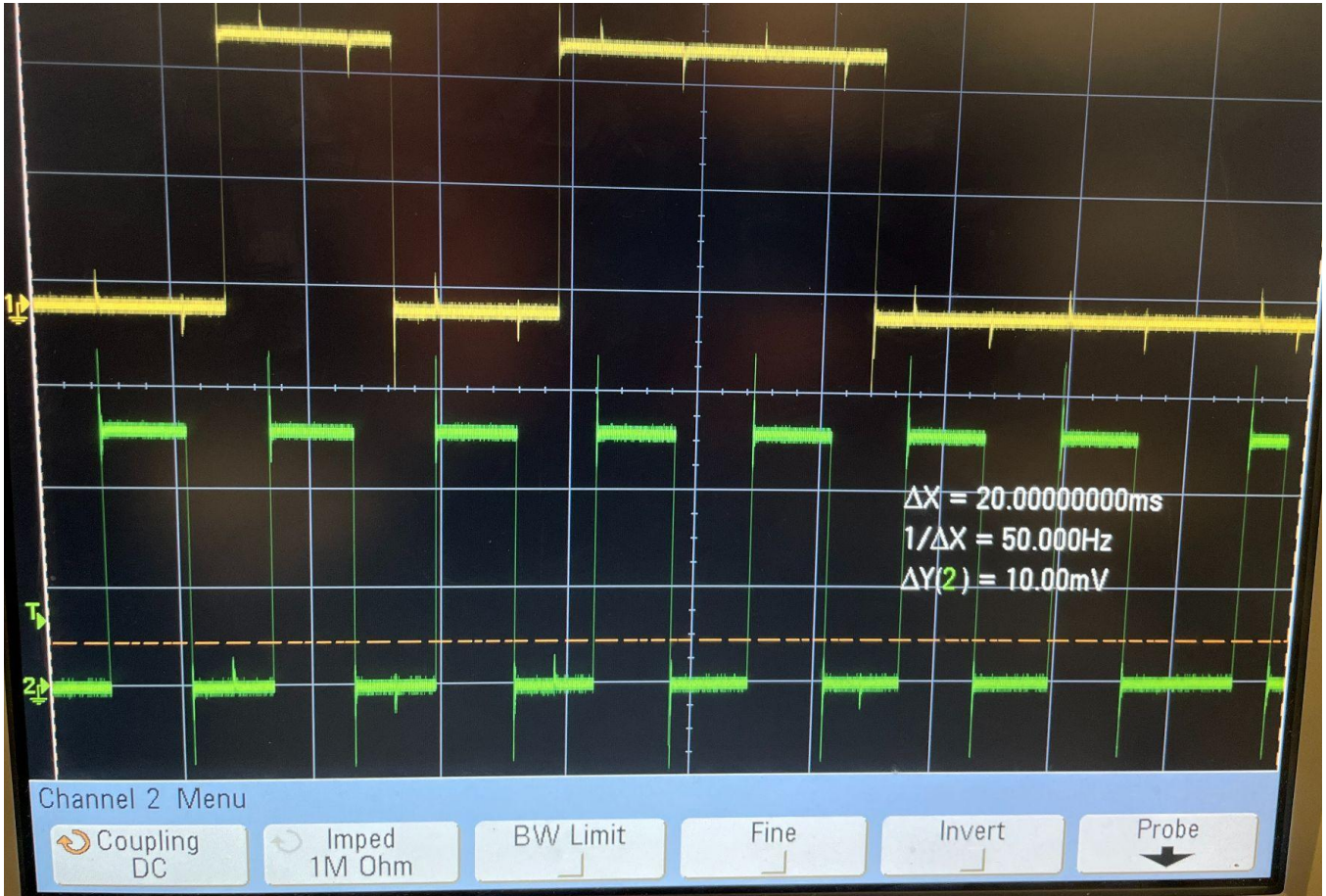


Project build: FPGA +PC



Successes and failures: fpga Sub-section

Successes and failures: fpga Sub-section



Command

Clock

Successes and failures: fpga Sub-section

Clock

Response

Successes and failures: FPGA + PC

Successes

- Managed to get the lens to move according to the command input by the user
- Managed to carry out SPI protocol using the state machine through the verilog code without using standard IP block

Failures

- There are certain values that we need to put in for the argument. We did not have enough time to figure out what are the available argument values.

What did we learn? (Technical)

- SPI Protocol
- General Process of engineering
  - Implementation/Code => Simulation => Testing => Debugging => Success
- OK Modules
- Rigid-Flex/Flex PCBs
  - Layer stackup requirements
  - Routing rules
  - Routing debugging
  - Navigating mechanical design constraints
  - Vias and Tented Vias

What did we learn? (soft skills)

- Ask more questions to prevent misunderstandings later
- Being seen as ‘dumb’ for asking certain questions
- Always have a time cushion for every step
- Dealing with setbacks and how to move forward from them

Conclusion and final thoughts

- Proud of what we have accomplished, disappointed we couldn’t make it a reality
- Learned hard lessons to carry with us into the future
- Enjoyed putting our skills to the test and picked up necessary knowledge to achieve our goals
- Picked the right team members to do this project with

Ethical concerns

- Privacy concerns using the camera
- Mechanical hazards in the event of an accident

Special Thanks to

- **Illinois BioSensors Lab**
  - Professor Viktor Gruev
  - Zhongmin Zhu
- **Our TA:** Zhicong Fan

Citations

[1] “Home,” *Flex PCBs | Rigid Flex PCBs | PCB Unlimited*. [Online]. Available: https://www.pcbunlimited.com/products/rigid-flex-pcbs. [Accessed: 24-Nov-2022].

[2] “What is Flex PCB? - an overview of Flex and Rigid-Flex PCB - news,” *PCBway*. [Online]. Available: https://www.pcbway.com/blog/News/What_is_Flex_PCB____An_Overview_of_Flex_and_Rigid_Flex_PCB.html. [Accessed: 24-Nov-2022].

[3]“Basics of Flex Circuit Design - Minco Products.” [Online]. Available:https://www.minco.com/wp-content/uploads/Minco_BasicsofFlexDesign.pdf [Accessed: 29-Sep-2022].

[4] M. Grusin, “Serial Peripheral Interface (SPI) - learn.sparkfun.com,” *Sparkfun.com*, 2019. https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[5] “Final thoughts images – browse 2,487 stock photos, vectors, and video,” Adobe Stock. [Online]. Available: https://stock.adobe.com/search?k=final+thoughts. [Accessed: 27-Nov-2022].

[6] “Ethics issues in the engineering profession,” Railway Age, 12-Sep-2022. [Online]. Available: https://www.railwayage.com/regulatory/ethics-issues-in-the-engineering-profession/. [Accessed: 27-Nov-2022].