ELECTRICAL & COMPUTER ENGINEERING

Lab Journal / Silas Curfman
AOP ECE-522 / Fall 2022
Hardware / Software Co-Design

Journal Outline

I. Overall Course Objectives
II. Primary Resources
III. Extra Resources
IV. Module by Module Progress
V. Archived notes & lab work
VI. Closing notes, personal review

Overall Course Objectives

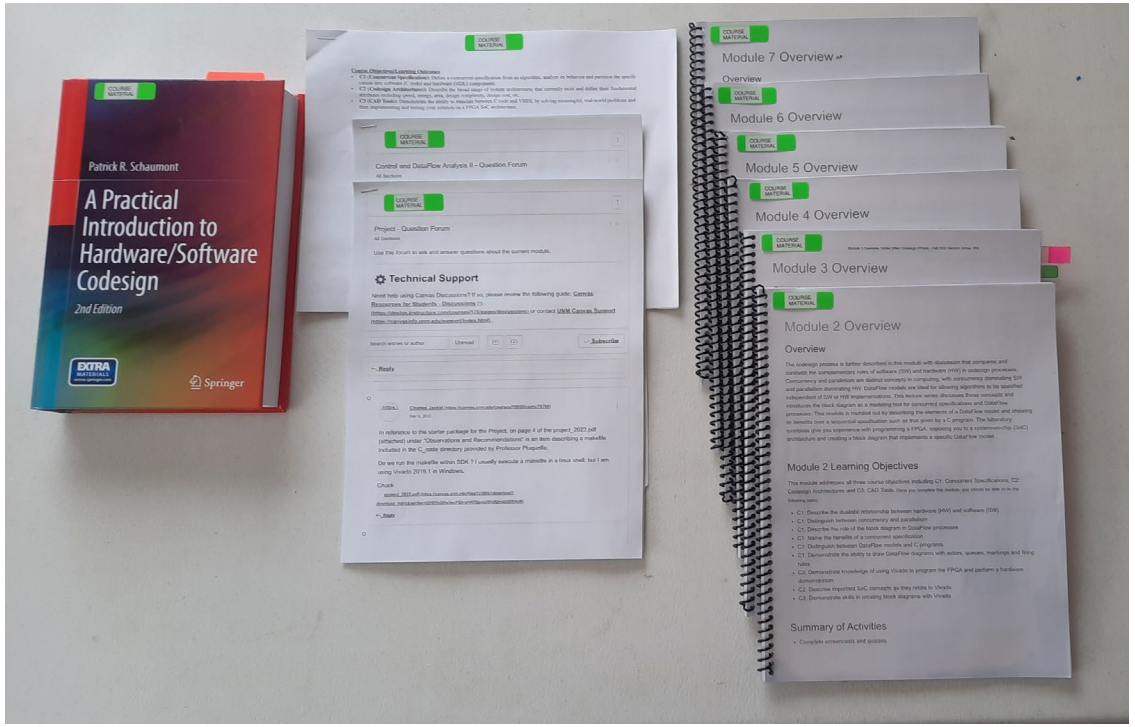
journal files\syllabus.pdf
journal files\CodesignCourseMap.pdf
journal files\schedule.pdf

From Course Map:

Course Objectives/Learning Outcomes

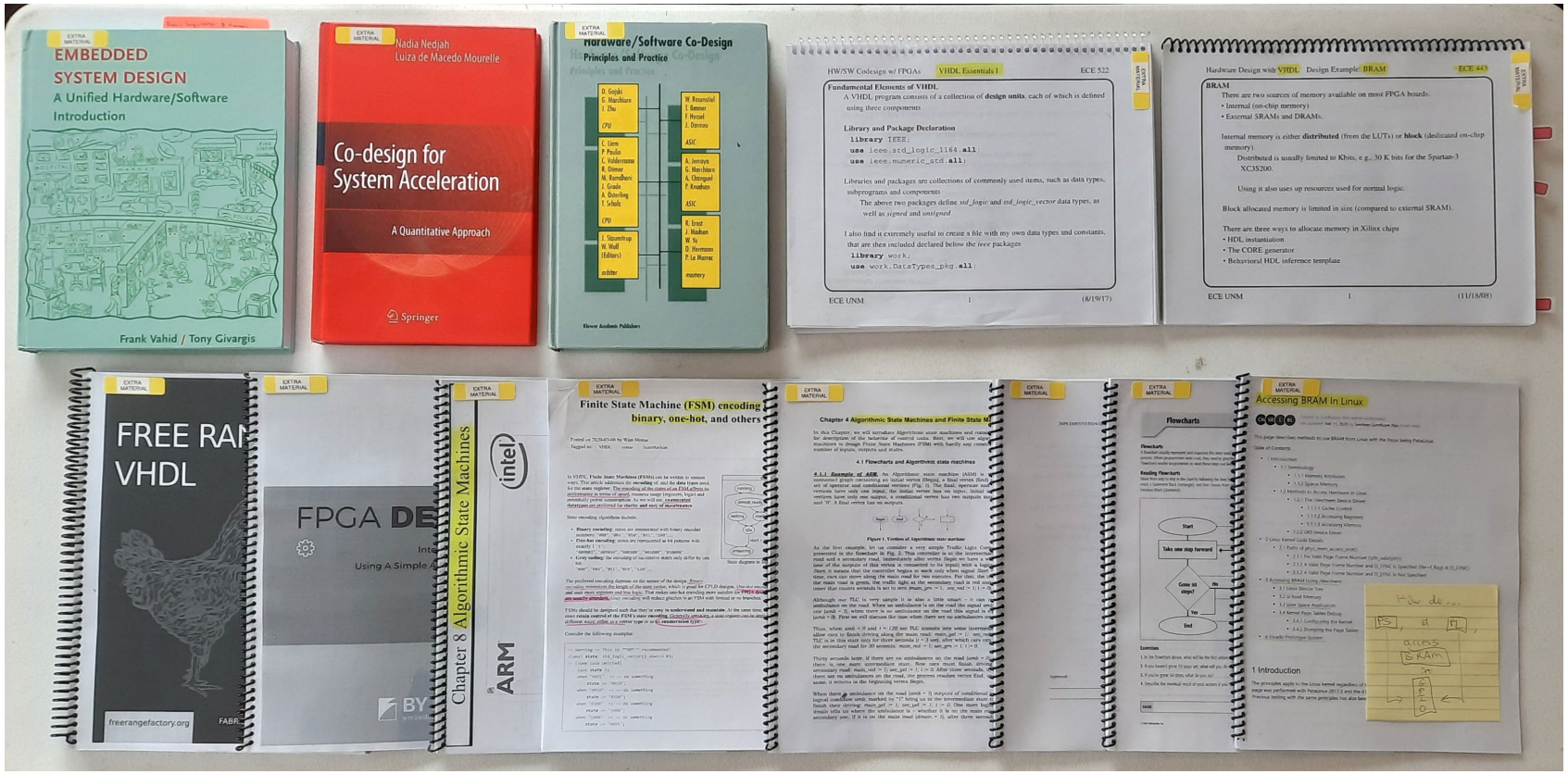
- **C1 (Concurrent Specification):** Define a concurrent specification from an algorithm, analyze its behavior and partition the specification into software (C code) and hardware (HDL) components.
- **C2 (Codesign Architectures):** Describe the broad range of system architectures that currently exist and define their fundamental
attributes including speed, energy, area, design complexity, design cost, etc.
- **C3 (CAD Tools):** Demonstrate the ability to translate between C code and VHDL by solving meaningful, real-world problems and
then implementing and testing your solution on a FPGA SoC architecture.

Course Materials (Primary Resources)



- Instructor led lectures / videos
- Instructor supplied slides / pdfs
- Textbook : A practical Introduction to Hardware / Software Codesign (Schaumont)
- Online student discussion forum

Extra Materials (Secondary Resources)



See list, next page

Extra Materials (Secondary Resources)

ELECTRICAL & COMPUTER ENGINEERING

OTHER TEXTS:

- Hardware / Software Codesign Principles and Practice (Staunstrop)
- Embedded System Design (Vahid)
- Co-design for System Acceleration (Nedjah)
- Free Range VHDL (Mealy)
- FPGA Design / Interfacing Over AXI Using a Simple Address Data Bus (Bodlovic)

OTHER ARTICLES:

- journal files\algorithmic state machines.pdf
- journal files\amd algorithmic state machines.pdf
- journal files\Flow_Charts.pdf
- journal files\flowchartandalgorithmbasics.pdf
- journal files\fpga algorithmic state machines.pdf
- journal files\vhdl cookbook.pdf

OTHER, INSTRUCTOR SUPPLIED:

- VHDL Essentials I-V, Videos + PDFs
  http://ece-research.unm.edu/jimp/codesign/index.html

OTHER VIDEOS / CHANNELS:

- https://www.youtube.com/watch?v=fqUuvwl4QJA&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=17
- LBEBooks YT Channel (videos 90-96)
  https://www.youtube.com/watch?v=DMSaYhD1GkM&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=16&ab_channel=LBEbooks
- Nandland YT Channel / Block RAM in FPGA
  https://www.youtube.com/watch?v=fqUuvwl4QJA&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=17&ab_channel=nandland
- MrProfScott YT Channel / Algorithmic State Machine Do's and Don'ts
  https://www.youtube.com/watch?v=6aCw7JtHmzg&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=11&ab_channel=MrProfScott
- Jon Morss / Zybo Z7-10 Petalinux PWM Demo
  https://www.youtube.com/watch?v=UhQ4-XlooiQ&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=9
- Intel FPGA / VHDL Basics
  https://www.youtube.com/watch?v=zm-RA6BsYmc&list=PLMHcivtPq9KB7B9LAgu4TuU7HOyAb-XJ6&index=2&ab_channel=IntelFPGA

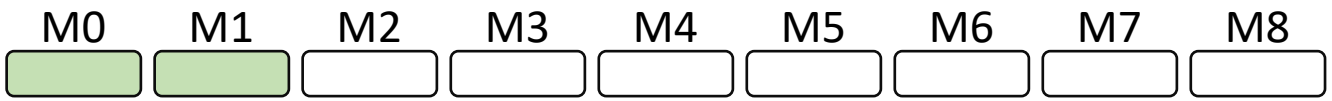
ELECTRICAL & COMPUTER ENGINEERING

Module 0: Course Prep

Module Objectives: pre purchase materials, set up infrastructure / home-lab, configure host computer.
Module Deliverables: NA

Entry	Notes
M0.1	Purchased books and zybo board in time for delivery before course.
M0.2	Preconfigured home-lab server for 2-3 anticipated VM's, using VMWARE / ESXI.
M0.3	Completed some self study / introduction to FPGA'sand VHDL.

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
DO use VM's	
DO order early any and all supplies	



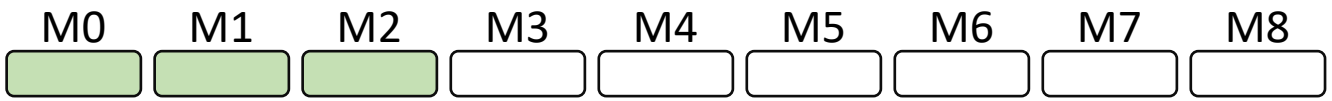
ELECTRICAL & COMPUTER ENGINEERING

Module 1: Introduction to codesign with FPGA's

Module Objectives: Introductory, additional reference texts
Module Deliverables: Quiz

Entry	Notes
M1.0	Purchased all 3 reference texts.
M1.1	Reviewed instructor provided web links and began VHDL Essentials

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
DO make use of instructor's supplied web links to VHDL Essentials! Was very helpful.	



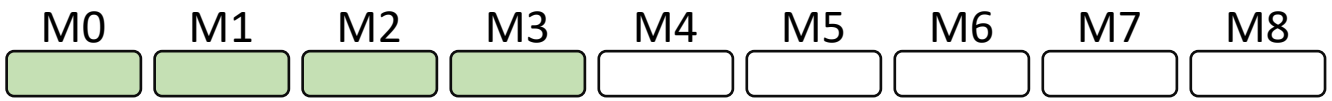
ELECTRICAL & COMPUTER ENGINEERING

Module 2: Introduction & Dataflow

Module Objectives: dualism HW/SW, concurrency & parallelism, diagrams, IDE usage
Module Deliverables: Quiz, Screenshot

Entry	Notes
M2.0	Intro to diagraming flow, tokens, markings, firings, queues, and states.
M2.1	Discuss PS / PL differences, barriers, ways to overcome & interact with each other
M2.2	Intro to Zynq architecture. Keywords: PS, PL, GPIO, AXI, BRAM, design wrapper

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
Having set aside server space for multiple VM's was good but did not plan on each VM needing so much HDD space. Found setting up VM's with 175 – 200 gb virtual disks worked well.	Host OS + IDE problems. Rebuilt Dev VM 2-3 times. Each new IDE build requires new download from Xilinx. Xilinx hosted downloads are VERY VERY SLOW. Allow for extra time.
At this point continuing to work through instructor's material "VHDL Essentials" along side Module 2.	



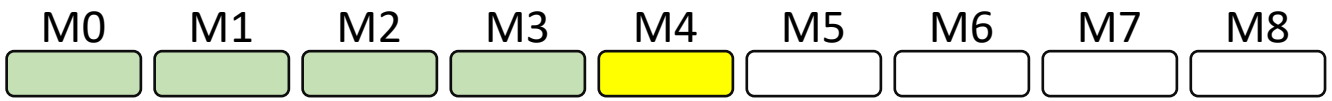
ELECTRICAL & COMPUTER ENGINEERING

Module 3: DataFlow Models

Module Objectives: Derive SDF, graph PASS, discuss SDF, evaluate DGF's, design RTL FSM.
Module Deliverables: screenshot of functioning GPIO_BRAM in Vivado

Entry	Notes
M3.0	Getting into some hardware challenges on Zybo board. Stable IDE, host OS, Bootable linux SD card, & serial vs ethernet comms becoming extremely important to completing modules.
M3.1	SDF lectures. Lots of material to cover.
M3.2	Lectures introduce LM_ULM, this becomes major component in understanding next modules
M3.3	FSM of LM_ULM demonstrates method to conceptualize LM_ULM algorithm, VERY KEY
M3.4	Introduce other necessary VHD elements; Top.VHD, DataTypes_pkg.VHD, & main program LoadUnloadMem.VHD

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
Completed review of VHDL Essentials	Incorrect understanding of how GPIO, BRAM & LM_ULM work together to bridge gap between PS & PL. This misunderstanding caused major problems in next modules
Class discussion forum VERY HELPFUL as we all diagnosed different hardware issues at somepoint with Zybo boards.	Thought BRAM was part of or somehow contained inside GPIO. This became a problem
	Should have stopped longer to understand BRAM write and read executions. Misunderstanding write and read calls via BRAM became biggest problem later.



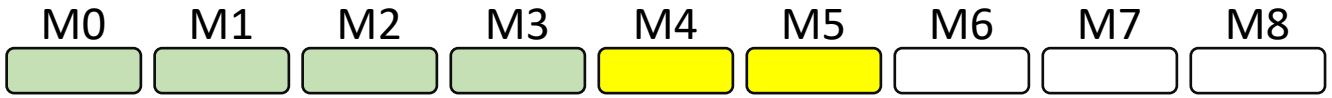
ELECTRICAL & COMPUTER ENGINEERING

Module 4: DataFlow Implementations

Module Objectives: translate DFG to software, write C from DFG, discuss tradeoffs, DFG pipelining
Module Deliverables: Vitis / SDK implementation, screenshot of C application, proof runningon Zybo hardware.

Entry	Notes
M4.0	Getting much, much, deeper in diagramming data flow.
M4.1	Data flow schedules, requires careful reading.
M4.2	Getting into single vs multi threading, FIFO's, dynamic schedules
M4.3	Good place to note, C can do WHILE loops, VHDL cannot. Need to be actively thinking about how applying SPEC to SW will be a different approach than applying it to HW.
M4.4	Very important to develop abstraction skills, need to think in terms of State Machines that are neither natively SW or HW but can be developed into either.
M4.5	Clock cycles, timings, and knowing what does and doesn't happen on either rising or falling edge of signal is getting EXTREMELY IMPORTANT.
M4.6	WARNING: Beginning to get behind submitting labs. Lectures and quizzes OK though at this point.

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
Began taking WORKFLOW notes, specifically what steps were needed for my particular environment.	Getting a little lost in the IDE. Some confusion about what makes Vitis different than SDK (nothing really)
Increased VM server storage, just in time. Had to revert the VM to a previous snapshot when dealing with a VITIS install that went bad. Having the extra storage allowed me to keep spin up a 2nd backup VM to do a couple quick tests to confirm VITIS needed to be reinstalled. Otherwise would have lost much more time.	Had a problem with my SD card on the zybo. Was able to log into zynq via serial connection so I thought things were fine but nothing would TXFR over. Took a few days to realize SD card was not working correctly. Started over with SD card and eventually solved problem. Slow to fix because since I could log in via serial I thought SD card must be good. My mistake.



ELECTRICAL & COMPUTER ENGINEERING

Module 5: FSMD

Module Objectives: Create FSM + Datapath from C, implement in hardware via VHDL, create ASM-D
Module Deliverables: Midterm, complete the remaining ASMD for HISTO.

Entry	Notes
M5.0	Progressing from a DISCRETE understanding of a single concept or tool to the ability to INTEGRATE it with others. This is not a small distinction.
M5.1	Must be able CONCEPTUALIZE (where you want to go) desired end product and create an ABSTRACTION (how you're going to get there) of the necessary build. MUST DO BOTH OF THE ABOVE EQUALLY WELL IN PS AND PL DOMAINS. Having a solid understanding of what makes the PS and PL sides different is VERY KEY.
M5.2	INTERPRETATION and ENCAPSULATION of how the elements of your design, regardless of SW or HW, interact with each other is NECESSARY.
M5.3	Self assessment at this point is difficult. You don't know what you don't know until you make that discovery. This module tests HOW YOU THINK more than it tests your TECHNICAL knowledge.
M5.4	HOW YOU THINK plays in to , can you communicate this to OTHERS (diagrams, FSM's, etc). Can you ARTICULATE what you are trying to do and why?

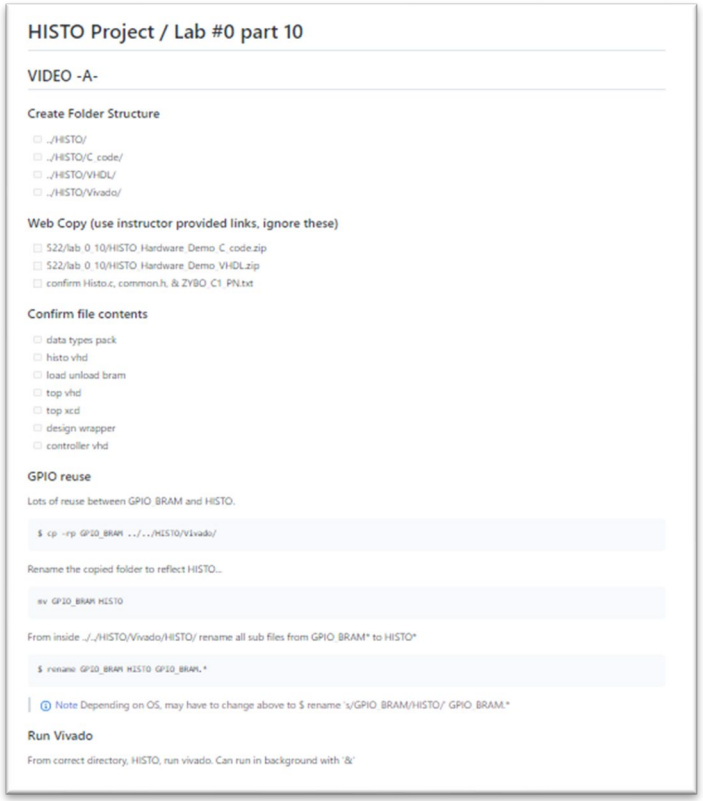
FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
Embedded System Design (Vahid) Chapter 2 examples of FSM of GCD was a huge help. Walked through all chap 2 examples carefully.	DANGER: Lost lots of time completing ASM-D for HISTO algorithm. Difficulty was making jump from just a CONCEPT to actual PRODUCTION.
!!! Video Lessons 93 – 98 by LBE Books YT Channel. Were HUGE HELP. Walked through creating VHDL model from textual SPECIFICATION for a proposed Greatest Common Divisor model and a proposed Integer Square Root model. WORTH GOING THROUGH CAREFULLY. Big GAIN for UNDERSTANDING but also LOST TIME this module overall.	Not a wrong turn, but SLOWED DOWN by stopping to comb over how HISTO.VHD interacted with the other .VHD sources. Not strictly necessary but felt I had to do it. Moving forward without solid grasp of dependcies would have been a guessing game. GRANULAR learning style slowing down progress but don't know any other way when trying to ASSIMILATE knew knowldege
Created a markdown document detailing steps needed to repeat XILINX WORKFLOW	

Module 5: FSMD

Module Objectives: Create FSM + Datapath from C, implement in hardware via VHDL, create ASM-D
Module Deliverables: Midterm, complete the remaining ASMD for HISTO.

Link to XILINX workflow hosted on git hub…

https://github.com/0xbadc0fee/UNM-Setups/blob/7ead4290575b7b94786053301b26beed72a6bd73/522/lab_0_10/@lab_0_10.md



HISTO Project / Lab #0 part 10

VIDEO -A-

Create Folder Structure

- ../HISTO/
- ../HISTO/C_code/
- ../HISTO/VHDL/
- ../HISTO/Vivado/

Web Copy (use instructor provided links, ignore these)

- 522/lab_0_10/HISTO_Hardware_Demo_C_code.zip
- 522/lab_0_10/HISTO_Hardware_Demo_VHDL.zip
- confirm Histo.c, common.h, & ZYBO_C1_PN.txt

Confirm file contents

- data types pack
- histo vhd
- load unload bram
- top vhd
- top xcd
- design wrapper
- controller vhd

GPIO reuse

Lots of reuse between GPIO_BRAM and HISTO.

```
$ cp -rp GPIO_BRAM ../../HISTO/Vivado/
```

Rename the copied folder to reflect HISTO...

```
mv GPIO_BRAM HISTO
```

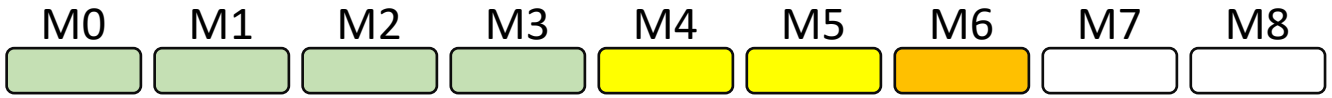
From inside ../../HISTO/Vivado/HISTO/ rename all sub files from GPIO_BRAM* to HISTO*

```
$ rename GPIO_BRAM HISTO GPIO_BRAM.*
```

Note Depending on OS, may have to change above to $ rename 's/GPIO_BRAM/HISTO/' GPIO_BRAM.*

Run Vivado

From correct directory, HISTO, run vivado. Can run in background with '&'



ELECTRICAL & COMPUTER ENGINEERING

Module 6: Control and Data Flow Analysis I

Module Objectives: Perform data flow analysis on a c program, analyze CFG's & DFG's, map C constructs to CFG / DFG
Module Deliverables: Lab#1, quizzes

Entry	Notes
M6.1	Lecture notes and videos VERY VERY IMPORTANT. Go over very carefully. KNOW the DFG construction process inside and out.
M6.2	Spotting, deconstructing, and rebuilding CONDITIONAL logic is CRITITCAL. Must be able to TRANSLATE logic out of PS domain and into PL domain without breaking the model.
M6.3	This is the module I hit the wall in. I was able to correct some misconceptions that got overlooked since Module 3 but not before running out of time in the course.

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)
Stopping to literally sketch out the architecture helped me understand where I went wrong. Problem arose probably back in M3 but I didn't realize it till this module. <u>LIGHTBULB MOMENTS:</u> • BRAM is not 1:1 connected to GPIO • GPIO 0 & GPIO 1 roles as RX / TX for PS side. • LM_ULM is for PS to PL comms not utilized for HISTO <-> BRAM txfr!!! • BRAM reads & writes are looking at data ONE CLOCK CYCLE APART • MUX SEL to feed _ADDR to BRAM must be made at least ONE CLOCK CYCLE before the WRITE is called • A call to WRITE without specifying a "_DIN" can be used to CLEAR memory because default is set to all ZEROES!!!	DANGER: A BAD understanding of BRAM derailed me completely here. Misunderstood WHERE BRAM is, READ execution, WRITE execution, how it's fed "_DIN" and "_ADDR", why "_WE" doesn't have to be reset to 0 inside the SM when reading, and how the MUX SEL controls the _ADDR being fed to be used as EITHER a read target or a write target. HAD TO GO BACK TO BEGINNING AND REBUILD UNDERSTANDING OF BRAM. 1st attempt at lab 1 took wrong turn. Tried to shim new states in wrong part of stack, made too many unnecessary changes. Misunderstood SENSITIVITY list and did not account for changes necessary.

ELECTRICAL & COMPUTER ENGINEERING

Module 6: Control and Data Flow Analysis I

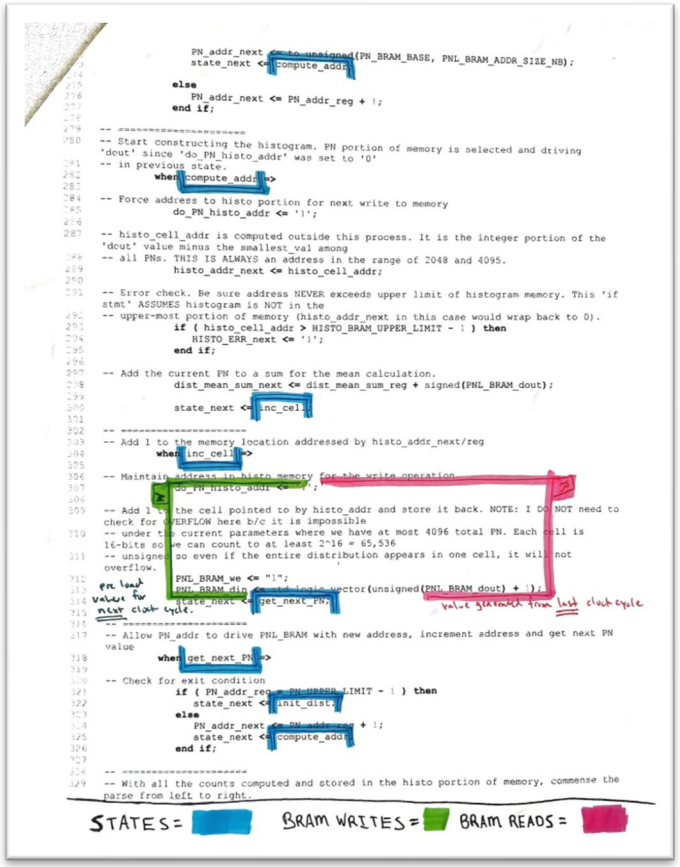
Links to notes / drawings:

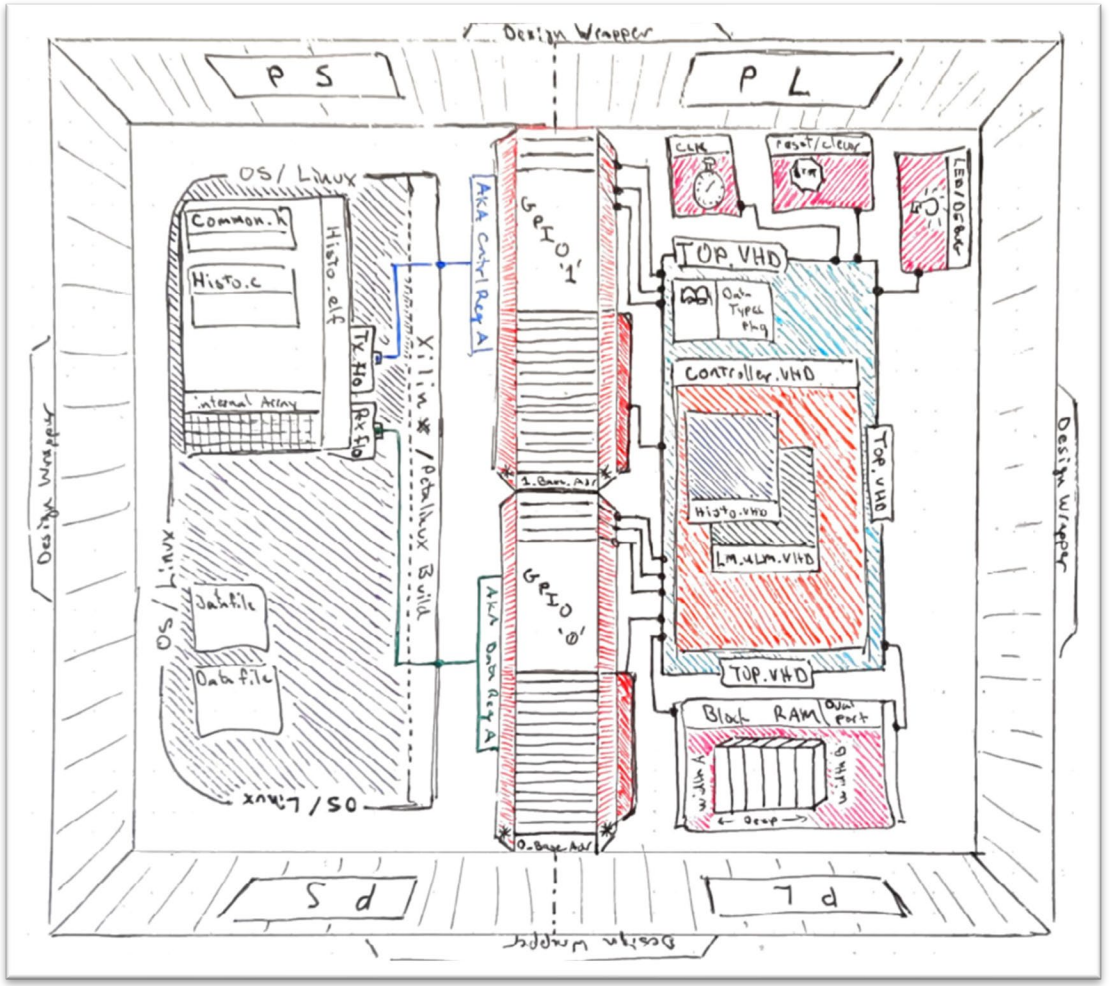
journal files\lab0_architecture_graphic.pdf

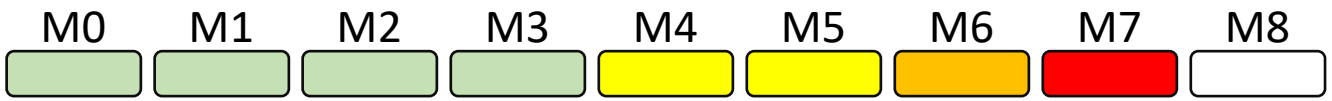
journal files\ece522HISTO_rgb.pdf

journal files\lab1_attempt1_todo.pdf

journal files\parsing states.pdf







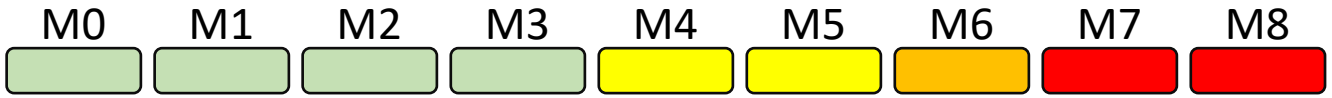
ELECTRICAL & COMPUTER ENGINEERING

Module 7: Control and Data Flow Analysis II

Module Objectives: Translate C program to FSMD, contstruct hardware from FSMD, Optimize, Xilinx HLS,
Module Deliverables: Quizzes, Lab#2

Entry	Notes
M7.1	DID NOT FINISH // SEE NOTES IN M6

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)



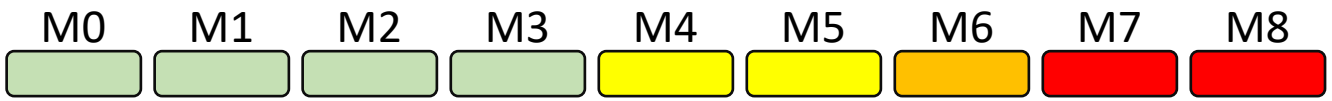
ELECTRICAL & COMPUTER ENGINEERING

Module 8: Final Project

Module Objectives: HW/SW implementation of a K-means clustering algorithm on an FPGA
Module Deliverables:

Entry	Notes
M8.1	DID NOT FINISH // SEE M6 NOTES

FORWARD LEAPS (HELPS)	WRONG TURNS (HURTS)



ELECTRICAL & COMPUTER ENGINEERING

Archived Notes & Lab Work

All copies of my handwritten notes as well as versions of my C and VHDL programs are available for reference.

Hand written course notes.

journal files\lab0_architecture_graphic.pdf

journal files\ece522HISTO_rgb.pdf

journal files\lab1_attempt1_todo.pdf

journal files\parsing states.pdf

journal files\sgc_notes_mod6.pdf

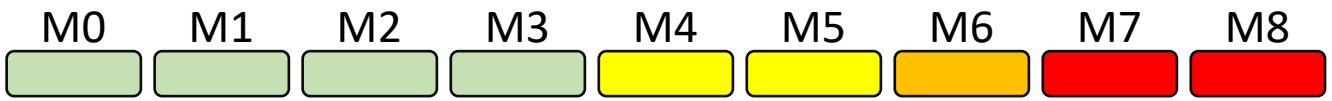
journal files\code analysis notes handwritten.pdf

Code Repos

XILINX WorkFlow Document:
https://github.com/0xbadc0fee/UNM-Setups/blob/main/522/lab_0_10/%40lab_0_10.md

Lab #0 (complete)
https://github.com/0xbadc0fee/ece522-lab0

Lab #1 (unfinished)
https://github.com/0xbadc0fee/ece522-lab1



ELECTRICAL & COMPUTER ENGINEERING

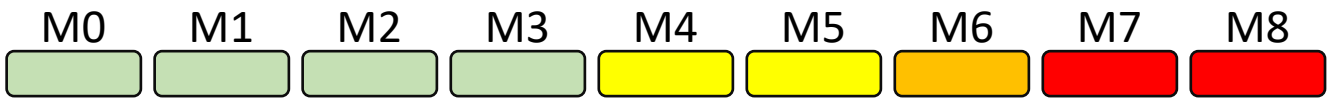
Conclusions / Personal Review

STRENGTHS:

1. Got an early start on class, no issues getting course materials and Zybo board
2. Extra work on setting up home lab / infrastructure paid off.
3. Choice of staying with Linux for dev host was a good move. Window's sub system for linux was eventually workable for others in class but I lost much less time sticking with a native Linux set up.
4. Course materials were excellent, but the Extra Materials really helped me with some difficult parts. I would definitely rewatch the LEB Books lessons on translating a specification all the way through to a hardware model.

WEAKNESSES:

1. Misconceptions early on really slowed me down later. Specifically all things BRAM; what it is, where it sits, how and when it writes, how and when it reads, width constraints, etc.
2. Little experience parsing existing code into higher level logic / abstractions. Took too long getting up to speed being able to "see" quickly where a BRAM read was, or where a MUX was being used to direct an output, etc. Need more exposure to get faster.
3. No prior exposure to VHDL made things more difficult but not impossible. Background in PLC's and Codesys IDE was helpful, but still too different from VHDL to pick up quickly
4. Granular learning style. Until I have combed through new concepts at a very low level, I feel unprepared to move to next modules. This happened between M5 and M6 and cost me a lot of time.
5. Limited low level hardware exposure. Background in Mechanical Engineering,



ELECTRICAL & COMPUTER ENGINEERING

Conclusions / Personal Review

Closing:

ECE-522 was a very challenging course, but also an excellent course to attempt. The material is extremely relevant. FPGA's are understood to be a difficult technology to pick up. The 8 week timeline of the Accelerate Online Program makes that difficulty more pronounced. I was aware going into it that this would be a difficult course and I prepared as much as I could in advance. Throughout these 8 weeks, this course has been my only priority. While remote, I am also a full-time student and do not work or take on jobs outside of the course I'm studying. As time and effort go, I put more hours into this course than any other yet.

The instructor, the video lectures, the slides, and the textbook are all excellent materials and I can think of no other changes. The biggest consumers of time in the short span of this course were 1) Configuring the base hardware platform + IDE and 2) Learning / Improving on one's ability to "see through" code and mentally translate it to something neither SW nor HW so that it could then be turned into either, reoptimized, or reconfigured without deviating from it's initial intent.

It was a tough class, but it was fair. In this case I just wasn't able to keep up at the end.