EE/CprE/SE 491 WEEKLY REPORT

4/10/23 – 4/16/23

Group number: 20

Project title: mm Wave Imaging radar

Client &/Advisor: Mohammed Tayeb Al Qaseer

Team Members/Role: Nathan Ayers: Interface Lead, Rodrigo Romero: Lead FPGA Programming Engineer, Michael Levin: Lead DSP Engineer, Matthew Caron: Lead Hardware Designer.

- **Weekly Summary**
  - We had an in class presentation over testing this week and we met as a group to discuss our final presentation/semester wrap up. We also had a meeting with Dr. Tayeeb to discuss progress over the last two weeks, and he gave us good feedback and insight for the upcoming final presentation.

- **Past week accomplishments**
  - Nathan Ayers: This week I was given a new cable called C232HM-EDHSL-0, it has an FTDI chip, which is older than the one we are using for our project but it allows me to work on communication with GPIO ports while Matt can use the actual FPGA. I had to install an older model of driver (D2XX) but I made some big breakthroughs. I had previously struggled with linking the driver library in my compilations but after hours of diagnosing, and trial and error I got it to work, so now I have access to the FTDI functions. I made two different programs, one used example code to create a device info list, when I ran it, it recognized the device was plugged into my laptop and printed off the expected information. The second program was to test the GPIO pins, these are very complex and in my opinion are way more complicated than they should be, so they will be difficult for me over the entire project. I will include code and pictures at the bottom of this document.
  - Rodrigo Romero: Meet with advisor and another teammate to work on the project. Plan to develop the programming structure of the equipment has been developed.
  - Michael Levin: Made the first design for multiplying complex numbers in vivado.
  - Matthew Caron: Got the Xilinx Virtual Cable working with Vivado and the Alchitry Au so we can now use just Vivado with programming the FPGA
- **Pending issues**
  - Nathan Ayers: None, I'm happy to have my new USB/FTDI device so that I can make progress, while Matt, Michael, and Rodrigo can as well.
  - Rodrigo Romero: Work on the slides for final presentation and website editions.
  - Matthew Caron: finding the time to work on the design

- **Individual contributions**

NAME	Individual Contributions *(Quick list of contributions. This should be short.)*	Hours this week	HOURS cumulative
Michael Levin	I started the template for the report and started the design for multiplying complex numbers in vivado	3	16
Nathan Ayers	Created a working environment for the D2XX drivers. Created C code for checking and reading connected devices. Wrote C code to blink an LED, so I feel much more confident about how GPIO works.	9	41
Rodrigo Romero	Worked on vivado and its IDE. Set up the planning for upcoming tasks.	4	19
Matthew Caron	Got the Xilinx Virtual Cable working with Vivado and the Alchitry Au so we can now use just Vivado with programming the FPGA	8	25

- **Plans for the upcoming week**
  - Nathan Ayers: Work more with the code created this week, I want to get to a point where I can read data from the pins and display that in my console. I also want to "translate" the code from C to C# as I have chosen that for the interface language.
  - Rodrigo Romero: Successfully Develop a template for the stage one of the software structure. Started to work on the plan for the final presentation.
  - Michael Levin: I plan on dissecting some similar DSP blocks that are offered in Vivado to see how the intricacies of a ADC works in verilog code.
  - Matthew Caron: work more on schematic design and work on echoing back data with the FPGA so Nathan can do more work with UI and usb communication

Appendix

Nathan Ayers' code and images:

Figure 1: C232HM USB 2.0 Hi-Speed to MPSSE Cable

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <windows.h>
#include <windef.h>
#include <winnt.h>
#include <winbase.h>
#include <string.h>
#include <math.h>
#include <stdbool.h>
#include <stdint.h>
#include "C:\Users\nayer\CDM v2.12.36.4 WHQL Certified\ftd2xx.h"
#include <sys/time.h>

int main(){
   FT_STATUS ftStatus;
   FT_HANDLE ftHandle;
   FT_DEVICE_LIST_INFO_NODE *devInfo;
   DWORD numDevs;
   // create the device information list
   ftStatus = FT_CreateDeviceInfoList(&numDevs);

   if (ftStatus == FT_OK) {
      printf("Number of devices is %d\n",numDevs);
   }

   if (numDevs > 0) {
      // allocate storage for list based on numDevs
      devInfo =
(FT_DEVICE_LIST_INFO_NODE*)malloc(sizeof(FT_DEVICE_LIST_INFO_NODE)*numDevs);
      // get the device information list
      ftStatus = FT_GetDeviceInfoList(devInfo,&numDevs);

      if (ftStatus == FT_OK) {
         for (int i = 0; i < numDevs; i++) {
            printf("Dev %d:\n",i);
            printf(" Flags=0x%x\n",devInfo[i].Flags);
            printf(" Type=0x%x\n",devInfo[i].Type);
            printf(" ID=0x%x\n",devInfo[i].ID);
            //printf(" LocId=0x%x\n",devInfo[i].LocId);
            printf(" SerialNumber=%s\n",devInfo[i].SerialNumber);
```

```
            printf(" Description=%s\n",devInfo[i].Description);
            printf(" ftHandle=0x%x\n",devInfo[i].ftHandle);
          }
        }
    }
    ftStatus = FT_Open(0, &ftHandle);
    if (ftStatus == FT_OK) {
      // FT_Open OK, use ftHandle to access device
      printf("Yay, I work!\n");
    }
    else {
      // FT_Open failed
      printf("Ouch\n");
    }
}
```

The code can be a little difficult to read, but the above code shows three of our D2XX/FTDI functions in use by a program. The first function, highlighted in green builds a device information list and then returns the number of devices connected. The second function, highlighted in yellow returns a device information list, essentially populating it given the number of devices. The third function, highlighted in blue simply opens the device and returns a handle to be used for subsequent accesses. All of these functions are honestly just for checking, helpful to an automated system, but still helpful in telling me that I am doing something right.

Code Output:

```
PS C:\Users\nayer\D2XX> gcc -o main anything.c -L"C:\Users\nayer\CDM v2.12.36.4 WHQL Certified\i386" -lftd2xx
PS C:\Users\nayer\D2XX> ./main
Number of devices is 1
Dev 0:
 Flags=0x2
 Type=0x8
 ID=0x4036014
 SerialNumber=FTYI9TEQ
 Description=C232HM-EDHSL-0
 ftHandle=0x0
Yay, I work!
```

So this code returned all of the output that was expected so it was a success!

Blink Code:

```
do{
    printf("c = %d\n", c); // Just to check our count essentially
    if (val == 0xff)
    {
       val = 0;
    }
    else
    {
       val = 0xff;
    }
    byOutputBuffer[0] = 0x80;
                // configure data bits low-byte of MPSSE port
    c++;
    byOutputBuffer[1] = val; // actual data, essentially high or low
                // initial state config above

    byOutputBuffer[2] = 0xff; //which pins are out/in
                // direction config above;

    printf("byOutputBuffer[1] = %d, dwNumBytesToSend = %d\n",byOutputBuffer[1], 3);

    ftStatus = FT_Write(ftHandle, byOutputBuffer, 3, &dwNumBytesSent);
                // send off the high GPIO config commands
                dwNumBytesToSend = 0;
                Sleep(milisecs);

      }while(c<count*2);
```

This is just a portion of the blink code but it is the only part that has to do with the blinking, I learned from various online resources that the pins can be controlled with a series of three hexidecimal numbers, each one byte. The first number determines which pins we are referring to, that doesn't apply to this cable I am using now as there are only 8 pins. The second number refers to data value of the output pins ie. the ones you are writing to. I just went back and forth between 0xFF and 0x00 to ensure high to low switching every second. The last number refers to the direction of the pins, with 1 being output and 0 being input. I found that this code works and set up and LED to the GPIO pins and it blinked every second for 10 seconds. It could be better and it will be better as I work on it more.