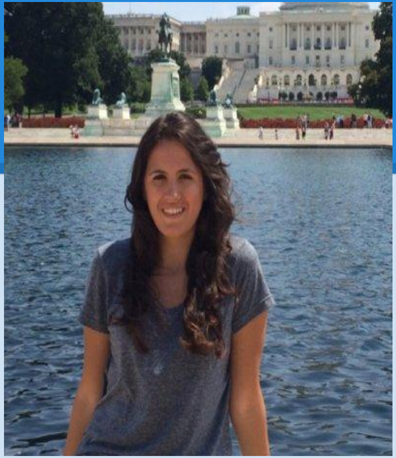
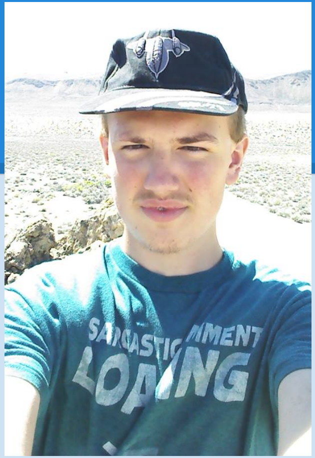
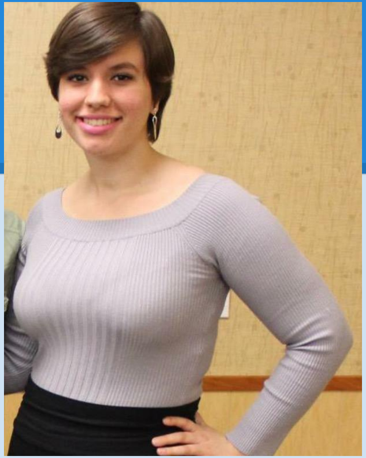
SDR - Spectrum Sensing

Christina Baaklini, Michael Collins, Nick Cooper, and Nicole DiLeo



Overview

- **Implementation of State Machines**
- **FFTW C Library**
- **Gnuplot**

Implementation of State Machines

- **State machines are often the backbone of FPGA development.**
- **They are divided into two basic output classes:**
  - **Mealy takes into account both internal state and inputs**
  - **Moore only utilizes internal state**
- **State machine described as ASM is easier to map to implementation in a hardware description language such as VHDL.**
- **In order to implement a state machine from the state diagram, VHDL is used. Typically a register (D FF) is used in conjunction with output logic (Me vs Mo), and next state arguments.**

Case statements

- **Case statements are the main implementation of FSMs**
- **Example is of a 2-state FSM for a model elevator controller (Mealy output logic)**

```
case sig1 is
when '0' =>
if (btn1='1') then
sig2 <= '0';
elsif (btn2='1') then
sig2 <= '1';
end if;
when '1' =>
if (btn1 = '1') then
sig2 <= '0';
elsif (btn2 = '1' then)
sig2 <= '1';
end if;
end case;
process
begin
led1 <= '0';
led2 <= '0';
case sig1 is
when '0' =>
if (btn1='1' and btn2='1') then
led1 <= '0';
led2 <= '0';
elsif (btn1='1') then
led1 <= '1';
led2 <= '0'
elsif (btn2='1') then
led1 <= '0';
led2 <= '1';
else
led1 <= '1';
led2 <= '0';
end if;
when '1' =>
if (btn1='1' and btn2='1') then
led1 <= '0';
led2 <= '0';
elsif (btn1='1') then
led1 <= '1';
led2 <='0'
elsif (btn2='1') then
led1 <= '0';
led2 <= '1';
else
led1 <= '0';
led2 <= '1';
end if;
end case;
end process;
```

FFTW C Library

```
in_ = (fftw_complex*)fftw_malloc(sizeof(fftw_complex)*fft_size_);
out_ = (fftw_complex*)fftw_malloc(sizeof(fftw_complex)*fft_size_);

plan_ = fftw_plan_dft_1d(fft_size_,in_,out_,FFTW_FORWARD,FFTW_ESTIMATE);
fft_data_.clear();

while (stop(index, fft_size_ , overlap_) < N) {

    for (unsigned int i = start(index, fft_size_, overlap_);
          i <= stop(index, fft_size_, overlap_); i++) {
        s.at(i-start(index, fft_size_, overlap_)) = iq_samples_.at(i);
    }

    for (unsigned int i = 0; i < fft_size_; i++){
        s2.at(i) = (window_.at(i))*(s.at(i));
    }

    for (unsigned int i = 0; i < fft_size_; i++){
        in_[i][0] = s2[i].real();
        in_[i][1] = s2[i].imag();
    }

    fftw_execute(plan_);
}
```

- **Fastest free implementation of Fast Fourier Transform**
- **Resolved issues with memory allocation**
- **Used `fftw_malloc` to allocate memory appropriately**
- **Can now transform IQ time samples into frequency domain**

https://github.com/FFTW/fftw3

Gnuplot

- Open-source, cross-platform graphing utility
- Used in Octave for plotting
- Can be controlled in C++ through gnuplot-iostream interface
- We plan to use this interface to generate waterfall plots, power vs. frequency, etc. in C++ implementation

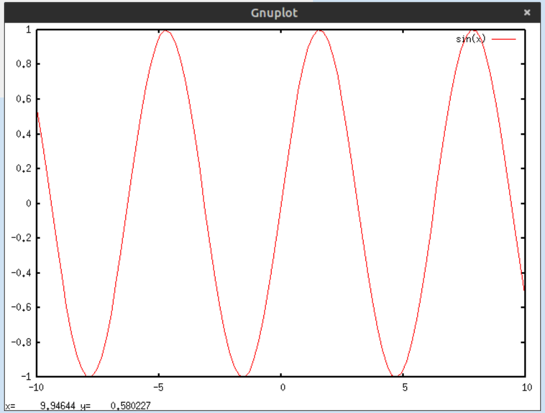
http://www.gnuplot.info/

https://github.com/dstahlke/gnuplot-iostream

```cpp
#include "gnuplot-iostream.h"

int main() {
    Gnuplot gp;
    gp << "set terminal x11\n";
    gp << "plot sin(x)\n";
    gp.flush();

    return 0;
}
```



Next Week

- **Take a look at Spectrum Sensing framework and begin to incorporate our FSM designs**
- **Incorporate C++ script with Wiserd receiver module**
- **Implement real-time processing and plotting**