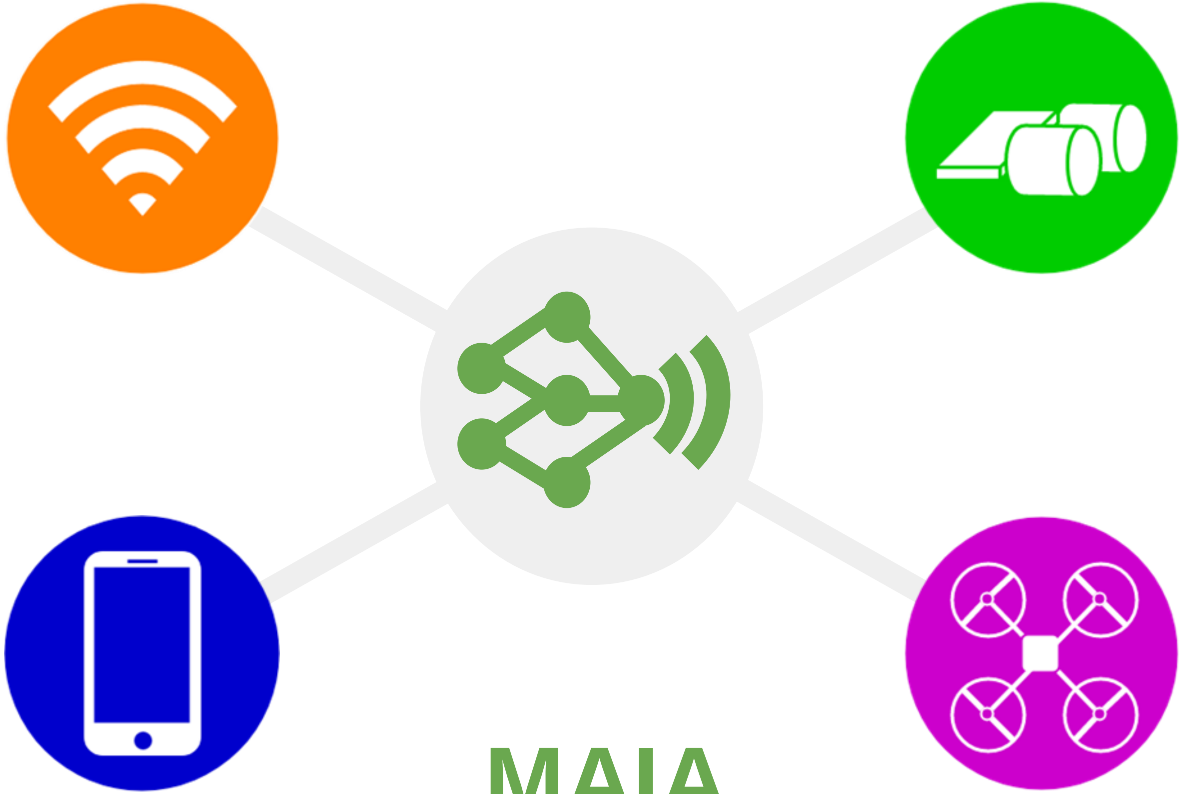
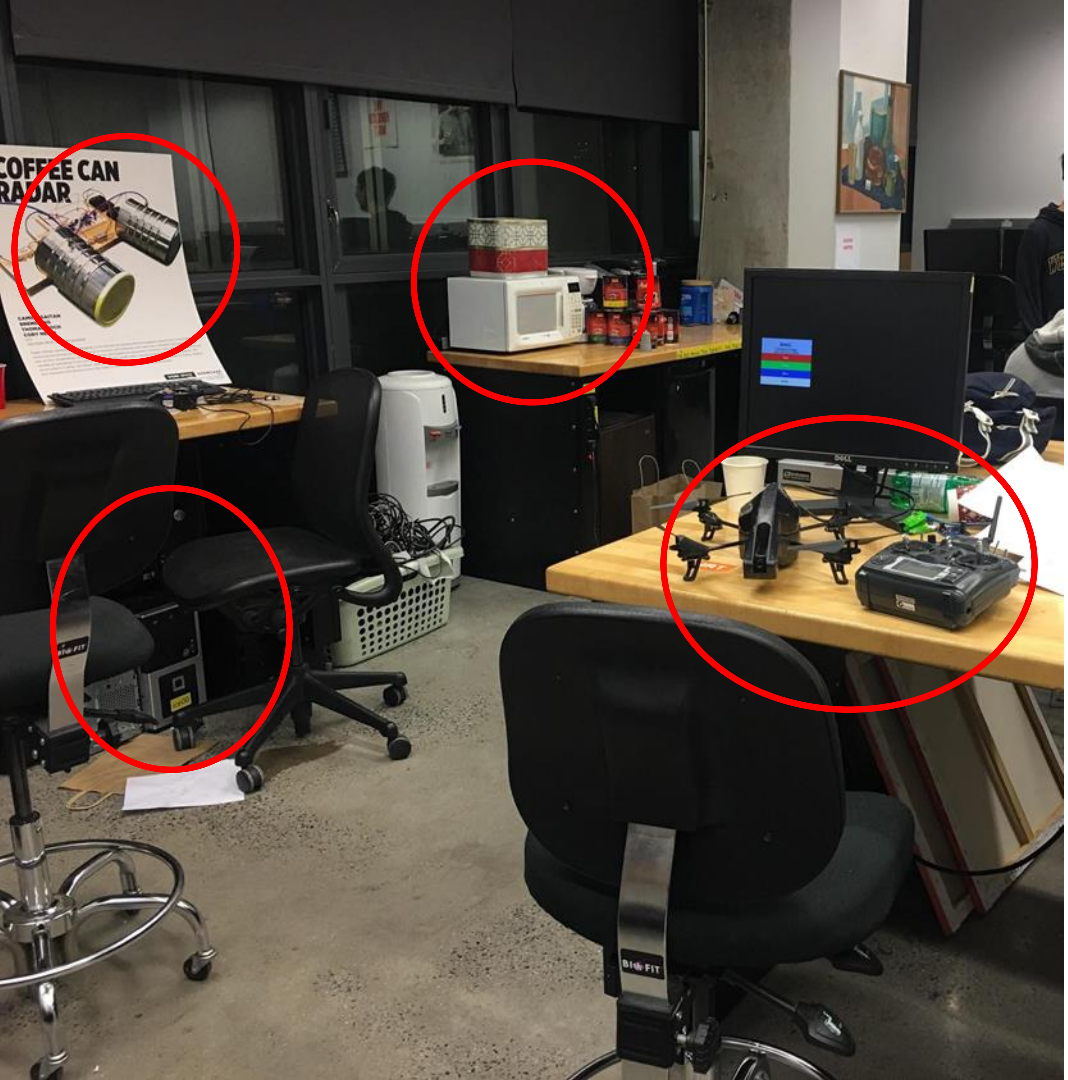


MAIA

Modulation Classification Artificial Intelligence Automata

Mid Year Report

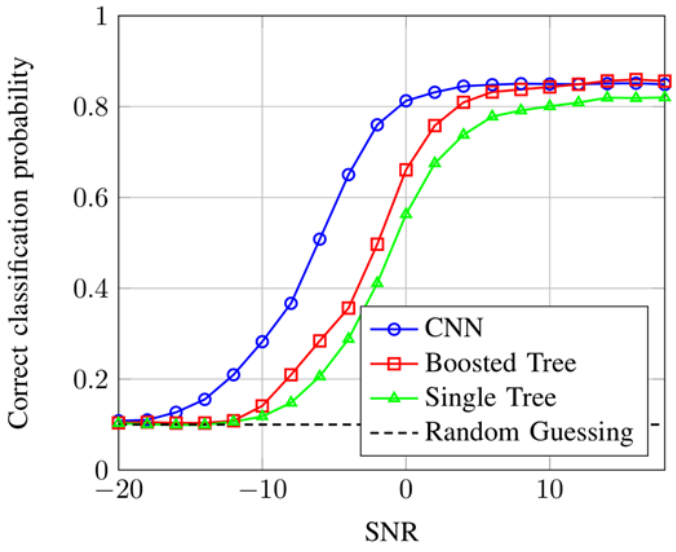


A lot of things run on the ISM band, how do we differentiate between them?

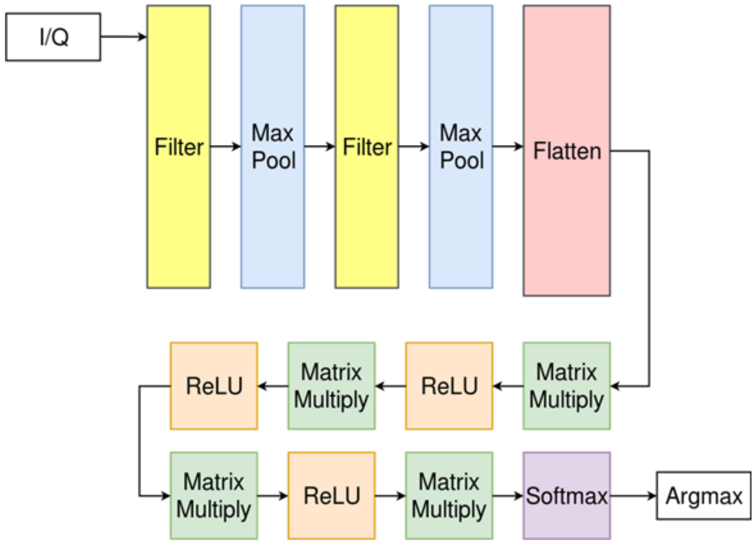
Devices may be identified by their **modulation schemes**

We can use **classification** to determine what is transmitting in our environment

What classifier should we use?



Accuracy of different classifiers



Neural network architecture

Decision : Use a convolutional neural network (CNN)

What hardware to choose?

	Central Processing Unit (CPU) 	Graphics Processing Unit (GPU) 	**Field Programmable Gate Array (FPGA)** 
Cost	$200	$1000	**$900**
Power Consumption	65 W	250 W	**60 W**
Latency	6 clock cycles	~ 500 cycles	**Depends on Implementation**
Throughput	252 GFlops	11 TFlops	**1.7 TMACs**

How do we use an FPGA?

Step 2 : ARM Core preprocesses data

Embedded software

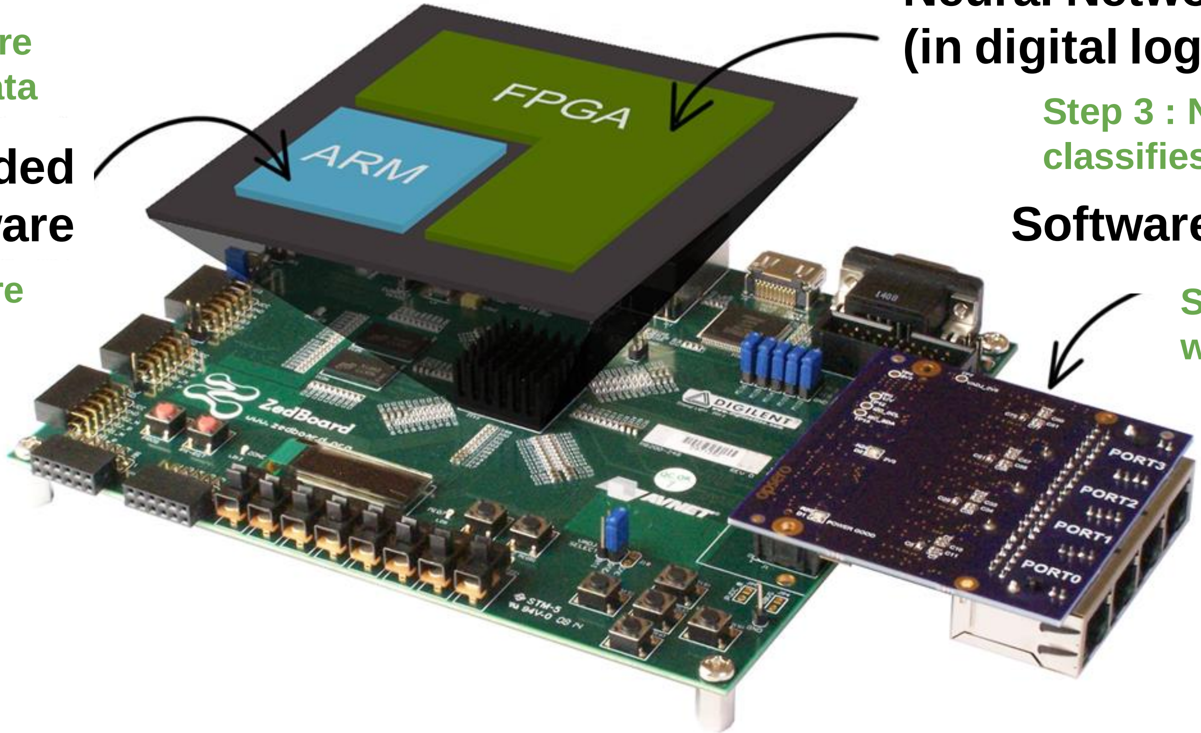
Step 4 : ARM Core outputs results

Neural Network Classifier (in digital logic)

Step 3 : Neural network classifies signals

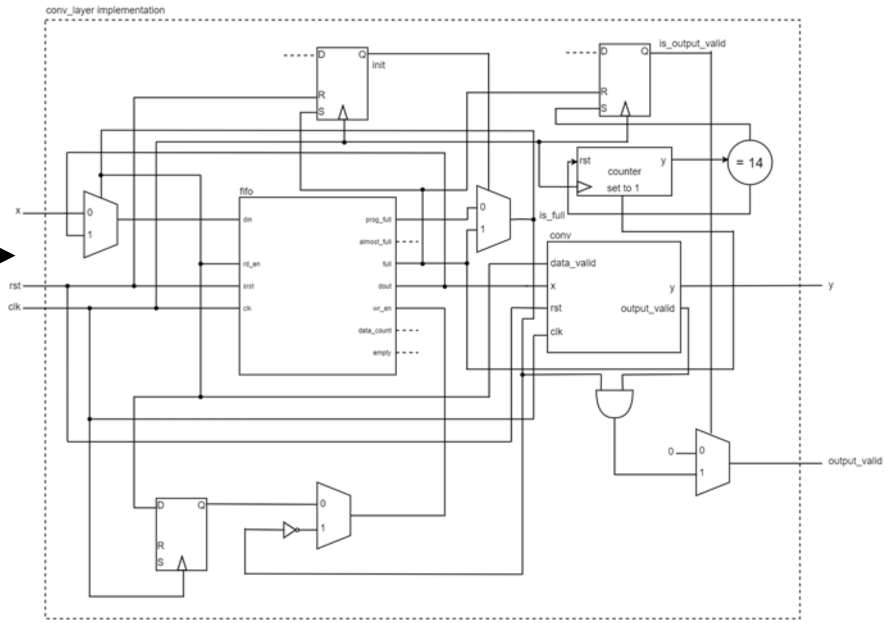
Software Defined Radio

Step 1 : Receives wireless signals



Challenge : Translate from Software to Hardware

```python
def build_model(self):
    with tf.variable_scope("LogReg"):
        # create placeholders for datapoints and labels
        self.x = tf.placeholder(tf.float32, (None, self.num_dim, self.num_samples,1), name = 'x')
        tf.summary.histogram('x',self.x)
        print(self.x.get_shape())
        self.dropout_rate = tf.placeholder(tf.float32, name='dropout')
        self.y = tf.placeholder(tf.int32, (None, self.num_mods), name = 'y')
        # Convolution + Relu : 128 filters, size 2*8 , output dimension should be 128 * 121
        self.net0 = tf.layers.conv2d(inputs = self.x, filters=128, kernel_size=[2,8],
                                     padding="valid", activation = tf.nn.relu, use_bias=False, name='conv0')
        tf.summary.histogram('conv0',self.net0)
        net0_1 = tf.layers.dropout(inputs=self.net0, rate=self.dropout_rate);

        # Max Pooling : size 2, stride 2, output dimension should be 128 * 60
        net1 = tf.layers.max_pooling2d(inputs=net0_1, pool_size=[1,2], strides=2, name='maxpool0')

        # Convolution + Relu : 64 filters, size 1 * 16, output dimension should be 64 * 45
        net2 = tf.layers.conv2d(inputs = net1, filters=64, kernel_size=[1,16],
                                     padding="valid", activation=tf.nn.relu, use_bias=False, name='conv1')
        net2_1 = tf.layers.dropout(inputs=net2, rate=self.dropout_rate)

        # Max Pooling : size 2, stride 2, output dimension should be 64 * 22
        net3 = tf.layers.max_pooling2d(inputs=net2_1, pool_size=[1,2], strides=2, name='maxpool1')

        # Flatten : Output dimension should be 1408
        net4 = tf.contrib.layers.flatten(inputs=net3)

        # FC + Relu : Output dimension should be 128
        net5 = tf.layers.dense(inputs=net4, units=128, activation=tf.nn.relu, name='dense0')
```

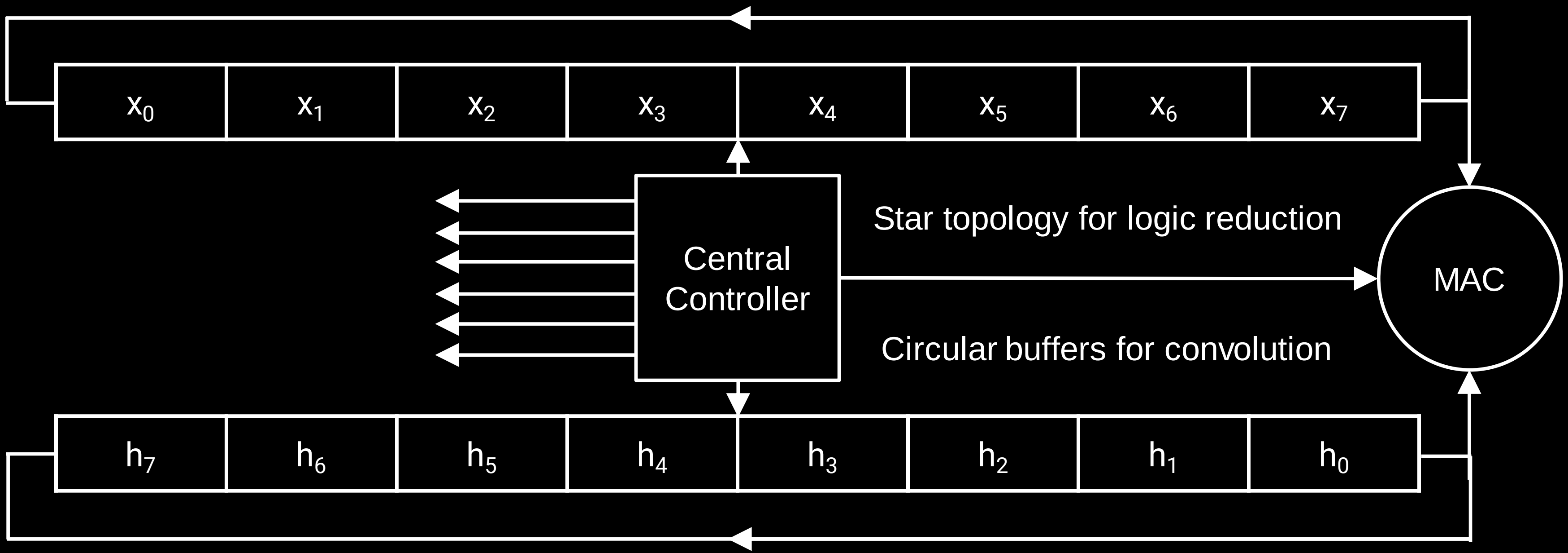


Game Plan

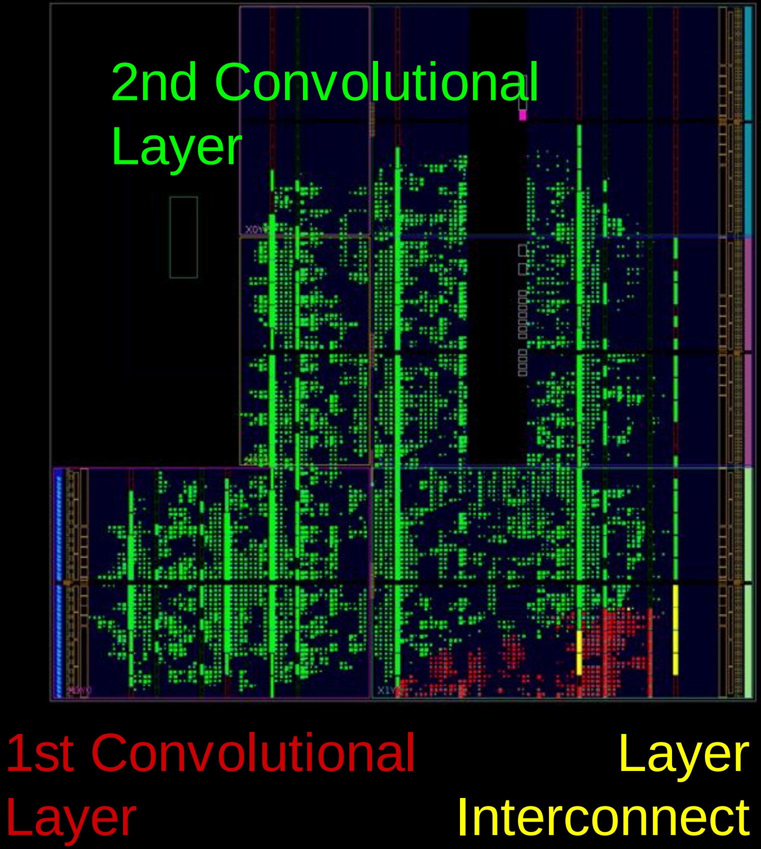
Neural Network Related	Non-Neural Network Related
Train Neural Net on GPU	Design Transmitter for Testing
Design Neural Net in Hardware	Verify Transmitter
Implement Hardware in FPGA	Interface FPGA with receiver
Hardware Optimization	Verify Receiver

Current Progress

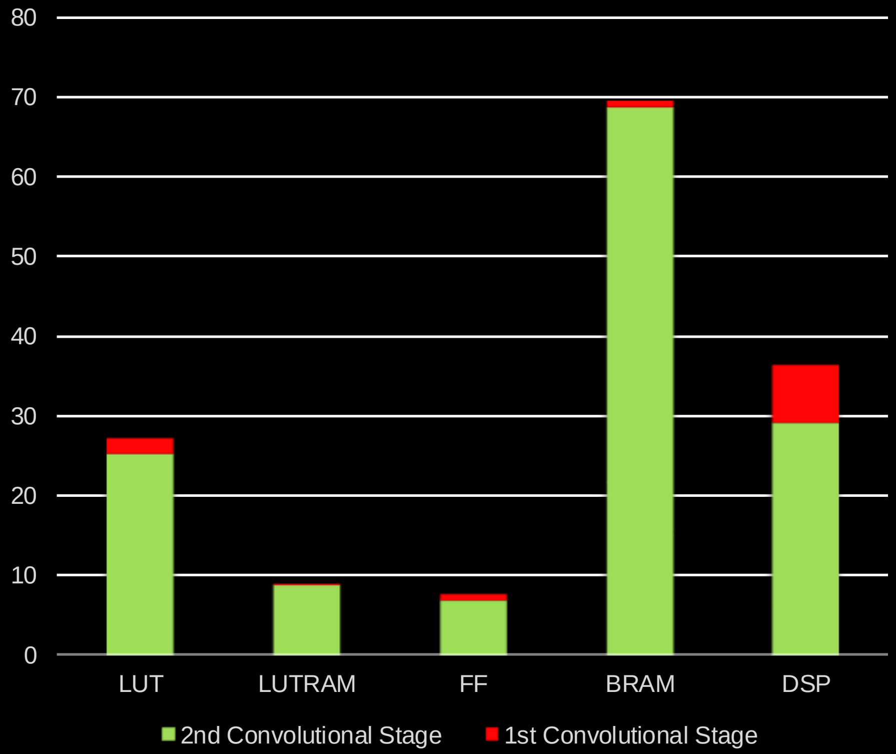
Common Design Patterns



FPGA Fabric Layout



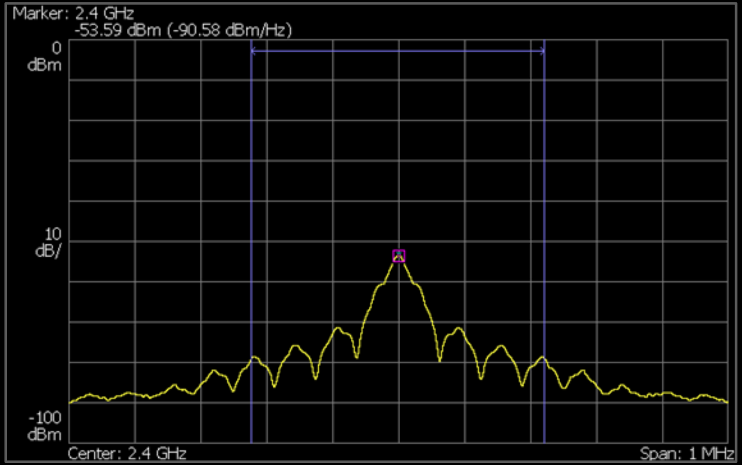
Resource Utilization of Different Stages



Experimental Setup

FPGA radio receiver
(with software neural network for verification)

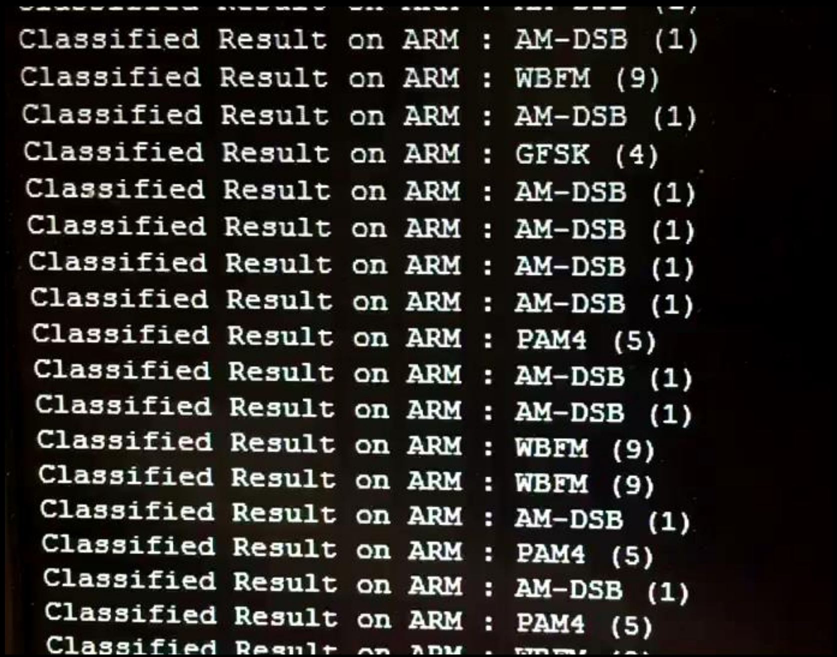
Transmitted spectrum displayed on spectrum analyzer



Software defined radio transmitting BPSK

Transmitter is verified!

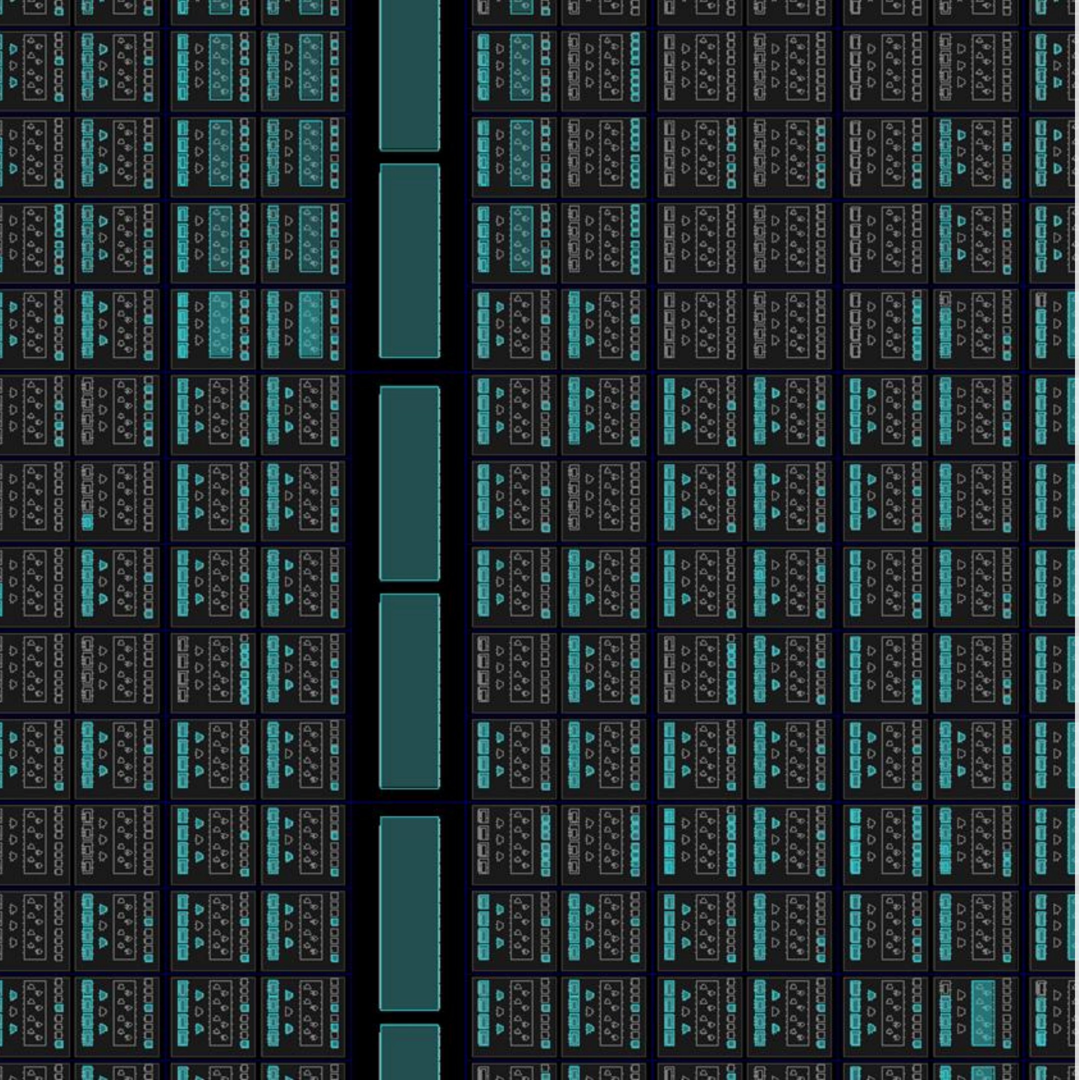
Current Status : Live classification of signals with Neural Network on ARM Core



Transmitter is sending BPSK Signals

Receiver only classified BPSK once

Next Steps : Debug Receiver



Next Steps

Finish interfacing receiver and FPGA

Finish implementing baseline hardware

Hardware Optimization