Wireless Modulation Classification with Custom Machine Learning Hardware

Brenda So, Cory Nezin
Advisor: Professor Toby Cumberbatch

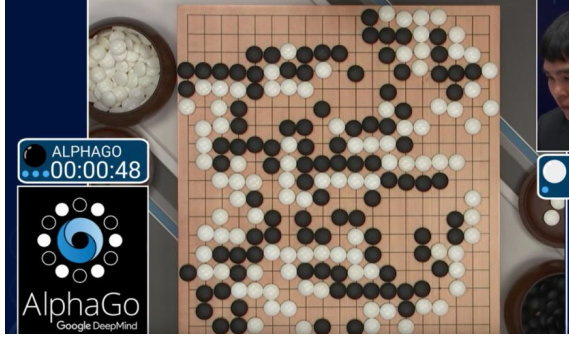
Neural Network Explosion

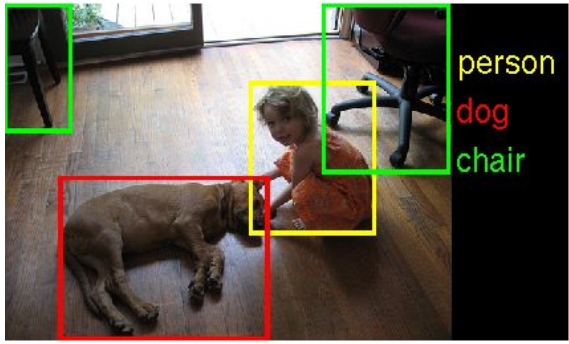
Advancements in GPUs led to popularity of **convolutional neural networks (CNNs)**

CNNs are found to supercede human and traditional machine learning performance

Recent research expand to areas such as **communication electronics** (O'Shea 2017)

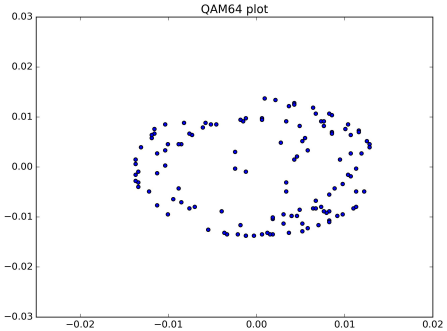
Challenges: Communication electronics are usually built on **small FPGA embedded devices without GPUs**

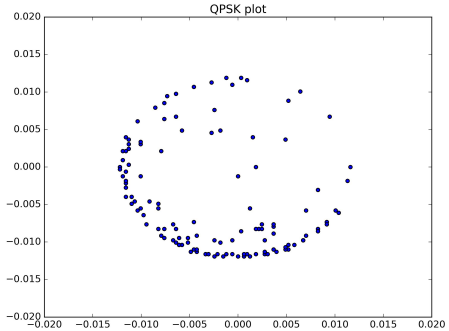


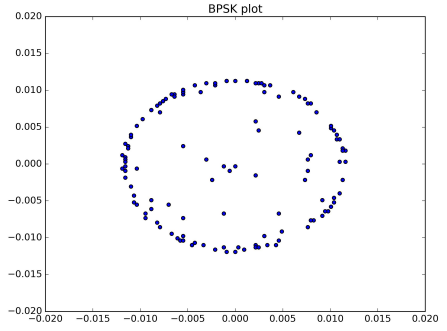


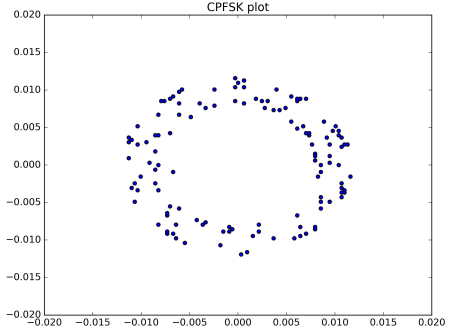
Communications use case: Modulation Classification

Which of the following is BPSK??





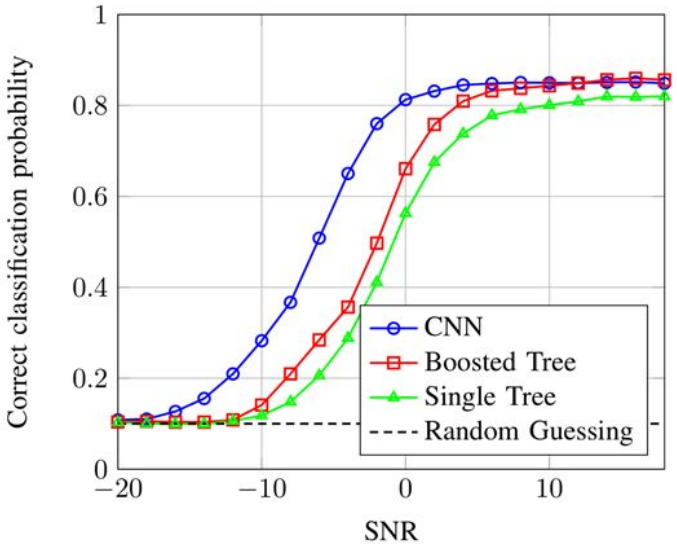




Need to identify types of wireless signal around us with low power and latency (e.g. Army signal classification challenge)

Goal: Put a neural network on an FPGA to classify modulations in real time

State of the Art Classifier



State-of-the-Art CNN Classifier Results
(O'Shea and Hoydis 2017)

Benefits: High Accuracy, Trained on Open Source Dataset

Drawbacks: Trained on synthetic data, Not feasible on small embedded device

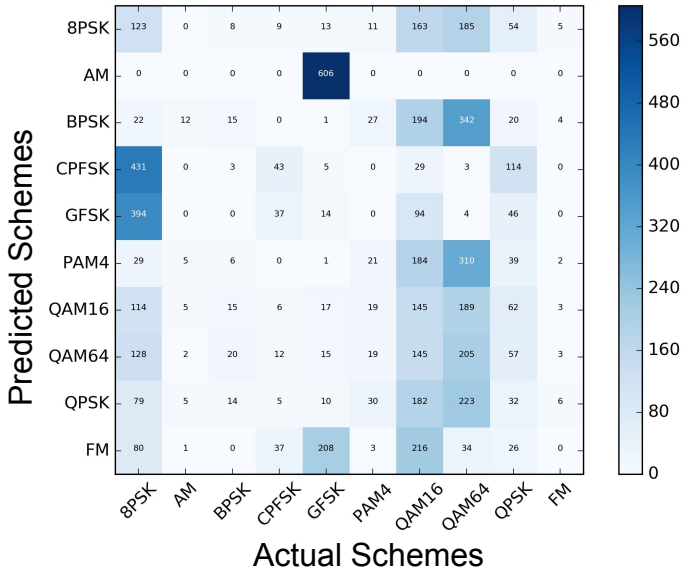
Large Memory Footprint : Over 330k weights in model

High Power Consumption : Trained and Tested on GPU

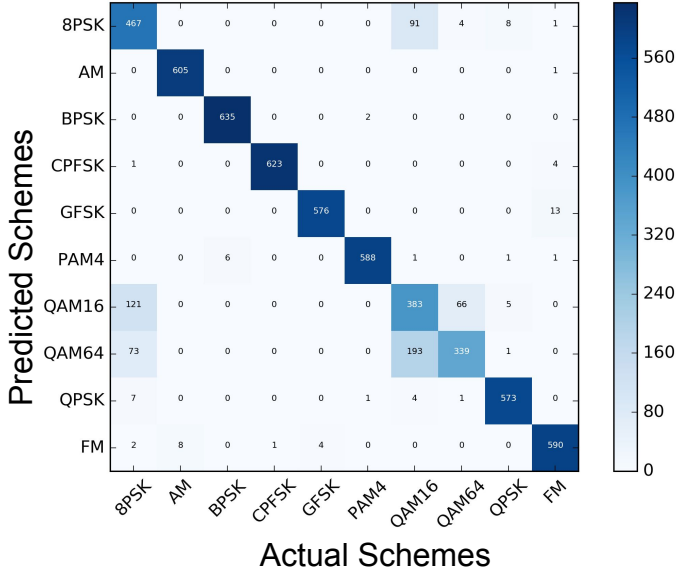
Synthetic data does not fully capture real life environments

Real-Life Data Collection

We found out that the classifier cannot classify real-life data, therefore we collected real-life data, retrained the model and obtained better accuracy



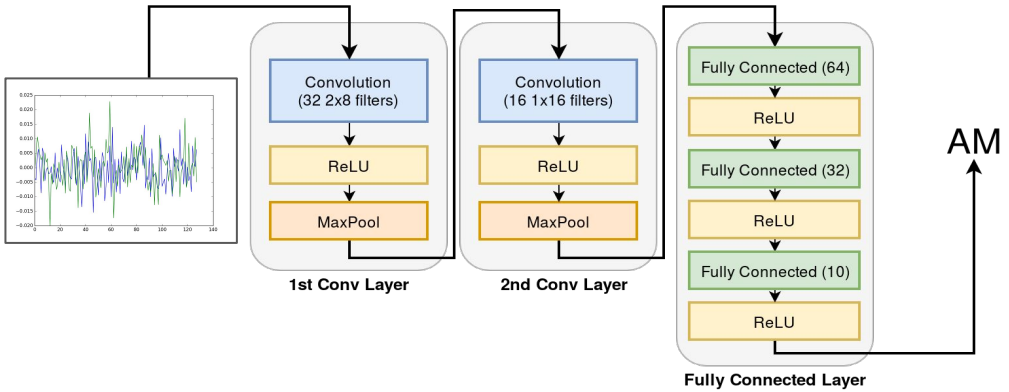
Model trained with synthetic data

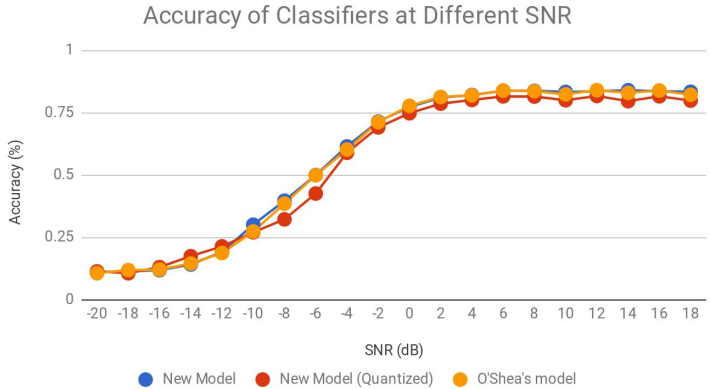


Model trained with real data

Architecture Redesign

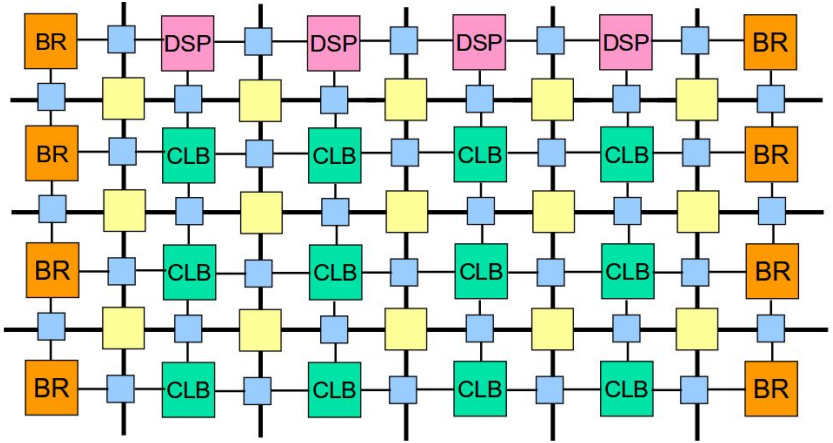
Since the pre-existing model is too large to fit on an FPGA, we investigated 40 different architectures layouts and model sizes.





New Design reduces model size by 90% with similar accuracy

FPGA Challenge : Hardware Constraints



Depiction of FPGA fabric

There are different digital elements in FPGAs:

- **Processing Elements**
  - **CLB** : Contains Lookup tables and registers
  - **DSP** : Processors for **multiplication**
  - **Switches and Routing Logic**
- **Memory Elements**
  - **LUTRAM**: Memory in logic
  - **BRAM**: Dedicated memory in RAM

Benefits: Low latency and power, optimized for parallel fixed point operations

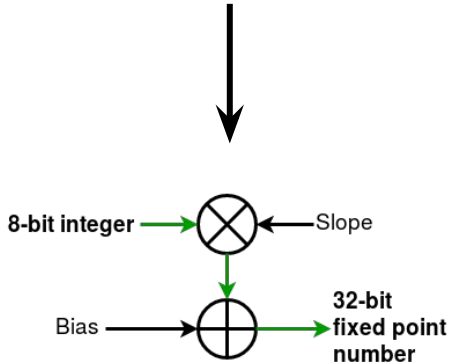
Challenges: Radio module uses 30% memory and logical elements, hence need to minimize processing and memory elements during design

Design Patterns of Neural Networks

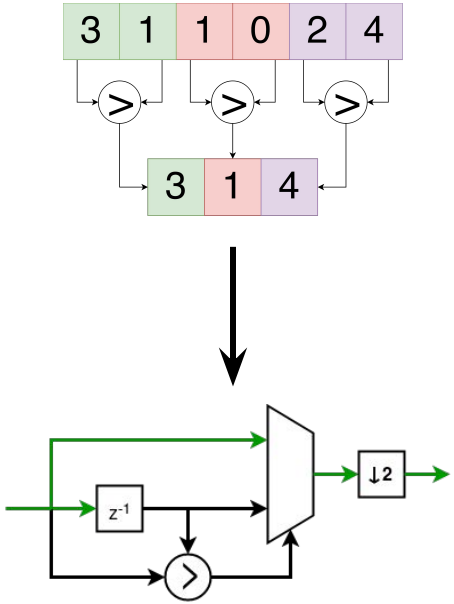
Slope Bias Loader

Stores 32-bit numbers as 8-bit integers

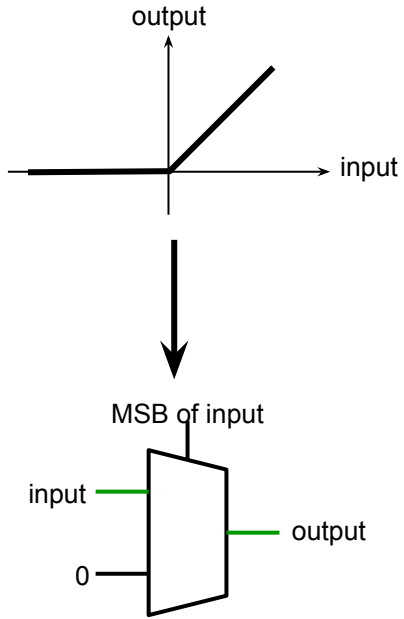
> *Real world value*
> *= slope x integer + bias*



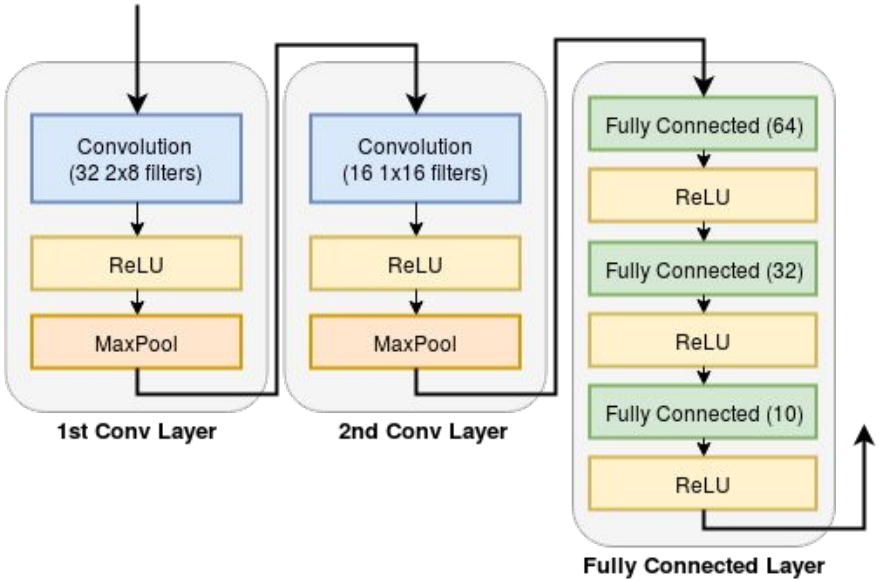
Maximum Pooling (MaxPool)



Rectified Linear Unit (ReLU)



Architecture Breakdown

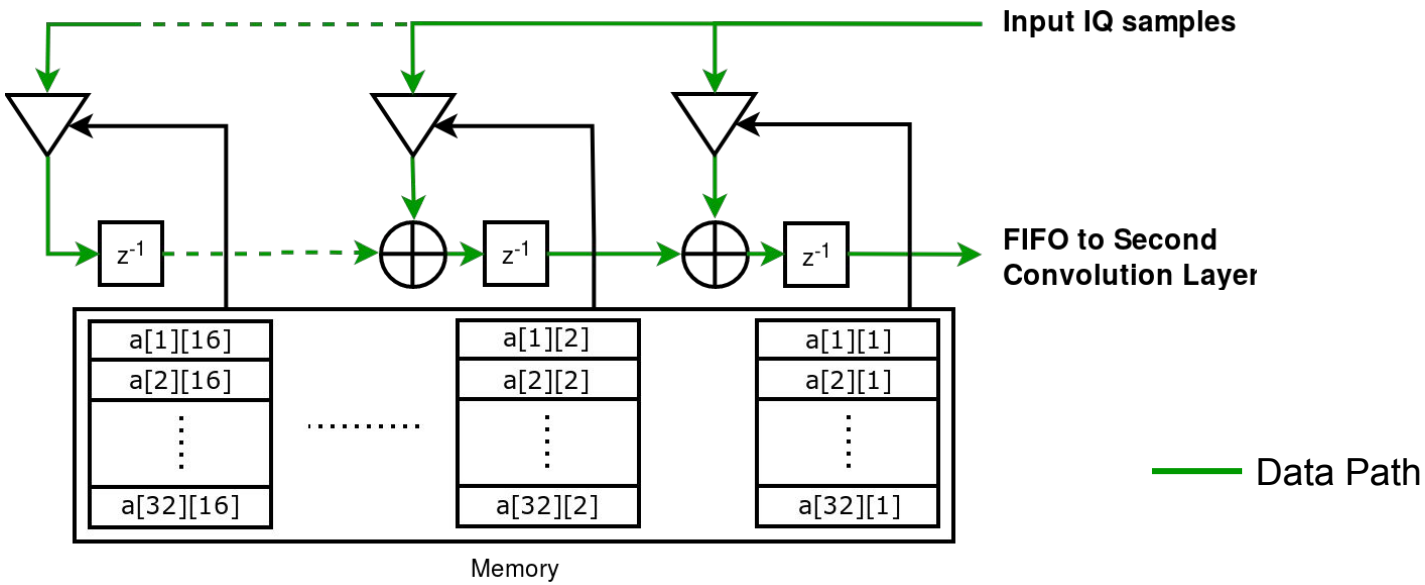


The architecture is broken down into 3 parts: **First Convolutional Layer**, **Second Convolutional Layer**, **Fully Connected Layer**

First Convolutional Layer

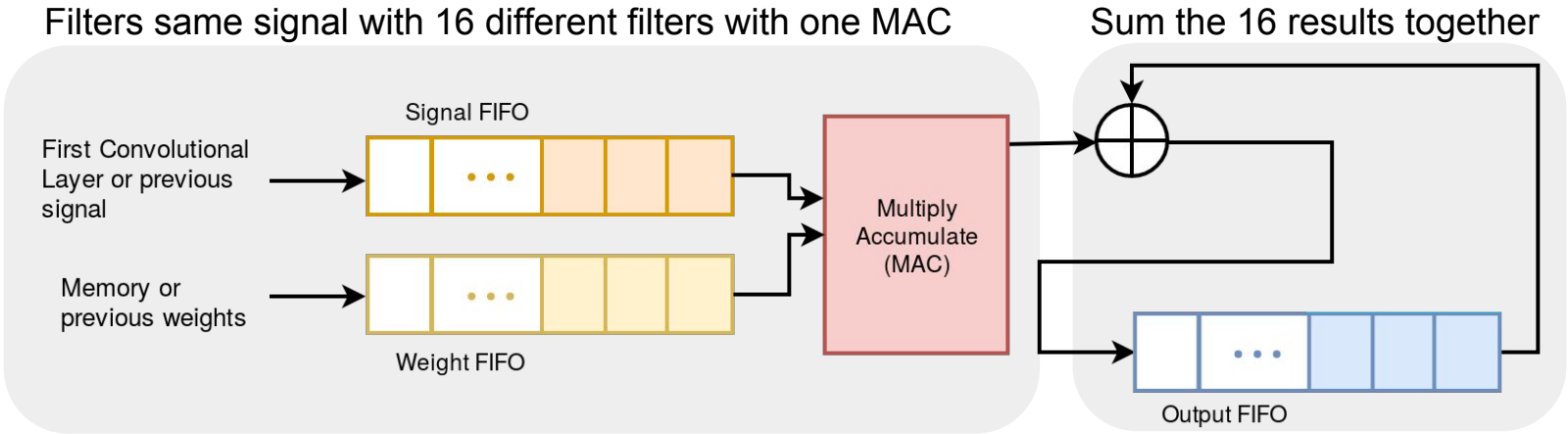
Operation: 64 8-tap FIR filters, summing every 2 results (i.e. sum IQ outputs)

Design: Time Domain Multiplexing FIR filters (64 filters -> 2 filters)



Second Convolutional Layer

Operation: 512 16-tap FIR filters, summing every 16 results



Design:

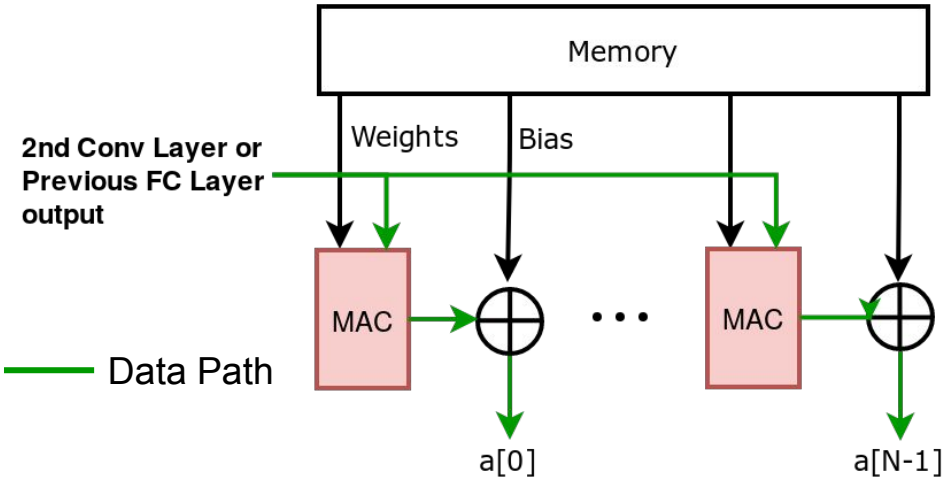
Accumulate one value at a time to save DSP slices

Performs 16 filter operations in parallel with 16 DSP slices

Uses static instead of dynamic memory to save power

Fully Connected (FC) Layer

Operation: Matrix Multiply & Bias Vector Addition



FC Layer with N-1 output array size

Design

Output Stationary Approach: The output of each MAC corresponds to one element in the output array directly

Shared same unit between last three fully connected layers (106 MACs -> 64 MACs)

Comparison with other hardware

Model	Power Usage	Latency
GPU (O'Shea)	70 W	80 ms
GPU (New Model)	70 W	90 ms
FPGA (New Model)	**3.4 W**	**< 1 ms**

GPU latency is the average latency per inferred sample

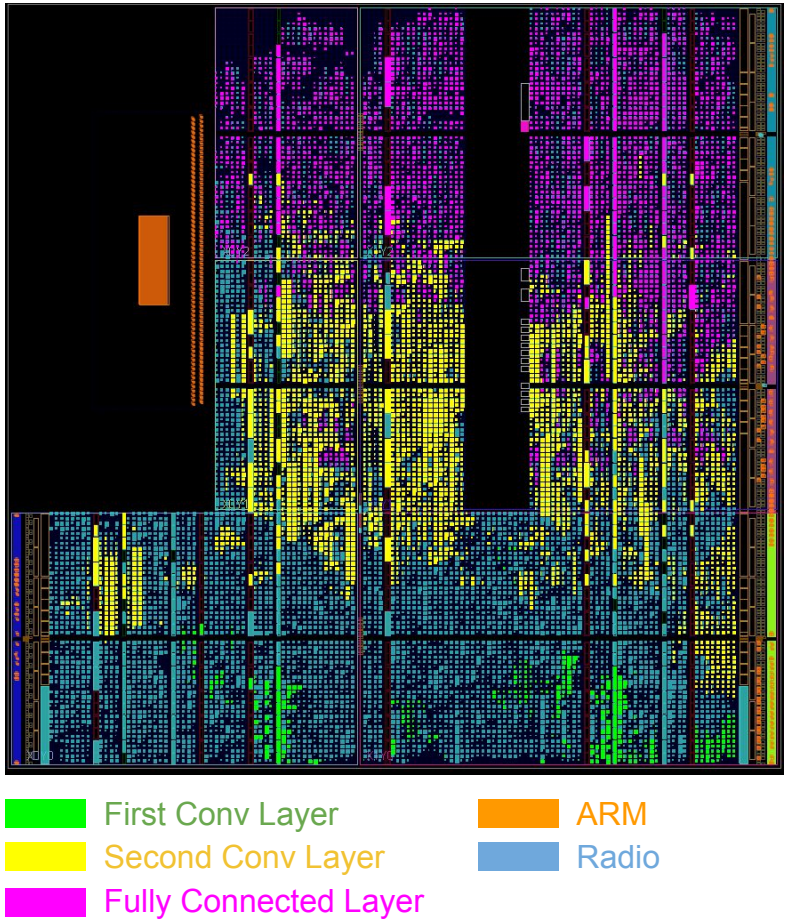
FPGA latency is the estimated latency from the number of clock cycles required with a 50MHz clock

GPU power usage was measured by hardware monitoring software

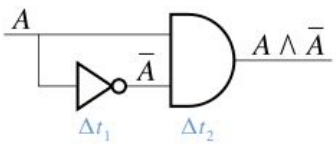
FPGA has lower latency and power when compared to GPUs

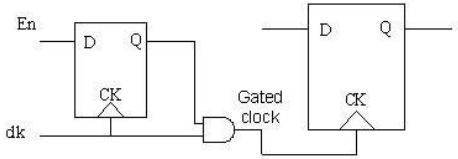
FPGA Layout and Utilization

Resource	CNN Model (% used)	Radio Module (% used)
Processing		
Look up Table	60%	25%
Flip flop	9%	24%
DSP Slices	51%	31%
Memory		
LUTRAM	22%	1%
BRAM	24%	4%

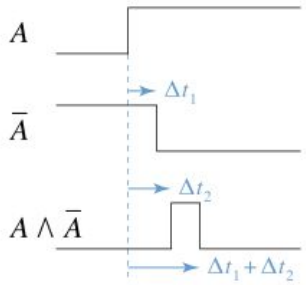


Timing Issues

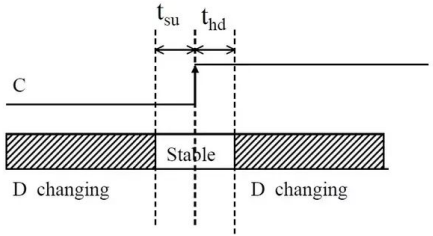




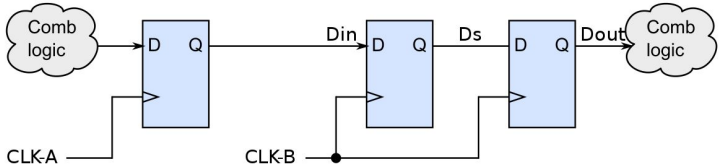
Imperfect Gated Clock



Race Conditions



Setup/Hold Time Violations



Metastability

Mitigation Strategies:

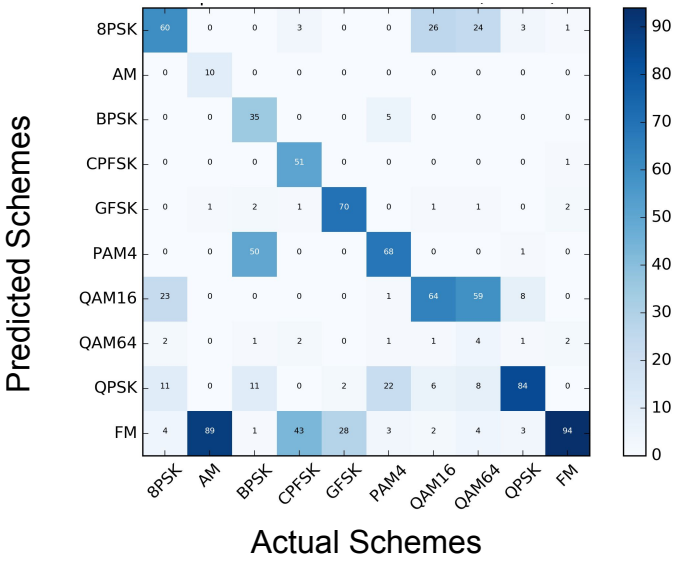
Synchronize the whole system at one 50MHz (½ Max Frequency) clock

Use clock enable pins instead of gated clocks

Redesign parts of the system that are asynchronous

Demonstration

Experimental Results



One-Off Experimental Results on FPGA

Hardware	CPU	FPGA
One-off Accuracy	57%	54%
Voting Accuracy	69%	62%
Time per classification	45 ms	0.6 ms

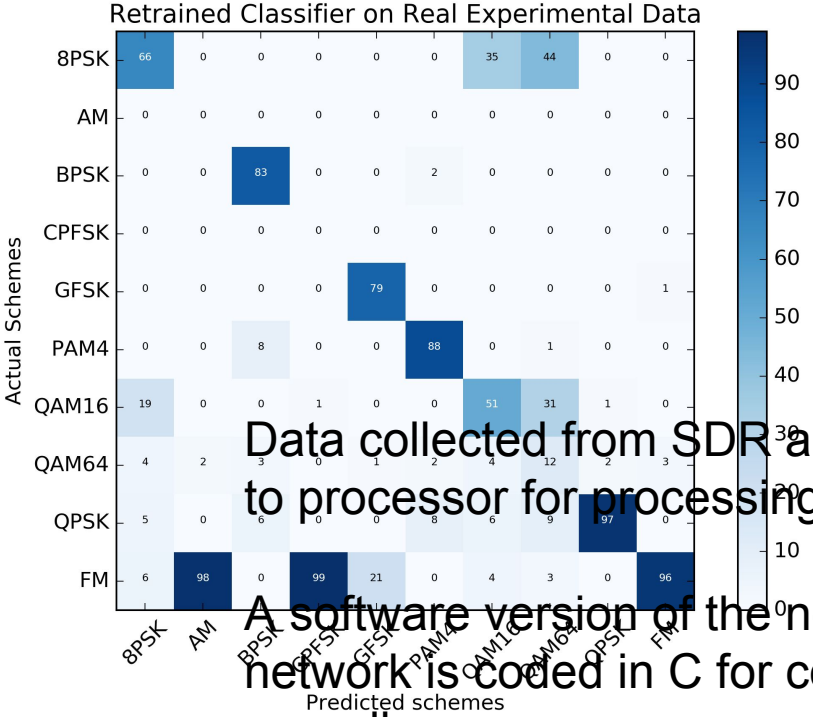
Data collected from radio and streamed for processing
A software version of the neural network is coded in C for comparison as well
We tested instant and voting classification and compared their accuracies

Conclusion

We classified modulations with a neural network on an FPGA

Custom Architecture can make significant improvements over GPUs in terms of power and latency

Demonstrated that CNNs on FPGAs can be possible for mobile artificial intelligence solutions aimed towards wireless applications



Hardware	CPU	FPGA
One-off Accuracy	57%	54%
Voting Accuracy	62%	69%
Time per classification	0.6 ms	45 ms

Data collected from SDR and streamed to processor for processing

A software version of the neural network is coded in C for comparison as well

Hardware Visualization

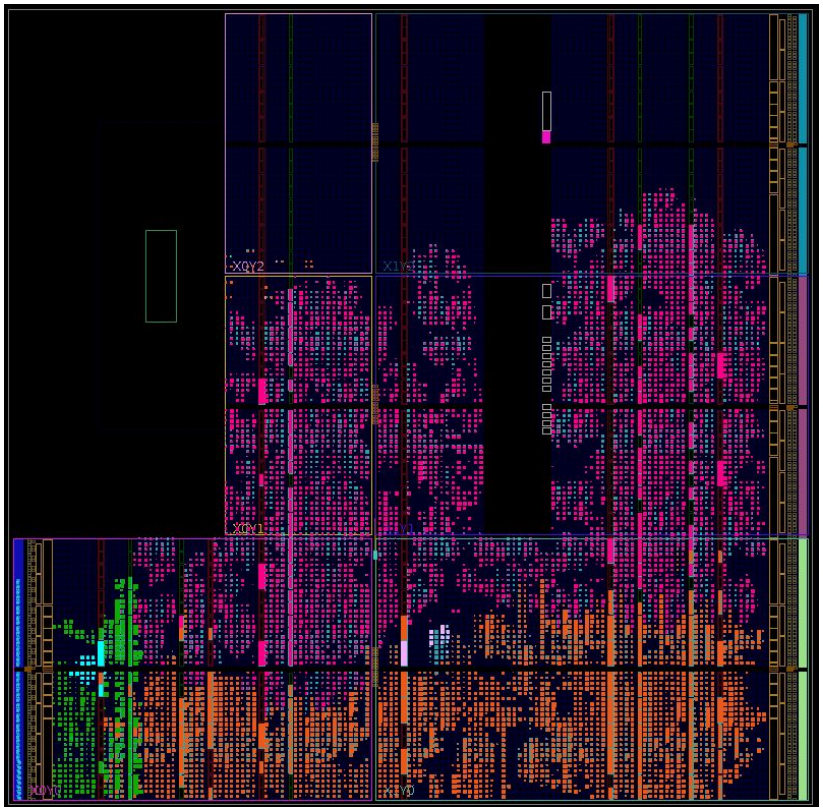
Green: conv1

Orange: conv2

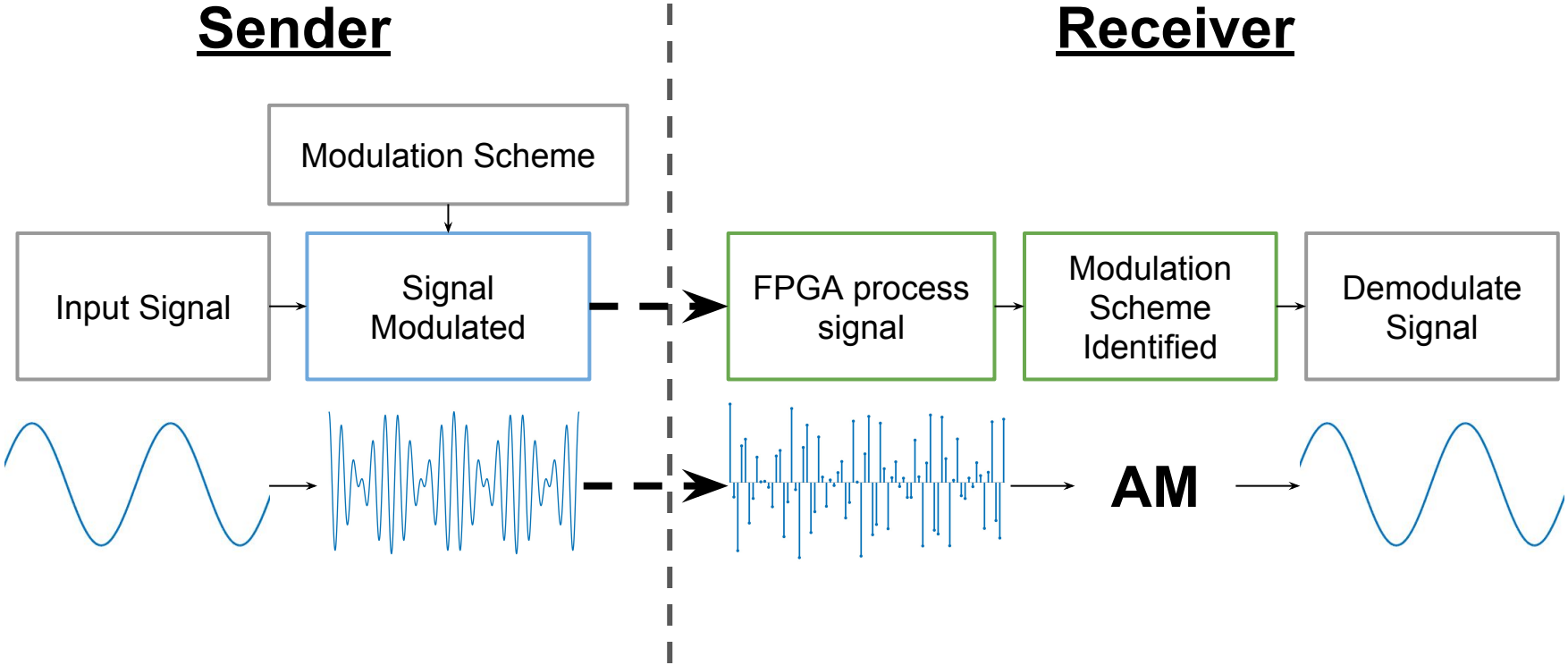
Purple: fully connected

Cyan: Interconnect FIFO 1

Pink: Interconnect FIFO 2



Communications use case : Modulation Classification



First Convolution Layer

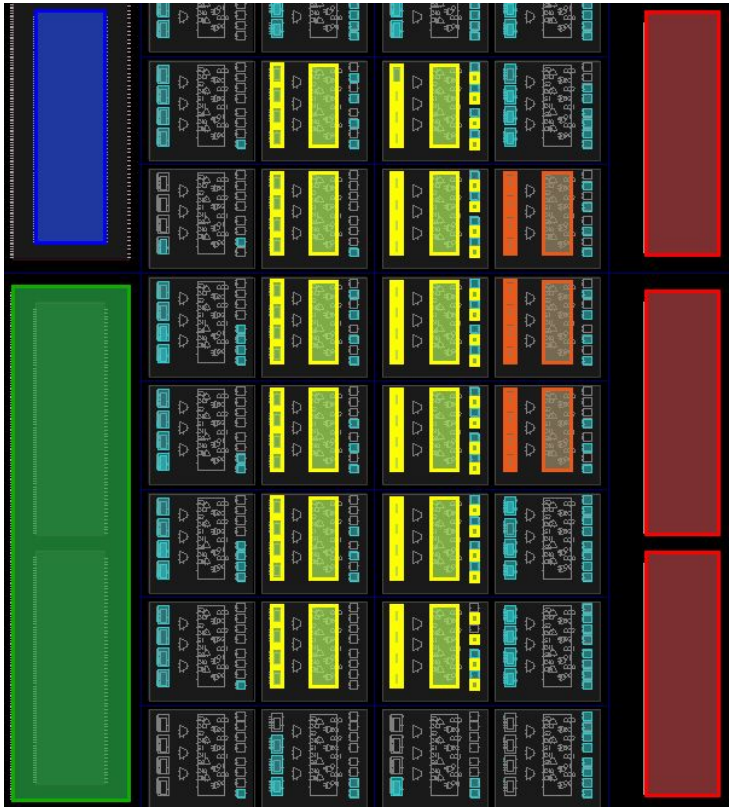
Yellow: Summing two filters

Blue: Fifo

Green: Block Ram

Orange: Downsample

Red: FIR filter



Second Convolution Layer

Green: Block Ram

Blue: MAC Controller

Yellow: Filter FIFO

Pink: Output FIFO

Red: Multiply Accumulator

Orange: Conv2 Controller

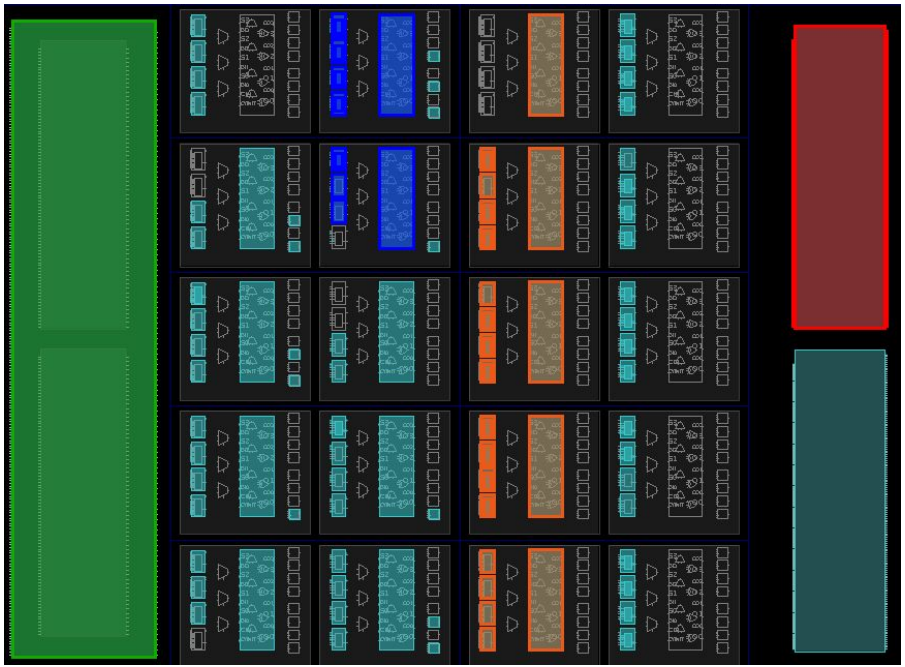
Fully Connected Layer

Green: Block RAM

Blue: Slope-Bias Loader 1

Orange: Slope-Bias Loader 2

Red: Multiply Accumulator

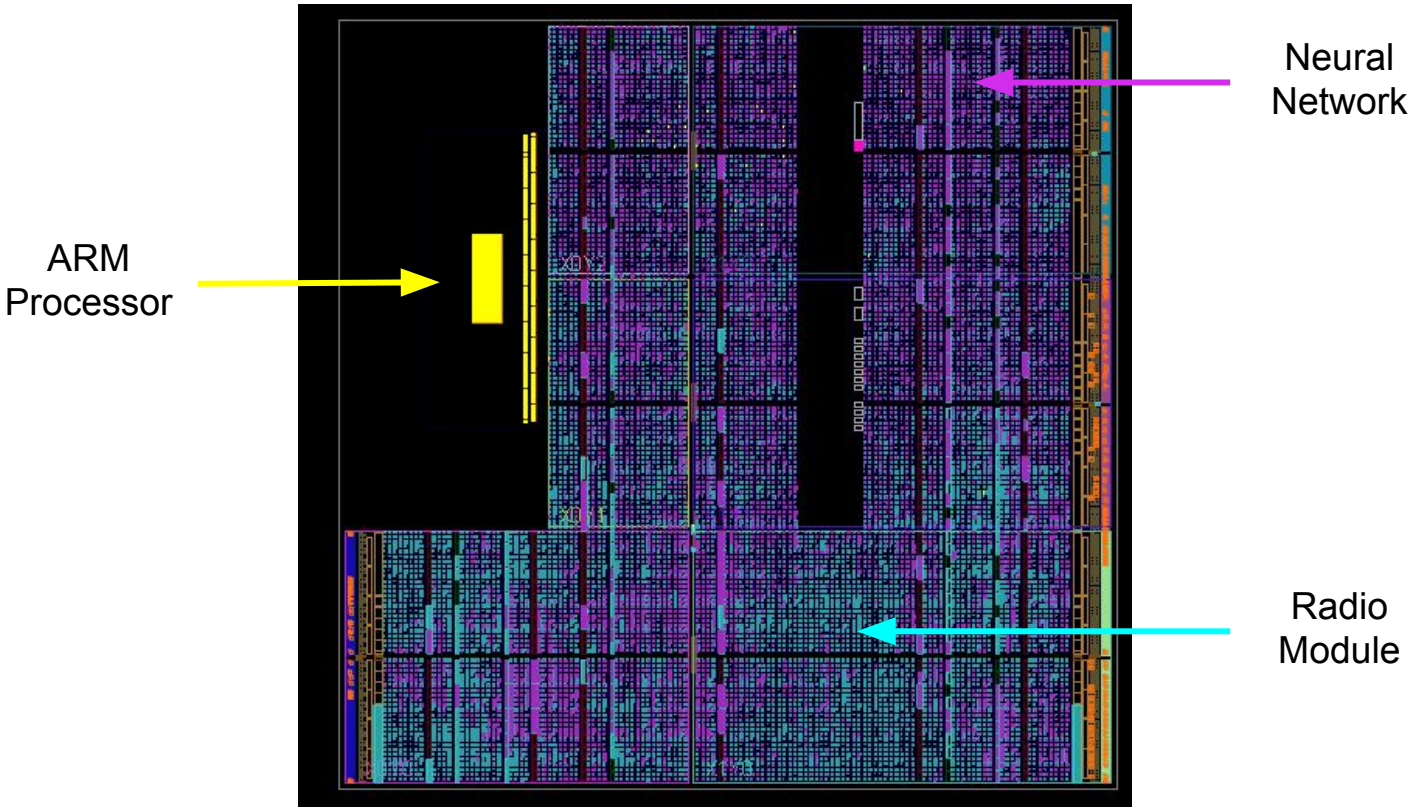


Lessons learnt

DO NOT TRUST VIVADO , NOT ITS PINS, NOT ITS DATAS

MIXED FILES-- NONO

System Layout



Solution: Pruning and Quantization

Hypothesis: A **similar accuracy** to O'Shea's model can be achieved with a **smaller CNN**

We investigated the accuracy of 40 combinations of the following three methods

1. Reduce the size of Convolution (Conv) Layers
2. Reduce the size of Fully Connected (FC) Layers
3. Reduce the number of Fully Connected Layers

Quantization: We also used **slope-bias scaling** to store 32-bit fixed point numbers as 8-bit integers.

FPGA Processing Pipeline

Step 2 : ARM Core preprocesses data

Neural Network Classifier (in digital logic)

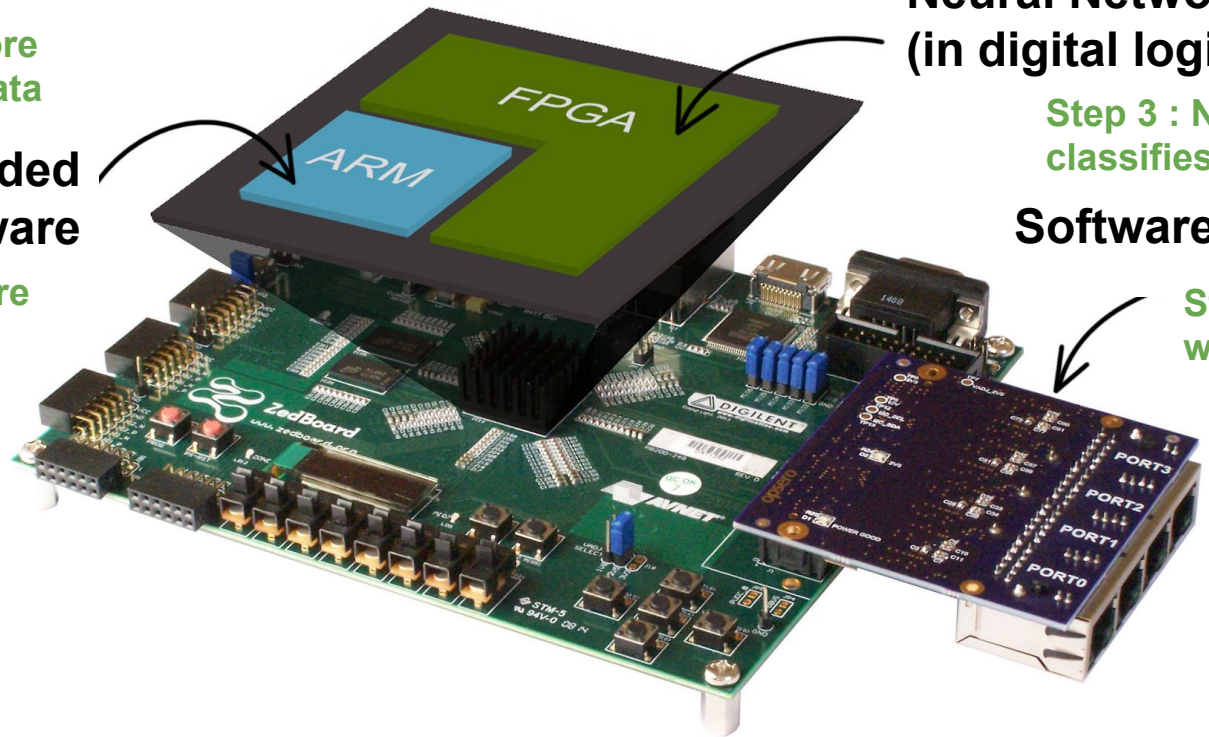
Step 3 : Neural network classifies signals

Embedded software

Software Defined Radio

Step 4 : ARM Core outputs results

Step 1 : Receives wireless signals



Hardware Resource Utilization

Resource	Total Available	CNN Model (% used)	Radio Module (% used)
Processing Element			
Look up Table (LUT)	53200	**31903 (60%)**	13206 (25%)
Flip flop (FF)	106400	**9597 (9%)**	25136 (24%)
DSP Slices	220	**128 (51%)**	69 (31%)
Memory Element			
LUTRAM	17400	**3903 (22%)**	210 (1%)
BRAM	140	**33.5 (24%)**	6 (4%)

Both radio module and CNN are successfully implemented within the FPGA

There is still room for developing other useful modules (e.g. ADCs, DDC)