A Fully Parallel DNN Implementation and its Application to Automatic Modulation Classification

Philip Leong | Computer Engineering Laboratory
School of Electrical and Information Engineering,
The University of Sydney
http://phwl.org/talks

THE UNIVERSITY OF SYDNEY

- Focuses on how to use parallelism to solve demanding problems
  - Novel architectures, applications and design techniques using VLSI, FPGA and parallel computing technology
- Research
  - Reconfigurable computing
  - Machine learning
  - Signal processing
- Collaborations
  - Xilinx, Intel, Exablaze
  - clustertech.com

Introduction

(Stephen Tridgell PhD work)

- Hard to make fully parallel implementations of a NN on contemporary FPGA due to size
- Fit entire DNN on FPGA by exploiting unstructured sparsity and the following techniques:
  1. Buffering of streaming inputs in a pipelined manner
  2. Ternary weights implemented as pruned adder trees
  3. Common subexpression elimination
  4. Digit serial arithmetic for throughput matching
  5. Sparsity control
  6. Incremental precision throughput matching
- Apply to automatic modulation classification (AMC), an integral component in intelligent radio

Overview

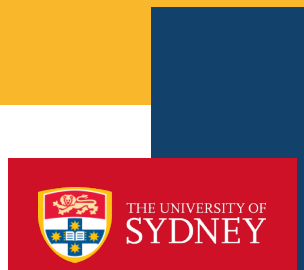
Optimising CNNs

Application to AMC

Overview

Optimising CNNs

Application to AMC



Network Studied

- VGG-7 network
- Ternary weights
- 16-bit activations
- Accept a single pixel every cycle (p=1)
  - W*W image takes W*W cycles

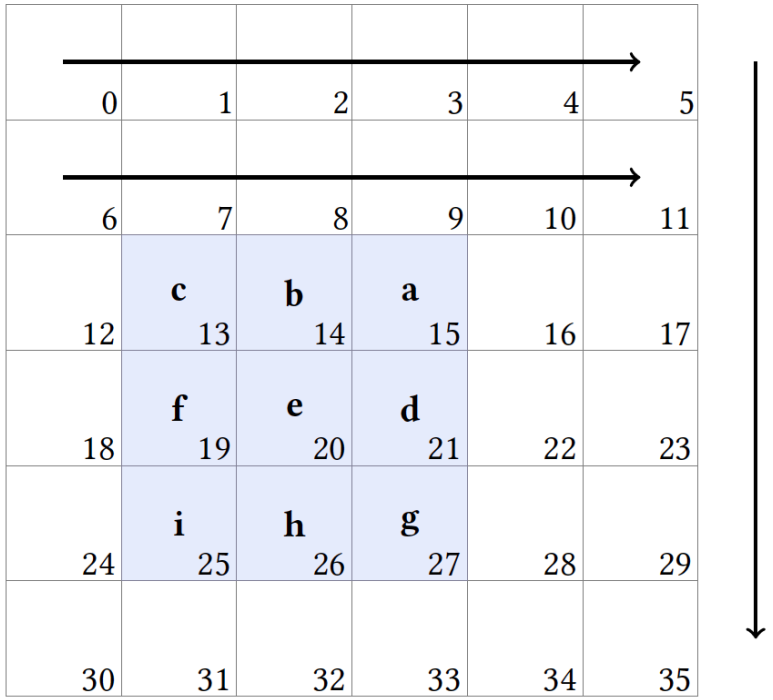
Layer	Num Mults	Num Mults	With Sparsity	With CSE
Conv1	32*32*3*3*3*64	1769472	716800	630784
Conv2	32*32*3*3*64*64	37748736	8637440	4653056
Conv3	16*16*3*3*64*128	18874368	4559616	2654720
Conv4	16*16*3*3*128*128	37748736	9396480	5213440
Conv5	8*8*3*3*128*256	18874368	4656768	2504640
Conv6	8*8*3*3*256*256	37748736	9356736	4731840
Dense	4096*128	524228	524228	1048456[1]
SM	128*10	1280	1280	2560[1]
Total	153289924	153 MMACs/Image	38 MMACs/Image	21 MOps/Image

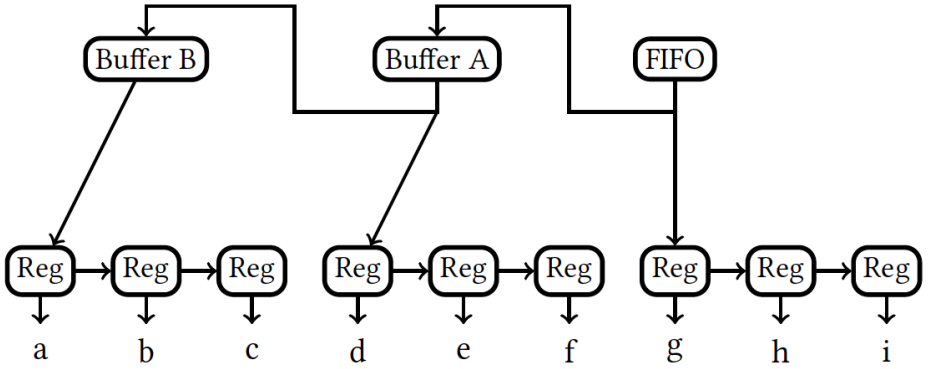
[1] Obtained by converting one MACs to two Ops

Operation	Image Size In	Channel In	Channel Out
Buffer	32x32	3	3
Conv	32x32	3	64
Scale and Shift	32x32	64	64
Buffer	32x32	64	64
Conv	32x32	64	64
Scale and Shift	32x32	64	64
Buffer	32x32	64	64
Max Pool	32x32	64	64
Buffer	16x16	64	64
Conv	16x16	64	128
Scale and Shift	16x16	128	128
Buffer	16x16	128	128
Conv	16x16	128	128
Scale and Shift	16x16	128	128
Buffer	16x16	128	128
Max Pool	16x16	128	128
Buffer	8x8	128	128
Conv	8x8	128	256
Scale and Shift	8x8	256	256
Buffer	8x8	256	256
Conv	8x8	256	256
Scale and Shift	8x8	256	256
Buffer	8x8	256	256
Max Pool	8x8	256	256
FIFO	4x4	256	256
MuxLayer	4x4	256	4096
Dense	1x1	4096	128
Scale and Shift	1x1	128	128
MuxLayer	1x1	128	128
Dense	1x1	128	10

1. Buffering of Streaming Inputs

Implement Pipelined 3x3 Convolution





Input FIFO outputs the pixel each cycle to both Buffer A and the first stage of a shift register.
Buffer A and Buffer B delay the output by the image width

2. Ternary Weights as Pruned Adder Trees

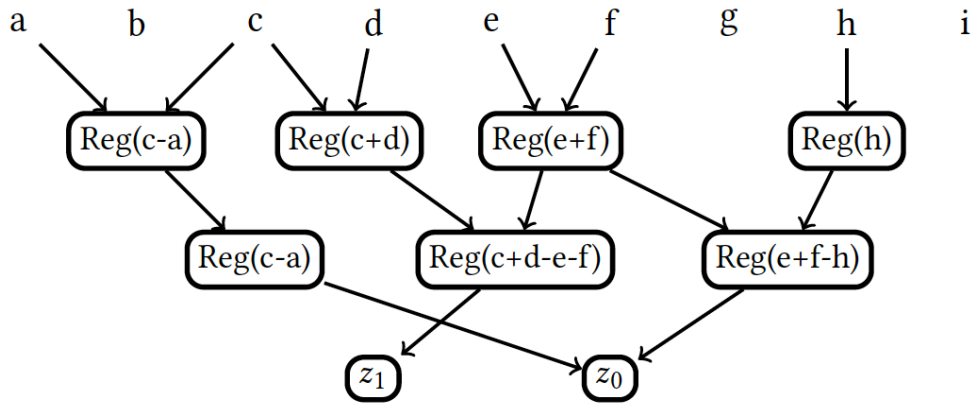
- Weights are ternary
  - So multiplication with ± 1 is either addition or subtraction
  - Since we have many multiplications with 0 matrix is sparse

$$\begin{matrix} a \times (-1) & b \times 0 & c \times 1 \\ d \times 0 & e \times 1 & f \times 1 \\ g \times 0 & h \times (-1) & i \times 0 \end{matrix}$$

3. Common Subexpression Elimination

- Weights are ternary
  - Reduces convolution to constructing adder tree
  - Subexpression merged to reduce implementation

$$\begin{array}{ccc} a \times (-1) & b \times 0 & c \times 1 \\ d \times 0 & e \times 1 & f \times 1 \\ g \times 0 & h \times (-1) & i \times 0 \end{array}$$



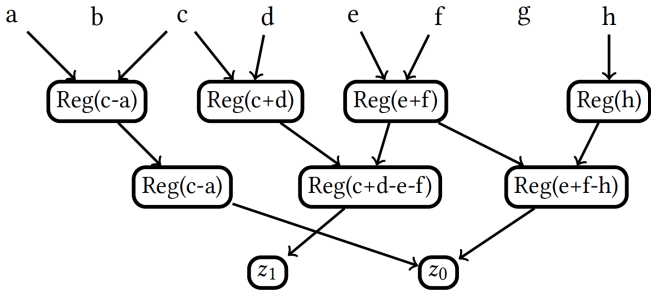
Computing $z_0 = c + e + f - (a + h)$ and $z_1 = c + d - e - f$

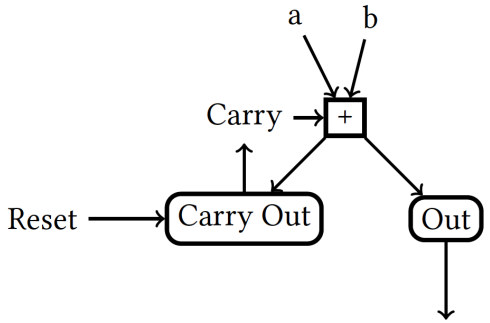
Improvement in using CSE

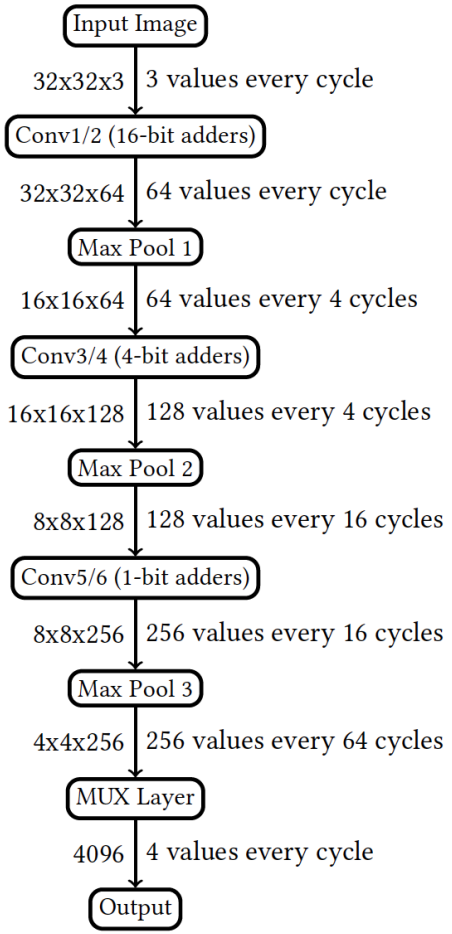
Layer	% decrease in Adds+Regs	% decrease in CLBs	%decrease in FFs
1	-29.0	-41.4	-48.4
2	-47.7	-32.6	-40.0
3	-42.3	-42.1	-39.6
4	-44.6	-44.6	-42.3
5	-45.8	-58.8	-45.8
6	-49.4	-60.8	-52.1

4. Digital Serial Arithmetic for Throughput Matching

- Used 16-bit fixed point
- Each layer followed by batch normalization with floating point scaling factor
- Suppose that for a given layer, p pixels arrive at the same time
  - For $p \geq 1$ have p adder trees in parallel
  - For $p < 1$ word or bit-serial adders can match input rate with hardware resources
  - 4-bit digit serial has 1/4 area
  - 1-bit bit serial has 1/16 area
- Avoids idle adders







5. Sparsity Control

- CIFAR10 dataset
- Weights are compared with threshold
  - 0 if less than threshold, $s(\pm 1)$ otherwise $\Delta^* \approx \epsilon \cdot E(|W|)$ r)
- We introduce the idea of changing ϵ to control sparsity

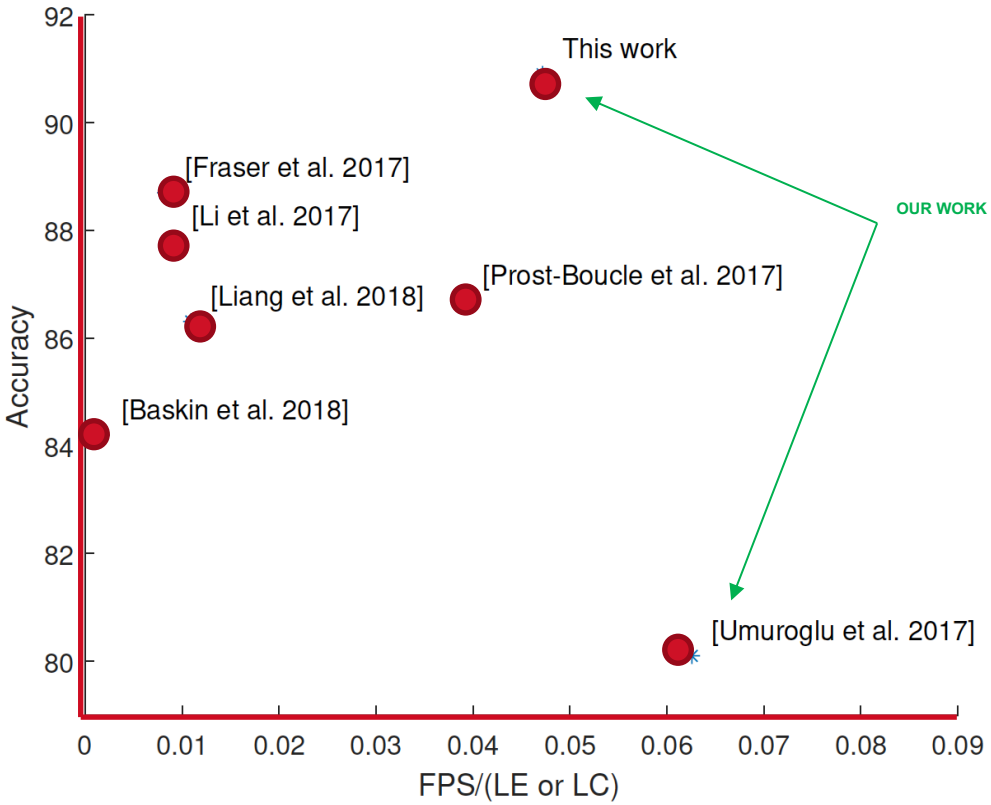
TNN Type	ϵ	Sparsity (%)	Accuracy
Graham [Graham 2014] (Floating Point)	-	-	96.53%
Li et al. [Li et al. 2016], full-size	0.7	$\approx$ 48	93.1%
Half-size	0.7	$\approx$ 47	91.4%
Half-size	0.8	$\approx$ 52	91.9%
Half-size	1.0	$\approx$ 61	91.7%
Half-size	1.2	$\approx$ 69	91.9%
Half-size	1.4	$\approx$ 76	90.9%
Half-size	1.6	$\approx$ 82	90.3%
Half-size	1.8	$\approx$ 87	90.6%

THE UNIVERSITY OF SYDNEY

Breakdown of Layer Sparsity

Layer Type	Input Image Size	Num Filters	ϵ	Sparsity
Conv2D	$32 \times 32 \times 3$	64	0.7	54.7%
Conv2D	$32 \times 32 \times 64$	64	1.4	76.9%
Max Pool	$32 \times 32 \times 64$	64	-	-
Conv2D	$16 \times 16 \times 64$	128	1.4	76.1%
Conv2D	$16 \times 16 \times 128$	128	1.4	75.3%
Max Pool	$16 \times 16 \times 128$	128	-	-
Conv2D	$8 \times 8 \times 128$	256	1.4	75.8%
Conv2D	$8 \times 8 \times 256$	256	1.4	75.4%
Max Pool	$8 \times 8 \times 256$	256	-	-
Dense	4096	128	1.0	76.2%
Softmax	128	10	1.0	58.4%

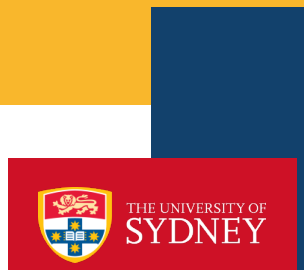
CIFAR10 Accuracy vs Speed (FPGA Implementations)



Overview

Optimising CNNs

Application to AMC



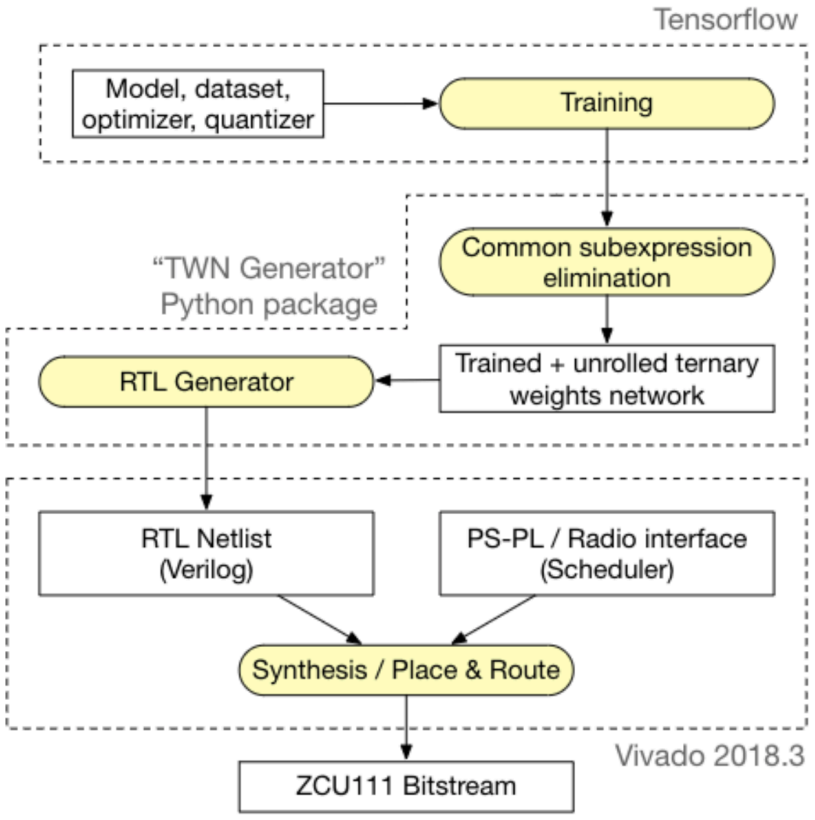
THE UNIVERSITY OF SYDNEY

Automatic Modulation Classification

- Identify modulation type from raw radio signal
  - A step towards general problem of interpreting RF scenes from raw signals is a fertile research problem
- Reconfigurable computing an excellent solution for this problem
  - FPGA enable integration of radio and machine learning in single device
  - Latency, size, weight and power are crucial in applications

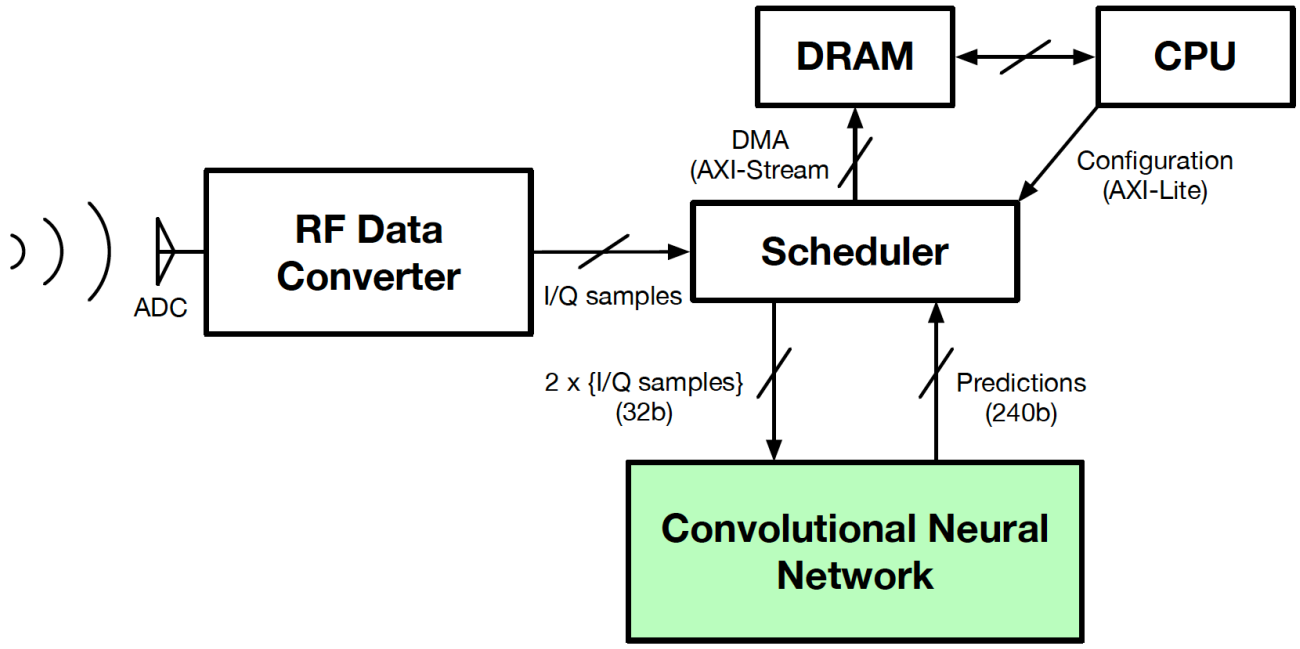
Implementation

- System implemented on ZCU111 RFSoC
  - 8x 12-bit 4.096GSPS ADCs
  - 8x 14-bit 6.554GSPS DACs
  - Arm Cortex-A53
  - Arm Cortex-R5
- Open Source Verilog generator
  - https://github.com/da-steve101/twn_generator



FPGA Implementation

› Ternary Modulation classifier: 488K class/s, 8us latency



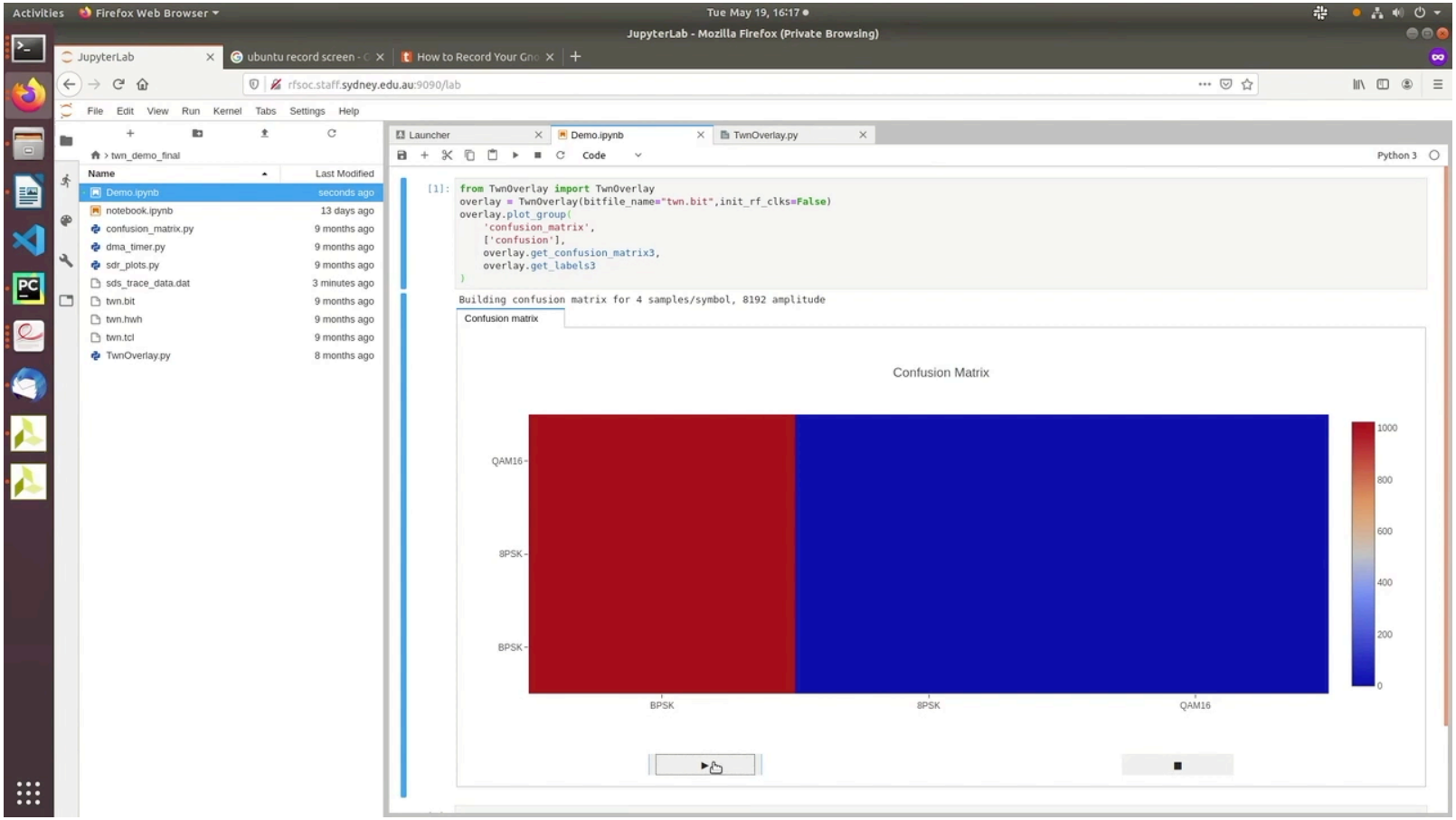
6. Incremental Precision Throughput Matching

- Use incremental precision activations instead of 16 bit
  - Adjust precision to match the throughput
  - Same area as ternary activations
  - Almost 5% accuracy gain

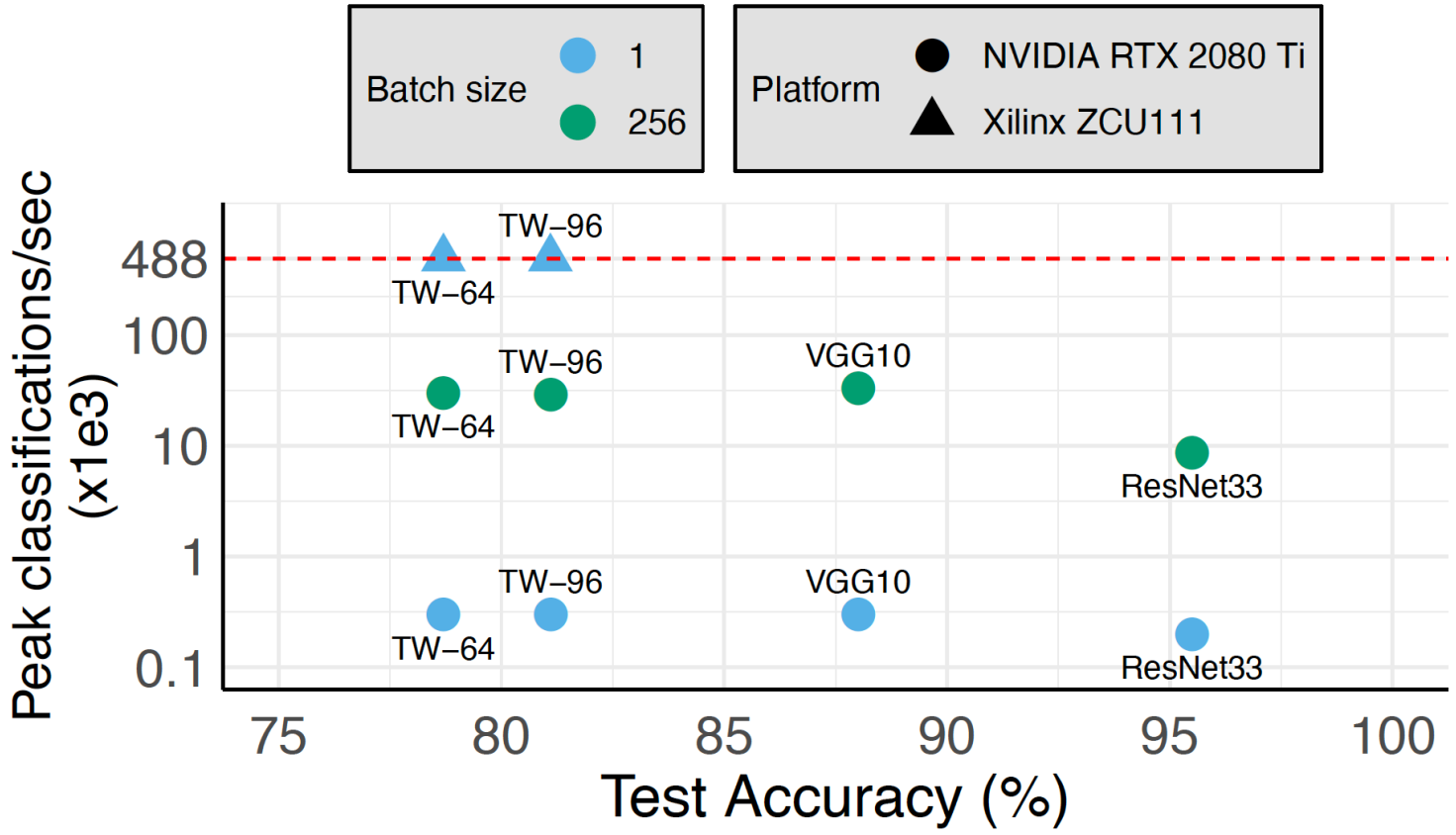
Model	TW-64	TW-96	TW-BA-128	TW-INCRA-128
CLBs	28k (53.5%)	47k (89.3%)	43k (80.7%)	42k (80.2%)
LUTs	124k (29.1%)	232k (54.7%)	234k (55.1%)	211k (49.6%)
FFs	217k (25.5%)	369k (43.4%)	333k (39.2%)	324k (38.1%)
BRAMs	524 (48.5%)	524 (48.5%)	523 (48.4%)	512.2 (48.3%)
DSPs	1496 (35%)	1207 (28.3%)	1408 (33.0%)	1407 (32.9%)
Accr	78.7	81.1	75.9	80.2

Video Demonstration

QAM16/8PSK/BPSK



O'Shea at al, RadioML Dataset



Conclusion

- Presented an optimized network for AMC which
  - Applies common subexpression elimination and digit serial arithmetic to a fully unrolled ternary network
  - Integrates the entire design on a single chip for a low-latency batch size 1 implementation
- These serve to achieve a level of performance higher than previously reported
- Challenge of achieving state of the art accuracy remains
- As FPGAs become larger, we believe these techniques will become more common

THE UNIVERSITY OF SYDNEY

References

[1] Stephen Tridgell, Martin Kumm, Martin Hardieck, David Boland, Duncan Moss, Peter Zipf, and Philip H. W. Leong. Unrolling ternary neural networks. *ACM Trans. Reconfigurable Technol. Syst.*, 12(4):22:1–22:23, October 2019. URL: ternary_trets19.pdf, doi:10.1145/3359983.

[2] Stephen Tridgell, David Boland, Philip HW Leong, Ryan Kastner, Alireza Khodamoradi, and Siddhartha. Real-time automatic modulation classification using rfsoc. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2020, New Orleans, LA, USA, May 18-22, 2020*, 82–89. IEEE, 2020. URL: https://doi.org/10.1109/IPDPSW50202.2020.00021, doi:10.1109/IPDPSW50202.2020.00021.