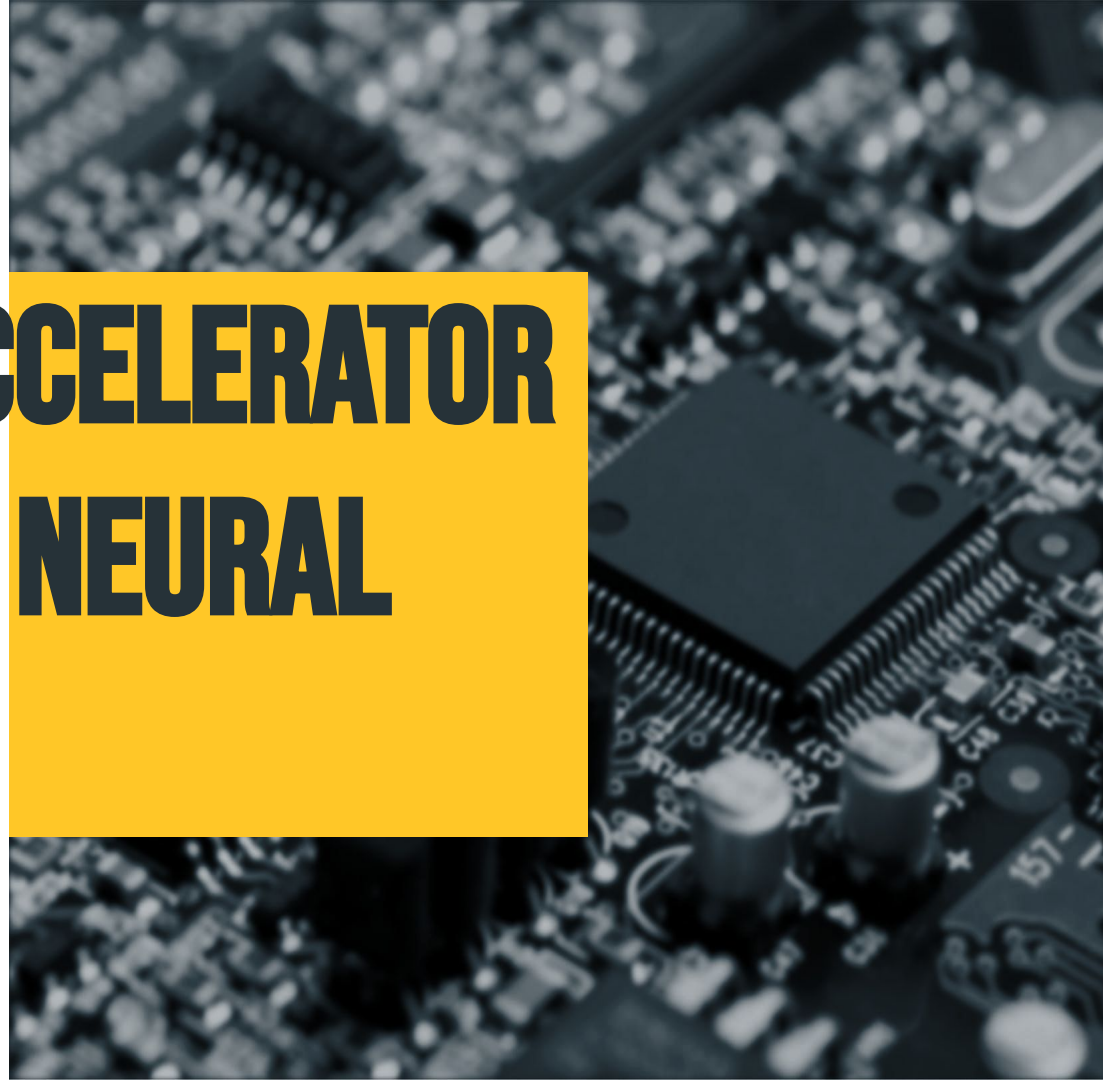
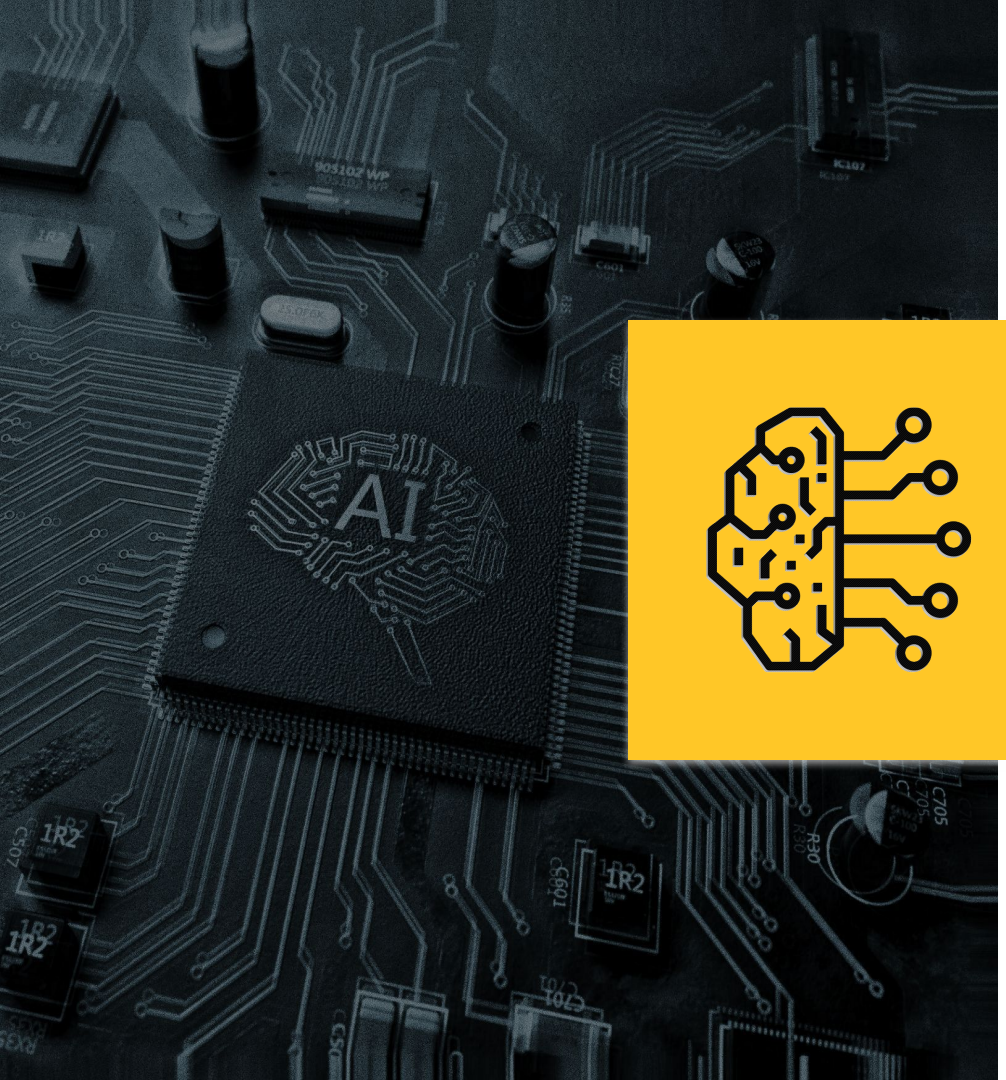
HARDWARE ACCELERATOR MODULES FOR NEURAL NETWORKS



GROUP 10

Harinandan Nair	B180718EC
Hemanth S	B180550EC
Emmanuel Thomas Devasia	B180711EC
Ellickal Allen Appukuttan	B180702EC

PROJECT GUIDE: Dr. Dhanaraj K J



INTRODUCTION

In recent years, AI/ML has become a major part of our daily lives and the technology around us. With this increasing prevalence, more and more complex problems are being solved using AI requiring equally complex models, and increasing number of parameters.

MOTIVATION

With an enormous amount of computation needed for machine learning algorithms such as CNN/DNN , the Von Neumann bottleneck has gained immense significance in recent years

To make these solutions widely available there is a requirement for efficient computational hardware for neural networks.

PROBLEM STATEMENT

Our primary aim was to improve performance/energy efficiency of neural networks, via the use of dedicated hardware

It was found that most significant gains can be attained using analog components with dedicated functions.

The project focuses on using In-Memory computing to reduce the impact of Von-Neumann bottleneck on neural networks, and implement efficient pathways for the most frequent Computations such as Matrix Multiplication

PAST WORKS

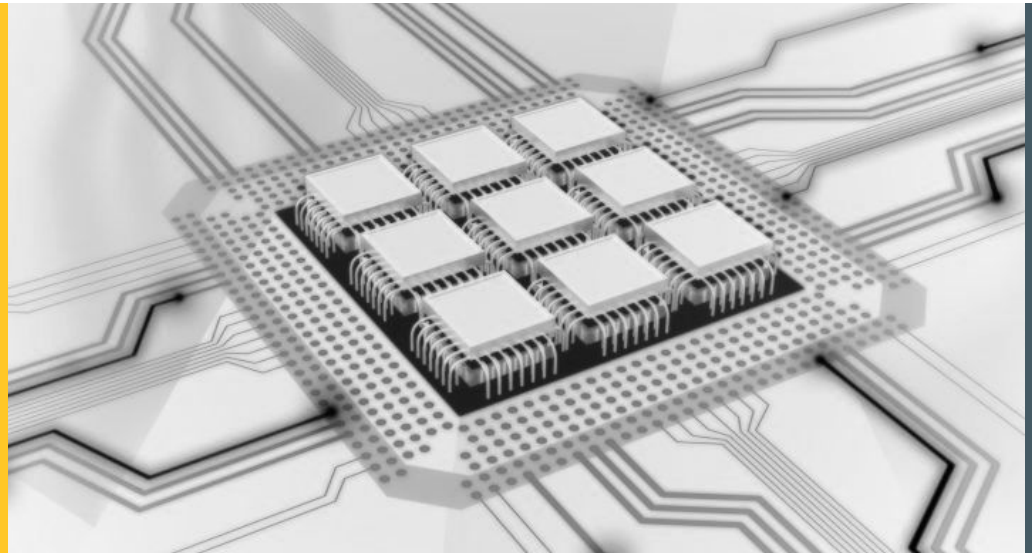
01 Literature Review

02 Profiling of LeNet Architecture

03 Implementation of XNOR-SRAM bit cell

SRAM BASED IMC

- Use of binary weights and inputs reduce computation
- Binary-weight DNN and CNN opens possibility form SRAM based IMC
- Activation value applied as word line voltages
- Weights are stored in SRAM bit cell
- MAC computation simplified to bitwise operations



01

XNOR-SRAM BASED IMC

XNOR-SRAM CELL

- Operates in two modes: Memory mode and XNOR mode
- In the memory mode, it performs row-by-row digital read/write as regular SRAM
- In the XNOR mode, it performs in-memory XAC computation with all rows asserted simultaneously
- The read worldline (RWL) driver translates each ternary/binary input activation to four RWLs according to Table 1

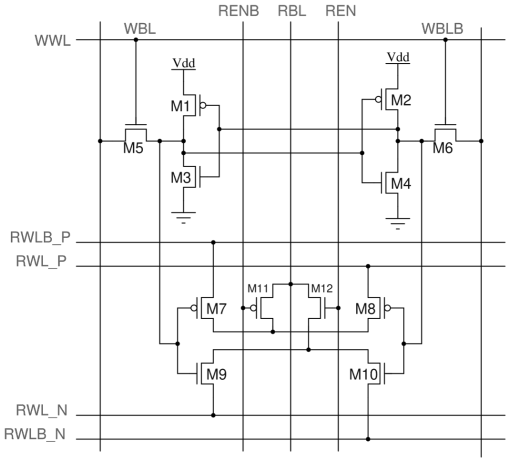


Fig 1: *XNOR-SRAM bitcell*

Input / RWL	+1	-1	0 odd row	0 even row
RWL_P	VDD	0	0	VDD
RWLB_P	0	VDD	0	VDD
RWL_N	VDD	0	VDD	0
RWLB_N	0	VDD	VDD	0

Table 1: *RWL driver logic*

XNOR-SRAM ARRAY

- Implemented a 16 x 16 XNOR-SRAM array using LTspice
- Verified the array operations with XAC value.

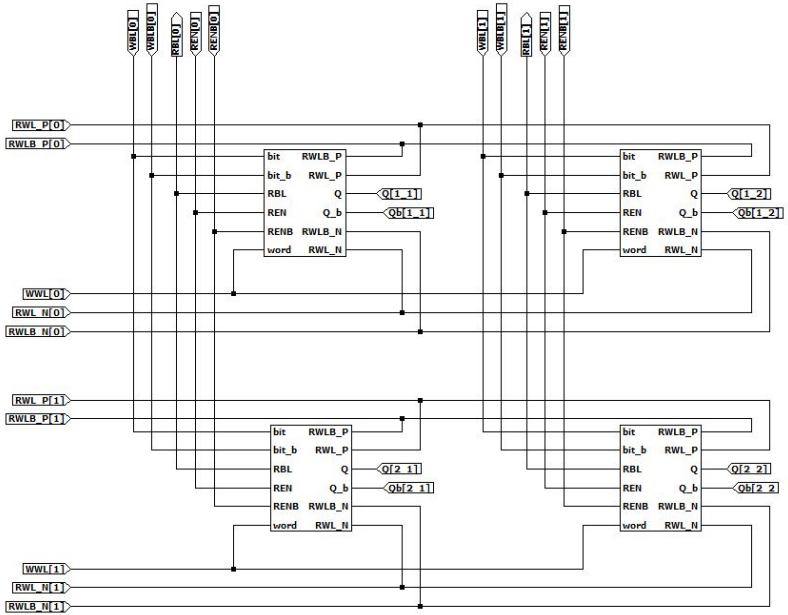


Fig 2: *2 x 2 array portion of XNOR-SRAM Array implemented in LTspice*

IMC OPERATION

- Bit cell produces XNOR outputs by different combinations of pull-up/pull-down paths
- A resistive voltage divider is formed in a column, RBL is the output voltage
- The relationship between XAC Value and RBL voltage:

$$N = u + d$$
$$XAC = u - d$$
$$V_{RBL} = \frac{XAC+N}{2N}$$

- *u* is the number of PU paths and *d* is the number of PD paths

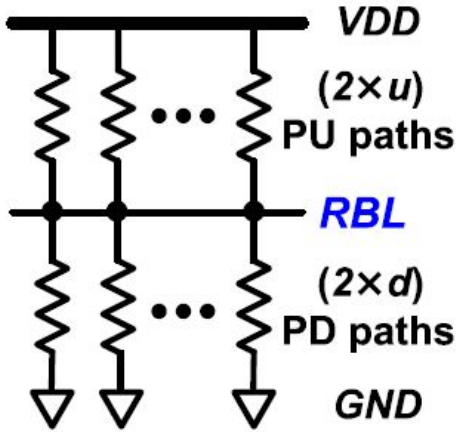


Fig 3: *PU/PD paths for V_{RBL} [2]*

ANALYSIS AFTER IMC

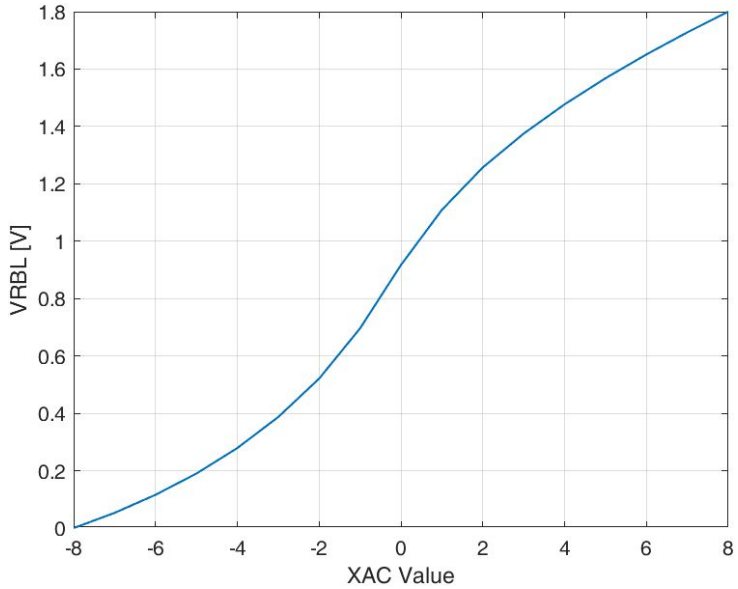


Fig 4: *XAC value mapped to V_{RBL}*

- V_{RBL} is a symmetric and monotonic function of XAC value
- The ADC digitize the analog V_{RBL}
- Flash ADC shared among all the columns
- XAC value high is concentrated around zero (*data distribution from MLP for MNIST*)
- Reference voltage for ADC is obtained by mapping the XAC value to the V_{RBL}

ANALOG-TO-DIGITAL CONVERTOR

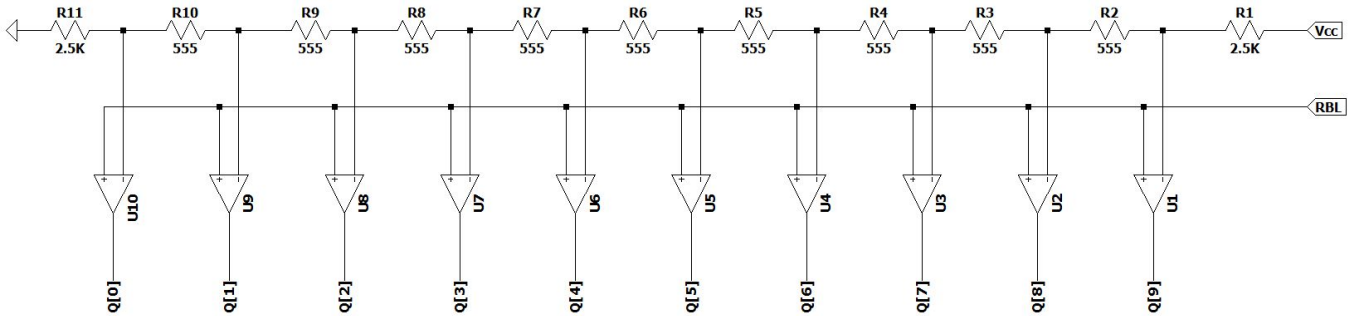


Fig 5: *Flash ADC designed in LTspice*

- 11 quantization level
- Output of the SRAM macro are processed by a digital ALU

POWER CONSUMPTION OF ARRAY

- Measured the power consumption of XNOR-SRAM array
- Power consumption depend upon the XAC value
- The input data that corresponds to XAC value near zero consumes more power

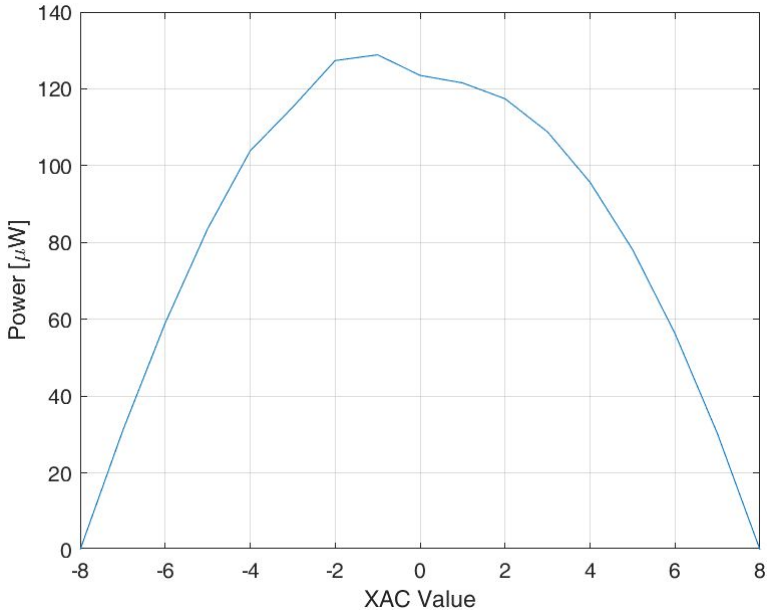
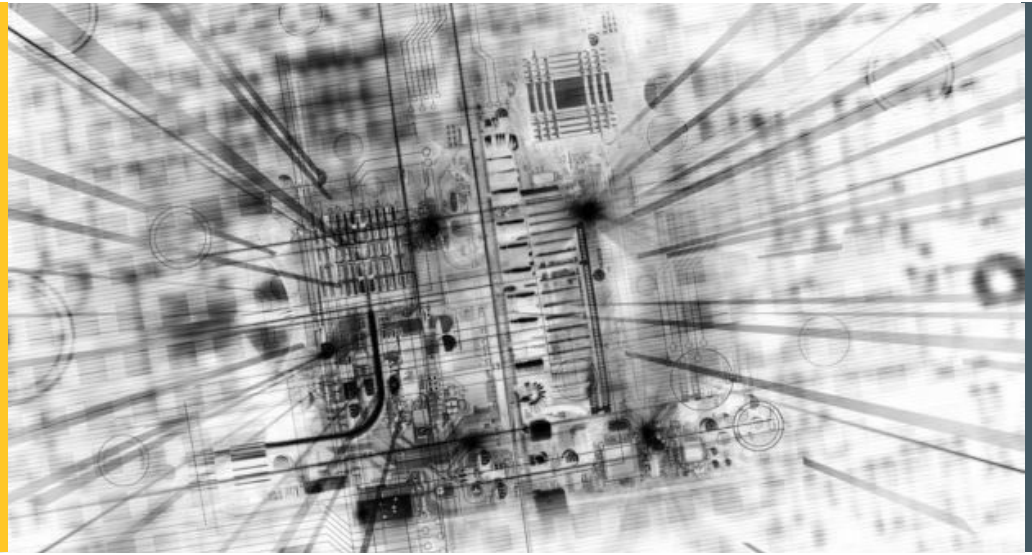


Fig 6: *XAC value mapped to* V_{RBL}



02

ALTERNATIVE SRAM BASED IMCs

CUSTOM 8T SRAM CELL FOR BNN

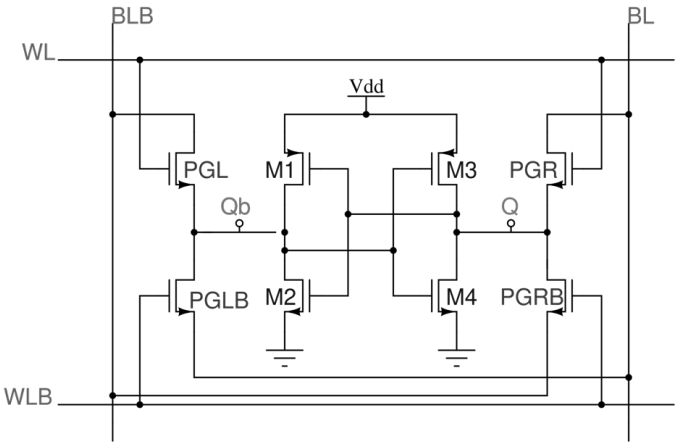


Fig 7: *Custom 8T SRAM cell implemented in LTspice*

- The binary neuron activation of 0/1 is represented by WL of 0/1 and synaptic weight is stored in Q and Qb
- To evaluate the XNOR function, BL and BLB will be charged by current (iC) or discharged by current (iD) depending on the input and weight pattern,
- With binary activation function, the differential comparison result of total currents of BL and BLB determines the output polarity.

Weight			Input			XNOR (Product)		
Value	WL	WLB	Value	Q	Qb	Value	BL	BLB
+1	1	0	+1	1	0	+1	iC	iD
+1	1	0	-1	0	1	-1	iD	iC
-1	0	1	+1	1	0	-1	iD	iC
-1	0	1	-1	0	1	+1	iC	iD

Table 2: *Coding scheme of inputs and weights*

SINGLE ENDED 8T-SRAM CELL

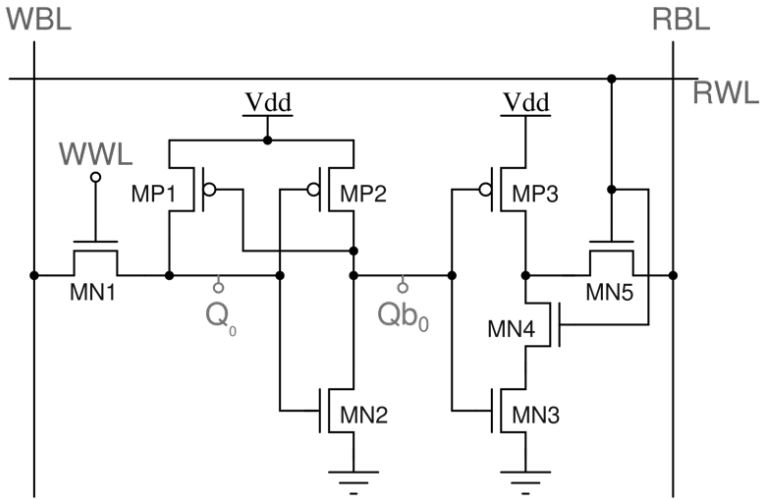


Fig 8: *Single ended 8T-SRAM cell implemented in LTspice*

- For BNN calculation, we allocate 0 or 1 as a binary input, which can be implemented by applying a positive pulse through RWL.
- The 0 state of the input is the basic low state, and input 1 can be expressed by applying a positive short pulse.
- The weight value can be stored as

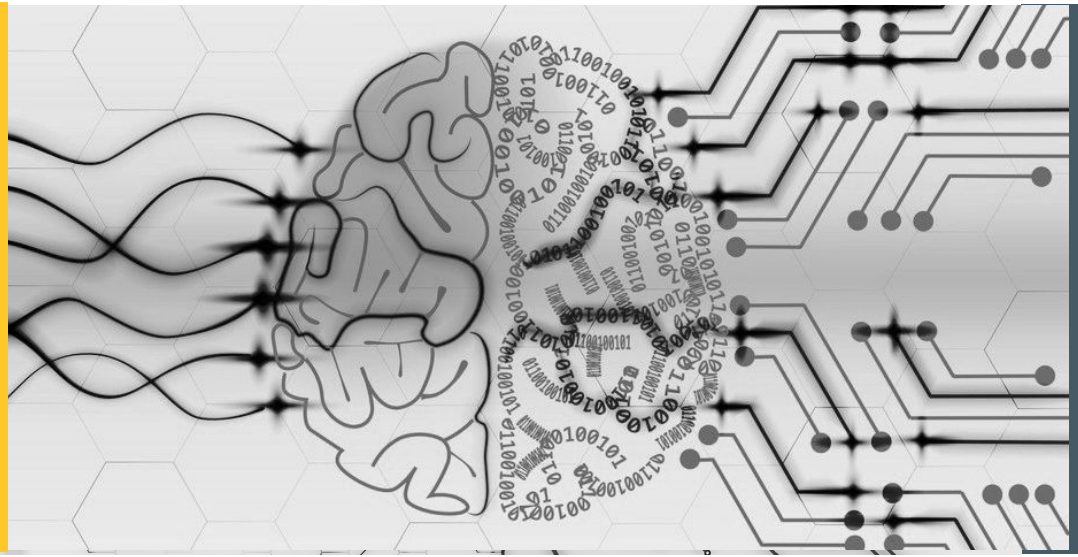
  −1 : Q = L, Qb = H
  +1 : Q = H, Qb = L

  through the write unit of SRAM cell.

POWER CONSUMPTION COMPARISON

Average power consumed by each SRAM cells where observed using LTspice

1. XNOR-SRAM : 19.8μW
2. CUSTOM 8T-SRAM : 55.4μW
3. SINGLE ENDED 8T- SRAM : 31.1μW



03

BINARIZING THE NEURAL NETWORK

DOT PRODUCT

- Parameters of the neural networks like input and weights are binarized
- Operations like dot product and convolution are modified accordingly to reduce data loss after binarization [4].
- The dot product is approximated with the following equation.

$$X^T W \approx \beta H^T \alpha B$$

$$H = sign(X)$$

$$\beta = \frac{1}{n}||X||_l$$

$$B = sign(W)$$

$$\alpha = \frac{1}{n}||W||_l$$

CONVOLUTION OPERATION

- Convolution operation is approximated with the following equation.

$$I * W \approx (sign(I) \circledast sign(W)) \odot K\alpha$$

- $\circledast$ represents convolution operation using XNOR and bitcount operations
- $\odot$ represents element wise product. Also,

$$K = A * k,\ where\ \forall i, j; k_{ij} = \frac{1}{w \times h}$$

$$A = \frac{\sum [I_{:,:,i}]}{c}$$

BINARIZING WEIGHTS AND INPUTS

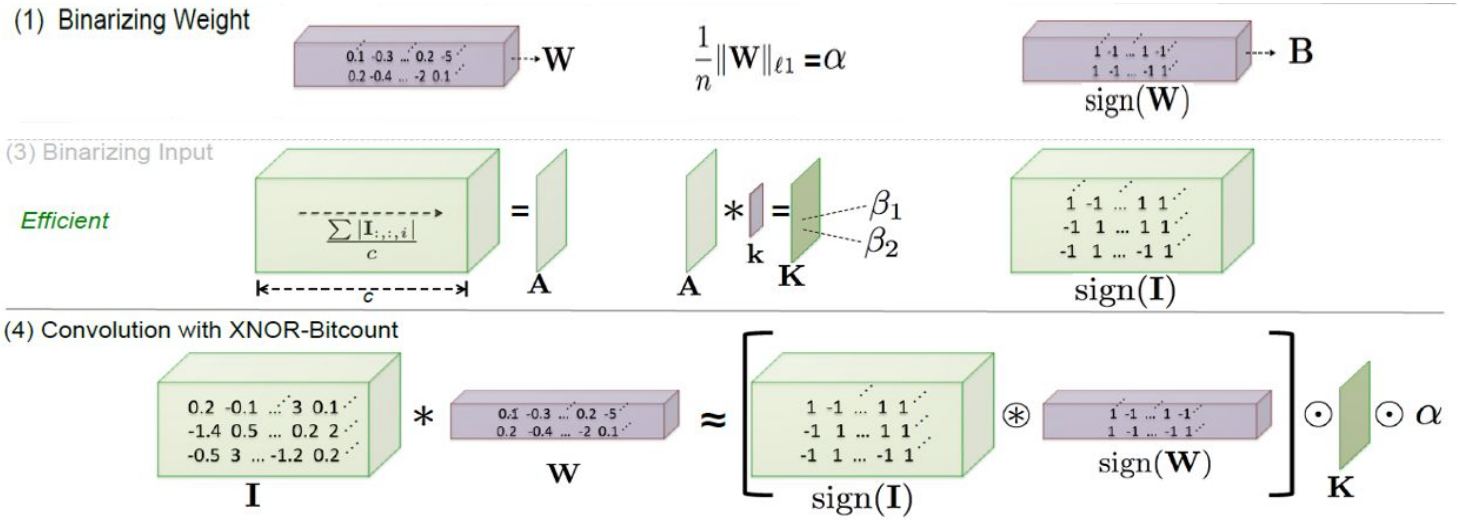


Fig 9: *Approximating convolution using binary operations [6]*

04

MAPPING NEURAL NETWORK TO XNOR-SRAM MACRO

MAPPING CONVOLUTION LAYER

- A mapping scheme where the same location pixel (x, y) of the kernel from all the kernels for different input/output feature maps will be stored in the same XNOR-SRAM array.

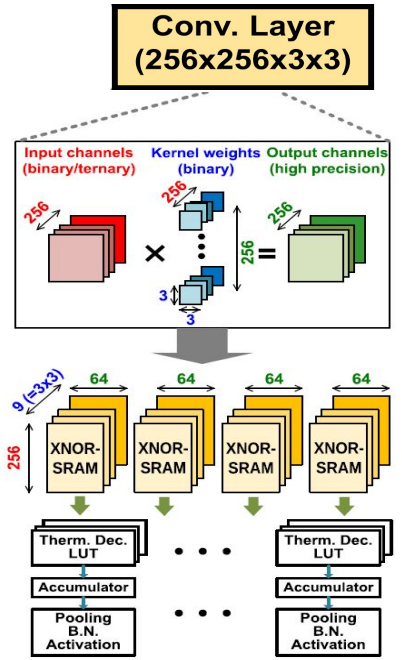


Fig 10: *Mapping Convolution layer to XNOR-SRAM array [3]*

MAPPING FC LAYER

- Fully connected layer of DNN is mapped directly to XNOR-SRAM rows.
- For the fully connected layers whose size is larger than 256 × 64, we break the large weight matrix into a number of small submatrices

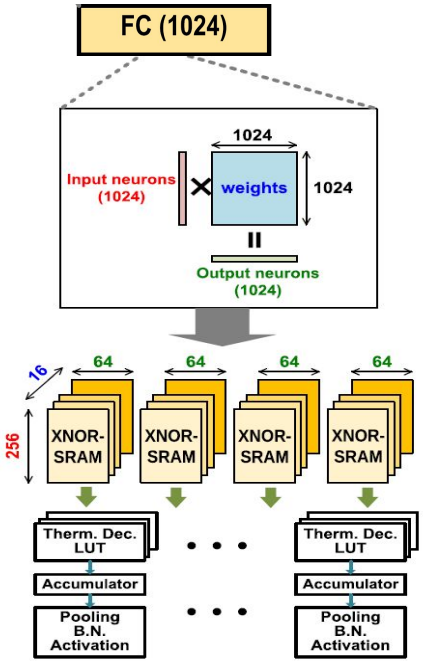
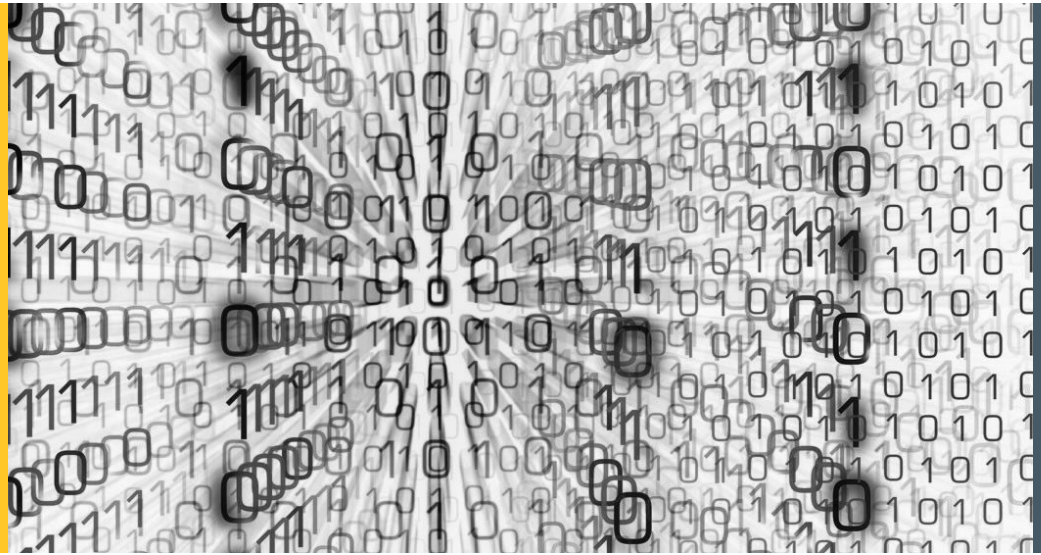


Fig 11: *Mapping FC layer onto XNOR SRAM array [3]*



05

PRACTICAL IMPLEMENTATION OF MATRIX MULTIPLICATION CIRCUITS

MICROKERNEL

- Hardware for arbitrarily large matrices is not a scalable approach
- Fixed size microkernels required
- A single 4 x 4 Matrix Multiplication can be improved using IMC Circuits
- Algorithm mentioned in [7] enables us to compute all elements of a 4 x 4 matrix multiplication in a fast and energy efficient manner

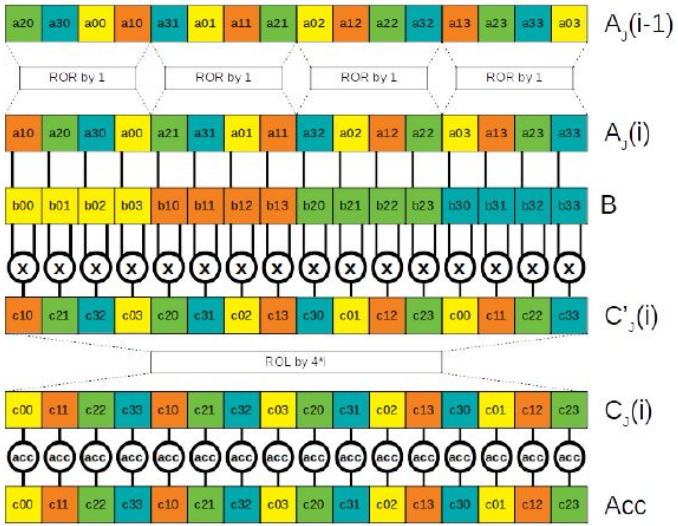
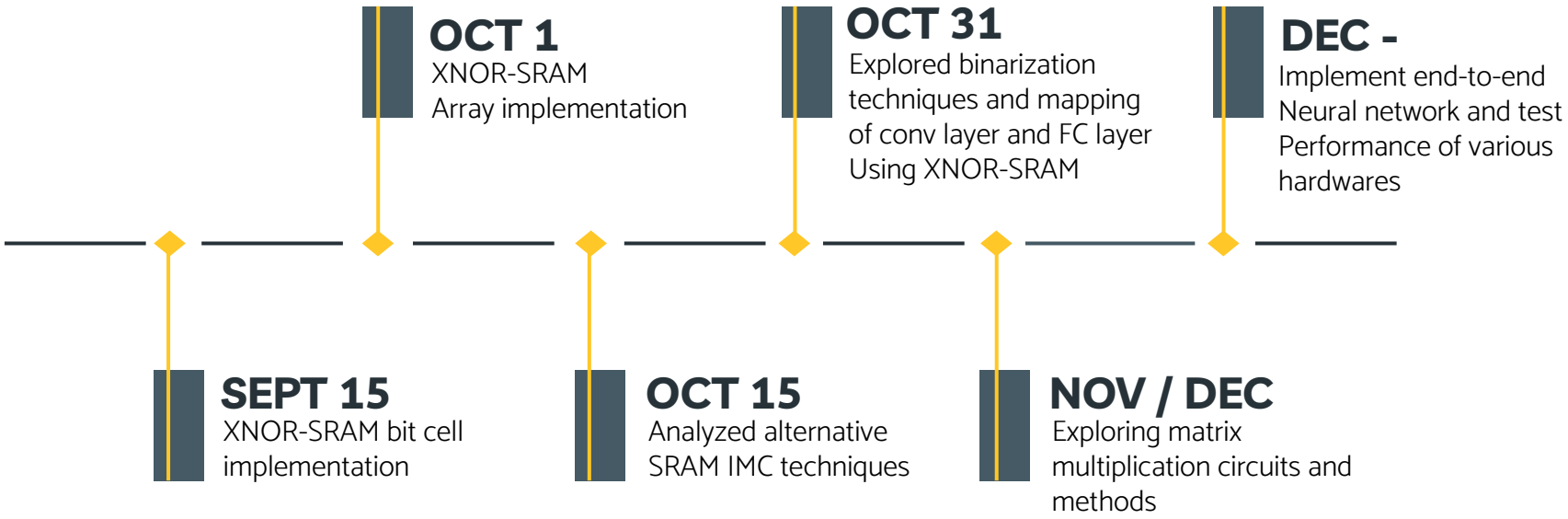


Fig 12: *Proposed vectorization scheme for 4 × 4 microkernel in [7]*

FUTURE PLAN

- Implement various analog components using LTSpice
- Implement an end-to-end Neural Network in MATLAB
- Replace certain functions with their analog netlists developed in LTSpice and benchmark performance of different alternatives

TIMELINE


SEPT 15
XNOR-SRAM bit cell implementation
OCT 1
XNOR-SRAM Array implementation
OCT 15
Analyzed alternative SRAM IMC techniques
OCT 31
Explored binarization techniques and mapping of conv layer and FC layer Using XNOR-SRAM
NOV / DEC
Exploring matrix multiplication circuits and methods
DEC -
Implement end-to-end Neural network and test Performance of various hardwares

REFERENCES

[1] J. von Neumann, "First draft of a report on the EDVAC," in *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27-75, 1993, doi: 10.1109/85.238389.

[2] S. Yin, Z. Jiang, M. Kim, T. Gupta, M. Seok and J. Seo, "Vesti: Energy-Efficient in-Memory Computing Accelerator for Deep Neural Networks," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 48-61, Jan. 2020, doi: 10.1109/TVLSI.2019.2940649.

[3] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, “Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

REFERENCES

[4] H. Jiang, R. Liu, and S. Yu, “8t xnor-sram based parallel compute in-memory for deep neural network accelerator,” in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 257–260.

[5] Y. Kim, S. Li, N. Yadav, and K. K. Choi, “A novel ultra-low power 8t sram-based compute-in-memory design for binary neural networks,” *Electronics*, vol. 10, no. 17, 2021. [Online]. Available:https://www.mdpi.com/2079-9292/10/17/2181

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” pp.525–542, 2016.

[7] K. MAMBU, H.-P. Charles, and M. Kooli, “Efficient 8-bit matrix multiplication on Computational SRAM architecture,” in *DATE 2020 ; Computing-in-Memory (CiM) Workshop*, Grenoble, France, Mar. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02934838

THANK YOU