

# **Algorithm-Hardware Co-design for Low-Power Smart Home AI Devices**

BY

NIKOLAI ILIEV

B.Sc., Northwestern Univ., 1986

M.Sc., University of California Irvine, 1988

THESIS

Submitted as partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Electrical and Computer Engineering  
in the Graduate College of the  
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:

Amit Trivedi, Chair and Advisor

Igor Paprotny

Wenjing Rao

Vitali Metlushko

Zhao Zhang

Arun Subramanian, Mechanical and Industrial Engineering

Copyright by

Nikolai Iliev

2020

*to my parents and to my family*

## ACKNOWLEDGMENTS

I'm indebted to many people for their continuing support leading to this dissertation. First, I would like to thank my adviser, Amit Trivedi, for his support and trust. His supportive guidance helped me navigate through the problems in neuromorphic systems and his trust gave me enough freedom to enjoy exploring the field. I would also like to thank Igor Paprotny for introducing me to the field of Smart-Dust sensor networks and their self-localization and implementation problems. I enjoyed being exposed to his insights and benefited from our discussions. I would also like to thank Wenjing Rao and Zhao Zhang for commenting on my research and providing valuable feedback on different VLSI architectural topics.

Being a member of the AEON lab at the University of Illinois at Chicago (UIC) was a great experience for me. The collaborations and discussions over the years helped me grow, personally and intellectually. I thank my co-authors, Alberto Gianelli, Shama Nasrin, Ahish Shylendra, and Mahdi Salarian, for their contributions in the projects. Many of the ideas in our work emerged from our discussions and teamwork. Also many thanks go to my labmates : Abhinaya Ganesh, Edoardo Roba, Andrea Ciccardi, and Priyesh Shukla.

To my parents, Dr. Kirilka Petrova Chureva-Ilieva, MD Ph.D. 1962 Medical Academy Sofia, Bulgaria, and Yuli Nikolov Iliev, PE M.S. 1959 Civil and Mech. Eng., MEI Technical Univ. Sofia, Bulgaria, thank you for all your support and encouragement. To Ivan Petrov Churev, mathematician, Ph.D. 1980, Institute of Mathematics BAN, Shumen, Bulgaria, thank you for sparking my interest in advanced mathematics. To my wife Magdalena and son Johnny, thank you for putting up with my endless theoretical discussions at home and for making it all possible.

## **ACKNOWLEDGMENTS (Continued)**

NI

## **PREFACE**

This dissertation is an original intellectual product of the author, N. Iliev. All of the work presented here was conducted in the AEON Lab at the University of Illinois at Chicago.

The main theme in this work is hardware algorithm acceleration for sensor-equipped embedded hardware/software System-on-Chip, SoC, as found in indoor mobile platforms such as indoor robotic assistants. The accelerators are designed to assist the SoC in platform self-localization in indoor space, in human voice control of the platform, and in neural network processing of data acquired by the platform's sensors.

The project has been partially supported by UIC College of Engineering.

The results of these works have previously appeared (or is appearing) as journal and conference publications (Iliev and Trivedi, 2019) IEEE Sensors Letters, (Iliev et al., 2019) IEEE Embedded Systems Letters, (Iliev and Trivedi, 2017) IEEE International Conference on Computer Design, ICCD, (Iliev and Paprotny, 2015) IEEE Sensors Journal, (Gianelli et al., 2019) IEEE Symposium on Low-Power and High-Speed Chips and Systems, (Salarian et al., 2018) IEEE Transactions on Multimedia, and (Salarian et al., 2016) IEEE International Symposium on Multimedia (ISM).

The copyright permissions for reusing the published materials have been presented in Appendix A.

Nick Iliev  
April 15, 2020

## CONTRIBUTION OF AUTHORS

A version of Chapter 4 and Chapter 5 has been published in IEEE ICCD 2017 Conference and in 2019 IEEE Sensors Letters journal. Amit Trivedi, my adviser, was the lead investigator in the spatial self-localization project. I was responsible for building and developing the majority of the ideas, implementing the FPGA and ASIC based spatial self-localization, SSL, accelerator, collecting and analyzing the data and finally, composing the manuscript. Igor Paprotny was involved in the early stages of concept formation of optical Angle of Arrival sensors for localization and localization of mobile platforms in general. A version of Chapters 8 and 9 has been published in IEEE Embedded Systems Letters journal. Amit Trivedi, my adviser, was the lead investigator in this project. I was responsible for building and developing the majority of the ideas, implementing the FPGA and ASIC speaker-recognition, SpkrRec, accelerator, collecting and analyzing the data and finally, composing the manuscript. Alberto Gianelli was involved in the early stages on the definition of the SpkrRec interface to a generic Gaussian Mixture Model, GMM, LUT-free architecture for speaker recognition. A version of Chapters 12 and 13 has been submitted for publication to IEEE Transactions on Computers.

## TABLE OF CONTENTS

<b><u>CHAPTER</u></b>	<b><u>PAGE</u></b>
<b>1 INTRODUCTION . . . . .</b>	<b>1</b>
1.1 Spatial Self-Localization . . . . .	1
1.2 Speaker and or Command Recognition . . . . .	1
1.3 Fully Connected Layers for Neural-Network Processing . . . . .	2
1.4 Thesis Organization . . . . .	3
<b>2 SSL INTRODUCTION . . . . .</b>	<b>9</b>
<b>3 SSL BACKGROUND AND NOTATION . . . . .</b>	<b>11</b>
3.1 RNN formulation of AOA-based 3D Localization . . . . .	13
3.1.1 Complexity of Primal-Dual Linear Program Solution . . . . .	17
<b>4 SSL DESIGN FEATURES AND PARAMETERS . . . . .</b>	<b>18</b>
4.1 Low Power Digital Microarchitecture for RNN-based Localization . . . . .	18
4.1.1 Matrix-vector product with sub-matrix scheduling . . . . .	20
4.1.2 Low Power Comparator Unit . . . . .	21
<b>5 SSL SIMULATION AND IMPLEMENTATION RESULTS . . . . .</b>	<b>23</b>
5.1 Simulation Results . . . . .	23
5.1.1 Functional Simulations . . . . .	23
5.1.2 FPGA Implementation Results . . . . .	25
5.1.3 PDK45 ASIC Implementation . . . . .	27
5.2 Conclusion . . . . .	28
5.2.1 Comparison to Cholesky decomposition with forward/backward substitution . . . . .	30
5.2.2 Comparison to QR (QRD) Decomposition using Householder transformation . . . . .	30
5.2.3 Comparison with Adjoint Matrix method for matrix inversion to solving Linear System . . . . .	31
5.2.4 Comparison with Cayley – Hamilton Method for NxN matrix inversion . . . . .	31
5.2.5 Comparison with EKF . . . . .	32
5.2.6 Comparison with EIF . . . . .	32
5.2.7 Comparison with SPKS . . . . .	33
5.2.8 Comparison with Particle filter . . . . .	33
<b>6 SPEAKER RECOGNITION INTRODUCTION . . . . .</b>	<b>36</b>



## TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
<b>7</b>	<b>BACKGROUND IN POWER SAVING TECHNIQUES FOR GMM CLASSIFIERS OF SPEAKERS</b> . . . . .	39
7.1	Complexity of online k-Means clustering (Lloyd’s algorithm) . . . . .	40
<b>8</b>	<b>SPKRREC DESIGN FEATURES AND PARAMETERS</b> . . . . .	41
8.1	Online k-Means Clustering Architecture . . . . .	41
8.1.1	Squared Euclidean Distance Computation Unit . . . . .	41
8.1.2	Nearest Centroid Identification Unit . . . . .	42
8.1.3	Centroid Update Unit . . . . .	44
<b>9</b>	<b>SPKRREC SIMULATION AND IMPLEMENTATION RESULTS</b> . . . . .	46
9.1	Simulation Setup . . . . .	46
9.2	Functionality Characterization . . . . .	48
9.3	Energy Efficiency Characterization . . . . .	48
9.4	Comparison to alternative frame reduction approaches . . . . .	52
9.5	CONCLUSION . . . . .	53
<b>10</b>	<b>INTRODUCTION TO ACCELERATION OF FULLY CONNECTED LAYERS</b> . . . . .	56
<b>11</b>	<b>BACKGROUND ON FULLY CONNECTED LAYERS IN CNNs</b> . . . . .	62
<b>12</b>	<b>DESIGN FEATURES AND PARAMETERS OF FC-ACCEL</b> . . . . .	68
12.1	HBM Data-Prefetch Unit and On-chip Buffer, DPR-BUF . . . . .	69
12.2	Matrix-Vector Multiplier Unit . . . . .	72
12.3	Vector Accumulator Unit . . . . .	75
12.4	ReLU and Bias Addition Unit . . . . .	76
12.5	Main Processing Sequence . . . . .	77
<b>13</b>	<b>SIMULATION AND ASIC IMPLEMENTATION RESULTS OF FC-ACCEL FOR FC LAYERS IN ALEXNET AND VGG16</b> . . . . .	80
13.1	CMOS ASIC Implementation . . . . .	83
13.2	Characterization of a pipelined 16x16 PE . . . . .	85
13.3	FC8 Latency for 1 HBM Shared by 128 PEs . . . . .	87
13.4	FC-Accel Complexity Analysis . . . . .	88
13.5	Up-Scaling to Larger FC Layers . . . . .	89
<b>14</b>	<b>OPEN PROBLEMS AND ACCELERATOR EXTENSIONS FOR THEM</b> . . . . .	94
14.1	Spatial Localization - SSL Extensions . . . . .	94
14.1.1	Non-Linear Cost Function for AOA based Estimation . . . . .	94
14.1.2	Multi-Modal methods . . . . .	97
14.2	Speaker Recognition - SpkrRec Extensions . . . . .	98

## TABLE OF CONTENTS (Continued)

<b><u>CHAPTER</u></b>		<b><u>PAGE</u></b>
	14.2.1 Learning the Optimal Number of Clusters . . . . .	98
	14.2.2 Simulated Annealing schedule . . . . .	99
	14.2.3 Integration with a GMM-scoring block and Command Recognition for Robotic Arm control . . . . .	100
	14.3 Fully Connected layers in DNNs - FC-Accel Extensions . . . . .	101
<b>15</b>	<b>CONCLUSION . . . . .</b>	<b>109</b>
	<b>APPENDIX . . . . .</b>	<b>113</b>
	<b>CITED LITERATURE . . . . .</b>	<b>121</b>
	<b>VITA . . . . .</b>	<b>127</b>

## LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	FC8 LATENCY FOR 8X8 AND 16X16 PE WITH 1 HBM AND 128 PE MODULES. THE NUMBER OF READ ACCESSES TO THE HBM ARE LISTED FOR EACH PE DIMENSION. . . . .	87
II	FC8 LATENCY FOR 8X8 AND 16X16 PE WITH 1 HBM AND 128 PE MODULES. THE NUMBER OF READ ACCESSES TO THE HBM ARE LISTED FOR EACH PE DIMENSION. . . . .	93

## LIST OF FIGURES

<b>FIGURE</b>		<b>PAGE</b>
1	Sensor localization in 3D with 4 anchors and Angle-of-Arrival (AOA) measurements. . . . .	13
2	RNN-based co-processor for 3D localization with $M$ anchors. $\theta(n+1)$ represents the predicted target location at step $n+1$ . . . . .	19
3	Distributed arithmetics for vector-vector multiplication. . . . .	20
4	Comparator unit for three 16-bit signed two's complements. . . . .	22
5	The comparison of functional simulations against fixed-point Verilog. Verilog simulations are done in Q(10,17) format. . . . .	24
6	Histogram of the estimated coordinates. In each subplot, two thousand samples from the fixed point Verilog simulations are used. . . . .	25
7	(a) Standard deviation of localization error at increasing standard deviation of AOA noise. (b) Average localization error for thousand runs computed as Euclidean distance from the ground truth at varying number of anchors. (c) Localization error at various anchor configurations considering four anchors. Results in (a) and (b) are for configuration C2. . . . .	26
8	Power characterization of (a) FPGA and (b) ASIC implementation. FPGA is implemented in 180 nm technology. ASIC is implemented using 45 nm predictive technology models. . . . .	27
9	SI computations flow: (a) The standard architecture where all $N$ frames of MFCC vectors are processed, for a total of $N \times M$ GMM evaluations. (b) The proposed architecture adds a clustering layer (CL) to map MFCC frames to $k$ centroids which reduces the total GMM evaluations to atmost $k \times M$ . . . . .	37
10	Overview of our clustering approach. . . . .	40
11	Datapath for computing the squared Euclidean distance between a new MFCC feature vector $x_t$ and all $k$ centroids $z_i$ . . . . .	42

## LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
12	(a) Systolic array for sorting a serial input stream of $M$ words in $M$ clock cycles. (b) Processing cell in the array. . . . .	43
13	(a) Centroid update unit for updating only the winning centroid. (b) Bit-serial divider for dividing an element from $SUB$ by the current value from $CNT\_BLK$ . Only the quotient is used in the downstream processing. . . . .	45
14	(a) Convergence of the cumulative error for centroids 1 and 4. Note that the residual value of 20 is reached after 35 frames (iterations) for centroid 1 and after 54 frames for centroid 4. Different centroid estimates converge to their residual values for a different number of frames. (b) Evolution of posterior log-probabilities of 38 test speakers. . . . .	47
15	(a) Success rate vs. number of speakers and number of centroids per speaker - 10 speakers (blue) 20(green) 38(black). (b) Reductions in number of operations: blue for GMM scoring with non-clustered frames; orange for the proposed GMM scoring with forty centroids. . . . .	49
16	(a) Energy dissipation vs. number of speakers. Blue trace is scoring all 500 MFCC test frames with all GMM models. Red trace is scoring forty centroids with all GMM models. (b) Breakdown of power dissipation for the centroid estimator architecture. . . . .	52
17	Error rate vs. time to classify a test speaker. . . . .	54
18	Fully Connected FC8 layer in AlexNet or VGG-16: 4096 input and 1000 outputs. Groups of 8 are indicated in the input and output vectors respectively. . . . .	60
19	The equivalent matrix-vector multiplication for the FC layer in Fig. 1. The weights are grouped in $8 \times 8$ sub-matrices (tiles) $W_1$ , $W_2$ , etc. Each column of sub-matrices is mapped, during its time-slot, to a set of 128 MACs and 128 PEs. This is the columnwise block(tile) decomposition used in FC-Accel. The same set of MACs and PEs is reused for all 512 time-slots during processing. . . . .	61
20	Fully Connected layer FC F5-F6 in a typical CNN. F5 is a 1D vector of 120 input scalar features and F6 is a 1D vector of 10 output scalar features. . . . .	63
21	HBM 3D SDRAM memory stack technology (left) and GPU or general purpose (FC Accelerator instead of GPU) application (right). . . . .	65

## LIST OF FIGURES (Continued)

<b><u>FIGURE</u></b>		<b><u>PAGE</u></b>
22	High level architecture block diagram. Each HBM has a dedicated data-pre-fetch and on-chip buffer unit, DPR-BUF, with 8 rate matching FIFOs. An HBM's DPR-BUF ensures that 1024 bits of weights, stored in the on-chip buffer, are aligned for a single cycle read by the PE. Input and output memories have dedicated address generators. One top-level controller schedules the data flow in all 128 PE channels. . . . .	70
23	Read access timing from JESD235C. The access starts with the first read request to column address Ca. After R T-cycles (T0 to T6 for R=6 example) a burst of 4 128-bit words, Da to Da+3, is available on DQ[127:0]. Similarly, the second read request to column address Cb generates a burst of 4 128-bit words, Db to Db+3. The DPR-BUF combines the 8 128-bit words and writes them into 8 corresponding FIFOs. The 8 FIFOs are then read into the 1024 bit on-chip buffer. . . . .	72
24	Data prefetcher and on-chip buffer for HBM read accesses. One HBM is shown driving weights to its PE. The main controller issues two read requests to column addresses Ca and Cb. Each request generates a burst of 4 128 bit transactions on DQ. The prefetcher write 8 128 bit DQ values into 8 corresponding FIFOs. A read request is then issued to all FIFOs, and their output is stored in a single 1024 bit register. This aligns the weights read-out cycle with the HBM-IN read out of the next 8 16-bit input feature values. . . . .	73
25	DPR-BUF HBM memory read out cycles m1 to m8 in the 500MHz domain and FIFO write cycles wr1 to wr8. FIFO read cycle Rd and PE processing cycles P1 to P3 are in the 662 MHz domain. The FIFO is read every 4th cycle. . . . .	74
26	MV-mult micro-architecture for 8x8 tile of weights. . . . .	75
27	Vector accumulator V-Accum datapath. . . . .	76
28	Main controller sequence. All 128 PEs are processing a new 8x1 feature vector from In-MEM in each state ST1 ... ST512. In each state an 8x8 tile of weights is read from the HBM corresponding to each PE. . . . .	77
29	Processing latency for FC8 in usec for several benchmarks. . . . .	81
30	Processing Blocks performance. . . . .	82
31	Power per PE processing block. . . . .	83
32	GOPS comparison with ASIC and FPGA platforms for FC8 acceleration. . . . .	84

## LIST OF FIGURES (Continued)

<b><u>FIGURE</u></b>	<b><u>PAGE</u></b>
33 FC-Accel with 128 PEs. PDK 45nm standard cells implementation. . . . .	85
34 Implementation data for pipelined 16x16 PE at 662 MHz. The y axis is log( ) base 10. . . . .	86
35 Up-scaling the propped micro-architecture for handling FC6 and FC7 lay- ers. In each pass 128 HBMs and 128 16x16 PEs are reused. The input and output memories are connected as in Fig.3 . . . . .	91
36 Estimated performance of FC6 and FC7 layer acceleration with FC-Accel 128x1 array of pipelined 16x16 PEs and two passes over the array. . . . .	92
37 NLS-LM processing pipeline for 3D AOA measurements. The processing iterations continue until parameters updates (3D coordinate updates) become smaller than a threshold value. . . . .	96
38 Matlab simulation of NLS-LM processing pipeline and a possible systolic array realization. . . . .	103
39 RNN architecture extension for solving a constrained Quadratic Program, QP, as well as an LP. . . . .	104
40 SFM processing to generate P'1...P'5 in local camera coordinate system . . .	105
41 Module for Horn acceleration, with neural network for PCA and eigenvalue decomposition. . . . .	106
42 The elbow method for optimal K selection . . . . .	107
43 Spotted word recognition for control of diadic manipulator robotic arm TR45.	108

## **LIST OF ABBREVIATIONS**

ANOVA	Analysis of Variations
SSL	Spatial Self-Localization
HMM	Hidden Markov Model
SpkrRec	Speaker Recognition
FC-NN	Fully Connected Neural Network layer
GMM	Gaussian Mixture Model
RNN	Recurrent Neural Network
RMSE	Root-Mean-Square Error
UIC	University of Illinois at Chicago



## SUMMARY

Recent advances in CMOS VLSI technology have enabled the tremendous growth of devices at the edge of the cloud and in indoor environments: IoT indoor appliances, mobile indoor medical assistants, mobile indoor manufacturing platforms, indoor drone assistants, and others. As anticipated, this growth (in edge-device numbers and capabilities) is generating large communications and data processing workloads for the servers in the cloud. One approach to help manage this trend is to make the edge-nodes more intelligent and able to process more data onboard (within the edge-node) before communicating with the servers. This thesis proposes hardware accelerator solutions to three types of onboard (within platform) processing: Spatial Self-Localization (SSL) which localizes the platform in space, Speaker Recognition (SpkrRec) which allows human voice control of the platform and authentication of the human speaker, and Fully-Connected layer evaluation in Neural Networks ( FC-NN ) for accelerated neural network processing withing the platform. Onboard processing is assumed to include a multi-core SoC (CPU/GPU), conventional SRAM and DRAM memory as well as high bandwidth memory, HBM or 3D-DRAM, and communication and sensing subsystems. The SSL, SpkRec, and FC-NN accelerators can be integrated with the SoC's peripheral bus structures such as AXI-Stream, AXI-Lite, AXI-HBM, JESD235A, JESD235B, GPMC, DMA, and similar high-speed processor interfaces.

# CHAPTER 1

## INTRODUCTION

This research has been motivated by three important problems in real-time integrated hardware-software implementations of indoor mobile robotic platforms at the edge of the cloud:

### 1.1 Spatial Self-Localization

Spatial self-localization (in 2D or 3D space) is performed within the device. Example applications include simultaneous localization and mapping (SLAM) and navigation for autonomous mobile robots. We focus on optical angle-of-arrival, AOA, localization based on fixed known position LEDs or infrared, Ir, beacons or anchors mounted in an indoor environment and photodetectors built in the mobile platform. These are lower cost localization systems when compared to 3D camera or image sensor based AOA detection, but can offer comparable performance in most indoor cases. Recent implementations include (Popoola and et al, 2019), (Arafa et al., 2015), and (Navarro and et al, 2017).

### 1.2 Speaker and or Command Recognition

Speaker and or command recognition is performed within the device to allow human control of the device and speaker identification. Example applications include a limited number of voice commands from a set of known speakers, which are used to control indoor drones as in (Fuhrman and et al, 2019), (Sugadev and et al, 2016). Another application is spotted word (command) recognition for control of diadic manipulator robotic arm (Elemery, 2011).

### 1.3 Fully Connected Layers for Neural-Network Processing

Fully connected layers processing, for embedded neural-networks, is performed within the device to achieve real-time inferencing with the DNN. Example applications include deep CNN processing such as AlexNet or VGG-16, where a typical fully-connected layer has 4096 input features and 1000 output neuron activations, and can increase up to 250088 input features and 4096 output neuron activations, as in (Krizhevsky and et al, 2017) and (Simonyan and et al, 2014). Another application is bounding-box object localization in an image using reinforcement learning for training and a Q-Network for inferencing with several fully-connected layers as implemented in (Caicedo and et al, 2015).

The main contribution for the spatial self-localization problem is a block Least-Squares formulation of the 3D or 2D Angle-of-Arrival (AOA) equations for increasing number of anchors. An incoming stream of 3D or 2D AOA sensor measurements is used to define the overdetermined system of equations. The AOA equations are then used to formulate a primal-dual linear program and solved via a novel Recurrent Neural Network (RNN) hardware accelerator which does not use matrix inversion. Key results for FPGA and CMOS ASIC implementations have been published in (Iliev and Trivedi, 2019) for the 3D case and in (Iliev and Trivedi, 2017) for the 2D case.

The main contribution for the speaker recognition problem is a novel online K-means cluster generator for the following Gaussian Mixture Model (GMM) evaluation (scoring). Online Euclidean distances and K-means centroids (for incoming MFCC frames) are computed with novel low-power pipelined datapaths. Key results for FPGA and CMOS ASIC implementations have been published in (Iliev et al., 2019).

The main contribution for the fully connected (FC) layers problem is a novel online accelerator based on 8x8 processing elements (PEs) for matrix-vector multiplication, a column of blocks (tiles) decomposition of the weights matrix, and 128 multiply-accumulate (MAC) units integrated with 128 High Bandwidth Memory (HBM) units for storing the pre-trained weights. On chip routing such as 2D mesh routing with Network on Chip, NoC, is not required. Compression of the DNNs features or weights is not required either. The architecture can be scaled to 16x16 PEs for handling larger FC6 and FC7 layers in AlexNet and in VGG16. Key results for CMOS ASIC implementations of the FC8 layer accelerator have been published in the journal of IEEE Embedded Systems Letters.

#### 1.4 Thesis Organization

This thesis is organized into four parts and 15 chapters. Descriptions of each part and each chapter of the dissertation are as follows:

**Part I, Spatial Self-Localization:** This part contains an introduction to the 2D or 3D AOA spatial self-localization (SSL) research problem, a review of the existing works and a description of the key design features and parameters of the SSL-Accel accelerator for edge node devices.

**Chapter 2:** This chapter provides the reader with the motivation for the spatial self-localization problem and identifies the research questions that the work is focused on such as handling an increased number of anchors with low latency and keeping circuit power consumption within limits. Our

approach to these problems is explained and a list of related contributions is included in this chapter, as well.

**Chapter 3:** In this chapter we review some background on AOA localization systems that will be referenced later in the thesis.

**Chapter 4:** This chapter presents the key design features and parameters of the SSL accelerator, SSL-Accel, and FPGA and ASIC implementations. Datapath flow and controller (scheduler) diagrams and state machines are covered as well as the RNN operation with fixed-point data types for all layers.

**Chapter 5:** Simulation results are presented, and FPGA and CMOS ASIC PDK 45 nm implementation details and performance are covered as well. A conclusion follows with complexity comparisons of the SSL-Accel RNN dual-LP approach to various matrix-inversion architectures for solving overdetermined systems as well as Kalman filter based and particle filter based localizers.

**Part II, Speaker Recognition:** This part contains a background on speaker recognition based on Gaussian Mixture Models (GMMs), and a description of key design features and parameters of the online k-Means clustering accelerator SpkrRec-Accel for edge node devices.

**Chapter 6:** This chapter provides an overview of speaker identification using GMM models of a set of enrolled speakers.

**Chapter 7:** This chapter introduces the research problem of reducing the dimensionality of speech features for a collection of speakers so that overall identification latency and circuit power are reduced to meet implementation constraints.

**Chapter 8:** This chapter presents the key design features and parameters of the SpkrRec-Accel accelerator. Datapath flow and controller (scheduler) timing diagrams and state machines are covered as well as novel distance computing and centroid update datapaths with fixed-point data processing.

**Chapter 9:** Simulation results are presented, and FPGA and CMOS ASIC PDK 45nm implementation details and performance are covered in this chapter. A comparison with other approaches is also presented.

**Part III, Fully Connected Layers in Deep Neural Networks:** This part presents current solutions to the FC layers problem in deep CNNs and DNNs such as AlexNet and VGG-16. It then details a novel, higher performance, accelerator architecture FC-NN-Accel and its key design features and parameters for edge node devices.

**Chapter 10:** This chapter provides an introduction on the impact of FC (or dense) layers to overall CNN and DNN performance. Current solutions for edge node devices, implemented as CMOS ASIC or FPGAs, are reviewed and possible areas of improvement are identified.

**Chapter 11:** This chapter contains some background information on FC layers in DNNs, their relation to convolutional layers, and other FC specific features relative to their hardware


acceleration. Parallel matrix-vector multiplication algorithms are also reviewed including rowwise block decomposition, columnwise block decomposition, and checkerboard block decomposition of the FC layer's weights matrix.

**Chapter 12:** In this chapter, a novel solution is proposed to address the areas for improvement.

The FC-NN-Accel architecture is presented in detail with its key design parameters such as tile size for decomposition of the weights matrix into tiles; array of High Bandwidth Memory (HBM JESD235A) for driving an array of processing elements (PEs) in a maximally parallel organization using column of blocks (tiles) decomposition; PE design with novel matrix-vector multiplier, accumulator, and ReLU and biasing circuits; streaming data-prefetching from the HBM array and streaming address-generation for the output memory. A CMOS ASIC PDK45 nm implementation is described as well.

**Chapter 13:** This chapter presents an ASIC implementation of FC-Accel as well as simulated performance results for FC-NN-Accel with 128 HBM weights memories (1 MB each), a 4KB input features memory (FC layer only), and compares achieved results with other AlexNet and VGG-16 implementations with published FC layer performance results.

**Part IV, Conclusion:** This part contains open problems and possible extensions of the SSL, SprkRec, and FC-NN accelerators to address some of the problems. Example problems are robust location estimation in noisy environments with multi-modal methods such as AOA and image-based methods, joint speaker and command recognition accuracy, and power consumption and scalability in fully connected layer inference. Finally concluding remarks are presented.



**Chapter 14:** This chapter discusses a few possible extensions of our work and also reviews related open problems which can be addressed by these extensions.

**Chapter 15:** A concluding summary of the thesis is presented in this chapter.



# Part I

## Spatial Self-Localization

## CHAPTER 2

### SSL INTRODUCTION

In this chapter I discuss a novel low power circuit to self-localize a mobile sensor node in three-dimensional (3D) space using a passive optical receiver. Self-localization of sensors, where a sensor node computes its spatial location by itself, reduces transmission demand, and improves real-time conformity of mobile wireless sensor systems. Our approach forms an over-determined system of angles-of-arrival (AOAs) to mobile sensor received from an optical anchor grid. Optical AOA based localization algorithms have shown a lot of advantages for indoor localization, over RF RSSI power, optical RSSI power, ultrasound time-of-flight, and other method as shown in our recent survey on spatial localization (Iliev and Paprotny, 2015). The AOA system is solved with a linear program (LP) solver, which is implemented using a non-linear feedback recurrent neural network (RNN). To solve the primal and dual LP optimization problems in the AOA system, we show a single multi-functional data path that does not require matrix inversions; thereby, enables within-sensor low power computations to self-localize. Additionally, unlike other optical indoor positioning architectures, our approach does not require measurements of received signal strength (RSS) and, thereby, is insensitive to power and alignment imbalances in the anchor grid. We show proof-of-concept FPGA and ASIC simulations of our approach and validate its operation under noisy AOA data and for different numbers of anchors. An FPGA implementation in 180

nm CMOS achieves a peak  $\sim 0.12$  mega localization operations per second per Watt (MOPS/W), while ASIC design in 45 nm CMOS shows a peak  $\sim 7.7$  MOPS/W.

## CHAPTER 3

### SSL BACKGROUND AND NOTATION

Many emerging mobile wireless sensor networks such as mobile robotics, indoor positioning for pick-and-place robotics, and smartphone localization in visible light communication (VLC) systems rely on the indoor spatial localization of sensors. However, spatial-localization of sensors faces several challenges in indoor environments. Radio frequency (RF)-based localization suffers from multipath reflections, fading, and shadowing of the radio signals (Tang and Dodds, 2007). Similarly, localization based on received signal strength (RSS) from optical beacons suffers from power imbalances and fading (Harnilovic, 2004). Contrarily, indoor localization based on angle-of-arrival (AOA) measurements, such as using a passive corner-cube photoreceiver or an image sensor, has shown more promising performance by being insensitive to signal power (Bergen et al., 2018), (Arafa et al., 2015). Moreover, optical localization based on AOA also obviates synchronization between transmitter and receiver or between co-operating transmitters as needed in optical time-of-arrival and optical time-difference-of-arrival (Wang et al., 2013).

While optical localization based on AOA measurements is promising for indoor applications, a *self-localization*, where a sensor node can estimate its spatial position on its own, is also critical. Unlike traditional schemes, where a server collects data from all sensors to estimate and communicate their positions, self-localization can significantly suppress communication demand and

associated energy-overheads. Therefore, in this work, we focus on low power architectures to self-localize a sensor based on its AOA measurements.

Our approach uses a recurrent neural network (RNN) to perform all computations necessary for self-localization within the sensor node. Unlike localization methods such as Kalman filter (EKF) (Gasparri and Pascucci, 2010) and extended information filter (EIF) (Taparugssanagorn et al., 2013), our approach doesn't require matrix inversions. Meanwhile, implementation of matrix inversion methods (Yan et al., 2010), (Karkooti et al., 2005) can add significant latency to the prediction. We discuss a digital CMOS micro-architecture for the RNN using fixed-point arithmetic comprising time-multiplexed matrix-vector and vector-scalar multiplications along with other arithmetic operations. The proposed micro-architecture uses a core  $3 \times 3$  matrix-vector multiplier unit to solve the 3D localization problem with measurements from  $M$  anchors, where  $M$  is in a multiple of two. Average localization error based on optical AOA measurements decreases with an increase of available anchors. While our preliminary conference paper in (Iliev and Trivedi, 2017) showed RNN-based self-localization in 2D, in this work, we expand the scheme for 3D self-localization. Unlike (Iliev and Trivedi, 2017), the proposed 3D micro-architecture is reconfigurable to a number of anchors. We also discuss novel datapaths based on distributed matrix-vector arithmetic with dynamic bit-plane selections. Sec. II discusses our RNN formulation of AOA-based 3D localization. Sec. III presents the proposed micro-architecture for 3D localization. Sec. IV presents simulation results for FPGA and ASIC implementations. Sec. V concludes. We use bold lowercase English and Greek letters for vectors and bold uppercase letters for matrices.

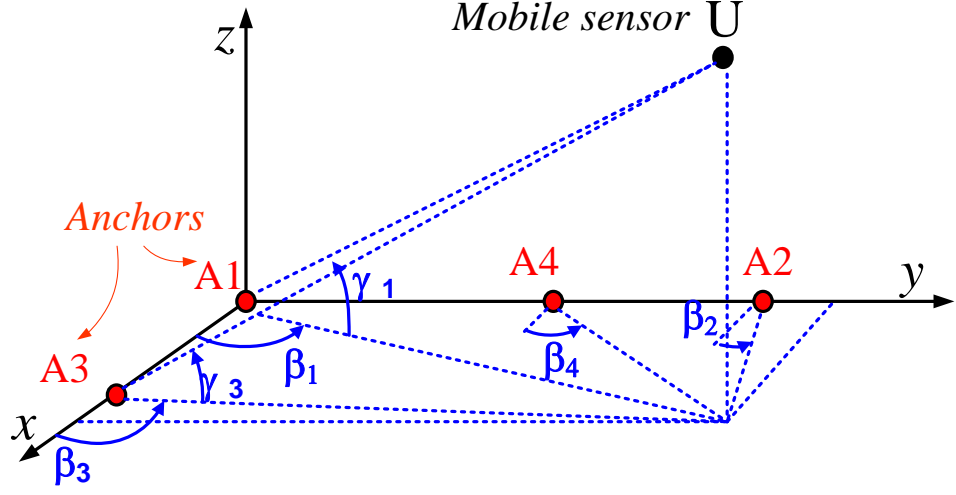


Figure 1: Sensor localization in 3D with 4 anchors and Angle-of-Arrival (AOA) measurements.

### 3.1 RNN formulation of AOA-based 3D Localization

Fig. 1 shows a test-case where an optical receiver  $U$  is localized using an anchor grid  $A_1$ – $A_4$ . We derive a system of linear equations to localize  $U$  using an arbitrary number of anchors. The angle-of-arrival (AOA)-based localization has been analyzed in (Gavish and Weiss, 1992) in its most general form. The problem can be solved with a Maximum Likelihood (ML) algorithm (Vaghefi et al., 2010) when the exact positions of anchors are available and the measurement noise follows a Gaussian distribution. In this case, the ML problem yields to a nonlinear minimization as

$$\theta_{ML} = \arg \min_{\alpha} (\tilde{\alpha} - \alpha)^T \times C \times (\tilde{\alpha} - \alpha). \quad (3.1)$$

Here, the spatial coordinates of  $U$  are estimated by  $\theta_{ML}$  and  $\tilde{\alpha}$  is the AOA measurements vector from  $M$  anchors given as

$$\tilde{\alpha} = [\tilde{\alpha}_1 \tilde{\alpha}_2 \dots \tilde{\alpha}_M]^T = \tilde{\alpha}_0 + n, \quad (3.2)$$

with  $\tilde{\alpha}_0$  as the true angle (in radians) and  $n$  is the measurement noise.  $C$  in Eq. 3.1 is the inverse covariance matrix of  $n$ .

Notice that each  $\alpha_i$  in Eq. 14.2 corresponds to two angle measurements (longitude and latitude) from an anchor  $A_i$ , as shown in Fig. 1; therefore,  $\alpha_i = (\beta_i, \gamma_i)$ . Here,  $\beta_i$  and  $\gamma_i$  depend on the location of  $A_i$ , i.e.,  $(x_i, y_i, z_i)$ , and the estimate for  $U$ , i.e.,  $(x_T, y_T, z_T)$ , as

$$\beta_i = \tan^{-1} \left( \frac{y_T - y_i}{x_T - x_i} \right) \text{ \& } \gamma_i = \tan^{-1} \left( \sin \beta_i \times \frac{z_T - z_i}{y_T - y_i} \right). \quad (3.3)$$

For small measurement errors, (Vaghefi et al., 2010) shows that Eq. 3.1 can be reduced to the following Least Squares problem

$$D \times \begin{bmatrix} x_T & y_T & z_T \end{bmatrix}^T = b, \quad (3.4a)$$

$$D = \begin{bmatrix} \tan(\beta_1) & -1 & 0 \\ \tan(\beta_2) & -1 & 0 \\ 0 & \tan(\gamma_1) & -\sin(\beta_1) \\ \vdots & \vdots & \vdots \\ \tan(\beta_{M-1}) & -1 & 0 \\ \tan(\beta_M) & -1 & 0 \\ 0 & \tan(\gamma_{M-1}) & -\sin(\beta_M) \end{bmatrix}, \quad (3.4b)$$

$$b = \begin{bmatrix} x_1 \times \tan(\beta_1) - y_1 \\ x_2 \times \tan(\beta_2) - y_2 \\ y_1 \times \tan(\gamma_1) - z_1 \times \sin(\beta_1) \\ \vdots \\ x_{M-1} \times \tan(\beta_{M-1}) - y_{M-1} \\ x_M \times \tan(\beta_M) - y_M \\ y_{M-1} \times \tan(\gamma_{M-1}) - z_{M-1} \times \sin(\beta_{M-1}) \end{bmatrix}. \quad (3.4c)$$

Therefore, with adequate anchors, Eq. 4 forms an overdetermined system where each pair of anchors adds three additional equations. We reformulate the system of equations as a set of primal-dual constrained linear programs shown below

$$\textbf{Primal problem: } \min_{\theta} c^T \theta \quad \forall D \times \theta = b, \quad (3.5a)$$



$$\textbf{Dual problem: } \max_{\phi} b^T \phi \quad \forall D^T \times \phi \leq c. \quad (3.5b)$$

Here,  $\theta = [x_T \ y_T \ z_T]$  is the objective variable of the primal problem and corresponds to the unknown target location; and  $\phi$  is the corresponding variable of the dual problem. The cost function  $c$  can be selected arbitrarily since the main goal is to satisfy the constraints and the cost function simply rejects spurious solutions. For the optimizations, the anchor coordinates are translated to the first quadrant.

Constrained linear programs such as the one in Eq. 3.5 can be efficiently solved using a RNN as discussed in (Oskoei and Amiri, 2006). The corresponding neuron equations are shown below

$$\frac{d}{dt} \begin{bmatrix} \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \theta - (\theta + D^T \phi - c)^+ \\ D \times (\theta + D^T \phi - c)^+ - b \end{bmatrix} \quad (3.6)$$

The respective discrete time RNN equations for solving the primal-dual constrained programs are shown below

$$\begin{bmatrix} \theta(n+1) \\ \phi(n+1) \end{bmatrix} = \begin{bmatrix} \theta(n) + dt \times (r(n) - \theta(n)) \\ \phi(n) + dt \times (b - D \times r(n)) \end{bmatrix} \quad (3.7a)$$

$$r(n) = \max[(\theta(n) + D^T \phi(n) - c), 0] \quad (3.7b)$$

Here,  $r(n)$  is the output activation of recurrent neurons. In Eq. 7, we use Euler's forward method for numerical integration of the continuous time variant with  $dt$  as the time step for controlling the rate of convergence.

### 3.1.1 Complexity of Primal-Dual Linear Program Solution

The number of iterations of primal-dual algorithms for solving constrained linear programs is given by

$$O(\sqrt{N} * \log \frac{N}{\varepsilon}) \quad (3.8)$$

where  $N$  is the dimension of  $\theta$  and  $\varepsilon$  is the desired accuracy with  $\varepsilon > 0$  (epsilon away solution), (Cai and et al, 2014).

In the proposed SSL-Accel architecture the number of operations, per iteration, is

$$O(2 * N^2) \quad (3.9)$$

multiplications with  $N$  the dimension of  $\theta$  and an additional  $O(N)$  number of add, sub, and compare-vs-zero operations. Note that since batch Least Squares is used (with a batch size of  $N$ ) generating a corresponding primal-dual linear program for each batch, an increase in the number of nodes does not affect the above complexity.

## CHAPTER 4

### SSL DESIGN FEATURES AND PARAMETERS

*Parts of this chapter have been presented in (Iliev and Trivedi, 2017) and (Iliev and Trivedi, 2019). Copyright © 2017-2019, IEEE.*

This chapter presents the key design features and parameters of the SSL accelerator, SSL-Accel. Datapath flow and controller (scheduler) timing diagrams and state machines are covered as well as the RNN operation with fixed-point data types for all layers. Simulation results are presented, and FPGA and CMOS ASIC PDK 45 nm implementation details and performance are covered as well.

#### 4.1 Low Power Digital Microarchitecture for RNN-based Localization

We have implemented a programmable digital fixed-point arithmetic co-processor, as shown in Fig. 2 for RNN-based localization using Eq. 3.7. The co-processor works with the main processor and AOA sensors. Measured AOA values for  $[\tilde{\alpha}_1 \cdots \tilde{\alpha}_M]$  and the anchor's known coordinates are used by the main processor to compute the matrix  $D$  and column vector  $b$  using Eq. 3.4. The cost vector  $c$  in Eq. 3.7 is set as the first three elements in the column vector of  $b$  to avoid spurious outputs. The main processor writes  $D$  and  $b$  to the co-processor's internal registers. The RNN block is scalable to the number of anchors  $M$ . The value of  $M$  is a multiple of two in the current implementation. An overdetermined  $D$  [dimension:  $(3M/2) \times 3$ ] is handled by time-sharing  $3 \times 3$  submatrices of  $D$  and  $3 \times 1$  subvectors of  $b$ . The scheduler feeds each submatrix of  $D$  and each  $3 \times 1$  subvector of  $b$  to the downstream solvers.

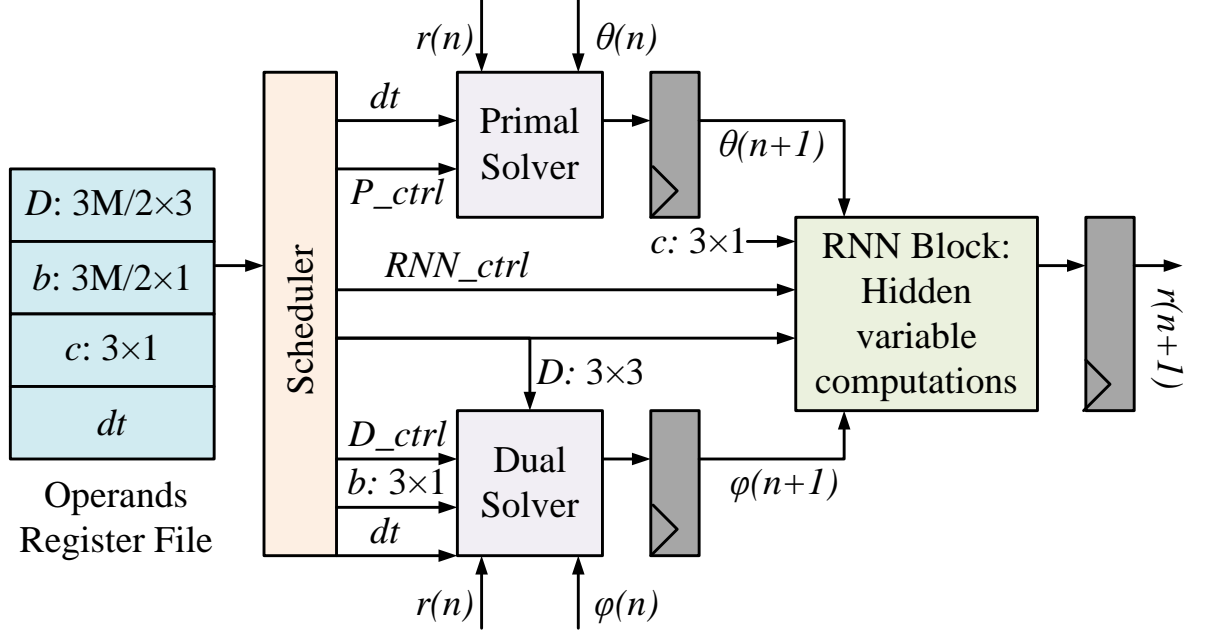


Figure 2: RNN-based co-processor for 3D localization with  $M$  anchors.  $\theta(n+1)$  represents the predicted target location at step  $n+1$ .

In the co-processor in Fig. 2, the primal-dual data paths and RNN block include a matrix vector multiplier, a vector-scalar multiplier, and five add/subtract blocks. A single vector comparator  $\text{Max}(x, 0)$  implements the neuron's activation function. The primal variables at time step  $n$  are  $\theta(n) = [x_T(n) \ y_T(n) \ z_T(n)]$ ; the dual variables are  $\phi(n) = [(\phi_1(n) \ \cdots \ \phi_{3 \times M/2}(n))]$ ; and  $D$ ,  $c$ , and  $b$  are the constants for a given set bearing (angle) measurements. Time step  $dt$  (programmed by the main processor) controls the rate of convergence and the number of iterations. Fixed point two's complement  $Q17.10$  representation is used for all arithmetic operations.

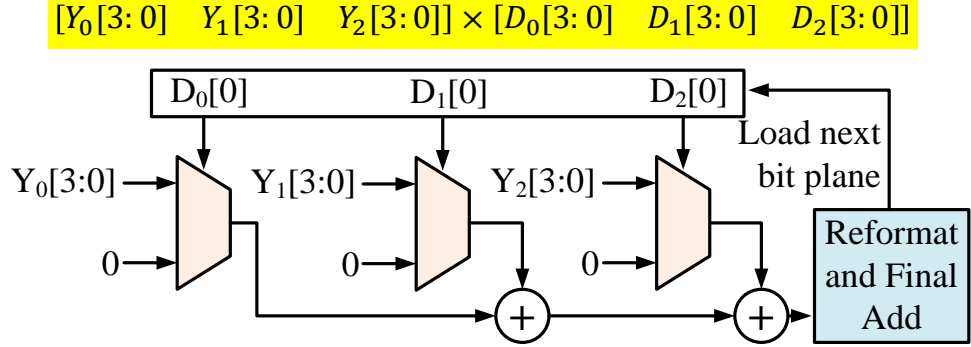


Figure 3: Distributed arithmetics for vector-vector multiplication.

The dual data path shown in Fig. 2 generates dual variables  $(\phi_1(n), \dots, \phi_{3 \times M/2}(n))$  which are used in the RNN block. The matrix-vector and vector-scalar multipliers are reused; this data path also uses Q17.10 fixed-point representation for all its arithmetic operations. A dedicated fifteen state FSM controller schedules all operations in both data paths, samples the input registers ( $D$  submatrices,  $b$  and  $c$  sub-vectors,  $M$ , and  $dt$ ), and drives all updates to the output registers  $[x_T(n) \ y_T(n) \ z_T(n)]$  and  $[(\phi_1(n) \ \dots \ \phi_{3 \times M/2}(n))]$ . The following subsections describe the main new processing blocks contributed in the proposed micro-architecture.

#### 4.1.1 Matrix-vector product with sub-matrix scheduling

The matrix-vector product operation in the above equations is implemented by reusing a core  $3 \times 3$  matrices and  $3 \times 1$  vector multiply unit implemented using distributed arithmetic for lower power. Using 45 nm CMOS predictive development kit, the implementation of the units has been optimized for 1 V supply voltage (VDD) using 8388 combinatorial cells. An operands register

file holds  $3 \times 3$  submatrices of  $D$  and corresponding  $3 \times 1$  subvectors of  $b$ . A scheduler selects the submatrices and respective subvectors in a time slot for the single-cycle matrix-vector product operation. Vectors are multiplied using distributed arithmetic operations on each bit-wise element of the data values. This is shown in Fig. 3 where for simpler illustration four-bit signed arithmetic is used. Note that distributed bitplane arithmetic in the figure obviates power hungry multipliers and uses cyclic iterations over the bitplane to compute the vector product. In Fig. 3, each  $3 \times 1$  subvector of  $Y$  and  $3 \times 3$  submatrix of  $D$  is a 4-bit two's complement fixed point fraction in the  $Q<4,0,t>$  format. The final  $D^T \times Y$  product is read out from the products register file in  $N/3$  time slots. A similar datapath is used for the matrix-vector product of the  $N \times 3$   $D$  matrix and the  $3 \times 1$   $r(k)$  vector in Eq. 3.7.

#### 4.1.2 Low Power Comparator Unit

The  $\max()$  operator needed in the computation of vector  $r(n)$  in Eq. 3.7 is optimized for low power since it's a fully combinatorial circuit. Each 16-bit two's complement element of the  $3 \times 1$   $r(n)$  vector is generated as shown in Fig. 4. The  $E_1(n)$ ,  $E_2(n)$ , and  $E_3(n)$  busses are driven by three bit-parallel subtractors for the difference terms in  $r(n)$ . The ASIC implementation of the units is optimized to 255 combinatorial cells, with  $7.7 \mu\text{W}$  of leakage power and  $6.8 \mu\text{W}$  of dynamic power at 200k kHz processing rate.

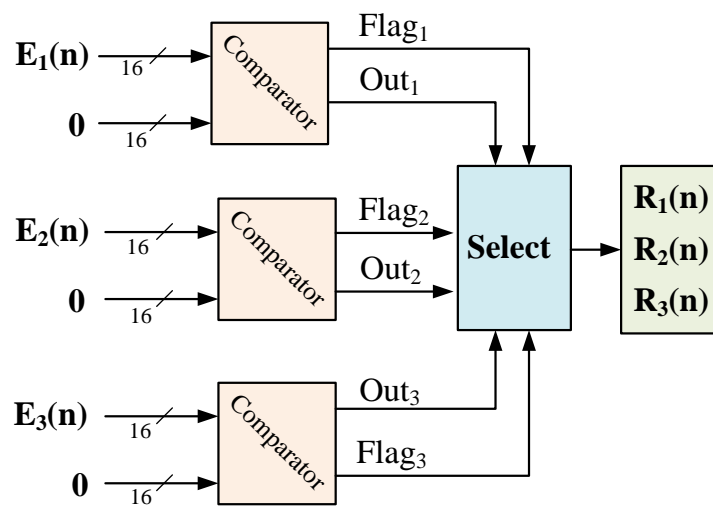


Figure 4: Comparator unit for three 16-bit signed two's complements.

## CHAPTER 5

### SSL SIMULATION AND IMPLEMENTATION RESULTS

*Parts of this chapter have been presented in (Iliev and Trivedi, 2017) and (Iliev and Trivedi, 2019). Copyright © 2017-2019, IEEE.*

#### 5.1 Simulation Results

##### 5.1.1 Functional Simulations

The proposed RNN co-processor for low power self-localization is evaluated using functional simulations and post-implementation fixed-point Verilog simulations using a benchmark setup for 3D optical AOA localizations in (Bergen et al., 2018). Using the setup, we observe correct convergence to spatial coordinates in all cases with varying numbers of anchors. An example of four anchors is shown in Fig. 5. Verilog simulations follow the trajectory of functional simulations. The coordinate estimates converge to stable values in about 600-time steps. The estimation error is  $\sim 3.18$  cm. Our average localization error also compares favorably with the 3–5 cm average localization error published in (Bergen et al., 2018), (Arafa et al., 2015) for four anchors in a one m<sup>3</sup> working area. Note that the proposed architecture uses only six measured angles instead of all eight possible angles and therefore is also a more compact solution for an embedded implementation.

Histograms of the estimated 3D coordinates under varying Gaussian noise are shown in Fig. 6. The average Euclidean distance for the error is  $\sim 3.1$  cm as computed over all simulated test cases.



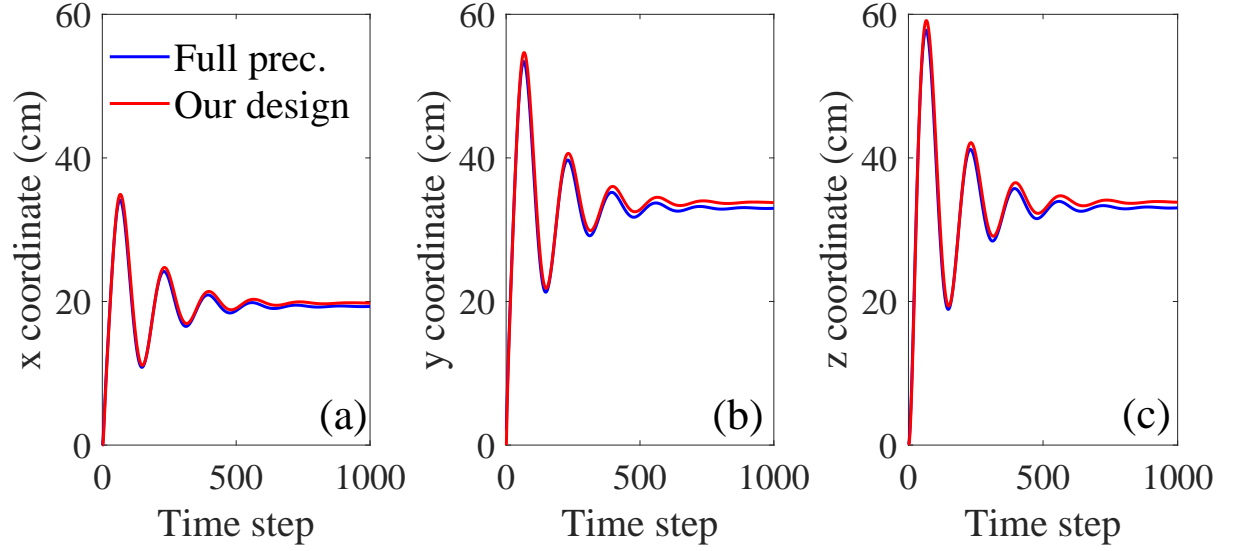


Figure 5: The comparison of functional simulations against fixed-point Verilog. Verilog simulations are done in Q(10,17) format.

The combined error in all three coordinate estimates increases with each increase in measurement noise power, as shown in Fig. 7(a). Fig. 7(b) shows the localization error against increasing number of anchors. In one  $\text{m}^3$  working area all anchors have the same  $z$  coordinate. The target is above the  $z$  plane of all anchors. The Gaussian noise for measured AOAs is simulated with unit degree variance. In Fig. 7(b), error decreases as the number of anchors increases. The figure also shows that increasing the number of anchors from two to four decreases the localization error by 42%. In (Bergen et al., 2018), authors reported a similar decrease in the error by 30% when the number of anchors is increased from three to six for AOA based positioning. Fig. 7(c) shows

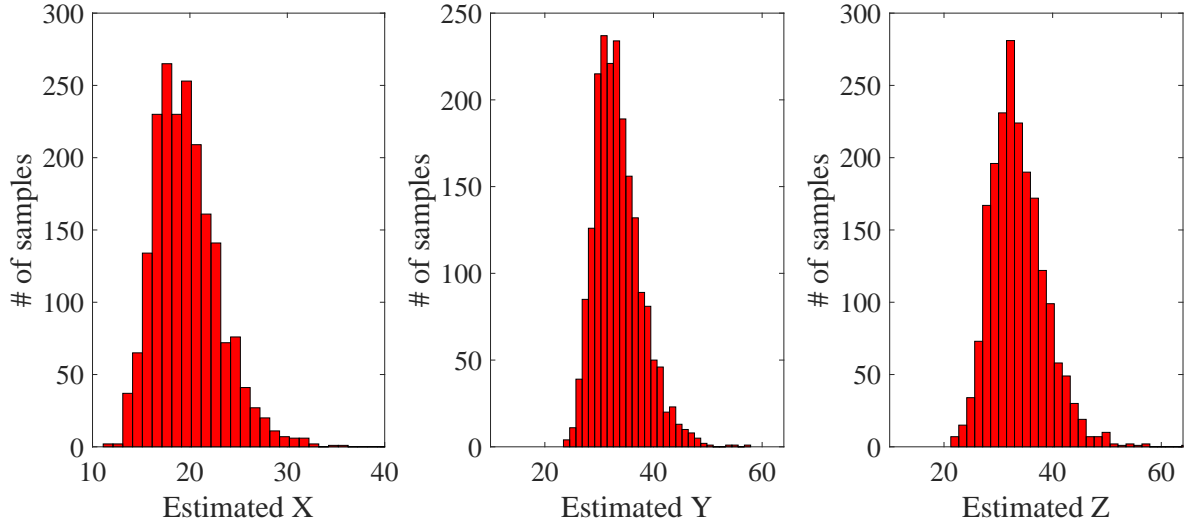


Figure 6: Histogram of the estimated coordinates. In each subplot, two thousand samples from the fixed point Verilog simulations are used.

localization error at various anchor configurations where symmetric square shows the least error similar to (Bergen et al., 2018).

### 5.1.2 FPGA Implementation Results

We also implemented the coprocessor using FPGA with the Microsemi (Actel) ProASIC3E A3PE3000 device (180 nm process) and Libero SoC v11.8 toolset. ModelSim Verilog simulations were done by setting the internal registers with pre-computed values of  $D$ ,  $b$ ,  $c$ ,  $dt$ , and setting a max iteration limit. A statistical power analysis tool in Libero SoC (SmartPower) was also used to generate the worst possible switching activity on all inputs and derive the resulting total static and dynamic power dissipation. The design's clock gating and resource scheduling for the primal

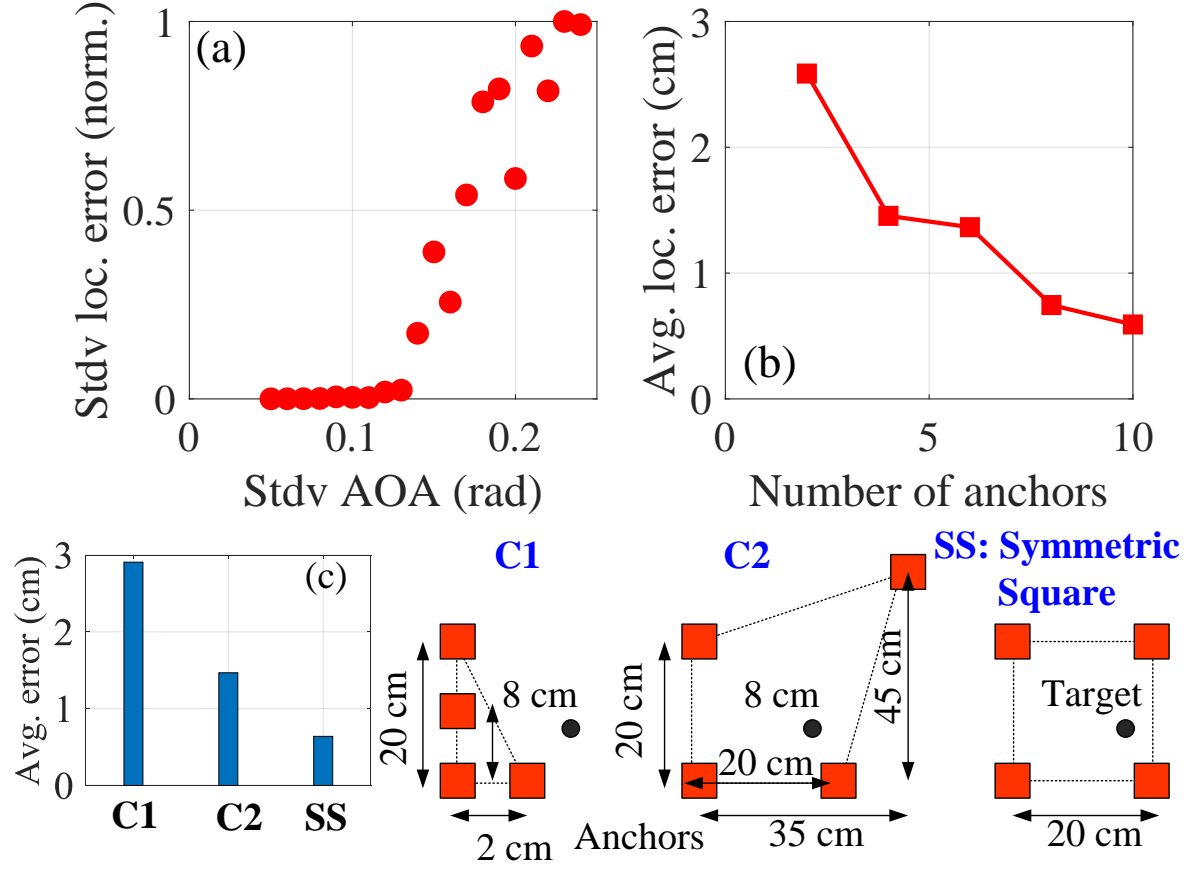


Figure 7: (a) Standard deviation of localization error at increasing standard deviation of AOA noise. (b) Average localization error for thousand runs computed as Euclidean distance from the ground truth at varying number of anchors. (c) Localization error at various anchor configurations considering four anchors. Results in (a) and (b) are for configuration C2.

and dual distributed arithmetic datapaths are utilized to minimize power dissipation of the design.

Fig. 8(a) shows the power consumption and the number of mega operations per second per Watt (MOPS/W) at the varying clock frequency. At the maximum characterization frequency, 24 MHz,

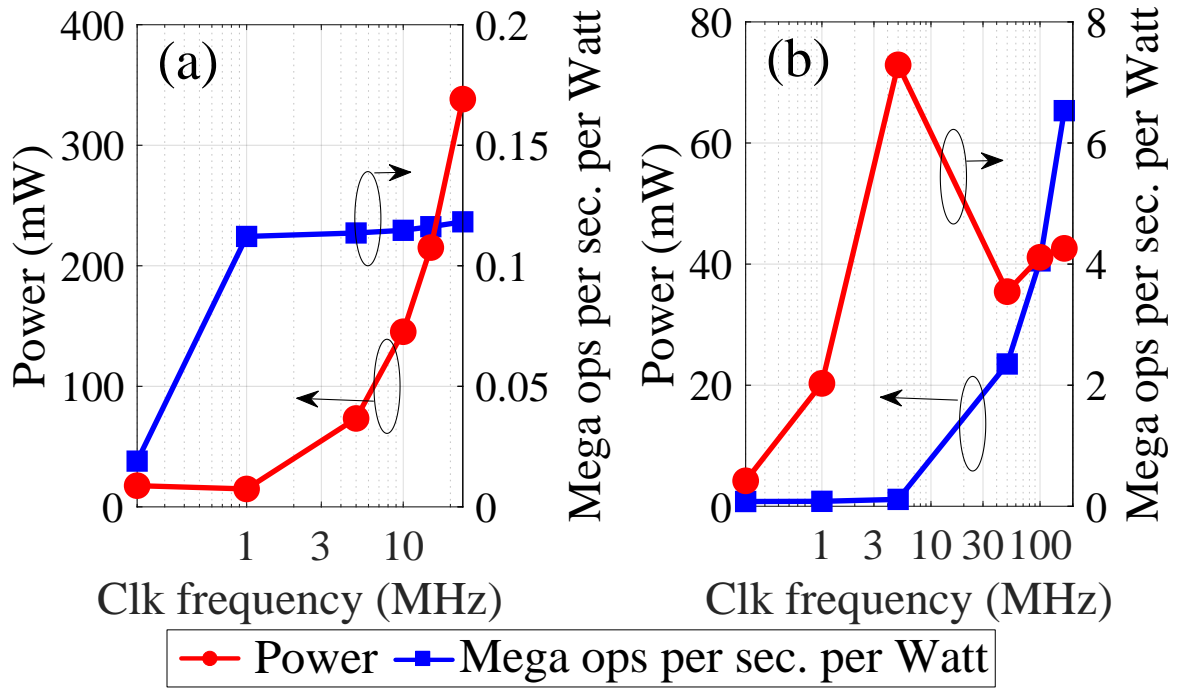


Figure 8: Power characterization of (a) FPGA and (b) ASIC implementation. FPGA is implemented in 180 nm technology. ASIC is implemented using 45 nm predictive technology models.

the design consumes 236 mW power and does ~167 thousand localization operations per second per Watt. Notably, the required localization performance of typical sensor applications is much lower (typically hundreds of operations per second). At a moderate clock speed of 200 KHz, the design consumes 38 mW power and performs eight thousand operations per second per Watt.

### 5.1.3 PDK45 ASIC Implementation

We have also implemented the proposed coprocessor in 45 nm CMOS predictive technology models using Cadence RTL Compiler (RC) and Encounter EDI. Clock gating was used in the

RC tool. Fig. 8(b) summarizes the achieved worst-case power and performance in terms of MOPS/W at the varying clock frequency. The gate-level netlist from RTL compiler was analyzed for the worst case power using RTL Compiler's statistical power analysis tool. The dynamic power dissipation can be reduced even further if static RAM cells are used instead of register files for  $D$  and the other vectors in the datapaths. Compared to FPGA, ASIC design achieves higher energy efficiency due to scaled technology and customized computing cells. At the maximum characterized frequency, 167 MHz, the design consumes 65 mW power. The design achieves a peak  $\sim 7.7$  MOPS/W at 5 MHz clock speed.

## 5.2 Conclusion

This work has proposed a novel micro-architecture for 3D indoor localization which allows a flexible number of anchors to achieve the smallest possible localization error. We enable low power computations by avoiding matrix inversions using an RNN to map localization computations. We also use a distributed arithmetic matrix-vector multiplier without look-up tables with dynamic bit-plane selections of each operand. Our FPGA implementation in 180 nm technology achieves a peak  $\sim 0.12$  MOPS/W, while ASIC implementation in 45 nm technology achieves a peak  $\sim 7.7$  MOPS/W.

There are several matrix inversion algorithms used in embedded hardware for solving the pseudo-inverse problem of overdetermined least-squares (LS) systems [1-3] below. Specifically, we refer to the methods developed for smaller size non-sparse matrices like in our case, unlike those designed for larger sparse matrices, often used in image processing. We find two competitive implementation to our case, namely: (i) fixed-point Cholesky Decomposition [1], and (ii) QRD-

RLS [2]. We qualitatively compare our approach against Cholesky Decomposition-based matrix inversion. Prior work [1] has discussed fixed-point Cholesky Decomposition for matrix inversion. In our CMOS 45 nm implementation, RNN-based solver operates at clock frequency of 167 MHz and requires 600 cycles to converge. Therefore, total computing time for localization is 3.6 s. Our present solution is for 3D localization using four anchors which is equivalent to 63 matrix inversion (see Eq. (4a) above). Comparatively, in [1] at Table I, a total of 748 clock cycles are needed for solving  $4 \times 4$  matrix inversion alone when using fixed-point Cholesky decomposition method. Therefore, at the same clock frequency, our implementation will incur lower localization latency. Note that our implementation also handles a larger matrix size  $6 \times 3$  than the  $4 \times 4$  in [1]. Our method achieves such computational efficiency by only using matrix-vector multiplication, vector addition/subtraction, and vector comparison (element-by-element greater-than-zero comparison). Meanwhile, Cholesky decomposition requires matrix-vector multiplication, vector scaling (element-by-element division by scalar), scalar square root, matrix-matrix multiplication, and matrix-matrix subtraction. Similarly, in [2], authors have discussed FPGA implementation of QRD-RLS method. In [2], to invert a  $4 \times 4$  matrix, generating the upper triangular matrix requires 777 cycles and back substitution requires 156 cycles. Therefore, even as compared to QRD-RLS, our method is less computationally expensive.

[1] Yan, Mingjian, Brighton Feng, and Tommy Song. "On matrix inversion for LTE MIMO applications using texas instruments floating point DSP," IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS. IEEE, 2010. [2] Karkooti, Marjan, Joseph R. Cavallaro, and Chris Dick. "FPGA implementation of matrix inversion using QRD-

RLS algorithm,” Asilomar Conference on Signals, Systems, and Computers. 2005. [3] Yang, Depeng, et al. “An FPGA implementation for solving least square problem,” 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. IEEE, 2009. We now present  $O()$  complexity analysis for our RNN – LP architecture in the following., For a  $N \times N$  matrix  $D$  and Linear system  $D * x = b$  where  $x$  is the  $3 \times 1$  vector of coordinates.

Our RNN-LP approach needs  $O(2*N^2)$  multiplication operations and  $O(N)$  add, sub, compare-vs-zero operations. For  $N = 3$ , we then have  $O(2*(3)^2) + O(3) = 18 + 3 = 21$  total operations (fixed-point mult, add,sub,compare).  $N=3$  and a  $3 \times 3$   $D$  matrix is the core matrix in the hardware architecture. In summary, our RNN – LP approach takes 21 total operations for a  $N = 3$  matrix and resulting  $3 \times 3$  linear system.

### **5.2.1 Comparison to Cholesky decomposition with forward/backward substitution**

To solve  $D * x = b$  ( $N \times N$  system), first decompose  $D$  to  $D = L * L$  then forward/backward substitution Cholesky takes  $O(N^3/3)$  operations to decompose the matrix into  $L * L$  format After  $L$  is found, forward substitution takes  $O(N^2)$  operations. Backward substitution also takes  $O(N^2)$  operations. For  $N = 3$ , this is  $O((3^3)/3) + O(9) + O(9) = 9 + 9 + 9 = 27$  total operations, greater than our total of 21 operations.

### **5.2.2 Comparison to QR (QRD) Decomposition using Householder transformation**

To solve  $D * x = b$ , we decompose  $D$  to  $D = Q * R$  This takes  $O(N^3)$  total operations, using the Householder transformation. We don't consider a Givens rotation (transformation) since it

typically requires a systolic array architecture, to avoid square-root and division operations. Our RNN-LP architecture is not based on a systolic array (but can be).

For  $N = 3$ , this is  $O(3^3) = 27$  total operations, greater than our total of 21 operations.

### **5.2.3 Comparison with Adjoint Matrix method for matrix inversion to solving Linear System**

We quote complexity analysis results from: “An Approach to Design a Matrix Inversion Hardware Module using FPGA” G. A. Kumar et al. 2014 IEEE ICCICCT conference. This Adjoint-Matrix inversion method needs  $2N^2 + N$  multiplications,  $N^2$  divisions, and  $N^2 + N + 1$  adds/subs (arithmetic operations). For  $N = 3$ , solving a  $3 \times 3$  linear system via the Adjoint-Matrix method takes  $21 \text{ mults} + 9 \text{ divisions} + 13 \text{ arith ops} = 43 \text{ ops}$  in total. Again this is larger than our 21 operations.

### **5.2.4 Comparison with Cayley – Hamilton Method for NxN matrix inversion**

The above reference also quotes the Cayley-Hamilton method’s complexity for the  $N=3$  case: For  $N=3$ , Cayley-Hamilton needs  $45 \text{ mults} + 9 \text{ divisions} + 20 \text{ arith ops} = 74 \text{ ops}$  in total; larger than our 21 operations.

In the following we also compare our RNN-LP architecture to several within-sensor localizer/tracker implementations of EKF (Extended Kalman Filter), EIF (Extended Information Filter), and SPKS (Sigma-point, or unscented, Kalman Smoother), all based on CPU and/or GPU processors with some hardware acceleration for scalar mathematical operations. We’ll cover a particle filter (within-sensor) implementation after this section on Kalman-based methods. The main motivation



for our RNN dual linear-program solver is that it avoids completely any matrix inversions, scalar divisions, scalar square-root operations, matrix-matrix multiplications, Jacobian matrix computations, and random variable generation (sampling from PDFs). We therefore have a much lower hardware complexity for solving the localization problem in 3D than Kalman-based or particle-filter based methods. We summarize the main differences in computational complexity below:

### **5.2.5 Comparison with EKF**

The work by A. Taparugssanagorn et al. IEEE Computer Society 2013 "A Hexagonal Coverage LED-ID Indoor Positioning Based on TDOA with Extended Kalman Filter" builds an EKF inside the sensor node, and in addition to one matrix-inversion and 6 matrix-matrix multiplications, requires two types Jacobian matrix computations: 1 -  $A_{k-1}$  is the Jacobian matrix of the partial derivatives of  $f$  (target motion law) with respect to  $R$  (error covariance of the state) 2 -  $L_k$  is the Jacobian matrix of the observation function  $l$  around the priori state estimate. It is clear that the complexity of this EKF method for localization exceeds our RNN LP method by a considerable amount.

### **5.2.6 Comparison with EIF**

A recent EIF work by A. Gasparri et al., IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 9, NO. 10, OCTOBER 2010 "An interleaved Extended Information Filter for Self-Localization in Sensor Networks", implements an Extended (Kalman) Information Filter inside a sensor node: as shown in their Tables 8, 9 and 10, this requires multiple matrix inversions, matrix-matrix multiplications, and matrix-matrix additions/subtractions. In contrast, our method uses

only matrix-vector multiplication, vector addition/subtraction, and vector comparison (element-by-element greater-than-zero comparison). This is a total of 3 types of operations. Computing Cholesky decomposition, for matrix inverse, requires matrix-vector multiplication, vector scaling (element-by-element division by scalar), scalar square root, and matrix-matrix multiplication and matrix-matrix subtraction. This is total of 5 types of operations. This further illustrates the lower complexity of our RNN LP method which does not use matrix inversion.

### **5.2.7 Comparison with SPKS**

The work by A. Paul et al. IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, VOL. 3, NO. 5, OCTOBER 2009 ” RSSI-Based Indoor Localization and Tracking Using Sigma-Point Kalman Smoothers ” The SPKS, a type of unscented Kalman filter, is implemented as forward and backward filters, followed by a smoothing step in the sensor node; the algorithm’s steps are listed on page 867 - 869 of the reference and include: 6 matrix inversions,  $3 \times N$  square-root operations for sigma-point calculation, when  $N$  is the number of iterations, and 11 matrix-matrix multiplications. The complexity of this SPKS method again exceeds our RNN LP method by a considerable amount.

### **5.2.8 Comparison with Particle filter**

The work by M. Peasegood, Proceedings of the IEEE International Conference on Mechatronics Automation Niagara Falls, Canada • July 2005 ” Localization of Multiple Robots with Simple Sensors ”, builds a particle filter localizer in a node (robot) with 800 particles to represent the state (coordinate) in each direction. The main 3 steps, done at each iteration are: 1 - sampling: basically

800 random variables have to be generated in hardware, or selected from a previous set of 800 (previous iteration). This requires a hardware random number generator, for a uniform or other distribution

2 - state update: the state of each particle (800 total) is updated to reflect the motion of the robots since the last iteration of the algorithm: this is vector-scalar multiplications, and vector-vector additions/subtractions

3 - Weighting: For each particle (800 total), a weighting is applied representing the degree of belief in the the position estimate of the particle. The weighting factor is computed using a scalar divider, and a scalar square-root module.

In conclusion, even though the PF method does not require matrix-inversion and Jacobian matrices, the large number of particles (needed for good approximation of the coordinates posterior distribution) makes the PF complexity quite high. Our RNN LP method has a lower complexity since at each iteration it does not have to generate and process large numbers (800 or more) of random variables for each x, y, z dimension.

# Part II

## Speaker Recognition

## CHAPTER 6

### SPEAKER RECOGNITION INTRODUCTION

This chapter provides an overview of closed (enrolled) set Speaker Recognition based on Gaussian Mixture Models, GMMs. Edge node (or IoT) devices are proliferating to all aspects of our lives. As our reliance on IoT devices is increasing, an easier way to interact with the devices is desired. Since humans communicate using speech, it is only natural to extend speech-based control to IoT devices. A necessary functionality in IoT devices for this is speaker identification (SI) for their speech-based control. However, power dissipation of typical SI is high since the number of computations in SI for even a small database of enrolled speakers are high, which limits SI in many low power IoTs.

This work addresses the challenge by discussing a novel microarchitecture for SI combining k-means clustering with Gaussian mixture model (GMM) scoring that reduces hardware complexity, operates within power constraints of typical IoT devices, and scales to a database of high number of enrolled speakers. The most recent SI systems, such as i-Vector-based (Campbell et al., 2006), can only meet a subset of the above constraints and usually require multi-core processors with large memory size and footprints consuming  $>10$  mW power in normal operation (Ramos et al., 2013). Adding new speakers in the approaches typically requires rebuilding the classifier, which makes it difficult to train them in an IoT device. Meanwhile, our approach consumes  $<2$  mW power in normal operation for twenty speakers (a typical smart home application) and achieves a

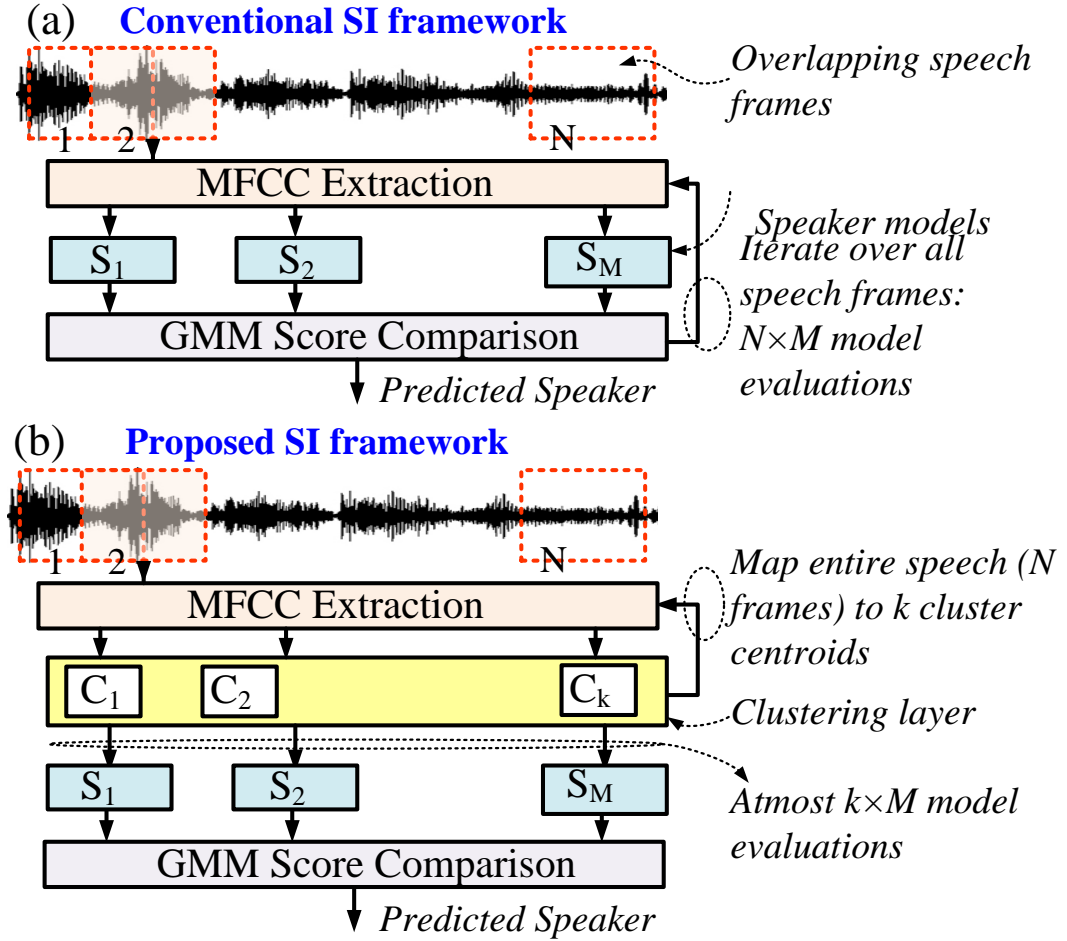


Figure 9: SI computations flow: (a) The standard architecture where all  $N$  frames of MFCC vectors are processed, for a total of  $N \times M$  GMM evaluations. (b) The proposed architecture adds a clustering layer (CL) to map MFCC frames to  $k$  centroids which reduces the total GMM evaluations to atmost  $k \times M$ .

comparable accuracy to the traditional approaches on TIMIT speech corpus (Reynolds and Rose, 1995).

Fig. 9 compares our approach against the conventional framework. SI pipeline in our approach consists of three computing layers: (i) extracting mel-frequency cepstral coefficients (MFCC) from an incoming audio stream, (ii) online k-means clustering of MFCCs, and (iii) GMM scoring of cluster centroids to compute the posterior probabilities of speaker models to obtain the predicted speaker. Note that compared to the conventional framework, mapping MFCC features to  $k$  centroids in our approach reduces the computing effort at the downstream GMM scoring. For example, to score  $N$  speech frames against  $M$  speaker models, the conventional approach requires  $N \cdot M$  GMM scores, while our approach requires  $k \cdot M$  ( $k \ll N$ ) GMM scores by compressing the feature variations to  $k$ -dimensions. However, our approach also adds an extra computing layer (clustering) in SI. Since MFCC extraction and GMM scoring in our approach are the same as the conventional, in this paper, we focus on the middle (clustering) layer. We discuss an energy efficient micro-architecture for the clustering layer (CL) applying pipelining and time-multiplexed operations in 45 nm CMOS technology.

## CHAPTER 7

### BACKGROUND IN POWER SAVING TECHNIQUES FOR GMM

#### CLASSIFIERS OF SPEAKERS

Several power saving techniques have been investigated on the original GMM classifier-based SI. In (Gianelli et al., 2019), look-up table free SI approach was discussed. In (McLaughlin et al., 1999), the number of speech frames evaluated against speaker GMM models are reduced such as only every  $n^{th}$  frame is scored. Variations of the approach are described in (Pazhayaveetil, 2008) as simple downsampling (SDS) and conditional downsampling (CDS). Unlike simply rejecting speech frames, our approach considers all speech frames but minimizes computing load by reducing their dimensionality by clustering. A variable frame rate (VFR) algorithm is presented in (McLaughlin et al., 1999). However, real-time determination of similarity between high-dimensional speech frames is expensive and becomes a bottleneck to power scaling in SI. Clustering of speakers training set features is also proposed in (Sun et al., 2003); however, batch clustering is used to prune out GMM models of less likely speakers. In contrast, our approach directly prunes the incoming MFCC frames and retains all speaker models.

There are CMOS implementations of k-means clustering for image segmentation (pixel clustering) applications and in real-time clustering of sensor data. We are not aware of a CMOS implementation of k-means clustering (Lloyd's algorithm) for speaker identification, which is a novel contribution of this work.



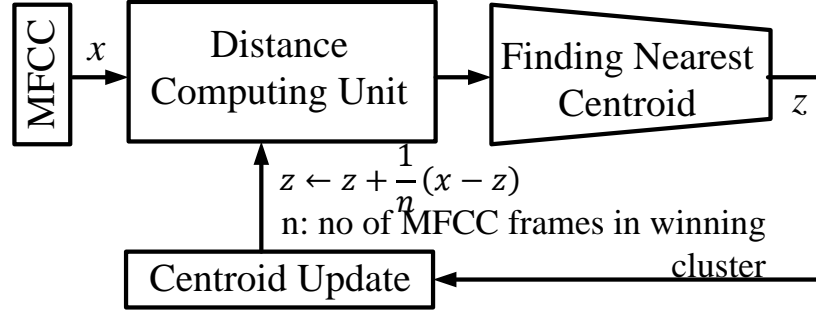


Figure 10: Overview of our clustering approach.

### 7.1 Complexity of online k-Means clustering (Lloyd's algorithm)

An upper bound of time complexity is  $O(k \cdot D \cdot N)$ , where  $k$  is number of clusters,  $D$  is dimensions of the data vectors,  $N$  is number of data vectors to cluster, (Har-peled and et al, 2005).

In the proposed SpkrRec architecture, the total number of computations, per iteration, is upper-bounded by  $O(D \cdot k + 2 \cdot D)$  additions,  $O(D \cdot k)$  multiplications,  $O(k \cdot \log(k))$  3-way comparisons, and  $O(D)$  divides where  $D$  is dimension of the data vector and  $k$  is number of clusters.

## CHAPTER 8

### SPKRREC DESIGN FEATURES AND PARAMETERS

*Parts of this chapter have been presented in (Iliev et al., 2019) and (Gianelli et al., 2019).  
Copyright © 2019, IEEE.*

#### 8.1 Online k-Means Clustering Architecture

We employ Lloyd’s on-line  $k$ -Means algorithm (Kinnunen et al., 2006) since it only updates one (closest) centroid at a time instead of all centroids which saves dynamic switching power. Fig. 10 shows a high-level computing flow of the algorithm. For an incoming vector, the algorithm computes its distance from the current centroids. The centroid with the minimum distance to the input vector is identified and updated. Subsequently, we discuss the micro-architecture of each building block in Fig. 10.

##### 8.1.1 Squared Euclidean Distance Computation Unit

The distance computing unit in Fig. 10 computes the distance of the input vector to the current centroid vectors. Fig. 11 shows our pipelined datapath for computing the distance between input  $x_t$  and centroid  $z_i$ . Each distance value is saved in the output register  $R_D$ . Our current implementation uses 6-bit quantization of the MFCC features. Test vectors are stored in the register file  $R_T$ . Centroids are stored in the register files  $R_C$ . The datapath is time-multiplexed in  $k$  time-slots to compute the distance between  $x_t$  and each of the  $k$  centroids. In the pipeline phase 1, parallel 16-bit subtractors compute the difference between  $x_t$  and the  $z_i$  for the current time slot. In phase

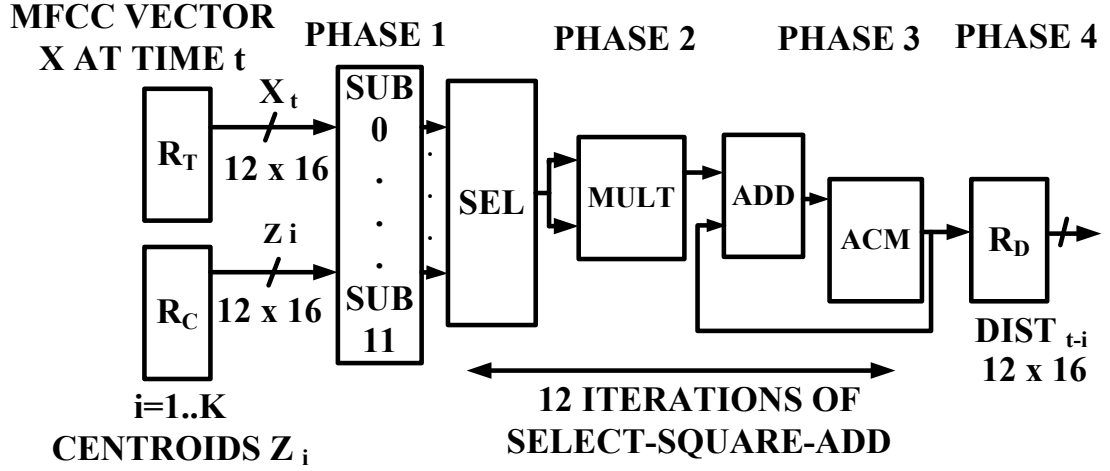


Figure 11: Datapath for computing the squared Euclidean distance between a new MFCC feature vector  $x_t$  and all  $k$  centroids  $z_i$ .

2, a single square unit computes the square of a difference vector element. In phase 3, a single adder adds the squared difference element to a single accumulator register. A dedicated controller schedules twelve iterations of phase 2 and phase 3 using 85 internal states in order to process all twelve elements of MFCC vectors. In phase 4, the final squared Euclidean distance value is sent to the Nearest Centroid Identification Unit.

### 8.1.2 Nearest Centroid Identification Unit

This block finds the smallest of  $k$  distance values stored in  $R_D$  in linear time using a systolic sorting array. The array architecture is shown in Fig. 12(a), and the basic cell architecture is shown in Fig. 12(b). In our current implementation, the array has  $k = 40$  symmetric processing elements. Data elements are received serially by the array, i.e., one per clock cycle. Data elements are sorted

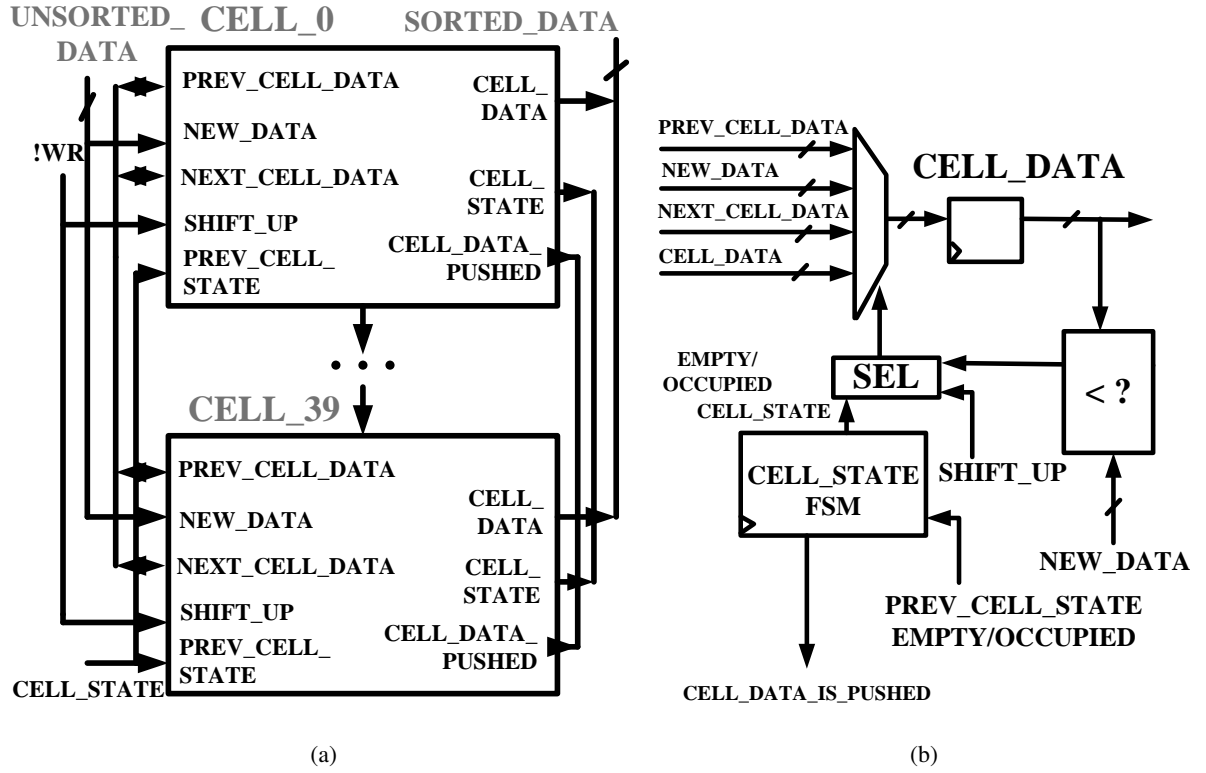


Figure 12: (a) Systolic array for sorting a serial input stream of  $M$  words in  $M$  clock cycles. (b) Processing cell in the array.

in parallel, while the next element is clocked in. After the  $k^{th}$  element has been clocked in, the array is already sorted with the smallest value at the top ready to be clocked out. A synchronous pipeline design is used in order to avoid switching activity of hierarchical trees of combinatorial comparators. The cluster index corresponding to the smallest distance  $D_{min}$  is also registered. Only the centroid corresponding to the smallest distance value is updated in the following update unit.

### 8.1.3 Centroid Update Unit

The main processing blocks are shown in Fig. 13(a) to implement the centroid update equation in Fig. 10. *CNT\_BLK* is a counter block with  $k$  counters, one counter per cluster to keep track of the number of MFCC frames in the cluster. *DEC\_RD\_RC* is a decoder of *INDEX\_CLUSTER* (the winning centroid index produced by the Nearest Centroid Identification Unit) and also performs the readback of the corresponding centroid from the register file  $R_C$ . *SUB* is a block with twelve parallel subtractors for finding the difference between the twelve element 16-bit centroid and the incoming MFCC frame. *SER\_DIV* is a serial divider block, used twelve times to divide each of the twelve output elements from *SUB* with the current value from *CNT\_BLK*. *ACCUM* is a 16-bit accumulator to add the new updated value from *SER\_DIV* to the centroid's previous element value. A dedicated controller schedules twelve iterations of *SER\_DIV* and *ACCUM* to update all twelve elements of the selected centroid from  $R_C$ . After the update is complete, the new centroid is written back into  $R_C$ . The bit-serial divider architecture is shown in Fig. 13(b).

Notably, k-means is an unsupervised algorithm that groups a dataset into a user-specified number ( $K$ ) of clusters. To determine the optimal  $K$ , we vary  $K$  over a range from 10 to 50, ie. the "elbow method", and for each case plot the sum of squared error (SSE) estimated using the Euclidean distances between centroid estimate and data points. The plot typically looks like an arm, and the "elbow" is the value of  $K$  that is best, with diminishing SSE returns for any further increase in  $K$ . In our implementation, we have done off-line "elbow" analysis to determine that  $K=30,40,50$  will produce small SSE for the sets of speakers we consider. The hardware is given one of these  $K$  values and then has to update (iterate for each test frame) the estimates of the  $K$  centroids. An

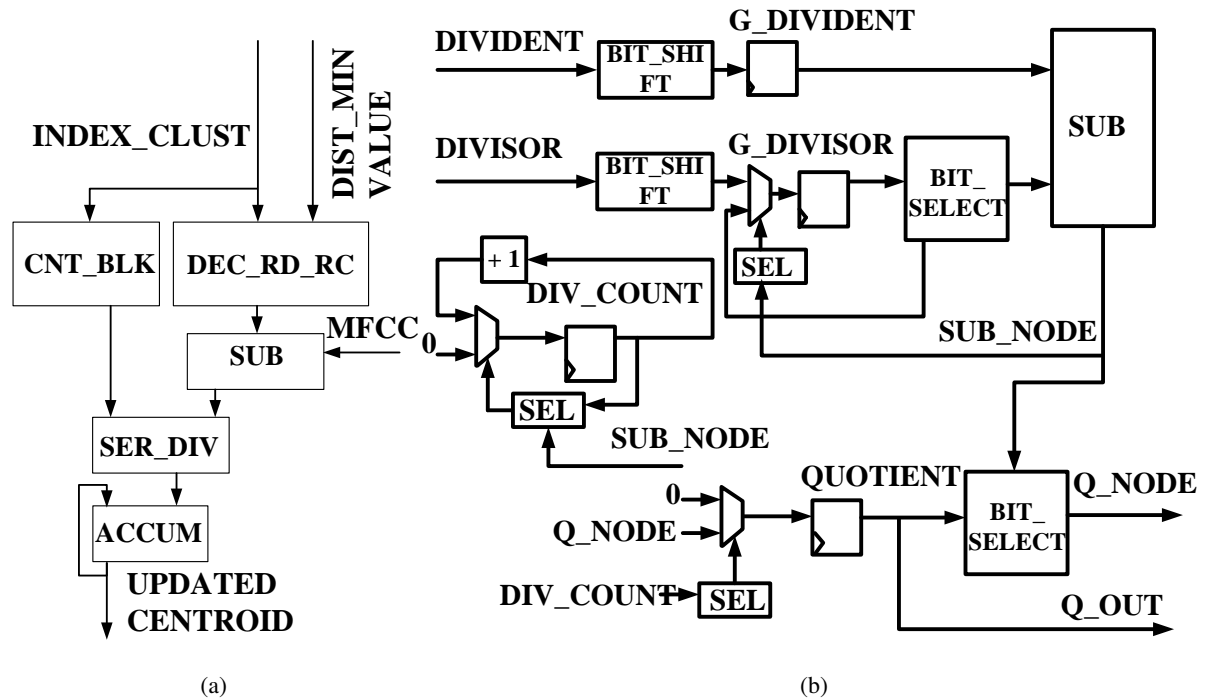


Figure 13: (a) Centroid update unit for updating only the winning centroid. (b) Bit-serial divider for dividing an element from *SUB* by the current value from *CNT\_BLK*. Only the quotient is used in the downstream processing.

example of these updates, for centroids 1 and 4, is shown in Fig. 14(a). Using the "elbow" plot, the optimal K is determined. Our current approach to determine the optimal number of cluster only considers the accuracy. In future research, we will expand the approach to consider energy constraints for the hardware in addition to accuracy constraints.

## CHAPTER 9

### SPKRREC SIMULATION AND IMPLEMENTATION RESULTS

*Parts of this chapter have been presented in (Iliev et al., 2019) and (Gianelli et al., 2019).  
Copyright © 2019, IEEE.*

#### 9.1 Simulation Setup

The speaker identification microarchitecture described in this paper was evaluated on TIMIT corpus (Garofolo et al., 1990). MATLAB GMM models were trained using SI and SX database for 38 speakers with 1344 utterances from their SA1 and SA2 sentences. MATLAB model of the MFCC clustering and the following GMM maximum-likelihood classifier was developed using the training data set and was the reference model for the fixed-point Verilog implementation. ModelSim was used for Verilog simulations of CL. The CL stimulus was collected from functional simulations which generated 500 MFCC test frames from each test speaker. In real-time test mode, we want to keep K as small as possible to reduce the amount of time for cluster building. The accuracy of the proposed method is sensitive to the number of clusters (K) chosen for a given number of speakers. In Fig. 15(a), we used TIMIT-based simulations to determine the optimal K for a fixed N, by varying K from 10 to 50. The results show that for K above 30 produce success rate more than 90%, and for K lower than 10, the accuracy is below 80%.

We implemented the CL design with the NCSU PDK 45 nm standard cell library and the Cadence EDI tool flow. One MHz processing clock was used for the synthesis constraint resulting in a netlist of 75741 standard cells. Switching activity for dynamic power analysis was obtained from

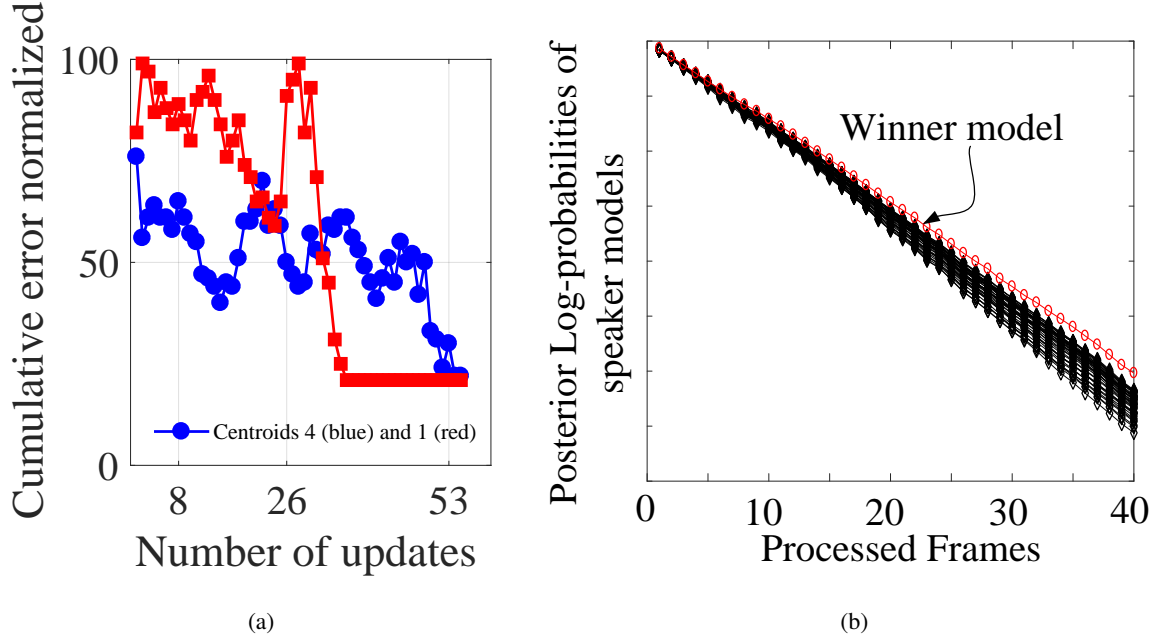


Figure 14: (a) Convergence of the cumulative error for centroids 1 and 4. Note that the residual value of 20 is reached after 35 frames (iterations) for centroid 1 and after 54 frames for centroid 4. Different centroid estimates converge to their residual values for a different number of frames. (b) Evolution of posterior log-probabilities of 38 test speakers.

ModelSim simulations. The proposed design dissipates 0.637 mW (not including register files) when estimating forty centroids from a speaker's 500 MFCC test frames. Total power dissipation is 1.6 mW if the register files are included. This can be reduced significantly if SRAM modules are used for storage.



## 9.2 Functionality Characterization

We have simulated in fixed-point Verilog forty centroid estimates, over 500 MFCC test frames for each test speaker, and compared them to their floating-point values as computed by MATLAB-based clustering function. Fig. 14(a) shows an example for centroids 1 and 4 with the cumulative error for all twelve elements of each vector. The error is defined as the difference between MATLAB-based value and the final (after the 500<sup>th</sup> frame) value from the Verilog simulation. The error is also normalized so that the largest value between fixed-point and floating-point representations is hundred. As can be seen from the figure, a small residual error of 20% remains for these two centroid estimates. The residual error is also less than 20% for the remaining centroid estimates. Convergence is detected by using a `Cluster_Threshold` register (user writable), one for each cluster; if the cumulative error remains below the threshold, further iterations for the cluster's centroid will be ended.

Fig. 14(b) shows the simulated evolution of the posterior log-probabilities of 38 test speakers when scored for different numbers of fixed-point centroids. The red trace in the figure shows the winning (most-likely) speaker and the black traces are the remaining speakers. After CL, GMM-based SI is simulated functionally using MATLAB.

## 9.3 Energy Efficiency Characterization

Fig. 15(a) shows the simulated recognition success rate for different numbers of test speakers and centroids estimated in each speaker's test frames. As can be seen from the figure, the more speakers we add to the system, the more centroids are needed to maintain a desirable success

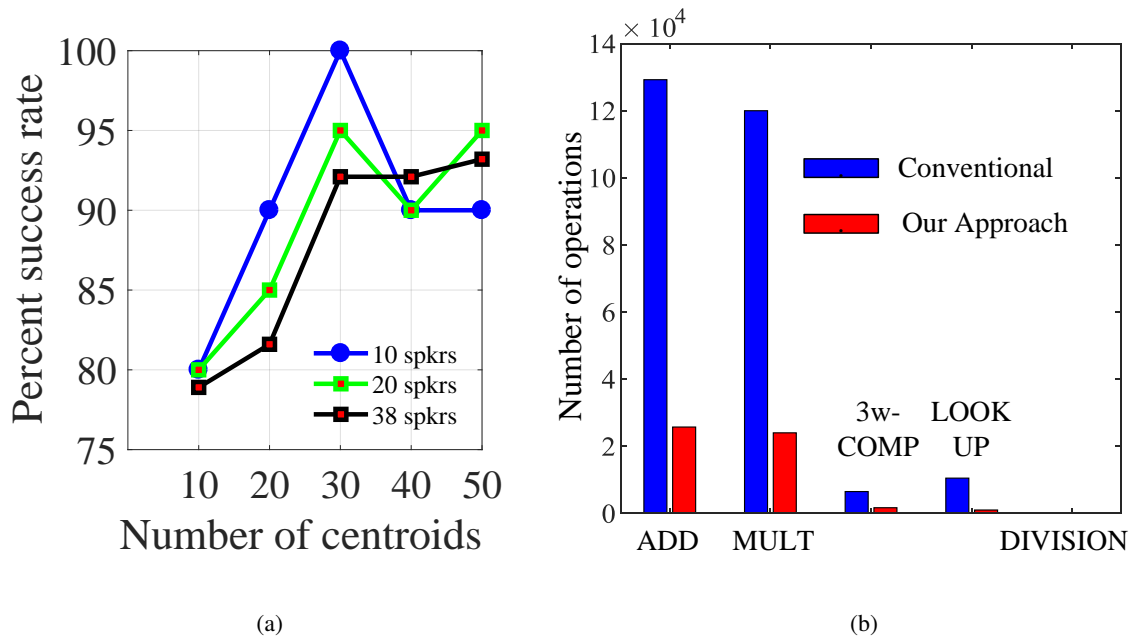


Figure 15: (a) Success rate vs. number of speakers and number of centroids per speaker - 10 speakers (blue) 20(green) 38(black). (b) Reductions in number of operations: blue for GMM scoring with non-clustered frames; orange for the proposed GMM scoring with forty centroids.

rate, eg. over 92%. However, the number of necessary centroids is still far less than total number of speech frames (5000 for  $\sim 5$ s speech test). Note that for 20 centroids, the set of 38 speakers achieves 82 % success rate, while the sets of 20 and 10 speakers achieve 85 % or higher success rates. Therefore, to achieve higher success rate for high number of speakers requires more centroids. Currently, the results are shown at coarsely varying number of centroids (K). Future research will explore this aspect in more detail and study the optimal dependence of K to the

number of speakers. In Fig. 15(a), each point in the blue (10 speakers), green (20 speakers), and black (38 speakers) traces was computed with 500 updates of the centroid estimates. If more updates are used, the centroid estimation can be more accurate, resulting in better success rates; however, energy consumption also increases. Therefore, considering the low power constraints in our design, we limit the number of updates to 500 in our analysis. In our current methodology, we vary the number of centroids ( $K$ ) from 10 to 50 and plot the sum of squared error at varying  $K$ . In our future work, we will explore an algorithm to automatically predict the optimal  $K$  and number of updates given the SI parameters.

More centroids in our SI approach also incur more energy consumption. To balance energy consumption and SI accuracy, we configured CL for 30 centroids for 10 and 20 speaker sets, and for 40 centroids for a 38 speaker set. The success rate for 10 and 20 speakers starts to decrease when the number of centroids exceeds 30. This indicates that the centroid estimates, for 10 and 20 speakers, were biased after 500 iterations, but still provided at least 90% success rate. The biasing of the centroid estimates can occur since k-means clustering is known to be sensitive to outliers in the data.

The success rate for 10 and 20 speakers starts to decrease when the number of centroids exceeds 30. This indicates that the centroid estimates, for 10 and 20 speakers, were biased after 500 iterations (frames), but still provided at least 90 % success rate for these two sets of speakers. The biasing of the centroid estimates can occur since k-means clustering is known to be sensitive to outliers in the data. In our approach we avoid this case by configuring the CL hardware to 30 centroids for the 10 and 20 speaker sets. Such instability of the centroid estimates typically

results when the dataset is not naturally separable into  $K$  clusters but we are attempting to find  $K$  clusters in the dataset. For large  $K$ , k-means clustering can be sensitive to outliers in the data and biased centroids can result. Lloyd’s clustering as well as Ward clustering and cosine-PLDA clustering can exhibit this; see Fig. 3 in “Hierarchical speaker clustering methods for the NIST i-vector Challenge” by E. Khoury et al. where Ward and cosine-PLDA clustering shows increasing success rate (decreasing FRR error rate) for 101000 to 161000 clusters; then shows decreasing success rate from 161000 to 301000 clusters.

In Fig. 15(b), the reduction in the number of multiplications, additions, 3-way comparisons, and table lookups (GMM log and exp processing) in our approach are due to the much smaller number of frames (only centroids). The division operation is unique to the CL, and we discuss its overheads subsequently.

Fig. 16(a) shows the dissipated energy in mJ as the number of test speakers grows for the standard GMM approach (blue trace) vs. our centroid-based approach (red trace). The standard approach consumes  $T \cdot N \cdot M \cdot P_{GMM}$  energy, where  $T$  is 10 msec frame duration;  $N$  is the total number of frames, 500 in our case;  $M$  is the number of speaker models; and  $P_{GMM}$  is the total power for scoring a GMM model. We use power dissipation value of 1.24 mW per MFCC vector element from (Kai, 2009) for a CMOS 65 nm implementation which we normalize for CMOS 45 nm used in our implementation to estimate the power of MFCC and GMM scoring units. Our  $k$  centroid-based approach reduces GMM scoring power to  $T \cdot K \cdot M \cdot P_{GMM}$  where  $k \ll N$ . It adds a negligible overhead of  $T \cdot N \cdot P_C$  of dissipated energy for estimating the  $k$  centroids from  $N$  frames.  $P_C$  is the power of the proposed centroid estimator architecture and is 0.637 mW as extracted from

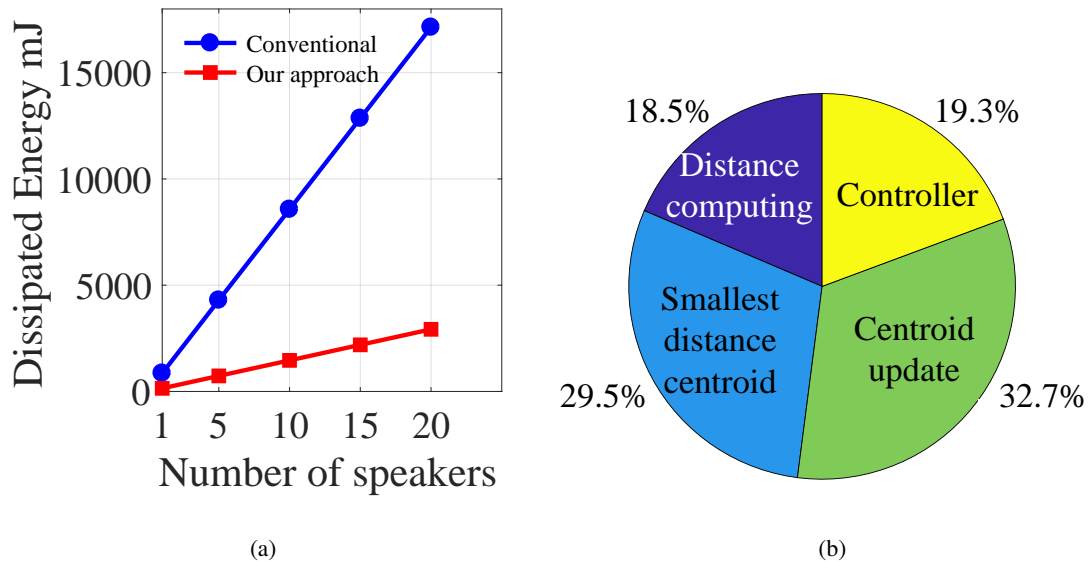


Figure 16: (a) Energy dissipation vs. number of speakers. Blue trace is scoring all 500 MFCC test frames with all GMM models. Red trace is scoring forty centroids with all GMM models. (b) Breakdown of power dissipation for the centroid estimator architecture.

our implementation. Therefore, we observe a 6 fold (from 8569 to 1462.2 mJ) decrease in the dissipated energy for classifying among ten speakers. Fig. 16(b) shows the breakdown of power dissipation for each major block in the proposed architecture.

#### 9.4 Comparison to alternative frame reduction approaches

In Fig. 17, we compare our implementation to an alternate LBG  $k$ -means implementation for frame reduction and to fixed rate frame skipping as described in (Kinnunen et al., 2006). The data in Fig. 7 in (Kinnunen et al., 2006) is reproduced in the figure. We focus on the decimation and

clustering-based methods since averaging and random-subsampling have not shown acceptable performance with the TIMIT corpus of speaker data as shown in (Kinnunen et al., 2006). In the frame-skipping approach, extending frame skipping to greater skipping frequency results in even worse error rate (Woszczyna, 1998). The LBG clustering method proposed by (Kinnunen et al., 2006) also prunes speakers with low probabilities and therefore achieves a slightly better performance than frame-skipping after running for more than 50 ms. Similar to (Kinnunen et al., 2006), a non-monotonic decrease in the error rate is observed in Fig. 17 due to randomness of MFCC features. Some MFCC features are “outliers,” much farther away from the cluster’s centroid; therefore, updates based on the features shows non-monotonicity in error trends. As can be seen from Fig. 17, our method provides the best error rate at the smaller test lengths by considering all speech frames but only lowering their dimension by online clustering. Thus, our approach can further minimize SI energy by only considering a smaller spoken segment.

## 9.5 CONCLUSION

We have discussed a novel low-power architecture for SI by combining  $k$ -means clustering and GMM scoring of the cluster centroids. To minimize overhead of the additional CL for SI, we have discussed novel Euclidean distance computing and online centroid updating low-power micro-architectures. Our implementation has achieved a  $6\times$  decrease in total dissipated energy for classifying among ten speakers with an integrated GMM-centroid scoring system.

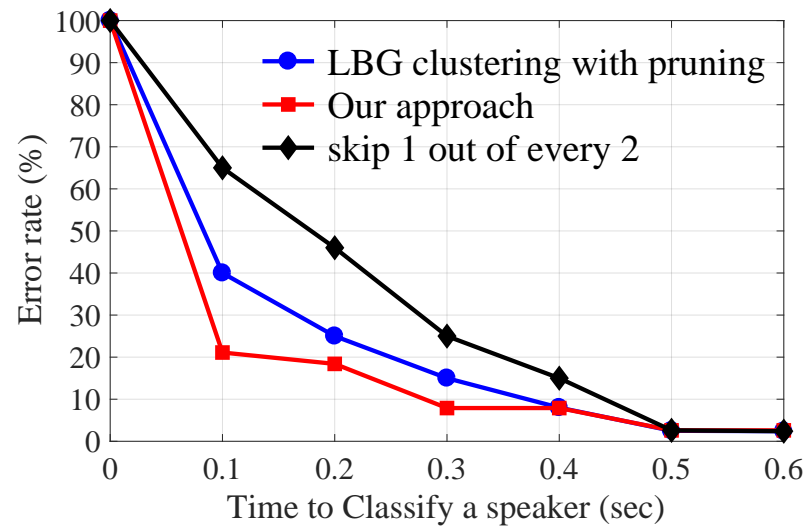


Figure 17: Error rate vs. time to classify a test speaker.

## Part III

# Fully Connected Layers in DNN



## CHAPTER 10

### INTRODUCTION TO ACCELERATION OF FULLY CONNECTED LAYERS

An important problem in real-time integrated hardware-software implementations of devices at the edge of the cloud is low-latency evaluation (inference) of fully connected (FC) layers for neural-network processing performed within the device. Example applications include deep CNN processing such as AlexNet or VGG-16, where a typical FC layer such as FC8 has 4096 input features and 1000 output neuron activations. Another application is bounding-box object localization in an image using reinforcement learning for training and a Q-Network for inferencing with several FC layers. Typical commercial neural network hardware accelerators, such as Intel's Movidius and Google's TPU dedicate specific micro-instructions and micro-architectural processing resources for FC layer evaluation. GPUs and multi-core SoCs rely on software implementations of FC layers and are therefore limited in the amount of parallelism that can be exploited from the FC layer's structure. ASIC and FPGA based FC implementations, as accelerators for GPU and/or SoC, allow the greatest amount of parallelism in FC implementations and is our proposed approach. FC layer evaluation is usually a dense matrix-vector multiplication problem of considerable size and a very large number of weights, in the 1 million to 100 million range. External DRAMs are typically used for weights storage and therefore DRAM read accesses are the major contributor to FC evaluation latency (limited DRAM bandwidth) and to power consumption for each inference pass (Sze and et al, 2017). This is one area for improvement in FC evaluation. We

solve the bandwidth problem by using High Bandwidth Memory, HBM, as shown in the following chapters. As an example, AlexNet has an FC8 layer with 4096 input neurons and 1000 outputs, which is similar to the FC8 layer in VGG-16 with 4 million weights. An HBM2 (Flashbolt from Samsung) can easily store all the weights in its 16 GB of paged banks. It has been shown (Sze and et al, 2017) that dense FC layer evaluation is a major contributor to latency during CNN and DNN inferencing, when compared to the initial sparse convolutional layers. Therefore recent research has focused on hardware acceleration of FC layers in particular. Fig. 18 shows such an FC layer which is the focus of our work.

The evaluation of the FC layer in the figure, for one vector of input features, is formulated as a matrix-vector multiplication problem as shown in Fig. 19 .

Hardware acceleration of DNNs has typically focused on both convolutional (CONV) and FC layers. This imposes some restrictions on the micro-architecture which has to handle both sparse CONV specific kernels, as well as dense, weights based FC layers. Yuran et al. (Yuran and et al, 2017) accelerate FC and CONV layers with a common processing element (PE) which is based on a matrix multiplier. Convolutions are unrolled to matrix multiplications for the PEs to process. The same PEs have to accelerate the FC layers as well which can create a resource contention problem. Our solution differs from this approach since we have PEs dedicated to the FC layers only, and the sizes of the FC weights tiles (sub-matrices) are not dictated by CONV kernel and loop-unrolling considerations. Instead our PEs are optimized to reduce latency processing of the FC layer and minimize number of passes to process the entire FC layer. Jiantao et al. (Jiantao and et al, 2016) propose to compress the FC layer weights by using Singular Value Decomposition

(SVD) This approach may not always work since SVD may not exist or be numerically stable for some large FC weights matrices. In his implementation PEs are shared for CONV and FC processing and are not optimized for FC layers specifically as in our proposed FC-Accel architecture. Ning et al. (Ning and et al, 2016) present a global summation architecture to completely replace the matrix multiplications in the FC layers. A mathematical identity replaces multiplications with accumulators for each feature map. This places a large hardware resource requirement for FC layers with large feature maps; only small image sizes of 32x32 have been processed with the global summation method. In contrast, our FC-Accel can handle FC 25088-4096 feature layers in VGG16. (Huimin and et al, 2016) propose an accelerator PE for both CONV and FC layers, with a batch-based computing method for the FC layers only. This differs from FC-Accel which operated on the entire FC layer (all feature maps) and uses tiles (batches) only for the weights matrix. Their solution also has to apply two different computing patterns on FC layers which is not needed in our approach: FC-Accel uses the same computing pattern for all FC layers. (Li, 2018) proposes a PE architecture for matrix-vector multiplication in FC layers. An entire row of weights is fetched from off-chip memory for the PEs to process. FC-Accel fetches only tiles (sub-matrices) of weights from a given column for all PEs to process and processes all rows simultaneously, column by column. The recent NVIDIA Volta GV100 architecture (NVIDIA, 2018) uses Tensor Cores for matrix arithmetic. HBM3s (JEDEC, 2020) are used for weights and data storage. Each Tensor Core can complete 64 floating point mixed-precision operations per clock. FC-Accel computes 128 16-bit fixed point operations per clock. The CNAPS ASIC (Hammerstrom, 1990) has a SIMD architecture with an array of 16x8 scalar multipliers for matrix-vector

multiplication, MVM, while FC-Accel uses 8x8 or 16x16 arrays of scalar multiplier for MVM. The DianNao series of ASICs (Chen and et al, 2014) implement an array of 64 16-bit integer MACs. FC-Accel uses 128 16-bit fixed-point MACs instead. The DaDianNao and ShiDianNao ASICs (Du and et al, 2015) store all weights on chip (eDRAM or SRAM) while FC-Accel uses on-chip HBMs with silicon interposers for storing weights for all FC layers and input features to these layers. Google's recently announced Edge Tensor Processing Unit, Edge TPU, (Google, 2020) uses up to 65536 8-bit MAC units which limits forward inference to 8-bit precision. HBM is also used for weights and features storage. By contrast, FC-Accel maintains 16-bit fixed point precision in forward inference passes. The recently described EIE ASIC (Han and et al, 2016) accelerates both CONV and FC layers by using compression to derive a compressed network model. The resulting matrix-vector multiplications are of smaller dimensions however an 800 MHz processing clock is needed to achieve 102 GOPS for FC8 layer processing. In comparison, the proposed FC-Accel needs an 662 MHz clock for FC8 processing and achieves 1048 GOPS without using compression. This performance improves on the Tetris DNN accelerator which also uses 3D memory (Hybrid Memory Cube, HMC, similar to HBM) (Gao and et al, 2017). The published Tetris performance for 16 3D engines (14 x 14 PEs) and 16 HMC vaults is  $16 \times 39.2$  GOPS = 627.2 GOPS. which is 40.15 % less than FC-Accel's performance.

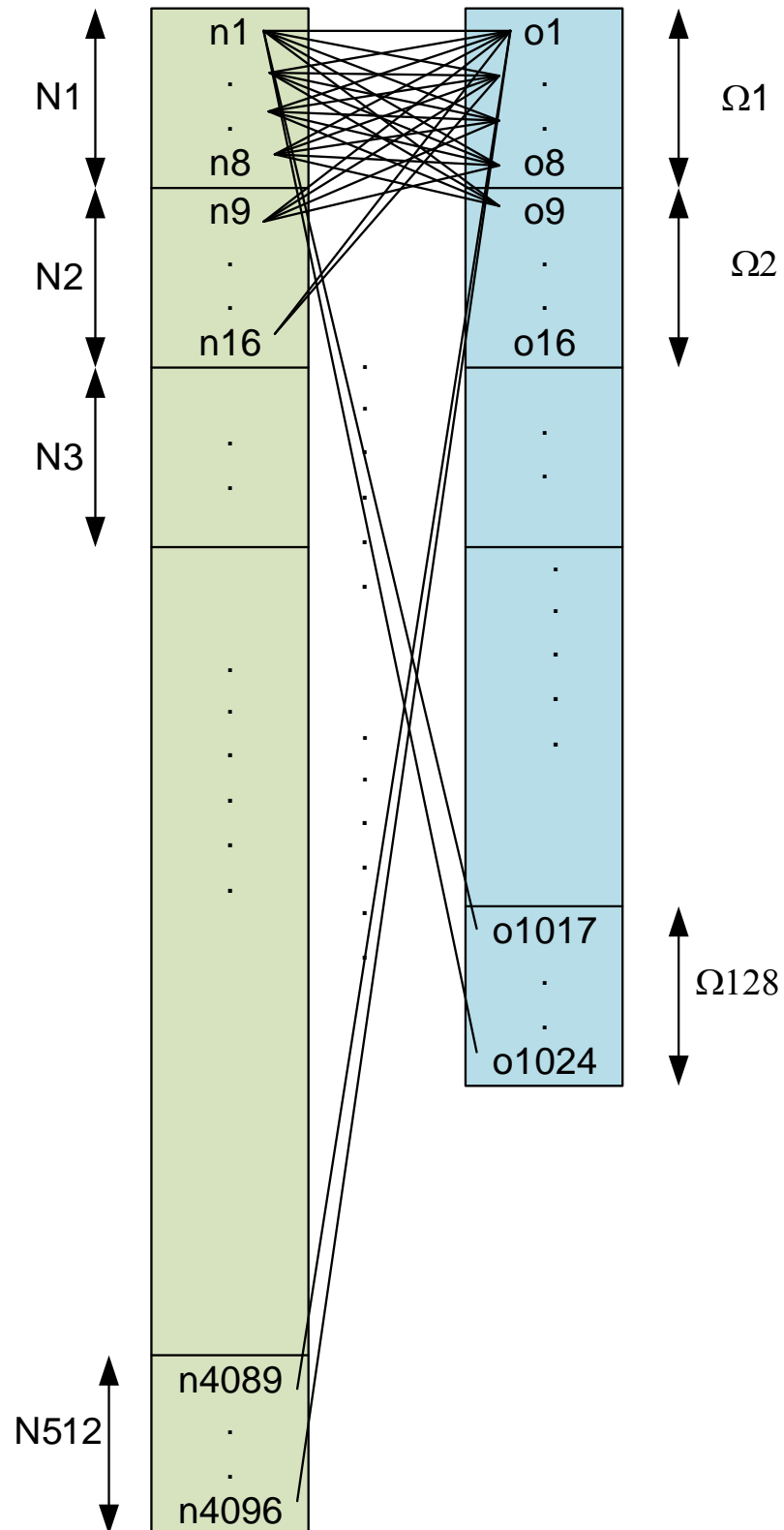


Figure 18: Fully Connected FC8 layer in AlexNet or VGG-16: 4096 input and 1000 outputs. Groups of 8 are indicated in the input and output vectors respectively.

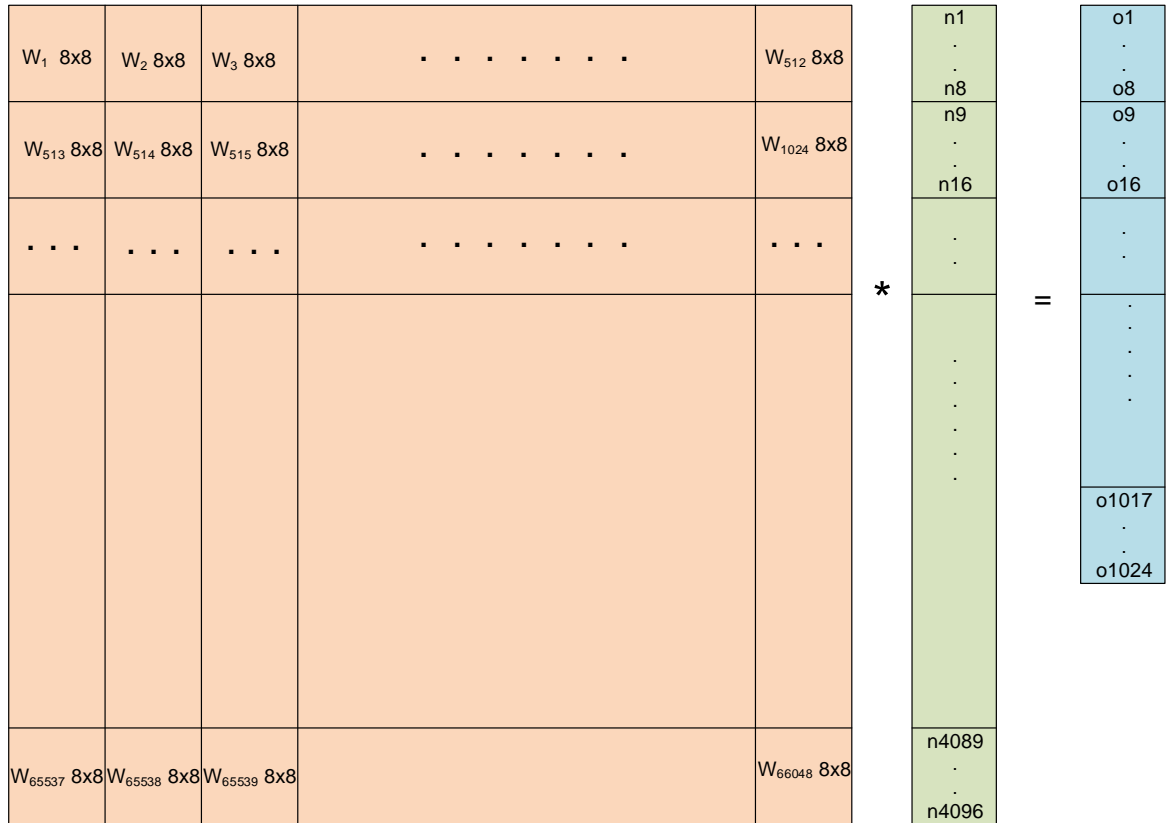


Figure 19: The equivalent matrix-vector multiplication for the FC layer in Fig. 1. The weights are grouped in 8x8 sub-matrices (tiles)  $W_1$ ,  $W_2$ , etc. Each column of sub-matrices is mapped, during its time-slot, to a set of 128 MACs and 128 PEs. This is the columnwise block(tile) decomposition used in FC-Accel. The same set of MACs and PEs is reused for all 512 time-slots during processing.

## CHAPTER 11

### BACKGROUND ON FULLY CONNECTED LAYERS IN CNNs

A typical CNN consists of several types of layers as shown in Fig. 20. Each CONV layer processes 2D input features with 2D convolutional kernels and produces stacks (in the 3rd vertical dimension) of 2D output feature maps. Each CONV layer is followed by a pooling or downsampling layer. The function of the downsampling layer is to reduce the spatial size of its input to a smaller size. As shown in the figure the last downsampling layer, S4, is followed by a fully connected, FC F5-F6 layer which receives a 1D vector of 120 scalar values (features) generated by the last downsampling layer S4. The FC layer's input is F5 with 120 inputs. The FC layer's output is F6 with 10 output features. The number of weights in the FC F5-F6 layer is  $1200 = 120 * 10$ . The output features are then mapped to a set of 10 output classes. This can be done in a variety of ways, for example using a SoftMax activation function which is not shown. Notice that processing 1200 16-bit fixed-point multiplications can incur a FC processing latency which grows with the FC layer size: as we will see later, some FC layers in large CNNs can have between 4 million to 103 million fixed-point multiplications. These multiplications can have few zero terms and are non-sparse or dense. The resulting processing latency for all FC layers can be very large, especially if it is all done in software.

The output  $y^l$  for a fully connected layer  $l$  is mathematically represented as shown in the equation

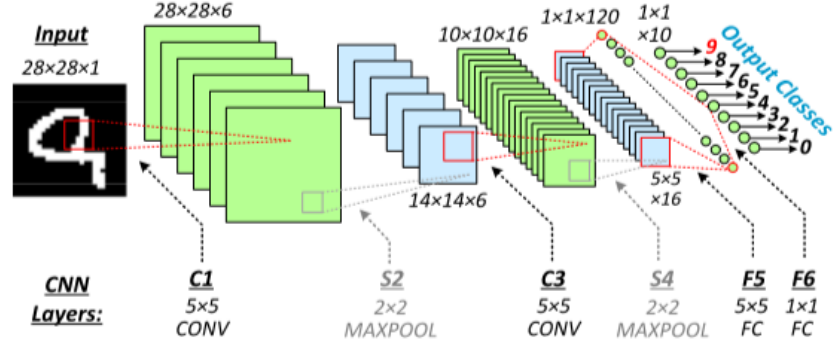


Figure 20: Fully Connected layer FC F5-F6 in a typical CNN. F5 is a 1D vector of 120 input scalar features and F6 is a 1D vector of 10 output scalar features.

$$y^l = s(y^{l-1} * W^l + b^l) \quad (11.1)$$

where,  $W^l$  and  $b^l$  are the weights and the bias vectors of the layer  $l$ , and  $s$  is the activation function used. (Alecci, 2017) .

Notice that a weight for a single connection between an F5 input and an F6 output is not shared (reused) with any other connection. This is unlike convolutional layers, where weights (parameters) can be shared accross several input and output connections. Because of non-sharing of weights, FC layers require substantial storage memories for all their weights and lots of training data and training time for the off-line learning of these FC weights. In addition, low latency weights read out from the large memories requires high bandwidth and 3D stacks of DRAMs such as wide-IO, HBM, and HMC memory, has been specifically developed to address this high band-



width requirement. In the following chapters on FC-Accel we show our HBM-based architecture for overall latency reduction of FC processing.

Pooling layers don't have weights to be learned during training. They consist of filters that slide, with a preset stride value, across its input layer and a reduction operator such as max or average.

Recent developments of 3D stacks of SDRAMs (synchronous DRAMs also referred to as DRAMs below) include wide-IO memory, High Bandwidth Memory (HBM), and Hybrid Memory Cube (HMC) memory. They all improve on the basic DDR4 SDRAM limitation of a 64 bit physical bus width. Wide-IO is targeted at smartphones, and has a maximum physical bus width of 512 bits, starting at 128 bits. HBM is targeted at high performance graphics engines (GPUs) and general purpose computing. It has a maximum physical bus width of 1024 bits in one stack, or 4096 bit bus width in four stacks. It's also a recent JEDEC235 standard, with HBM2, HBM3, and HBM4 as latest standard editions, each one reducing power consumption from the initial 3.2 W (HBM1) of power consumption with a 500 MHz bus clock. HMC is targeted at high-end servers in the cloud and has been adopted by Intel and Micron for their high-end server chipsets. It uses high-speed serial data links in order to implement bus widths of 4096 bits or more and uses parallel-to-serial and serial-to-parallel converters on both ends.

Fig. 21 shows 8 DRAM stacks, each providing a 128 bit physical write/read bus interface. The combined bus width is 1024 bits as shown on the Silicon interposer layer. The combined bus speed can be 500MHz. The JEDEC235C HBM standard does not show a physical bus of 1024 bits but 8 128-bit transactions over a 128 bit bus. User logic, such as our DPR\_FIFO\_BUF block described later (part of each PE), has to assemble 1024 bits from the 8 128-bit transactions. The

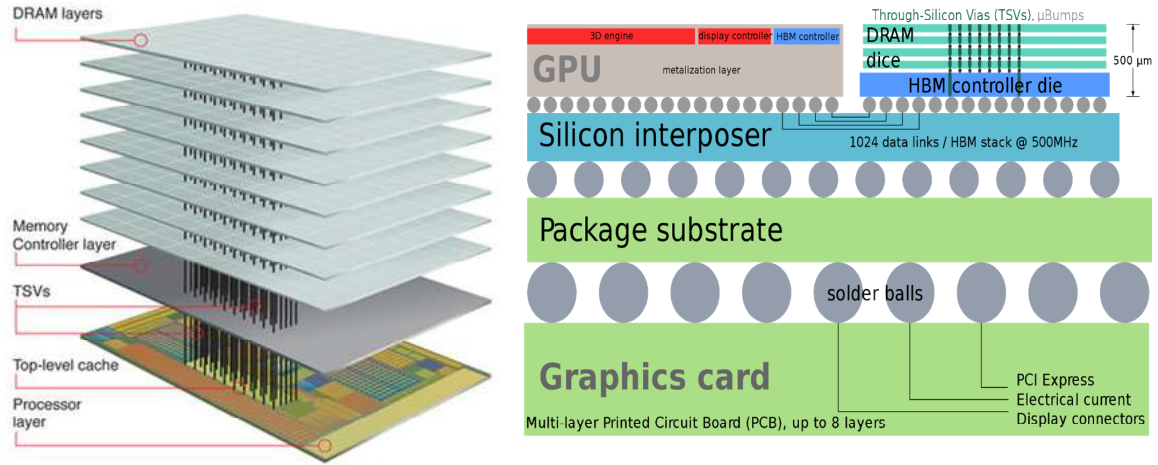


Figure 21: HBM 3D SDRAM memory stack technology (left) and GPU or general purpose (FC Accelerator instead of GPU) application (right).

block can be implemented in the silicon interposer layer, or in the PE (GPU replaced by PE in the diagram).

We have chosen HBM memory for our FC layer accelerator since it can be integrated with the 3D/2D localization (SSL-Accel) and speaker recognition (SpkrRec-Accel) accelerators in one mobile robotic platform. A smartphone form factor does not support the mobile robotic requirement, for example it does not have a robotic arm. Further integration between the 3 accelerators can allocate unused memory in the HBM stacks between the accelerators, as required, and under the supervision of the real-time OS memory manager.

The matrix-vector multiplication problem for very large matrices has been the subject of intensive research since the advent of parallel multi-core processing systems and parallel programming techniques for them, see (Quinn, 2004). Sequential (non-parallel) matrix-vector multiplication (2 nested loops) has a time complexity of :

$$O(m * n) \quad (11.2)$$

for multiplication of  $m \times n$  matrix and  $n \times 1$  vector. The number of scalar multiplications and additions is

$$O(m * 2(n - 1)) \quad (11.3)$$

To improve on this, parallel matrix-vector multiplication decomposes matrix  $m$  using three types of decompositions (Quinn, 2004):

- rowwise matrix decomposition (or rowwise block striped decomposition)
- columnwise matrix decomposition (or columnwise block striped decomposition)
- checkerboard matrix decomposition (or checkerboard block decomposition)

In each case the decomposition generates sub-matrices of smaller dimensions which are then mapped to processing tasks, or processing units (PUs). A communications network is assumed between all PUs, and message-based communication protocols are required such as all-gather, all-to-all, and 2D grid collective communications.

We have chosen columnwise matrix decomposition (or column of tiles (blocks) decomposition) for the proposed FC-Accel architecture due to its high performance as shown in (Quinn, 2004). It also has a simple communications protocol which is easy to implement with a 1D array of HBM memories mapped to a 1D array of processing elements (PEs).

A lower bound on the execution time is

$$Tp(calc) = [n * (2 * ceil(n/p) - 1) + n] * d + comms - T \quad (11.4)$$

where  $d$  is the execution time of a basic scalar operation (multiplication or addition)  $p$  and  $p$  is  $m/t$  PEs.  $comms - T$  is time for communications between PEs to exchange input data, weights, and status message exchanges between PEs; we don't status messages in FC-Accel, and can set  $comms - T$  to 1 cycle for HBM read out and 1 cycle for partial result accumulation in a PE. In the following we will expand on the implementation of column of tiles decomposition of the weights matrix  $m$ .

## CHAPTER 12

### DESIGN FEATURES AND PARAMETERS OF FC-ACCEL

FC-Accel solves the DDR4 SDRAM's limited (64-bits) bandwidth constraint by storing all FC layer weights in Hight Bandwidth Memory (HBM, see JESD235A/B/C standard (JEDEC, 2020) ) in order to maximize the memory bandwidth of each read-out access from the weights memories. In this first design exploration study, we consider 128 HBMs driving 128 PEs (1D array of PEs). The HBM read-out bus is 128 bits wide which allows the read out of 64 16-bit weights for each PE's matrix multiplier in two read requests (two column addresses) resulting in 8 128-bit bus cycles, using the BL4 mode of the memory. The timing is shown in Fig. 23. Fig. 22 shows a high-level view of the proposed architecture. It implements a column of tiles decomposition of the original weights matrix. The 128 HBMs connect to each PE's data-prefetch and on-chip buffer unit, DPR-BUF. This unit schedules a stream of two reads to two sequential column addresses so that a stream of 8 128-bit read bus cycles is generated. The read-out data for each set of 8 128-bit cycles is stored in 8 FIFOs inside DPR-BUF, one FIFO per Da or Db sequential transaction in Fig. 23. The FIFOs match the HMB bus rate, 500MHz (typical rate for GPU applications), to the PE's 662MHz processing clock rate. In a following section we show our pipelined PE design which runs at 662 MHz. For this choice of clock rates, a FIFO depth of 5 is sufficient. The 8 FIFOs in DPR-BUF then drive a 1024 bit on-chip buffer in DPR-BUF. This on-chip buffer is then transfered to its PE in 1 cycle. The 128 PEs are reused in each of the 512 time slots which map

to the 512 columns of Fig 19. The weights matrix in Fig 19 is broken up in 8x8 tiles of weights, which dictates the 8x8 PE design as well as the DPR-BUF's 1024 read-out on-chip buffer from the PE's HBM. Accordingly the input data is divided up into tiles of 8 elements each. Input data is stored in an HBM\_IN and is read out in 1 clock cycle, overlapped with the read-out cycle from weights memories HBM1 to HBM128. Other tile sizes, multiples of 8x8, are therefore possible for example 512 columns and 512 rows (square matrix in Fig 19 ) or 4096 inputs and 4096 outputs (FC7 in AlexNet and VGG16), 25088 inputs and 4096 outputs (FC6 in VGG16) and so on. The following sections detail the micro-architecture of each PE sub-block.

In this first design exploration study with a 1D array of 128 PEs, our choice of an HBM dedicated to each row of PEs avoids the need for complex 2D mesh routing and network-on-chip, NoC, hardware required to implement the routing infrastructure, as described in (Kim et al., 2016).

Notice that off-line training may produce several sets of weights (for several training optimality criteria) which can be stored in different pages in each HBM. During real time operation, between inferencing passes, a new page may be selected in some or in all HBMs and the FC layer will use a new set of weights for the next inference pass. Therefore HBM-based weights storage allows dynamic (real-time) weights selection between inference passes.

### **12.1 HBM Data-Prefetch Unit and On-chip Buffer, DPR-BUF**

The weights tiles stored in an HBM contain a set of 64 16-bit two's complement values for a specific 8x8 matrix-vector multiplier (MV-mult). The scheduler has to drive all inputs of the MV-mult in one clock cycle during its scheduled time slot. The MV-mult inputs include 8 16-bit two's

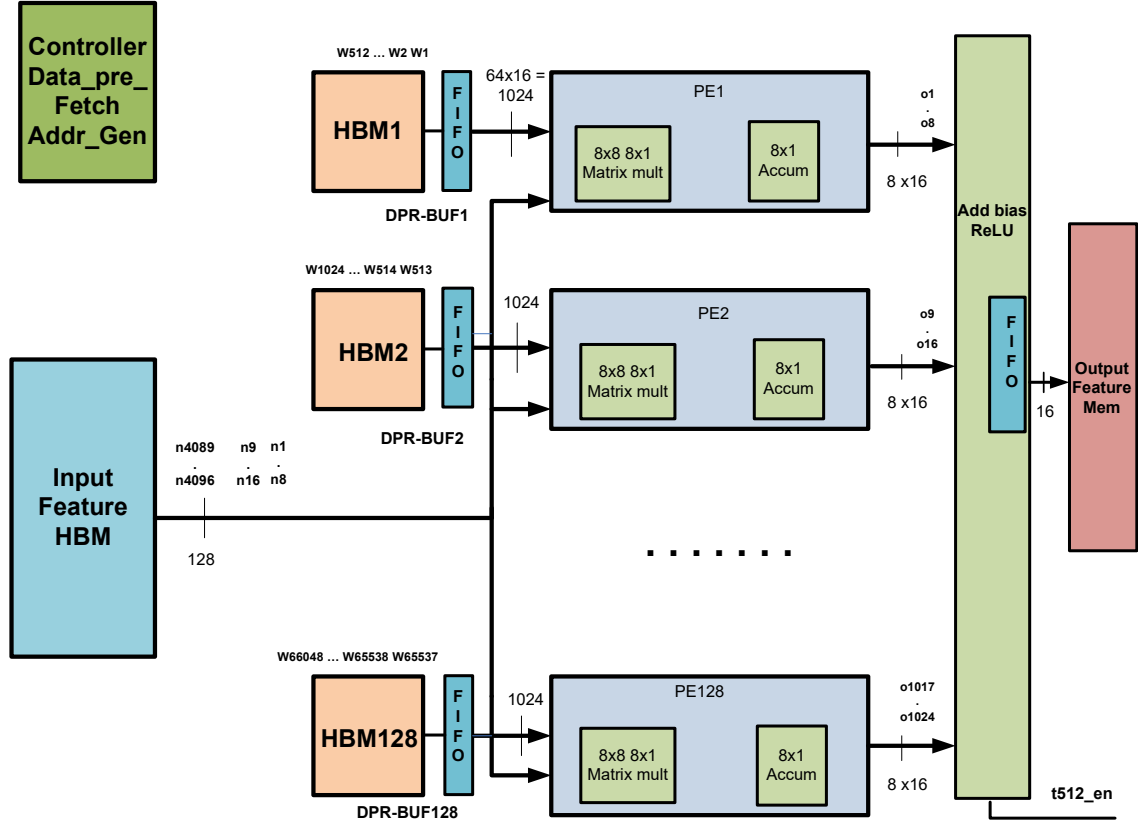


Figure 22: High level architecture block diagram. Each HBM has a dedicated data-pre-fetch and on-chip buffer unit, DPR-BUF, with 8 rate matching FIFOs. An HBM's DPR-BUF ensures that 1024 bits of weights, stored in the on-chip buffer, are aligned for a single cycle read by the PE. Input and output memories have dedicated address generators. One top-level controller schedules the data flow in all 128 PE channels.

complement values of input features from HBM\_IN (1 cycle 128 bits read out from HBM\_IN) as well as 64 16-bit weight values, which form a 1024 bit parallel bus of weights to the MV-mult. The DPR-BUF ensures that this 1024 bit bus is driven by 8 128-bit bus outputs of each HBM

as shown by 4 Da and 4 Db transactions in Fig.23. Note that an HBM's 8 DRAMs make up a stack and each DQ[127:0] output of a DRAM contributes to a portion of the DPR-BUF's 1024-bit on-chip buffer after being rate-matched by its FIFO. Two clock domains, a 500MHz wr\_clk (write into FIFO), and a 662 MHz rd\_clk (read from FIFO), are used in the DPR-BUF. This matches the HBM's 500MHz DQ[127:0] bus to the 662 MHz clock domain used in the pipeliend PEs and up to the ReLU's output FIFO write port.

Fig. 23 is from the JESD235C HBM2 standard and shows how 1024 bits can be read out with two read requests, using burst length of 4, BL4, with R=6 to two column addresses in the same bank. The two read requests generate 8 128-bit transactions on the DQ[127:0] bus which is sampled by the DPR-BUF. Following the main controller's sequence, the DPR-BUF initiates two read accesses to all HBMs during cycles T0 to T9 overlapped with a read access to the Input features memory HBM\_IN for the next input value in order for them to align at the MV-mult interface. This is shown in the following Fig. 24 .

The 8 128-bit read out cycles, in the 500 MHz clock domain, from an weights HBM (in BL4 mode) fill up its DPR-BUF's 1024-bit buffer. In the 662 MHz clock domain, the 1024 bit buffer is read in 1 cycle, Rd, overlapped with read out of the input from HBM-IN. The following 3 662 MHz cycles are processing cycles P1, P2, P3. If not empty, the FIFO is then read in the next Rd cycle and so on. In the 500 Mhz domain, the HBM is read in cycles m1 to m8. After each mx cycle, the FIFO is written in its corresponding wrx cycles. This is shown in Fig. 25. Note that cycle m8 is followed by cycle sw, to allow for HBM bank switching if the two read commands map to different banks. The main control sequence can allow for more sw cycles if needed.



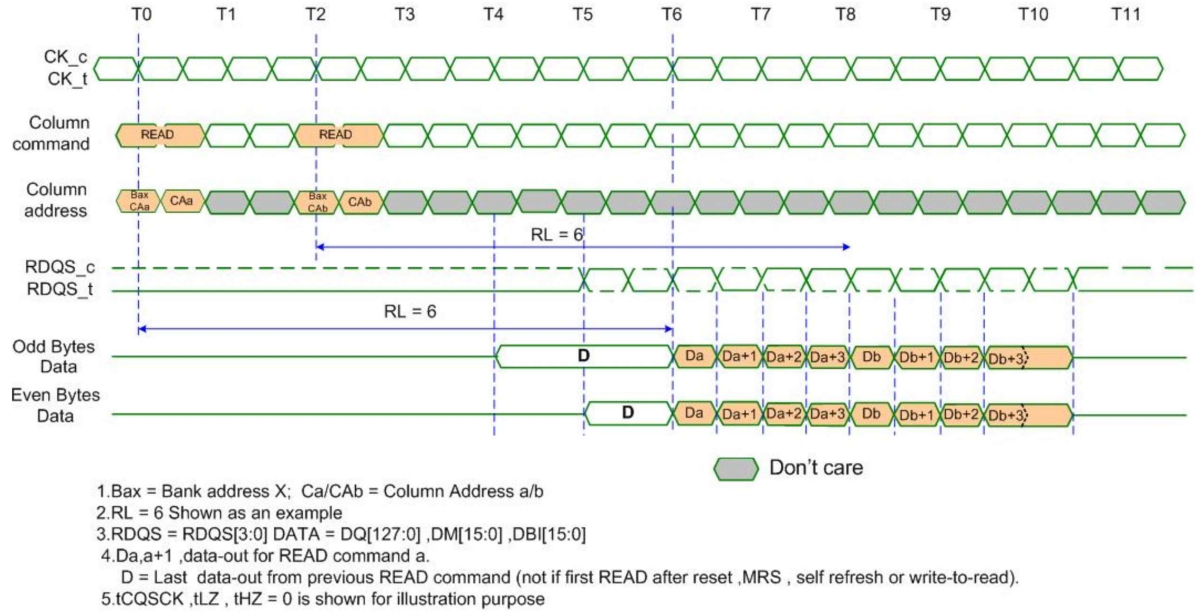


Figure 23: Read access timing from JESD235C. The access starts with the first read request to column address Ca. After R T-cycles (T0 to T6 for R=6 example) a burst of 4 128-bit words, Da to Da+3, is available on DQ[127:0]. Similarly, the second read request to column address Cb generates a burst of 4 128-bit words, Db to Db+3. The DPR-BUF combines the 8 128-bit words and writes them into 8 corresponding FIFOs. The 8 FIFOs are then read into the 1024 bit on-chip buffer.

## 12.2 Matrix-Vector Multiplier Unit

Each PE contains a dedicated 8x8 MV-mult for fixed-point data in the Q(17,10) format. The choice of an 8x8 tile in the weights matrix in Fig. 19 determines the size of the MV-mult as well as the number of HBMs and PEs in the system. We use 8x8 tiles of weights as an example implementation and other sizes are possible in the proposed architecture as well. MV-mult contains

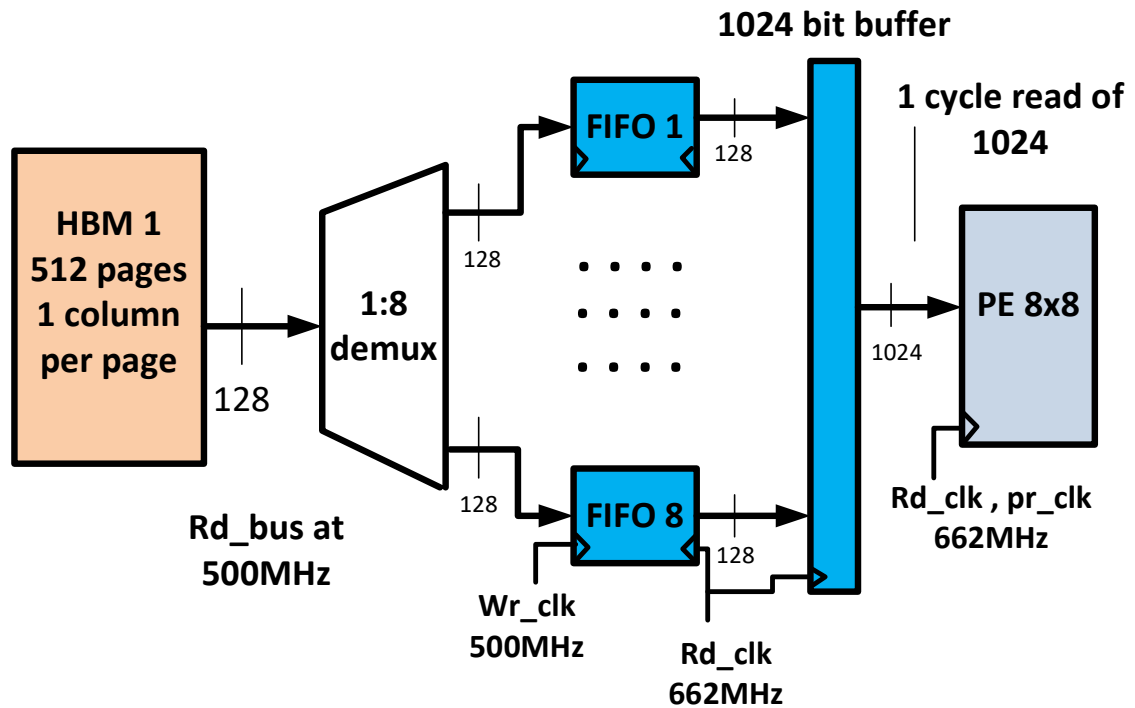


Figure 24: Data prefetcher and on-chip buffer for HBM read accesses. One HBM is shown driving weights to its PE. The main controller issues two read requests to column addresses Ca and Cb. Each request generates a burst of 4 128 bit transactions on DQ. The prefetcher writes 8 128 bit DQ values into 8 corresponding FIFOs. A read request is then issued to all FIFOs, and their output is stored in a single 1024 bit register. This aligns the weights read-out cycle with the HBM-IN read out of the next 8 16-bit input feature values.

an array of 64 scalar multipliers where both operands have the same bit width in the Q(17,10) format. Each product is also truncated and rounded to fit into Q(17,10). The selection of 17 bits from the total of 34 bits (before truncation) is configurable and can be decided by the dynamic

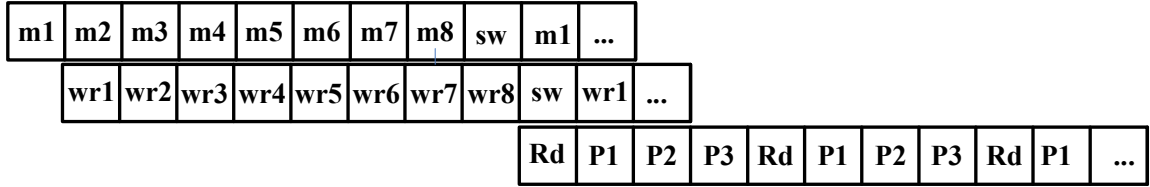


Figure 25: DPR-BUF HBM memory read out cycles m1 to m8 in the 500MHz domain and FIFO write cycles wr1 to wr8. FIFO read cycle Rd and PE processing cycles P1 to P3 are in the 662 MHz domain. The FIFO is read every 4th cycle.

range of the FC layer from offline calibration. A two-stage pipeline is implemented by a dedicated register at the output of each scalar multiplier. Note that a different 7 stage pipeline is used to break up the adder tree, as described below in a section on a pipelined PE design. An adder tree of seven Q(17,10) adders sums all partial products for each of the 8 rows. A zero-detector is used for each operand to gate off switching within the module when one or both operands are zero. The output 8x1 vector of products is available in 1 100 MHz clock cycle in an ASIC PDK 45 nm implementation. Fig. 26 shows the details of MV-mult. Note that for the pipelined PE described later, the critical path in the seven adder tree is reduced to 1.51 nsec using a seven stage pipeline. This allowed us to run the pipelined design of a PE at 662 MHz and increase the max throughput of the accelerator considerably.

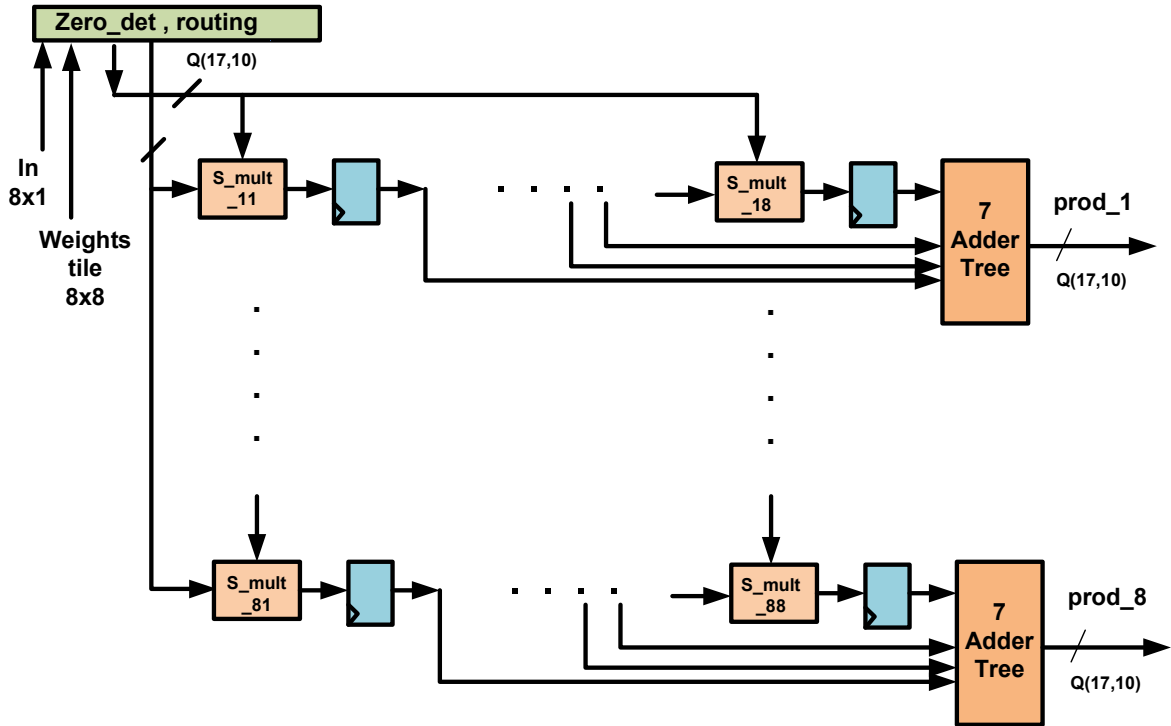


Figure 26: MV-mult micro-architecture for 8x8 tile of weights.

### 12.3 Vector Accumulator Unit

Each PE in Fig. 22 has an 8x1 vector accumulation unit (V-Accum) for adding up the partial products generated during each of the 512 time-slots. A V-Accum maps to each 8x1 row of the weights matrix in Fig. 19 ; for example V-Accum-1 to o1-o8, V-Accum-2 to o9-o16 and so on. Each V-Accum receives the prod-1 to prod-8 outputs from its upstream MV-mult. A new partial product is accumulated in 1 clock cycle. Fig. 27 shows the details of V-Accum.

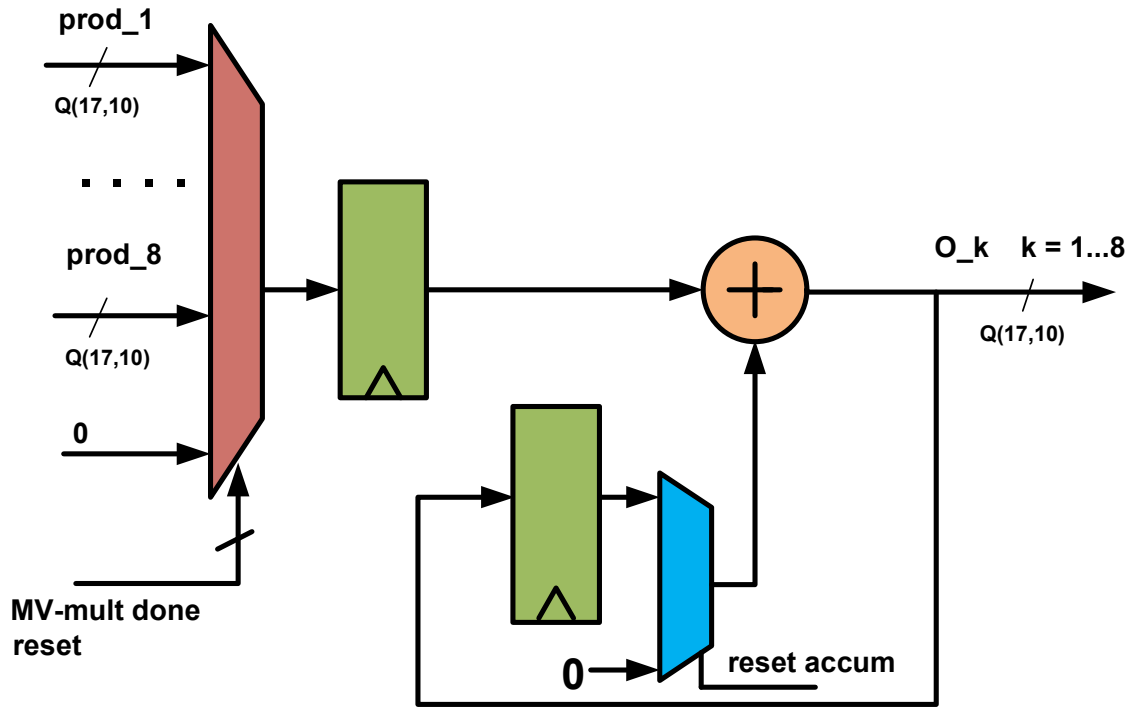


Figure 27: Vector accumulator V-Accum datapath.

#### 12.4 ReLU and Bias Addition Unit

The activation function we use is a rectified linear unit (ReLU) which introduces the  $\max()$  non-linearity as  $\text{out} = \max(\text{in}, 0)$ . A set of bias vectors can be added to each PE output as shown in Fig. 22. Each bias vector,  $\text{biasN} \dots \text{biasN}+7$ , has 8 Q(17,10) elements which are added with 8 adders to the corresponding PE output vector elements  $\text{oN} \dots \text{oN}+7$ . The outputs of each adder are then compared with 0. The combined addition and comparison are done in one clock cycle. Note that this is done only after the 512th (last) time-slot as indicated by the  $t512\_en$  signal in Fig. 22. Each

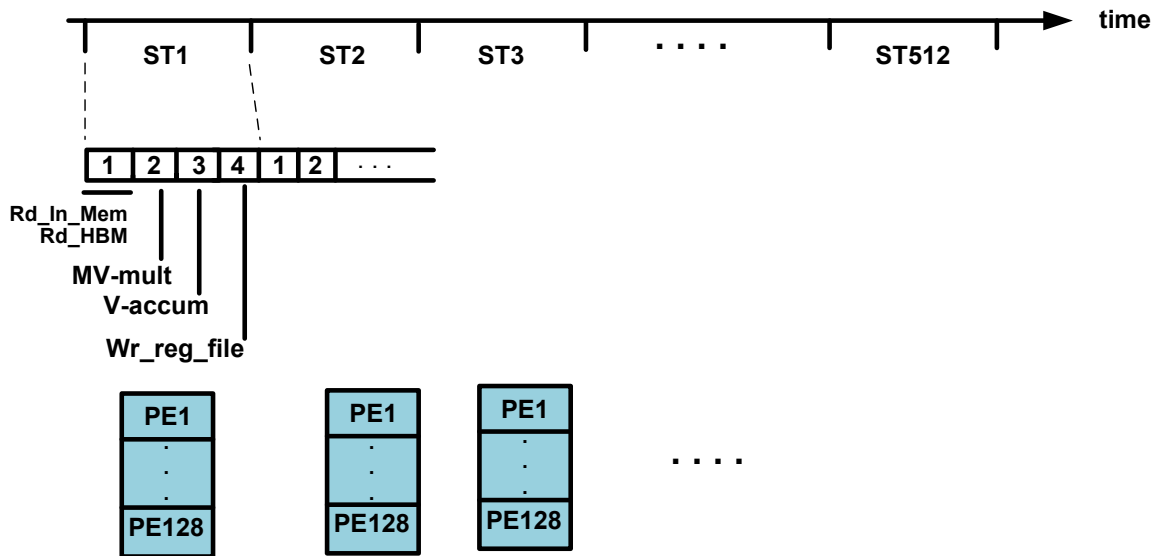


Figure 28: Main controller sequence. All 128 PEs are processing a new 8x1 feature vector from In-MEM in each state ST1 ... ST512. In each state an 8x8 tile of weights is read from the HBM corresponding to each PE.

element is then written into an output FIFO for streaming to the Output Feature Memory. A 2kB 16-bit wide SRAM memory is used for storing the output features. The output FIFO's write clock is 100 MHz (non-pipelined PE) and 662 MHz (pipelined PE), the output FIFO's read clock is 150 MHz, and it implements rate matching to the Output Feature Memory. A FIFO depth of 774 16 bit words, or approximately 2 kB is required for this choice of write/read clock frequencies.

### 12.5 Main Processing Sequence

The main controller shown in Fig.22 implements the processing sequence shown in Fig.28.

The sequence is for a 4096-1000 layer such as Alex-8 (FC8) in AlexNet or VGG-16 (FC8) in VGG16, as in Table III in (Han and et al, 2016). Using 8x8 tiles for the weights matrix in Fig 19, the equivalent matrix of tiles is 128x512. The main control sequence therefore has 512 states, ST1 to ST512 as shown. All 128 PEs are processing an input 8x1 feature vector with their corresponding tile of weights in each state. Note that in ST1 reads to input features memory, HBM\_IN, and reads from the weights FIFO in the PE's DPR\_BUF are overlapped. Each column in the weights matrix is processed in sequence, and all rows in a column are processed in parallel with each row's dedicated PE. We call this a column-row-column schedule. This schedule ensures that the computing load is equally shared among all PEs. It also achieves almost optimal load balancing among all PEs since all are utilized in each control time slot. Using this schedule we read the entire input features memory only once, where each read transaction returns a 8x1 (for 8x8 PE) or 16x1 (for 16x16 PE) vector of inputs. Similarly each row's HBM weights memory is also read only once. This minimal access pattern to the memories contributes to low processing latency and minimizes power consumption due to memory accesses. The same control sequence, ST1 to ST512, can be applied to 4096-4096 layers such as FC7 Alex-7 and FC7 VGG-7 in (Han and et al, 2016). To maximize throughput, 512 8x8 PEs in one pass, can be used for processing in each state. Alternatively, 128 16x16 PEs can be used in two passes to cover all 256 rows, for details see section "Up-Scaling to Larger FC Layers" below. For the larger layers, eg. FC6 Alex-6, 9216-4096, the control sequence has ST1 to ST1152 or 1152 states. The number of 8x8 PEs remains at 512 for one pass or 128 16x16 PEs with two passes. For FC6 VGG-6, 25088-4096,

the control sequence has ST1 to ST3136 or 3136 states. The number of 8x8 PEs remains at 512 for one pass or 128 16x16 PEs with two passes.

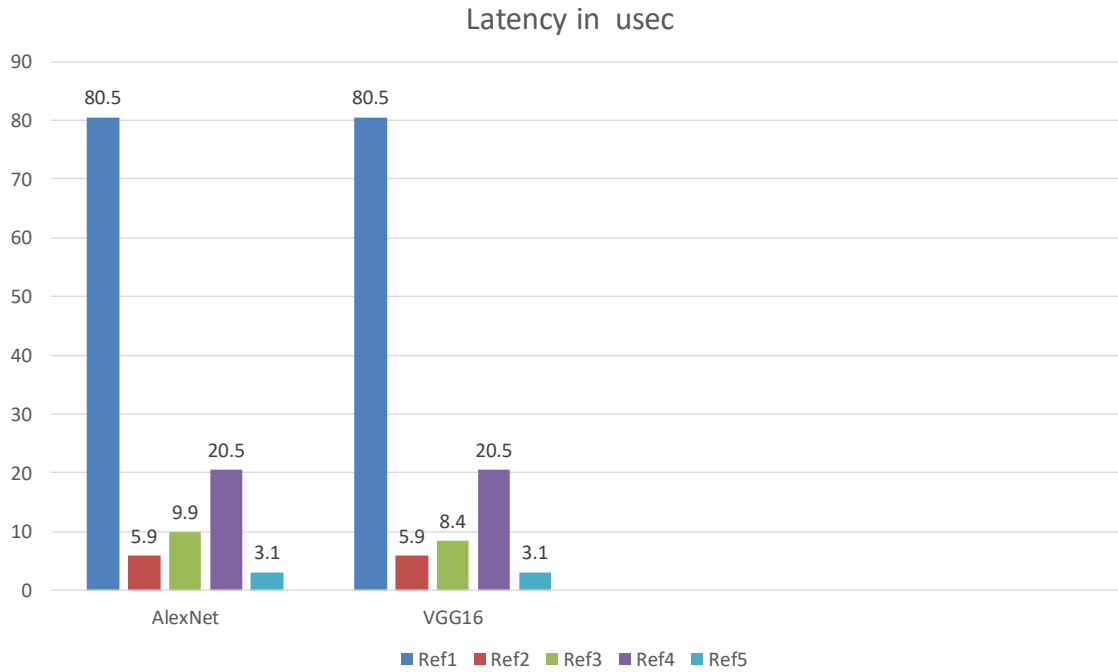


## CHAPTER 13

### SIMULATION AND ASIC IMPLEMENTATION RESULTS OF FC-ACCEL FOR FC LAYERS IN ALEXNET AND VGG16

The FC-Accel microarchitecture for the Alex-8/VGG-8 4096-1000 FC8 layer was implemented in fixed-point Q(17,10) (data and weights) Verilog and simulated in ModelSim SE. A Python floating-point implementation of the same layer was used as reference. Pipelined and non-pipelined PEs were implemented with 662 MHz and 100 MHz clocking respectively. The seven adders tree in the original PE was pipelined to reduce its critical path delay to 1.51 nsec with a 662 MHz clock. Non-zero values were used for all data features and for all weights. The following Fig. 29 summarizes the achieved processing latency for the specified design parameters and compares with recent comparable benchmarks. For FC8, FC-Accel achieves a 63% reduction in processing latency (VGG-16) when compared to EIE from (Han and et al, 2016) which is also an ASIC implementation in a TSMC 45 nm process.

The fully connected layer FC8 in both AlexNet and in VGG16 has the same 4096-1000 dimensions. Our FC-Accel latency is based on non-zero values for all input features and all weights. The data for GPU Titan X and EIE is from (Han and et al, 2016). We summarize a pipelined 8x8 PE FC-Accel implementation using an 662 MHz clock and 7 pipeline stages for the 7 adder tree in Fig. 26. The non-pipelined version uses 100 MHz clocking. Using pipelining brings the worst



Ref1 - (Han and et al, 2016) GPU (Titan X) Batch size 1, dense  
 Ref2 - (Han and et al, 2016) GPU (Titan X) Batch size 64, dense  
 Ref3 - (Han and et al, 2016) EIE with compression, pipelined PE, 800MHz  
 Ref4 - **this work** (128 non-pipelined 8x8 PEs, 100MHz)  
 Ref5 - **this work** (128 pipelined 8x8 PEs, 662MHz)

Figure 29: Processing latency for FC8 in usec for several benchmarks.

case critical path delay to 1.51 nsec and considerably improves latency. However it increases power dissipation as shown below.

Fig. 30 reports the operations/sec for each major processing block in FC-Accel.

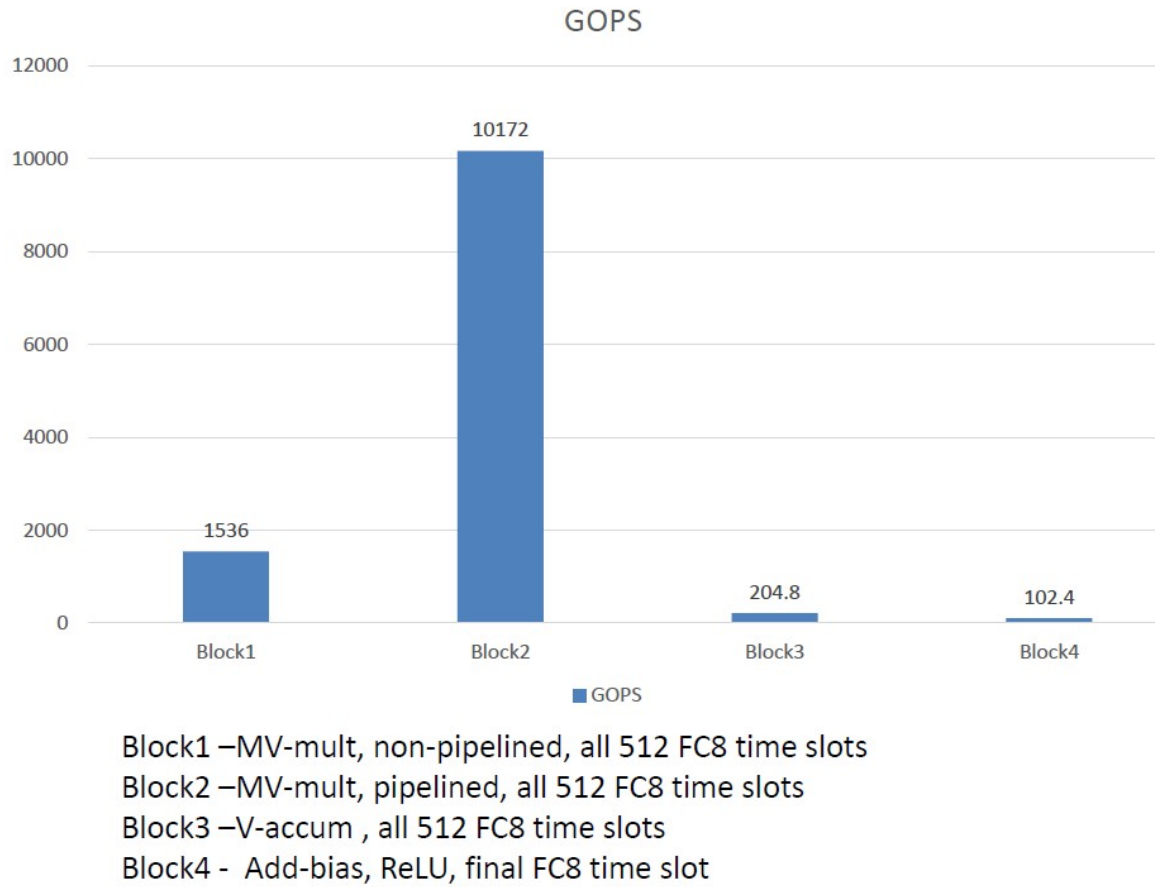


Figure 30: Processing Blocks performance.

The total (dynamic and leakage) power consumption in the pipelined 8x8 PE is shown in Fig. 31 for each processing block along with the cell counts.

Fig. 32 compares the achieved Giga operations per second, GOPS, for the 4096-1000 FC8 layer with other comparable benchmarks (all units are in GOPS) and shows the speedups achieved by

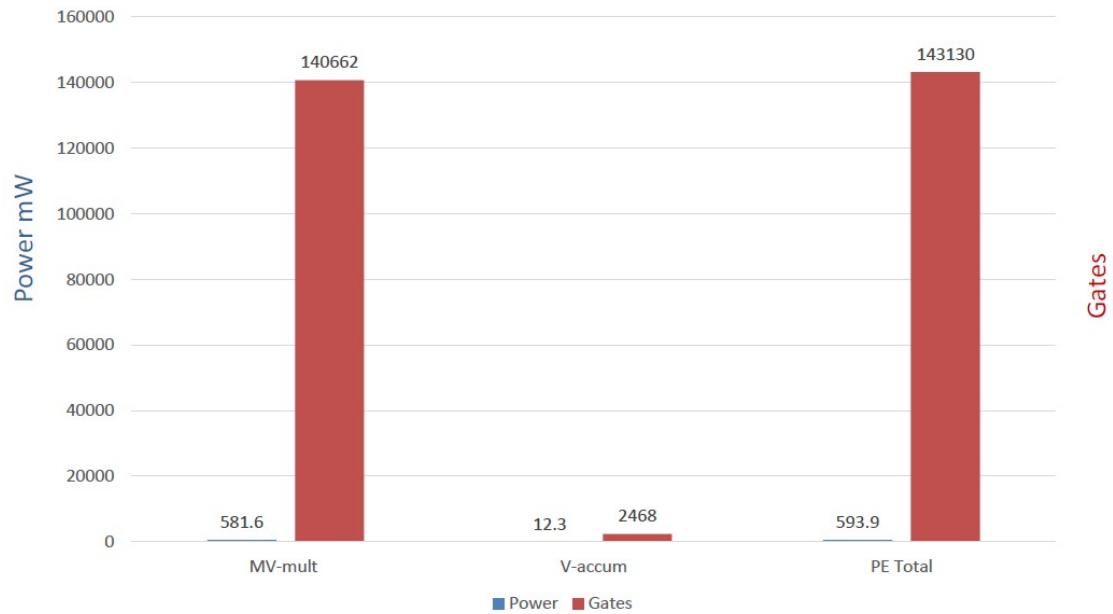


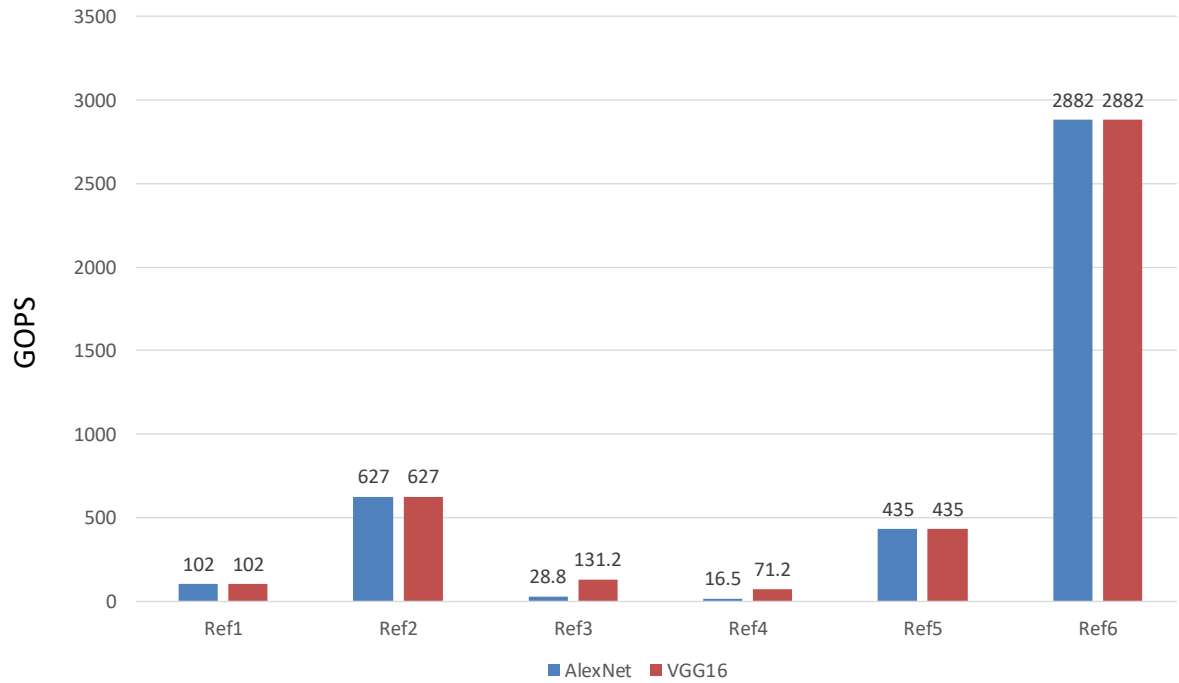
Figure 31: Power per PE processing block.

FC-Accel in ASIC PDK 45 nm technology. The FPGA implementations are from Table 17 in (Li, 2018). The TETRIS implementation is from (Gao and et al, 2017).

We note that the TETRIS accelerator uses 16 Hybrid Memory Cubes, 16 HMCs, a type of 3D DRAM memory different from HBM.

### 13.1 CMOS ASIC Implementation

We have implemented the Alex-8/VGG-8 FC8 layer using the CMOS ASIC PDK 45 nm standard cell library for synthesis. The Cadence RTL Compiler (RC) tool was used and the design achieved



Ref1-EIE ASIC 45nm with compression, pipelined PE, 800MHz

Ref2-TETRIS ASIC 45nm, 500MHz, 16 3D engines, 16 HMCs

Ref3-VC707 FPGA 28nm, 150MHz, 13.5 W

Ref4-ZC706 FPGA 28 nm, 150MHz, 8.9 W

**Ref5-this work**, 128 non-pipelined PEs, 100MHz, 17 W

**Ref6-this work**, 128 pipelined PEs, 662MHz, 90.1 W

Figure 32: GOPS comparison with ASIC and FPGA platforms for FC8 acceleration.

timing closure with a 100 MHz clock for the non-pipelined PE and 662 MHz for the pipelined PE.

Fig. 33 summarizes timing, area, and power for the non-pipelined and pipelined FC-Accel design, with 128 8x8 pipelined or non-pipelined PEs. Note that power consumption due to HBM,

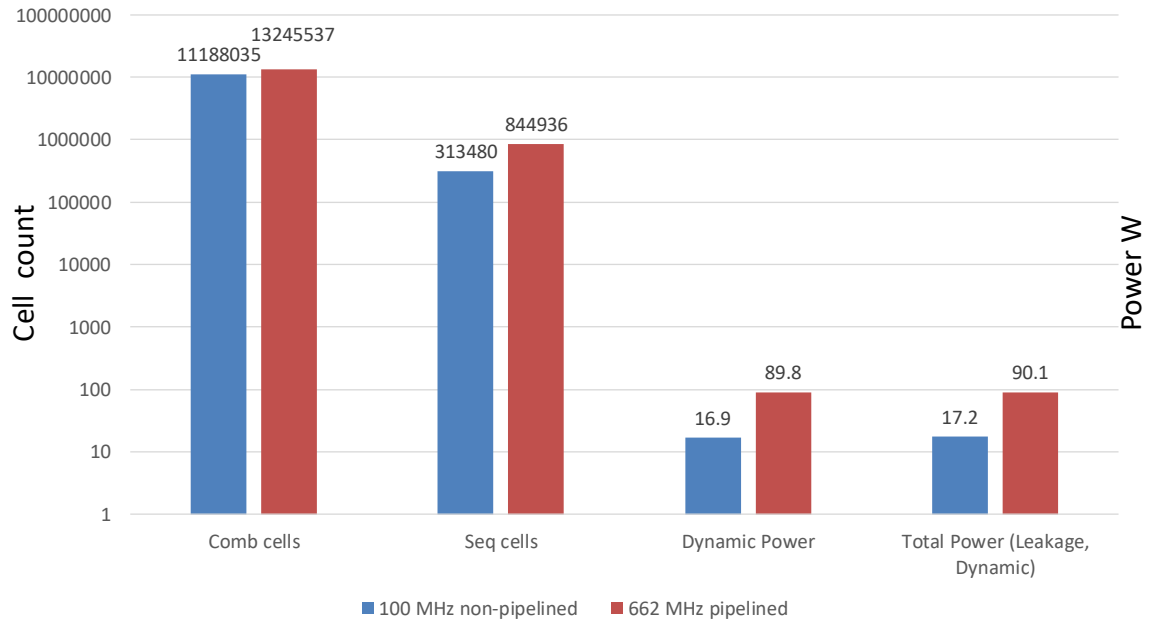


Figure 33: FC-Accel with 128 PEs. PDK 45nm standard cells implementation.

input, and output memories and their interfaces (input DPR\_BUF and output ReLU FIFO) is not included.

### 13.2 Characterization of a pipelined 16x16 PE

In this section we present ASIC PDK 45 nm implementation results for a pipelined 16x16 PE which is used in the up-scaling of the proposed architecture to the larger FC7 and FC6 layers. The 16x16 design is similar to the 8x8 PE design the main difference being MV-mult which is now a 16x16 matrix-vector multiplier. Fig. 26 now has 16 rows S\_mult\_11 to S\_mult\_16\_1 and 16

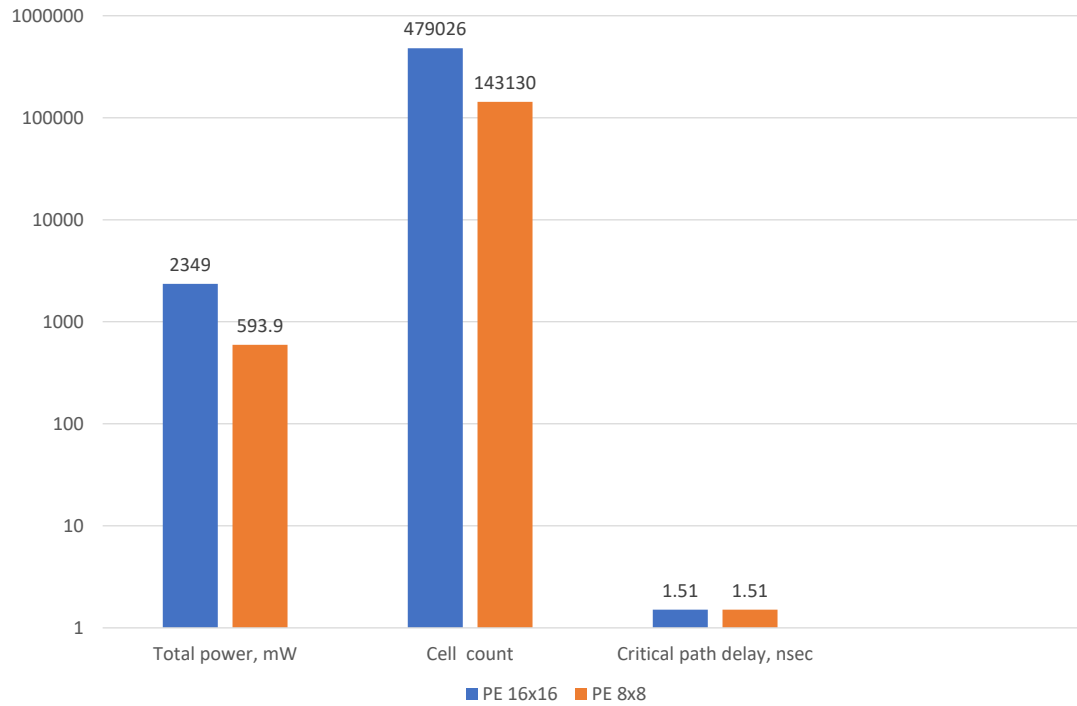


Figure 34: Implementation data for pipelined 16x16 PE at 662 MHz. The y axis is log( ) base 10.

columns S\_mult\_11 to S\_mult\_1\_16. A 15 adder tree sums all partial products for each row. The adder tree is pipelined with a 15 stage pipeline which reduces the critical path delay to 1.51 nsec or 662 MHz clocking.

The accumulator data path is also derived from Fig. 27 and has 16 product inputs, prod\_1 to prod\_16 and 16 accumulated outputs, O\_1 to O\_16.

Fig.34 summarizes cell count and worst-case statistical power for the 16x16 PE with a 662 MHz clock. The pipelined 8x8 PE implementation is also listed for comparison.

	AlexNet		VGG-16	
	8x8 PE	16x16 PE	8x8 PE	16x16 PE
FC8 latency in usec	396	173	396	173
Number of read accesses to a single HBM	128	256	128	256

TABLE I: FC8 LATENCY FOR 8X8 AND 16X16 PE WITH 1 HBM AND 128 PE MODULES. THE NUMBER OF READ ACCESSSES TO THE HBM ARE LISTED FOR EACH PE DIMENSION.

### 13.3 FC8 Latency for 1 HBM Shared by 128 PEs

We have also analysed FC8 latency performance for a reduced architecture with 1 HBM shared by all 128 PEs, for both 8x8 and 16x16 dimensions. The HBM has to be read for a tile of weights for each PE in a row; the result of the matrix-vector multiplication is then used to update the PE's accumulator in the row. The number of read accesses depends on the size of the tile; for 8x8 tiles 128 accesses are required, for 16x16 tiles 256 accesses are required. For the 16x16 case, there's 64 rows, however the HBM has to be read 4 times for each row in order to deliver 4096 bits to the row's PE. As expected the latency for FC8 processing increases significantly due to a single HBM shared among all PEs. By contrast, if an HBM is dedicated to each row's PE, a single read access is needed (for 8x8 PE), or 4 read accesses (for 16x16 PE) resulting in a minimal latency. Table I summarizes the increase in latency for each tile size. As can be seen from the data, the larger 16x16 PE does reduce latency as expected (compared to the 8x8 PE), even though twice as many read accesses have to be done to HBM for its weights. The advantage of a single shared HBM is reduction in the total power dissipation.



A similar analysis of the reduced architecture with a single shared HBM is shown for the larger FC7 and FC6 and the resulting significant latency increases are shown in the following section on "Up-Scaling to Larger FC Layers".

### 13.4 FC-Accel Complexity Analysis

The time complexity for multiplication of  $n \times m$  matrix and  $m \times 1$  vector is  $O(n * m)$ . This is the number of scalar fixed-point multiplications. Using 1 clock per scalar multiplication, that is  $O(n * m)$  clock cycles. Here a scalar is a 16-bit fixed point value. Using square tile of size  $t$  (1 tile per PE) we have  $O(m/t)$  iterations using  $n/t$  parallel PEs. Each PE does  $t * t$  scalar multiplications and  $t$  scalar additions in 1 clock cycle. The  $O(m/t)$  iterations, to process all columns, take  $O(m/t)$  clock cycles, with all  $n/t$  PEs computing in parallel.

Total time complexity is

$$O(n * m) \quad (13.1)$$

clock cycles, with one scalar multiplication and addition per cycle. The total number of operations (multiplications and additions), for all  $m/t$  iterations, is

$$O(n * m + \frac{n * m}{t}) \quad (13.2)$$

These are the operations running in parallel in each PE, in a 1D array of  $n/t$  PEs.

### 13.5 Up-Scaling to Larger FC Layers

In this section we present estimated performance with an up-scaling of the proposed micro-architecture for the larger FC6 and FC7 layers in AlexNet and VGG16. We use the 16x16 PE described above for these layers in order to efficiently process the larger sizes of the weight matrices. The 16x16 tile of weights reduces the number of rows and columns of matrix in Fig. 19 and simplifies the processing schedule as well. Both layers have 256 matrix rows with 16x16 PEs. Since a 16x16 PE has 256 weights for its matrix-vector multiplier, this is  $256 \times 16 = 4096$  bits of weights for each PE in each row. The HBM's DPR\_BUF 8 FIFOs for that row can be read in 4 cycles (1024 bits per cycle, from 4 consecutive locations in each FIFO) to deliver these bits to the row's PE. The up-scaled micro-architecture has 256 16x16 PEs and 256 HBMs to supply the weights and process the inputs in a single pass. To save resources, we propose 128 16x16 PEs and 128 HBMs and two passes to process all the inputs.

We also use an HBM for the input memory so that 16 16 bit words can be read out in 1 cycle to provide the 16x1 input vector to each of the 16x16 PEs. The main control sequence in Fig. 28 therefore increases by 3 cycles to 7 cycles: 4 for reading the HBM's DPR\_BUF 8 FIFOs with weights (overlapped with 1 cycle for reading the HBM with inputs), and 3 for matrix-vector multiplication, accumulation, and write back.

Fig. 35 shows the scheduling with the up-scaled micro-architecture for FC6 and FC7 when 128 16x16 PEs and 128 HBMs are used with two passes. In the horizontal (time) direction, AlexNet FC6 requires 576 (9216/16) time slots per pass, VGG16 FC6 requires 1568 (25088/16) time slots per pass, and FC7 requires 256 (4096/16) time slots for either network.

Two pages in each HBM are required to store all the weights for the HBM's row (PE). The first page is used in pass 1 and the second page in pass 2. The maximum number of weights for the FC6 layer in VGG16 is  $25088 \times 4096$  or 102,760,448 weights. Using 2 bytes per weight, this requires 268 MB of storage which is easily stored in the new HBM2 16 GB part (Flashbolt) from Samsung. Each page stores 134 MB of weights. The FC6 layer in AlexNet is  $9216 \times 4096$  so it has less weights and can also fit in the 16 GB HBM2.

The switching between pages in an HBM is assumed to be negligible as well as the saving of the 128 PE's outputs after each pass to output memory. Using the largest layer, FC6 VGG16, each pass will require 1568 time slots; using a pipelined  $16 \times 16$  PE at 662 MHz and 7 cycles per time slot will therefore require 16.6 usec for a pass. Both passes will take 33.2 usec for processing the entire layer. This is compared to 34.4 usec as reported in Table IV in (Han and et al, 2016) which uses compression and is a considerable improvement since the saving is accumulated over each forward inference.

Fig. 36 below summarizes the estimated total latency for FC6 and FC7 processing in both networks with the proposed up-scaled micro-architecture with 128 pipelined  $16 \times 16$  PEs. For comparison, latency from Table IV in (Han and et al, 2016), which uses compression in each layer, is also included.

A reduced architecture with a single shared HBM for all PEs will severely increase processing latency as shown in Table II. As can be seen from the data, the larger  $16 \times 16$  PE does reduce latency as expected (compared to the  $8 \times 8$  PE), for both FC6 and FC7, even though twice as many

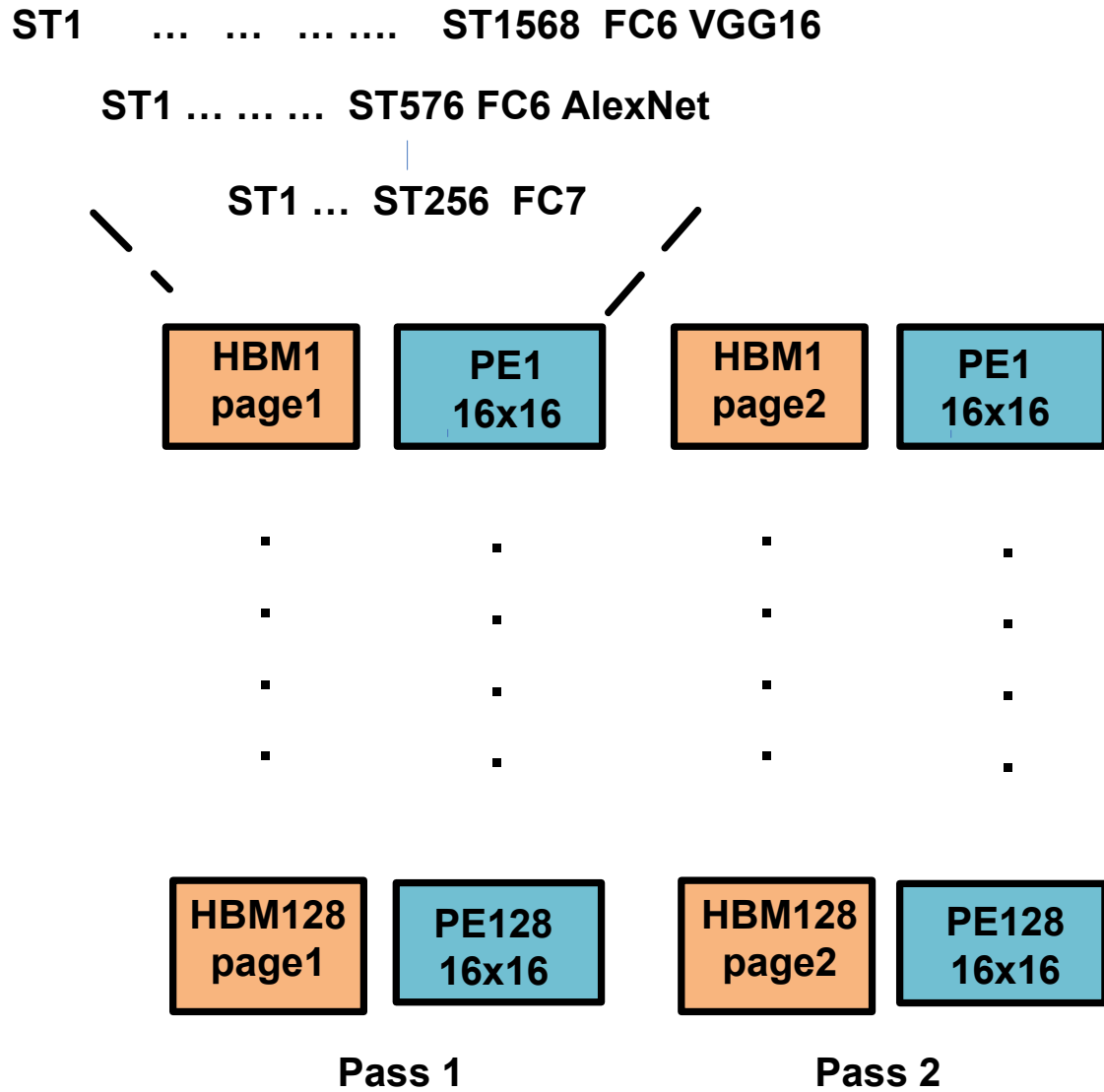


Figure 35: Up-scaling the proposed micro-architecture for handling FC6 and FC7 layers. In each pass 128 HBMs and 128 16x16 PEs are reused. The input and output memories are connected as in Fig.3

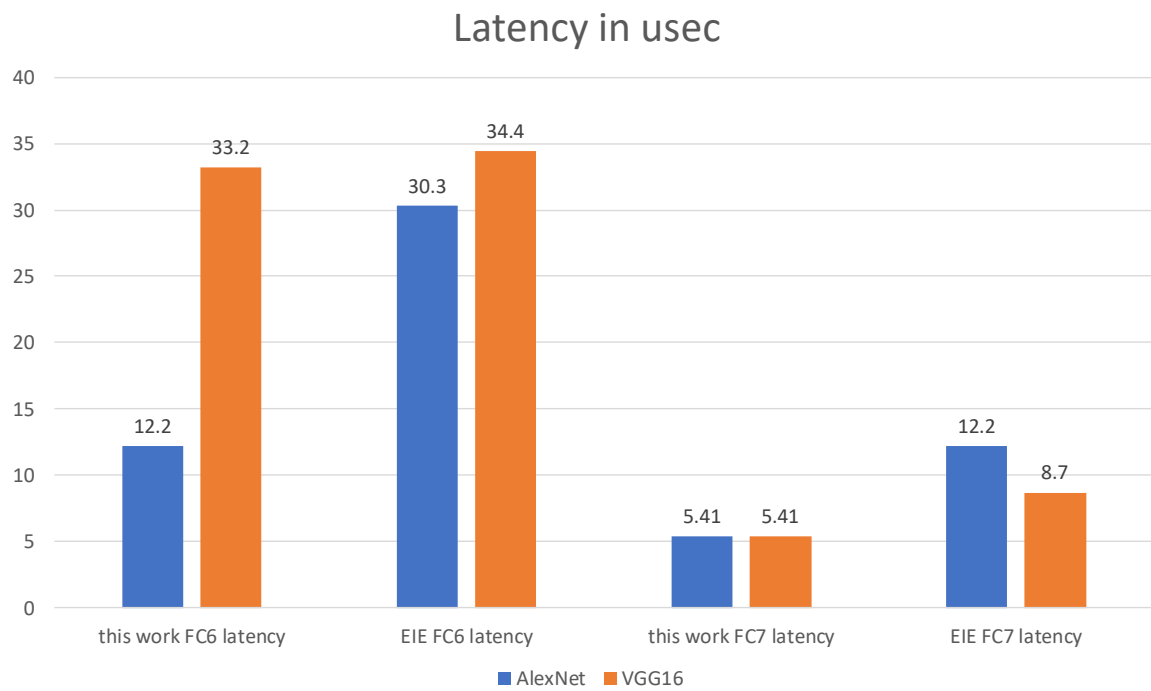


Figure 36: Estimated performance of FC6 and FC7 layer acceleration with FC-Accel 128x1 array of pipelined 16x16 PEs and two passes over the array.

read accesses have to be done to HBM for its weights. The advantage of a single shared HBM is reduction in the total power dissipation.

	AlexNet		VGG-16	
	8x8 PE	16x16 PE	8x8 PE	16x16 PE
FC7 latency in msec	1.58	0.693	1.58	0.693
FC6 latency in msec	3.56	1.56	9.7	4.2
Number of read accesses to a single HBM	512	1024	512	1024

TABLE II: FC8 LATENCY FOR 8X8 AND 16X16 PE WITH 1 HBM AND 128 PE MODULES. THE NUMBER OF READ ACCESSSES TO THE HBM ARE LISTED FOR EACH PE DIMENSION.

# Part IV

# Conclusion

## CHAPTER 14

### OPEN PROBLEMS AND ACCELERATOR EXTENSIONS FOR THEM

*Parts of this chapter have been presented in (Iliev and Paprotny, 2015), (Salarian et al., 2016) and (Salarian et al., 2018). Copyright © 2015-2018, IEEE.*

This chapter discusses a few possible extensions of our work and also reviews related open problems.

#### 14.1 Spatial Localization - SSL Extensions

##### 14.1.1 Non-Linear Cost Function for AOA based Estimation

An open problem in non-linear optimization theory is how to efficiently minimize a non-linear, non-convex, cost function such as the one in Chapter 3

$$\theta_{ML} = \arg \min_{\alpha} (\tilde{\alpha} - \alpha)^T \times C \times (\tilde{\alpha} - \alpha). \quad (14.1)$$

Here, the spatial coordinates of  $U$  are estimated by  $\theta_{ML}$  and  $\tilde{\alpha}$  is the AOA measurements vector from  $M$  anchors given as

$$\tilde{\alpha} = [\tilde{\alpha}_1 \tilde{\alpha}_2 \dots \tilde{\alpha}_M]^T = \tilde{\alpha}_0 + n, \quad (14.2)$$

with  $\tilde{\alpha}_0$  as the true angle (in radians) and  $n$  is the measurement noise.  $C$  in Eq. 3.1 is the inverse covariance matrix of  $n$ .

Direct minimization of this cost function gives the maximum-likelihood solution  $\theta_{ML}$  and algorithms such as Non-linear Least squares Levenberg–Marquardt, NLS-LM, have been applied with some success. However NLS-LM is very expensive computationally since it uses matrix inversion and Jacobian and Hessian matrix computations. To illustrate this point, we briefly present a MATLAB floating point model of NLS-LM processing and ensuing hardware architecture that we have considered before selecting our RNN dual-LP approach in Chapter 4.

The processing pipeline for NLS-LM with 3D AOA measurements is presented in Fig. 37 below.

We have developed a floating point MATLAB model of the NLS-LM processing pipeline and used it to simulate 3D AOA localization testcases. Fig. 38 below shows a 3D MATLAB plot of estimated 3D coordinates vs ground truth. A possible systolic array realization is then described in the block diagram.

The very large amount of arithmetic processing in the NLS-LM approach is now clear.

We now present an extension of our RNN approach for minimizing a quadratic approximation of the cost function. One possible extension of the SSL accelerator is to apply the following approximation of the arctan function

$$\beta_i = \tan^{-1} \left( \frac{y_T - y_i}{x_T - x_i} \right) \cong (\pi/4) * \left( \frac{y_T - y_i}{x_T - x_i} \right) \text{ for } -1 \leq \left( \frac{y_T - y_i}{x_T - x_i} \right) \leq 1 \quad (14.3)$$

This will generate quadratic terms as well as linear terms for the cost function above. We then have a constrained quadratic program and the RNN dual-primal Quadratic architecture can be used to solve it. Figure 39 shows the datapath extentions required for solving a quadratic program



### 3D AOA Coordinate equations

- The 3D localization problem, with 2 anchors and 1 unknown node measuring AOA to both anchors can be mathematically modelled as a highly non-linear cost function of a parameter vector  $p$  :
- 
- $J(p) = \sum_{i=1..m} (y(t_i) - y_{est}(t_i; p))^2$
- $i = 1..m$
- Where  $y(t_i)$  is a set of 3 AOA measurements at time  $t_i$ ;
- $y_{est}(t_i; p)$  is the best current estimate for  $y(t_i)$
- Vector (3x1)  $p$  is the best current estimate of coordinates  $x_{est\_i}, y_{est\_i}, z_{est\_i}$  of the unknown node
- There's  $m$  measurements,  $i = 1..m$ , each taken at time  $t_i$

### 3D AOA processing pipeline

- The above non-linear cost function  $J(p)$  of squared error between measured AOA's and estimated AOA's can be efficiently minimized by the state-of-the-art Levenberg-Marquardt algorithm. Computations for each iteration include Jacobian matrix, Hessian matrix, matrix inverse, and dot product of vectors, and scalar / vector division and scaling.

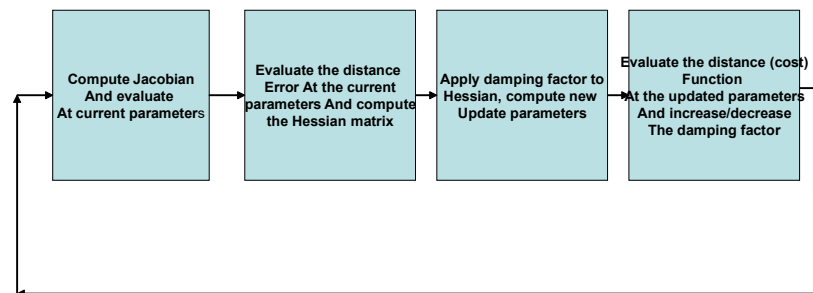


Figure 37: NLS-LM processing pipeline for 3D AOA measurements. The processing iterations continue until parameters updates (3D coordinate updates) become smaller than a threshold value.

: a QP block is muxed in using the "select QP or LP" signal. The QP block is the addition of the identity matrix  $I$  and the AOA matrix  $A$ .

Note that the  $\pi/4 * x$  approximation is different than the small AOA measurement error approximation we used in Chapter 3. That approximation transformed the non-linear cost function to an overdetermined Least squares problem. The RNN dual-primal linear program solver architecture in Chapter 4 was then applied to it.

Since the  $\pi/4 * x$  approximation transforms the original cost function to a quadratic program, we expect better estimation accuracy and more robust behaviour with stronger measurement noise.

#### **14.1.2 Multi-Modal methods**

Another open problem in AOA based localization is the growth of uncertainty in the estimated coordinates as the noise in the angle measurements becomes too strong or when the positions (configurations) of the anchors results in reduced accuracy, as shown in Chapter 5 Fig. 7 (c) .

Multi-modal methods can be applied in this case: the AOA based localization system can be re-configured as an image-based coordinate estimator. An example of a multi-modal localizer is the work by (Georgiev and et al, 2004). Their mobile robotic platform uses two methods. The first method uses odometry, compass, GPS, and an Extended Kalman Filter (EKF). The second method uses camera pose estimation using an onboard camera. The second method is used when the uncertainty in the EKF generated estimates is too large. Their second method is also economical since it does not use LIDAR or optical-flow sensors. We choose to enhance the SSL AOA based localizer with a camera pose estimation system. This is a software-hardware co-design, with software implementing the image search (matching) and SFM portions described in the following. Camera pose estimation can be done using Horn's transform (coordinate registration) algorithm,

a Structure From Motion, SFM, algorithm, an image feature search (matching) algorithm, and a database (off-line generated) of images of anchors in the indoor environment of the mobile robot. We have implemented all of the above algorithms, as reported in (Salarian et al., 2018) and in (Salarian et al., 2016), in software. The Horn transform is one of most computationally expensive of all, so it is a good candidate for hardware acceleration. The key step is eigenvalue decomposition of a matrix, and we propose to extend the SSL accelerator with a neural network for Principle Component Analysis, PCA, which will also generate the eigenvalue decomposition. The figures below illustrate the basic Horn accelerator extension and its integration with SFM (SFM generates the  $P'1$  to  $P'5$  coordinates).

The GPS tags in the figures are replaced with indoor anchors known coordinates. The estimated Query GPS\_tag5 is the robot's camera center 3D coordinate in the indoor anchor's coordinate system, based on the 4 matching images. AOA measurements have not been used in this processing.

## **14.2 Speaker Recognition - SpkrRec Extensions**

### **14.2.1 Learning the Optimal Number of Clusters**

An open problem in unsupervised k-Means clustering is the a-priori selection of the number of clusters  $K$ . Our current SpkrRec accelerator is given the value of  $K$  before the start of all processing of the incoming MFCC feature frames. One possible extension to SpkrRec is to add a Cluster Discovery module, ClustD, and associated controller interacting with the existing k-Means centroid computing engine. ClustD will iterate several times over the entire training dataset starting with  $K=1$ , and after each iteration increasing the value of  $K$  by one. After reaching a stopping

criteria, the final value of K will be used by SpkrRec in future on-line centroid estimation. During each iteration, the error (Euclidean distance) between each dataset element and the current cluster's centroid (computed via k-Means) is computed in ClustD. The sum of all element's squared errors, SSE, is also computed in each iteration and stored for that iteration. The stopping criteria can check SSE versus a threshold; when SSE remains below the threshold for several iterations, the optimal K value has been reached. Figure 42 below illustrates the discovery of the optimal K (3) value for a hypothetical dataset.

For K=3, the SSE value is always below threshold Th1. The Th1 value can be estimated during off-line training and should result in a K value which satisfies the SpkrRev classification accuracy requirement.

This is the so-called elbow method for determining the K value and is easy to integrate with the existing k-Means centroid computing engine.

#### **14.2.2 Simulated Annealing schedule**

Another open problem in on-line k-Means clustering with Lloyd's online algorithm is the optimal division factor applied to the difference term during each iteration. This is the  $1/n$  division in Fig. 10 in Chapter 7. We propose to extend SpkrRec to use the following update rule instead of  $1/n$  :

$$z_i = z_i + \alpha * (x - z_i) \text{ with } \alpha \in (0, 1) \quad (14.4)$$

where

$$\alpha = \exp[-\beta * r_{xz_j}] \quad (14.5)$$

here

$$\beta = \frac{1}{T} \quad (14.6)$$

the inverse temperature,  $x$  is the a new candidate member for current cluster, and

$$r_{xz_j} \quad (14.7)$$

is the Euclidean distance between  $x$  and the cluster's centroid  $z_j$ .

This new learning rule will require an additional  $\exp()$  operation, control registers, controller states, and a look-up table for implementing it. Simulated annealing learning rule for k-Means has been shown to avoid the problem of locally optimal solutions (estimated centroids) due to initial value dependence of k-Means and also due to its sensitivity to outliers, see (Takano, 2018).

### **14.2.3 Integration with a GMM-scoring block and Command Recognition for Robotic Arm control**

A natural extension of our SpkrRec work is integration with the GMM scoring block such as the one in (Gianelli et al., 2019) or (Chan et al., 2004). The resulting GMM processor can then be used to generate log-likelihood estimates of word models (spotted words in a sentence). Spotted word recognition is also based on MFCC features and is very similar to speaker recognition we considered earlier. Fig. 43 shows a spotted word (isolated word) recognizer from (Elemery, 2011) which combines GMM and Hidden

Markov Models, HMMs, for a known set of command words, used to control a diadic manipulator robotic arm TR45. Both GMM and HMM models are independently developed and trained with the same set of command words characterized by k-dimensional MFCC feature vectors. Both models are used in real time robotic arm operation for increased recognition accuracy.

The Likelihood block after the GMM processor (SpkrRec and GMM-score) implements the following equation:

$$L_{GMM} = \frac{1}{T} \sum_{t=1}^T \log P_{GM}(X_t) \quad (14.8)$$

Here

$$P_{GM}(X_t) \quad (14.9)$$

is the Gaussian mixture density with M Gaussian k-dimensional components, evaluated at the t-th observation of input k-dimensional word X. The likelihood of the HMM model's output is computed as well and the final Decision block selects an Action for the robotic arm based on the winning likelihood value.

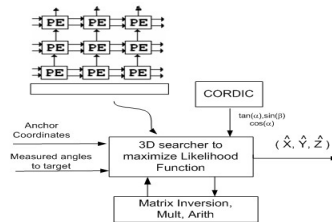
### **14.3 Fully Connected layers in DNNs - FC-Accel Extensions**

One extension of FC-Accel, to reduce total power at the expense of a small increase in latency, can target a reduction of the number of HBMs in the system. We can reduce the 128 HBMs for 128 PEs (1D array 128x1 of 128 PEs) architecture to a 16 x 1 1D hardware array of PEs. The number of HBMs

is reduced to 16, and each is dedicated to one of the 16 PEs. This direct 16-to-16 mapping also avoids the need for a NoC routing infrastructure. We call this a 1D-16-FC-Accel architecture where each PE is still based on a  $8 \times 8$  matrix-vector multiplier. To evaluate the entire matrix-vector product in Fig. 22 will require 8 passes of the new  $16 \times 1$  array. 16 HBMs dedicated to the 16 PEs are used for storing weights for each pass. Each pass uses a dedicated page of weights from the HBMs.

## CMOS VLSI architecture for NLS Levenberg-Marquardt Hardware Accelerator

- Node architecture includes CPU with hardware accelerators : systolic array processing for low power. Self-timed (clock-less) processing is used in specific cases to obtain lowest possible power in each accelerator and fastest real-time response.
- Acceleration of 3D coordinate computation : ultra low power sub-threshold design ; architectural techniques for low power : dense (super) pipelining ; two-phase latch-based design ; architectural optimizations to reduce idle time of any memory cells in order to reduce leakage energy.



- Solve system of 3 linear equations in 3 unknowns ( for 3D ) or 2 linear equations in 2 unknowns ( for 2D ) ; assume fixed-point math
- Equations require  $\tan(\cdot)$  of an angle : - can use systolic CORDIC algorithm; fixed-point math
- To handle angle measurement errors, a Maximum-Likelihood algorithm, such as Stansfield algorithm can be implemented in a systolic array : it requires matrix multiplication and matrix inversion : example likelihood function =  $(A^T R^{-1} S^{-1} A)^{-1} A^T R^{-1} S^{-1} b$
- For Ultra-Low-Power (ULP) design, use sub-threshold circuit techniques : these have been used for an FFT hardware accelerator (Univ. Michigan); for example a supply voltage of 270 mV , clock frequency of 30 MHz, and resulting throughput of 240 Msamples/sec

## 3D AOA Coordinates Computation Hardware Accelerator

- Current Results from floating point sims : 3D coordinates via non-linear Least squares Levenberg-Marquardt (non-Bayesian estimation ) - simulation examples : Red-True Target, Blue-Estimates

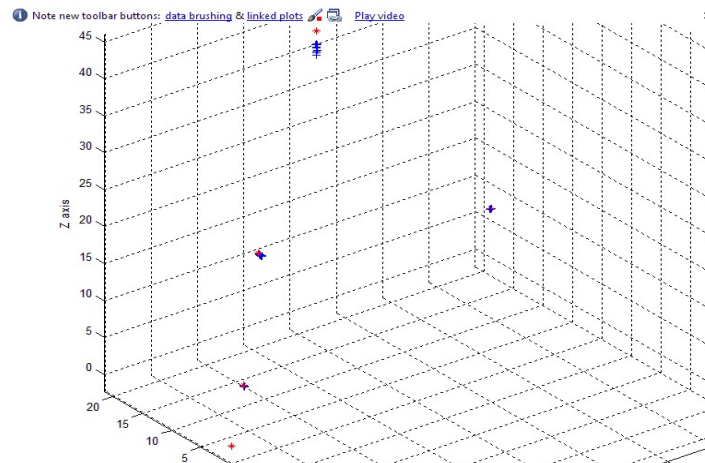


Figure 38: Matlab simulation of NLS-LM processing pipeline and a possible systolic array realization.



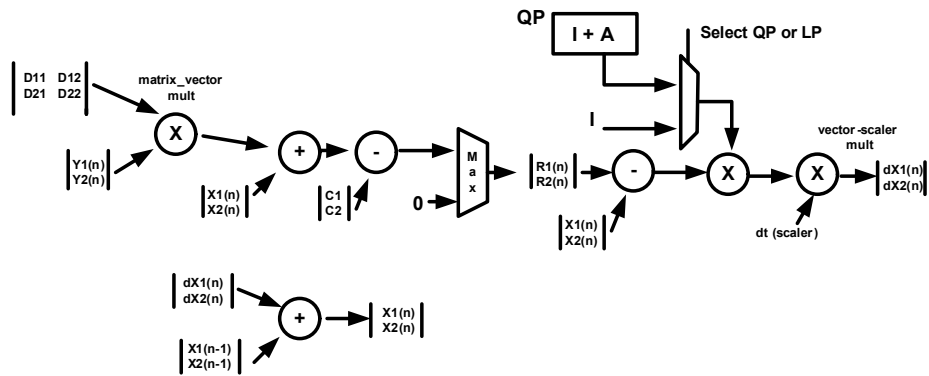


Figure 39: RNN architecture extension for solving a constrained Quadratic Program, QP, as well as an LP. .

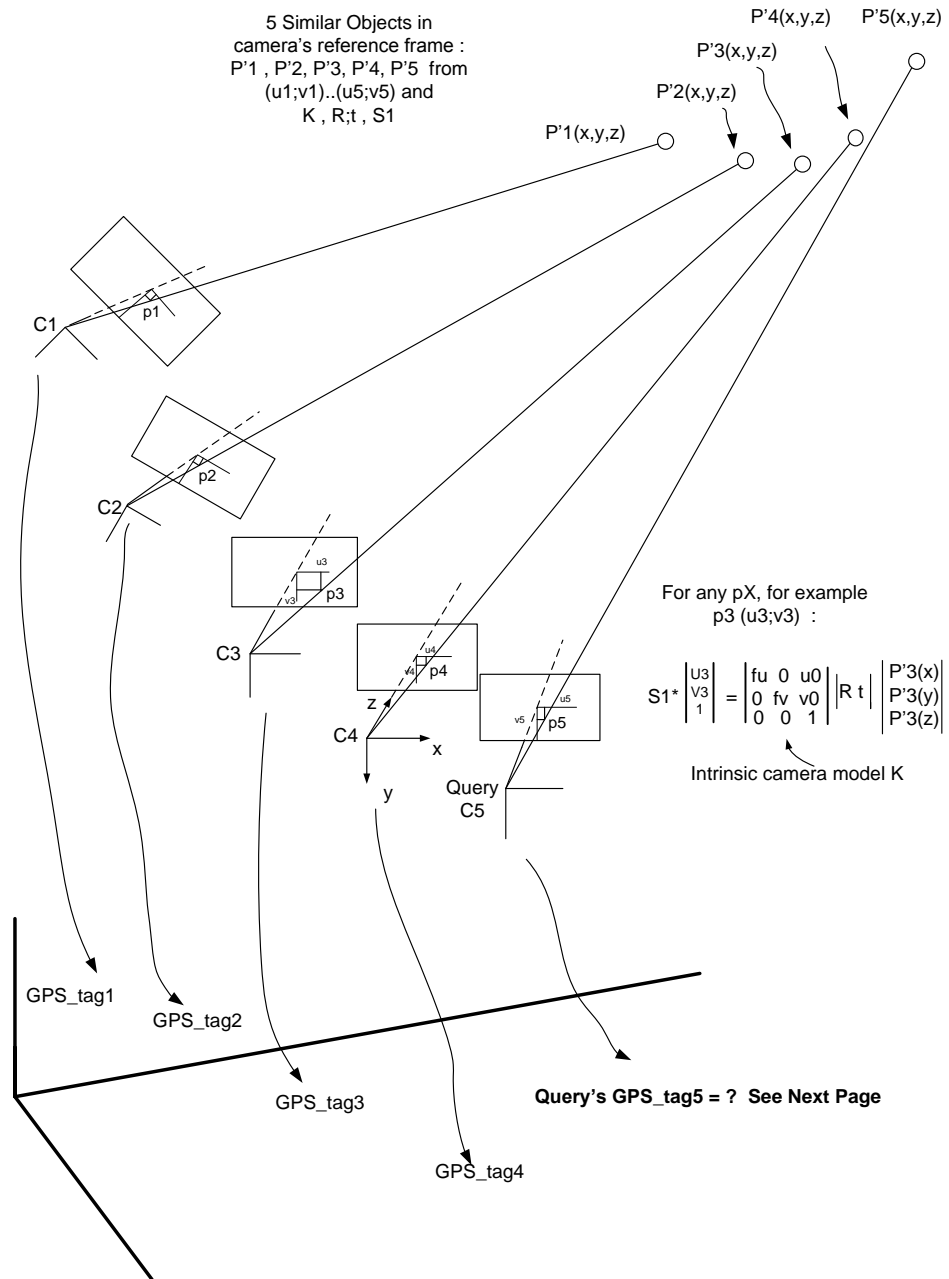
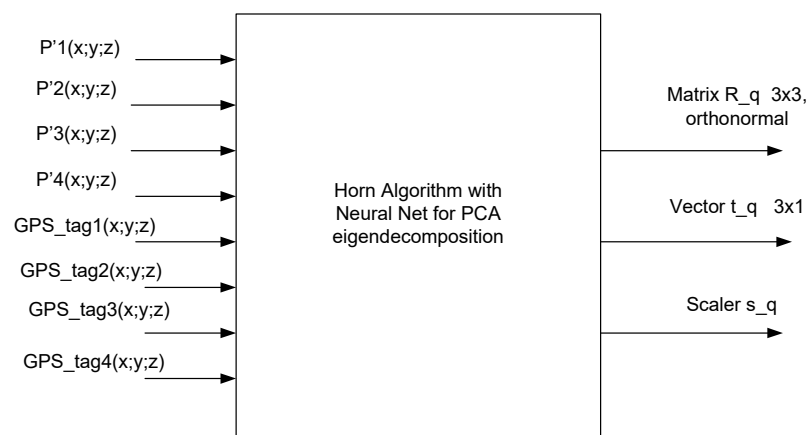


Figure 40: SFM processing to generate  $P'1 \dots P'5$  in local camera coordinate system .



$$\text{Query's } GPS\_tag5 = t_q + s_q \cdot R_q \cdot P'5$$

Example Intrinsic Camera Model K :  
 For Virtual Calibrated Camera with resolution 640x480  
 Principal (center) point  $(u_c ; v_c) = (320; 240)$   
 Effective focal lengths  $f_u = f_v = 800$

Figure 41: Module for Horn acceleration, with neural network for PCA and eigenvalue decomposition.

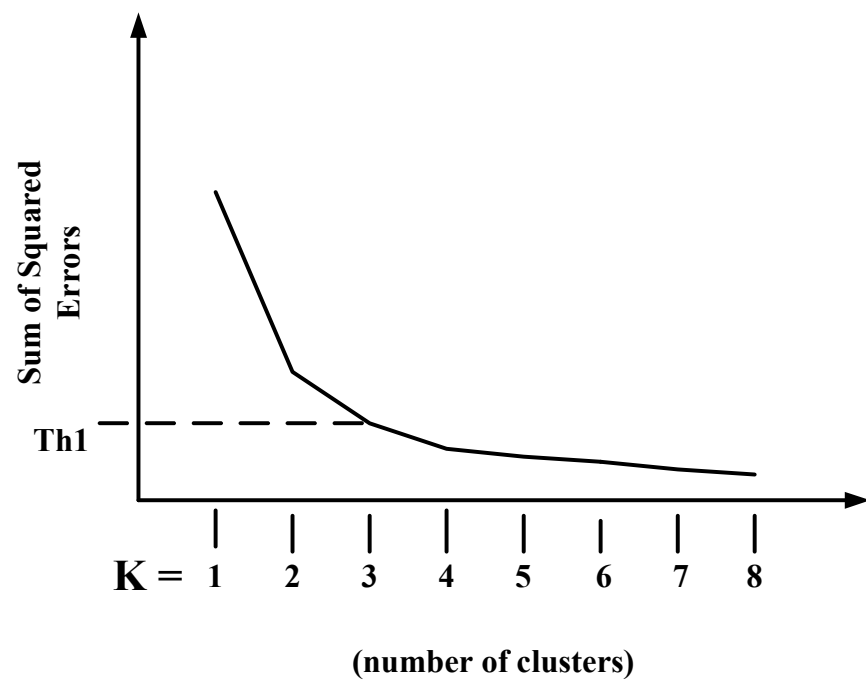


Figure 42: The elbow method for optimal K selection

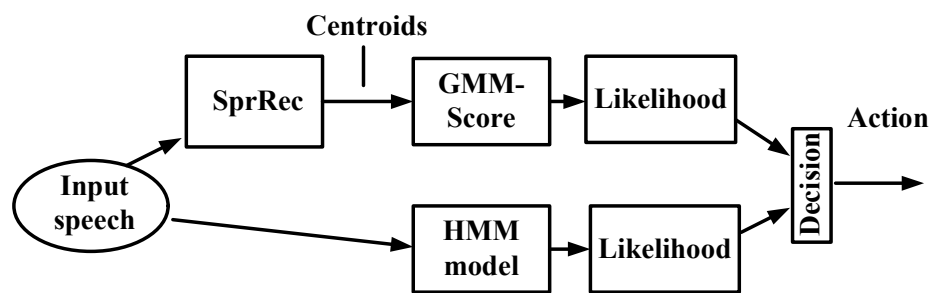


Figure 43: Spotted word recognition for control of diadic manipulator robotic arm TR45.

## CHAPTER 15

### CONCLUSION

The problems of algorithm-hardware co-design for smart home AI devices require many innovative technical solutions as we have seen in the previous chapters. In this concluding chapter I'll summarize the main contributions of my research to each of the three AI-enabling technology areas considered above.

In Part I, Spatial Self-Localization, as the number of edge nodes increases in a network, their localization and tracking will increase the computational load on the existing anchors, who are neighbors to the new node. This can be solved if the newly added edge node has on-board self-localization capability, such as a dedicated low-power, low latency processor accelerator for estimating the new node's position and a AOA measurement sensor.

We have shown that accelerators based on our RNN architecture, for 2D and 3D cases with small AOA measurement error, can provide low-power, low-latency, and accurate 2D and 3D coordinate estimation. Matrix inversion, Jacobian, and Hessian matrix processing is avoided with the proposed RNN architecture. An increase in the number of new nodes does not increase the complexity of the proposed RNN.

Our number of iterations for the primal-dual solution is:

$$O(\sqrt{N} * \log \frac{N}{\epsilon}) \quad (15.1)$$

where  $N$  is the number of coordinates and  $\epsilon$  is the desired accuracy  $\epsilon > 0$  (epsilon away).

The proposed RNN architecture has the following number of operations complexity, per iteration :

$$O(2 * N^2) \quad (15.2)$$

this is multiplications with  $N$  the dimension of  $\theta$ . An additional  $O(N)$  number of additions, subtractions, and compare-vs-zero operations is required. Note that since batch Least Squares is used (with a batch size of  $N$ ) generating a corresponding primal-dual linear program for each batch, an increase in the number of nodes does not affect the above complexity.

In Part II, Speaker Recognition, real-time scoring (evaluation) of all GMM models (for all enrolled speakers) with every incoming MFCC frame can present a considerable computational challenge with implications on overall hardware latency, power, and area.

To solve this problem we have proposed a novel low-power architecture for SI by combining real-time k-means clustering and GMM scoring of each cluster's centroid. This has achieved drastic reduction in number of MFCC frames to process with the same number of speaker models. To minimize overhead of the additional Cluster building Layer, CL, for SI, we have discussed novel parallel Euclidean distance computing and online centroid updating micro-architectures. Our implementation has achieved a 6x decrease in total dissipated energy for classifying among ten speakers with an integrated GMM-centroid scoring system

The proposed solution has an upper bound on time complexity as

$$O(k * D * N) \quad (15.3)$$

, where  $k$  is number of clusters,  $D$  is dimensions of the data vectors,  $N$  is number of data vectors to cluster.

The total number of computations, per iteration, is upper-bounded by  $O(D*k + 2*D)$  additions,  $O(D*k)$  multiplications,  $O(k*\log(k))$  3-way comparisons, and  $O(D)$  divides where  $D$  is dimension of the data vector and  $k$  is number of clusters

In Part III, Fully Connected Layers in DNNs, the dense (non-sparse) weights matrix can limit real-time inferencing performance of the DNN due to large latencies of the FC layers. Storing the weights in off-chip DDR4 64-bit wide SDRAM places severe bandwidth and number of read-accesses constraints on any accelerator (or Processing Element, PE) design that attempts to solve the FC layer latency problem.

To solve this problem we have proposed a novel HBM based FC layer accelerator with a very wide 1024-bit read-access bus, a novel 8x128 to 1024 FIFO based data-prefetch architecture, and a novel pipelined PE for matrix-vector multiplication. A 1D array of 128 8x8 PEs and HBMs does not need mesh routing (or Network On Chip, NoC, routing) and does not use compression of the DNN model's weights. We use full 16 bit fixed-point precision for both data and weights. The proposed FC-Accel architecture is based on column of block(tiles) decomposition of the original weights matrix. The proposed architecture can be up-scaled to the larger FC6 and FC7 layers in AlexNet and in VGG-16. For AlexNet FC6 our accelerator achieves a 60 % latency reduction over a compression-based FC accelerator. For VGG16 FC6 our accelerator achieves a 3.5 % latency reduction over a compression-based FC accelerator.



Total time complexity is  $O( n*[2*\text{ceil}(n/p-1) - 1] + n]*d )$  clock cycles for a  $m \times n$  matrix of weights, with tile size of  $t$ ,  $p = m/t$  PEs, and  $d$  is the time for a basic scalar addition or multiplication. Total number of operations ( multiplications and additions ) is  $O( m*2(n-1) )$ .

## **APPENDIX**

### **COPYRIGHT PERMISSIONS**

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis.

The list of the articles include (Iliev and Trivedi, 2019) IEEE Sensors Letters, (Iliev et al., 2019) IEEE Embedded Systems Letters, (Iliev and Trivedi, 2017) IEEE International Conference on Computer Design, ICCD, (Iliev and Paprotny, 2015) IEEE Sensors Journal , (Gianelli et al., 2019) IEEE Symposium on Low-Power and High-Speed Chips and Systems, (Salarian et al., 2018) IEEE Transactions on Multimedia, and (Salarian et al., 2016) IEEE International Symposium on Multimedia (ISM).



## Low-Power Sensor Localization in Three-Dimensional Using a Recurrent Neural Network

Author: Nick Iliev

Publication: IEEE Sensors Letters

Publisher: IEEE

Date: Dec. 2019

Copyright © 2019, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)



## Low Power Speaker Identification by Integrated Clustering and Gaussian Mixture Model Scoring

Author: Nick Iliev

Publication: IEEE Embedded Systems Letters

Publisher: IEEE

Date: March 2020

Copyright © 2020, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)



## Low Power Spatial Localization of Mobile Sensors with Recurrent Neural Network

Conference Proceedings: 2017 IEEE International Conference on Computer Design (ICCD)

Author: Nick Iliev

Publisher: IEEE

Date: Nov. 2017

Copyright © 2017, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)



## Review and Comparison of Spatial Localization Methods for Low-Power Wireless Sensor Networks

Author: Nick Iliev

Publication: IEEE Sensors Journal

Publisher: IEEE

Date: Oct. 2015

Copyright © 2015, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)



## Low Power Speaker Identification using Look Up-free Gaussian Mixture Model in CMOS

Conference Proceedings:

2019 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)

Author: Alberto Gianelli

Publisher: IEEE

Date: April 2019

Copyright © 2019, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE



## Improved Image-Based Localization Using SFM and Modified Coordinate System Transfer

Author: Mahdi Salarian

Publication: Multimedia, IEEE Transactions on

Publisher: IEEE

Date: Dec. 2018

Copyright © 2018, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)





## Accurate Image Based Localization by Applying SFM and Coordinate System Registration

Conference Proceedings: 2016 IEEE International Symposium on Multimedia (ISM)

Author: Mahdi Salarian

Publisher: IEEE

Date: Dec. 2016

Copyright © 2016, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE](#)

## CITED LITERATURE

- [Alecci, 2017] Alecci, J.: Transfer learning for rain detection in images. Department of Computer Science, Ph.D., TUDelft University, 2017.
- [Arafa et al. , 2015] Arafa, A., Dalmiya, S., and et al: Angle-of-arrival reception for optical wireless location technology. OSA Optics Express, 23(6):7755–7766, 2015.
- [Bergen et al. , 2018] Bergen, M. H., Schaal, F. S., and et al: Toward the implementation of a universal angle-based optical indoor positioning system. Front. of Optoelectronics, 2018.
- [Cai and et al, 2014] Cai, X. and et al: Complexity analysis of primal-dual interior-point methods for linear optimization based on a new parametric kernel function with a trigonometric barrier term. Abstract and Applied Analysis, <http://dx.doi.org/10.1155/2014/710158>:1–11, 2014.
- [Caicedo and et al, 2015] Caicedo, J. and et al: Active object localization with deep reinforcement learning. International Conference on Computer Vision, ICCV, 17:2488–2496, 2015.
- [Campbell et al. , 2006] Campbell, W. M., Sturim, D. E., and Reynolds, D. A.: Support vector machines using gmm supervectors for speaker verification. IEEE Signal Process. Lett., 13(5):308–311, 2006.
- [Chan et al. , 2004] Chan, A., Mosur, R., A. R., and Sherwani, J.: Four-layer categorization scheme of fast gmm computation techniques in large vocabulary continuous speech recognition systems. Proc. INTERSPEECH'04, pages 689–692, 2004.
- [Chen and et al, 2014] Chen, T. and et al: Diannao: A smallfootprint high-throughput accelerator for ubiquitous machine-learning. ACM Sigplan Notices, 49(4):269–284, 2014.
- [Du and et al, 2015] Du, Z. and et al: Shidiannao: Shifting vision processing closer to the sensor. ACM SIGARCH Computer Architecture News, 43(3):92–104, 2015.
- [Elemery, 2011] Elemery, J. e. a.: Hidden markov model/gaussian mixture models (hmm/gmm) based voice command system: A way to improve the control of remotely operated robot arm tr45. Scientific Research and Essays, available online at <http://www.academicjournals.org/SRE>, 6(2):341–350, 2011.
- [Fuhrman and et al, 2019] Fuhrman, T. and et al: An interactive indoor drone assistant. <https://arxiv.org/abs/1912.04235>, 2019.

- [Gao and et al, 2017] Gao, M. and et al: Tetris: Scalable and efficient neural network acceleration with 3d memory. Platform Lab, Stanford Univ. <https://platformlab.stanford.edu/pdf/MingyuGao.pdf>, pages 1–25, 2017.
- [Garofolo et al. , 1990] Garofolo, J., Lamel, J., and et al: Timit acoustic-phonetic continuous speech corpus cd-rom. National Institute of Standards and Technology and DARPA, 1990.
- [Gasparri and Pascucci, 2010] Gasparri, A. and Pascucci, F.: An interlaced extended information filter for self-localization in sensor networks. IEEE Transactions on Mobile Computing, 9(10):1491–1504, 2010.
- [Gavish and Weiss, 1992] Gavish, M. and Weiss, A. J.: Performance analysis of bearing-only target location algorithm. IEEE Trans. Aerosp. Electron. Syst., 28:817–828, 1992.
- [Georgiev and et al, 2004] Georgiev, A. and et al: Localization methods for a mobile robot in urban environments. IEEE Transactions on Robotics, 20(5):851–864, 2004.
- [Gianelli et al. , 2019] Gianelli, A., Iliev, N., Nasrin, S., and Trivedi, A. R.: Low power speaker identification using look up-free gaussian mixture model in cmos. In IEEE Symposium on Low-Power and High-Speed Chips and Systems, 2019.
- [Google, 2020] Google: Edge tpu, google’s purpose-built asic designed to run inference at the edge. <https://cloud.google.com/edge-tpu/>, 2020.
- [Hammerstrom, 1990] Hammerstrom, D.: A vlsi architecture for high-performance, low-cost, on-chip learning. Neural Networks, 1990., 1990 IJCNN International Joint Conference on, pages 537–544, 1990.
- [Han and et al, 2016] Han, S. and et al: Eie: Efficient inference engine on compressed deep neural network. ACM/IEEE 43rd Annual International Symposium on Computer Architecture, pages 1–6, 2016.
- [Har-peled and et al, 2005] Har-peled, S. and et al: How fast is the k-means method. Algorithmica, 41(3):185–202, 2005.
- [Harnilovic, 2004] Harnilovic, S.: Wireless optical communication systems. Springer, 2004.
- [Huimin and et al, 2016] Huimin, L. and et al: A high performance fpga-based accelerator for large-scale convolutional neural networks. 26th International Conference on Field Programmable Logic and Applications (FPL)., pages 1–9, 2016.

- [Iliev et al. , 2019] Iliev, N., Gianelli, A., and Trivedi, A. R.: Low power speaker identification by integrated clustering and gaussian mixture model scoring. IEEE Embedded Systems Letters, DOI: 10.1109/LES.2019.2915953(5):1–4, 2019.
- [Iliev and Paprotny, 2015] Iliev, N. and Paprotny, I.: Review and comparison of spatial localization methods for low-power wireless sensor networks. IEEE Sensors Journal, 15(10):5971–5987, 2015.
- [Iliev and Trivedi, 2017] Iliev, N. and Trivedi, A. R.: Low power spatial localization of mobile sensors with recurrent neural network. IEEE International Conference on Computer Design, ICCD, pages 297–300, 2017.
- [Iliev and Trivedi, 2019] Iliev, N. and Trivedi, A. R.: Low-power sensor localization in three-dimensions using a recurrent neural network. IEEE Sensors Letters, 3(12):1–4, 2019.
- [JEDEC, 2020] JEDEC: Jesd235a,b,c high bandwidth memory (hbm) 3d dram standard. www.jedec.org, 2020.
- [Jiantao and et al, 2016] Jiantao, Q. and et al: Going deeper with embedded fpga platform for convolutional neural network. ACM FPGA' 16 Conference DOI: <http://dx.doi.org/10.1145/2847263.2847265>, pages 26–35, 2016.
- [Kai, 2009] Kai, Y.: Hardware optimization and exploration of feature extraction and feature scoring for speech recognition. Ph.D. Thesis Carnegie Mellon University, 2009.
- [Karkooti et al. , 2005] Karkooti, M., Cavallaro, J. R., and Dick, C.: Fpga implementation of matrix inversion using qrd-rls algorithm. In Asilomar Conference on Signals, Systems, and Computers, 2005.
- [Kim et al. , 2016] Kim, D., Kung, J., and et al: Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory. ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), DOI: 10.1109/ISCA.2016.41(12):380–392, 2016.
- [Kinnunen et al. , 2006] Kinnunen, T., Karpov, E., and Franti, P.: Real-time speaker identification and verification. IEEE Transactions on Audio, Speech, and Language Processing, 14(1):277–288, 2006.
- [Krizhevsky and et al, 2017] Krizhevsky, A. and et al: Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.
- [Li, 2018] Li, S.: Towards efficient hardware acceleration of deep neural networks on fpga. Ph.D. Thesis University of Pittsburgh, 2018.

- [McLaughlin et al. , 1999] McLaughlin, J., Reynolds, D. A., and Gleason, T.: A study of computation speed-ups of the gmm-ubm speaker recognition system. Proc. Eurospeech '99, pages 1215–1218, 1999.
- [Navarro and et al, 2017] Navarro, D. and et al: Indoor positioning system based on a psd detector, precise positioning of agents in motion using aoa techniques. Sensors, doi:10.3390/s17092124 [www.mdpi.com/journal/sensors](http://www.mdpi.com/journal/sensors), 17:1–26, 2017.
- [Ning and et al, 2016] Ning, L. and et al: A multistage dataflow implementation of a deep convolutional neural network based on fpga for high-speed object recognition. IEEE Southwest Symposium on Image Analysis and Interpretation, pages 165–168, 2016.
- [NVIDIA, 2018] NVIDIA: cusparse. <http://developer.nvidia.com/cusparse>., 2018.
- [Oskoei and Amiri, 2006] Oskoei, H. G. and Amiri, N. M.: An efficient simplified neural network for solving linear and quadratic programming problems. Applied Mathematics and COmputation Journal, 175:452–464, 2006.
- [Pazhayaveetil, 2008] Pazhayaveetil, U. C.: Hardware implementation of a low power speech recognition system. Ph.D. Thesis North Carolina State University, 2008.
- [Popoola and et al, 2019] Popoola, O. and et al: Optical boundaries for led-based indoor positioning system. Computation, [www.mdpi.com/journal/computation](http://www.mdpi.com/journal/computation) doi:10.3390/computation7010007, pages 1–25, 2019.
- [Quinn, 2004] Quinn, M.: Parallel programming in c with mpi and openmp. McGraw-Hill, New York, NY., 2004.
- [Ramos et al. , 2013] Ramos, L. R., Lopez, G. M., and et al: Real-time speaker verification system implemented on reconfigurable hardware. Springer Journal Sign. Processing., 4(12):89–103, 2013.
- [Reynolds and Rose, 1995] Reynolds, D. A. and Rose, R. C.: Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Transactions on Speech Audio Processing., 3(1):72–83, 1995.
- [Salarian et al. , 2016] Salarian, M., Iliev, N., and Ansari, R.: Accurate image based localization by applying sfm and coordinate system registration. IEEE International Symposium on Multimedia (ISM), DOI: 10.1109/ISM.2016.0045:189–192, 2016.
- [Salarian et al. , 2018] Salarian, M., Iliev, N., Cetin, A., and Ansari, R.: Improved image-based localization using sfm and modified coordinate system transfer. IEEE Transactions on

Multimedia, 20(12):3298–3310, 2018.

- [Simonyan and et al, 2014] Simonyan, K. and et al: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.
- [Sugadev and et al, 2016] Sugadev, M. and et al: Design of voice and hand gesture controlled quadcopter. Research Journal of Pharmaceutical, Biological and Chemical Sciences, 7(2):15–25, 2016.
- [Sun et al. , 2003] Sun, B., Liu, W., and Zhong, Q.: Hierarchical speaker identification using speaker clustering. IEEE Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, pages 299–304, 2003.
- [Sze and et al, 2017] Sze, V. and et al: Efficient processing of deep neural networks: A tutorial and survey. IEEE Proceedings, 105(12):2295–2329, 2017.
- [Takano, 2018] Takano, J. e. a.: Harmonic mean similarity based quantum annealing for k-means. INNS Conference on Big Data and Deep Learning 2018, [www.elsevier.com/locate/procedia\(144\):298–305](http://www.elsevier.com/locate/procedia(144):298–305), 2018.
- [Tang and Dodds, 2007] Tang, B. and Dodds, D.: Synchronization of weak indoor gps signals with doppler using a segmented matched filter and accumulation. Proc. IEEE Canadian Conference on Electrical and Computer Engineering, pages 1531–1534, 2007.
- [Taparugssanagorn et al. , 2013] Taparugssanagorn, A., Siwamogsatham, S., and Pomalaza-Ráez, C.: A hexagonal coverage led-id indoor positioning based on tdoa with extended kalman filter. In 2013 IEEE 37th Annual Computer Software and Applications Conference, pages 742–747. IEEE, 2013.
- [Vaghefi et al. , 2010] Vaghefi, R. M., Gholami, M. R., and Strom, E. G.: Bearing-only target localization with uncertainties in observer position. IEEE 21st PIMRC Workshop, Sept.26-30, 2010.
- [Wang et al. , 2013] Wang, T. Q., Sekercioglu, Y. A., and et al: Position accuracy of time-of-arrival based ranging using visible light with application in indoor localization systems. Journal of Lightwave Technology, 31(20):3302–3308, 2013.
- [Woszczyna, 1998] Woszczyna, M.: Fast speaker independent large vocabulary continuous speech recognition. Ph.D. Thesis Universitat Karlsruhe; Institut fur Logik, Komplexitat and Deduktionssysteme, 1998.

- [Yan et al. , 2010] Yan, M., Feng, B., and Song, T.: On matrix inversion for lte mimo applications using texas instruments floating point dsp. In IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS, pages 633–636. IEEE, 2010.
- [Yuran and et al, 2017] Yuran, Q. and et al: Fpga-accelerated deep convolutional neural networks for high throughput and energy efficiency. Concurrency Computat: Pract. Exper, Wiley Online Library, 29(e3850):1–20, 2017.

## VITA

# Nikolai Iliev

Education	Ph.D. Electrical and Computer Engineering University of Illinois at Chicago	2013 – 2020
	M.S. Electrical Engineering <b>Thesis: Failure isolation in linear stochastic systems : the associative recall approach</b> Committee: prof. J. Sklansky, prof. A. Stubberud, prof. B. Bavarian, prof. G. Hostetter ECE Dept. University of California Irvine	1986 – 1988
	B.S. Electrical Engineering <b>Senior Thesis: ECG Analog Circuits and DSP for data acquisition</b> Project Lead: prof. A. Sahakian EECS Dept. Northwestern University	1982 – 1986
Patents	<b>N.J. Iliev</b> , Freescale Semiconductor, Inc., Programmable Phase Mapping and Phase Rotation Modulator and Method, US7412008B2.	
	<b>N.J. Iliev</b> , Freescale Semiconductor, Inc., Circuit and Method for Generating Fixed Point Vector Dot Product and Matrix Vector Values, US8165214B2.	
	Amit R. Trivedi, Balajee Vamanan, and <b>Nick Iliev</b> , “Flexible and High-Speed Packet Classification in Software Defined Networking using Self-Organizing Maps,” UIC ECE 2017	
	<b>N. Iliev</b> , “LMS hardware accelerator for 4G OFDM comb-type channel estimation in the time domain”, Freescale Semiconductor Inc. 2008	
	<b>N. Iliev</b> , “Channel estimation hardware accelerator for time domain real FIR channel models”, Freescale Semiconductor Inc. 2007	



**N. Iliev**, "GF multiplicative inverse/exponentiation hardware unit", Freescale Semiconductor Inc. 2006

**N. Iliev**, "Distributed Systolic vector dot-product matrix-vector product hardware unit", Freescale Semiconductor Inc. 2006

**N. Iliev**, "Digital low pass filter (LPFIR) with programmable order and coefficients", Freescale Semiconductor Inc. 2005

**N. Iliev**, "Mixed-Signal simulator model for approximate simulation of analog biquad IIR filter stages with digital busses", Motorola Inc. 2004

**N. Iliev**, "Encoding of Symbolic Inputs in Serial Decomposition of PLAs ; area/speed optimization for random logic ASIC blocks", Intel Corp. 1997

**N. Iliev**, "FDDI (optical fiber token ring) Station Manager controller architecture and topology map builder", IBM Corp. 1993

#### Publications

**N. Iliev** and A. R. Trivedi, "Low-Power Sensor Localization in Three-Dimensional Using a Recurrent Neural Network," in IEEE Sensors Letters, vol. 3, no. 12, pp. 1-4, Dec. 2019, Art no. 3502504.

**N. Iliev**, A. Gianelli and A. R. Trivedi, "Low Power Speaker Identification by Integrated Clustering and Gaussian Mixture Model Scoring," in IEEE Embedded Systems Letters, vol. 12, no. 1, pp. 9-12, March 2020.

A. Gianelli, **N. Iliev**, S. Nasrin, M. Graziano and A. R. Trivedi, "Low Power Speaker Identification using Look Up-free Gaussian Mixture Model in CMOS," 2019 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS), Yokohama, Japan, 2019, pp. 1-3.

S. Hung, **N. Iliev**, B. Vamanan and A. R. Trivedi, "Self-Organizing Maps-Based Flexible and High-Speed Packet Classification in Software Defined Networking," 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSI-SID), Delhi, NCR, India, 2019, pp. 545-546.

M. Salarian, **N. Iliev**, A. E. Çetin and R. Ansari, "Improved Image-Based Localization Using SFM and Modified Coordinate System Transfer," in IEEE Transactions on Multimedia, vol. 20, no. 12, pp. 3298-3310, Dec.

2018.

**N. Iliev** and A. R. Trivedi, "Low Power Spatial Localization of Mobile Sensors with Recurrent Neural Network," 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, 2017, pp. 297-300.

M. Salarian, **N. Iliev** and R. Ansari, "Accurate Image Based Localization by Applying SFM and Coordinate System Registration," 2016 IEEE International Symposium on Multimedia (ISM), San Jose, CA, 2016, pp. 189-192.

**N. Iliev** and I. Paprotny, "Review and Comparison of Spatial Localization Methods for Low-Power Wireless Sensor Networks," in IEEE Sensors Journal, vol. 15, no. 10, pp. 5971-5987, Oct. 2015.

Kun Fang, **Nick Iliev**, Ehsan Noohi, Suyu Zhang, Zhichun Zhu, "Thread-fair memory request reordering," in IEEE 3rd JILP Workshop on Computer Architecture, Energy Track, 2012

N. Jachimiec, **N. Iliev** and J. Stine, "Strategies for VLSI implementations of finite field inversion algorithms," 48th Midwest Symposium on Circuits and Systems, 2005., Covington, KY, 2005, pp. 1589-1592 Vol. 2.

J. E. Stine, **N. Iliev**, et al., "A framework for high-level synthesis of system on chip designs," 2005 IEEE International Conference on Micro-electronic Systems Education (MSE'05), Anaheim, CA, USA, 2005, pp. 67-68.

**N. Iliev**, J. E. Stine and N. Jachimiec, "Parallel programmable finite field GF (2/sup m/) multipliers," IEEE Computer Society Annual Symposium on VLSI, Lafayette, LA, USA, 2004, pp. 299-302.

D Reeves, **N Iliev**, "Group address recognition with perfect hashing hardware," in Proc. of IEEE Workshop on Architecture and Implementation of High-Performance Communication Subsystems, 1992

A. Rindos, **N. Iliev** and I. Viniotis, "FDDI (frame level) error management design: station parameters and management evaluation process," Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics, Charlottesville, VA, USA, 1991, pp. 819-

824 vol.2.