# STOCHASTIC COMPUTING SYSTEM HARDWARE DESIGN FOR CONVOLUTIONAL NEURAL NETWORKS OPTIMIZED FOR ACCURACY, AREA AND ENERGY EFFICIENCY

HAMDAN USAMAH HAMDAN ABDELLATEF

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JANUARY 2020

## DEDICATION

This thesis is dedicated to my wonderful parents who have raised me to be the person I am today, my beloved wife who supported me through my study, my lovely daughter, and my family.

**ACKNOWLEDGEMENT**

The past three years were the most challenging but productive in my life. Now, the Ph.D. journey comes to an end. I am blessed with the completion of this thesis. Alhamdulillah, I am deeply grateful and thankful to Almighty Allah, who made me able and gave me the strength to overcome the hardships and complete this thesis. I would like to express my appreciation to the people that have involved directly or indirectly in my work.

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Mohamed Khalil Mohd. Hani for his guidance, enthusiastic encouragement, and useful critiques of this research work. I have gained valuable knowledge during his supervision. My very high appreciation for him is not only for academic supervision but also for sharing his philosophy of life and treating me as his son.

My sincere appreciation also goes to my co-supervisor Assoc. Prof. Dr. Shaikh Nasir Bin Shaikh Husin for his guidance and dedication towards my research work. It was also a privilege to work closely with the members of VeCAD Lab, namely Sayed Omid Ayat, Mohd Ikmal Fitri bin Maruzuki, Arbab Alamgir, and Shehryar Masud Rizvi.

Most importantly, I would like to thank my family, especially my dear parents and my beloved wife, for their love and the boundless supports during this journey. Also, I thank my daughter for creating happiness and motivation during the past year. I am thankful to my friends for making me feel at home while being too far away. This Ph.D. journey would have been impossible without all of you. Thank you.

## ABSTRACT

Stochastic computing (SC) is an alternative computing paradigm that can lead to designs that offer lower area and power consumption compared to that of the conventional binary-encoded (BE) deterministic computing. In SC, numbers are encoded as a bit-stream of '0's and '1's, where SC computation elements (or functions) operate on one or more bit-streams. To obtain accurate results, some functions require the bit-streams to be correlated, while others require uncorrelated bit-streams or a combination of both. The relationship between SC function accuracy and correlation is not well studied in previous works. Thus, managing the correlation across the SC system is a key challenge in the effort to achieve optimum accuracy. In addition, to perform SC computation, the input values are converted from BE domain to SC; then on the completion of the computation, back to BE to obtain the results. The conversion processes require circuitry that typically consume over 80% of the overall SC system area, hence this is another key challenge of the problem. To address the above mentioned challenges, this thesis proposes a framework of an end-to-end system design optimized for accuracy and area. The framework provides guidelines to design an effective SC function or system that exploit correlation. This framework is applied in designing the SC functional units and the complete SC system for convolutional neural network (CNN), which is the dominant approach in the implementation of recognition systems. This thesis shows that although CNN is a compute-intensive and resource-demanding algorithm, through the proposed SC design framework, it is possible to implement CNN in an embedded system with limited area and power budget. Several novel SC-based functions are proposed that outperform previous works and obtain significant area savings and high accuracy to replace the BE equivalent functions. These functions include inner product, max pooling, ReLU activation function, and average pooling. Then, some training considerations are specified to enable achieving low error rates for SC-based CNN. Experimental results show that the SC-based CNN attained no or minor accuracy degradation compared to BE counterpart. SC-based CNN achieves 99.6% and 96.25% classification accuracy using MNIST digit classification and AT&T face recognition datasets, respectively. Moreover, the SC-based CNN of ResNet-20 model achieves 86.5% classification accuracy using CIFAR-10 object dataset. To rapidly map an SC system into FPGA, a generic design strategy for high-level synthesis of SC computation engines is proposed. The SC-based CNN hardware on FPGA obtains the lowest resource utilization compared to previous works on FPGA-based CNN accelerators. In addition, the proposed hardware architecture achieves 277.46 GOP/s/W energy efficiency, which outperforms previous works.

# ABSTRAK

Pengkomputeran stokastik (SC) merupakan sebuah paradigma pengkomputeran alternatif yang dapat membawa kepada reka bentuk yang menawarkan penggunaan ruang dan kuasa yang lebih rendah berbanding dengan pengkomputeran berketentuan binari terkod (BE) konvensional. Dalam SC, nombor dikodkan sebagai strim-bit '0' dan '1', dengan elemen pengiraan (atau fungsi) beroperasi pada satu atau lebih strim-bit. Untuk mendapatkan keputusan yang tepat, beberapa fungsi memerlukan strim-bit yang berkorelasi, sementara yang lain memerlukan strim-bit tak berkorelasi atau gabungan kedua-duanya. Hubungan antara ketepatan dan korelasi fungsi SC tidak dikaji dengan baik dalam kajian terdahulu. Oleh itu, menguruskan korelasi seluruh sistem SC merupakan cabaran utama untuk mencapai ketepatan optimum. Selain itu, untuk melaksanakan pengiraan SC, nilai input ditukar daripada domain BE kepada SC; setelah selesai pengiraan, kembali kepada BE untuk mendapatkan keputusan. Proses penukaran ini memerlukan jalan kerja litar yang biasanya menggunakan lebih 80% daripada keseluruhan kawasan sistem SC; oleh itu, ini adalah satu lagi cabaran utama masalah ini. Bagi menangani cabaran yang dinyatakan di atas, tesis ini mencadangkan satu rangka kerja reka bentuk sistem hujung-ke-hujung yang dioptimumkan untuk ketepatan dan kawasan. Rangka kerja ini menyediakan garis panduan untuk mereka bentuk fungsi SC atau sistem berkesan yang mengeksploitasi korelasi. Rangka kerja ini digunakan dalam mereka bentuk unit berfungsi SC dan sistem SC yang lengkap bagi rangkaian neural konvolusi (CNN) yang merupakan pendekatan dominan dalam pelaksanaan sistem pengecaman. Kami menunjukkan bahawa walaupun CNN merupakan pengiraan intensif dan algoritma menuntut sumber daya, menerusi rangka kerja reka bentuk SC yang dicadangkan ini, CNN dapat dilaksanakan dalam satu sistem terbenam dengan kawasan dan bajet kuasa yang terhad. Beberapa fungsi berasaskan SC terbaharu dicadangkan yang mengatasi kajian terdahulu dan mencapai penjimatan kawasan yang ketara dan ketepatan yang tinggi untuk menggantikan BE yang setara. Fungsi ini termasuk produk dalaman, pengumpulan maksimum, fungsi pengaktifan ReLU dan pengumpulan purata. Kemudian, kami menentukan beberapa pertimbangan latihan supaya boleh mencapai kadar ralat rendah untuk CNN berasaskan SC. Keputusan eksperimen menunjukkan bahawa CNN berasaskan SC tidak menunjukkan penurunan ketepatan atau penurunan ketepatan yang kecil berbanding BE. CNN berasaskan SC masing-masing mencapai 99.6% dan 96.25% ketepatan klasifikasi menggunakan klasifikasi digit MNIST dan set data pengecaman wajah AT&T. Selain itu, CNN berasaskan SC bagi model ResNET-20 mencapai 86.5% ketepatan klasifikasi menggunakan set data objek CIFAR-10. Untuk memetakan sistem SC ke dalam FPGA dengan cepat, kami mencadangkan satu strategi reka bentuk generik untuk sintesis peringkat tinggi enjin pengiraan SC. Perkakasan CNN berasaskan SC pada FPGA memperoleh penggunaan sumber paling rendah berbanding dengan semua kajian terdahulu berkenaan pemecut CNN berasaskan FPGA. Di samping itu, seni bina perkakasan kami mencapai kecekapan tenaga 277.46 GOP/s/W yang mengatasi semua kajian terdahulu.

**TABLE OF CONTENTS**

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| ANN | – | Artificial Neural Network |
| APC | – | Accumulative Parallel Counter |
| ASIC | – | Application-Specific Integrated Circuit |
| BE | – | Binary-Encoded deterministic computing |
| BISC | – | Binary-Interfaced Stochastic computing |
| BNN | – | Binarized Neural Network |
| BP | – | BiPolar encoding |
| BRAM | – | On-chip Block RAM |
| CNN | – | Convolutional Neural Network |
| CI | – | Correlation-induced |
| CS | – | Correlation-sensitive |
| CSh | – | Circular Shift |
| CSNG | – | Correlated Stochastic Number Generator |
| CTR | – | Counter |
| DFF | – | D Flip-Flop |
| DNN | – | Deep Neural Network |
| DSP | – | Digital Signal Processing |
| DTE | – | Data Transfer Engine |
| BE/SC | – | Binary encoded-to-stochastic computing |
| FCSh | – | Flip Circular Shift |
| FEB | – | Feature Extraction Block |
| FIR | – | Finite Impulse Response |
| FF | – | Flip-Flop |
| FFT | – | Fast Fourier Transform |

| | | |
|---|---|---|
| FPGA | – | Field Programmable Gate Array |
| FSM | – | Finite State Machine |
| GOP | – | Giga OPerations |
| GPGPU | – | General-Purpose Graphics Processing Unit |
| GPP | – | General-Purpose Processor |
| GPU | – | Graphics Processing Unit |
| HDL | – | Hardware Description Language |
| HLL | – | High-Level Language |
| HLS | – | High-Level Synthesis |
| II | – | Initiation Interval |
| IOBD | – | Input Output Block Diagram |
| LCTR | – | Local down-counter |
| LFSR | – | Linear Feedback Shift Register |
| LUT | – | LookUp Table |
| MAC | – | Multiply-accumulate |
| MAE | – | Mean Absolute Error |
| MSE | – | Mean Squared Error |
| MUX | – | MUltipleXer |
| ORL | – | Olivetti Research Laboratory |
| PE | – | Processing Element |
| PL | – | Programmable Logic |
| PP | – | Progressive Precision |
| PS | – | Processing System |
| PDT | – | Probability Domain Transformation |
| PSNR | – | Peak-Signal-to-Noise Ratio |
| ReLU | – | Rectified Linear Unit |
| RC | – | Robert Cross |

| | | |
|---|---|---|
| RNG | – | Random Number Generator |
| RTL | – | Register Transfer Level |
| SC | – | Stochastic Computing |
| SCC | – | Stochastic Computing Correlation |
| SDK | – | Software Development Kit |
| SGD | – | Stochastic Gradient Descent |
| SN | – | Stochastic Number |
| SNG | – | Stochastic Number Generator |
| SNR | – | Signal-to-Noise Ratio |
| SoC | – | System-on-Chip |
| SC/BE | – | Stochastic computing-to-binary encoded |
| TFF | – | Toggle Flip-Flop |
| UP | – | UniPolar encoding |

## LIST OF SYMBOLS

| | | |
|---|---|---|
| $\alpha$ | – | Learning Rate |
| $\eta$ | – | Energy-efficiency |
| $\rho$ | – | Correlation Coefficient |
| $\delta_{SCC}$ | – | Variation in stochatic computing correlation |
| $\tau$ | – | Clock period |
| $conv()$ | – | Convolution function |
| $corr()$ | – | Cross-correlation function |
| $dB$ | – | deciBel |
| $\frac{\partial L}{\partial x}$ | – | derivative of loss with respect to input |
| $\frac{\partial L}{\partial w}$ | – | derivative of loss with respect to weights |
| $\frac{\partial L}{\partial z}$ | – | gradient from next layer |
| $k_{CS}$ | – | Amount of bit-wise circular shift |
| $L$ | – | Stochastic number Length |
| $max()$ | – | The maximum function |
| $min()$ | – | The minimum function |
| $n$ | – | Binary number bit-width |
| $Tn$ | – | Input feature maps Tile size |
| $Tm$ | – | Output feature maps Tile size |
| $Tr$ | – | Feature map height (rows) Tile size |
| $Tc$ | – | Feature map width (columns) Tile size |
| $X_b$ | – | The $b^{th}$ bit in the stochastic number (bit-stream) $X$ |
| $X_{b,i}$ | – | $b^{th}$ bit of SN of probability $x_i$ ($i^{th}$ element in vector $x$) |
| $\wedge$ | – | AND logic operator |
| $\vee$ | – | OR logic operator |

| | | |
|---|---|---|
| $\oplus$ | – | XOR logic operator |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 Background of research

Stochastic Computing (SC) is an alternative paradigm of computation that considers data as probabilities. Low-area/power cost and error tolerance are some of SC advantages. However, many challenges should be overcome before SC becomes widespread [1]. In this thesis, many SC challenges have been addressed.

The case study used to show the applicability of SC paradigm is object classification using convolutional neural network (CNN). CNN is the state-of-the-art algorithm for object recognition applications. Designing an Field Programmable Gate Array (FPGA) accelerator based on the conventional binary arithmetic calculations for deep CNNs incurs high hardware cost and energy-efficiency achievable is low. Since deep CNNs are both compute and memory intensive, it is impractical to use deep CNN accelerators in embedded system platforms that typically has limited area and power budget. Therefore, novel alternative computing paradigm such as SC is urgently needed to overcome this hurdle. In this thesis, a low-area and energy-efficient CNN hardware is designed using a High-Level Synthesis (HLS) tool targeting FPGA.

### 1.1.1 Stochastic computing

Stochastic computing is a computing paradigm, which was first introduced by Gaines [2] in the 1960s, as an alternative to the conventional binary-encoded deterministic computing technique. From hereon in this thesis, for convenience, the abbreviation, BE, is used to refer to this conventional binary-encoded deterministic computing method. In SC, data being processed are represented by bit-streams (referred

to as stochastic numbers (SN)), and the value of the data is encoded as the probability of 1s appearing in the bit-stream. For example, the data bit-stream X = 1001 encodes the value of 0.5 since the probability of 1s appearing in X is 0.5 (=2/4); there are two 1s and the bit-stream is 4 bits long.

The main advantage of an SC element is its low hardware cost and posses a high tolerance for soft errors. SC elements of multiplication, addition, and subtraction can be performed using simple logic functions. For example, as shown in Figure 1.1, the SC multiplier is an AND gate. Referring to Figure 1.1, the SNs $X$ and $Y$ are multiplied to obtain the SN $Z$. $X$ is 11010111, hence $p_x = \frac{6}{8}$. $Y$ is 11001010, hence $p_y = \frac{4}{8}$. Therefore, the output of the AND gate is $Z$ = 11000010, which means $p_z = \frac{3}{8}$. Now, $\frac{6}{8} \times \frac{4}{8} = \frac{3}{8}$; therefore, this is a multiplication operation in SC domain.

Today there is renewed interest in SC for applications in mobile and embedded devices that usually demand error-tolerant solutions with low area and low power. Consequently, in recent years, there have been more active research conducted to adopt SC in a wide range of embedded solutions for image processing [3], neural networks [4], digital filters [5], and CNNs [6, 7].

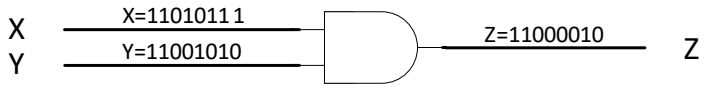



Figure 1.1: Using AND gate to perform SC multiplication

However, SC has significant drawbacks that have to be addressed before it can be viable for application in designing complex practical circuits [1]. One fundamental weakness is that an SC implementation can have a long latency arising from long input bit-streams. Data precision depends on bit-stream length; hence, higher precision requires a longer bit-stream. The crucial second drawback of SC is due to the fact that, unlike BE computation, SC operations (which are based on random numbers) do not necessarily yield consistent results, giving rise to the issue of accuracy. Moreover, the

SC circuit might lose the low-area advantage when many stochastic number generators (SNGs) are required to generate uncorrelated bit-streams. SNGs are complex circuits and can account for as much as 80% of the total circuit cost [8].

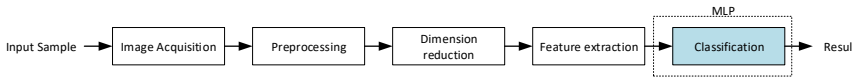
Aside from quantization errors, the correlation between SNs is also a source of inaccuracy in SC circuits. To operate correctly, some SC circuits require uncorrelated data inputs; others require correlated inputs. Hence, the inaccuracies due to correlation arise because of over-correlated operands in the former case, and in the latter case, because of operands that are not sufficiently correlated. Research work in [9] has shown that circuits that exploit correlation can result in improved accuracy in SC-based designs. It also showed that, by exploiting correlation, further gains can be made in area and delay reductions.

Previous works on utilizing correlation in SC designs were limited to the design of basic circuits or functional units, such as an edge detection filter in [10]. Typically, a large complex system, such as CNN, consists of massive amount of successive computations. For example, CNN consists of a series of layers that include convolutions, activation functions, and pooling. Such a system could not be realized previously (by exploiting correlation), because the SC-based functional units induces the correlation. Consequently, the correlation between SNs after each computation is reduced or lost, resulting in significant errors. To prevent these errors from occurring, the correlation has to be maintained end-to-end across the complete system. One may think that there is an on-the-fly solution. The designer simply regenerates the SNs by inserting conversion circuits whenever inputs have to be correlated to restore any lost correlation. However, this solution is infeasible since it introduces long conversion latency and significantly increases area cost.

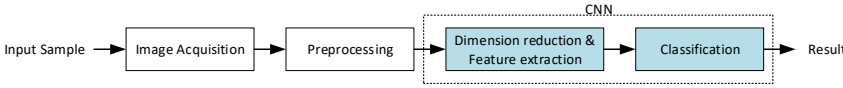
### 1.1.2 Convolutional neural network

Deep learning has emerged as a new area of machine learning research, which enables a system to automatically learn complex information and extract representations at multiple levels of abstraction. CNN is recognized as one of the most promising types

of artificial neural networks (ANNs) taking advantage of deep learning and has become the dominant approach for almost all recognition and detection tasks [11]. Originally inspired by biological processes, CNN is a special case of feed-forward ANN, which requires minimal preprocessing, and combines the feature extraction and classification tasks in one trainable block as shown in Figure 1.2. In CNN, the number of trainable parameters (weights) are reduced significantly because the weights are shared by some neurons. Recently, various CNNs have been used in image and video recognition tasks and have been successfully applied to computer vision and machine learning applications such as object recognition [12, 13], face recognition [14, 15], handwritten character and digit recognition [16, 17].




(a) Image recognition with ANN




(b) Image recognition with CNN combine dimension reduction, feature extraction, and classification processes

Figure 1.2: Processes covered by ANN and CNN in a recognition system

The typical CNN is composed of four types of processing layers: convolutional layer, an activation layer, pooling layer, and a fully-connected layer as in ANNs. Each of these layers transforms a volume of feature maps to another. To achieve acceptable classification accuracy, CNN performs millions of convolutions and sub-sampling operations with significant amount of intermediate data, where the convolution operations consume more than 90% of the computing effort in the CNN [18]. Despite its high classification accuracy, a deep CNN is highly-demanding of resources, computation effort, and energy consumption. Therefore, the implementation of CNN has become complex and challenging due to its large requirements of computation resources, which limits its applicability, especially in any resource-constrained applications.

Industrial and academic demands lead to larger depth and width of CNNs for a better quality of results and recognizing bigger datasets, resulting in complicated topologies and increased computation resources required for implementation. For example, to improve the accuracy performance for image recognition using ImageNet dataset [19], the depth of CNNs grow from 8 layers in AlexNet model [12] at 2012 to 152 layers in ResNet-152 model [13] at 2016. Therefore, a practical implementation of large-scale CNNs require high-performance server clusters with accelerators such as GPUs and FPGAs. However, there is a trend to rapidly adopt machine learning algorithms in the mobile and embedded systems. In order to deploy CNNs in these resource-constrained systems, designers must conquer the challenges of implementing resource-hungry CNNs in embedded systems with limited area and power budget. To overcome the limitation of low-power and low-hardware footprint CNN developers take advantage of highly-parallel or dedicated hardware such as General-Purpose Graphics Processing Unit (GPGPU) [12], FPGA [20], and Application-Specific Integrated Circuit (ASIC) [21] to implement CNNs.

Neural networks have very high computational complexity and high error-tolerance at the algorithmic level, which allows using SC for CNN implementation [22]. Our study is not the first work using SC to solve the resource-hungry problem of CNN. Previous works [23, 24, 25, 26, 22, 7, 27, 28, 6, 29] have implemented CNN basic functions in SC. Despite many strengths in the previous works for basic functions, there are still many gaps in finding more efficient SC circuits for the CNN basic functions with high accuracy, smaller area, cheaper conversion circuits, and lower latency. Besides, there are some limitations in the previous works regarding the applicability in the multi-stage designs since intermediate regeneration or decorrelation or special circuitry are required which introduce severe latency increase or area cost. Furthermore, some works (such as [6]) cannot be generalized to any CNN architecture or require re-arrangement of the CNN layers. The effective stochastic computing convolutional neural network (SC CNN) is an open field of research and has many problems to be addressed.

### 1.1.3 FPGA design using HLS tools

System-on-Chip (SoC) size is rapidly increasing; hence, the design productivity problem is becoming more and more serious. In the mid-1980s, the gate-level design shifted up to register transfer level (RTL) design when the number of gates exceeded 100K. A hundred thousand gates is assumed to be the maximum limit to design in several months with appropriate human resources. Nowadays, a system design commonly exceeds one million gates that requires several hundreds of thousand lines of RTL description. Consequently, it is time to shift up to a higher level of abstraction, that enables designers to have less number of descriptions and higher reusability. This is the same situation as what happened in software programming. Previously, the assembly language had to be shifted up to a higher level language like the C language to increase the scalability, and an object-oriented language such as the C++ language to increase reusability. A higher level C description involves fewer codes and accelerates simulation. These two facts are the main effects of higher-level shifting to HLS [30].

The combination of reconfigurable hardware architectures, such as FPGAs and HLS tools allow designers to achieve a specialized hardware design, and at the same time, address the time-to-market problem. FPGAs are reconfigurable integrated circuits that can be configured by the end-user to implement digital circuits. Also, since FPGAs are reconfigurable, they allow quick refinement and optimization of a hardware design compared to ASICs with no additional manufacturing costs. The designer writes or modifies the Hardware Description Language (HDL) for a component and then use an FPGA vendor tool-chain for the synthesis of the bitstream to configure the FPGA. HLS tools start from a software programmable high-level language (HLL) (e.g., C, C++, and SystemC) to automatically produce a circuit specification in HDL that performs the same function as specified in the HLL. HLS is an interesting tool for both software and hardware engineers. HLS enables software engineers to gain speed and energy efficiency of hardware, without requiring deep hardware expertise. On the other hand, for hardware engineers, HLS accelerates the design of the system at a high-level of abstraction and speed up the design space exploration. This is important in the design of complex systems such as CNNs, and suitable for FPGA design where many alternative implementations can be easily generated, deployed onto the target device, and compared [31].

Starting from early 2000, many FPGA-based accelerators have been proposed for CNN computing because general-purpose processors cannot implement CNN efficiently. Most of the early works focused on either improving the performance of computing engine or the off-chip memory communication issue. Recent works tried to improve both (performance of compuation engine and utilization of memory bandwidth), like works in [20, 32, 33, 34, 35, 36] where some have used HLS such as [20, 33]. Until now, almost all existing CNN implementations are based on BE which uses the conventional binary arithmetic, which requires huge hardware; therefore, their performance is severely limited by hardware budget and memory bandwidth of existing FPGA platforms. Nevertheless, more efficient design is required to overcome the challenge of mapping CNNs to resource-constrained environments.

## 1.2 Problem Statements

This thesis tackles many issues that can be categorized into (a) problems in existing SC CNNs, (b) limitations in SC system design, and (c) low energy-efficiency in CNN accelerators.

### 1.2.1 Summary of problems in existing CNN based on stochastic computing

There have been several attempts to design efficient SC circuits for CNNs. However, most of the previous works on CNN based on stochastic computing (SC CNN for short) [23, 27, 25, 28, 24, 37, 22, 29, 38] have two main problems. First, they do not scale to harder recognition problem. They had acceptable accuracy for simple digit classification problem using MNIST dataset, but they cannot perform well in more complex recognition problem such as object recognition. Second, they incur large overhead due to the conversion of fixed-point data and weights into stochastic bit-streams, which significantly reduces energy and area efficiency of SC CNNs.

In SC CNN, the key functions are the activation function, pooling, and the inner product. The inner product is applied in convolution. In most previous works [27, 25, 28, 24, 7, 29, 38, 6], the SC inner product function is based on the accumulative parallel counter (APC). This APC-based inner product is correlation-sensitive, so many RNGs are required to generate uncorrelated SNs to ensure accurate computations, which severely affect the circuit area cost. Furthermore, the output of an APC-based inner product is in binary format; hence, for this output to be used as an input for another SC functional unit, a regeneration circuit (or a specially designed intermediate circuitry) is required. In addition, APC-based inner product has a relatively high area compared to traditional SC inner product circuits.

The SC CNN max pooling or ReLU in previous works has a higher area than the BE implementation, or the area is not specified. Yu et al. [7] proposed accurate SC-based ReLU and max functions. However, the area cost in these functions is larger than the BE counterpart. Yu et al. in [7] proposed an RNG sharing scheme to reduce the conversion circuit area, but their inner product function (APC-based) requires uncorrelated SNs, which limited the amount reduction of area cost in the conversion circuits area.

Li et al. [6] proposed a Highly Efficient stochastic computing-based Inference Framework (HEIF) for deep neural networks. The HEIF framework introduces a new SC-based ReLU, and the authors performed a holistic and module-level optimization. However, this work has limitation in the stack order of the layers and cannot be generalized easily to other CNN models such as ResNet. In addition, this work did not discuss the conversion circuit area cost, especially that their convolution operation requires uncorrelated SNs. In addition, their SC-based ReLU has 10× higher area cost than the BE counterpart. A common limitation in all previous works on SC CNN is they did not use RNG sharing scheme, and many did not consider the conversion circuits in their area cost evaluation.

Finally, the previous works on SC average pooling used the conventional SC scaled addition, which is highly inaccurate. Moreover, if the CNN functions are changed, consequently the derivatives of the SC functions should be obtained. If SC CNN is trained with the default backward functions, the accuracy will be very low.

Sim and Lee [26] proposed a new SC multiply-accumulate (MAC module) unit with higher accuracy and lower latency than conventional SC counterparts, but this method is limited to what is called binary interfaced SC (or hybrid SC CNN). This method cannot be used in pure SC system. Any hybrid BE-SC system (including hybrid SC CNNs) have the problem of multiple BE-SC domain conversions. Consequently, SC dataflow will be destroyed resulting much higher latencies, where latency increase will be exponential with the number of operations. In addition, hybrid approaches introduce large hardware overhead due to conversion.

Precision in conventional BE is the number of bits used to represent a number while in SC the precision is the length of the SN. The previous [23, 27, 25, 28, 24, 37, 22, 7, 29, 38, 6, 39] SC CNN implementation required precision for SN of L=64 up to 8192 bits. Long SNs increase the number of clock cycles required to perform any operation. Increased latency will affect not only the operation speed but also energy consumption as energy = power × time.

To evaluate the application level accuracy, previous works did not use the state-of-the-art CNN models, such as ResNet. Although AlexNet [12] deep CNN is used in [6], the AlexNet error rate on ImageNet is 36.7%. This error rate is high compared to other models such as ResNet [13]. ResNet-18 and ResNet-152 achieved error rates of 27.88% and 21.43% respectively.

### 1.2.2 Limitations of existing techniques of SC system design that exploit correlation

It is usual to assume that the accuracy in an SC system is dependent on the interacting SNs being highly independent or uncorrelated (in a loosely specified way). However, Alaghi and Hayes [9] has shown that, contrary to intuition, correlation can be exploited in an SC design. The circuits that exploit correlation are generally smaller and more accurate than those with uncorrelated inputs. In addition, the circuits with correlated inputs can provide a cheap implementation for a complex operation, such as the max operation, that is very hard to realize in traditional uncorrelated SC circuits.

If an SC system is to be designed using the SC circuits with correlated inputs, the correlation between SNs should be managed throughout the system; otherwise, the accuracy will be severely degraded.

Alaghi and Hayes [9] proposed a general framework for analyzing and designing combinational circuits with correlated inputs. Although such circuits can be significantly more efficient and more accurate than traditional SC circuits, the framework is limited to design of SC-based combinational circuits and cannot be used to design a complete SC system with correlated SNs. Hence, we have this idea to extend their design framework in finding a way to maintain the correlation throughout the SC system, towards realizing an SC system design with correlation that works. Maintaining the desired level of correlation between SNs is difficult [1]. Consider the problem of decorrelation, i.e., systematic elimination of undesired correlation, the counterpart problem in traditional SC. Ting and Hayes [40] have developed a theory for placing isolation-based decorrelators and have obtained conditions for a placement to be valid.

On the other hand, for SC circuits with correlated inputs, Lee et al. [41] have designed a synchronizer (also referred as a correlator) that increases the correlation between two SNs. Although not adequate by itself, this correlator is a good candidate to be considered in the proposed extended framework. Other correlation manipulation circuits would be also needed, such as a correlated SN generator that generates an SN correlated to a intermediate SN. Any SC system, although it is designed to exploit correlation, still requires uncorrelated SNs. A previous study showed that random number generators (RNGs) could take up to 80% of the SC circuit area [42]. This high area cost due to the RNGs is a serious problem that to be addressed. To solve this issue, RNG sharing schemes were proposed in [5, 43], where one RNG was used for multiple SNGs.

The above discussion suggests that the solution to the problem of designing an SC system with correlation would entail extending the framework proposed in [9] to include providing design guidelines for managing the correlation through a SC system and how to build an optimal SC system with exploiting correlation. Managing the correlation depends on two conditions, namely the correlation-sensitivity of the

SC functions at its inputs and the correlation induced at its outputs. The effect of correlation between input SNs on the behavior of the different SC operations has been studied before, where the correlation sensitivity term was introduced. However, how much different SC operations change the level of correlation at the output SNs should also be investigated, and this has not been done before. Finally, the design framework for building an SC system with correlation should have the target objective of reducing the area cost of conversion circuits.

### 1.2.3 The issue of energy efficiency achievable in conventional CNN accelerators

CNN is the dominant approach for recognition applications, but it is highly compute-intensive. In the feed-forward computation of CNN, a previous study [18] reported that convolution operations would occupy over 90% of the computation time. Optimal FPGA accelerators had been proposed for CNN [20, 32, 33, 34, 35, 36] using many hardware optimization techniques. However, the CNN implementation consumes high resource utilization and obtain a low energy efficiency smaller than 25GOPS/W. Although there are two FPGA implementation works that obtained high energy efficiency using binarized CNN [44] and FFT/Winograd CNN [45], using SC might produce greater energy efficiency. It should be noted that SC can be used in conjunction with any of the previous solutions.

HLS accelerates system design at a high-level of abstraction and speeds up the design space exploration. Typically, the SC systems are designed at the gate/RTL level, which reduces design productivity. To our knowledge, there are no previous work on design of SC modules and systems that employ designing at the C-based high-level of abstraction to facilitate the exploration of the design space.

## 1.3 Objectives

The goal of this work is to propose an accurate, low-area, and energy-efficient SC system hardware for CNN. In this thesis, this CNN is designed for recognition applications, and targeted for implementation in embedded FPGA devices. To achieve this goal, the following are the main objectives of this work:

1. To propose a framework for the design of an effective end-to-end SC system hardware composed of a series of SC processing units that exploit correlation. The proposed SC system is targeted for deployment as a hardware computation engine in applications of image processing and convolutional neural networks.
2. To develop a CNN based on the proposed stochastic computing design framework that achieves high classification accuracy in recognition applications. The associated sub-objectives are:
    i. To propose SC functions for CNN functional units that include the convolution, activation function, and pooling, which are optimized for accuracy and resource utilization.
    ii. To modify SC CNN training to achieve high classification accuracy. This includes the derivation of the backward functions for the proposed SC CNN functions.
3. To develop an SC CNN computation engine on FPGA platform using the HLS design method. This SC system hardware is used to demonstrate how the SC design led to the optimization in resource utilization and energy efficiency. The objective also involves the proposing of a novel strategy/ methodology for the design of a generic SC functional unit or system at a high-level of abstraction.

## 1.4 Scope of work

In this thesis, we utilize a combination of tools to support modeling, design, and implementation of the proposed algorithms and hardware. The approaches, software tools, performance measures, and case studies are summarized as follows:

- The entire work targets the embedded applications that have meager resources and require very low power consumption such as mobile, IoT, and wearable devices.
- The proposed SC algorithms are developed using MATLAB tool. Mean absolute error is computed to quantify errors for the SC functions.
- The CNN basic functions in this work are the inner product for the convolutional layer, average and max for pooling layer, and max for ReLU activation function layer.
- The used CNN models are modified LeNet-5 [16] and ResNet-20 [13]. The supervised training mode is used. The computation of the error gradients is based on the backpropagation algorithm.
- The effectiveness of the resulting SC CNN is demonstrated, in terms of accuracy, with different datasets that represent complex and real-world problems. The datasets used to verify and analyze the performance of the proposed SC CNN are limited to the following: (a) handwritten digit classification using the MNIST database, (b) face recognition using AT&T database, and (c) object recognition using CIFAR-10 dataset.
- The development and training of different SC CNN models are performed using deep learning toolbox in MATLAB. The MATLAB is also used any preprocessing of the datasets such as normalization and resizing.
- The performance of a CNN model using particular dataset is evaluated based on its classification accuracy and misclassification error rate.
- For the hardware implementation, only CNN inference is considered, and training remains on the software since in embedded devices training is done off-line.

- The test platform is FPGA-SoC ZYNQ Z706 development board. In this work, performance is measured by Giga Operations per Second (GOP/s), latency is measured in the number of clock cycles, energy-efficiency is measured by Giga Operations per Second per Watt GOP/s/W, and resource utilization is measured using the number of utilized BRAMs, DSPs, FFs, and LUTs in the FPGA chip.
- The SC CNN hadware is designed at the high-level of abstraction using Vivado HLS 2018.3 tool. The used description is written in C++. The functionality is validated via Vivado HLS simulation and co-simulation using a C++ testbench. The latency of the accelerator is obtained from the Vivado HLS co-simulation results. The resource utilization and power consumption are collected from the implementation report of Vivado HLx 2018.3 design suite. The Vivado power analysis tool is used to analyze the power consumption of the SC hardware.

## 1.5 Research contributions and achievements

1. A design framework for an efficient design of an effective SC system hardware that exploits correlation. The corresponding research outputs are:
    i. Characterization of correlation in SC elements and functions that include correlation-sensitivity and correlation-induction. In addition, quantifying correlation in RNG generated sequences.
    ii. Novel algorithms and designs for correlation manipulation circuitry that includes a correlator, a correlated stochastic number generator, and a RNG sharing scheme.
    iii. Design guidelines for the efficient design of an end-to-end SC system hardware optimized for accuracy, area, and energy-efficiency.
2. An accurate and low-area SC-based CNN based on novel SC functional units. The following details out this contribution:
    i. A novel SC functions that outperform all previous work [6, 7, 5] in low-area cost. In addition, those SC functions achieve higher or comparable accuracy compared to previous works. These SC functions are inner

product, SC ReLU, and maximum functions that create the convolution, activation, and max-pooling layers of a SC CNN, respectively . In addition, a new SC average pooling layer is developed based on the adder in [46].

ii. Considerations to achieve high classification accuracy after training process. These include the input normalization and the derived backward functions.

3. An efficient SC CNN convolution computation engine developed using high-level synthesis for FPGA implementation. This engine outperforms previous work [20, 47, 32, 34, 44, 35, 33, 36, 45] in terms of low resource utilization and high energy efficiency.

## 1.6 Thesis organization

This thesis is organized into 7 chapters. Chapter 2 describes the background theory of SC and CNN. It also covers the literature review of the previous related works.

Chapter 3 presents the methodology for the research work performed in this thesis. This includes the approach taken to conduct the research, the tools and platform used, and the high-level synthesis design flow for mapping algorithms towards FPGA-based accelerators.

Chapter 4 describes the proposed design framework of effective SC system hardware that exploits correlation. This chapter includes a comprehensive study of the impact of correlation on successive SC computation through the system. Based on this study, the correlation in SC functions is characterized, and correlation manipulation circuits are discussed. Guidelines are recommended to design an SC functional unit or system efficiently.

Chapter 5 presents the SC-based CNN development, where the proposed novel SC functions are discussed. Then, SC CNN training considerations to achieve high

classification accuracy are described. This chapter presents a MATLAB simulation model for SC CNN that enable creating, testing, and training different CNN models in SC domain. Finally, the proposed high-level design strategy to create a generic SC system hardware and high-level synthesis of the SC CNN hardware are presented.

Chapter 6 presents the experimental results and the analysis of SC CNN and its functions. First, the SC functions are examined. Then, the SC CNN accuracy is evaluated using different models and datasets and benchmarked with previous work. The low resource utilization and high energy-efficiency of SC CNN hardware are proved in this chapter after extensive comparisons with previous works on FPGA-based CNN accelerators. Discussions and justifications for accuracy, low-area cost, and energy efficiency are included in this chapter.

Chapter 7 summarizes the thesis, re-stating the contributions based on the results, and suggests directions for future research works.

## REFERENCES


1. A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE T COMPUT AID D*, vol. 37, pp. 1515–1531, 2017.

2. B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 149–156.

3. P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE T VLSI SYST*, vol. 22, pp. 449–462, 2014.

4. V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló, "A new stochastic computing methodology for efficient neural network implementation," *IEEE T NEUR NET LEAR*, vol. 27, pp. 551–564, 2016.

5. H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," *IEEE T EMERG TOP COM*, 2016.

6. Z. Li, J. Li, A. Ren, R. Cai, C. Ding, X. Qian, J. Draper, B. Yuan, J. Tang, Q. Qiu *et al.*, "Heif: Highly efficient stochastic computing based inference framework for deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

7. J. Yu, K. Kim, J. Lee, and K. Choi, "Accurate and efficient stochastic computing hardware for convolutional neural networks," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 105–112.

8. J. P. Hayes, "Introduction to stochastic computing and its challenges," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–3.

9. A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *IEEE International Conference on Computer Design; 6-9 October 2013; Asheville, NC, USA*. New York, NY, USA: IEEE, 2013, pp. 39–46.

10. A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*. IEEE, 2013, pp. 1–6.

11. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

12. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

13. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

14. S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

15. Y. Zhang, D. Zhao, J. Sun, G. Zou, and W. Li, "Adaptive convolutional neural network and its application in face recognition," *Neural Processing Letters*, vol. 43, no. 2, pp. 389–399, 2016.

16. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

17. C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Cascaded heterogeneous convolutional neural networks for handwritten digit recognition," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 657–660.

18. J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in *International conference on artificial neural networks*. Springer, 2014, pp. 281–290.

19. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

20. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

21. S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 243–254.

22. H. Sim, D. Nguyen, J. Lee, and K. Choi, "Scalable stochastic-computing accelerator for convolutional neural networks," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 696–701.

23. A. Ren, Z. Li, Y. Wang, Q. Qiu, and B. Yuan, "Designing reconfigurable large-scale deep learning systems using stochastic computing," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2016, pp. 1–7.

24. J. Li, Z. Yuan, Z. Li, C. Ding, A. Ren, Q. Qiu, J. Draper, and Y. Wang, "Hardware-driven nonlinear activation for stochastic computing based deep convolutional neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1230–1236.

25. A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," *ACM SIGOPS Operating Systems Review*, vol. 51, no. 2, pp. 405–418, 2017.

26. H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

27. J. Li, A. Ren, Z. Li, C. Ding, B. Yuan, Q. Qiu, and Y. Wang, "Towards acceleration of deep convolutional neural networks using stochastic computing," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 115–120.

28. Z. Li, A. Ren, J. Li, Q. Qiu, B. Yuan, J. Draper, and Y. Wang, "Structural design optimization for deep convolutional neural networks using stochastic computing," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 250–253.

29. X. Ma, Y. Zhang, G. Yuan, A. Ren, Z. Li, J. Han, J. Hu, and Y. Wang, "An area and energy efficient design of domain-wall memory-based deep convolutional neural networks using stochastic computing," in *2018 19th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2018, pp. 314–321.

30. K. Wakabayashi, "C-based behavioral synthesis and verification analysis on industrial design examples," in *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*. IEEE Press, 2004, pp. 344–348.

31. R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi *et al.*, "A survey and evaluation of fpga high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2016.

32. J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

33. C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

34. Z. Liu, Y. Dou, J. Jiang, J. Xu, S. Li, Y. Zhou, and Y. Xu, "Throughput-optimized fpga accelerator for deep convolutional neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 10, no. 3, p. 17, 2017.

35. K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga,"

*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 35–47, 2018.

36. S. O. Ayat, M. Khalil-Hani, and A. A.-H. Ab Rahman, "Optimizing fpga-based cnn accelerator for energy efficiency with an extended roofline model," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 26, no. 2, pp. 919–935, 2018.

37. V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 13–18.

38. Z. Li, J. Li, A. Ren, C. Ding, J. Draper, Q. Qiu, B. Yuan, and Y. Wang, "Towards budget-driven hardware optimization for deep convolutional neural networks using stochastic computing," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 28–33.

39. H. Sim and J. Lee, "Cost-effective stochastic mac circuits for deep neural networks," *Neural Networks*, 2019.

40. P.-S. Ting and J. P. Hayes, "Isolation-based decorrelation of stochastic circuits," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*. IEEE, 2016, pp. 88–95.

41. V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1417–1422.

42. W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 93–105, 2011.

43. B. Yuan, Y. Wang, and Z. Wang, "Area-efficient scaling-free dft/fft design using stochastic computing," *IEEE T CIRCUITS-II*, vol. 63, pp. 1131–1135, 2016.

44. Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 65–74.

45. Y. Liang, L. Lu, Q. Xiao, and S. Yan, "Evaluating fast algorithms for convolutional neural networks on fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

46. V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 13–18.

47. M. Alawad and M. Lin, "Stochastic-based deep convolutional networks with reconfigurable logic fabric," *IEEE Transactions on multi-scale computing systems*, vol. 2, no. 4, pp. 242–256, 2016.

48. A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM T EMBED COMPUT S*, vol. 12, pp. 1–19, 2013.

49. D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to probability*. Athena Scientific Belmont, MA, 2002, vol. 1.

50. A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 76.

51. S. Liu and J. Han, "Energy efficient stochastic computing with sobol sequences," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 650–653.

52. V. T. Lee, A. Alaghi, R. Pamula, V. S. Sathe, L. Ceze, and M. Oskin, "Architecture considerations for stochastic computing accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2277–2289, 2018.

53. V. Sehwag, N. Prasad, and I. Chakrabarti, "A parallel stochastic number generator with bit permutation networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 2, pp. 231–235, 2017.

54. B. D. Brown and H. C. Card, "Stochastic neural computation. i. computational elements," *IEEE Transactions on computers*, vol. 50, no. 9, pp. 891–905, 2001.

55. Z. Li, A. Ren, J. Li, Q. Qiu, Y. Wang, and B. Yuan, "Dscnn: Hardware-oriented optimization for stochastic computing based deep convolutional

neural networks," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*. IEEE, 2016, pp. 678–681.

56. M. H. Najafi and M. E. Salehi, "A fast fault-tolerant architecture for sauvola local image thresholding algorithm using stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 2, pp. 808–812, 2016.

57. E. Vahapoglu and M. Altun, "Accurate synthesis of arithmetic operations with stochastic logic," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 415–420.

58. ——, "From stochastic to bit stream computing: Accurate implementation of arithmetic circuits and applications in neural networks," *arXiv preprint arXiv:1805.06262*, 2018.

59. Y.-N. Chang and K. K. Parhi, "Architectures for digital filters using stochastic computing," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 2697–2701.

60. H. Ichihara, S. Ishii, D. Sunamori, T. Iwagaki, and T. Inoue, "Compact and accurate stochastic circuits with shared random number sources," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*. IEEE, 2014, pp. 361–366.

61. M. M. Wong, M. D. Wong, C. Zhang, and I. Hijazin, "A new stochastic inner product core design for digital fir filters," in *MATEC Web of Conferences*, vol. 125. EDP Sciences, 2017, p. 05006.

62. Y. Liu and K. K. Parhi, "Linear-phase lattice fir digital filter architectures using stochastic logic," *Journal of Signal Processing Systems*, vol. 90, no. 5, pp. 791–803, 2018.

63. ——, "Architectures for recursive digital filters using stochastic computing," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3705–3718, 2016.

64. B. Parhami and C.-H. Yeh, "Accumulative parallel counters," in *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, vol. 2. IEEE, 1995, pp. 966–970.

65. P.-S. Ting and J. P. Hayes, "Stochastic logic realization of matrix operations," in *2014 17th Euromicro Conference on Digital System Design*. IEEE, 2014, pp. 356–364.

66. K. Kim, J. Lee, and K. Choi, "Approximate de-randomizer for stochastic circuits," in *2015 International SoC Design Conference (ISOCC)*. IEEE, 2015, pp. 123–124.

67. K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

68. A. Zhakatayev, S. Lee, H. Sim, and J. Lee, "Sign-magnitude sc: getting 10x accuracy for free in stochastic computing for deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

69. B. Yuan and Y. Wang, "High-accuracy fir filter design using stochastic computing," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 128–133.

70. A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 76.

71. M. H. Najafi and D. J. Lilja, "High-speed stochastic circuits using synchronous analog pulses," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 481–487.

72. P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*. IEEE, 2011, pp. 154–161.

73. Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

74. K. Kim, J. Lee, and K. Choi, "An energy-efficient random number generator for stochastic circuits," in *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 2016, pp. 256–261.

75. T.-H. Chen, P. Ting, and J. P. Hayes, “Achieving progressive precision in stochastic computing,” in *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on*. IEEE, 2017, pp. 1320–1324.

76. A. Alaghi and J. P. Hayes, “On the functions realized by stochastic computing circuits,” in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*. ACM, 2015, pp. 331–336.

77. R. K. Budhwani, R. Ragavan, and O. Sentieys, “Taking advantage of correlation in stochastic computing,” in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

78. D. Jenson and M. Riedel, “A deterministic approach to stochastic computation,” in *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 2016, p. 102.

79. M. H. Najafi, S. Jamali-Zavareh, D. J. Lilja, M. D. Riedel, K. Bazargan, and R. Harjani, “Time-encoded values for highly efficient stochastic circuits,” *IEEE T VLSI SYST*, vol. 25, pp. 1644–1657, 2017.

80. M. H. Najafi and D. Lilja, “High quality down-sampling for deterministic approaches to stochastic computing,” *IEEE T EMERG TOP COM*, 2018.

81. P. Ting and J. P. Hayes, “Eliminating a hidden error source in stochastic circuits,” in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–6.

82. X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

83. Y. LeCun, C. Cortes, and C. J. Burges. The mnist database. [Online]. Available: http://yann.lecun.com/exdb/mnist/

84. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

85. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

86. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

87. E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra *et al.*, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2017, pp. 5–14.

88. M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, "A massively parallel coprocessor for convolutional neural networks," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*. IEEE, 2009, pp. 53–60.

89. C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "Cnp: An fpga-based processor for convolutional networks," in *2009 International Conference on Field Programmable Logic and Applications*. IEEE, 2009, pp. 32–37.

90. S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 247–257, 2010.

91. C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems." in *ISCAS*, vol. 2010, 2010, pp. 257–260.

92. V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 682–687.

93. M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 13–19.

94. A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2018.

95. S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for floating-point programs and multicore architectures," Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), Tech. Rep., 2009.

96. B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-d convolvers for fast digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 3, pp. 299–308, 1999.

97. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

98. M. A. Awan and S. M. Petters, "Race-to-halt energy saving strategies," *Journal of Systems Architecture*, vol. 60, no. 10, pp. 796–815, 2014.

99. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

100. I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

101. M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

102. M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

103. R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, 2017.

104. M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv preprint arXiv:1312.5851*, 2013.

105. A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4013–4021.

106. M. Alawad and M. Lin, "Survey of stochastic-based computation paradigms," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 98–114, 2016.

107. S. R. Faraji, M. H. Najafi, B. Li, D. J. Lilja, and K. Bazargan, "Energy-efficient convolutional neural networks with deterministic bit-stream processing," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1757–1762.

108. C. Wolf, J. Glaser, and J. Kepler, "Yosys-a free verilog synthesis suite," in *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, 2013.

109. MATLAB, "Deep learning toolbox," https://www.mathworks.com/products/deep-learning.html.

110. L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. UK: Strathclyde Academic Media, 2014.

111. Xilinx, "Vivado design suite - hlx editions," https://www.xilinx.com/products/design-tools/vivado.html.

112. ——, "Vivado design suite user guide high-level synthesis," 2018.

113. ——, "Vivado design suite user guide designing ip subsystems using ip integrator," 2019.

114. ——, "Vivado design suite user guide power analysis and optimization," 2019.

115. Berkeley Logic Synthesis and Verification Group. Abc: A system for sequential synthesis and verification. [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/abc/

116. The nangate open cell library. [Online]. Available: https://projects.si2.org/openeda.si2.org/projects/nangatelib

117. "7 series fpgas data sheet: Overview," https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

118. “Zynq-7000 soc data sheet: Overview,” https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

119. T.-H. Chen and J. P. Hayes, “Design of division circuits for stochastic computing,” in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 116–121.

120. Ica99 synthetic benchmarks. [Online]. Available: http://sound.media.mit.edu/ica-bench/

121. B. Yuan, C. Zhang, and Z. Wang, “Design space exploration for hardware-efficient stochastic computing: A case study on discrete cosine transformation,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6555–6559.

122. P. Jamieson, W. Luk, S. J. Wilton, and G. A. Constantinides, “An energy and power consumption analysis of fpga routing architectures,” in *2009 International Conference on Field-Programmable Technology*. IEEE, 2009, pp. 324–327.

123. A. Alaghi and J. P. Hayes, “Strauss: Spectral transform use in stochastic circuit synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1770–1783, 2015.

## LIST OF PUBLICATIONS

This appendix lists downs the papers written based on the findings from the work done in this thesis. It also includes publications that are related to the work done in this thesis. The following is a summary of these papers:

1. Abdellatef, H., Khalil-Hani, M., Shaikh-Husin, N. and Ayat, S. O. Stochastic Computing Correlation Utilization in Convolutional Neural Network Basic Functions. *Telkomnika*, 2018. 16(6). (Scopus).
2. Abdellatef, H., HANI, M. K. and Shaikh-Husin, N. Accurate and compact stochastic computations by exploiting correlation. *Turkish Journal of Electrical Engineering & Computer Sciences*, 2019. 27(1): 547–564. (ISI, IF 0.708).