



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

A Novel Approximate Synthesis Flow for  
Convolutional Neural Networks

Jaewoo Kim

Department of Electrical Engineering

Graduate School of UNIST

2018

# A Novel Approximate Synthesis Flow for Convolutional Neural Networks

Jaewoo Kim

Department of Electrical Engineering

Graduate School of UNIST

# A Novel Approximate Synthesis Flow for Convolutional Neural Networks

A thesis  
submitted to the Graduate School of UNIST  
in partial fulfillment of the  
requirements for the degree of  
Master of Science

Jaewoo Kim

06 / 14 / 2018

Approved by



---

Advisor

Seong-Jin Kim


# A Novel Approximate Synthesis Flow for Convolutional Neural Networks

Jaewoo Kim

This certifies that the thesis of Jaewoo Kim is approved.

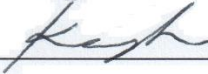
06 / 14 / 2018

signature



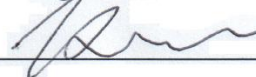
Advisor: Seong-Jin Kim

signature



Seokhyeong Kang: Thesis Committee Member #1

signature



Kyung Rok Kim: Thesis Committee Member #2

## Abstract

The portability of emerging computing systems demands further reduction in the power consumption of their components. Approximate computing can reduce power consumption by using a simplified or an inaccurate circuit.

In this paper, we exploit the approximate computing to improve the energy efficiency of a finite impulse response (FIR) filter. We propose an approximate synthesis technique for an energy-efficient FIR filter with an acceptable level of accuracy. We employ the common subexpression elimination (CSE) algorithm to implement the FIR filter and replace conventional adder/subtractors with approximate ones. While yielding an acceptable rate of accuracy, the proposed flow can attain a maximum energy saving of 50.7% in comparison with conventional FIR filter designs.

Further, we propose the approximate synthesis flow for the Convolutional Neural Networks (CNN) to exploit the error resiliency of neural networks. Recently CNN is showing an outstanding performance in the field of image recognition and its application is expected to be widely expanded. However, the intensive computational requirement limits the practical use of CNN hardware.

Proposed approximate synthesis flow is applied to Multiply-and-Accumulate (MAC) operations of CNN to improve energy efficiency. Our proposed flow can find an energy-efficient approximate MAC module with acceptable error rate. Energy consumption of the MAC modules for convolution of  $3 \times 3$  and  $5 \times 5$  matrices are improved 46.4 % and 43.4 % while output quality degradation of the handwritten digit recognition was negligibly low.

## Acknowledgements

It's been three years since I joined SoC Design Laboratory. During that time, I was able to grow significantly both academically and personally. In particular, I would like to express my deep gratitude to Professor Seokhyeong Kang, my advisor. Thanks to his encouragement and guidance, I was able to finish master's course. He is one of the most personally good person I have ever met, and he is a knowledgeable and insightful researcher. I think it is a great luck in my life to meet Professor Kang.

I would also like to thank my fellow members of SoCDL. Thanks to the help and cooperation of Yesung Kang, I was able to successfully carry out the projects I undertook. I also thank Sunmean Kim, Taeho Lim, and SangGi Do, who worked on the projects together, and Seungwon Kim, Mingyu Woo, Daeyeon Kim, and Juyeon Choi for making a good atmosphere so that I can concentrate on research.

I also really appreciate Prof. Seong-jin Kim and Prof. KyungRok Kim, the committee members of master thesis review, giving me the advice and encouragement.

Finally, I would like to thank my family who have always understood and supported my choices. I give them all my glory.

The materials in this thesis is based on the following publications

From Chapter I to Chapter IV is based on:

- Yesung Kang, **Jaewoo Kim** and Seokhyeong Kang, “A Novel Approximate Synthesis Flow for the Energy-Efficient FIR filter”, Proc. IEEE International Conference on Computer Design, 2016, pp. 96-102.

## VITA

1991            Born, Daegu, South Korea

2016            B.S., Electrical and Computer Engineering,  
                  Ulsan National Institute of Science and Technology, Ulsan, South Korea

- Yesung Kang, **Jaewoo Kim**, and Seokhyeong Kang, “A Novel Approximate Synthesis Flow for the Energy-Efficient FIR filter”, *Proc. IEEE International Conference on Computer Design*, 2016.
- Yesung Kang, **Jaewoo Kim**, Sunmean Kim, Sunhae Shin, E-san Jang, Jae Won Jeong, Kyung Rok Kim, and Seokhyeong Kang, “A Novel Ternary Multiplier based on Ternary CMOS Compact Model”, *Proc. IEEE International Symposium on Multiple-Valued Logic*, 2017.



## Contents

I. Introduction .....	1
II. Related Works .....	4
2.1 Common Subexpression Elimination (CSE) .....	4
2.2 Approximate Computing .....	5
III. Approximate Synthesis flow for FIR filter .....	7
3.1 The Proposed Approximate Adder/Subtractor .....	7
3.2 Approximate Synthesis Flow .....	9
IV. Experimental Setup and Results .....	13
4.1 Experimental Setup .....	13
4.2 FIR Filter Implementation .....	13
4.3 Image Filter Experiment .....	17
V. Approximate Synthesis Flow for Convolutional Neural Networks .....	19
5.1 Convolutional Neural Network (CNN) .....	19
5.2 Related Work .....	21
5.3 Approximate Synthesis Flow for MAC module .....	21
5.4 Experimental Setup and Results .....	22
5.4.1 Experimental Setup .....	22
5.4.2 Power Consumption .....	23
5.4.3 Classification Accuracy .....	25
VI. Conclusion .....	27

## List of Figures

**Figure 1.1** Structure of the conventional FIR filter and the proposed approximate FIR filter.

**Figure 2.1** Schematic of the FIR filter. The coefficients of the FIR filter are (105, 831, 621, 815), and  $FAS = 3$ .

**Figure 3.1** Proposed Addition Arithmetic in [6]

**Figure 3.2** Proposed approximate adder/subtractor.

**Figure 3.3** Structure of the approximate part.

**Figure 3.4** Schematic of the k-th carry generator and the sum generator in the approximate part.

**Figure 3.5** Proposed synthesis flow.

**Figure 4.1** Accuracy vs. delay domain of the proposed synthesis flow and the exhaustive research of FIR Filter

**Figure 4.2** Accuracy vs. power domain of the proposed synthesis flow and the exhaustive research of FIR Filter

**Figure 4.3** Accuracy vs. Energy domain of the proposed synthesis flow and the exhaustive research of FIR Filter

**Figure 4.4** Filtered images using optimized FIR filters.

**Figure 5.1** Architecture of convolutional neural network [26]

**Figure 5.2** Computation of convolution layer [26]

**Figure 5.3** Working principle of an 8-bit Multiplier-less Artificial Neuron [27]

**Figure 5.4** Structure of the MAC module for the convolution of  $3 \times 3$  filter

**Figure 5.5** MNIST handwritten numbers database

**Figure 5.6** Accuracy vs. delay domain of the proposed synthesis flow and the exhaustive research of MAC modules for (a)  $3 \times 3$  filter and (b)  $5 \times 5$  filter

**Figure 5.7** Accuracy vs. power domain of the proposed synthesis flow and the exhaustive research of MAC modules for (a)  $3 \times 3$  filter and (b)  $5 \times 5$  filter

**Figure 5.8** Accuracy vs. Energy domain of the proposed synthesis flow and the exhaustive research of MAC modules for (a)  $3 \times 3$  filter and (b)  $5 \times 5$  filter

## List of Tables

**Table 4.1** Approximation results in 4-tap FIR filter with FAS = 3.

**Table 4.2** Approximation results in 25-tap filter with FAS = 4.

**Table 4.3** Specifications of the experimented FIR filters.

**Table 4.4** Experimental results of FIR filters with the approximate synthesis flow

**Table 5.1** Shape parameters of convolution layer

**Table 5.2** Network parameter of CNN for MNIST

**Table 5.3** Approximation results of MAC module

**Table 5.4** Classification accuracy of baseline and approximate CNN with  $3 \times 3$  kernel

## **List of Algorithms**

**Algorithm 1.** Sensitivity-based approximate synthesis flow

## Nomenclature

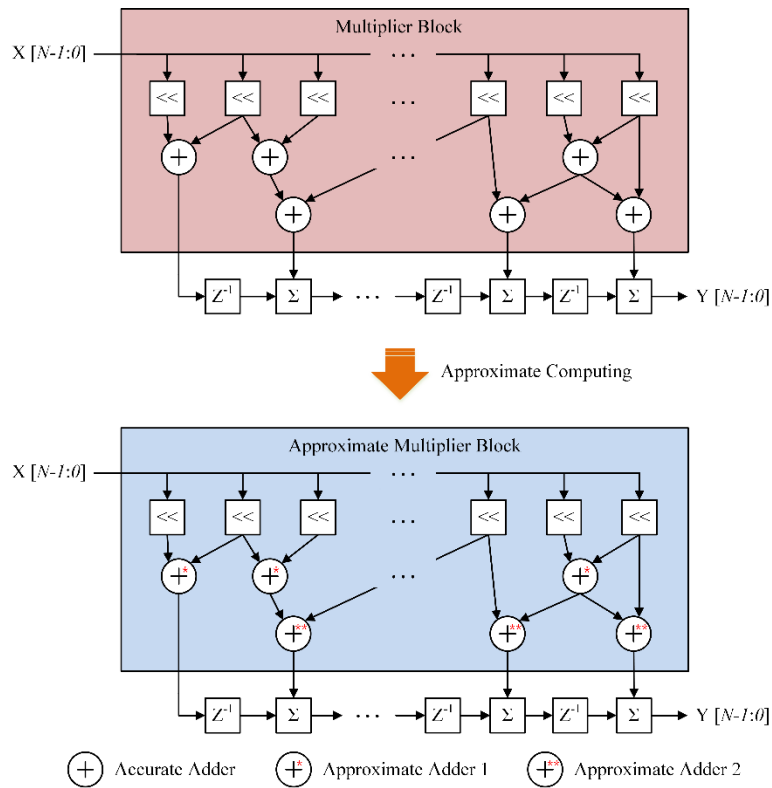
<b>IoT</b>	Internet of Things
<b>IIR</b>	Infinite Impulse Response
<b>FIR</b>	Finite Impulse Response
<b>MAC</b>	Multiply-and-Accumulate
<b>SPT</b>	Signed-Power-of-Two
<b>CSD</b>	Canonical Signed Digit
<b>CSE</b>	Common Subexpression Elimination
<b>AS</b>	Adder Step
<b>FAS</b>	Filter Adder Step
<b>MSB</b>	Most Significant Bit
<b>LSB</b>	Least Significant Bit
<b>TCL</b>	Tool Command Language
<b>RTL</b>	Register Transfer Level
<b>EDA</b>	Electronic Design Automation
<b>PSNR</b>	Peak Signal-to-Noise Ratio
<b>CNN</b>	Convolutional Neural Network

# Chapter I

## Introduction

As semiconductor technologies continue to develop, electronic devices are becoming smaller and more portable. Consequently, as the battery size of Internet of Things (IoT) devices decreases and power consumption increases, the urgent need for energy-efficient systems has generated research interests in approximate computing techniques. Approximate computing can be applied to vision, search, and image processing, which do not require a 100% of accurate results.

In this paper, we apply approximate computing to a digital filter for image processing. The digital filter can be implemented through an infinite impulse response (IIR) filter and a finite impulse response (FIR) filter. The FIR filter shows better phase linearity and stability than the IIR filter. However, it consumes more power because of its complex design, and hence reduces the overall energy efficiency of the system. To improve the energy efficiency of the FIR filter, several proposals have sought to reduce their design complexity [1–5]. However, these approaches only focused on reducing the number of adder steps [1–3], providing an accuracy estimation model [4], or developing an approximate adder [5], separately.



**Figure 1.1: Structure of the conventional FIR filter and the proposed approximate FIR filter**

Figure 1.1 shows the conventional multiply-and-accumulate (MAC) structure of the FIR filter. A popular idea for complexity reduction here is a multiplier-less FIR filter [1], where multiplication is implemented with shifters and adders rather than multipliers. Integer coefficients are transformed into a proper one for shift and addition operations. In conventional FIR filters, all coefficients are expressed in signed-power-of-two (SPT) space rather than signed binary, since SPT can reduce the number of nonzero digits. In the SPT codes, a canonical signed digit (CSD) code is well known to effectively reduce the complexity of FIR filters.

Another key idea in conventional FIR filters is a common subexpression elimination (CSE) algorithm. Chia et al. [2] proposed a CSE algorithm to reduce redundancy among CSD coefficients. Choi et al. [3] analyzed the criticality of each coefficient of a FIR filter and applied tighter constraints on more critical coefficients during the CSE algorithm. Choi's FIR filter yielded 25%-30% power saving at low voltages with minor passband/stopband ripples. Kahng et al. [4] implemented a FIR filter using an approximation at the synthesis level. They replaced certain modules with approximated ones based on lookup tables in order to reduce power consumption with only a small degradation in the quality of output. Gupta et al. [5] implemented a FIR filter using an approximated circuit. They proposed mathematical models for error and the power consumption of the approximate adders.

Chia et al. [2] and Malcolm et al. [1] only focused on reducing the number of adder steps. Choi et al. [3] considered voltage scaling to save power, but the errors incurred along the critical path were observed to usually be more critical than those due to approximations. Kahng et al. [4] and Gupta et al. [5] applied approximate computing to a FIR filter but did not provide any automated synthesis flow for the approximation. If the size of the design of the FIR filter becomes larger, it becomes difficult to find optimum configurations for the approximate adders.

In this paper, we propose a novel approximate synthesis technique that reduces energy consumption by replacing conventional adders/subtractors in the FIR filter with approximated adders/subtractors with automated synthesis flow, as shown in Figure 1.1. The following are the main contributions of our paper:

- An accuracy-configurable adder/subtractor is proposed, which is energy efficient and has relatively high accuracy.
- The maximum error due to the configurations of the proposed adder/subtractor is analyzed to estimate output quality.
- A novel approximate synthesis flow for the FIR filter is proposed. Using the proposed approximate synthesis flow, we can save energy/power consumption and improve performance to yield a reasonable level of accuracy.



The rest of the thesis is organized as follows. In Chapter II, the CSE algorithm and previous approximate computing techniques are briefly introduced. In Chapter III, our proposed approximate adder/subtractor is introduced and its accuracy is analyzed. The proposed approximate synthesis flow is also described here. Chapter IV contains a description of the experimental setup and a discussion of the results. In Chapter V, approximate synthesis flow for the convolutional neural networks is described, and we offer a summary of our work in Chapter VI.

## Chapter II

### Related Work

#### 2.1 Common Subexpression Elimination (CSE)

As discussed in Chapter I, the CSE algorithm can reduce the design complexity of the FIR filter. In this Chapter, we briefly introduce the CSE algorithm proposed in [2]. The following terms are used to explain CSE algorithm.

- Adder Step (*AS*) : the number of adders that are used to implement the coefficients of the FIR filter.
- Filter Adder Step (*FAS*) : the number of adders along the critical path of the FIR filter. *FAS* is always greater than or equal to  $\max(\log_2 k)$  where  $k$  is the number of non-zero bits of the coefficients.

At the beginning of the CSE algorithm, all coefficients are converted into canonical signed-digit codes and their consecutive zeros are eliminated using a right-shift operation. Set  $C_N$  is constructed from the converted coefficients, and another set  $N_C$  is constructed by decomposing  $C_N$ . At the first iteration of the CSE algorithm, each value in  $C_N$  is checked to determine if it is decomposable by the other values in  $C_N \cup \{1\}$ . If the value is decomposable, it moves into a set  $C_P$ . Otherwise, the algorithm checks if the value is decomposable using values in  $C_N \cup N_C \cup \{1\}$ , and the decomposed value moves to  $C_P$ . The values in  $N_C$ , which are used in the decomposition, are moved to  $C_N$ . These procedures are repeated until  $C_N$  is empty. Following the CSE algorithm, the CSD values in  $C_P$  are used to synthesize the multiplier block in Figure 1.1.

For further explanation, we use an example. Let  $FAS = 4$ ; the coefficients are

$$h_0 = 105_{(10)} = 10\bar{1}01001_{(2)} \quad h_1 = 831_{(10)} = 10\bar{1}0100000\bar{1}_{(2)}$$

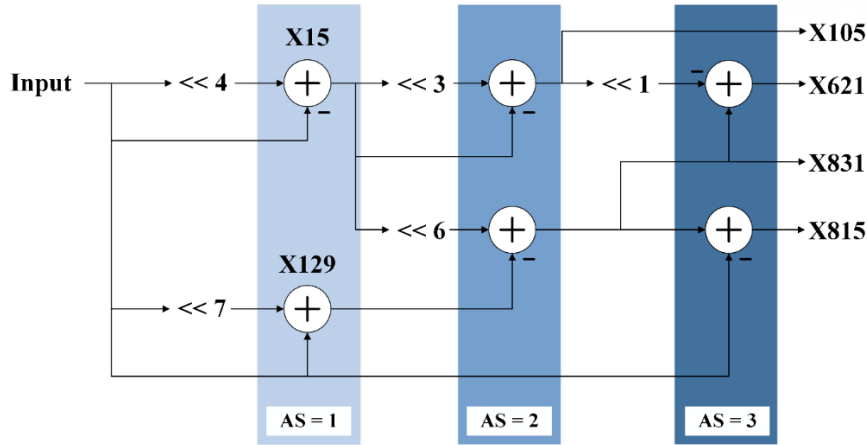
$$h_2 = 621_{(10)} = 10100\bar{1}0\bar{1}01_{(2)} \quad h_3 = 815_{(10)} = 10\bar{1}010\bar{1}0001_{(2)}$$

For simplicity, the CSD coefficients are expressed in integer format. Prior to the first iteration,

$$C_P = \varnothing$$

$$C_N = \{105, 831, 621, 815\}$$

$$N_C = \{3, 5, 7, 9, 13, 15, 17, 19, 23, 27, 31, 39, 47, 51, 63, 67, 97, 109, 113, 123, 125, 127, 129, 137, 155, 159, 193, 209, 257, 273, 493, 497, 509, 513, 625, 637, 641, 751, 767, 1007, 1023, 1071, 1087\}$$



**Figure 2.1: Schematic of the FIR filter. The coefficients of the FIR filter are (105, 831, 621, 815), and  $FAS = 3$ .**

At the first iteration, 815 and 621 are decomposed by 831 and 105, respectively:  $815 = 831 - 1 \times 24$ ,  $621 = 831 - 105 \times 21$ . At the next step, 105 and 831 are decomposed. The result of the decomposition is  $105 = 15 \times 23 - 15$  and  $831 = 15 \times 26 - 129$ , respectively. At the last step, 15 and 129 are decomposed:  $15 = 1 \times 24 - 1$  and  $129 = 1 \times 27 + 1$ . Following the iteration,  $C_P = \{105, 831, 621, 815, 15, 129\}$

$$C_N = \varnothing$$

$$N_C = \{3, 5, 7, 9, 13, 17, 19, 23, 27, 31, 39, 47, 51, 63, 67, 97, 109, 113, 123, 125, 127, 137, 155, 159, 193, 209, 257, 273, 493, 497, 509, 513, 625, 637, 641, 751, 767, 1007, 1023, 1071, 1087\}$$

The iterations terminate when  $C_N$  is empty. The synthesized FIR filter from the CSE algorithm is shown in Figure 2.1.

## 2.2 Approximate Computing

Approximate computing generates sufficiently good results with low power rather than exact results. It can be used for noise-tolerant applications. Various approximate arithmetic designs have been proposed in past research. Lu et al. [7] introduced a fast adder with shorter carry chains that considers only the previous  $k$  bits of input in computing a carry bit. Verma et al. [8] proposed a variable-latency speculative adder (VLSA), which is a reliable version of the Lu adder [7] with error detection and correction. Shin et al. [9] also proposed a data path redesign technique for various adders that reduces the lengths of critical paths in the carry chain. Zhu et al. [6] proposed three approximate adders—ETAI, ETAIL, and ETAILM. ETAI is divided into an accurate part and an inaccurate part to achieve approximate

results. ETAIL reduces carry propagation to speed up the adder, and ETAILM modifies ETAIL by connecting carry chains in accurate MSB parts. Gupta et al. [5] conducted approximations at the transistor level, and proposed approximate full adder cells to design multi-bit adders for video applications to save power and area. Kahng et al. [10] proposed an accuracy-configurable approximate (ACA) adder. In an approximate mode, it carries out approximations by cutting carry chains. In an accurate mode, it recovers accuracy by error detection and correction circuits. The ACA adder can save power consumption in the approximate mode and provide precise results in the accurate mode. Venkatesan et al. [11] proposed a systemic design methodology for approximation computing that eliminates certain nodes from the original set of nodes, and analyzes how the eliminated nodes affect accuracy and power consumption through approximation.

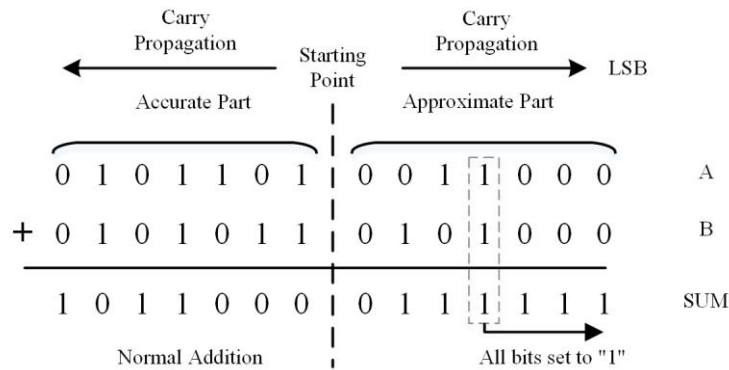
Several studies have been devoted to approximate multipliers [12–17]. For DSP applications, fixed-width approximate multipliers have been proposed in [13–15]. They eliminate  $(W-1)$  LSBs of  $(2W - 1)$  partial products obtained from a  $W \times W$  multiplication. Cho et al. [13] and Wand et al. [15] proposed carry approximation techniques in multiplication. Lu et al. [16] proposed a broken-booth multiplier, but this has a low probability of yielding the correct result rate. Kulkarni et al. [12] introduced an approximate multiplier based on  $2 \times 2$  approximate multiplication with an error probability of  $1/16$ . The simplified  $2 \times 2$  approximate multiplier only has five unit cells, whereas the accurate one has eight unit cells. Not only does the simplification reduce the lengths of the critical paths of approximate multipliers, it also consumes less power and outperforms accurate multipliers.

## Chapter III

### APPROXIMATE SYNTHESIS FLOW FOR FIR FILTER

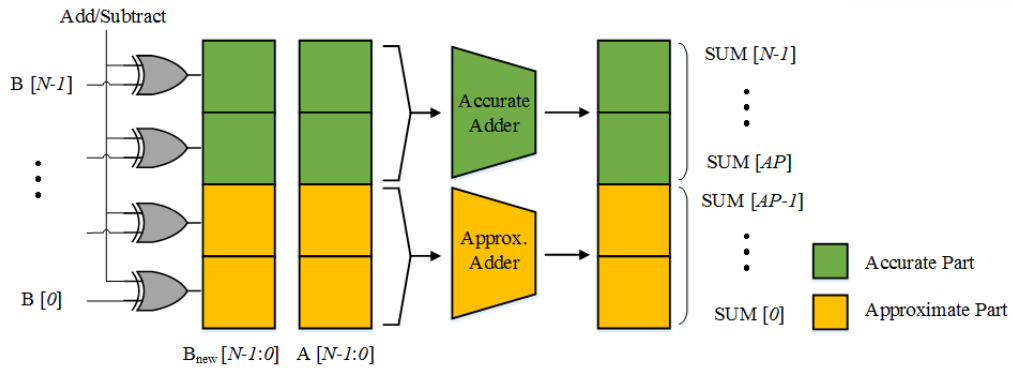
#### 3.1 The Proposed Approximate Adder/Subtractor

For the approximation of the FIR filter, we propose an accuracy configurable adder/subtractor. The basic principle of the proposed adder/subtractor is similar to that underlying Zhu's adder [6] shown in Figure 3.1. This adder detects carry generation conditions and generates "1" in all lower-sum bits without carry propagation to upper bits. To implement MAC circuits, both adders and subtractors are required. XOR gates are added in front of the adder to switch between it and the subtractor. For exact subtract operations, we should take 2's complement of the subtrahend by adding "1" to the 1's complement. The proposed approximate adder/subtractor, however, takes the 1's complement of the subtrahend as input because a carry in the approximate part is not propagated to the accurate part.

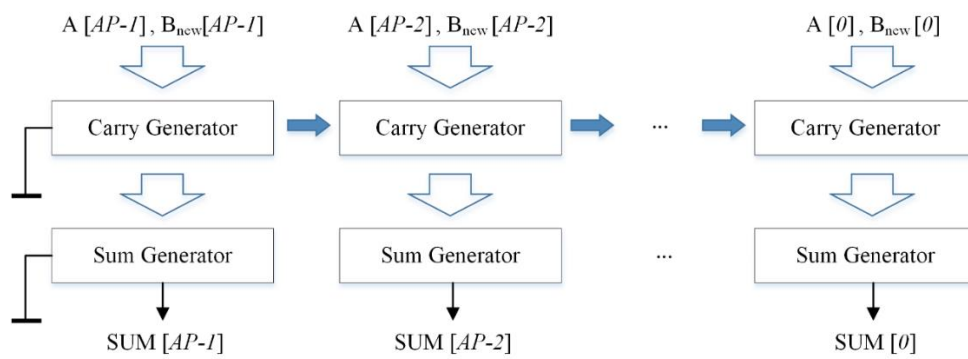


**Figure 3.1: Addition Arithmetic proposed in [6]**

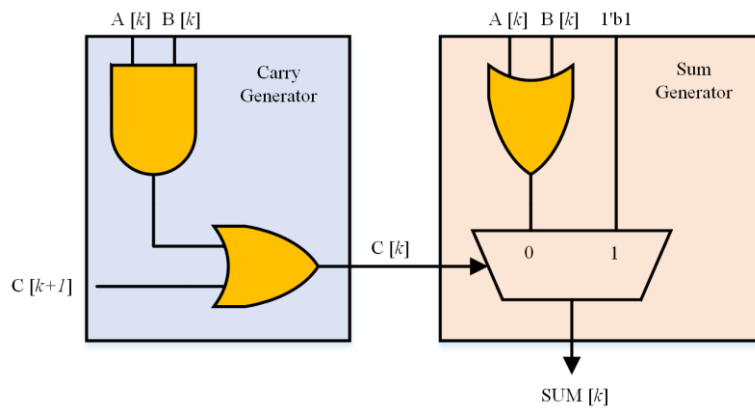
Our proposed adder is divided into two parts: an accurate part and an approximate part, as shown in Figure 3.2. The bit width of the adder is  $N$  and that of the approximate part is  $AP$ . The operating principle of the accurate part is identical to that of conventional adders. The structure of the approximate part is shown in Figure 3.3. It consists of  $AP$ -bit carry generators and  $AP$ -bit sum generators. As shown in Figure 3.3, the carry in the approximate parts is propagated from the most significant bit (MSB) of the approximate part to the least significant bit (LSB). The direction of carry propagation is the reverse of that in conventional adders.



**Figure 3.2: Proposed approximate adder/subtractor.**



**Figure 3.3: Structure of the approximate part.**



**Figure 3.4: Schematic of the  $k$ -th carry generator and the sum generator in the approximate part.**

Figure 3.4 shows a schematic diagram of the carry generator and the sum generator. If the carry is generated from previous carry generators, it passes to the next one. Otherwise, two input operands are compared, and the carry is generated if both are "1." The sum generator receives a carry from the carry generator. If a carry exists, the sum generator returns "1." Otherwise, it adds two input operands and

returns the sum value. The accuracy of the adder/subtractor is configurable by changing parameter  $AP$ , the bit width of the approximate part.  $AP$  can be configured from 0 to  $N$ . If  $AP$  is 0, the result of the proposed adder/subtractor is identical to that of the conventional adder/subtractor. If  $AP$  increases, the accuracy of the output is degraded while power consumption is reduced or performance is improved.

However, if  $AP$  is larger than a certain value, the propagation delay of the approximate part becomes that of the accurate part, and the benefits of further approximation are diminished. Hence, the  $AP$  should be appropriately configured during approximate synthesis flow.

The maximum error in approximation occurs when all input bits in the approximate part are "1." In this case, the two input operands are  $2^{AP} - 1$ . The outputs from the conventional adders are  $(2^{AP} - 1) \times 2$ , whereas the approximate adder returns  $2^{AP} - 1$ . In the results, the maximum error that can occur in the approximate adder is  $2^{AP} - 1$ . On the contrary, if the approximate part is truncated, the maximum error is  $(2^{AP} - 1) \times 2$ , which is twice that incurred by the proposed adder. For example, if  $N$ ,  $AP$ , and the two inputs are 8, 4,  $01101111_{(2)}$ , and  $00011111_{(2)}$ , respectively, four MSBs are computed in the conventional part and four LSBs are added in the approximate part. The outputs from the accurate and approximate parts are  $0111_{(2)} \times 2^4$  and  $1111_{(2)}$ , respectively, and the result is  $01111111_{(2)}$ ,  $127_{(10)}$ . Since the golden result of this addition is  $10001110_{(2)}$ ,  $142_{(10)}$ , the error is 15, which is equal to  $2^4 - 1$ . From the results, the amount of error can be reduced by using approximate adders when it compares to the truncation of some input bits.

To verify the quality of the output obtained by approximate computing, we use the accuracy metric proposed in [6], defined as follows:

$$accuracy = \min_{k=1, \dots, M} \left( 1 - \frac{|result_k - ref_k|}{|ref_k|} \right) \times 100\% \quad (1)$$

where  $M$  is the number of input patterns. The  $result_k$  is an approximate result generated from the  $k$ -th input pattern, and  $ref_k$  is the correct result.

### 3.2 Approximate Synthesis Flow

In this section, we describe the proposed approximate synthesis flow. The purpose of the synthesis flow is to find the optimum  $AP$  configurations of approximate adders. Using these optimum configurations, we can save energy/power consumption and improve performance while maintaining a higher accuracy than a certain minimum constraint,  $accuracy_{min}$ . However, finding the optimally configured  $AP$ s of the adders is difficult because the number of possible combinations of configurations is proportional to  $N^{M_{adder}}$ , where  $M_{adder}$  is the number of adders and  $N$  is the bit width of the adders. For

further explanation, we use the example in Figure 2.1. The bit width of the input, the coefficients, and the output in the example are 15, 12, and 28 bits, respectively. The coefficients are (105, 831, 621, 815), synthesized from the CSE algorithm introduced in Section 2.1, with  $FAS = 3$ . If the  $M_{adder}$  is 6 and  $N$  is 28 bits, the number of possible combinations of the  $APs$  is approximately  $4.82 \times 10^8$ . Since the size of the design of the example is small and the number of adders is conventionally greater than six, there are too many possible combinations of  $AP$  configurations in conventional FIR filters to analyze in this example. Searching all combinations is time and resource consuming, and is nearly impossible in cases of larger designs.

To handle this problem, we make two assumptions. First, the delays in the adders are comparable to those in the subtractors. Second, the actual arrival time of an adder/subtractor is comparable to that of another adder/subtractor with the same  $AS$ . Hence, we can conclude that changing  $APs$  in only one path is less effective than simultaneously changing the  $APs$  of adders. The number of possible combinations is then proportional to  $N^{FAS}$ . Considering that the  $FAS$  of the FIR filter is much smaller than that of  $M_{adder}$ , we can significantly reduce design space. Assuming  $FAS$  is 3 and  $N$  is 28 bits, the number of possible combinations of  $APs$  is 21,952. During approximate synthesis flow,  $AP$  is usually less than the half  $N$ , where the practical design space is approximately  $(N/2)^{FAS}$  (2,744 in this case), which is a more reasonable value than the number of all possible combinations,  $4.82 \times 10^8$ .

Algorithm 1 describes the procedure of our proposed approximate synthesis flow. The flow finds an approximate design with the minimum delay and the required accuracy (i.e., higher than  $accuracy_{min}$ ). In the first step, all adders in the baseline design are classified according to their  $AS$  (Line 2). All  $APs$  of the  $AS$  are then set to 0 (Line 2). Following this, the  $AP$  in each  $AS$  is perturbed by adding 1 (Line 5). The perturbed Verilog design is synthesized, and the delay in the design is calculated (Lines 6-7). Using the synthesized design, a gate-level simulation and static timing analysis are performed to calculate the power and accuracy (Lines 8-11). From the slack and accuracy, the sensitivity factor (SF) is calculated (Lines 12). The SF is defined as

$$SF = \begin{cases} \frac{accuracy - accuracy_{min}}{delay} & , \text{if } accuracy > accuracy_{min} \\ 0 & , \text{else} \end{cases} \quad (2)$$

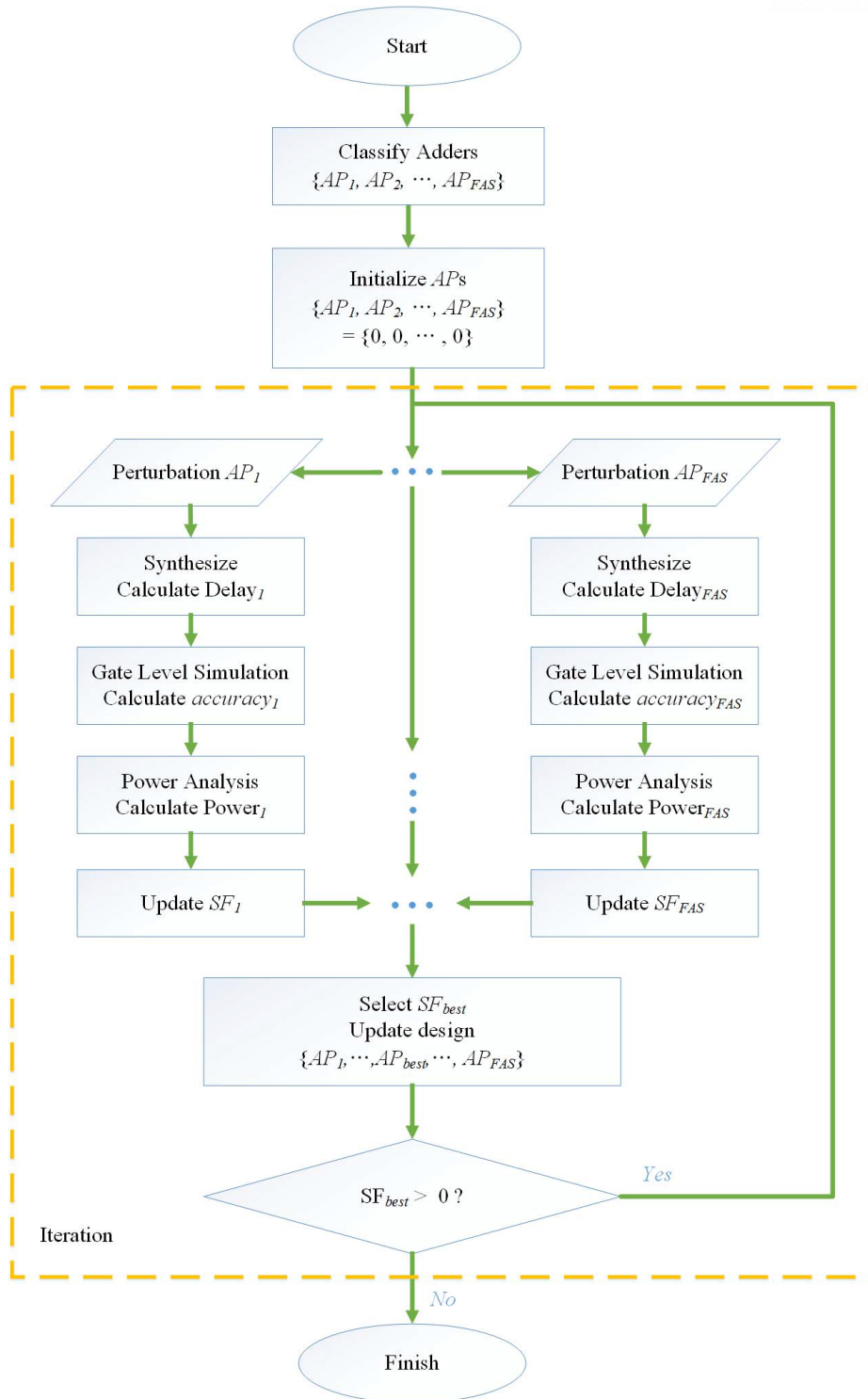
where accuracy is defined in Equation (1). The calculated  $SF_i$  is added to the SF list. Following calculations, the perturbed design is reverted to the original one (Line 13). If all perturbations and SF calculations are complete from the SF list, the design with the highest SF is selected (Line 15). The selected design is used as a seed for the next iteration (Line 16). If the highest SF is zero or negative, the flow returns a final solution, and ends. The proposed synthesis flow is summarized in Figure 3.5. Low-power or highly energy-efficient design, which are our main concerns here, can be achieved by



re-synthesizing the final solution of the synthesis flow with an appropriate clock constraint, i.e., the minimum available clock of the baseline design.

```
1  Classify adders according to  $AS$ 
2   $AP_i \leftarrow 0$ , where  $i = 1, \dots, FAS$ 
3  while  $SF_{best} > 0$  do
4    for  $i := 1$  to  $FAS$  do
5       $AP_i \leftarrow AP_i + 1$ 
6      Synthesis  $\{newAP_0, newAP_1, \dots, newAP_{FAS}\}$ 
7      Calculate  $delay_i$ 
8      Gate level simulation
9      Calculate  $accuracy_i$ 
10     Power analysis
11     Calculate  $power_i$ 
12     Calculate  $SF_i$ 
13     Recover design  $AP_i \leftarrow AP_i - 1$ 
14   end for
15    $SF_{best} = \max(SF_1, SF_2, \dots, SF_{FAS})$ 
16   if  $SF_{best} > 0$  then
17     Select  $\{AP_1, AP_2, \dots, AP_{FAS}\}_{best}$ 
18   end if
19 end while
20 Return  $\{AP_1, AP_2, \dots, AP_{FAS}\}$ 
```

**Algorithm 1.** Sensitivity-based approximate synthesis flow



**Figure 3.5: Proposed synthesis flow.**

## Chapter IV

### EXPERIMENTAL RESULTS

#### 4.1 Experimental Setup

The proposed synthesis flow is written in Tcl (Tool Command Language) and executed on a 2.6 GHz Intel Xeon E7-4860 Linux workstation. The FIR filter is implemented using the worst corner library of the TSMC 65nm technology node and an RTL compiler [18]. A tight timing constraint is used to synthesize the approximate design with minimum delay. Following the synthesis, the minimum delay in the FIR filters is calculated by the summation of the worst negative slack and the clock period.

For accuracy simulations, Cadence NC-Verilog is used [19]. We generate 10,000 random patterns for RTL simulations and compare the output patterns with the correct ones. The accuracy value is calculated according to Equation (1). We set  $accuracy_{min}$  to 95%.

Power consumption is reported using Synopsys PrimeTime-PX [20]. We calculate total power consumption, which includes static and dynamic power. The value change dump file generated from the previous gate-level simulation is used to calculate the switching activity of each net and the minimum clock period for each design is used to report the dynamic power.

#### 4.2 FIR Filter Implementation

We implement a FIR filter using our proposed approximate synthesis flow. We synthesize a four-tap FIR filter with the coefficient set  $\{105, 831, 621, 815\}$ . Figure 2.1 shows the structure of the implemented FIR filter. In this experiment, the bit width of the coefficients is set to 12. Since the largest coefficient is 831 in the four-tap FIR filter, 12 bits are sufficient to represent four coefficients in SPT. The bit widths of the input and output are set to 15 bits and 28 bits, respectively. For addition, 28-bit adders are used. The four given coefficients are implemented using six adders according to the previously introduced CSE algorithm. The  $AS$  of each coefficient is different. The  $AS$ s of  $\{15, 129\}$  are 1, those of  $\{105, 831\}$  are 2, and the  $AS$ s of  $\{621, 815\}$  are 3. In the first iteration of the synthesis flow, the accuracy configurations of the adders with the same  $AS$  are perturbed one by one. The perturbed designs ( $\{1,0,0\}$ ,  $\{0,1,0\}$ , and  $\{0,0,1\}$ ) are synthesized and simulated.  $\{0,0,1\}$ , which have the highest SF, is selected and set as seed of the following iteration. After several iterations, the final output is  $\{11,16,14\}$ . Figure 4.1-3 show an implemented design space using the proposed synthesis flow. The black dots are generated by randomly but separately configuring the  $AP$  of all adders. The red dots represent the results from iterations of the approximate synthesis flow. The white space shows the reachable design space with higher accuracy than  $accuracy_{min}$  by configuring the  $AP$ s of each adder.

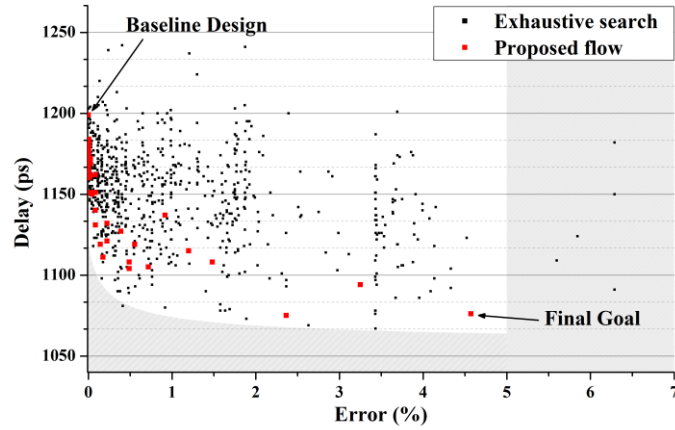


Figure 4.1: Accuracy vs. delay domain of the proposed synthesis flow (red) and the exhaustive research of FIR Filter (black)

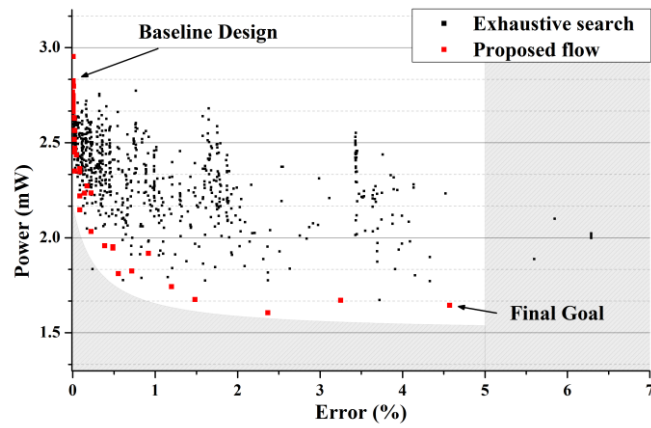


Figure 4.2: Accuracy vs. power domain of the proposed synthesis flow (red) and the exhaustive research of FIR Filter (black)

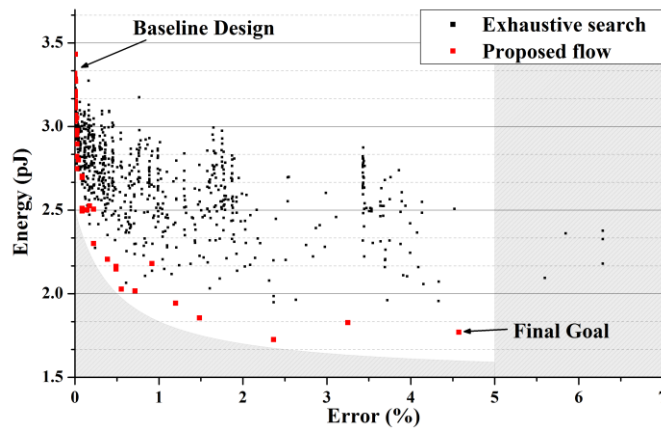


Figure 4.3: Accuracy vs. Energy domain of the proposed synthesis flow (red) and the exhaustive research of FIR Filter (black)

As shown in Figure 4.1, the proposed synthesis flow can successfully follow the minimum delay design. Moreover, it can be shown that the proposed synthesis flow can effectively reduce power and energy consumption.

Since the main concern of our work is obtaining high energy efficiency, we re-synthesize the design acquired from the synthesis flow and implement it using different timing constraints. We then select the result with the lowest energy consumption with a delay not exceeding that of the baseline design. In Figure 4.1, due to EDA tool noise, one design with close to 97.5% accuracy shows slightly lower delay and power consumption than the final solution design. Following re-synthesis, however, the energy consumption of the point is greater than that of the final solution.

	Delay [ps]	Power [uW]	Energy [fJ]
Baseline	1199	2796	3352
Flow result	1076	1687	1815
Min. Energy design	1198	1379	1652
	Improvement [%]		
	Delay	Power	Energy
Flow result	10.3	39.7	44.7
Min. Energy design	0.0	50.7	50.7

**Table 4.1: Approximation results in 4-tap FIR filter with FAS = 3.**

	Delay [ps]	Power [mW]	Energy [pJ]
Baseline	1988	10.7	21.3
Flow result	1876	8.9	16.7
Min. Energy design	1983	8.2	16.3
	Improvement [%]		
	Delay	Power	Energy
Flow result	5.6	16.8	21.6
Min. Energy design	0.0	23.3	23.5

**Table 4.2: Approximation results in 25-tap filter with FAS = 4.**

Table 4.1 summarizes the results of the approximate synthesis flow. Performance improves by 10.3%, and power consumption is reduced by 39.7% over conventional FIR filter design. The energy is calculated by multiplying delay and power. Energy consumption per operation is reduced by 44.7%. To achieve further energy reduction, we change the timing constraint and find the minimum energy design for which delay is shorter than the baseline design. In this way, we achieve up to 50.7% reduction in energy consumption. The runtime of the proposed synthesis flow is 84 minutes for the four-tap FIR filter.

We apply the approximate synthesis flow to a 25-tap FIR filter, the coefficients of which are  $\{-2423, -113, 1564, 762, -1816, -1517, 2276, 3140, -2434, -6205, 2726, 20680, 30093, 20680, 2726, -6205, -2434, 3140, 2276, -1517, -1816, 762, 1564, -113, -2423\}$ . The results are shown in Table 4.2. In the 25-tap case, we can improve the performance by 5.6% with power and energy savings of up to 16.8% and 21.6%, respectively. The runtime of the proposed synthesis flow is 407 minutes for the 25-tap FIR filter.

To verify our methodology, we apply the proposed synthesis flow to five different FIR filters [21–25]. The specifications of the FIR filters are summarized in Table 4.3. The delay, power, and energy information of the baseline designs of the FIR filters are also summarized in Table 4.3.

FIR Filter	Tap	FAS	Delay [ns]	Power [mW]	Energy [pJ]
[21]	15	3	0.98	5.68	5.5
[22]	15	4	1.27	4.06	5.1
[23]	28	4	1.15	11.6	13.4
[24]	34	3	1.17	13.4	15.7
[25]	39	3	1.20	18.4	22.1

**Table 4.3: Specifications of the experimented FIR filters.**

FIR Filter	Accuracy [%]	Delay [ns]	Power [mW]	Energy [pJ]	Energy Reduction [%]
[21]	97.83	0.93	4.34	4.06	26.9
[22]	95.32	1.14	3.17	3.62	29.5
[23]	96.03	1.15	8.13	9.34	30.1
[24]	95.66	1.15	8.34	9.59	38.9
[25]	95.19	1.12	13.60	15.27	30.8

**Table 4.4: Experimental results of FIR filters with the approximate synthesis flow.**

The FIR filters are synthesized using the proposed synthesis flow, whereas the bit width of the inputs, the coefficients, and the output width are set to eight, 16, and 24 bits, respectively. The results of the synthesis flow are shown in Table 4.4. The accuracies of the filters are higher than the threshold of 95%. The energy consumptions of the FIR filters are reduced by up to 38.9% and 31.2% on average.

### 4.3 Image FIR filter Experiment

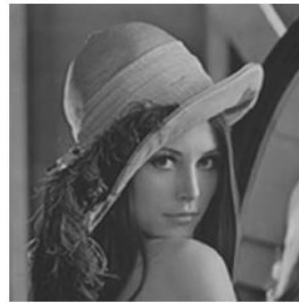
A FIR low-pass filter is implemented in [23] for blurred images. Since the image used is two-dimensional, we apply the FIR filter first in the vertical direction, and divide the output by filter gain. Following this, the FIR filter is applied in the horizontal direction, and the output is divided by filter gain once again. Figure 4.4 (a) shows the original image and Figure 4.4 (b) shows the blurred image processed by the baseline FIR filter. Figure 4.4 (c) shows the image processed by the proposed FIR filter. To verify the output quality of the processed image, peak signal-to-noise-ratio (PSNR) is used. PSNR is defined as

$$PSNR = 10 \times \log\left(\frac{255^2}{\sigma_{\text{noise}}^2}\right)$$

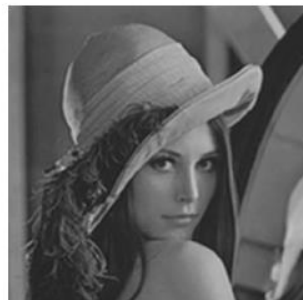
where  $\sigma_{\text{noise}}^2$  is the variance of the difference between Figure 4.4 (b) and others. FIR filters with varying accuracies are simulated. As accuracy decreases, the image becomes dark. This is because the proposed adder approximates the previous carry and the approximation error renders the result lower in value than the exact result. If the approximation error continues to increase, the results assume negative values, which are expressed as white dots. We find that we can achieve 30.8% energy saving with 46.3 dB PSNR.



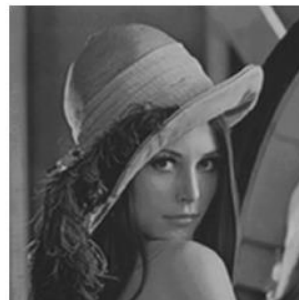
Original  
(a)



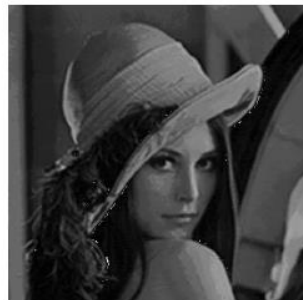
Baseline design  
(b)



Proposed  
PSNR = 46.3  
30.0% Energy reduced  
(c)



89.2% Accuracy  
PSNR = 40.6  
36.9% Energy reduced  
(d)



65.4% Accuracy  
PSNR = 14.7  
45.7% Energy reduced  
(e)



45.6% Accuracy  
PSNR = -9.5  
48.8% Energy reduced  
(e)

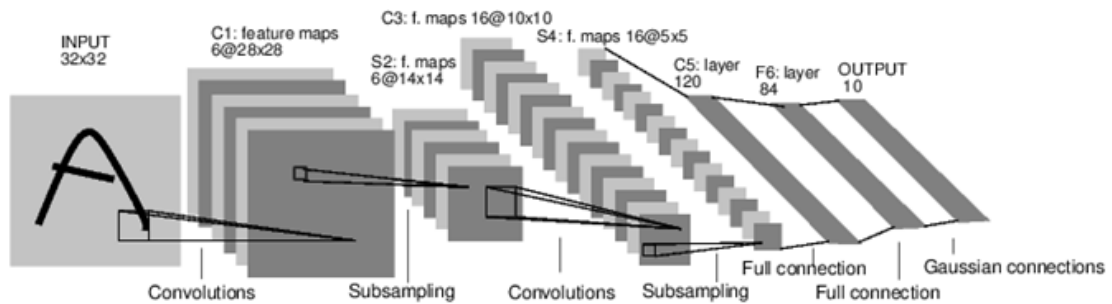
**Figure 4.4: Filtered images using optimized FIR filters.**



## Chapter V

# APPROXIMATIONS SYNTHESIS FLOW FOR CONVOLUTIONAL NEURAL NETWORKS

Neuromorphic computing is an emerging computing method inspired by the human brain. Neural network is one of the most actively studied areas of neuromorphic computing. The computations of the neural networks can be trained to operate as intended and this fact facilitates the application of approximation techniques to neural networks. Computation effort can be effectively reduced and the output quality loss caused by the approximation techniques can be significantly degraded. In this Chapter, we briefly introduce convolutional neural network (CNN) and previous researches about approximate technique for CNN. Then, we propose novel approximate synthesis flow for the CNN to implement energy-efficient CNN hardware and experimentally verify that the approximate synthesis flow can efficiently reduce the power consumption with acceptable accuracy.



**Figure 5.1: General architecture of convolutional neural network**

### 5.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is an artificial neural network showing an outstanding performance in analyzing visual image (e.g. object localization/detection, scene classification, etc.). Figure 5.1 shows the general architecture of the CNN. In the convolution layer, the element-wise multiplication of the input feature map matrix and kernel matrix is taken, and partial sums are accumulated to produce an element of output feature map. Output feature map contains the features of the input image/feature map specified by the kernel. These output feature maps are fed to the pooling layers to subsample the feature map to leave only meaningful results. It also reduces the size of the feature map and consequently reduce the amount of data to be computed. Passing through convolution layers and pooling layers extract high-level feature of the input image. The extracted feature data are fed to the fully-connected layer to compute final output results. CNN's output usually comes in the form

of probability, and the output of the highest probability is the result of image recognition. Figure 5.2 describes the computation in convolution layer and Table 5.1 is shape parameters of convolution layer in Figure 5.2. A kernel of  $C$  channels is convolved with an input feature map of  $C$  channels to produce a single channel of the output feature map. Thus,  $M$  kernels produce  $M$  channel of output feature map through the convolution layer. The number of output feature map is same as the number of inputs feature map. After additional processes like subsampling, computed output feature maps are fed to the next convolution layer as input feature map.

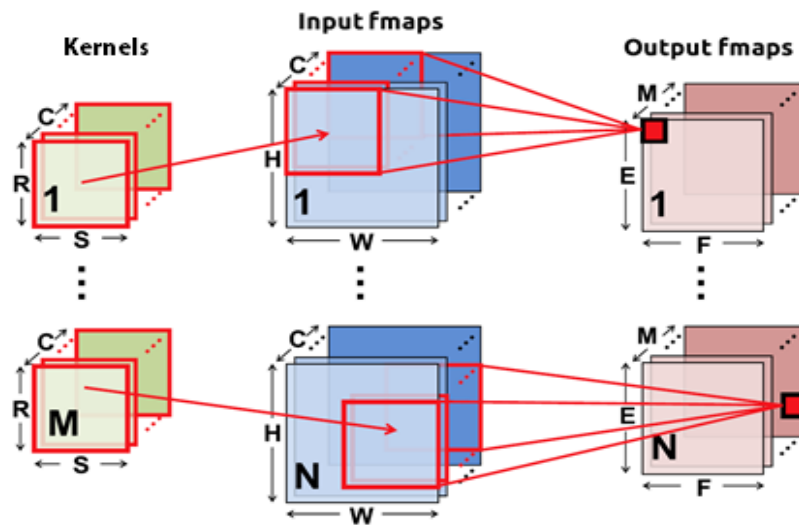


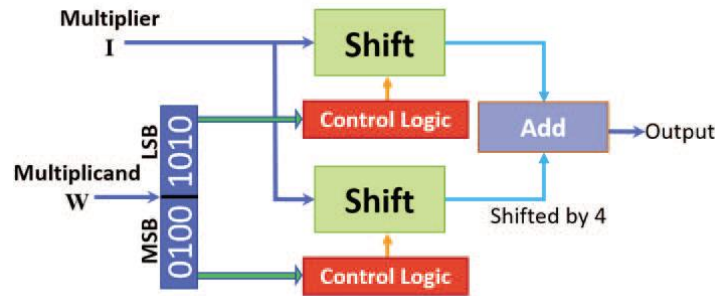
Figure 5.2: Computation of convolution layer [26]

Shape Parameter	Description
$N$	batch size of 3D fmaps
$M$	# of 3D kernels / # of ofmap channels
$C$	# of ifmap/kernel channels
$H/W$	ifmap plane height/width
$R/S$	kernel plane height/width
$E/F$	ofmap plane height/width

Table 5.1: Shape parameters of convolution layer [26]

## 5.2. Related Work

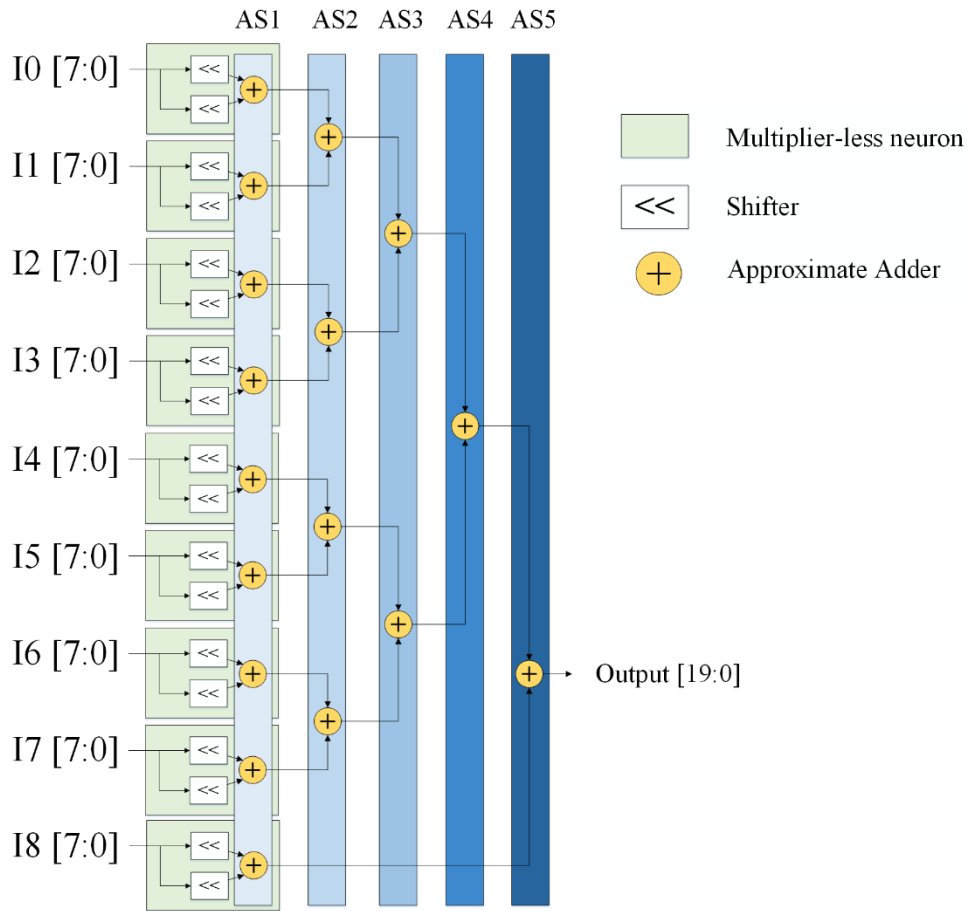
Despite CNN's outstanding image recognition capabilities, CNN's high computational intensity is one of the biggest obstacles to CNN's hardware implementation. One of the main computational efforts are multiply-and-accumulate (MAC) in the convolution layer. Many previous researches have attempted to reduce the computational cost of MAC through approximations. Sarwar et al. [27] proposed multiplier-less artificial neurons. They replaced the multiplier with shifters and adders to reduce the computational cost caused by the multiplier. Figure 5.3 shows the working principle of the proposed multiplier-less artificial neuron with 8-bit weight for neural networks. They split the weight in 4-bit units and round up/down the binary numbers of each unit considering the minimum loss output. These rounded up/down binary numbers in each unit have only 1 non-zero bit and it is regarded as the shifting factor of input data. When their multiplier-less neuron is applied to CNN, accuracy loss caused by the approximation was  $\sim 0.5\%$  while the power improvement was about 35%.



**Figure 5.3: Working principle of an 8-bit Multiplier-less Artificial Neuron [27]**

## 5.3. Approximate Synthesis Flow for MAC module

In the previous section, [27] showed low accuracy loss despite replacing the multiplier with shifters and adders. Since the working principle with shift and addition is similar to the FIR filter shown in Figure 2.1. Similar scheme of approximate synthesis flow for the FIR filter also can be applied to the multiplier-less artificial neuron. Figure 5.4 shows the structure of the approximate MAC module with  $FAS = 5$  for the convolution with  $3 \times 3$  kernel. This MAC module can compute element-wise multiplication of two  $3 \times 3$  matrices. Each artificial neuron was designed based on the multiplier-less artificial neurons proposed in [27] except that adder is replaced with approximate adder/subtractor explained in Section 3.1. Each neuron gets 8-bit input image/feature data and an 8-bit weight. The partial sums produced by neurons are accumulated through the adder steps to produce a 20-bit approximate output feature data. As in the case of the FIR filter, the adders belonging to the same adder step have the same  $AP$ -bit of the approximate adder / subtractor to reduce the number of combinations to a computable level.



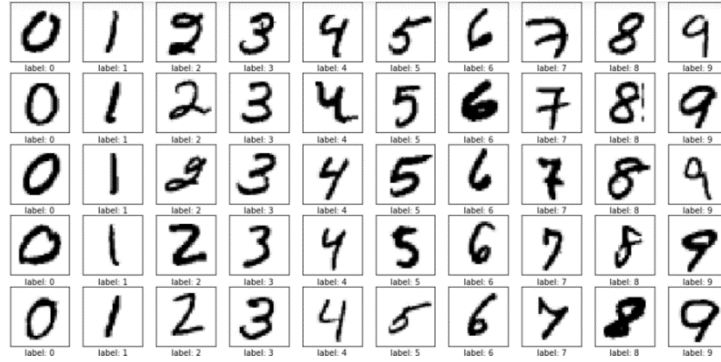
**Figure 5.4: Structure of the approximate MAC module for the convolution with  $3 \times 3$  kernel**

Our proposed approximate synthesis flow described in Section 3.2 finds an optimal approximate design of MAC module with the minimum delay and the required accuracy. The structure of MAC module for  $5 \times 5$  kernel also can be designed in similar way to the MAC module for  $3 \times 3$  kernel. It has 25 multiplier-less artificial neurons and each output is accumulated through 6 adder steps to produce 21-bit approximate output feature data.

## 5.4 Experimental Results

### 5.4.1 Experimental Setup

The MAC module for  $3 \times 3$  kernel and  $5 \times 5$  kernel are designed using Verilog to verify the power and energy consumption of the approximate MAC modules obtained by the proposed synthesis flow. Experimental setup for the analysis of power consumption was same as that of FIR filter.



**Figure 5.5: MNIST handwritten numbers database**

Layer name	Parameter
Input image	size: 28x28, channel: 1
Convolution	kernel: 5x5, channel: 20
Max pooling	kernel: 2x2, stride: 2
ReLU	
Convolution	kernel: 5x5, channel: 50
Max pooling	kernel: 2x2, stride: 2
ReLU	
Fully connected	channel: 500
ReLU	
Fully connected	channel: 10
Softmax	

**Table 5.2: Network parameter of CNN for MNIST**

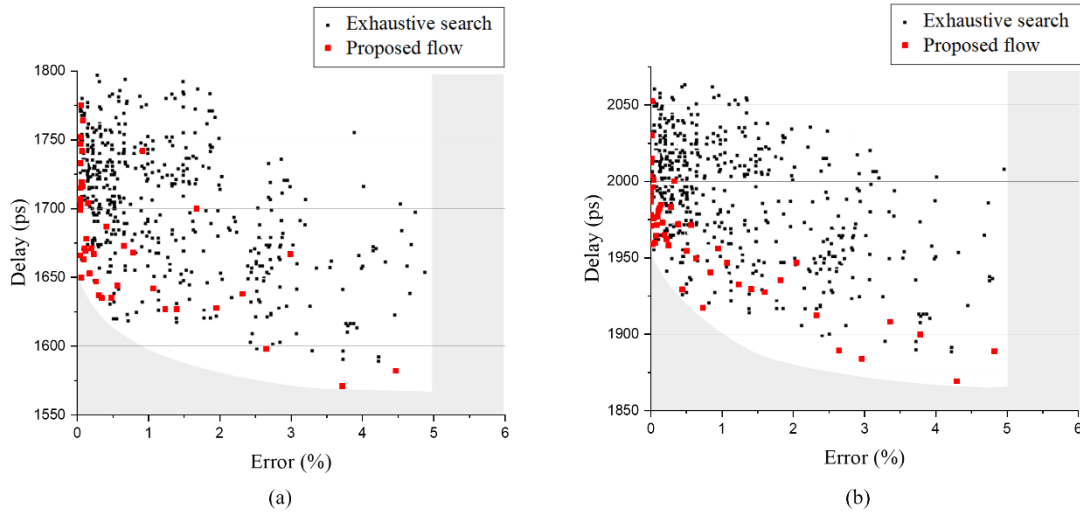
For the analysis of the output quality of the proposed approximate CNN, we implemented CNN model using C++ to evaluate the classification accuracy of approximate CNN. The network parameters of CNN model to be trained and tested are described in Table 5.2.

MNIST database is used for the evaluation of implemented CNN model. MNIST [29] is a database of handwritten numbers commonly used to train various image processing systems. The purpose is to determine what numbers grayscale  $28 \times 28$  pixel image represents for numbers from 0 to 9. MNIST database contains 60,000 training images and 10,000 testing images. Figure 5.5 shows the exemplary MNIST images.

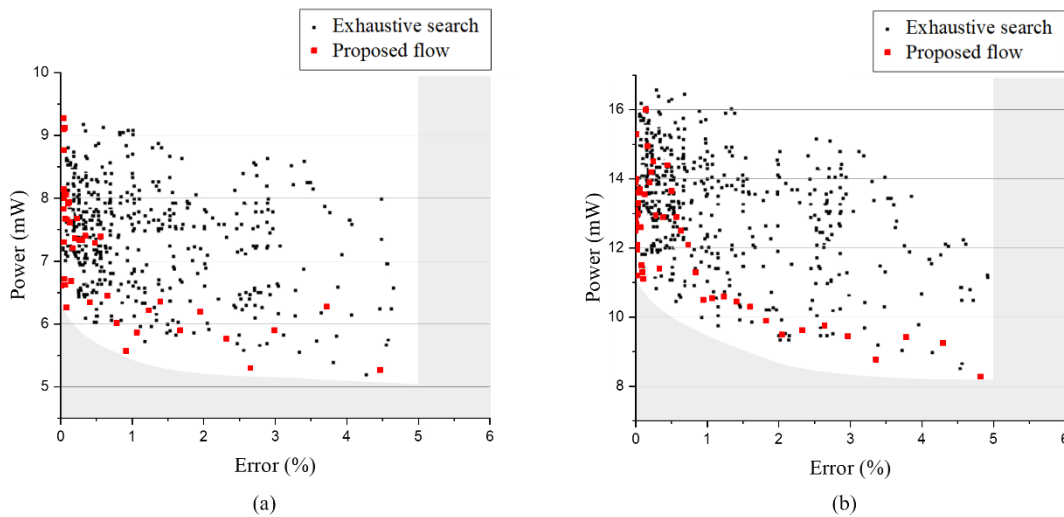
#### 5.4.2 Power Consumption

MAC modules for convolution with  $3 \times 3$  kernel and  $5 \times 5$  kernel are implemented using Verilog and approximated through the proposed synthesis flow to calculate the power consumption. At the end of the flow, MAC module for  $3 \times 3$  kernel had *AP* of {9,8,9,10,11} for each adder steps. The *AP* of MAC

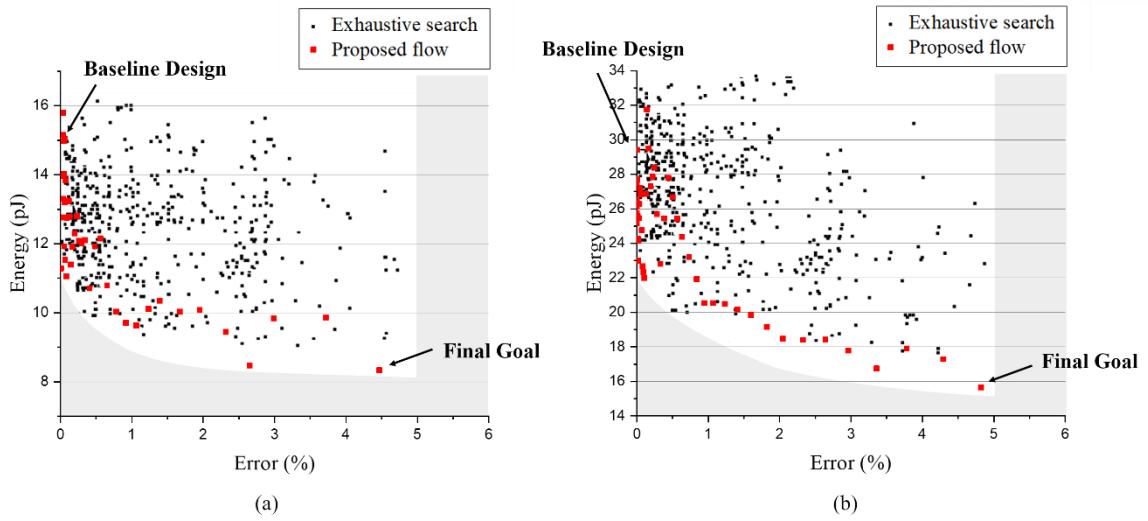
module for  $5 \times 5$  kernel was  $\{9,7,10,9,11,12\}$ . Figure 5.6-8 show the design space of MAC modules obtained by following the proposed synthesis flow. The black dots represent the results obtained by randomly configuring the *APs* and red dots represent the results from the proposed synthesis flow. The white area shows attainable design space by configuring *APs* within error budget. The fact that the red dots successfully follow the parabolic bottom line of white area prove that proposed flow help to find configurations of *APs* for the energy-efficient design.



**Figure 5.6: Accuracy vs. delay domain of the proposed synthesis flow (red) and the exhaustive research (black) of MAC modules for (a)  $3 \times 3$  kernel and (b)  $5 \times 5$  kernel**



**Figure 5.7: Accuracy vs. power domain of the proposed synthesis flow (red) and the exhaustive research (black) of MAC modules for (a)  $3 \times 3$  kernel and (b)  $5 \times 5$  kernel**



**Figure 5.8: Accuracy vs. Energy domain of the proposed synthesis flow (red) and the exhaustive research (black) of MAC modules for (a)  $3 \times 3$  kernel and (b)  $5 \times 5$  kernel**

Table 5.3 summarizes the final results from the approximate synthesis flow. Critical path delays of MAC modules for  $3 \times 3$  kernel and  $5 \times 5$  kernel are decreased 10.9% and 5%, respectively. Power consumption is improved 39.8% and 40.4%, respectively. The energy per operation is calculated by multiplying delay and power consumption. We achieved energy improvements up to 46.4% and 43.4% respectively compared to the baseline of MAC modules.

		Delay [ps]	Power [mW]	Energy [pJ]
$3 \times 3$ kernel	Baseline	1775	8.76	15.55
	Flow result	1582	5.27	8.34
$5 \times 5$ kernel	Baseline	1987	13.9	27.62
	Flow result	1889	8.28	15.64

**Table 5.3: Approximation results of MAC module**

### 5.4.3 Classification Accuracy

To verify the effect of approximation, we implemented the C++ CNN model and evaluated with 10,000 MNIST image samples. Weights are found by training the CNN model. To consider the worst error case, minimum accuracy of MAC operations in convolution layer is assumed. The image

classification result is shown in Table 5.4. The classification accuracy of the baseline model is 98.2%. The classification accuracy of our proposed approximate CNN model is 97.9 % and the output quality degradation caused by approximation is 0.3%. Considering the energy saving is up to 46.4%, it would be reasonable degradation when low energy consumption is needed.

<b>CNN model</b>	<b>Classification Accuracy (%)</b>
Baseline	98.2
Approximate	97.9

**Table 5.4: Classification accuracy of baseline and approximate C++ CNN model**



## Chapter VI

### CONCLUSION

In this paper, we apply approximate computing to a FIR filter to enhance efficient energy. The FIR filter has a MAC structure, and multipliers are replaced by shifters and adders/subtractors that are approximated.

For the approximation, we propose an approximate adder/subtractor in order that the accuracy of the approximate adder/subtractor is configurable and switching between the adder and the subtractor is possible. The error in the proposed approximate adder is analyzed.

Moreover, we propose a novel approximate synthesis flow that can find the optimal configurations of approximate adders. Using the proposed synthesis flow, we achieve up to 10.3% in terms of performance improvement and 50.7% in terms of power and energy saving over conventional FIR filter design.

For further research, we apply the approximate synthesis flow to Convolutional Neural Networks (CNN). We design multiplier-less MAC modules for the convolutional computation and find the optimal approximate design. We also verify the impact of approximation on output quality degradation in MNIST handwritten digits recognition.

Our proposed approximate synthesis flow improved energy consumption of approximate MAC modules for multiplication of  $3 \times 3$  and  $5 \times 5$  matrices up to 46.4 % and 43.4 % respectively while output quality degradation of the MNIST handwritten digit recognition was lower than 1 %.

## REFERENCES

- [1] D. M. Malcolm and G. D. Andrew, "Multiplierless FIR Filter Design Algorithms", Proc. SPL, 2005, pp.186-189.
- [2] C. Y. Yao, H. H. Chen, T. F. Lin, C. J. Chien and X. T. Hsu, "A Novel Common-Subexpression Elimination Method for Synthesizing Fixed-Point, FIR Filters", IEEE Trans. CAS I, 51(11) (2004) pp.2215-2221.
- [3] J. H. Choi, N. Banerjee and K. Roy, "Variation-Aware Low-Power Synthesis Methodology for Fixed-Point FIR Filters", IEEE. Trans. CAD, 28(1) (2009) pp.87-97.
- [4] A. B. Kahng, S. H. Kang, R. Kumar and J. Sartori, "Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits", Proc. ICCD, 2013, pp.47-53.
- [5] V. Gupta, D. Mohapatra, P. P. Sang, A. Raghunathan and K. Roy, "IMPACT:IMPrecise Adders for Low-Power Approximate Computing", Proc. ISLPED, 2011, pp.409-414.
- [6] N. Zhu, W. L. Goh and K. S. Yeo, "An Enhanced Low-Power High-Speed Adder for Error-Tolerant Application", Proc. ISIC, 2009, pp.69-72.
- [7] S. L. Lu, "Speeding Up Processing with Approximation Circuits", IEEE Computer, 37(3) (2004) pp.67-73.
- [8] A. K. Verma, P. Brisk and P. lenne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design", Proc. DATE, 2008, pp.1250-1255.
- [9] D. Shin and S. K. Gupta, "A Re-design Technique for Datapath Modules in Error Tolerant Applications", Proc. ATS, 2008, pp.431-437.
- [10] A. B. Khang and S. H. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs", Proc. DAC, 2012, pp.820-825.
- [11] R. Venkatesan, A. Agarwal, K. Roy and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing", Proc. ICCAD, 2011, pp.667-673.
- [12] P. Kulkarni, P. Gupta and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture", Proc. VLSI Design, 2011, pp.346-351.
- [13] K. J. Cho, K. C. Lee, J. G. Chung and K. K. Parhi, "Design of Low-Error Fixed-Width Modified Booth Multiplier", IEEE Trans. VLSI, 12(5) (2004) pp.522-531.

- [14] C. H. Chang and R. K. Satzoda, "A Low Error and High Performance Multiplexer-Based Truncated Multiplier", IEEE Trans. VLSI, 18(12) (2010) pp.1767-1771.
- [15] J. P. Wang, S. R. Kuang and S. C. Liang, "High-Accuracy Fixed-Width Modified Booth Multipliers for Lossy Applications", IEEE Trans. VLSI, 19(11) (2011) pp.52-60.
- [16] C. Liu, J. Han and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery", Proc. DATE, 2014, pp.95.
- [17] F. Farshchi, M. S. Abrishami and S. M. Fakhraie, "New Approximate Multiplier for Low Power Digital Signal Processing", Proc. CADs, 2013, pp.25-30.
- [18] Cadence RTL Compiler User Guide. <http://www.cadence.com>.
- [19] Cadence NCVerilog User Guide. <http://www.cadence.com>.
- [20] Synopsys PrimeTime User Guide. <http://www.synopsys.com>.
- [21] D. Goodman and M. Carey, "Nine Digital Filters for Decimation and Interpolation", Proc. TASSP, 1977, pp.121-126.
- [22] F. Xu, C. H. Chang and C. C. Jong, "Design of Low-Complexity FIR Filters based on Signed-Powers-of-Two Coefficients with Reusable Common Subexpression", Proc. TCAD, 2007, pp.1898-1907.
- [23] K. Johansson, "An Improved Synthesis Method for Low Power Hardwired FIR Filters", Ph. D dissertation, 2008.
- [24] D. Shi and Y. J. Yu, "Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders", Proc. TCAS I, 2011, pp.126-136.
- [25] S. Rosa, Vagner, E. Costa, J. C. Monteiro and S. Bampi, "An Improved Synthesis Method for Low Power Hardwired FIR Filters", SBCCI, 2004, pp.237-241.
- [26] Chen, Y. H., Krishna, T., Emer, J. S., & Sze, V., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks.", IEEE JSSC, 2017, 52.1: 127-138.
- [27] Sarwar, S. S., Venkataramani, S., Raghunathan, A., & Roy, K., "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing", Proc. DATE, 2016, pp. 145-150
- [28] Synopsys Design Compiler User Guide. <http://www.synopsys.com>.
- [29] MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>