DUAL SEGMENTED AND RECONFIGURABLE APPROXIMATE MULTIPLIERS FOR ERROR-TOLERANT APPLICATIONS

by

Ling Li

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
June 2021

Table of Contents

List of Tables iv

List of Figures v

Abstract vii

List of Abbreviations Used viii

Acknowledgments ix

Chapter 1 Introduction 1

1.1 Motivations 1

1.2 Contributions 3

1.3 Thesis Organization 4

Chapter 2 Research Background 5

2.1 Rounding Based Approximate Multiplier 5

2.2 Using Approximate Compressor in the Partial Product Matrix 7

2.3 Segment-based Approximate Multiplier 8

2.3.1 SSM-based Multiplier 11

2.3.2 DSM-based Multiplier 13

2.3.3 TOSAM 15

2.4 A Comparative Evaluation of Approximate Multiplier Designs 17

2.4.1 Error Characteristics 17

2.4.2 Circuit Characteristics 24

2.5 Conclusion 25

Chapter 3 Dual Segmentation Approximate Multiplier 26

3.1 Dual Segmentation Approximate Multiplier Implementation 26

3.2 Simulation and Synthesis Results 32

Chapter 4 A Merged Approximate Multiplier with Two Precisions 34

4.1 Proposed Approximate Multiplier Implementation 34

4.2 Simulation and Synthesis Results 41

Chapter 5 Conclusion and Future Work 43

5.1 Conclusion 43

5.2 Future Work 44

Bibliography 45

Appendix Permission 48

List of Tables

Table 2.1 Circuit's parameter of 8-bit static and dynamic segment selectors........11

Table 2.2 A comparison of delay, power and area of the approximate multiplier designs.........25

Table 3.1 Circuit's parameter of unsigned 16-bit exact and approximate multipliers.........33

Table 3.2 Circuit's parameter of unsigned 32-bit exact and approximate multipliers.........33

Table 4.1 Circuit's parameter of 16-bit merged and separated reconfigurable designs.........41

List of Figures

Figure 2.1 The schematic of the ROBA multiplier 6

Figure 2.2 Partial product matrix reduction of an 8-bit multiplier [6] 8

Figure 2.3 The schematic of segment-based approximate multiplier ($m < n$) 9

Figure 2.4 The *n*-bit dynamic segment selector 10

Figure 2.5 The *n*-bit static segment selector 11

Figure 2.6 A numeric example of the16-bit SSM-based multiplier (*n* = 16, *m* = 4) 13

Figure 2.7 A numeric example of the16-bit DSM-based multiplier (*n* = 16, *m* = 4) 14

Figure 2.8 A numeric example of the16-bit TOSAM (*k* = 3, *m* = 7) 16

Figure 2.9 A comparison of (a) MED, (b) MaxED and (c) MinED of the 8-bit approximate multiplier designs with all possible input operands 19

Figure 2.10 A comparison of (a) MED, (b) MaxED and (c) MinED of the 8-bit approximate multiplier designs with all possible input operands 21

Figure 2.11 A comparison of (a) MRED, (b) MaxRED and (c) MinRED of the 8-bit approximate multiplier designs with all possible input operands 22

Figure 2.12 A comparison of (a) MRED, (b) MaxRED and (c) MinRED of the 8-bit approximate multiplier designs with all possible input operands 24

Figure 3.1 Dual segmentation approximate multiplier architecture 27

Figure 3.2 A general example of the *m*-bit static segment approximation 28

Figure 3.3 A general example of the *m*-bit dynamic segment approximation 29

Figure 3.4 A general example of the *m*-bit dual segment approximation 30

Figure 3.5 Example of 16-bit dual segmentation multiplier (*k*=8, *m*=4) 31

Figure 4.1 The sign and magnitude format of the input operands 35

Figure 4.2 The schematic for mode selection 35

Figure 4.3 Building an 8-bit multiplier from four 4-bit multipliers.......................36

Figure 4.4 Building an 8-bit multiplier from 2-bit, 2×6 and 6-bit multipliers 37

Figure 4.5 The 16-bit SSM-based hybrid approximate multiplier for mode [4 and 8]...38

Figure 4.6 The 16-bit DSM-based hybrid approximate multiplier for mode [6 and 8]...39

Figure 4.7 The 16-bit DSM-based hybrid approximate multiplier for mode [4 and 6]...40

Abstract

Approximate multiplier circuit designs have shown substantial advantages in improving many operational features, such as power, area and delay, in many error-resilient applications such as image processing and deep learning applications.

Existing approximate multiplier circuits in this thesis are first reviewed, evaluated, and compared. The comparison results show that the segment-based multiplier has a good trade-off between accuracy and performance by adjusting segment size. A dual segmentation approximate multiplier is then proposed. Compared to the dynamic segment method (DSM)-based approximate multiplier, the proposed design can reduce the energy by 37.90% for 32-bit multipliers, and by 16.68% for 16-bit multipliers. The DSM and proposed multipliers have almost identical accuracy. A merged approximate multiplier with two configurable precisions is proposed for improving the multiplication performance in fixed point convolutional neural networks (CNN) accelerators. Compared with the single-precision approximate multiplier, the merged approximate multiplier achieves significant performance enhancements with minimal accuracy loss.

List of Abbreviations Used

CNN	Convolutional Neural Networks
DSM	Dynamic Segment Method
SSM	Static Segment Method
TOSAM	Truncation and Rounding-based Scalable Approximate Multiplier
ROBA	Rounding-based Approximate Multiplier
LOD	Leading One Detector
MUX	Multiplier
ED	Error Distance
MED	Mean Error Distance
MRED	Mean Relative Error Distance
MaxED	Maximum Error Distance
MinED	Minimum Error Distance
RED	Relative Error Distance

Acknowledgments

I would like to give my deepest gratitude to my supervisor, Dr. Kamal El-Sankary, for his continuous support and encouragement during my graduate study. This thesis would not have been finished without his profound knowledge and insightful suggestions. Furthermore, his motivation and attention to detail impressed me a lot. I would also like to express my gratitude to my co-supervisor Issam Hammad for his valuable guidance and insightful feedback which pushed me to sharpen my thinking and brought my work to a higher level. I also enjoyed the Deep Learning course instructed by him. I would like to thank Dr. Jason Gu and Dr. William J. Philips for being part of my supervisory committee.

I am also grateful to our department secretaries Nicole Smith, Tamara Cantrill and Ola Hamada for their help.

Finally, I would like to express my deepest gratitude to my entire family: my father Chenwei Li, my mother Hui Zhang, my grandfather Defu Li and my grandmother Tongfang Ding, for their unconditional love and support.

Chapter 1 Introduction

1.1 Motivations

As the physical dimensions of CMOS circuits are scaled down to a few tens of nanometers, it has been difficult to improve the circuit performances. Approximate computing has been considered as a potential alternative for error-resilient applications to reduce power, area, and delay, however at the cost of certain accuracy loss [1-11].

Multiplication is a fundamental high-energy operation in image processing and deep learning applications [1-22]. Prior works have explored different techniques to reduce the cost of multiplication using approximate multipliers. Examples of these techniques include rounding the multiplicands to the nearest power of two [11], partial product matrix simplification (e.g., using approximate compressors and sub-multipliers) [6-7], and segment-based approximate designs (e.g., truncating the operands to designated bit-width) [8-10]. Segment-based approximate multipliers allow for a trade-off between accuracy and performance by adjusting segment size *m*. In [8], a static segment method (SSM)-based multiplier was presented, which statically split the input operand into *i* (*m*-bit) segments and performed the multiplication utilizing the segment containing the most significant one. Hashemi et al. [9] extended the idea of leading one segment to implement the dynamic segment method (DSM) in approximate multiplier designs. In [10], a truncation and rounding-based scalable approximate multiplier (TOSAM) was proposed where the multiplicands were truncated with two different lengths and rounded to perform smaller core multiplications. The DSM-based multiplier can provide notably high accuracy, although it has a larger area and higher energy consumption than the SSM design. The SSM-based multiplier has lower accuracy, but it is faster and consumes less energy compared to other segment-based approximate multipliers [8]. This thesis is focused on the design and analysis of two such segment-based multipliers, namely, DSM-based multiplier and SSM-based multiplier.

1.2 Contributions

In this thesis, a dual segmentation approximate multiplier and a merged approximate multiplier with two configurable precisions are presented.

1. A comparative evaluation of existing approximate multipliers is presented in this thesis. All designs were developed in Verilog and circuit performances are obtained using Synopsys DC Compiler based on a 65nm process.
2. This thesis contributes to the design of segment-based approximate multiplier by proposing a novel truncation method. This dual approximate multiplier design uses the SSM to select an initial multiplication segment. Following that, the DSM is utilized to further reduce the initial segment size.
3. A merged approximate multiplier with two configurable precisions is first presented. The idea of this design is to use a 2-to-1 MUX as a mode selector to switch between low and high precisions. The smaller multiplier is used as a building block to build the larger multiplier.

1.3 Thesis Organization

The Thesis is organized as follows:

1. Chapter 2 reviews the related works in approximate multiplier design. A comparison of existing approximate multiplier designs is discussed.
2. Chapter 3 presents a dual segmentation approximate multiplier design and its electrical performance.
3. Chapter 4 presents the hybrid use of a segment-based approximate multiplier with two precisions.
4. Chapter 5 concludes the thesis and discusses the future work.

Chapter 2 Research Background

This chapter presents a review on existing approximate multiplier designs. Approximate multiplier designs mainly use three approximation approaches: i) rounding the multiplicands to the nearest power of two, ii) using approximate compressors to accumulate the partial products, and iii) segment-based approximate multipliers.

2.1 Rounding Based Approximate Multiplier

In [11], an approximate multiplier based on a technique named rounding-based approximate (ROBA) multiplier was proposed, which rounds the input operands to the nearest power of two. This multiplier design is appliable to both signed and unsigned multiplications and is constructed by modifying the conventional multiplication method at the algorithm level.

The schematic of the ROBA multiplier is shown in Figure 2.1, where the inputs and output are represented in two's complement format. $|\boldsymbol{X_r}|$ and $|\boldsymbol{Y_r}|$ are the rounded numbers of the absolute values of the signed inputs. The multiplication of two absolute inputs can be expressed as:

$$\begin{aligned} |\boldsymbol{X}| \times |\boldsymbol{Y}| &= [(|\boldsymbol{X}| - |\boldsymbol{X_r}|) + |\boldsymbol{X_r}|] \times [(|\boldsymbol{Y}| - |\boldsymbol{Y_r}|) + |\boldsymbol{Y_r}|] \\ &\cong |\boldsymbol{X_r}| \times |\boldsymbol{Y}| + |\boldsymbol{Y_r}| \times |\boldsymbol{X}| - |\boldsymbol{X_r}| \times |\boldsymbol{Y_r}| \end{aligned} \tag{2.1}$$

The ROBA multiplication consists of four steps: 1) the first step is to generate the absolute values of the signed inputs and to determine the sign of the output 2) the second step is to extract the nearest value for each input; 3) the third step is to use shift, add, subtraction operations to calculate the multiplication result; 4) the final step is to compute the final multiplication result by adding the proper sign to the unsigned result.

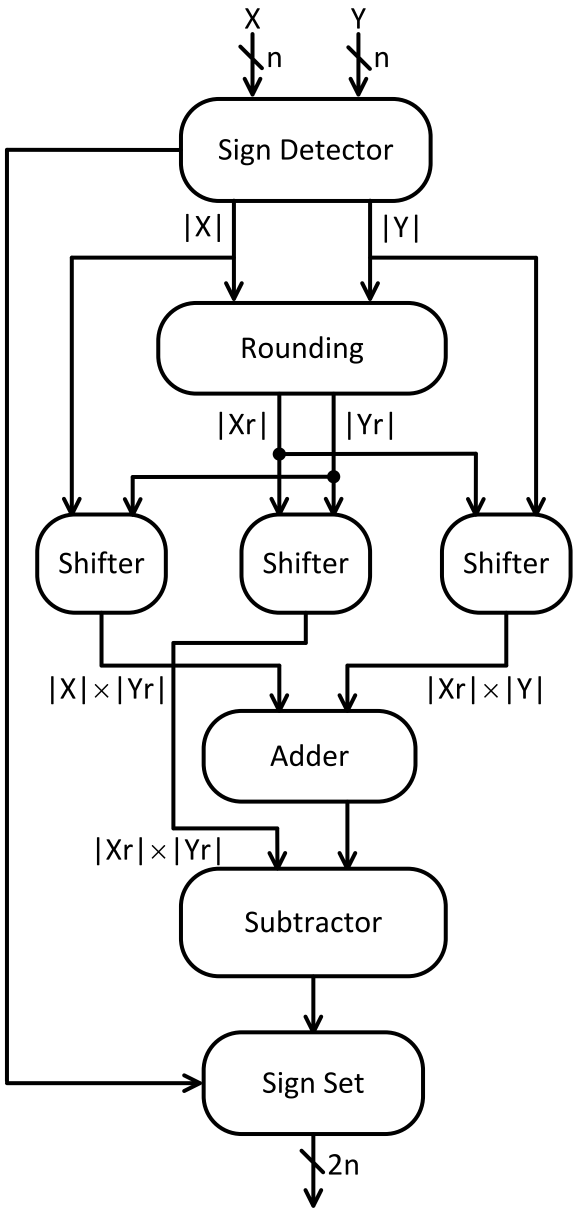


Figure 2.1 The schematic of the ROBA multiplier

2.2 Using Approximate Compressor in the Partial Product Matrix

The approximate compressor-based multiplier uses the algorithm to allocate approximate compressors, exact full adder and half adder in each column of the partial product matrix to save power while providing small error [6-7]. A novel approximate 4/2 compressor design is presented in [6] and then this compressor is used to build 8-bit and 16-bit approximate multipliers. In [7], a new family of the approximate compressor and an algorithm for the allocation of approximate compressors and exact adders in each column of the partial products matrix are used to build approximate multipliers. As an example, the reduction of the partial product matrix of an 8-bit multiplier is shown in Figure 2.2. There are two reduction steps. In the first reduction step, the proposed approximate compressors are only utilized in the less significant part of the partial product matrix to minimize the overall approximation error. The higher-order approximate compressor, 8/4 and 7/4, are composed of smaller approximate sub-compressor. For example, the approximate 8/4 compressor is composed of two approximate 4/2 compressors. Also, the approximate 7/4 compressor is composed of an approximate 4/2 compressor and an approximate 3/2 compressor. The height of the partial product matrix in the first reduction step is reduced from 8 to 4, and then the height is reduced to 2 in the second reduction step. The blue dot represents a carry from the previous adder.

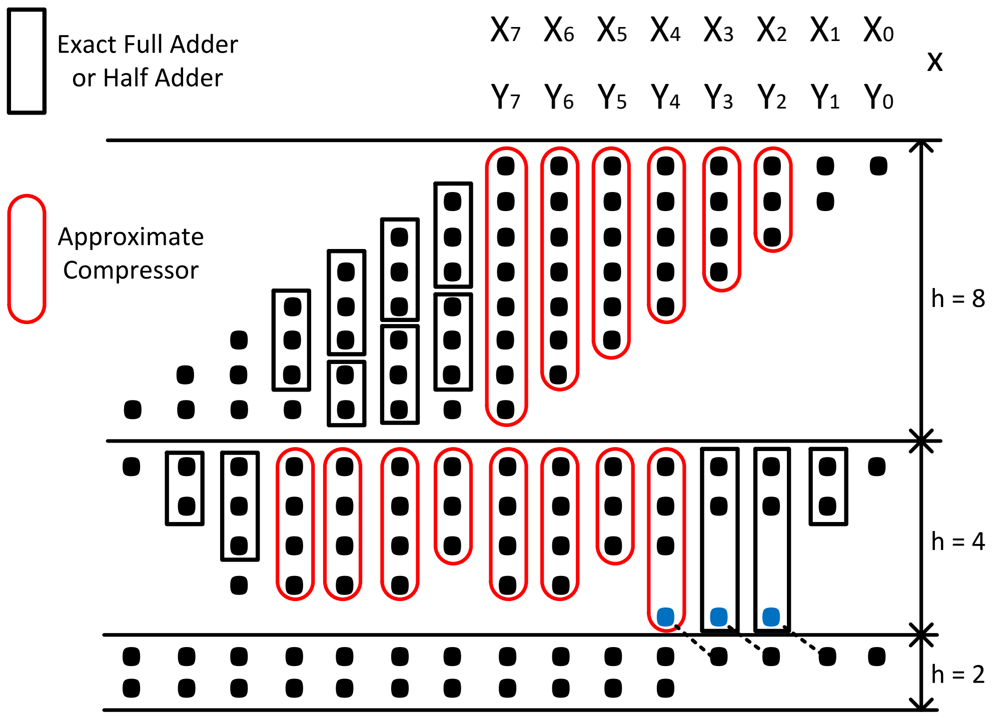


Figure 2.2 Partial product matrix reduction of an 8-bit multiplier [6]

2.3 Segment-based Approximate Multiplier

Figure 2.3 shows a simplified schematic of the segment-based approximate multiplier, where the segment selectors truncate the *n*-bit operands to the *m*-bit segments. The segment size *m* is smaller than the input operand size *n*. For example, the *n*-bit multiplication can be approximated using an *n/2*-bit multiplier or even an *n/4*-bit multiplier. Because the arithmetic operations are performed on the truncated values, the calculation core of the segment-based approximate multiplier is small and consumes less energy compared to that of the exact multiplier. In addition, the precision of the segmented method is controlled by adjusting the segment size *m* and is not significantly affected by the width of the input operands.

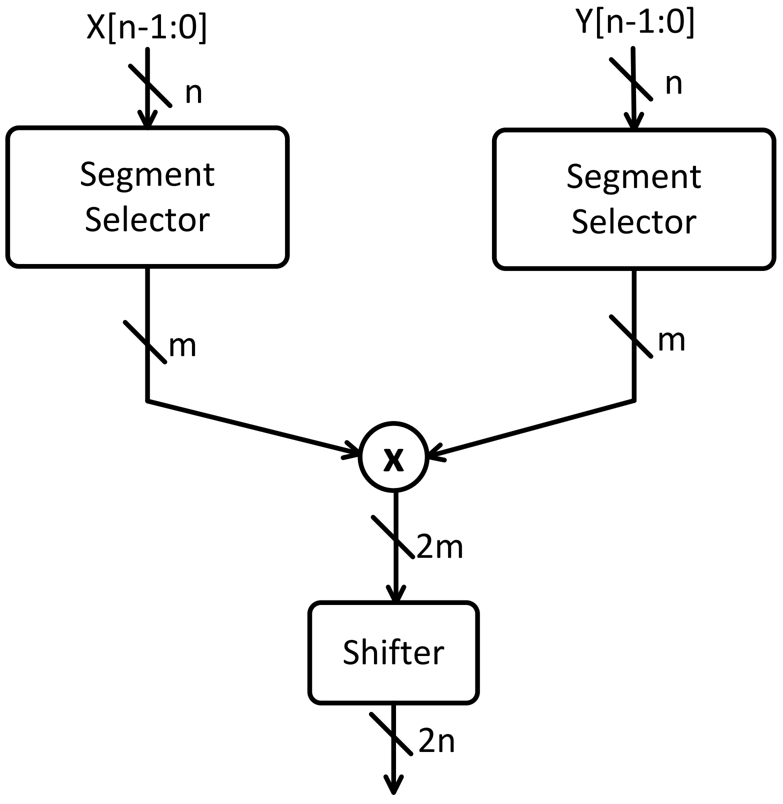


Figure 2.3 The schematic of segment-based approximate multiplier ($m < n$)

The segment selector can be classified into two categories: dynamic [9-10] and static [8]. In the dynamic segment selector, the n-bit leading one detector (LOD) is implemented to locate the most significant one in an n-bit operand. Then an encoder and a multiplexer (MUX) capture the following m-1 bits and set the least significant bit of the segment to one, as shown in Figure 2.4. Compared to the static segment selector, the dynamic segment selector requires utilizing extra complex circuitry. These extra components lead to significant delay, energy and area overheads that may considerably decrease the approximation benefits [8]. The LOD is the module with the highest power consumption and the largest area in the dynamic segment selector [8]. However, the dynamic segment selector delivers higher accuracy designs than the static segment selector. The implementation for the static segment selector is presented in Figure 2.5

where the *n*-bit multiplicand is split into *i* (*m*-bit) segments. Then using OR gates and a MUX, the segment which contains the leading one bit is detected. The static segment selector of the SSM-based approximate multiplier is much faster, smaller and consumes less energy compared to the dynamic segment selector as illustrated in Table 2.1. The 8-bit static and dynamic segment selectors are implemented in Verilog and their delay, area and power reports are obtained from Synopsys DC complier based on 15-nm libraries. As can be seen from Table 2.1, the 8-bit static segment selector consumes almost 89% less power than the dynamic segment selector. The delay and area of the 8-bit static segment selector are about 67% and 77% than those of the 8-bit dynamic segment selector.

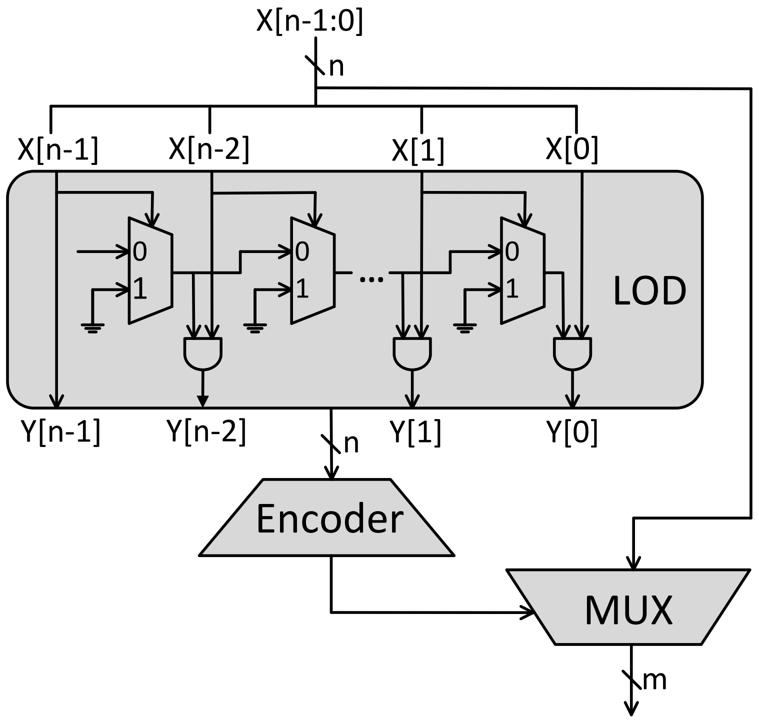


Figure 2.4 The *n*-bit dynamic segment selector

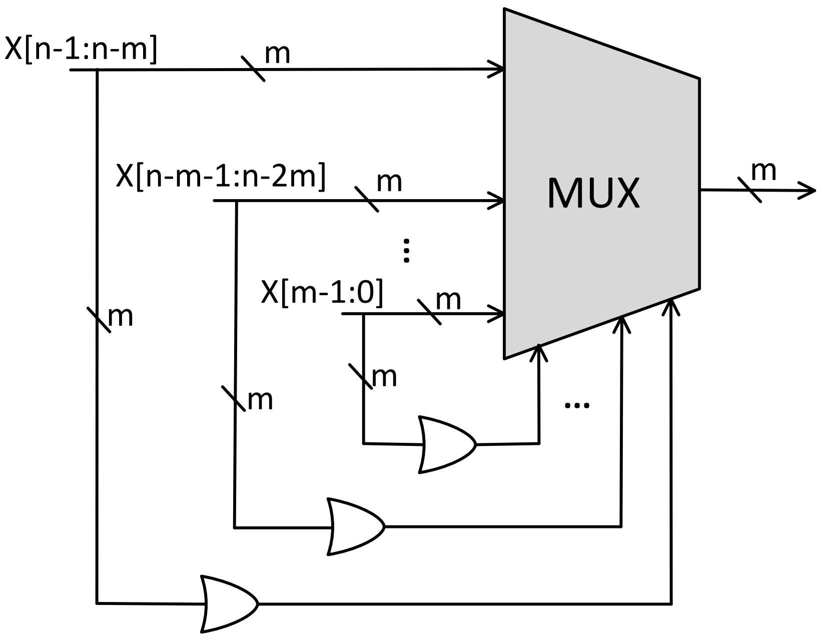


Figure 2.5 The *n*-bit static segment selector

Table 2.1 Circuit's parameter of 8-bit static and dynamic segment selectors

8-bit Segment Selector	**Delay (*ps*)**	**Power (*nW*)**	**Area (μm^2)**
Static	30.84	267.68	3.88
Dynamic	92.83	2543.08	17.15

2.3.1 SSM-based Multiplier

The pseudo-code of Algorithm 1 details the SSM applied to select a 4-bit segment from a 16-bit operand. First, the 16-bit operand is statically divided into four 4-bit segments. If the first segment ($X_{15}X_{14}X_{13}X_{12}$) starting from the most significant bit contains the leading-one-bit, the 16-bit operand is then truncated to the first segment. If there is no leading-one-bit in the first segment, the algorithm checks whether the leading-one-bit is in the following three 4-bit segments. If the upper *i*-1 segments are all zeros, the lower 4-bit segment is selected for accurate multiplication. Figure 2.6 illustrates an example using the SSM to approximate a 16-bit multiplicand with *i*=4 and *m*=4. The two 4-bit segments containing the leading one bit of each input operand are forwarded to the inputs of a 4-bit accurate multiplier. The result of this multiplication is shifted according to the starting bit positions of the 4-bit segments to generate the final approximate result.

Algorithm 1: 4-bit Static Segment Selection

INPUT: (i) *n*-bit operand, given by X*n*X*n*-1…X*0*; (ii) *i* possible *m*-bit segments

OUPUT: the selected *m*-bit segment, given by *S*

BEGIN

```
1.  for $n = 16; m = 4; i = 4$
2.      if $X[15] \,||\, X[14] \,||\, X[13] \,||\, X[12] = 1$
3.              $S = X[15]X[14]X[13]X[12]$;
4.      else if $X[11] \,||\, X[10] \,||\, X[9] \,||\, X[8] = 1$
5.              $S = X[11]X[10]X[9]X[8]$;
6.      else if $X[7] \,||\, X[6] \,||\, X[5] \,||\, X[4] = 1$
7.              $S = X[7]X[6]X[5]X[4]$;
8.      else
9.              $S = X[3]X[2]X[1]X[0]$;
10.     end if;
11. end for;
```

END

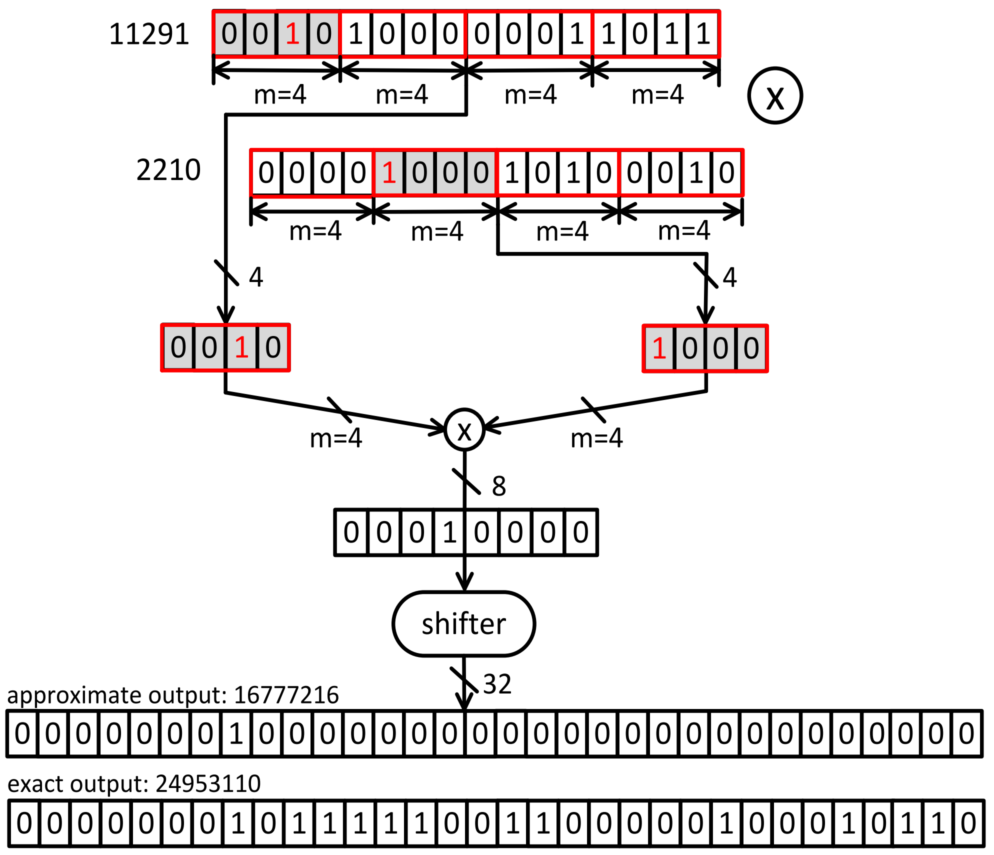


Figure 2.6 A numeric example of the16-bit SSM-based multiplier (n = 16, m = 4)

2.3.2 DSM-based Multiplier

The pseudo-code of Algorithm 2 details the DSM applied to select a 4-bit segment from a 16-bit operand. The DSM operates by detecting the leading one in the 16-bit number then extracting the following 4 bits and setting the least significant bit of the truncated values to one. Figure 2.7 shows a numerical example of a 16-bit DSM-based multiplier with the same inputs as implemented in the SSM-based multiplier. The truncated values are multiplied and shifted to the left to generate the final output. The DSM-based multiplier provides higher precision compared to the SSM-based multiplier as shown in Figure 2.7.

Algorithm 2: 4-bit Dynamic Segment Selection

INPUT: (i) *n*-bit operand, given by *XnXn-1…X0*

OUPUT: the selected *4*-bit segment, given by *S*

BEGIN

```
1.  for $i = 1$; $i < n$; $i$++
2.      if ( ($X[i] = 1$) && ($i >= 3$) )
3.              $S = X[i]X[i-1]X[i-2]1$;
4.      else if ( ($X[i] = 1$) && ($i < 3$) )
5.              $S = X[3]X[2]X[1]0$;
6.      end if;
7.  end for;
```

END

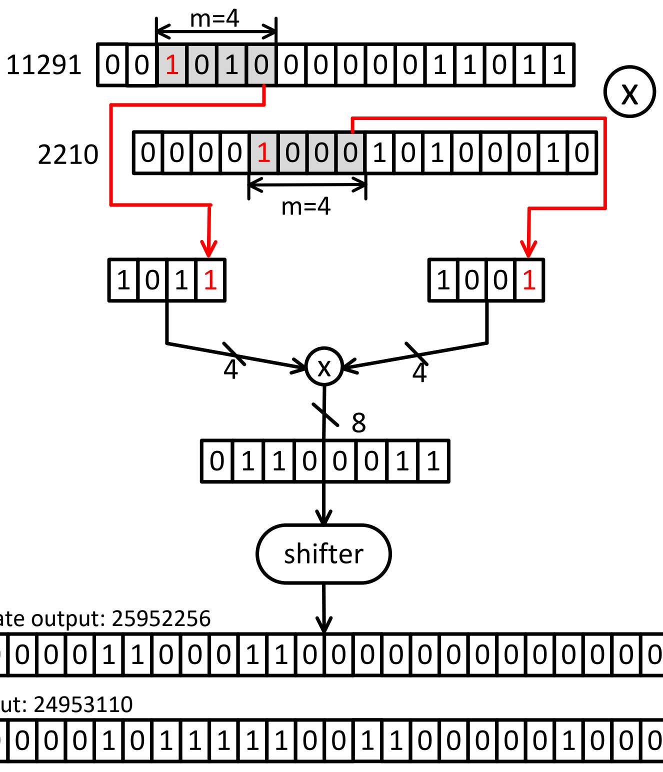


Figure 2.7 A numeric example of the16-bit DSM-based multiplier (n = 16, m = 4)

2.3.3 TOSAM

TOSAM [11] is the most recent segment-based approximate multiplier and it will be compared with the proposed dual approximate multiplier in Chapter 4. The input operands of TOSAM were truncated to two different sizes based on their leading-one-bit positions and rounded to perform smaller core multiplication and addition operations. TOSAM is appliable to both signed and unsigned multiplications. The following expression is used to perform the unsigned multiplication:

$$\boldsymbol{X} \times \boldsymbol{Y} \cong \boldsymbol{2^{k_A+k_B}} \times (\boldsymbol{X_r} \times \boldsymbol{Y_r} + \boldsymbol{1} + \boldsymbol{X_{m=7}} + \boldsymbol{Y_{m=7}}) \qquad (2.2)$$

where $\boldsymbol{k_A}$ and $\boldsymbol{k_B}$ denote the leading-one-bit position of each operand. $\boldsymbol{X_r}$ and $\boldsymbol{Y_r}$ are $\boldsymbol{k}$+1 bits segments and are generated by dynamic segment selectors. $\boldsymbol{X_{m=7}}$ and $\boldsymbol{Y_{m=7}}$ are the 7-bit truncated values of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Figure 2.8 shows an example of an unsigned 16-bit TOSAM using k=3 and m=7. The 16-bit TOSAM is estimated using two 16-bit dynamic segment selectors, a 4-bit accurate multiplier, adders and a shifter. As expected, the TOSAM is more complicated and costly compared to the SSM-based multiplier.

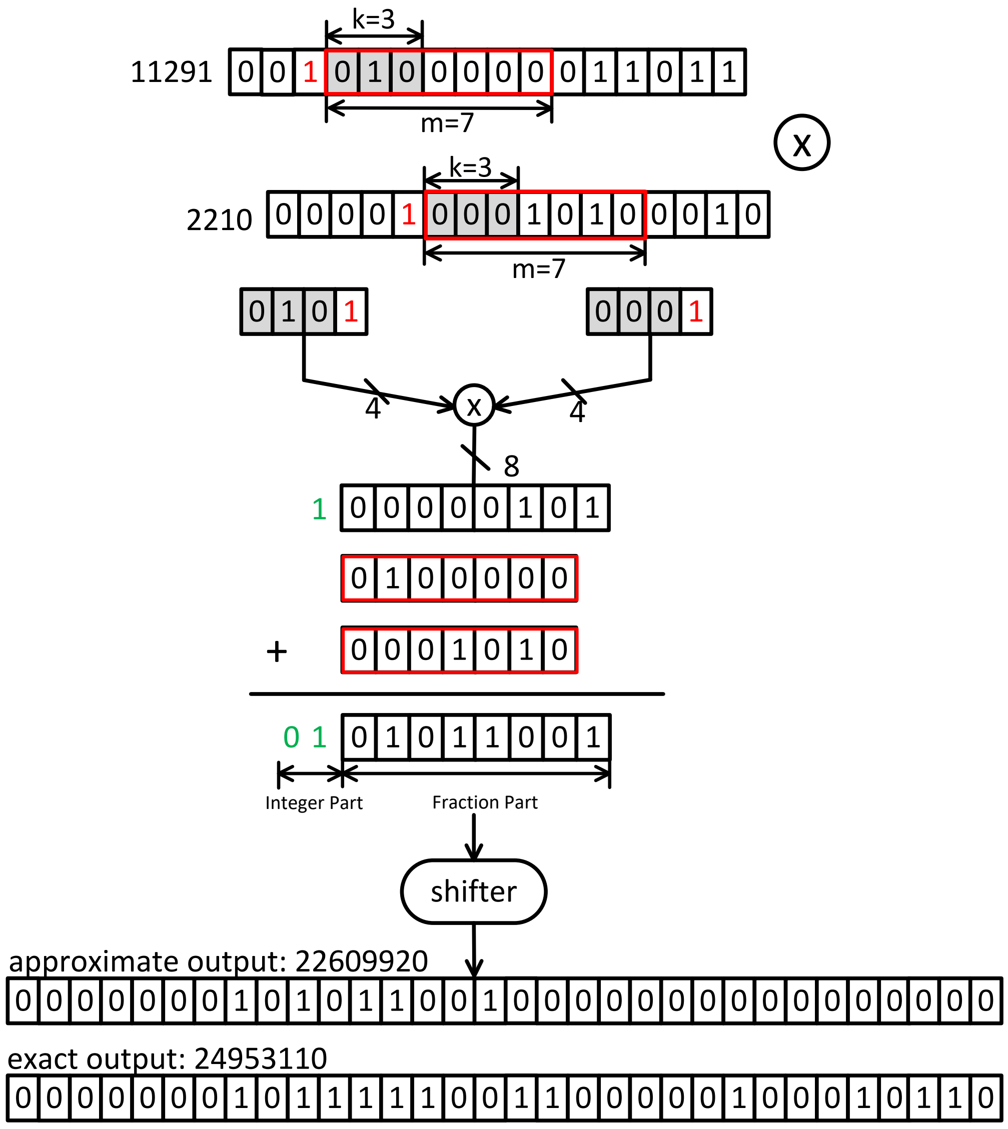


Figure 2.8 A numeric example of the16-bit TOSAM (k = 3, m = 7)

2.4 A Comparative Evaluation of Approximate Multiplier Designs

2.4.1 Error Characteristics

For establishing the error characteristics, approximate multipliers with the same n value generate the same output for the same inputs. The error distance (ED), mean error distance (MED), maximum error distance (MaxED), minimum error distance (MinED), mean relative error distance (MRED), maximum relative error distance (MaxRED) and minimum relative error distance (MinRED) are used as metrics to assess the error characteristics of approximate multipliers. For an n-bit approximate multiplier, the ED is defined as the absolute value of the difference between the accurate product ($\boldsymbol{P}$) and the approximate product ($\boldsymbol{Papp}$), i.e.,

$$\text{ED} = |\boldsymbol{Papp} - \boldsymbol{P}|, \tag{2.2}$$

The MED is the average of EDs for a set of outputs obtained by applying a set of inputs. The MaxED and MinED are defined as the maximum and minimum of the ED values among all possible inputs and outputs. The relative error distance (RED) is the ratio of ED over the accurate output, i.e.,

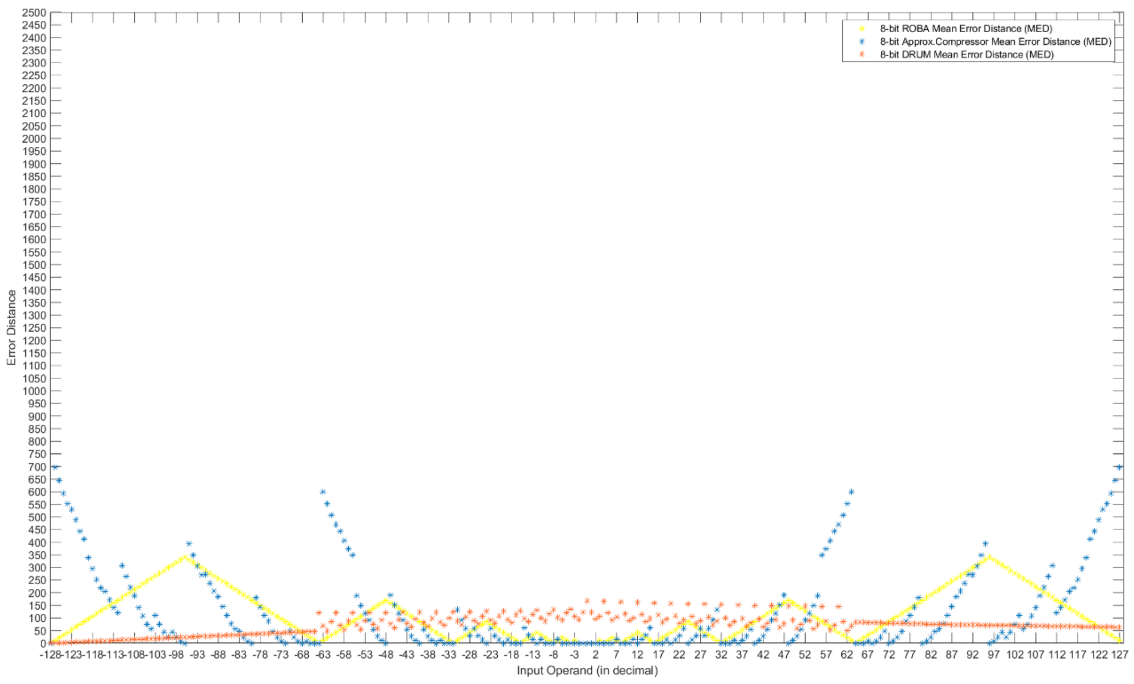
$$\text{RED} = \frac{|\boldsymbol{Papp} - \boldsymbol{P}|}{\boldsymbol{P}} = \frac{\boldsymbol{ED}}{\boldsymbol{P}}, \tag{2.3}$$

Similarly, the MRED, MaxRED and MinRED can be obtained.

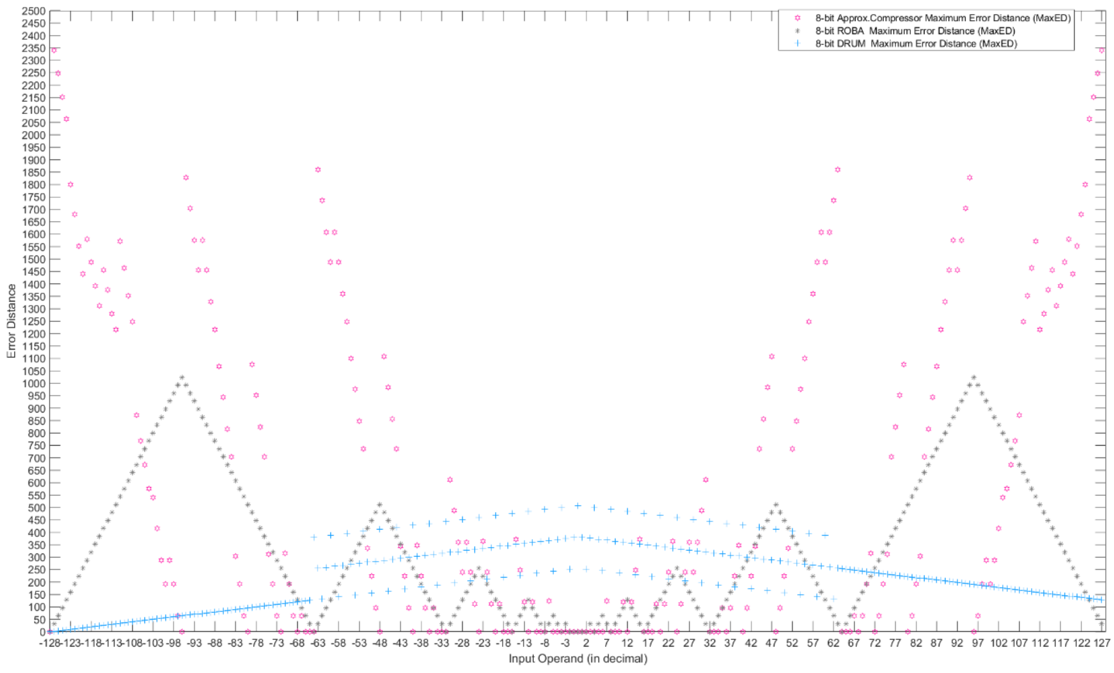
The 8-bit approximate multipliers are simulated by MATLAB with all possible combinations of operands and the MED, MaxED, MinED are obtained, as shown in

Figure 2.3. According to Figures 2.3.a and 2.3.b, the segment-based approximate multiplier has a relatively low MED and MaxED for large input operands. The segment-based multiplier also shows a stable MED and MaxED distribution with all possible approximate products. The approximate compressor-based multiplier has the largest MED and MaxED, as shown in Figure 2.3. and Figure 2.4., However, ROBA and the approximate compressor-based multiplier have a low MED and MaxED for small operands. As shown in Figure 2.5.a and Figure 2.6.a, ROBA has a relatively low MRED and MaxRED (up to 6% and 12%).

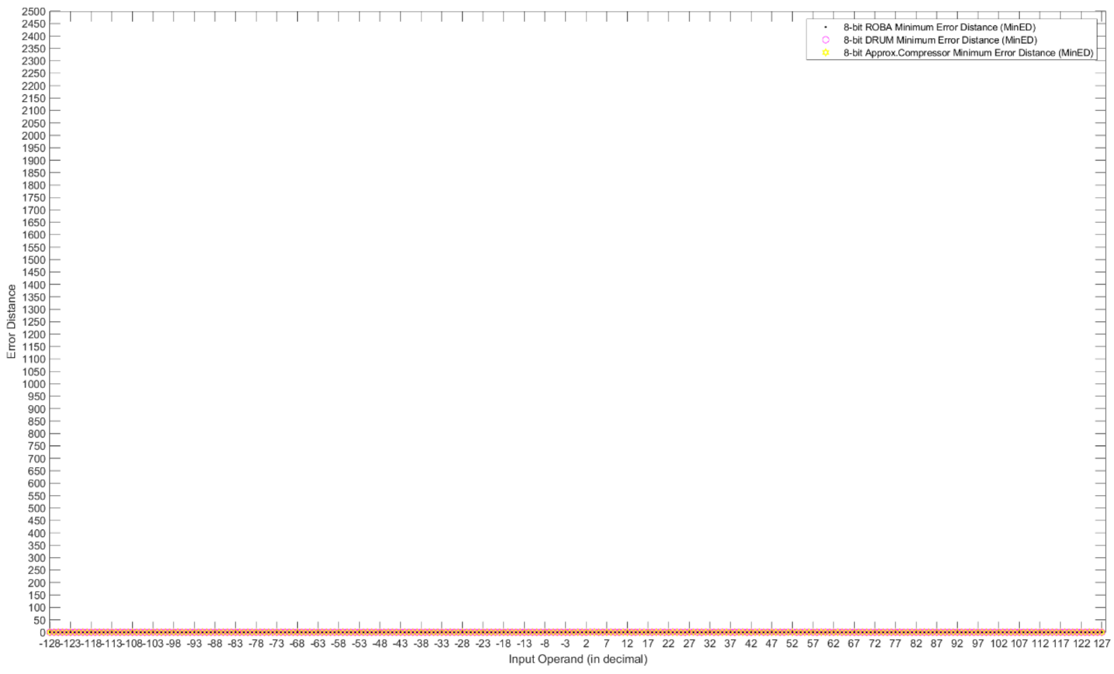
In summary, ROBA has the highest accuracy for small operands multiplication and segment-based approximate multiplier shows low MRED for large operands multiplication. The approximate compressor-based multiplier is not very accurate in terms of these metrics.



(a) MED

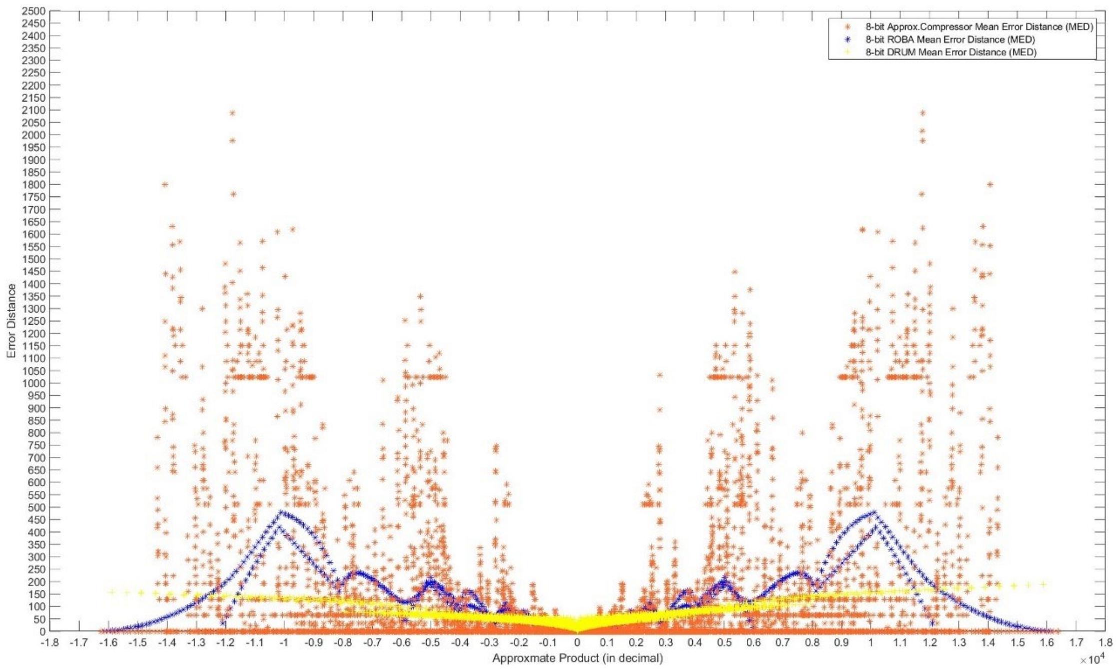


(b) MaxED

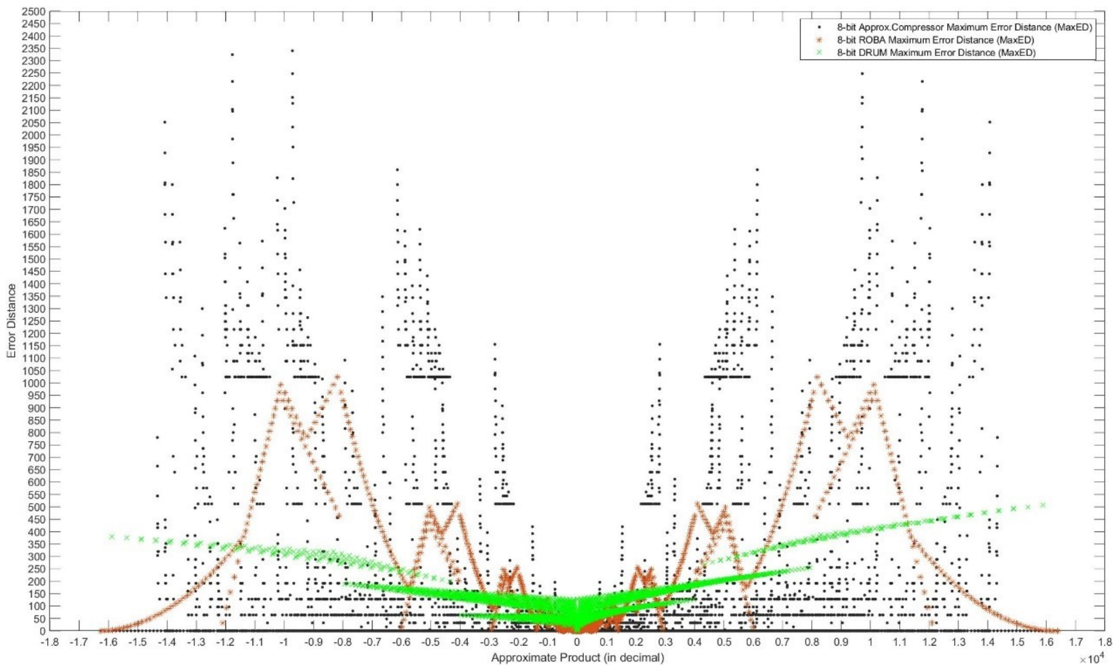


(c) MinED

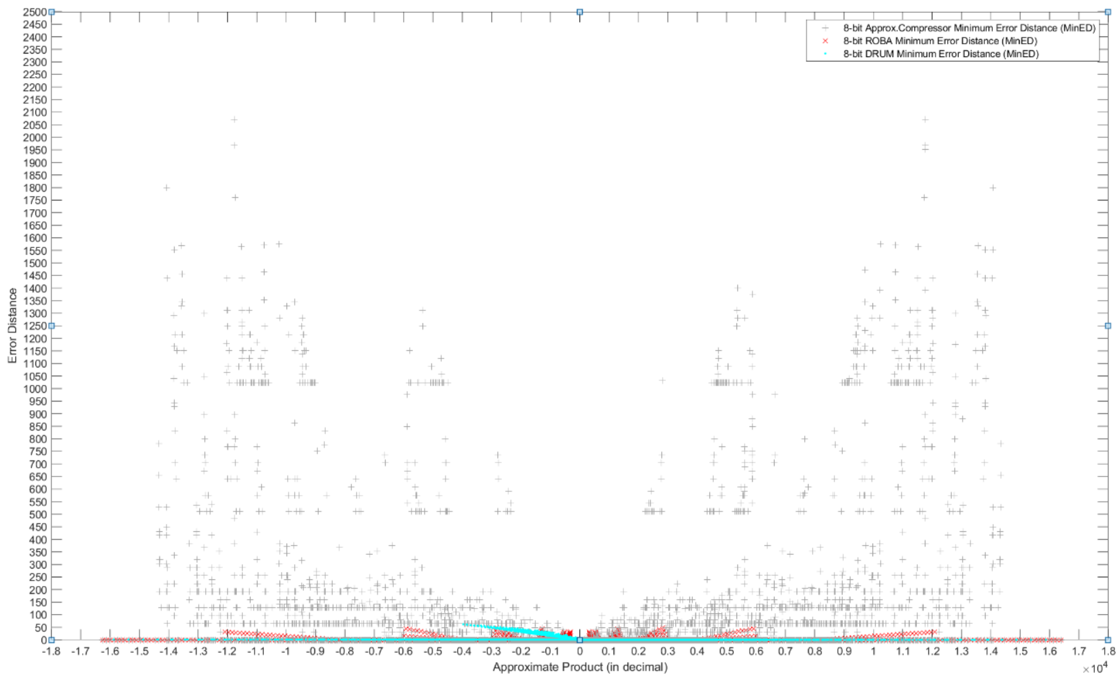
Figure 2.9 A comparison of (a) MED, (b) MaxED and (c) MinED of the 8-bit approximate multiplier designs with all possible input operands


8-bit Approx.Compressor Mean Error Distance (MED)
8-bit ROBA Mean Error Distance (MED)
8-bit DRUM Mean Error Distance (MED)
Error Distance
Approxmate Product (in decimal)
$\times 10^4$

(a) MED

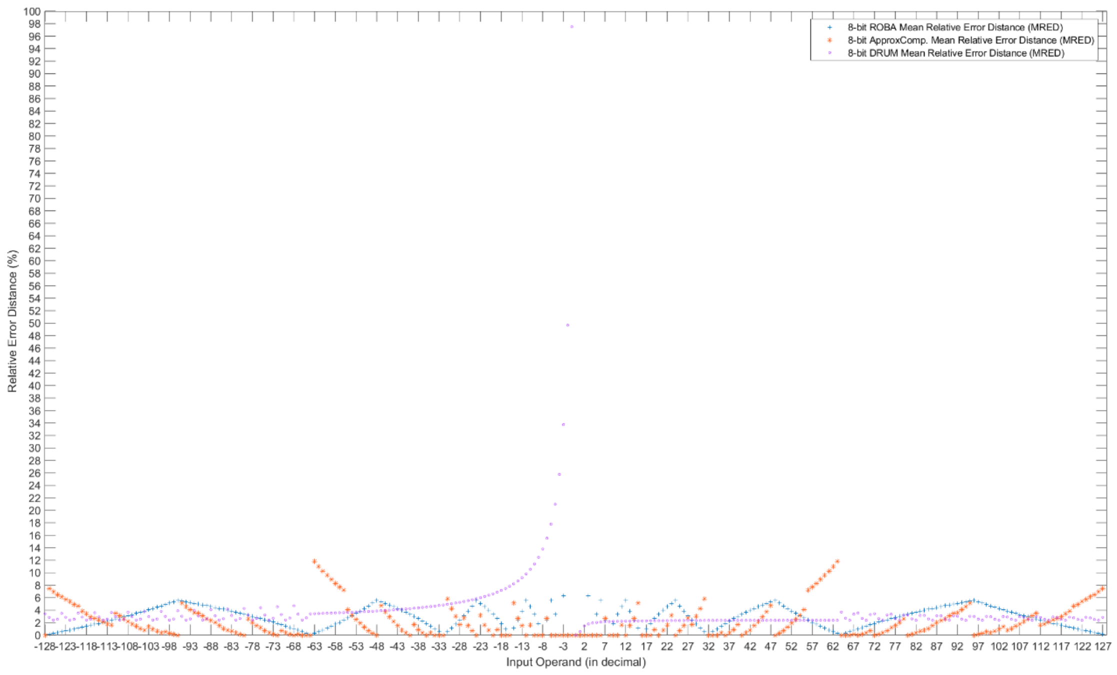

8-bit Approx.Compressor Maximum Error Distance (MaxED)
8-bit ROBA Maximum Error Distance (MaxED)
8-bit DRUM Maximum Error Distance (MaxED)
Error Distance
Approxmate Product (in decimal)
$\times 10^4$

(b)MaxED

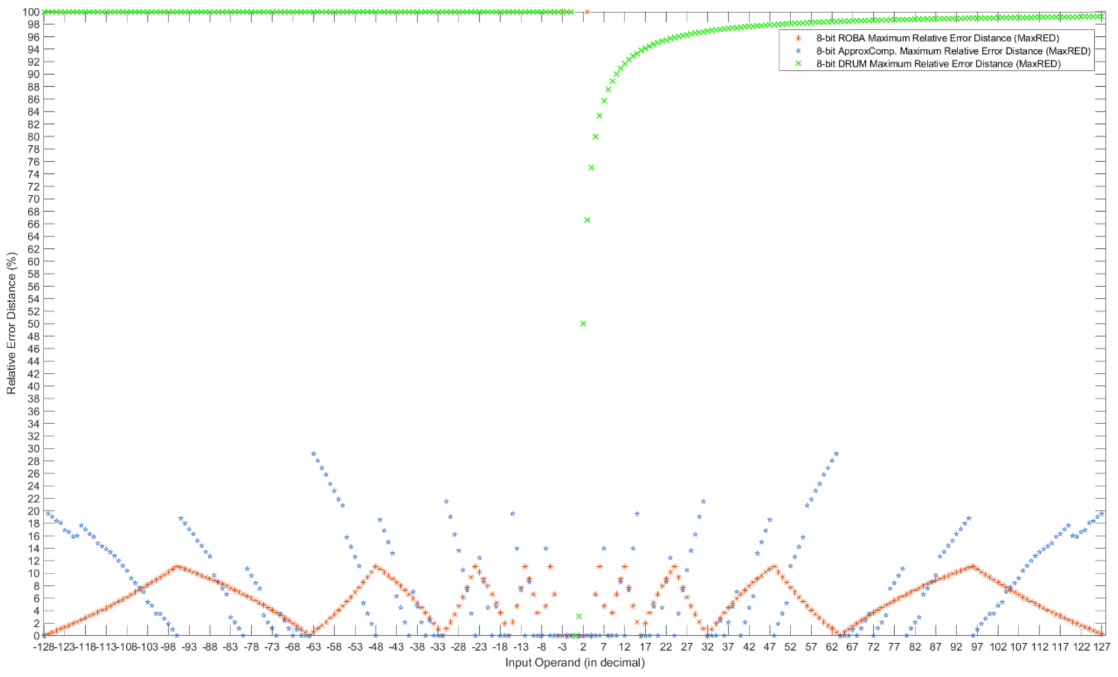


(c)MinED

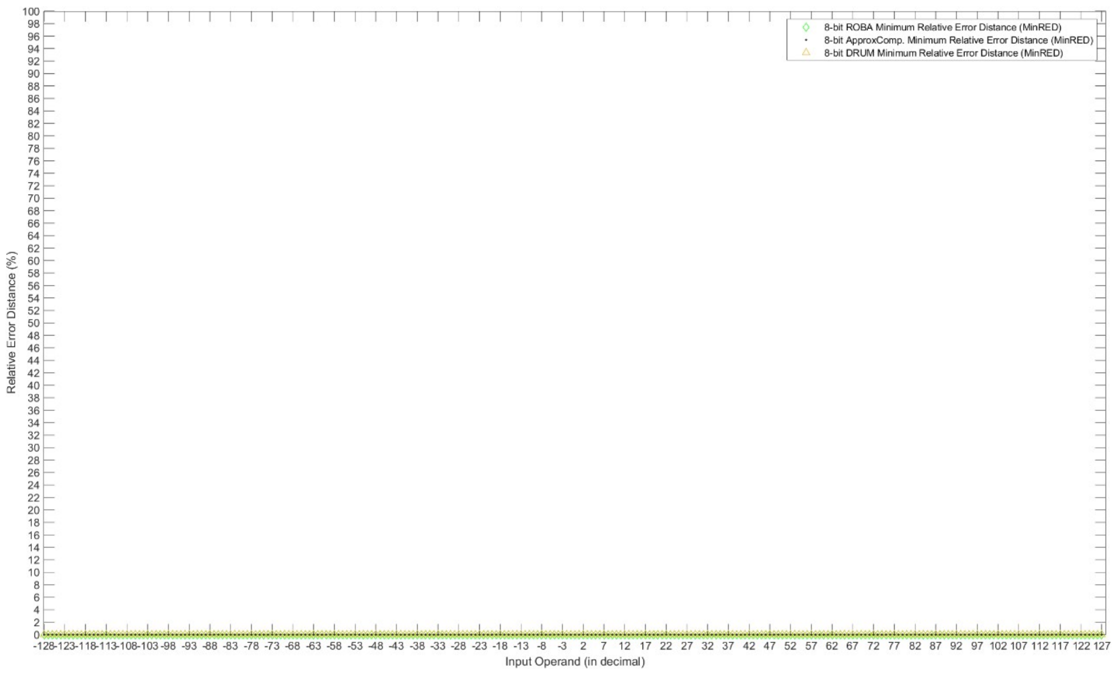
Figure 2.10 A comparison of (a) MED, (b) MaxED and (c) MinED of the 8-bit approximate multiplier designs with all possible input operands



(a) MRED

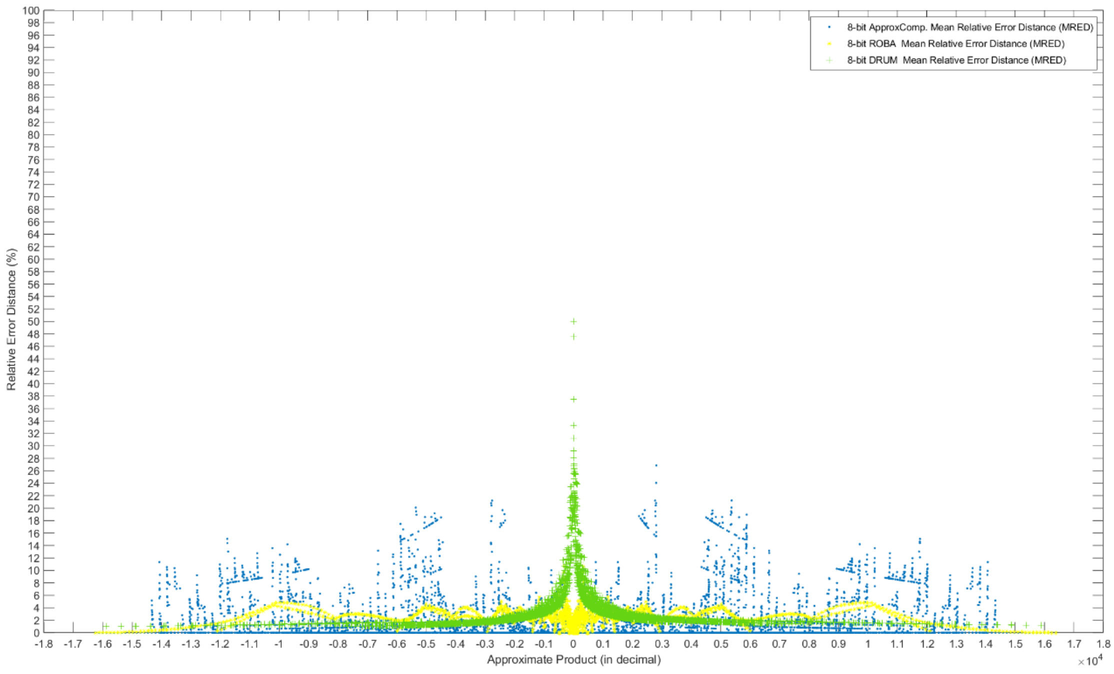


(b) MaxRED

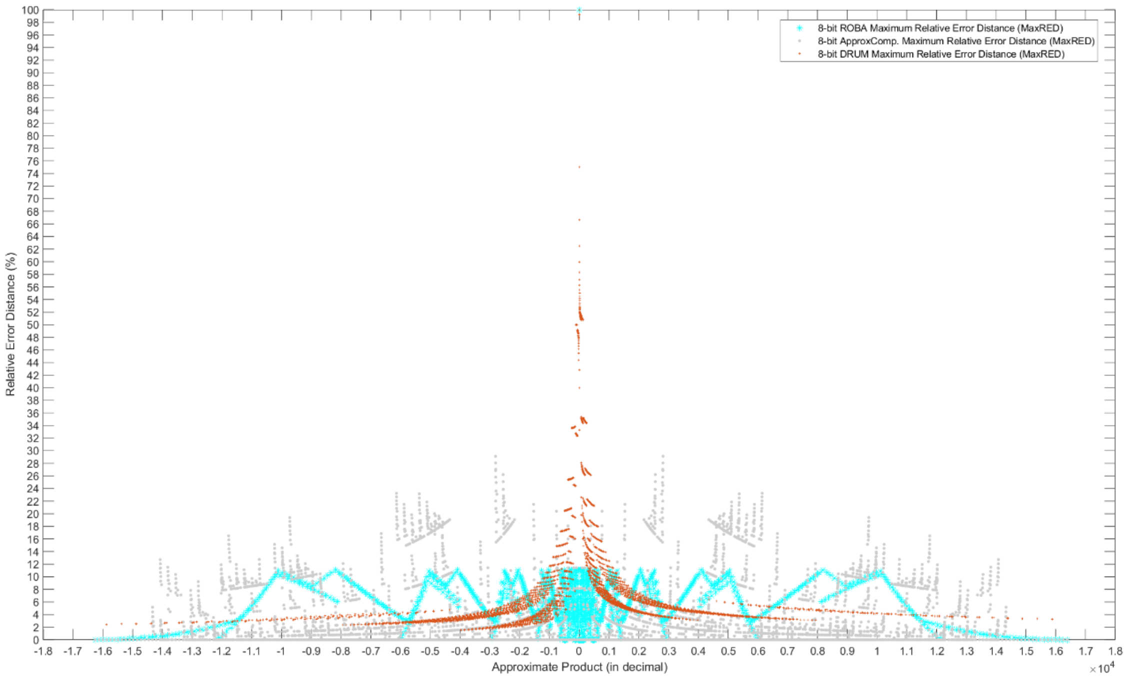


(c) MinRED

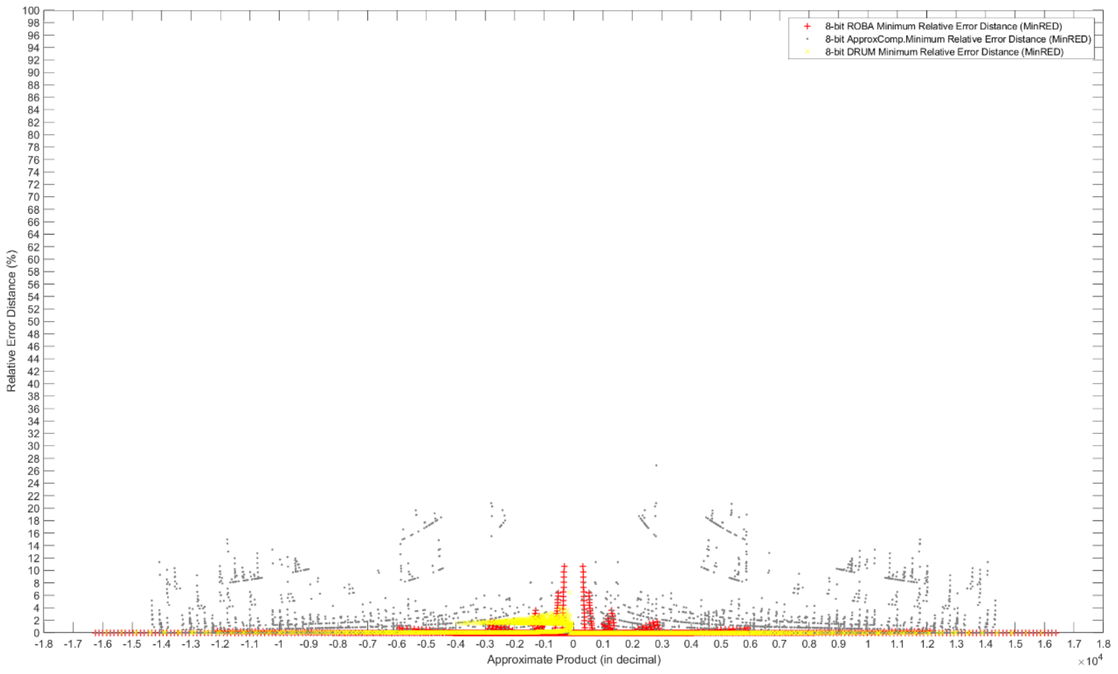
Figure 2.11 A comparison of (a) MRED, (b) MaxRED and (c) MinRED of the 8-bit approximate multiplier designs with all possible input operands



(a) MRED



(b) MaxRED



(c) MinRED

Figure 2.12 A comparison of (a) MRED, (b) MaxRED and (c) MinRED of the 8-bit approximate multiplier designs with all possible input operands

2.4.2 Circuit Characteristics

The considered 8-bit approximate multipliers and exact multiplier are implemented in Verilog HDL and synthesized using the Synopsys DC complier based on a CMOS TSMC 65 nm process. For a fair comparison, each multiplier is used to design an 8-bit multiplier.

As can be seen in Table 2.1, there are significant improvements in all considered approximate multiplier designs. For example, the segment-based approximate multiplier improves delay, power and area by up to 30%, 42% and 42.3%, respectively, compared to an exact booth multiplier. Simulation has shown that the new approximate compressor-based multiplier has the shortest critical path delay, the smallest area and significantly lower power consumption compared to ROBA and the segment-based multiplier.

Table 2.2 A comparison of delay, power and area of the approximate multiplier designs

Design *n*=**8**	**Delay (*ns*)**	**Power (μW)**	**Area (μm^2)**
Exact	2.4	73.3	942.5
ROBA	2.1	32.8	712.8
DSM-based (*m*=6)	1.7	42.4	543.6
Approximate Compressor-based	1.5	25.1	486.7

2.5 Conclusion

Among the considered approximate multipliers, the approximate compressor-based multiplier has small values for delay, area and power dissipation. However, it has large MED and MRED. The ROBA has a poor accuracy for some input operands (i.e., [17, 29], [37, 57], [72, 112] and [-113, -78]) and moderate hardware consumption. The segment-based shows a low accuracy for small input operands (i.e., [-33, 0]), but a relatively low circuit overhead. Also, only the segment-based approximate multiplier allows for a trade-off between accuracy and performance by adjusting segment size *m*.

Chapter 3
Dual Segmentation Approximate Multiplier

This chapter presented a new design for a dual segmentation approximate multiplier [12]. The proposed multiplier relies on using static segmentation initially followed by a dynamic segmentation. Using only the dynamic segment selector leads to reduced performance gains. Therefore, the proposed multiplier uses a mixture of the SSM and DSM approaches. The static segment selector truncates the n-bit input operands to $n/2$-bit segments or even $n/4$-bit segments, which can efficiently reduce the hardware overhead of the LODs. The efficiencies of the proposed multipliers as 16-bit and 32-bit designs were evaluated and compared against an exact multiplier and previously proposed segment-based approximate multipliers. The proposed design achieves a better performance-accuracy trade-off compared to previously proposed segment-based approximate multipliers.

3.1 Dual Segmentation Approximate Multiplier Implementation

In the proposed design, dual segmentation design, the static segment selector truncates the n-bit inputs to the k-bit segments as shown in Figure 3.1. Using this design, the dynamic segment selector of the DSM-based multiplier only needs to identify the leading one bit from the *k*-bit segment instead of the *n*-bit operand. After selecting the *m*-bit segments that contain the leading one bit of each *k*-bit segment, the chosen segments are applied to an *m*×*m* multiplier. The result of this multiplication is shifted according to the positions of the leading one bit of each *n*-bit operand to generate the final output. The main benefit of the dual segmentation logic is that the size of the most power-consuming components, LODs, is from *n*-bit down to *k*-bit for dynamic segment selectors.

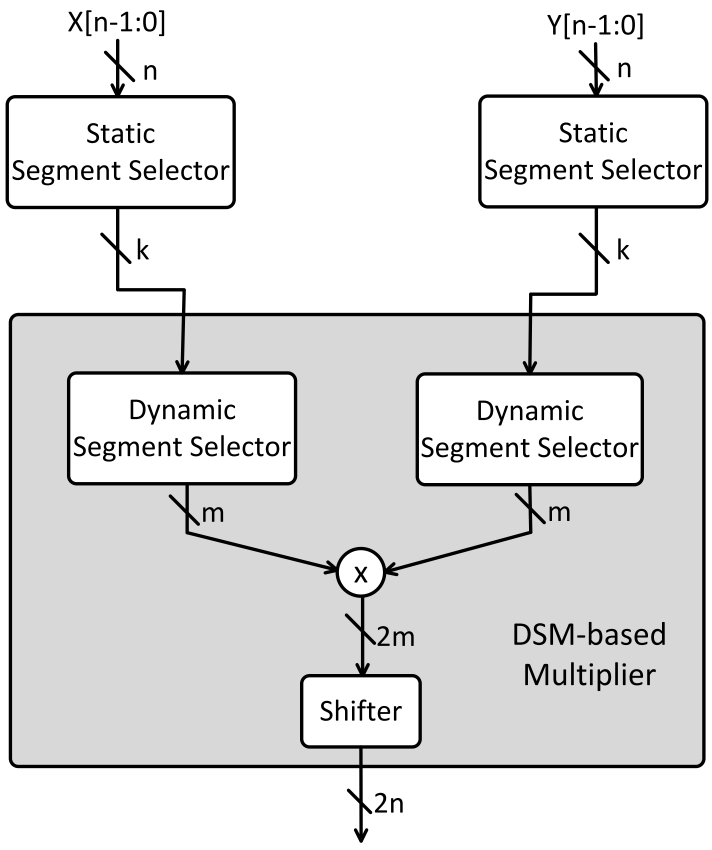


Figure 3.1 Dual segmentation approximate multiplier architecture

Figure 3.2 shows a general example of the *m*-bit static segment approximation. The *n*-bit number is split into three *m*-bit segments. The first m-bit segment starting from the most significant bit are all zeros, then the second *m*-bit segment which contains the leading one bit is chosen.

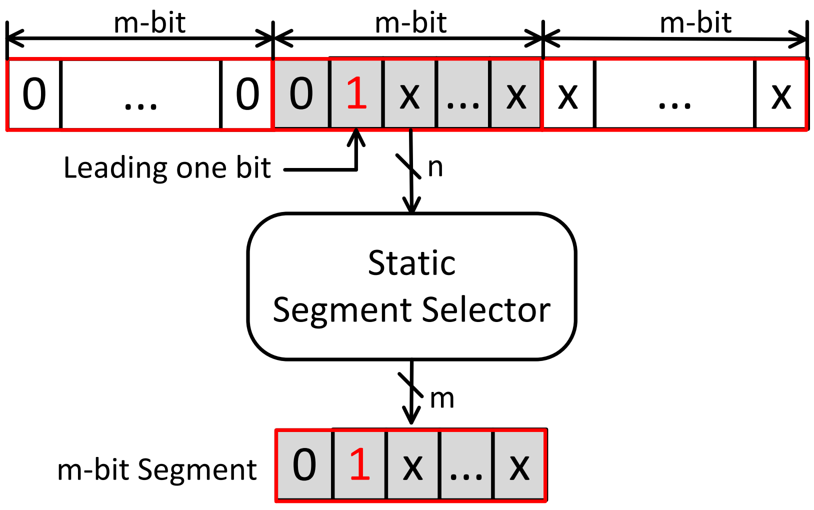
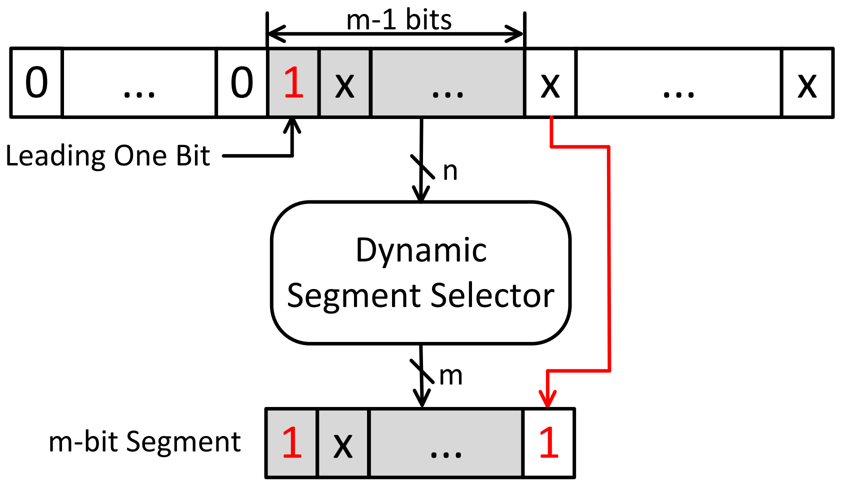
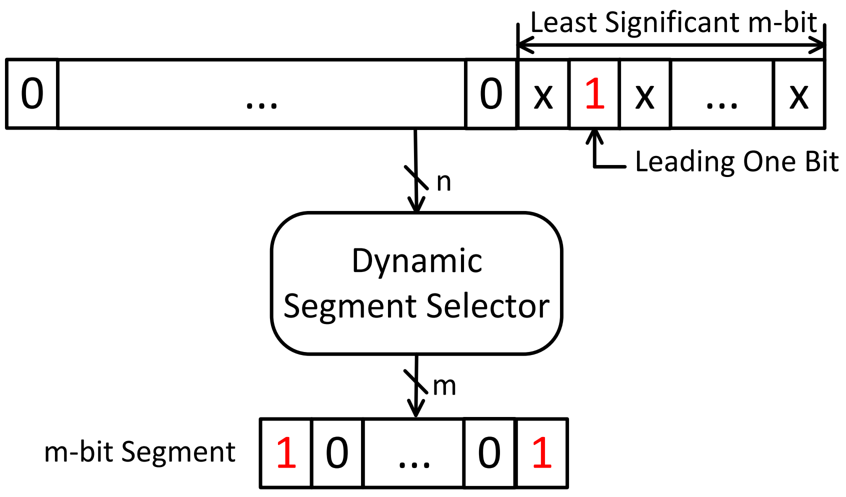


Figure 3.2 A general example of the *m*-bit static segment approximation

As can be seen in Figure 3.3.a and Figure 3.4.a, the dynamic segment selector captures *m*-2 bits segment in a number starting from the leading one bit and sets the least significant bit of the segment to one when the leading one bit is not in the least significant *m*-bit. In the case where the leading one bit is within the least significant *m*-bit of a number, then the most significant bit and the least significant bit of the *m*-bit segment are set as '1', and the bits between them are approximated by zeros (see Figure 3.3.b and Figure 3.4.b). Thus, the *m*-bit segment of the proposed selector and the dynamic segment selector always have the same most and least significant bits. Also, the dynamic segment selector and the proposed selector can produce the same *m*-bit segment when the leading one bit is not within the least *m*-bit of the *k*-bit segment. As expected, the proposed multiplier has similar accuracy as the DSM-based multiplier.



(a)

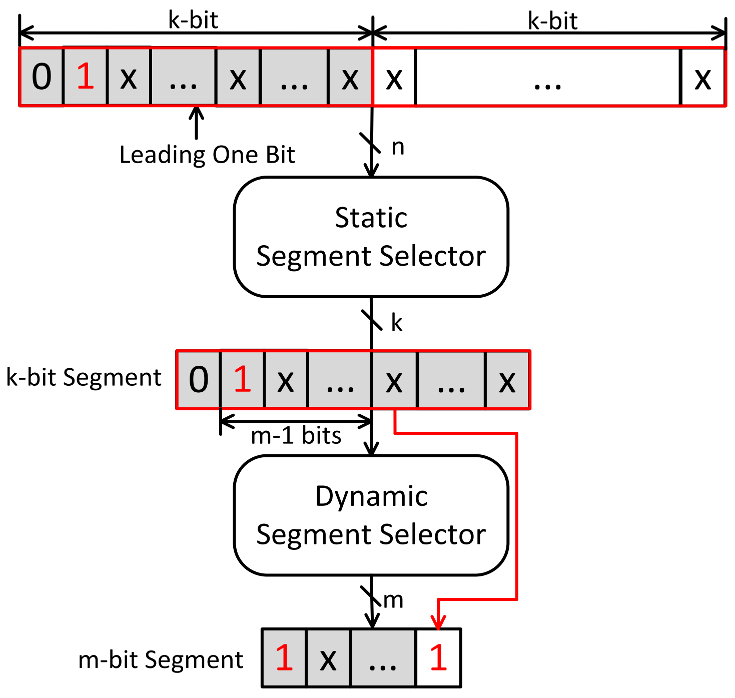


(b)

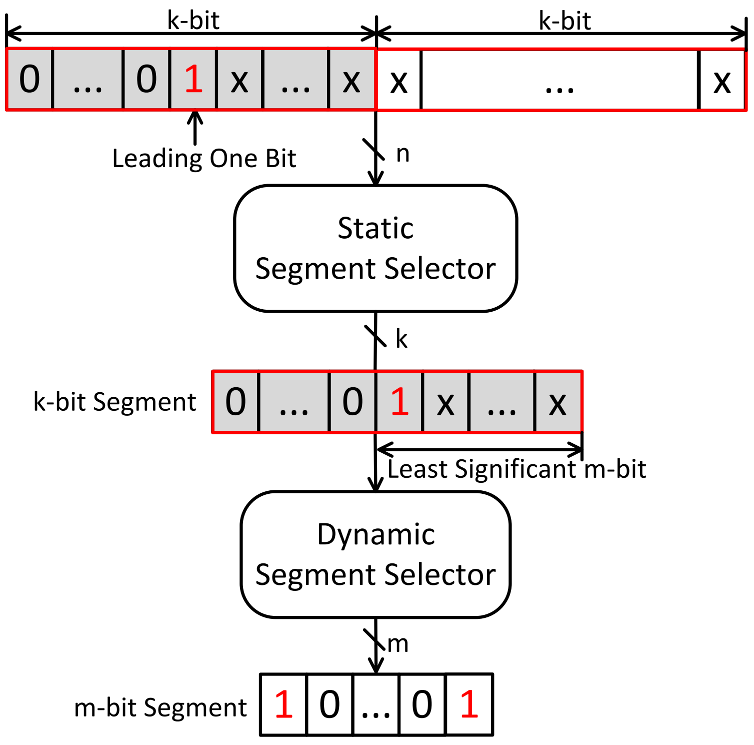
Figure 3.3 A general example of the *m*-bit dynamic segment approximation

(a) The leading one bit is not within the least significant *m*-bit

(b) The leading one bit is within the least significant *m*-bit



(a)



(b)

Figure 3.4 A general example of the *m*-bit dual segment approximation

(a) The leading one bit is not within the least significant *m*-bit

(b) The leading one bit is within the least significant *m*-bit

Figure 3.5 shows an example of a 16-bit dual approximate multiplication using *k*=8 and *m*=4. As can be seen in the figure, the result of the proposed multiplication is the same as that of the DSM-based approximate multiplication. This is because the leading one bit of inputs is not within the least *m*-bit of the *k*-bit segment. Thus, the same *m*-bit segments are captured in the proposed and DSM-based multipliers.

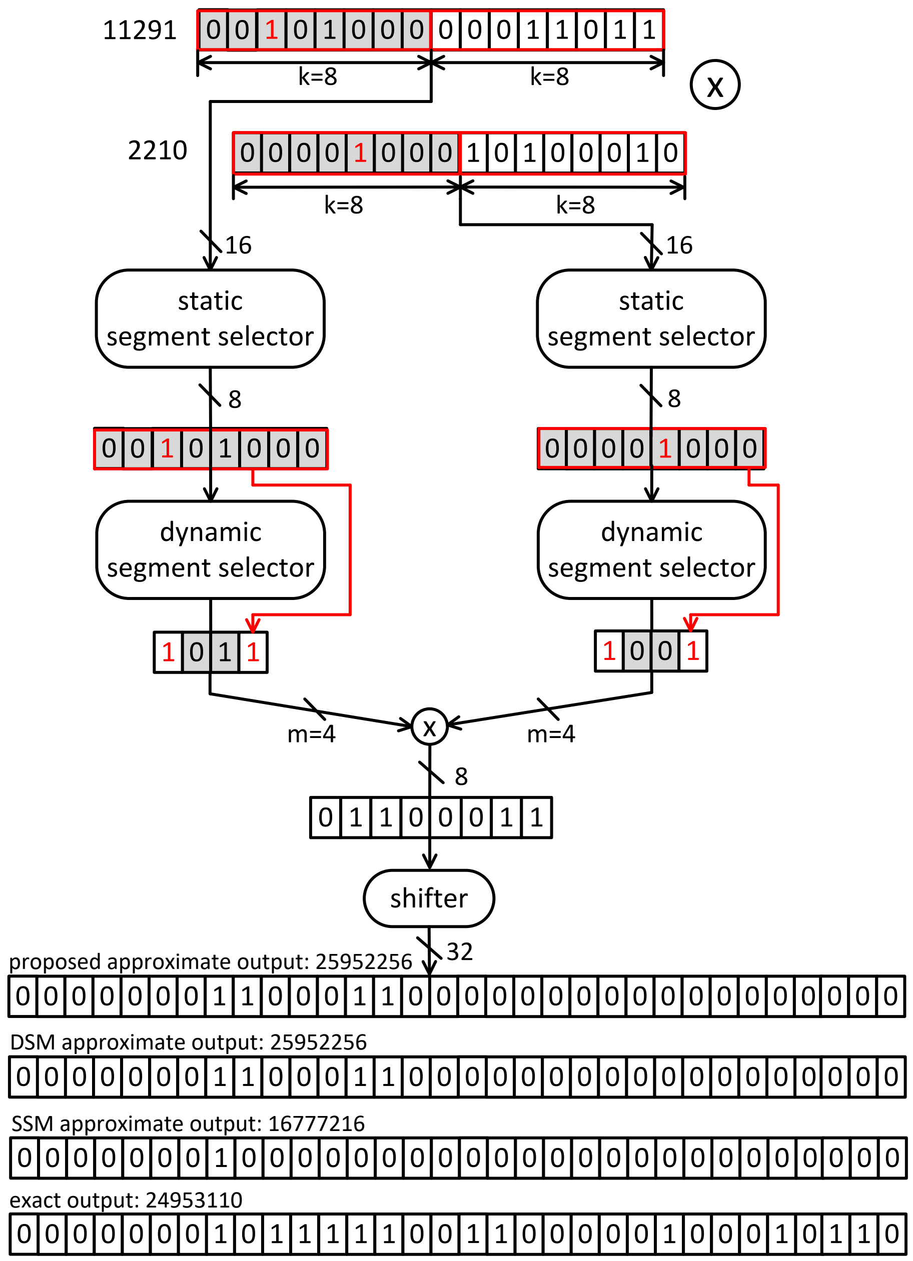


Figure 3.5 Example of 16-bit dual segmentation multiplier (*k*=8, *m*=4)

3.2 Simulation and Synthesis Results

In this section, the proposed design is compared with an exact multiplier and three previously proposed segment-based approximate multipliers. All designs were implemented using Verilog HDL and synthesized using Synopsys DC complier in a 65nm library at the typical process corner. All designs are unsigned multipliers. To evaluate the impact of changing the multiplier size, each multiplier is exploited to design 16-bit and 32-bit multipliers, and we fix k=8 and m=4. The accuracy is evaluated by using the MRE for 16-bit and 32-bit multiplier designs with ten million uniformly distributed random input pairs.

The results show that the SSM-based multiplier has the lowest delay, energy, area, energy-delay product (EDP) and power-delay-area (PDA) product, but it has the worst MRE. The DSM-based multiplier provides the highest accuracy. The proposed multiplier outperforms the DSM-based multiplier in terms of the delay, area and energy consumption while having almost the same accuracy level as the DSM-based multiplier.

The delay, energy, and EDP of the 16-bit proposed multiplier are approximately 19%, 17% and 27% lower than those of the DSM-based multiplier. Also, the 16-bit proposed multiplier has 21% smaller area compared to the DSM-based multiplier. As can be seen in Table 3.1, the MRE value of the 16-bit proposed multiplier is about 11% lower than the 16-bit SSM. Compared to the DSM approach, the proposed design can reduce the energy by 37.90% for 32-bit multipliers. The 32-bit proposed multiplier improves the speed, area, and energy up to 58%, 92%, and 98% compared to the exact multiplier as can be seen in Table 3.2. Additionally, the MRE value of the 32-bit proposed design is 18% lower than that of the 32-bit SSM-based multiplier. The TOSAM has the worst performance among all approximate multipliers.

Table 3.1 Circuit's parameter of unsigned 16-bit exact and approximate multipliers

Design k**=8** m**=4**	**Delay (*ns*)**	**Power (*µW*)**	**Area (*µm*^{2})**	**Energy (*fJ*)**	**EDP (*ns*×*fJ*)**	**PDA (*fJ*× *µm*^{2})**	**MRE (%)**
Exact	2.0	256.7	2626	518.6	1047.5	1361776	-
Proposed	1.3	34.6	676	48.1	66.9	32558	5.8
DSM	1.6	36.6	854	57.8	91.3	49358	5.7
SSM	0.9	15.9	395	14.3	12.9	5652	16.9
TOSAM (2,4)	1.6	37.9	854	60.7	97.1	51800	11.4

Table 3.2 Circuit's parameter of unsigned 32-bit exact and approximate multipliers

Design k**=8** m**=4**	**Delay (*ns*)**	**Power (*µW*)**	**Area (*µm*^{2})**	**Energy (*fJ*)**	**EDP (*ns*×*fJ*)**	**PDA (*fJ*× *µm*^{2})**	**MRE (%)**
Exact	3.8	1343.1	10652	5036.8	18887.9	53651727	-
Proposed	1.5	38.3	894	60.1	94.5	53813	6.8
DSM	2.2	44.9	1519	96.9	209.4	147220	6.7
SSM	1.0	17.9	629	17.6	17.2	11040	24.5
TOSAM (2,4)	2.3	45.7	1487	103.8	235.7	154394	12.3

Chapter 4
A Merged Approximate Multiplier with Two Precisions

This chapter presents the design of a merged approximate multiplier with two precisions. Compared to the single-precise design, the proposed design has a longer delay. However, it can achieve better performance in terms of area and power. This design can be used in a system that has multiple approximate multipliers-based CNN accelerators with different precisions, or in a single CNN accelerator built with precision reconfigurable approximate multipliers. This hybrid approximate multiplier can achieve a better performance-accuracy trade-off compared to the single-precision multiplier. The results of this chapter have been published in [14].

4.1 Proposed Approximate Multiplier Implementation

In this section, the design of a segment-based hybrid approximate multiplier is presented. The input operands are represented in sign and magnitude format, as illustrated in Figure 4.1. The most significant bit (MSB), X_{n-1}, is used as the sign bit and the remaining bits in the number are used to represent the magnitude of the binary number in the usual unsigned binary number format way.

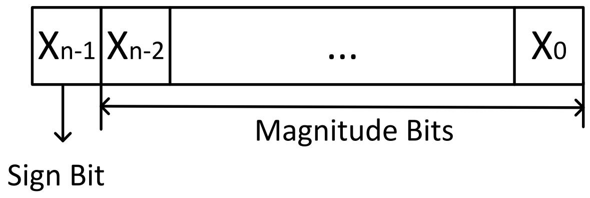


Figure 4.1 The sign and magnitude format of the input operands

In the proposed design, a 2-to-1 MUX is used as a mode selector to switch between the low precision multiplier (*m*=4) and high precision multiplier (*m*=8) (see Figure 4.2). When *Sel* is 0, the low precision mode (*m*=4) is activated. When *Sel* is 1, the multiplier is in the high precision mode (*m*=8). Using this design either the high or the low precision approximate multiplier will be power on or off based on the controller signal '*CN*'.

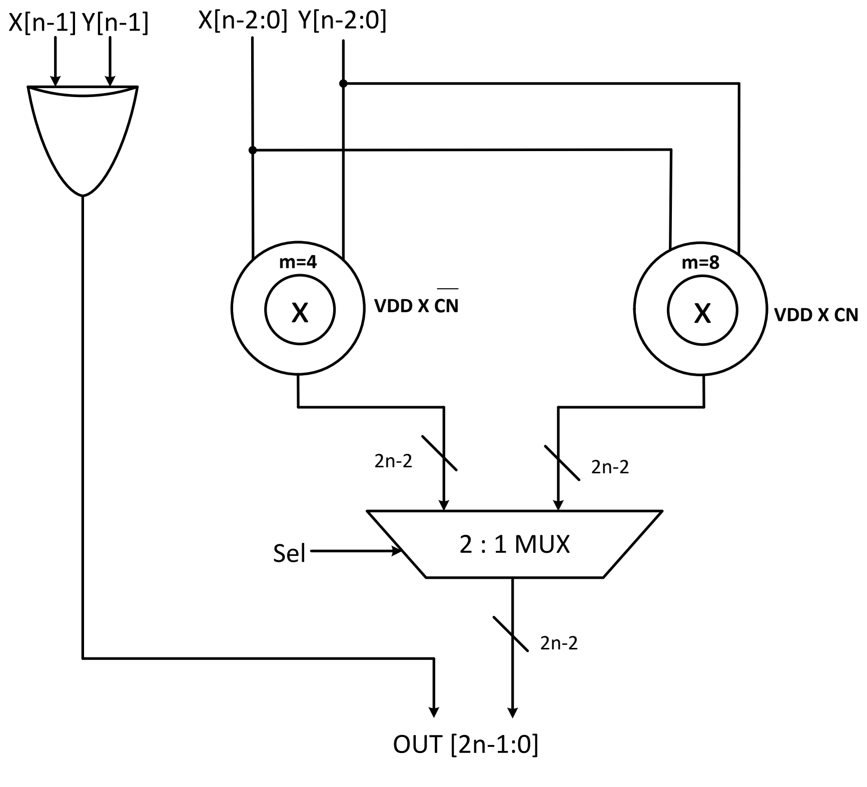


Figure 4.2 The schematic for mode selection

The larger core accurate multiplier is efficiently built from the smaller accurate multiplier. Figure 4.3 shows an example of an 8-bit accurate multiplier is built by four 4-bit accurate multipliers, where $\boldsymbol{X_H}$, $\boldsymbol{Y_H}$, $\boldsymbol{X_L}$ and $\boldsymbol{Y_L}$ are the upper and lower 4-bit segments of the inputs. Each row with eight dots represents the 8-bit product of the two 4-bit segments. Then the exact compressors are used in the partial product matrix to calculate the final result of the 8-bit accurate multiplier.

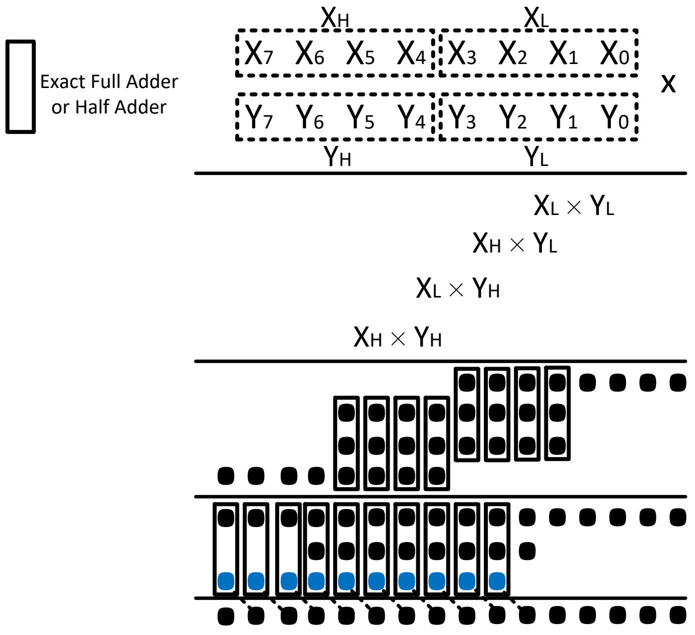


Figure 4.3 Building an 8-bit multiplier from four 4-bit multipliers

As can be seen from Figure 4.5, the 16-bit SSM-based hybrid approximate multiplier in mode (4 & 8) only has four 4-bit accurate multipliers and the three of them will be turned off when the low precision mode (m=4) is activated. The area is expected to be reduced. The two 2-to-1 MUXs are used to select the 4-bit segment for each 4-bit accurate multiplier. When *Sel 1* = *Sel 2* =0, the low precision mode (m=4) is activated and the MUXs select the 4-bit segment (*A'* and *B'*) truncated from the low-precision static segment selector. When *Sel 1* = *Sel 2* =1, the high precision mode (*m*=8) is activated and the two MUXs select the upper 4-bit of the inputs ($\boldsymbol{A_H}$, $\boldsymbol{B_H}$) while feeding them to one of the 4-bit accurate multipliers. The three other multipliers are used to calculate $\boldsymbol{A_L} \times \boldsymbol{B_L}$, $\boldsymbol{A_H} \times \boldsymbol{B_L}$ and $\boldsymbol{A_L} \times \boldsymbol{B_H}$. Finally, using the method described in Figure 4.3 to accumulate the partial products to generate the final product of the 8-bit multiplier.

Figure 4.6 shows the mode (6 & 8) of the 16-bit DSM-based hybrid approximate multiplier. The 8-bit accurate multiplier is built by one 2-bit multiplier, two 2×6 multipliers and one 6-bit multiplier, as illustrated in Figure 4.4. The 2-bit multiplier and two 2×6 multipliers will be turned off when the low precision mode (*m*=6) is activated. Similarly, Figure 4.7 shows that the 16-bit DSM-based hybrid approximate multiplier in mode (4 & 6) is built by one 2-bit multiplier, two 2×4 multipliers and one 4-bit multiplier.

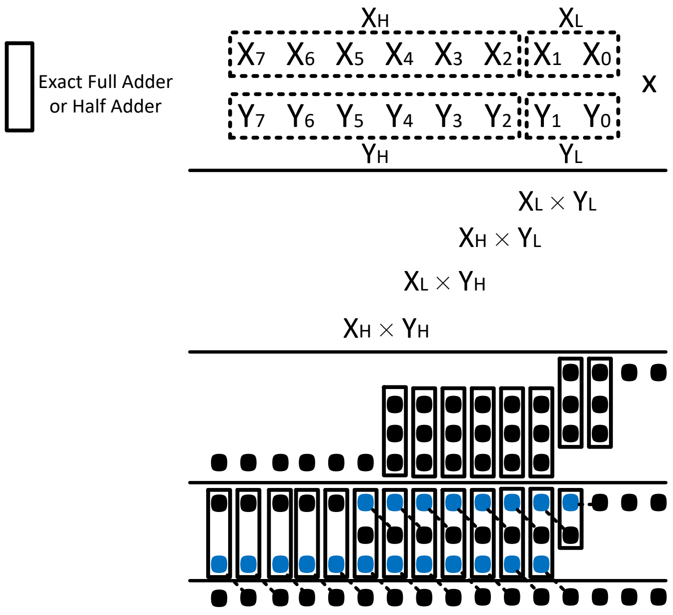


Figure 4.4 Building an 8-bit multiplier from 2-bit, 2×6 and 6-bit multipliers

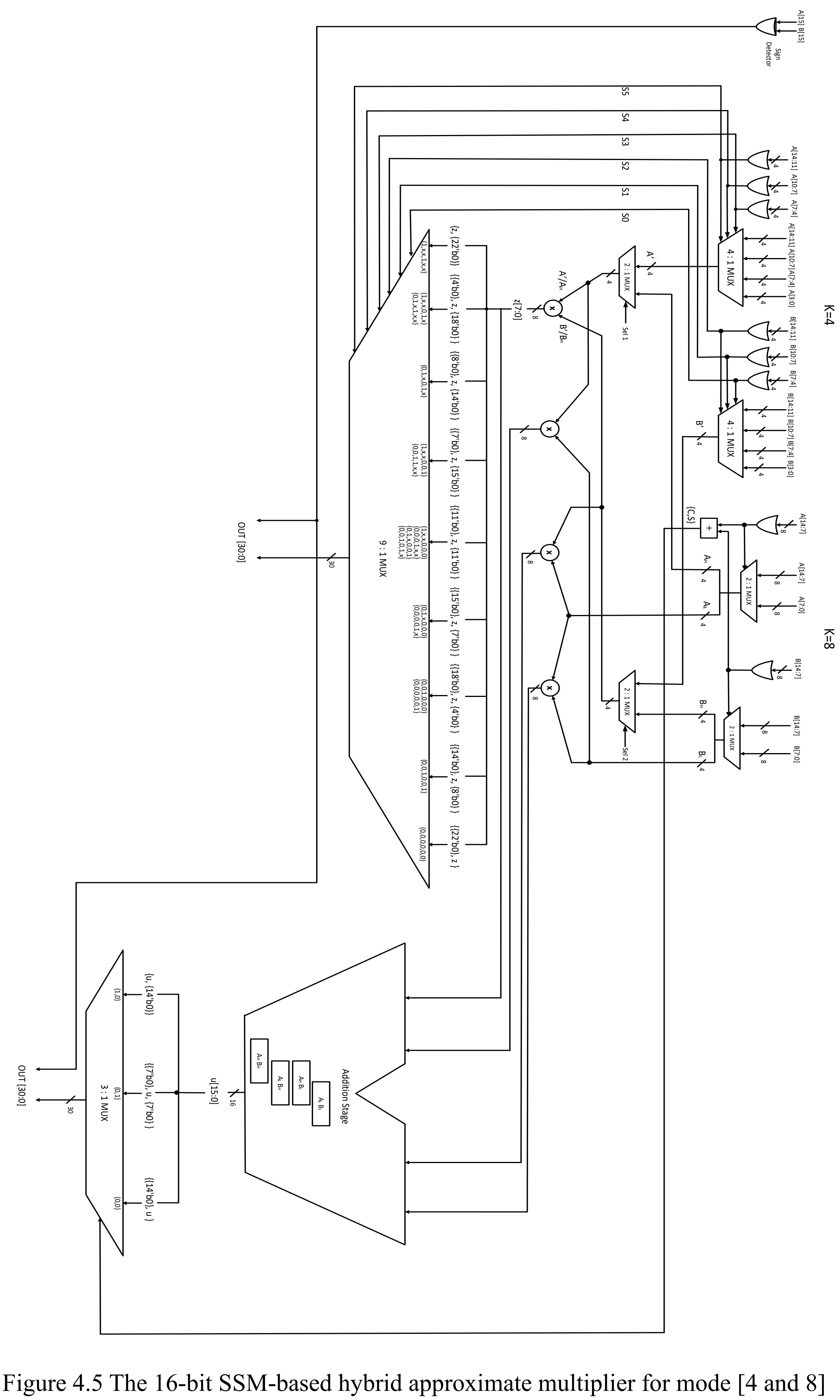


Figure 4.5 The 16-bit SSM-based hybrid approximate multiplier for mode [4 and 8]

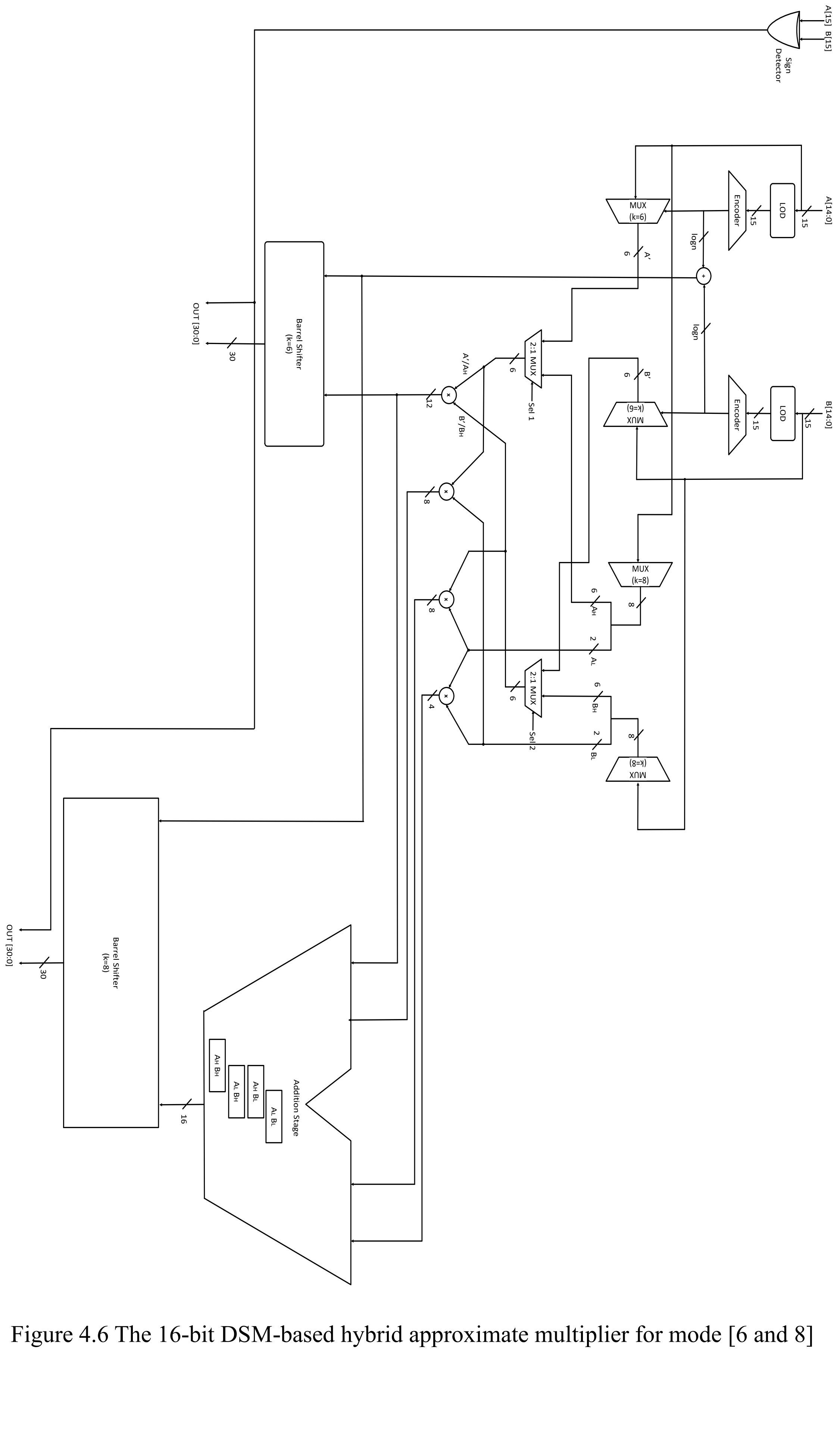


Figure 4.6 The 16-bit DSM-based hybrid approximate multiplier for mode [6 and 8]

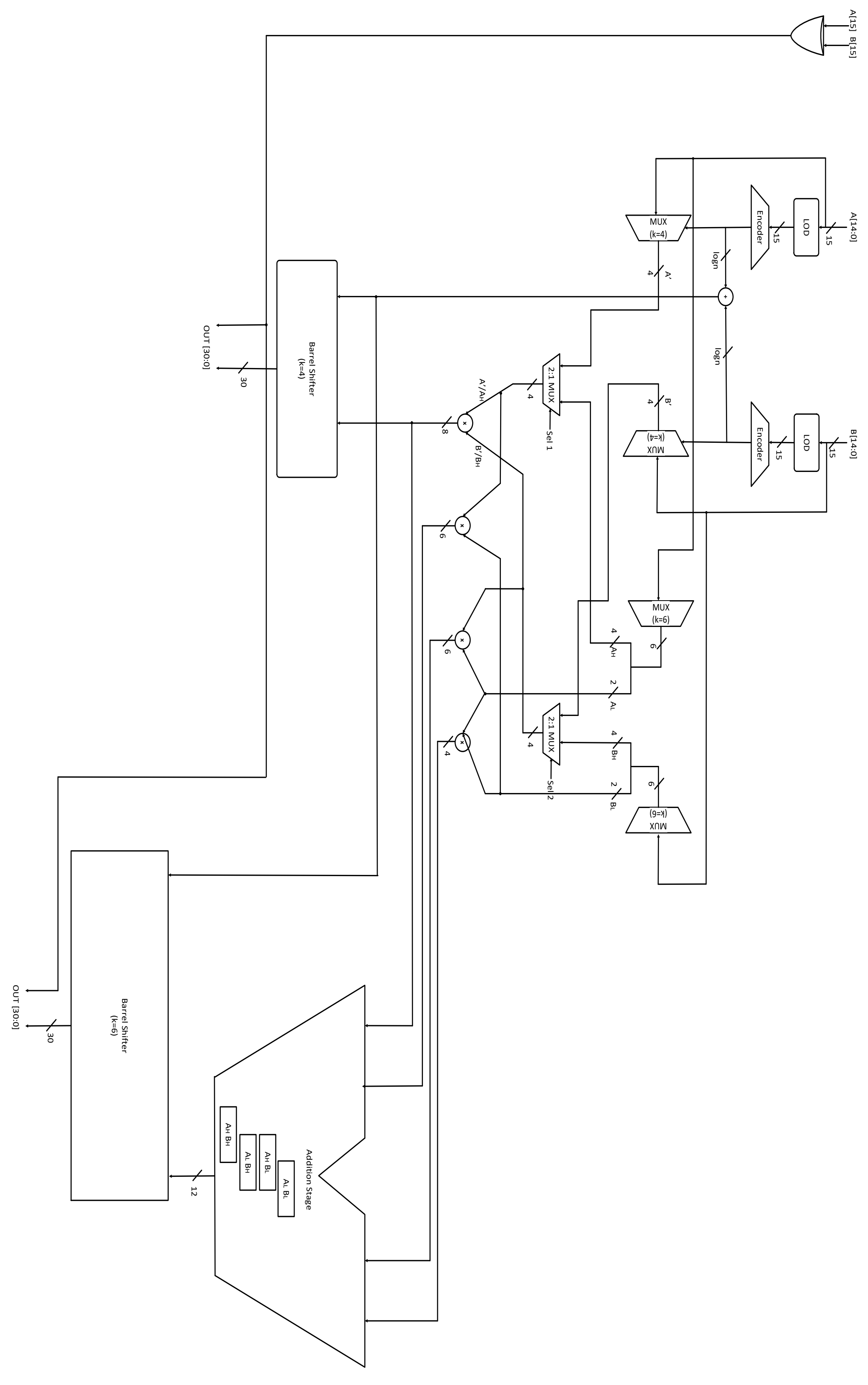


Figure 4.7 The 16-bit DSM-based hybrid approximate multiplier for mode [4 and 6]

4.2 Simulation and Synthesis Results

In this section, the proposed designs are compared with separated reconfigurable designs. The power consumption, critical path delay and design area of each design are shown in Table 4.1. All designs were implemented using Verilog HDL and synthesized using Synopsys DC complier in a 15nm library at the typical process corner.

Table 4.1 Circuit's parameter of 16-bit merged and separated reconfigurable designs

Design (4 & 8)	**Segment Type**	**Delay (*ps*)**	**Power (μW)**	**Area (μm^2)**	**Speed Increase vs. Exact**	**Power Reduction vs. Exact**	**Area Reduction vs. Exact**
Separated	**SSM**	378.9	22.3	157.4	33.1%	82.0%	69.8%
Merged	**SSM**	416.1	23.4	145.8	26.5%	81.1%	72.1%
Separated	**DSM**	448.8	51.9	216.9	20.7%	58.1%	58.4%
Merged	**DSM**	486.1	53.1	161.8	14.1%	57.1%	68.9%

The mode (4 & 8) of 16-bit SSM-based hybrid approximate multiplier achieves a 26.5% speed increase, an 81.1% power reduction, and a 72.1% area reduction compared to the 16-bit exact multiplier. In addition, the mode (4 & 8) of 16-bit DSM-based hybrid approximate multiplier design achieves a 14.1% speed increase, a 57.1% power reduction, and a 68.9% area reduction compared to the 16-bit exact multiplier. The merged reconfigurable design has a smaller area compared to a single-precision design. Overall, the improved area efficiency comes at the cost of slightly longer delay and larger power dissipation. However, the proposed design demonstrates good flexibility for trading power and delay for the smaller area.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

1. A comprehensive comparison of the existing approximate multipliers has been performed on 8-bit designs. The segment-based multipliers are very competitive in terms of both power and area, while the other designs have at least one major shortcoming in accuracy, delay or power. The segment-based design allows for a trade-off between accuracy and performance by adjusting segment size m.

2. The dual segmentation approximate multiplier achieves better performance in terms of the delay, area and energy compared to the DSM-based multiplier while having almost the same accuracy level as the DSM-based multiplier.

3. The merged approximate multiplier with two precisions can achieve significant performance gains compared to the exact multipliers. Also, it can achieve a significantly better performance-accuracy trade-off compared to the single-precision approximate multipliers.

5.2 Future Work

1. The merged approximate multiplier can be further improved to have a lower delay and power consumption. Since the area savings are realized with longer critical path delay and higher power consumption compared to the separated designs.
2. To evaluate the feasibility of the proposed multipliers in error-tolerant applications such as image processing applications.

Bibliography

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," *2013 18th IEEE European Test Symposium (ETS)*, Avignon, France, 2013, pp. 1-6, doi: 10.1109/ETS.2013.6569370.

[2] Q. Xu, T. Mytkowicz and N. S. Kim, "Approximate Computing: A Survey," in *IEEE Design & Test*, vol. 33, no. 1, pp. 8-22, Feb. 2016, doi: 10.1109/MDAT.2015.2505723.

[3] V. K. Chippa, S. T. Chakradhar, K. Roy and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, 2013, pp. 1-9, doi: 10.1145/2463209.2488873.

[4] S. Venkatachalam and S. Ko, "Design of Power and Area Efficient Approximate Multipliers," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1782-1786, May 2017, doi: 10.1109/TVLSI.2016.2643639.

[5] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris and K. Pekmestzi, "Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105-3117, Oct. 2016, doi: 10.1109/TVLSI.2016.2535398.

[6] G. M. Strollo, E. Napoli, D. D. Caro, N. Petra and G. D. Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers", *Circuits and Systems I: Regular Papers IEEE Transactions on*, vol. 67, no. 9, pp. 3021-3034, 2020.

[7] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018.

[8] S. Narayanamoorthy, H. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 23, no. 6, pp. 1180–1184, 2015.

[9] S. Hashemi, R. I. Bahar, and S. Reda, “DRUM: A dynamic range unbiased multiplier for approximate applications,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2015, pp. 418–425.

[10] S. Vahdat, M. Kamal, A. Afzali-Kusha and M. Pedram, "TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161-1173, May 2019.

[11] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, “RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[12] L.Li, I.Hammad, and K. El-Sankary, “Dual segmentation approximate multiplier,” in *Electronics Letters*, doi: 10.1049/ell2.12243, 2021.

[13] M. S. Kim, A. A. Del Barrio Garcia, H. Kim and N. Bagherzadeh, “The Effects of Approximate Multiplication on Convolutional Neural Networks,” in *IEEE Transactions on Emerging Topics in Computing*, doi: 10.1109/TETC.2021.3050989.

[14] I. Hammad, L. Li, K. El-Sankary and W. M. Snelgrove, “CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers With Various Precisions,” in *IEEE Access*, vol. 9, pp. 7220-7232, 2021, doi: 10.1109/ACCESS.2021.3049299.

[15] I. Hammad and K. El-Sankary, "Impact of Approximate Multipliers on VGG Deep Learning Network," in *IEEE Access*, vol. 6, pp. 60438-60444, 2018, doi: 10.1109/ACCESS.2018.2875376.

[16] I. Hammad, K. El-Sankary and J. Gu, "Deep Learning Training with Simulated Approximate Multipliers," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, China, 2019, pp. 47-51, doi: 10.1109/ROBIO49542.2019.8961780.

[17] Z. Liu, A. Yazdanbakhsh, T. Park, H. Esmaeilzadeh and N. S. Kim, "SiMul: An Algorithm-Driven Approximate Multiplier Design for Machine Learning," in *IEEE Micro*, vol. 38, no. 4, pp. 50-59, Jul./Aug. 2018, doi: 10.1109/MM.2018.043191125.

[18] Y. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017, doi: 10.1109/JSSC.2016.2616357.

[19] S. Yin and J. Seo, "A 2.6 TOPS/W 16-Bit Fixed-Point Convolutional Neural Network Learning Processor in 65-nm CMOS," in *IEEE Solid-State Circuits Letters*, vol. 3, pp. 13-16, 2020, doi: 10.1109/LSSC.2019.2954780.

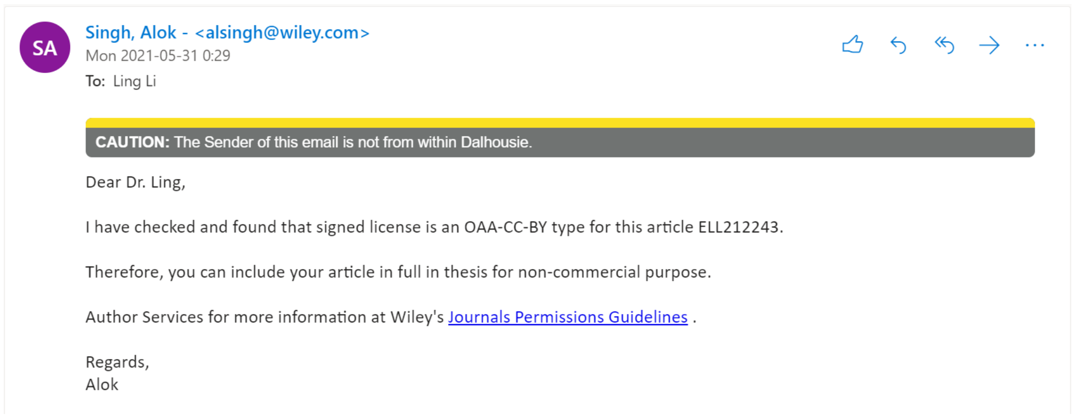
[20] Z. Yuan et al., "STICKER: An Energy-Efficient Multi-Sparsity Compatible Accelerator for Convolutional Neural Networks in 65-nm CMOS," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 2, pp. 465-477, Feb. 2020, doi: 10.1109/JSSC.2019.2946771.

[21] Xuechao Wei et al., "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1-6, doi: 10.1145/3061639.3062207.

[22] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou and Lingli Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1-9, doi: 10.1109/FPL.2016.7577308.

Appendix Permission

This thesis reuses materials from my accepted article. The permission is for Chapter 3. Chapter 3 was from "[12] L.Li, I.Hammad, and K. El-Sankary, "Dual segmentation approximate multiplier," in *Electronics Letters*, doi: 10.1049/ell2.12243, 2021." An editor of Wiley on behalf of the Institution of Engineering and Technology confirmed that the article could be included in full in the thesis. The email is attached (next page):



Singh, Alok - <alsingh@wiley.com>
Mon 2021-05-31 0:29
To: Ling Li

CAUTION: The Sender of this email is not from within Dalhousie.

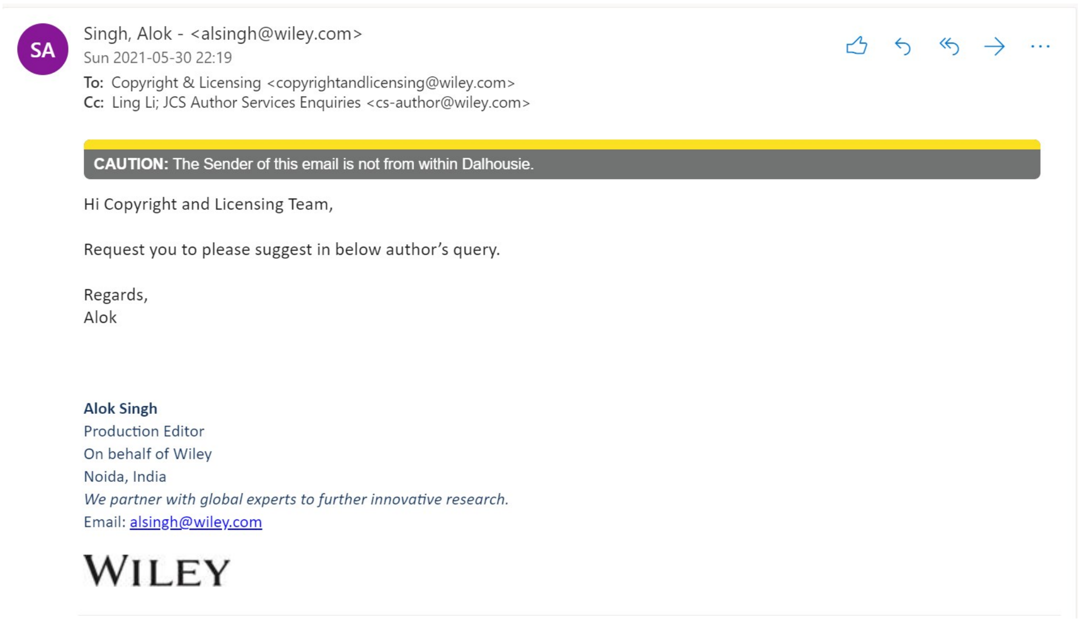
Dear Dr. Ling,

I have checked and found that signed license is an OAA-CC-BY type for this article ELL212243.

Therefore, you can include your article in full in thesis for non-commercial purpose.

Author Services for more information at Wiley's Journals Permissions Guidelines .

Regards,
Alok



Singh, Alok - <alsingh@wiley.com>
Sun 2021-05-30 22:19
To: Copyright & Licensing <copyrightandlicensing@wiley.com>
Cc: Ling Li; JCS Author Services Enquiries <cs-author@wiley.com>

CAUTION: The Sender of this email is not from within Dalhousie.

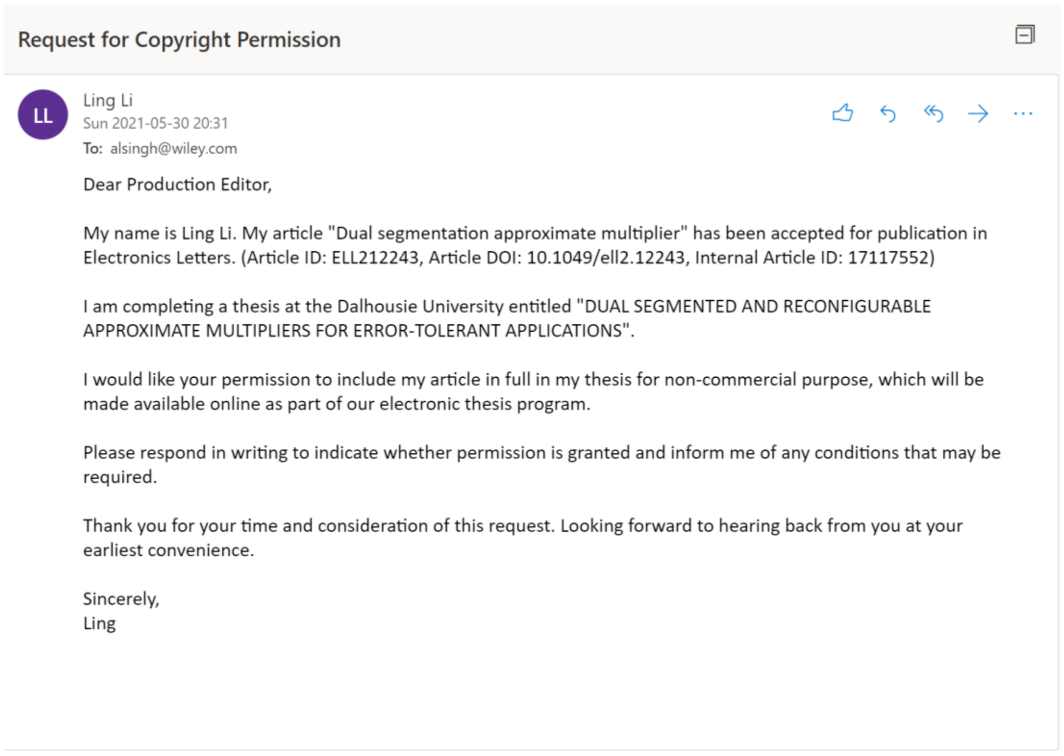
Hi Copyright and Licensing Team,

Request you to please suggest in below author's query.

Regards,
Alok

Alok Singh
Production Editor
On behalf of Wiley
Noida, India
We partner with global experts to further innovative research.
Email: alsingh@wiley.com

WILEY



Request for Copyright Permission

Ling Li
Sun 2021-05-30 20:31
To: alsingh@wiley.com

Dear Production Editor,

My name is Ling Li. My article "Dual segmentation approximate multiplier" has been accepted for publication in Electronics Letters. (Article ID: ELL212243, Article DOI: 10.1049/ell2.12243, Internal Article ID: 17117552)

I am completing a thesis at the Dalhousie University entitled "DUAL SEGMENTED AND RECONFIGURABLE APPROXIMATE MULTIPLIERS FOR ERROR-TOLERANT APPLICATIONS".

I would like your permission to include my article in full in my thesis for non-commercial purpose, which will be made available online as part of our electronic thesis program.

Please respond in writing to indicate whether permission is granted and inform me of any conditions that may be required.

Thank you for your time and consideration of this request. Looking forward to hearing back from you at your earliest convenience.

Sincerely,
Ling