반도체공학회 논문지 (TSE: Transactions on Semiconductor Engineering), Vol. 2, No. 4, Dec., 2024
https://doi.org/10.22895/tse.2024.0019 ISSN (On-line) 3022-2028

근사 4-2 컴프레서와 곱셈기 설계의 최근 연구 경향

(Recent Advances in Design of Approximate 4-2 Compressors and Multipliers)

전종민 [1], 김영민 [1,+]
(Jongmin Jeon[1], Youngmin Kim [1,+])

요 약

근사 컴퓨팅은 하드웨어 요소를 최적화하고 정확도 손실을 허용하는 유망한 접근 방식이다. 특히, 근사 곱셈은 고성능과 저전력을 동시에 요구하는 컴퓨팅에서 핵심적인 연산으로 널리 활용되고 있다. 이 중 근사 4-2 컴프레서는 저전력 소모, 처리 속도 향상, 회로 단순화 등의 이점을 통해 근사 곱셈기의 성능과 효율성을 크게 향상시킬 수 있는 기술로 주목받고 있다. n 비트의 두 수를 곱하는 곱셈기는 부분 곱 생성, 부분 곱 축소, 그리고 가산기 세 가지 단계로 구성된다. 근사 4-2 컴프레서를 사용하여 부분 곱 축소 단계를 단순화하거나, 오류 수정 모듈을 통해 근사로 인한 오류를 보상할 수 있다. 또한, 상수 수정 기법을 통해 오류를 줄이는 방법도 활용된다. 본 논문에서는 근사 4-2 컴프레서의 특성과 여러 모델을 비교하고 이를 바탕으로 8×8 근사 곱셈기에 대한 다양한 에러 지표와 합성 결과를 분석한다. 또한, 이미지 프로세싱을 적용한 결과를 통해 비교 분석을 수행한다.

ABSTRACT

Approximate computing is a promising approach that optimizes hardware components while tolerating some degree of accuracy loss. In particular, approximate multiplication is widely used as a key operation in computing systems that demand both high performance and low power consumption. Among various techniques, the approximate 4-2 compressor has gained attention for its ability to significantly enhance the performance and efficiency of approximate multiplier through benefits such as low power consumption, improved processing speed, and circuit simplification. A multiplier that computes the product of two n-bit numbers typically consists of three stages: partial product generation, partial product reduction, and adder. By using an approximate 4-2 compressor, the partial product reduction stage can be simplified, and errors introduced by approximation can be compensated for using error correction modules. Additionally, constant correction techniques can be employed to further reduce errors. This paper explores the characteristics of approximate 4-2 compressors and compares various models. Based on this analysis, we evaluate different error metrics and synthesis results for an 8×8 approximate multiplier. Furthermore, we conduct a comparative analysis by applying the multiplier to image processing tasks.

KEY WORDS
Approximate Computing; Approximate 4-2 Compressor; Error Recovery; Truncation; Constant Correction; Power consumption; Delay; Accuracy

I. 서 론

초대형 집적 회로(Very Large Scale Integration, VLSI) 기술의 발전으로 전자, 통신, 멀티미디어 분야에서 마이크로일렉트로닉스 기술의 응용이 매우 넓어졌다. VLSI에서 중요한 기준은 처리량을 타협하지 않으면서 면적과 전력을 줄이는 것이다 [1].

[1] School of Electronic and Electrical Engineering, Hongik University
[+]Corresponding author: Youngmin Kim, youngmin@hongik.ac.kr
(Received Dec. 10, 2024, Accepted Dec. 28, 2024)

정확성은 전통적인 산술 회로에서 중요한 요구사항이다. 하지만 정확한 계산은 많은 면적(Area), 전력(Power), 그리고 지연 시간(Delay)을 필요로 한다 [1], [2]. 근사 계산(Approximate Computing)은 VLSI 설계 분야에서 많은 관심을 받고 있으며, 하드웨어 요소를 최적화해 전력 소비와 설계 면적을 줄이면서도 약간의 정확도 손실을 감수하는 접근 방식이다 [3]. 또한, 회로 및 칩의 에너지를 효율적으로 줄이는 방법이기도 하다 [4]. 이미지 처리, 머신 러닝 등의 응용 분야에서는 전력 효율성을 개선하기 위해 근사 계산이 효과적인 방법으로 활용된다 [5]. 그리고 이 방법은 응용 프로그램의 오류 허용성을 활용하여 시스템 자원 성능을 향상시키는 방법이다 [3]. 오류 허용 응용 프로그램에서 곱셈기(Multiplier)는 중요한 역할을 한다 [6].

곱셈은 많은 디지털 신호 처리(Digital Signal Processing, DSP) 알고리즘에서 중요한 산술 연산이며, 곱셈기의 근사는 하드웨어 사용을 낮추는 효과적인 방법을 제공한다 [2]. 근사 곱셈기는 1970년대부터 광범위한 연구의 주제가 되어 왔으며, “디지털 필터링 응용을 위한 근사 이진 로그를 사용한 곱셈 및 나눗셈 생성” 이라는 제목의 기사가 발표되면서 이 연구 분야가 시작되었다. 그 이후로 근사 곱셈기의 설계 및 최적화와 관련하여 많은 발전이 이루어졌다 [7].

근사 곱셈기는 절사(Truncation) 및 비절사(Non-Truncation) 방식 두 가지 전략으로 설계된다. 절사 방식은 근사 곱셈기의 최하위 비트를 계산하지 않고 특정 상수로 대체하여 계산 복잡성을 줄이고 지연 시간을 감소시켜 전력을 줄이는 효과가 있으며, 또한 이때 오류의 발생을 줄이기 위해 오류 보상 모듈(Error Recovery Module)을 사용할 수 있다. 이 모듈을 추가하게 되면 추가적인 오버헤드를 발생시켜 면적, 전력 및 지연 시간이 약간 증가할 수 있다. 비 절사 방식에서는 최하위 비트를 절사하지 않는다 [1]. 정확한 4-2 컴프레서(Exact 4-2 Compressor)는 정밀도가 중요한 디지털 회로나 이미지 처리와 같은 응용 분야에서 핵심적인 역할을 한다. 이 컴프레서는 계산의 정확성을 유지해 이미지 처리 시 픽셀 값의 무결성을 보장하고, 오류를 최소화하여 고품질 출력을 제공한다. 실시간 처리 시스템에서는 일관된 성능이 필수적이며, 정확한 4-2 컴프레서는 이러한 요구를 충족시킨다. 특히, 캐리(Carry)와 캐리 아웃(Cout)과 같은 중요한 출력의 정확성을 보장해 시스템의 전체적인 기능성을 유지하는 데 기여한다. 하이브리드 설계에서 정확한 4-2 컴프레서는 전력 효율성과 정확성의 균형을 맞추는 기반이 되며, 전력 소비를 줄이기 위해 정확한 요소와 부정확한 요소를 적절히 조합하는 데 사용된다. 따라서, 오류를 최소화하고 일관된 성능을 유지하며, 에너지 효율성을 달성하는 데 정확한 4-2 컴프레서의 사용이 매우 중요하다 [8]. 최근에는 근사 4-2 컴프레서(Approximate 4-2 Compressor)를 활용한 근사 곱셈기(Approximate Multiplier) 구현 방식이 주목받고 있다. 이 방법은 일부 정확한 4-2 컴프레서를 단순화된 회로로 대체하여 작은 오류를 감수하는 대신, 전력 소비, 속도, 면적 측면에서 더 높은 효율성을 제공한다. 근사 4-2 컴프레서는 여러 장점을 통해 성능과 효율성을 크게 향상시킨다. 첫째, 저전력 소모가 가능하여 저전력 기반 응용에 적합하며, 설계를 최적화하면 전력 소비와 지연 시간을 동시에 줄일 수 있다. 둘째, 회로 구조를 간소화하여 칩의 물리적 크기를 줄일 수 있어 집적 회로 설계에 유리하다. 셋째, 곱셈기의 주요 경로를 단 하나의 근사 4-2 컴프레서로 제한함으로써 처리 속도를 향상시킬 수 있다. 넷째, 다양한 설계 옵션을 제공하여 설계자가 원하는 정확도와 전력 소비를 고려한 최적의 구조를 선택할 수 있다. 다섯째, 전력, 속도, 정확도 간의 균형을 잘 맞출 수 있다. 이러한 이유로 근사 4-2 컴프레서는 현대 디지털 회로 설계에서 중요한 대안으로 자리매김하고 있다 [9].

n비트의 두 수를 곱하는 곱셈기는 세 가지 주요 단계로 이루어진다. 부분 곱 생성(Partial Product Generation, PPG), 부분 곱 축소(Partial Product Reduction, PPR) [10], 그리고 가산기(Adder). 1) 피승수(Multiplicand)의 각 비트는 승수(Multiplier)의 각 비트와 AND 연산을 수행하여 n개의 부분 곱을 생성한다. 이 단계는 부분 곱을 생성하는 과정이므로 부분 곱 생성(PPG) 단계라고 불린다. 2) 생성된 n개의 부분 곱은 적합한 방법을 통해 두 개의 항으로 압축되며, 이 과정은 부분 곱 축소(PPR) 단계라고 한다. 3) 마지막으로, 두 항은 리플 캐리 가산기(Ripple Carry Adder) 또는 캐리 전파 가산기(Carry Propagation Adder)를 사용하여 최종

결과를 계산한다 [11]. 특히, PPR 단계는 면적, 지연 시간, 전력 소비 측면에서 가장 중요한 역할을 한다. 또한, 2단계에서 사용되는 기법들은 1단계와 3단계에서 사용되는 기법들과 결합하여 적용될 수 있다. 2단계는 주로 4−2 컴프레서를 사용하여 구성되며, 정확한 4−2 컴프레서, 근사 4−2 컴프레서, 전가산기(Full Adder, FA), 반가산기(Half Adder, HA)를 전략적으로 배치하여 최적화한다. PPR을 위해 3−2 컴프레서(Carry Save Adder)를 이용하기도 하지만, 4−2 컴프레서를 사용하면 더 나은 결과를 얻을 수 있다. 특히 PPR에 근사 4−2 컴프레서를 사용하면 전력, 지연 시간, 면적이 크게 감소된 근사 곱셈기를 설계할 수 있다 [12]. 근사 4−2 컴프레서에서 오류가 발생할 때, 근사 곱셈기에서 근사 4−2 컴프레서를 사용하는 열 중에서 최상위 비트 부분에 근사 4−2 컴프레서의 입력을 활용하여 오류를 보상할 수 있는 오류 보상 모듈을 만들거나, 근사 4−2 컴프레서의 입력만을 사용하여 정확한 컴프레서를 사용하는 열의 최하위 비트 부분에서 정확한 4−2 컴프레서에 캐리 인(Cin)에 연결하여 근사 곱셈기의 오류를 보상할 수 있다. 이때 하드웨어 복잡도를 줄이기 위해 근사 곱셈기 최하위 비트에 상수를 수정하여 절사 기법을 적용할 수 있다.

본 논문에서는 다양한 근사 4−2 컴프레서와 근사 곱셈기 설계 방법 및 성능에 관한 최신 연구들을 분석하고, 절사, 오류 보상 모듈과 같은 기법들이 성능에 미치는 영향을 살펴보고자 한다.

II. 본 론

1. Exact 4−2 compressor and Recently developed approximate 4−2 compressors

이 섹션에서는 다양한 근사 4−2 컴프레서 설계에 대한 도출 과정, 논리식, 그리고 근사 곱셈기 구조 분석을 통해 여러 가지 최신 연구를 검토한다.

가. Exact 4−2 Compressor [8]

정확한 4−2 컴프레서 [8]는 다섯 개의 입력(X1, X2, X3, X4, Cin)과 세 개의 출력(Sum, Carry, Cout)을 가지며, 이 회로는 두 개의 Full Adder 셀을 이용해 구현되며, 그 구조는 그림 1에 나타나 있다. 출력인 합(Sum)은 입력 신호와 동일한 가중치를 가지며, 캐리 인(Cin)은 하위 단계의 컴프레서에서 전달된 캐리(Carry)이고, 캐리 아웃(Cout)은 다음 단계 컴프레서($Cin_i = Cout_{i-1}$)로 전달되는 Carry이다. 출력 Carry는 한 비트 더 높은 위치의 부분 곱의 비트처럼 누적된다. Cout, Carry는 동일한 가중치를 가진다 [13]. 이러한 연결 구조는 그림 2에 표현되어 있으며, 식 (1), (2), (3)은 각각 Carry, Sum, Cout에 대한 논리식이다.

$$Carry = (X1 \oplus X2 \oplus X3 \oplus X4)C_{in} + (\overline{X1 \oplus X2 \oplus X3 \oplus X4})X4 \quad (1)$$

$$Sum = X1 \oplus X2 \oplus X3 \oplus X4 \oplus C_{in} \quad (2)$$

$$C_{out} = (X1 \oplus X2)X3 + (\overline{X1 \oplus X2})X1 \quad (3)$$

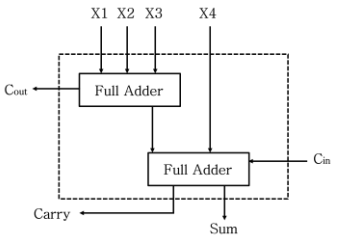

그림 1 . Exact 4−2 Compressor

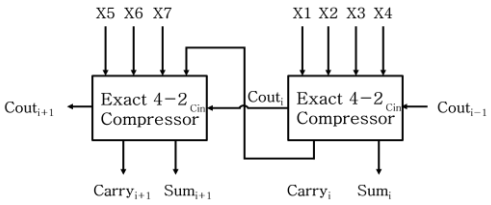

그림 2 . Exact 4−2 Compressor 연결 구조

나. Approximate 4−2 Compressor

(1) Compressor of [14]

[14]에서 첫 번 째로 제안된 근사 4−2 컴프레서는 정확한 4−2 컴프레서 [8]의 진리표를 기반으로 한다. 정확한 4−2 컴프레서 [8]의 진리표에서는 전체 32개의 상태 중 24개에서 입력 Cin과 출력 Carry가 동일한 값을 가진다. 이 특성을 활용하여 근사 4−2 컴프레서를 설계할 때, Carry를 단순히 Cin으로 처리할 수 있다. 또한, Sum과 Cout 논리식도 회로의 복잡성을 줄이기 위해 간소화되었다. 특히, Carry와 Cout은 동일한 가중치를 가지므로, 이전 단계에서 Carry와 Cout의 논리식을 서로 교환할 수 있다. [14]에서 두 번 째로 제안된 근사 4−2

컴프레서는 Carry가 식 (6)의 우측 논리식을 사용하고, Cout은 항상 Cin과 동일하게 설정된다. 또한, 첫 번째 단계에서 Cin이 0이므로 모든 단계에서 Cout과 Cin 역시 0이 되어 하드웨어 설계에서 이들 값을 생략할 수 있다. 이를 통해, 식 (6)의 좌측에 있는 Cout을 식 (4)의 좌측 Carry로 대체하면 논리식이 단순해지고 회로 설계가 더욱 간소화된다. 결과적으로, 이러한 설계 방식은 정확한 4−2 컴프레서 [8]와 비교하여 임계 경로 지연(Critical Path Delay)과 전력 소비를 효과적으로 줄일 수 있다 [14].

[14]에서 두 번 째로 제안된 설계는 두 개의 출력을 가지며, 첫 번 째로 제안된 설계와 비교해 출력 개수가 하나 줄었고 이에 따라 사용되는 게이트 수도 추가로 감소하여 더욱 간소화된 회로를 구현할 수 있었다. 식 (4), (5), (6)은 [14]에서 첫 번 째로 제안된 컴프레서의 Carry, Sum, Cout에 대한 논리식이고 식 (7), (8)은 두 번 째로 제안된 컴프레서의 Carry, Sum에 대한 논리식이다. 그리고 [14]에서 두 번 째로 제안된 컴프레서 회로(Circuit)는 그림 3. (a)에 나타나 있다.

$$Carry = C_{in} \tag{4}$$

$$Sum = \overline{C_{in}}\,(\overline{X1 \oplus X2} + \overline{\overline{X3 \oplus X4}}) \tag{5}$$

$$C_{out} = \overline{(\overline{X1X2} + \overline{X3X4})} \tag{6}$$

$$Carry = \overline{(\overline{X1X2} + \overline{X3X4})} \tag{7}$$

$$Sum = (\overline{X1 \oplus X2} + \overline{\overline{X3 \oplus X4}}) \tag{8}$$

(2) Compressor of [15]

[15]에서 제안된 근사 4−2 컴프레서의 성능을 개선하기 위해서는 입력 순서의 중요성을 분석하고 이를 이해하는 것이 필수적이다. 이 컴프레서의 진리표에서 입력이 X1X2X3X4 = “0000”, “1000”, “0100”, “0010”, “0001”인 경우에 정확한 출력을 생성해야 한다. 특히, X1과 X2의 AND 연산은 높은 확률로 정확한 Carry 출력을 만들지만, 동시에 8개의 잘못된 출력을 발생시킬 수 있다. 이 문제를 해결하기 위해 X3과 X4의 AND 연산을 OR 게이트와 결합하여 잘못된 출력을 5개로 줄일 수 있다. Carry 출력을 구현할 때 드모르간의 법칙을 활용하여 NAND 게이트만 사용하는 방식을 적용했다.

Sum 출력을 개선하기 위해 X1과 X2의 XOR 연산을 활용하였으며, 이는 입력이 X1X2X3X4 = “0010” 일 때를 제외하고는 대부분 정확한 결과를 제공한다. 또한, X3과 X4의 OR 연산을 결합함으로써 Sum 출력을 더욱 정확하게 만든다. 이러한 설계를 통해 지연 시간과 전력 소모가 개선된다. 결과적으로, 이 컴프레서는 [14]에서 두 번 째로 제안된 설계와 비교하여 게이트 수가 줄어들었고, 이는 전력 소모와 지연 시간에서 성능 향상을 가져왔다. 식 (9), (10)은 [15]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [15]에서 제안된 컴프레서 회로는 그림 3. (b)에 나타나 있다.

$$Carry = \overline{(\overline{X1X2})(\overline{X3X4})} \tag{9}$$

$$Sum = \overline{(\overline{(\overline{X1 \oplus X2})(\overline{X3 + X4})})} \tag{10}$$

(3) Compressor of [16]

[16]에서 제안된 근사 4−2 컴프레서는 [14]에서 두 번 째로 제안된 근사 4−2 컴프레서를 기반으로 Carry와 Sum 신호를 활용하여 최적화되었다. 우선, 드모르간의 법칙을 [14]에서 제안된 컴프레서의 Carry 신호에 적용함으로써 곱의 합 형태를 유도할 수 있다. 이렇게 생성된 Carry 신호는 하나의 복합 게이트로 구현되며, 이후 Sum 신호를 생성하는데 재사용된다. 또한, [14]에서 제안된 컴프레서의 Sum 신호를 위한 부울 방정식은 XOR 게이트를 AND 게이트와 OR 게이트로 변환한 후, 다시 드모르간의 법칙을 적용하여 네 개의 항을 포함하는 합의 곱 형태로 변환된다. 이 과정에서 Carry의 부정 표현이 나타나며, 최종적으로 Sum 신호는 Carry 신호를 활용한 세 개의 곱의 합으로 표현된다. 식 (11), (12)는 [16]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [16]에서 제안된 컴프레서 회로는 그림 3. (c)에 나타나 있다.

$$Carry = (X1 + X2)(X3 + X4) \tag{11}$$

$$Sum = X1X2 + X3X4 + \overline{Carry} \tag{12}$$

(4) Compressor of [17]

[17]에서 제안된 근사 4−2 컴프레서는 Sum과 Carry 연산에서 특정 입력 조건에 따른 오류를 식별하고 이를 기반으로 최적화된 설계를 제시한다. Sum 연산에서는 입력이 X1X2X3X4 = “0011”, “1111”인 경우에서 오류가 발생한다. 첫 번째 경우(X1X2 = “00”)에는 Sum의 결과가 항상 0이고, 두 번째 경우(X1X2 = “11”)에는

Sum의 결과가 항상 1이다. 이러한 조건을 기반으로, Sum 출력은 XOR(X1, X2)를 활용하여 결정되며, 2:1 멀티플렉서(Multiplexer, MUX)를 제어 신호로 사용한다.

Carry 연산에서도 오류가 발생하는 조건이 식별되었다. 입력이 X1X2X3X4 = “1100”, “1111” 일 때 발생하며, X1과 X2가 (1, 1) 일 때, Carry의 출력은 항상 1이 된다. 또한, XOR(X1, X2)가 1일 경우 Carry는 OR(X3, X4)와 연관되며, XOR(X1, X2)가 0일 경우에는 Carry가 0 또는 1일 수 있다. 최종적으로 Carry는 AND(X1, X2)와 OR 게이트를 조합하여 계산된다 [17]. 식 (13), (14)는 [17]에서 제안된 컴프레서의 Carry와 Sum에 대한 논리식이며, [17]에서 제안된 컴프레서 회로는 그림 3. (d)에 나타나 있다.

$$Carry = (X1 \oplus X2)(X3 + X4) + X1X2 \quad (13)$$

$$Sum = (X1 \oplus X2) \oplus (X3 \oplus X4) + X1X2 \quad (14)$$

다. Error Recovery Module for Approximate 4−2 Compressor

(1) Compressor of [12]

[12]에서 명시된 [18]의 근사 4−2 컴프레서의 진리표에 따르면, 입력 X1X2X3X4 = “0011” 일 때 정확한 출력은 “10”(Carry, Sum)이어야 하지만, [18]의 컴프레서는 “11”(Carry, Sum)을 생성했다. 이로 인해 컴프레서 [18] 결과와 정확한 4−2 컴프레서 결과 간에 차이(Difference, D)는 이진 산술을 사용하여 “11−10 = 1”로 계산된다. [18]의 컴프레서에서는 이 입력 경우에만 D = 1이며, 나머지 입력(X1X2X3X4 = “1011”, “0111”, “1111”)에서는 D = −1이 된다. 이에 반해, [12]에서 제안된 컴프레서는 모든 진리표 항목에서 D = 0 또는 D = −1이 되도록 설계되었다. 이는 입력 X1X2X3X4 = “0011” 일 때 “01”(Carry, Sum)을 출력하도록 변경하고, 나머지 입력(X1X2X3X4 = “1011”, “0111”, “1111”)은 [12]과 동일하게 유지함으로써 이루어졌다.

특히 [12]에서 제안된 컴프레서의 Carry는 기존 Carry [18]에 비해 곱 항 하나가 감소하여 더욱 간소화되었다. 또한, [12]에서는 D = −1 오류(Error)를 해결하기 위해 오류 보상 모듈(Error Recovery Module)을 사용한다. 이 모듈은 컴프레서 [12]의 입력이 X3X4 = “11” 일 때 발생하는 D = −1의 오류를 보정하도록 설계되었다. 이를 구현하기 위해 근사 4−2 컴프레서가 사용된 열 중 최상위 비트 부분의 입력(X3X4)을 활용한다. 이 입력은 두 개의 2−input AND 게이트의 입력으로 사용되며, 각각의 AND 게이트 출력은 하나의 2−input OR 게이트의 입력으로 연결된다. 따라서, Error Recovery Module은 두 개의 2−input AND와 하나의 2−input OR 게이트로 구성된다. 식 (15), (16)은 [12]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [12]에서 제안된 컴프레서와 Error Recovery Module 회로는 그림 3. (e), (f)에 나타나 있다.

$$Carry = X1X2 + X1X3 + X1X4 + X2X3 + X2X4 \quad (15)$$

$$Sum = (X1 \oplus X2) \oplus (X3 \oplus X4) \quad (16)$$

(2) Compressor of [19]

[19]에서 제안된 근사 4−2 컴프레서는 Carry에 대해 [18]과 동일한 논리식을 사용하지만, 오류 성능 향상의 대가로 Sum에 대한 논리식을 수정하였다. [12], [18]과 비교하여 회로가 약간 더 복잡해진다. [19]의 진리표에서, 이 설계는 두 개의 오류만 발생하므로 [12], [18]에서 제안된 컴프레서보다 더 정확하다. 컴프레서의 오류 확률은 부분 곱(Partial Product)이 피연산 비트들의 AND 연산으로 생성된다는 점을 기반으로 계산할 수 있다. 따라서 부분 곱이 1이 될 확률은 1/4, 0이 될 확률은 3/4이다. [19]의 진리표를 활용하여 컴프레서의 평균 오류 확률(Average Error Probability, Avg. Error Prob.)을 쉽게 구할 수 있다. 예를 들면, [12], [18]에서 제안된 컴프레서는 X1X2X3X4 = “0011”, “1011”, “0111”, “1111” 일 때 오류가 발생한다. 평균 오류 확률은 다음 값들을 모두 더해 계산된다: “0011”에서 9/256, “1011”에서 3/256, “0111”에서 3/256, 그리고 “1111”에서 1/256이다. 이를 합하면 평균 오류 확률은 16/256이 된다. 이러한 방법을 통해, [19]의 컴프레서는 4/256으로 계산된다 [19]. [19]에서 제안된 컴프레서는 평균 오류 확률이 [12], [18]보다 낮아 더 정확한 결과를 제공한다. [19]의 진리표에서는 입력 X1X2X3X4= “111X” 일 때, 항상 D = −1임을 보여준다. 이를 통해 [19]의 컴프레서 역시 [12]와 마찬가지로 Error Recovery Module을 도입하여 곱셈기의 오류율(Error Rate)을 더욱 낮출 수 있다. [19]에서 제안된 Error Recovery Module은 [12]에서 사용된 방식과 유사하게 설계된다. 오류 검출을 위해 기존 [12]의 2−input AND 게이트 대신 3−input AND

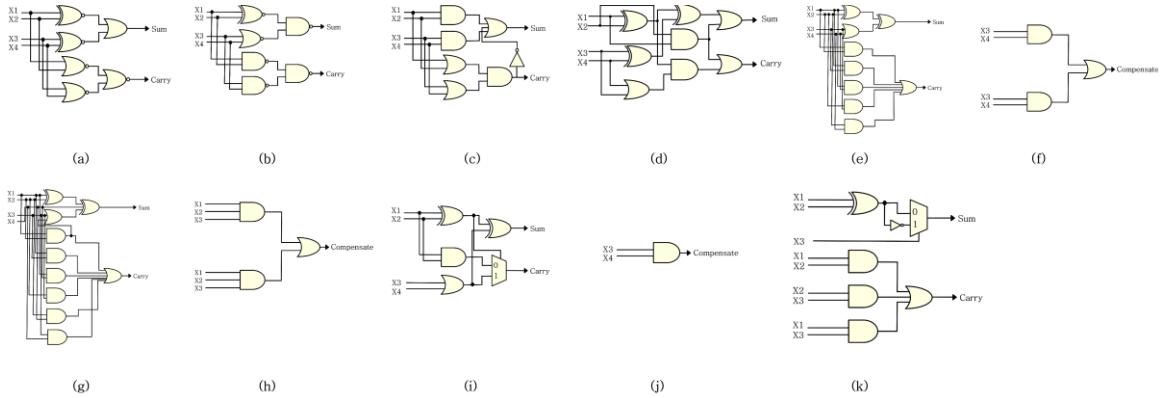

그림 3. Circuit of various approximate 4−2 compressors: (a) Compressor of [14]; (b) Compressor of [15]; (c) Compressor of [16]; (d) Compressor of [17]; (e) Compressor of [12]; (f) Error recovery module of [12]; (g) Compressor of [19]; (h) Error recovery module of [19]; (i) Compressor of [20] (j) Error recovery module of [20]; (k) Compressor of [11].

게이트가 사용되며, 이 모듈은 근사 4−2 컴프레서를 사용하는 열의 최상위 비트 입력(X1X2X3)을 활용한다. 3−input AND 게이트를 통해 생성된 두 오류 신호는 2−input OR 게이트로 결합되어, 부분 곱 행렬의 다음 열에 위치한 정확한 4−2 컴프레서의 Cin으로 연결된다. 이를 통해 곱셈기에서 발생하는 오류를 보상한다 [19]. 식 (17), (18)은 [19]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [19]에서 제안된 컴프레서와 Error Recovery Module 회로는 그림 3. (g), (h)에 나타나 있다.

$$Carry = X1X2 + X1X3 + X1X4 + X2X3 + X2X4 + X3X4 \quad (17)$$

$$Sum = (X1 \oplus X2) \oplus (X1X2 + X3) \oplus X4 \quad (18)$$

(3) Compressor of [20]

[20]에서 제안된 근사 4−2 컴프레서는 근사 곱셈기의 효율성을 높이기 위해 전력 소비와 면적을 줄이면서 정확성을 유지하도록 설계되었다. 또한 이 컴프레서를 근사 곱셈기에 도입하여 부분 곱 축소 단계에서 하드웨어 활용을 줄였다. [20]의 진리표에 따르면, 입력 X1X2X3X4 = “0011”, “0111”, “1011”, “1111” 네 가지 경우에서 D = −1인 오류가 발생한다. 이 오류를 보상하기 위해 Error Recovery Module을 사용한다. 이 모듈을 구현하기 위해서는 근사 4−2 컴프레서를 사용하는 열 중에서 최상위 비트 부분에 3개의 근사 4−2 컴프레서의 입력(X3X4)를 활용한다. 이 입력은 세 개의 2−input AND 게이트의 입력으로 사용된다. 식 (19), (20)은 [20]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [20]에서 제안된 컴프레서와 Error Recovery Module 회로는 그림 3. (i), (j)에 나타나 있다.

$$Carry = (X1X2)\overline{(X1 \oplus X2)} + (X3 + X4)(X1 \oplus X2) \quad (19)$$

$$Sum = (X1 \oplus X2) \oplus (X3 + X4) \quad (20)$$

(4) Compressor of [11]

[11]에서는 회로의 전력, 지연 시간, 면적을 줄이기 위해 정확한 4−2 컴프레서 [8]의 Cout과 Cin을 제거한 근사 4−2 컴프레서를 제안했다. [12], [20]에서 제안된 근사 4−2 컴프레서의 진리표에 따르면 입력 X1X2X3X4 = “0011”, “1011”, “0111”, “1111”에 대해 D = −1로 설정하여 설계되었다. 그러나 [11]에서는 입력 X4가 “1”인 경우, 나머지 입력 값에 관계없이 항상 D = −1로 설정하여 설계하였다. 특히, 제안된 근사 4−2 컴프레서의 진리표 [11]를 기반으로, 입력이 X1X2X3X4 = “XXX1” 일 때, 별도의 추가 게이트 없이 근사 4−2 컴프레서를 사용하는 열 중 최상위 비트 부분에서 세 개의 근사 4−2 컴프레서의 입력 중 “X4”를 활용하여 오류를 보상하였다. 식 (21), (22)는 [11]에서 제안된 컴프레서의 Carry, Sum에 대한 논리식이며, [11]에서

$$Carry = (X1X2) + (X2X3) + (X1X3) \quad (21)$$

$$Sum = \overline{X3}(X1 \oplus X2) + X3(\overline{X1 \oplus X2}) \quad (22)$$

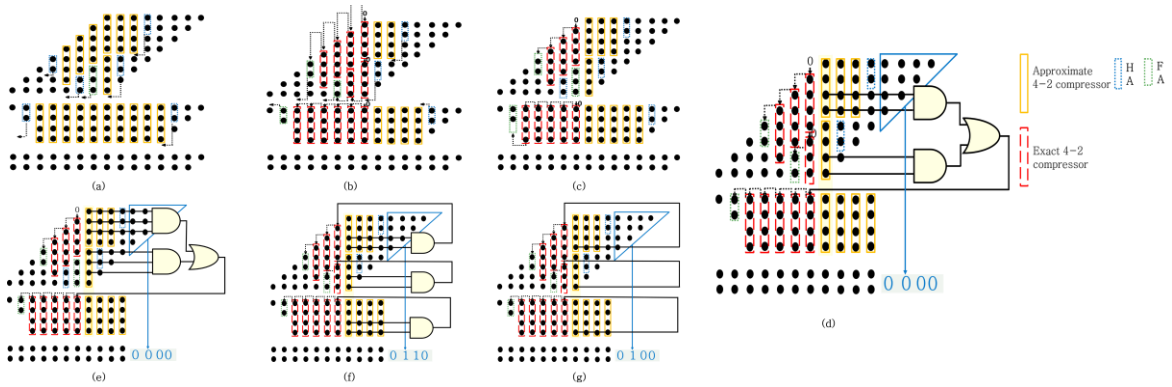

그림 4. Approximate multiplier architectures: (a) Architecture of [14]; (b) Architecture of [15]; (c) Architecture of [16], [17]; (d) Architecture of [12]; (e) Architecture of [19] (f) Architecture of [20]; (g) Architecture of [11].

제안된 컴프레서 회로는 그림 3. (k)에 나타나 있다.

2. Approximate Multiplier Design

[14]에서 제안된 근사 컴프레서를 8×8 부호 없는 Dadda 트리 곱셈기에 적용하여, 이를 통해 근사 곱셈기의 성능에 미치는 영향을 평가한다. PPG 단계에서는 AND 게이트로 부분 곱을 생성하고, PPR 단계에서는 부분 곱을 최대 네 개의 행으로 줄이기 위해 첫 번 째로 4 개의 Half Adder, 1 개의 Full Adder 및 7 개의 근사 컴프레서 [14]를 사용한다. 두 번 째로는 2 개의 Half Adder, 10 개의 근사 컴프레서 [14]를 사용한다. 마지막 단계에서는 캐리 전파 가산기(Carry Propagation Adder, CPA)를 이용해 부분 곱의 두 행을 계산한다. 따라서 8×8 Dadda 곱셈기의 축소 회로에는 두 단계의 축소 과정과 6 개의 Half Adder, 1 개의 Full Adder 및 17개의 근사 컴프레서 [14]가 필요하다. 이 근사 곱셈기 설계의 목적은 정확한 곱셈기 [8]에 비해 지연 시간과 전력 소모를 줄이는 것이지만, 그로 인해 오류 지표 값이 상대적으로 높아진다 [14]. 그림 4. (a)에는 [14]에서 제안한 근사 컴프레서를 이용한 근사 곱셈기 구조가 나타나 있다. 또한, [15]의 근사 곱셈기는 [14]와 동일한 8×8 Dadda 곱셈기를 사용하였지만, Dadda 곱셈기의 상위 비트가 곱셈 결과의 정확성을 좌우하는데 핵심적인 역할을 한다. 따라서 상위 8비트 영역에는 정확한 4−2 컴프레서 [8]를 사용하였다. [15]에서 제안된 근사 컴프레서는 정확한 값과 근사값 간의 차이를 최소화하기 위해 중간 7비트에 적용된다. 그림 4. (b)에 [15]에서 제안된 근사 컴프레서를 이용한 근사 곱셈기 구조가 나타나 있다.

근사 곱셈기를 컴프레서를 사용하여 설계할 때, 부분 곱 감소를 위한 두 가지 주요 구성 방식인 C−N 및 C−FULL 구성(Configuration)이 일반적으로 사용된다 [21]. 이러한 감소 방식은 부분 곱 행렬의 높이를 두 개의 행으로 줄인 후, 이를 리플 자리 올림 수 가산기(Ripple Carry Adder)로 합산하여 최종 곱셈 출력을 생성하는 것을 목표로 한다. C−N 구성은 최소한의 오류를 줄이기 위해 근사 컴프레서를 부분 곱 행렬의 최하위 절반 열에만 사용한다. 반면, C−FULL 구성은 정확도 감소를 감수하면서 모든 열에 컴프레서를 사용하여 하드웨어 효율성을 최적화한다 [16]. [17], [12], [19], [20], [11]의 근사 곱셈기는 C−N 구성을 적용한 곱셈기이다. 그림 4. (c)는 [16]과 [17]에서 제안된 곱셈기 구조를 보여주며, [16]은 해당 논문에서 제안된 근사 컴프레서를 사용해 구현되었고, [17] 역시 자체적으로 제안된 근사 컴프레서를 활용하여 구현되었다. 그림 4. (d), (e), (f), (g)는 각각 [12], [19], [20], [11]에서 제안된 컴프레서를 사용한 근사 곱셈기를 나타낸다. 하위 4비트는 별도에 게이트를 사용하지 않고 절사하여 [12]와 [19]에서는 “0000”, [20]에서는 “0110”, [11]에서는 “0100”을 상수로 사용한다. 앞에서 설명한대로, Error Recovery Module은 정확한 컴프레서를 처음으로 사용하는 부분 최하위 열에 정확한 4−2 컴프레서의 Cin으로 연결되어 사용된다. [11]에서는 근사 컴프레서의 입력 X4가 “1”인 경우, 나머지 입력 값에 관계없이 컴프레서의 정확한 결과와 근사 결과의 차이, D = −1이 되기 때문에 Error Recovery Module을 사용하지 않고 근사 컴프레서의 입력 X4만을 사용하여 오류를 보상하였다.

III. 비교 분석

이 섹션에서는 앞서 소개한 근사 4−2 컴프레서들을 이용하여 각 근사 곱셈기에

대한 에러 분석, 합성 결과, 이미지 프로세싱 적용을 설명한다.

1. 에러 분석

8 비트 근사 곱셈기에 대해 총 65,536 가지 경우를 대상으로 다양한 오류 지표를 활용한 오류 분석이 수행되었다. 근사 곱셈기의 출력 품질은 오류 거리(Error Distance, ED)를 사용하여 평가된다. ED 는 i 번째 정확한 출력 값($Exact._i$)과 근사 출력 값($Appr._i$)의 차를 통해 구한다. 이때 발생하는 오류 거리의 최대 값을 최대 오류 거리(Maximum Error Distance, ED Max)라고 한다. 평균 오류 거리(Mean Error Distance, MED)는 모든 ED 의 평균을 나타내며 n 은 비트 수를 의미 한다. 평균 상대 오류 거리(Mean Relative Error Distance, MRED)는 ED 를 정확한 출력으로 나누어 계산되며 부정확한 곱셈기를 크기와 거의 무관하게 비교할 수 있도록 정규화 오류 거리(Normalized Error Distance, NED)도 포함된다. 이러한 지표를 통해 근사 곱셈기의 성능이 평가되었다. 식 (23), (24), (25), (26)은 각각 ED, MED, MRED, NED 에 대한 공식이다 [8].

$$ED = Exact._i - Appr._i \tag{23}$$

$$MED = \frac{1}{2^{2n}}\sum_{i=1}^{2^{2n}} |ED_i| \tag{24}$$

$$MRED = \frac{1}{2^{2n}}\sum_{i=1}^{2^{2n}} \frac{|ED_i|}{Exact._i} \tag{25}$$

$$NED = \frac{1}{(2^n-1)^2}\sum_{i=1}^{2^{2n}} \frac{|ED_i|}{2^{2n}} \tag{26}$$

표 1 에서는 다양한 근사 곱셈기의 에러 분석을 보여준다. 표 1 에서 Exact [9]는 정확한 계산을 수행하는 곱셈기이므로 모든 오류 지표에서 0 을 산출한다. [15]와 [16]은 [14]와 비교하여 각각 MED 는 98.73%, 95.93%, MRED 는 99.81%, 96.36%, NED 는 98.72%, 95.93% 개선되었다. 또한, [17]은 [14], [16]과 비교하여 각각 MED 는 98.24%, 56.84%, MRED 는 99.72%, 92.31%, NED 는 98.25%, 57% 개선되었다. [19]는 [12]보다 MED, MRED, NED 값이 각각 15.97%, 16.85%, 16.67% 향상되었음을 보여준다. [11]은 [12]와 [20] 비교했을 때, MED, MRED, NED 값이 크게 증가했음을 보여준다. ED MAX 의 값은 [14]가 가장 높고 [20]이 가장 낮게 기록되었다.

표 1. VARILOUS APPROXIMATE MULTIPLIERS 에러 분석

Design	MED	MRED	NED (10^{-3})	ED MAX
Exact [8]	0	0	0	0
[14]	3310.95	4.1495	50.92	8,896
[15]	42.10	0.0077	0.65	216
[16]	134.74	0.1509	2.07	584
[17]	58.15	0.0116	0.89	912
[12]	31.00	0.0089	0.48	321
[19]	26.05	0.0074	0.40	369
[20]	27.51	0.0069	0.42	219
[11]	58.91	0.0177	0.91	239

2. 합성 결과

다양한 근사 4-2 컴프레서와 근사 곱셈기의 합성은 Verilog HDL 을 사용하여 45-nm CMOS 공정에서 Synopsys Design Compiler [22]에 적용하여 수행되었다. 표 2 는 정확한 4-2 컴프레서와 다양한 근사 4-2 컴프레서의 합성 결과를 보여준다. [14]와 비교했을 때, [15]와 [16]은 각각 전력은 32.02%, 15.05% 감소하였고 지연 시간은 16.90%, 68.31% 증가하였다. 그리고 면적은 15.41% 감소하였고 2.93% 증가하였다. 반면, [17]은 [14]와 [16]에 비해 전력, 지연 시간, 면적이 모두 증가한 결과를 보여준다. [19]는 [12]와 비교하여 전력, 지연 시간, 면적이 각각 28.12%, 57.23%, 31.53% 증가하였다. [11]은 [12]와 [20]에 비해 전력은 각각 9.52%, 22.07%, 면적은 16.63%, 20.82% 개선되었으나, 지연 시간은 0.13% 증가하였고 0.26% 감소하였다. [19]는 평균 오류 확률에서는 가장 낮은 값을 보였지만, [12], [20], [11]에 비해 전력, 지연 시간, 면적이 크게 나타났다.

표 2. VARIOUS APPROXIMATE COMPRESSORS 합성 결과

Compressor	Power (μW)	Delay (ns)	Area (μm^2)	Avg. Error Prob.
Exact [8]	8.1305	0.3640	14.82	0
[14]	2.6206	0.0852	7.85	100/256
[15]	1.7815	0.0996	6.64	46/256
[16]	2.2261	0.1434	8.08	100/256
[17]	4.3316	0.1846	11.21	19/256
[12]	3.0775	0.1543	8.12	16/256
[19]	3.9428	0.2426	10.68	4/256
[20]	3.5731	0.1549	8.55	16/256
[11]	2.7845	0.1545	6.77	64/256

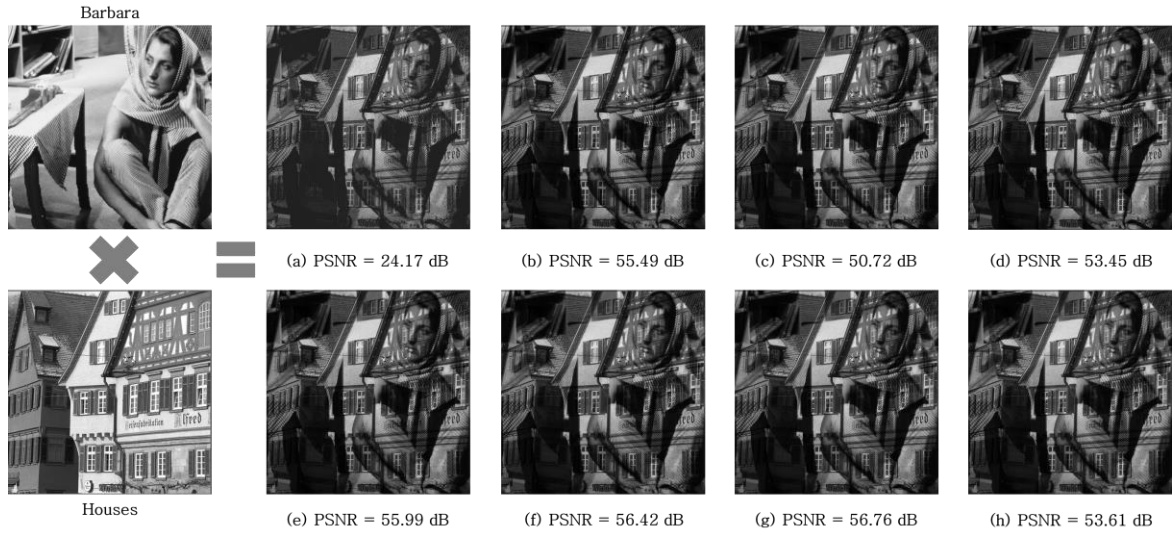

그림 5. Barbara (Image 1), Houses (Image 2)를 이용한 이미지 블렌딩 과정 (Image Blending Process)과 블렌딩된 이미지(blended image): (a) Image using [14]; (b) Image using [15]; (c) Image using [16]; (d) Image using [17]; (e) Image using [12]; (f) Image using [19]; (g) Image using [20]; (h) Image using [11].

표 3 은 정확한 곱셈기와 다양한 근사 곱셈기의 합성 결과를 보여준다. [14]와 비교했을 때 [15]와 [16]의 전력은 각각 54.24%, 38.18%, 면적은 24.52%, 18% 증가한 반면, 지연 시간은 동일했다. 컴프레서의 합성결과와 마찬가지로, [17]은 [14]와 [16]에 비해 전력, 지연 시간, 면적이 모두 증가한 결과를 나타냈다. [19]는 [12]와 비교했을 때 전력, 지연 시간, 면적이 각각 3.49%, 4.34%, 2.59% 증가하였다. [11]은 [12]와 [20]에 비해 전력이 각각 4.75%, 6.97%, 감소하고, 면적이 각각 3.66%, 4.52% 감소했으며, 지연 시간은 0.05% 증가하였고 0.33% 감소하였다.

표 3. VARIOUS APPROXIMATE MULTIPLIERS 합성 결과

Multiplier	Power (μW)	Delay (ns)	Area (μm^2)
Exact[8]	334	2.6766	521.27
[14]	165	2.4227	364.71
[15]	254.5	2.4227	454.15
[16]	228	2.4227	430.37
[17]	272.6	2.4297	458.48
[12]	238.1	2.1277	401.81
[19]	246.4	2.2200	412.21
[20]	243.8	2.1358	405.44
[11]	226.8	2.1288	387.11

3. 이미지 프로세싱 적용

오류에 강한 응용 프로그램에서 8비트 근사 곱셈기의 실용성을 입증하기 위해 이미지 블렌딩(Image Blending)을 사용하였다. 이 방법은 Xilinx Artix-7 FPGA(xc7a25tcsg325-3) 환경에서 Vivado 2024. 1 툴을 사용하여 구현되었다. 그림 5는 Barbara와 Houses 이미지를 사용한 이미지 블렌딩 과정과 그림 4에 있는 근사 곱셈기들을 각각 적용한 이미지를 보여준다. 피크 신호 대 잡음 비율(Peak Signal-to-Noise Ratio, PSNR)은 이미지 처리를 평가하는 데 널리 사용되는 척도이며, 이 값이 높을수록 이미지 품질이 우수함을 나타낸다. PSNR 값은 근사 곱셈기를 사용하여 블렌딩된 이미지와 정확한 곱셈기를 사용하여 블렌딩된 이미지를 비교하여 평가된다.

표 4에서는 다양한 근사 곱셈기에 PSNR 값과 평균 PSNR 값(Average, Avg.)을 보여준다. [14]와 비교했을 때, [15]와 [16]의 평균 PSNR 값은 증가하였다. 특히, [17]은 [16]의 평균 PSNR 값보다 7.74% 증가한 결과를 보였다. [19]는 [12]의 평균 PSNR 값보다 0.54% 증가했으며, [11]은 [12]와 [20]에 비해 각각 5.18%, 6.33% 낮은 평균 PSNR 값을 기록하였다. 표 1에

표 4. APPROXIMATE 4-2 COMPRESSOR 를 사용한 다양한 근사 곱셈기에 따른 이미지 프로세싱의 PSNR 값

Multiplier	Barbara Houses	Crowd Clown	Lighthouse Couple	Girlface Bridge	Avg.
[14]	24.17	20.22	24.02	20.44	22.21
[15]	55.49	56.25	55.16	54.44	55.34
[16]	50.72	49.47	50.08	48.13	49.60
[17]	53.45	52.70	53.20	54.40	53.44
[12]	55.99	56.64	55.87	55.41	55.98
[19]	56.42	56.73	56.09	55.86	56.28
[20]	56.76	57.15	56.55	56.23	56.67
[11]	53.61	53.56	52.71	52.42	53.08

결과에 따르면, [19]는 [20]에 비해 MED, NED 값은 더 낮게 나타났으나 MRED, ED MAX 값은 더 높게 기록되었다. 이러한 이유로 [19]의 평균 PSNR 값이 [20]보다 낮게 나타난 것으로 분석된다.

V. 결 론

본 논문에서는 여러 가지 8비트 근사 곱셈기의 성능을 다양한 오류 지표와 합성 결과를 통해 평가하고, 이를 이미지 프로세싱에 적용한 결과를 분석하였다. 오류 지표 분석 결과, [14]의 곱셈기는 다양한 오류 지표에서 가장 높은 값을 기록하여 평균 PSNR 값이 가장 낮게 나타났다. 반면, [19]의 컴프레서는 평균 오류 확률에서 [20]보다 우수했지만, MRED와 ED MAX에서 [20]의 곱셈기가 더 개선된 성능을 보였기 때문에, [20]의 평균 PSNR 값이 더 높게 측정된 것으로 분석되며, 다른 근사 곱셈기들과 비교했을 때도 가장 높은 값을 기록했다. 컴프레서와 곱셈기의 합성 결과에서 [15]의 컴프레서는 [14]와 비교하여 전력 소비와 면적이 각각 32.02%, 15.41% 감소하며 효율성을 높였으나, 지연 시간이 16.90% 증가하는 결과를 보였다. 하지만 [15]의 곱셈기는 컴프레서의 배치 방식 차이로 인해 [14]의 곱셈기보다 전력 소비와 면적이 증가했다. 한편, [19]의 컴프레서는 [12], [20], [11]에 비해 전력, 지연 시간, 면적 모두 증가한 결과를 보였으며, [19]의 곱셈기 역시 [12], [20], [11]과 동일한 구조를 사용했기 때문에 곱셈기의 합성 결과에서도 마찬가지로 전력, 지연 시간, 면적이 상대적으로 크게 나타났다. 결과적으로, 에너지 제약이 중요한 IoT에서는 전력 소모가 가장 적은 [14]의 근사 곱셈기를 사용하는 것이 유리하며, 고속 처리가 중요한 신호 처리에서는 지연 시간이 가장 빠른 [12]의 근사 곱셈기를 사용하는 것이 적합하다. 또한, 크기 제한이 있는 모바일 디바이스에서는 면적이 가장 작은 [14]의 근사 곱셈기를 사용하는 것이 효과적이고, 높은 연산 정확도를 요구하는 의료 기기에서는 정확도가 가장 높은 [20]의 근사 곱셈기를 사용하는 것이 바람직하다.

감사의 글

This work was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0012451, The Competency Development Program for Industry Specialist). This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2022-00156225) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). The EDA tool was supported by IC Design Education Center (IDEC), Korea.

참고 문헌

[1] Vidhyalakshmi. M, Padmapriya. S, “High Throughput and Less Error Approximate Compressor design for Partial Product Reduction in Multipliers”, in *2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC)*, 2024.

[2] Z. Yang, J. Han, F. Lombardi, “Approximate Compressors for Error-Resilient Multiplier Design”, in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2015.

[3] K. Naresh, Y. P. Sai, S. Majumdar, “Design of 8-bit Dadda Multiplier using Gate Level Approximate 4:2 Compressor”, in *2022 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID)*, 2022.

[4] L. H. Krishna, Ayesha Sk, J. B. Rao, et al., “Energy-Efficient Approximate Multiplier Design With Lesser Error Rate Using the Probability-Based Approximate 4:2 Compressor”, IEEE *Embedded Systems Letters*, vol. 16, no. 2, pp. 134-137, May 2023.

[5] M. Zhang, S. Nishizawa, S. Kimura, “Area Efficient Approximate 4-2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier”, IEEE *Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 5, pp.1714-1718, Mar. 2023.

[6] S. Venkatachalam, 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, Jan. 2017.

[7] E. L. Hall, D. D. Lynch and S. J. Dwyer, “Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications”, in IEEE *Transactions on Computers*, vol. C-19, no.2, pp. 97-105, Feb. 1970.

[8] Y. J. Chang, et al., “Imprecise 4-2 compressor design used in image processing applications”, in IET *Circuits, Devices & Systems*, vol. 13, no. 6, pp. 848-856, Apr. 2019.

[9] A. G. M. Strollo, E. Napoli, et al., “Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers”, in IEEE *Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 9, pp. 3021-3034, Sep. 2020.

[10] U. A. Kumara, P. Bikki, et al., “Power Efficient Approximate Multiplier Architectures for Error Resilient Applications”, in *2022 IEEE 19th India Council International Conference (INDICON)*, 2023.
[11] S. Kim, H. Seo and S. Kim, D. Kim, “Approximate Multiplier With Efficient 4-2 Compressor and Compensation Characteristic”, in *Journal of the KIECS*, vol. 17, no. 1, pp. 173-180, Feb. 2022.
[12] M. Ha and S. Lee, “Multipliers With Approximate 4-2 Compressors and Error Recovery Mod-ules”, in IEEE *Embedded System Letters*, vol. 10, no. 1, pp. 6-9, Mar. 2018.
[13] B. Fang, H. Liang, D. Xuin, et al., “Approximate multipliers based on a novel unbiased approximate 4-2 compressor”, in *Integration, the VLSI journal*, vol. 81, pp. 17-24, Nov. 2021.
[14] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, “Design and Analysis of Approximate Compressors for Multiplication”, in IEEE *Trans-actions on Computers*, vol. 64, no. 4, pp. 984-994, Apr. 2015.
[15] J. Gu and Y. Kim, “Design and Analysis of Approximate 4-2 Compressor for Efficient Multiplication” in IEIE *Transactions on Smart Processing and Computing*, vol. 11, no. 3, pp. 162-168, June 2022.
[16] H. Seok, H. Seo, J. Lee, and Y. Kim, “Design Optimization of a 4:2 Compressor for Low-cost Approximate Multipliers”, in IEIE *Transactions on Smart Processing and Computing*, vol. 11, no. 6, pp. 455-461, Dec. 2022.
[17] S. Jeon, J. Jeon, Y. Lee, and Y. Kim, “New Approximate 4:2 Compressor for High Accuracy and Small Area Using MUX Logic”, in *2024 International Conference on Electronics, Information, and Communication (ICEIC)*, 2024.
[18] Z. Yang, J. Han, and F. Lombardi, “Approximate Compressors for Error-Resilient Multiplier Design”, in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 183–186, 2015.
[19] A. G. M. Strollo, D. D. Caro, et al., “Low-Power Approximate Multiplier With Error Recovery Using a New Approximate 4-2 Compressor”, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020.
[20] U. A. Kumar, S. K. Chatterjee, and S. E. Ahmed, “Low-Power Compressor-based Approximate Multipliers with Error Correcting Module,” in IEEE *Embedded Systems Letters*, vol. 14, no. 2, pp. 59-62, June 2022.
[21] S. Hwang, H. Seok, and Y. Kim, “Design of an Approximate 4-2 Compressor with Error Recovery for Efficient Approximate Multiplication” in *Journal of Semiconductor Technology and Science (JSTS)*, vol. 24, no. 4, pp. 305-315, Aug. 2024.
[22] Silvaco Inc., “Nangate 45nm Open Cell Library.” [Online]. Available: https://silvaco.com

전종민 (Jongmin Jeon), 학생회원

2023년 2월 : 안양대학교 정보전기전자공학과 학사 졸업
2023년 9월 ~ 현재: 홍익대학교 전자전기공학과 석사 과정

<관심분야> 디지털 회로 설계, 근사 컴퓨팅

김영민 (Youngmin Kim), 정회원

1999년 8월 : 연세대학교 전자공학과 학사 졸업
2003년 4월 : 미시건대학교 EECS 석사 졸업
2007년 12월 : 미시건대학교 EECS 박사 졸업

2009~2015년 : UNIST 전자컴퓨터공학부 조교수
2015년~2019년 : 광운대학교 컴퓨터공학부 부교수
2019년~현재 : 홍익대학교 전자전기공학부 교수

<관심분야> 디지털 회로 설계, CAD VLSI