고효율 디지털 신호처리를 위한 근사 곱셈기 설계

하민호
포항공과대학교
전자전기공학과

이영주
포항공과대학교
전자전기공학과

이승구
포항공과대학교
전자전기공학과

Ⅰ. 서 론

최근 몇 년 사이에 정보통신기술 (ICT) 융합을 바탕으로 엄청난 양의 data 처리와 생성을 필요로 하는 이른바 4차 산업혁명이 발생하였다. 2017년 1월 라스베이거스에서 열린 Consumer Electronics Show (CES) 2017을 보면 4차 산업혁명이 우리 곁에 빠르게 다가오고 있고, 더 나아가 어떤 면에서는 이미 우리 생활 깊숙이 들어와 있음을 알 수 있다. CES 2017에서는 인공지능 (AI), 로봇, 사물 인터넷 (IoT) 등 4차 산업혁명을 대표하는 최신 기술들이 소개되었다. 4차 산업혁명을 대표하는 기술들은 공통적으로 아날로그 신호들을 감지하고, 이 신호들을 어플리케이션에서 처리하기 용이하도록 디지털 신호로 변환하는 과정이 필요하다. 변환된 디지털 신호는 용도에 맞게 디지털 신호처리 과정을 거치게 된다. 이러한 디지털 신호처리를 효율적으로 하기 위해서는 저전력/고성능 디지털 신호처리 프로세서가 필수적이다.

디지털 신호처리를 위해서는 복잡하고 많은 연산이 필요하다. 특히 곱셈은 디지털 신호처리에 필수적인 연산이다. 따라서 디지털 신호처리 프로세서에서 곱셈기는 가장 중요한 하드웨어 연산기 중 하나라 할 수 있다. 하지만 곱셈기는 일반적으로 energy-hungry 특성을 가지는 연산기로 알려져 있다[1]. 따라서 고효율 디지털 신호처리 프로세서 설계를 위해서는 저전력으로 동작하면서 고성능을 보이는 곱셈기 설계가 선행되어야 한다.

효율적인 디지털 신호처리를 위한 저전력/고성능 곱셈기 설계와 관련해서 많은 연구들이 진행되고 있다. 특히 기존의 100% 정확도를 필요로 하는 컴퓨팅 패러다임에서 벗어나 최종 결과값에 큰 영향을 주지 않는 약간의 오차를 허용하는 근사 컴퓨팅 기법이 하나의 대안적인 패러다임으로 부각되면서[2]-[4], 곱셈기 설계에도 근사 컴퓨팅 기법을 적

용하는 사례들이 증가하고 있다[1],[5]–[16]. 디지털 신호처리 어플리케이션의 경우 허용가능한 오차가 결과에 큰 영향을 주지 않거나, 디지털 신호처리 전 아날로그 신호 자체의 노이즈로 인해 디지털 신호처리 프로세서의 입력이 원래부터 부정확한 경우가 많다. 이는 근사적으로 연산 결과를 출력하는 것이 최종 결과값에 크게 영향을 주지 않을 수 있다는 것을 의미한다.

본 논문에서는 근사 컴퓨팅 기법의 개념을 간략히 살펴보고, 근사 컴퓨팅 기법을 저전력/고성능 곱셈기를 설계에 적용한 최신 연구에 대해 알아본다. 또한 근사 곱셈기가 실제 디지털 신호처리 어플리케이션에 어떻게 적용되었는가를 알아보고, 이를 통해 디지털 신호처리 프로세서에 근사 곱셈기를 사용함으로써 얻을 수 있는 효과를 알아보고자 한다.

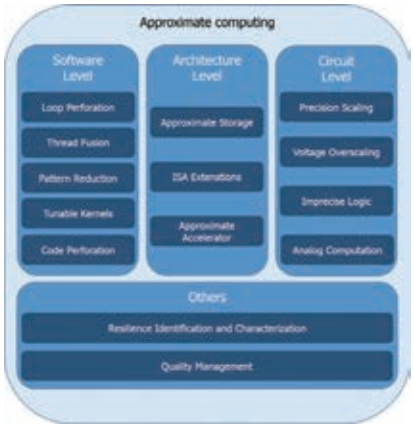
Ⅱ. 근사 컴퓨팅

근사 컴퓨팅 (approximate computing)은 디지털 시스템을 에너지 효율적으로 설계하는 방법 중 하나로 각광받고 있다[2]. 근사 컴퓨팅은 계산된 결과값의 손실을 일정 부분 허용하는 대신 전력 소모 및 연산 속도와 같은 하드웨어 특성에서 이득을 얻는 컴퓨팅 기법이다. 다행히 많은 디지털 신호처리 어플리케이션들은 출력되는 결과값의 정확도에 손실이 일부 있더라도 오차가 일정 수준만 넘지 않으면 의미 있는 결과값으로 사용이 가능한 경우가 많다. 따라서 디지털 신호처리 어플리케이션은 근사 컴퓨팅 기법 적용에 용이하다고 볼 수 있다.

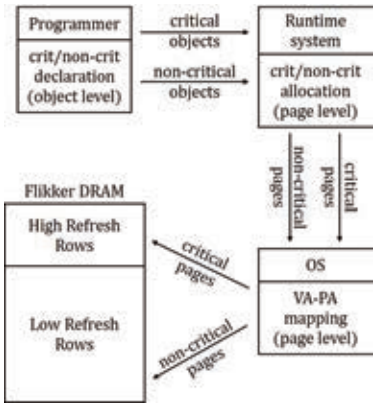
근사 컴퓨팅을 분류하는 방법은 다양하지만, 근사 컴퓨팅 기법이 적용되는 계층에 따라 분류하는 것이 일반적이다[4]. 근사 컴퓨팅을 적용 계층에 따라 분류하면 〈그림 1〉에서 보듯 1) 소프트웨어 레벨, 2) 아키텍처 레벨, 3) 회로 레벨로 나눌 수 있다.

대표적인 소프트웨어 레벨 근사 컴퓨팅 기법으로는 loop perforation[17]이 있다. Loop perforation은 프로그램 성능 향상을 위해 프로그램 내 약간의 품질저하가 발생해도 괜찮은 부분의 loop를 기존의 반복 횟수보다 적게 반복하여 성능을 향상시키는 기법이다. 실험 결과를 보면 최종 결과값의 품질 손실이 10% 미만으로 유지되면서 어플리케이션의 성능을 2~3배 증가시킬 수 있음을 보였다.

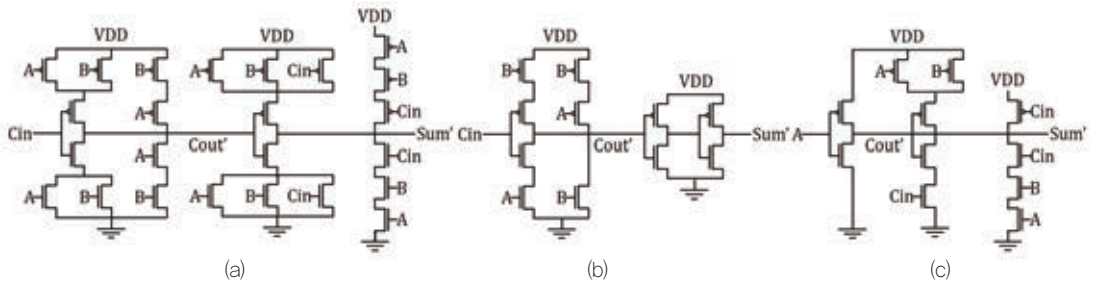
아키텍처 레벨에서의 대표적인 근사 컴퓨팅 기법 중 하나는 메모리 시스템에 근사 컴퓨팅 기법을 적용하여 저전력 동작을 구현하는 것이다. 그 중에서 가장 널리 알려진 기법 중 하나인 Flikker라는 기법을 소개한다[18]. Flikker는 DRAM의 refresh 주기를 조절하여 불필요한 전력소모를 줄이는 기법이다. DRAM을 구성하는 bank를 critical data를 저장할 bank와 non-critical data를 저장할 bank로 나눈 후, critical data가 저장된 bank에는 기존의 refresh 주기를 유지하고, non-critical data가 저장된 bank에는 기존의 refresh 주기보다 긴 주기로 refresh를 시키는 것이 핵심이다(〈그림 2〉 참고). 이를 통해 DRAM 전력 소모에서 큰 부분을 차지하는 refresh에 의한 전력 소모를 줄일 수 있음을 보였다.



〈그림 1〉 계층에 따른 근사 컴퓨팅 기법 분류[4]



〈그림 2〉 Flikker 시스템 개요[18]



〈그림 3〉 (a) 전통적인 mirror adder, (b) 근사화된 mirror adder 1, (c) 근사화된 mirror adder2[19]

부정확한 로직을 이용하여 연산기를 설계하는 방식은 회로 레벨 근사 컴퓨팅 기법 중 널리 쓰이는 방법 중 하나이다. 트랜지스터 레벨에서의 근사를 통해 가산기 셀을 설계하는 방식인 IMPACT가 대표적이다[19](〈그림 3〉 참고). IMPACT는 기존의 정확한 mirror adder를 트랜지스터 레벨에서 근사시킨 후, least significant bits (LSBs)에 근사 가산기 셀을 적용하였다. LSB에만 적용했기 때문에 결과값에 큰 손실없이 전력 소모를 줄일 수 있음을 보였다.

이외에도 error-resilient 부분을 판별하는 기법[19]이나 시스템 관점에서 전체적인 품질 관리하는 기법[20] 등에 대한 연구들이 진행되고 있다.

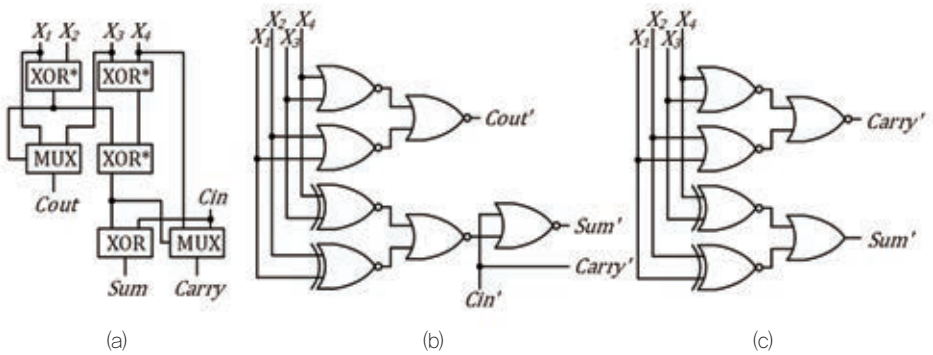
Ⅲ. 근사 곱셈기 설계 연구

최근까지 근사 곱셈기 설계와 관련된 여러 연구들이 수행되어 왔다[1],[5]-[16]. 본 논문에서는 그 중 대표적인 방법인 근사 4:2 compressor 기반 곱셈기와 근사 modified Booth 곱셈기에 대해 알아보고자 한다.

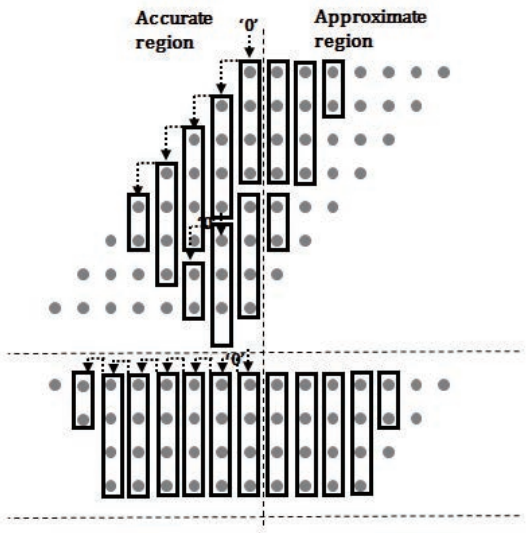
3.1. 근사 4:2 compressor 기반 근사 곱셈기 설계

곱셈기는 크게 세 부분으로 구성된다: 1) 부분곱 생성, 2) 부분곱 압축, 3) 최종 덧셈. 부분곱을 압축하는 과정이 곱셈 과정 전체에서 전력 소모와 성능을 결정하는데 큰 영향을 미친다. 이러한 부분곱 압축 과정을 효과적으로 수행하기 위해 4:2 compressor를 사용하는 방식이 제안되었다[21]. 더 나아가 기존의 정확한 4:2 compressor의 로직을 단순화하여 근사 4:2 compressor를 설계하고, 이 근사 4:2 compressor를 바탕으로 근사 곱셈기를 설계하는 연구들이 활발히 진행되어 왔다[9]-[11],[15].

2015년 IEEE Trans. Comput.에 실린 "Design and analysis of approximate compressors for multiplication"[10]에서 근사 4:2 compressor 개념이 처음 제안되었다. 이 논문에서는 기존의 정확한 4:2 compressor의 진리표를 수정하여 두 가지 버전의 근사 4:2 compressor를 제안하였다. 두 가지 버전의 근사 4:2 compressor (〈그림 4〉 참고)를 8-bit Dadda 곱셈기에 적용(〈그림 5〉 참고)하여 기존의 정확한 곱셈기와의 성능을 비교하였다.



〈그림 4〉 (a) 정확한 4:2 compressor[21], (b) 근사 4:2 compressor 디자인 1[10], (c) 근사 4:2 compressor 디자인 2[10]

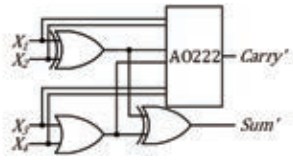


〈그림 5〉 정확한 4:2 compressor와 근사 4:2 compressor로 구성된 8-bit Dadda 곱셈기[11] ([10]: 8-bit accurate region, 7-bit approximate region으로 설계, [11]: 7-bit accurate region, 4-bit approximate region, 4-bit truncated region으로 설계)

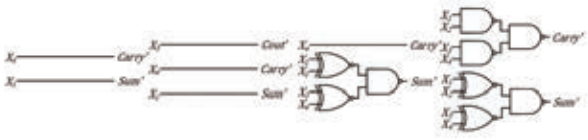
그 결과, 지연시간은 3.38~26.52% 향상, 전력소모는 17.50~58.58% 향상, 트랜지스터 개수는 14.03~48.15% 향상이 되었음을 알 수 있었다. Normalized error distance (NED)는 0.7827×10^{-3}~0.6065×10^{-1}로 나오는 것을 확인할 수 있었다. [9]에서는 [10]에서 제안된 근사 4:2 compressor를 그대로 사용하되 recursive multiplication을 기법을 도입하여 전체 부분곱 압축 부분을 작게 쪼개는 방법을 채택하였다.

이후 발표된 [11]에서는 [10]에서 고려되지 않았던 근사 4:2 compressor의 오차율까지 고려하여 낮은 오차율의 근사 4:2 compressor를 설계하는 방법을 제안하였다 (〈그림 6〉 참고). [10]에서 제안된 근사 4:2 compressor의 오차율은 25/64인 반면에 [11]에서 제안된 근사 4:2 compressor의 오차율은 1/256~1/16으로 [11]에서 제안된 방식이 [10]에서 제안된 방식에 비해 낮은 오차율을 보여주었다. 하드웨어 특성 (전력소모, 지연시간, 면적)을 보면 기존의 다른 근사 곱셈기들과 비교했을 때 전체적으로 더 나은 성능을 보여주었다.

가장 최근 연구인 [15]는 근사 4:2 compressor의 정확도를 고정시켜서 사용했던 기존 연구들과는 달리 동적으로 근사 4:2 compressor의 정확도를 변경시킬 수 있



〈그림 6〉 근사 4:2 compressor 디자인 3[11]



〈그림 7〉 dual-quality 4:2 compressor의 근사 부분: (a) 디자인 1, (b) 디자인 2, (c) 디자인 3, (d) 디자인 4

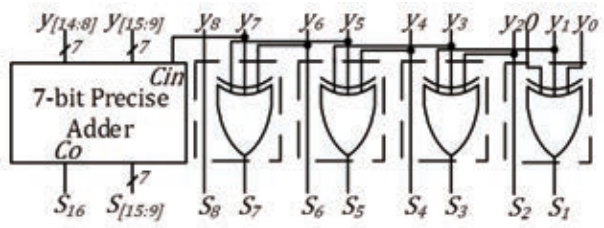
는 dual-quality 4:2 compressor를 제안했다. 근사 부분의 근사화 정도에 따라 총 4가지 버전의 디자인을 제안하였다 (〈그림 7〉 참고). 32-bit Dadda 곱셈기에 제안된 dual-quality 4:2 compressor를 적용하였을 때, 제안된 디자인이 평균 46% 지연시간 향상 및 68% 전력소모 향상이 된다는 결과를 보였다.

3.2. 근사 Modified Booth 곱셈기 설계

부분곱의 개수를 줄이는 방법도 곱셈기의 성능을 향상시킬 수 있는 방법이다. 부분곱을 줄이는 대표적인 방법이 Booth encoding이다. Modified Booth 곱셈기 설계에 근사 컴퓨팅 기법을 적용하여 encoding의 복잡성을 완화시킴으로써 면적, 전력소모, 지연시간을 향상시키는 연구들이 제안되었다[13], [16].

[13]에서는 피승수의 홀수 배를 생성하는 복잡성으로 인해 속도가 느린 radix-8 Booth 곱셈기에 근사 컴퓨팅 기법을 적용하였다. 추가적으로 캐리 전파가 없이 트리플 피승수를 생성하기위한 기록 가산기의 덜 중요한 부분을 구현하기 위해 2-bit 가산기를 사용하였다 (〈그림 8〉 참고). 제안된 근사 곱셈기는 정확한 Booth 곱셈기보다 빠르며 전력 효율이 높다는 것을 보였다. 또한 15-bit truncation이 있는 곱셈기는 다른 근사 Booth 곱셈기 설계와 비교할 때 하드웨어 및 정확도면에서 전체적으로 가장 좋은 성능을 보였다.

[16]에서는 근사 radix-4 modified Booth encoding (MBE)과 근사 Wallace tree를 이용한 부분곱 어레이



〈그림 8〉 근사적인 8-bit을 가진 근사 기록 가산기[13]

〈표 1〉 16-bit 근사 Modified Booth 곱셈기 성능 비교 (NanGate 45 nm Open Cell Library 사용)[16]. Product-delay product (PDP), Normalized mean error distanc (NMED), Error rate (ER)

16-bit MBE		Power (μW)	Delay (ns)	Area (μm²)	PDP (pJ)	NMED (10^{-5})	ER (%)
[13]	R8ABM1	376.7	1.23	1516	0.463	1.92	44.06
	R8ABM2-C9	332.6	1.22	1332	0.406	4.43	99.74
	R8AMB2-C15	217.3	1.18	912	0.256	5.73	99.99
[16]	R4ABM1 (p=12)	535.0	0.95	2209	0.508	0.31	85.72
	R4ABM1 (p=14)	516.6	0.95	2169	0.49	0.93	93.75
	R4ABM1 (p=16)	479.9	0.94	2066	0.451	3.10	93.98
	R4ABM1 (p=18)	448.4	0.94	2002	0.421	12.29	94.04
	R4ABM2 (p=12)	515.4	0.94	2077	0.484	0.27	98.49
	R4ABM2 (p=14)	479.7	0.92	2004	0.441	0.62	99.61
	R4ABM2 (p=16)	448.7	0.92	1875	0.412	3.02	99.86
	R4ABM2 (p=18)	437.4	0.89	1788	0.389	11.24	99.96

를 기반으로 근사 Booth 곱셈기를 설계하는 방법을 제안하였다. 두 가지 방식의 근사 radix-4 MBE가 제안되었다. 지연시간, 면적 및 전력소모와 같은 하드웨어적 특성과 근사 곱셈기의 오차를 모두 고려할 때 제안된 근사 radix-4 MBE 기반 곱셈기가 가장 효율적임을 보였다. 〈표 1〉은 [13]과 [16]에서 제안된 곱셈기의 성능을 비교한 표이다.

본 논문에서는 대표적인 두 가지 근사 곱셈기 설계 기법만을 보였으나, 이외에도 많은 근사 곱셈기 설계 방식들이 있음을 알려둔다.

Ⅳ. 디지털 신호처리 과정에서 근사 곱셈기의 활용 예

대표적인 디지털 신호처리 어플리케이션으로 이미지 처리를 들 수 있다. 앞에서 설명한 논문들을 비롯한 많은 논문들에서 근사 곱셈기의 실질적 성능을 보이기 위해 다양한 이미지 처리 어플리케이션 결과를 이용하였다. 대표적인 이미지 처리 어플리케이션 예를 통해 근사 곱셈기가 실질적인 디지털 신호처리 프로세서에 활용 가능성이 크다는 것을 보이고자 한다.

(a) (b) (c)

〈그림 9〉 (a) 이미지 합성 과정, (b) 기존의 곱셈기를 이용한 이미지 합성 결과, (c) 근사 곱셈기 ([10]의 디자인 4)를 이용한 이미지 합성 결과

4.1. 이미지 합성 (Image multiplication)

[10], [15] 및 [16]에서는 이미지 처리 어플리케이션으로 이미지 합성을 택했다. 이미지 합성이란 사이즈가 같은 두 이미지를 동일 위치의 픽셀끼리 곱해서 이미지를 합치는 과정이다. 즉, 이미지 합성은 여러 번의 곱셈이 필요한 어플리케이션이므로 곱셈기의 성능을 잘 보여줄 수 있다. [10]에서 제안한 근사 4:2 compressor 기반 곱셈기를 이용해 이미지 합성을 한 결과는 아래 〈그림 9〉과 같다.

〈그림 9〉를 보면 알 수 있듯이 기존의 정확한 곱셈기를 이용한 이미지 합성 결과와 근사 곱셈기를 이용한 이미지 합성 결과를 육안으로 보았을 때 구분하기 힘들다는 것을 알 수 있다. 근사 곱셈기와 정확한 곱셈기의 하드웨어 특성(CMOS 32 nm 공정)을 비교해보았을 때 전력소모가 26.15% 향상되었고, 트랜지스터 개수는 22.42% 향상되었다. 따라서 이미지 합성처럼 약간의 오차가 발생해도 최종 결과를 사용하는데 큰 문제가 없는 경우라면 근사 곱셈기를 사용하는 것이 큰 이득이라는 것을 알 수 있다.

4.2. 이미지 필터 (Image filter)

[9], [11] 및 [15]에서는 이미지 처리 어플리케이션으로 이미지 필터 중 하나인 선명화 필터를 택했다. 이미지 선명화 필터는 아래와 같이 동작한다.

$$Y(i,j)=2X(i+m,j+n)-\frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2}X(i+m,j+n)Mask_{sharpening}(m+3,n+3) \quad (1)$$

$$Mask_{sharpening}=\begin{bmatrix}1&4&7&4&1\\4&16&26&16&4\\7&26&41&26&7\\4&16&26&16&4\\1&4&7&4&1\end{bmatrix} \quad (2)$$

[11]에서 제안된 근사 4:2 compressor 기반 근사 곱셈기를 이용해 이미지 선명화 필터를 한 결과는 아래 〈그림 10〉과 같다.

〈그림 10〉을 보면 알 수 있듯이 이미지 합성에서와 마찬가지로 기존의 정확한 곱셈기를 이용한 이미지 선명화 필터 결과와 근사 곱셈기를 이용한 이미지 선명화 필터 결과를 육안으로 보았을 때 구분하기 힘들다는 것을 알 수 있다. 근사 곱셈기와 정확한 곱셈기의 하드웨어 특성(CMOS 65 nm 공정)을 비교해보았을 때 전력소모가 29.22% 향상되었고, 면적은 21.54% 향상되었다. 이미지 선명화 필터 어플리케이션에서도 근사 곱셈기가 유용할 수 있음을 알 수 있다.

Ⅴ. 결 론

AI, 로봇, IoT 등 4차 산업혁명을 대표하는 최신 기술들이 발전됨에 따라 더 많은 분야에서 고효율 디지털 신호처리를 필요로 할 것이다. 고효율 디지털 신호처리를 위해서는 디지털 신호처리만을 따로 전담하는 전용 프로세서가 필요하다. 많은 디지털 신호처리 어플리케이션들에서 곱셈은 가장 중요한 연산 중 하나이므로 디지털 신호처리 프로세서를 구성하는 연산 블록 중 곱셈기는 중요한 역할을 차지하는 것은 자명하다. 따라서 고효율 디지털 신호처리 프로세서설계를 위해서는 저전력/고성능 곱셈기 설계에 대한 연구도 수반되어야 한다.

〈그림 10〉 (a) 이미지 선명화 필터 예시, (b) 기존의 곱셈기를 이용한 이미지 선명화 필터 결과, (c) 근사 곱셈기 ([11]의 디자인 3)를 이용한 이미지 선명화 필터 결과

이러한 요구에 맞춰 최근 들어 저전력/고성능 곱셈기 설계에 근사 컴퓨팅 기법을 도입하는 연구가 늘어나고 있다. 많은 디지털 신호처리 어플리케이션들이 입력에 노이즈가 섞여있거나, 결과값이 반드시 정확할 필요가 없다는 성질을 가지고 있기 때문에 디지털 신호처리 프로세서에 이용되는 곱셈기 설계에 근사 컴퓨팅 기법을 적용하는 것이 가능하다.

여러 근사 곱셈기 연구를 통해 근사 곱셈기가 기존의 일반적인 곱셈기와 비교하였을 때 전력소모, 지연시간, 면적 등과 같은 하드웨어 특성에서 많이 향상되었음을 볼 수 있었다. 또한 실제 디지털 신호처리 어플리케이션에 사용하였을 때 근사 곱셈기를 이용해서 얻은 결과값이 어플리케이션 사용에 큰 지장이 없을 정도의 오차만 발생하는 것을 확인할 수 있었다.

앞서 이야기한 것처럼 기술이 발전할수록 더 많은 디지털 신호처리가 필요할 것이다. 특히 가면 갈수록 모바일이나 임베디드 환경에서의 디지털 신호처리가 증가할 것으로 예상된다. 따라서 고효율 디지털 신호처리 프로세서 설계 연구는 필수적이며, 이를 위해서는 저전력/고성능 곱셈기에 대한 연구도 계속적으로 증가할 것으로 생각된다.

참 고 문 헌

[1] S. Narayanamoorthy et al., "Energy-efficient approximate multiplication for digital signal processing and clsssification applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2015.

[2] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," Proc. European Test Symposium (ETS), 2013.

[3] S. Mittal, "A survey of techniques for approximate computing," ACM Comput. Surv., 2016.

[4] Q. Xu et al., "Approximate computing: a survey," IEEE Design & Test, 2016.

[5] K. Bhardwaj et al., "Power- and area-efficient approximate wallace tree multiplier for error-resilience systems", Proc. Int.

Symp. Quality Electronic Design (ISQED), 2014.

[6] B. Shao and P. Li, "Array-based approximate arithmetic computing: a general model and applications to multiplier and squarer design," IEEE Trans. Circuits Syst. I, Reg. Papers, 2015.

[7] G. Zervakis et al., "Hybrid approximate multiplier architectures for improved power-accuracy trade-offs," Proc. IEEE/ACM Int. Symp. Low Power Electronics and Design (ISLPED), 2015.

[8] S. Hashemi et al., "DRUM: A dynamic range unbiased multiplier for approximate applications," Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 2015.

[9] N. Maheshwari et al., "A Design approach for compressor based approximate multipliers," Proc. Int. Conf. VLSI Design (VLSID), 2015.

[10] A. Momeni et al., "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., 2015.

[11] Z. Yang et al., "Approximate compressors for error-resilient multiplier design," Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), 2015.

[12] M. Shafique et al., "Cross-Layer approximate computing: from logic to architectures," Proc. ACM/EDAC/IEEE Design Automation Conf. (DAC), 2016.

[13] H. Jiang et al., "Approximate radix-8 booth multipliers for low-power and high-performance operation," IEEE Trans. Comput., 2016.

[14] R. Zendegani et al., "RoBA multiplier: a rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2017.

[15] O. Akbari et al., "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2017.

[16] W. Liu et al., "Design of approximate radix-4 booth multipliers for error-tolerant computing," IEEE Trans. Comput., 2017.

[17] S. Misailovic et al., "Quality of service profiling," Proc. ACM/IEEE International Conference on Software Engineering (ICSE), 2010.

[18] S. Liu et al., "Flikker: Saving refresh-power in mobile devices through critical data partitioning," Proc. Int. Conf. Architect. Support Programm. Lang. Oper. Syst. (ASPLOS), 2011.

[19] A. Sampson et al., "EnerJ: Approximate data types for safe and general low-power computation," Proc. Int. Conf. Programm. Lang. Design Implement. (PLDI), 2011.

[20] M. Samadi et al., "SAGE: Self-tuning approximation for graphics engines," Proc. Int. Symp. Microarchitect. (MICRO), 2013.

[21] C.-H. Chang et al., "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits Syst. I, Reg. Papers, 2004.

하민호

• 2015년 2월 성균관대학교, 전자전기공학부 학사졸업
• 2017년 2월 포항공과대학교, 전자전기공학과 석사졸업
• 2017년 3월 ~ 현재 포항공과대학교 전자전기공학과 박사과정

〈관심분야〉
Embedded System Architecture, Approximate Computing

이영주

- 2008년 2월 KAIST, 전기 및 전자공학과 학사졸업
- 2010년 2월 KAIST, 전기 및 전자공학과 석사졸업
- 2014년 2월 KAIST, 전기 및 전자공학과 박사졸업
- 2014년 5월~2015년 2월 벨기에 IMEC 연구원
- 2015년 3월~2017년 2월 광운대학교 전자공학과 조교수
- 2017년 2월~현재 포항공과대학교 전자전기공학과 조교수

〈관심분야〉
Embedded SoC 설계 및 최적화

이승구

- 1985년 2월 University of Kansas, Electrical Engineering 학사졸업
- 1987년 2월 University of Michigan, Ann Arbor, Electrical Engineering and Computer Science 석사졸업
- 1990년 2월 University of Michigan, Ann Arbor, Electrical Engineering and Computer Science 박사졸업
- 1990년 3월~1990년 10월 University of Delaware 조교수
- 1991년 6월~현재 포항공과대학교 전자전기공학과 교수

〈관심분야〉
Mobile Ad-Hoc Networks, Parallel and Distributed Computing, Real-Time Computing, Fault-Tolerant Computing, ASIC Design