2025년 한국계산과학공학회 추계학술대회

복소수 진법 체계 기반 ALU 설계: 성능 평가 및 응용 방안

(Design of ALU Based on Complex-base Numeral System: Performance Evaluation and Application Methods)

권도한[1]

[1]안동고등학교 (Mail: me@dohan.in)

초 록

1960년 Donald Knuth가 처음 제안한 복소수 진법(An Imaginary Number System, DOI: 10.1145/367177.367233)은 복소수를 기수(Radix)로 사용하는 새로운 수 체계로, 기수로 어떤 복소수를 선택하느냐에 따라 숫자 집합과 표현법이 달라진다. 복소수 진법의 장점은 실수부와 허수부를 분리해 계산하지 않고 하나의 수로써 복소수 연산을 수행할 수 있다는 점이며, 이는 복소수 연산의 처리 속도 향상에 기여할 수 있다. 이후 연구에서는 다양한 기수를 탐구하면서 모든 복소수를 표현할 수 있는 체계를 확립하였고, 표현에 0과 1만을 사용하는 체계를 CBNS(Complex Binary Number System)로 정의하여 이진수와 유사한 표현을 갖게 되면서 컴퓨터 과학적 응용 가능성이 확대되었다. 본 논문은 국내에서 상대적으로 연구가 활발하지 않은 복소수 진법을 대상으로 수학적 이론을 정리하고, 복소수 연산에 특화된 ALU 설계 및 하드웨어 구현 방안을 제시한다. 설계한 ALU의 성능을 기존 방법과 비교·평가하고, 복소수 진법의 특성을 활용한 응용 사례들을 논의한다. 특히 특정 기수를 사용하는 복소수 진법에서 소수부를 복소평면에 나타냈을 때 생성되는 프랙털 구조를 영상 손실 압축 기법에 적용하는 가능성을 탐색한다.

키워드— 복소수 진법, 논리회로, ALU, 프랙털, 양자컴퓨팅, 영상 손실압축, AI

목 차

I 서론

복소수 진법은 기수(Radix)로 복소수 $\rho \in \mathbb{C}$, $|\rho| > 1$를 취함으로써 전통적인 위치 기수법(Positional Numeral System)을 복소수의 범위로 확장한 체계이다. 복소수 진법에서 임의의 복소수 $z \in \mathbb{C}$는

$$z = \sum_{k=m} d_k \rho^k, \qquad d_k \in D \subset \mathbb{C}$$

와 같이 표현된다. 여기서 D는 *자리수 집합(Digit set)*이라 부르며 일반적으로 유한 집합이고, 각 자리수 d_k는 보통 $|d_k| < |\rho|$ 같은 크기 제한을 만족하도록 선택한다. 또한 자리수의 노름(Norm)은 아르키메데스 성질(Archimedean property)을 만족하여 전통적 진법에서의 직관과 정렬성이 유지되도록 가정한다.

본 논문에서 사용하는 주요 표기와 가정은 다음과 같다.

- $\boldsymbol{\rho \in \mathbb{C}}$ $(|\rho| > 1)$: 기수(radix).
- $\boldsymbol{D \subset \mathbb{C}}$: 유한한 숫자 집합(자리수 집합). $|D|$는 정수이며, 보통 $|D| = |\rho|^2 + 1$ or $|\rho|^2$를 만족한다.
- D와 ρ의 선택에 따라 모든 복소수 또는 특정 부분공간(예: 가우스 정수)이 표현 가능한지, 그 표현이 유일한지가 결정된다.

복소수 진법의 대표적 예는 다음과 같다.

- $\boldsymbol{\rho = -1+i}$ $(D = \{0,1\})$: 흔히 *Complex Binary Number System (CBNS)*이라 불리는 체계로, 비부호화(unsigned) 방식으로 모든 가우스 정수(Gaussian integers)를 유한 표현으로 나타낼 수 있다. 이 진법은 이산화된 실수 격자 위에서 간단한 비트 조작으로 사칙연산을 수행할 수 있어, 컴퓨터 과학적 응용 가능성을 보인다.
- $\boldsymbol{\rho = 2i}$ $(D = \{0,1,2,3\})$: *Quater-imaginary base*로 알려진 고전적 예이다. 이 기수와 자리수 집합을 통해 이론적으로 모든 복소수를 표현할 수 있다(유한 표현/무한 소수 표현을 조합).
- $\boldsymbol{\rho = -n+i}$ **or** $\boldsymbol{\rho = n-i}$ $(n \geq 1 \in \mathbb{N})$: 소수부(즉, 음의 거듭제곱 부분)를 복소 평면에 표시하면 자기유사성(self-similarity)을 갖는 프랙털 모양의 집합이 나타난다. 특히 $\rho = -1+i$의 경우에는 *Twindragon*으로 불리는 잘 알려진 프랙털이 등장하며, 해당 프랙털은 진법의 기하학적 성질을 시각적으로 드러낸다.

본 논문의 주요 기여는 다음과 같다.

- 복소수 진법 기반 ALU 아키텍처 제안
- 표현 가능성과 연산 효율성 분석
- 시뮬레이션 환경에서의 성능 평가
- 응용 가능성 제시

다음 2장에서는 $-1+i$ 진법의 수학적 정의와 연산 규칙을 정리하고, 3장에서는 이를 적용한 ALU 설계 및 성능 평가를 다룬다. 4장에서는 $-1+i$ 진법이 생성하는 프랙털 구조를 이용한 영상 손실 압축 방안을 살펴보고, 5장에서 결론 및 향후 연구 방향을 제시한다.

II $-1+i$ 진법

복소수 진법 중에서 특히 $-1+i$를 기수로 하는 체계는 Complex Binary Number System (CBNS)으로 알려져 있으며, 자리수 집합으로 $D=\{0,1\}$을 사용한다.

$$z = \sum_{k=m}^{n} d_k(-1+i)^k, \qquad d_k \in \{0,1\}$$

여기서 $m \le n$이고 양의 지수부($k>0$)는 정수부, 음의 지수부($k<0$)는 소수부에 대응한다. 기수의 크기 $|\rho| = \sqrt{2}$이므로, 자리수 집합의 크기 조건 $|D| = |\rho|^2 = 2$를 만족하여, 모든 가우스 정수를 고윳값으로 표현할 수 있다.

II.1 진법 변환 알고리즘

$-1+i$ 진법 또한 일반적인 진법 변환 방법처럼 수행할 수 있다. 가우스 정수 Z가 $a+bi$로 정의 될 때, 이를 $-1+i$ 진법으로 변환하는 과정은 $Z_{old} = Z_{new}(-1+i)+r$를 만족하면서 Z_{new} 또한 가우스 정수가 되도록하는 r을 반복적으로 기록하는 하나의 규칙으로 정의된다.

이때 $Z_{old} = a+bi, Z_{new} = x+yi$로 가정하고 식을 다음과 같이 전개한다.

$$Z_{\text{old}} = Z_{\text{new}}(-1+i)+r$$

$$\Downarrow$$

$$a+bi = (x+yi)(-1+i)+r$$

$$a+bi = r-x-y+(x-y)i$$

$$\begin{cases} r-x-y=a \\ x-y=b \end{cases}$$

$$r-2y = a+b \quad \Rightarrow \quad 2y = r-(a+b)$$

이때, y가 정수여야 함으로 $r-(a+b)$ 가 짝수여야 한다. 따라서 r은 다음과 같이 정의된다.

$$r \equiv x+y \pmod 2$$

이상을 모두 일반화 하면 가우스 정수 $Z=a+bi$를 $-1+i$ 진법으로 변환하는 알고리즘을 다음과 같이 도출할 수 있다.

1. a와 b가 모두 짝수이거나 홀수인 경우 0을 기록하고 Z를 $-1+i$로 나눈다.
2. 그렇지 않으면, 1을 기록하고 $Z-1$을 $-1+i$로 나눈다.
3. 나눈 값이 0이 될 때 까지 반복한다.

II.2 산술 연산 규칙

$-1+i$ 진법에서의 산술 연산은 전통적인 방법과 유사하나, 특정 패턴에서의 올림(Carry)과 내림(Borrow)이 복잡하다.

덧셈 (Addition)

올림이 발생하는 $1+1$을 제외하고 일반적인 이진 덧셈과 동일하다. $-1+i$진법에서 $1+1=1100$이 되며 이는 올림이 두 비트 이상 떨어진 위치에서 발생함을 뜻한다

뺄셈 (Subtraction)

빌림이 발생하는 $0-1$을 제외하고 일반적인 이진 뺄셈과 동일하다. $-1+i$진법에서 $0-1$의 연산이 발생하는 위치를 k라고 하면, 피감수 a와 감수 b가 다음에 따라 변화한다.

1. $a_k \to a_k + 1$
2. $a_{k+1} \to a_{k+1}$ (변경 없음)
3. $a_{k+2} \to a_{k+2} + 1$
4. $a_{k+3} \to a_{k+3} + 1$
5. $a_{k+4} \to a_{k+4} + 1$

6. $b_k \to 0$

곱셈 (Multiplication)

쉬프트-앤-애드(shift-and-add) 방식을 사용하며, 각 비트 곱한 후 $(-1+i)^k$만큼 쉬프트하여 덧셈한다. 이때 (자료형의 한계) $+ 1 = 0$으로 계산되는 기존 ALU에 착안하여 $111 + 11$을 0으로 정의하여 계산을 최적화 할 수 있다.

나눗셈 (Division)

먼저, 역수를 근사하여 구한 후 곱셈하여 나눗셈을 계산한다. 가우스 정수 $z \in \mathbb{C}$의 역수 $w = \frac{1}{z}$를 구하기 위해 함수 $f(w)$를 아래와 같이 정의한다.

$$f(w) = \frac{1}{w} - z$$

위 함수에 대해 뉴턴-랩슨 방법을 적용하여 방정식 $f(w) = 0$에 대한 근의 근사값을 구한다. 뉴턴-랩슨 방법의 점화식에 $f(w)$를 대입하여 정의하면 다음과 같다.

$$w_{s+1} = w_s - \frac{f(w_s)}{f'(w_s)}$$

$$f'(w) = -\frac{1}{w^2}$$

$$\frac{f(w)}{f'(w)} = \frac{\frac{1}{w} - z}{-\frac{1}{w^2}} = (\tfrac{1}{w} - z)(-w^2) = -w + zw^2$$

$$\begin{aligned} w_{s+1} &= w_s - \frac{f(w_s)}{f'(w_s)} \\ &= w_s - \left(- w_s + z w_s^2 \right) \\ &= 2w_s - z w_s^2 \\ &= w_s \left(2 - z w_s \right) \end{aligned}$$

z를 $-1+i$ 진법으로 나타낸 후 위치 기수법의 정의에 따라 식을 풀어 적었을 때, 곱해지는 $(-1+i)^K$ 중 가장 큰 K를 택한다. 기본적으로 초깃값 w_0은 K에 따라 $(-1+i)^{-K}$로 정해지며, $|zw_0 - 1| < 1$을 만족하지 않는 경우에만 $w_0' = i\, w_0$을 활용한다. 원하는 만큼 근사한 후 이를 피젯수와 곱하여 나눗셈을 계산한다.

III ALU 설계 및 성능 평가

제안하는 $-1+i$ 진법 ALU는 가우시안 정수 하나를 $-1+i$ 진법으로 저장하는 레지스터와, 덧셈기(Adder), 뺄셈기(Subtractor), 곱셈기(Multiplier)로 구성된 연산 장치로 이루어진다. 나눗셈기(Divider)의 경우 논리회로 상에서의 구현이 아직 이루어지지 않아 향후 연구가 필요한 대목이다. 각 연산 장치의 입력과 출력은 이진 신호 집합이지만, 이 비트들은 기존의 2진수 대신 $-1+i$ 진법에서의 비트열로 해석된다.

회로 설계 수준에서는 T.Jamil이 제안한 4비트(Nibble) 단위 회로 설계를 참조하였다. 예를 들어 4비트 덧셈기는 입력 비트 8개(2개의 4비트 수)에서 출력 비트 12개를 생성하는 진리표를 가지며, 8×256 디코더와 게이트로 구현 가능하다. 뺄셈기와 곱셈기도 유사하게 구현할 수 있다.

III.1 HDL 설계

실제 하드웨어 구현을 가정하여 Verilog HDL로 설계 후 Vivado 시뮬레이션 환경에서 동작을 검증하였다. 예를 들어 16비트 CBNS 덧셈/곱셈기의 클럭 주기를 측정한 결과, 전통적 방식으로 실수/허수부 덧셈기를 두 번 사용하는 경우와 비교하여 유사한 지연 시간을 가지면서도 복소수 연산을 한 번에 처리하는 이점을 확인하였다.

덧셈기 (Adder)[11]

4비트(Nibble) 입력에 대해 총 12비트를 출력한다. 진리표를 그리고 각 출력에 대해 온라인 카르노 맵 분석기를 사용하여 최적화 하였다. 각 출력 비트의 논리 식은 AND, OR, XOR 게이트 조합으로 구현되며 병렬 구조를 통해 지연을 최소화한다. FPGA에 따라 사용되는 게이트 수에 차이가 발생하며 평균 330개에서 최저 175개이다.

뺄셈기 (Subtractor)[12]

4비트(Nibble) 입력에 대해 총 11비트를 출력한다. 빌림이 발생할 때 피감수의 상위 비트(k+2, k+3, k+4)를 조정하며, 진리표를 그리고 각 출력에 대해 온라인 카르노 맵 분석기를 사용하여

최적화 하였다. 각 출력 비트의 논리 식은 AND, OR, XOR 게이트 조합으로 구현되며 병렬 구조를 통해 지연을 최소화한다. FPGA에 따라 사용되는 게이트 수에 차이가 발생하며 평균 864개에서 최저 220개이다.

곱셈기 (Multiplier)[13]

4비트(Nibble) 입력에 대해 총 12비트를 출력한다. 진리표를 그리고 각 출력에 대해 온라인 카르노 맵 분석기를 사용하여 최적화 하였다. 각 출력 비트의 논리 식은 AND, OR, XOR 게이트 조합으로 구현되며 병렬 구조를 통해 지연을 최소화한다. FPGA에 따라 사용되는 게이트 수에 차이가 발생하며 평균 771개에서 최저 420개이다.

IV 영상 프랙털 압축

앞서 2장에서 살펴본 $-1+i$ 진법은 소수부를 복소평면에 나타내어 트윈드래곤 프랙털을 얻을 수 있다. 본 절에서는 전통적인 영상 프랙털 압축과 같이 2차원 영상을 여러 개의 트윈드래곤 모양으로 분할한 뒤 각 타일에 대해 웨이블릿 변환을 결합하는 *프랙탈-웨이블릿* 압축 기법을 제안한다.

각 픽셀을 복소평면의 한 점 z로 사상하고, 영상을 유한 개의 타일 $\{T_k\}_{k=1}^{K}$로 분할한다. 각 타일 T_k의 점들에 대해 $\alpha z+\beta$를 통해 축소, 회전, 이동을 통해 전체 이미지를 근사한다 (반복함수계 Iterated function system; IFS). 각 타일에 대해 2D 웨이블릿 변환을 적용하여 계수 집합 $w_k[n]$를 얻는다. 역변환에서는 저장된 계수만 복원해 타일을 재구성한다.

모든 타일 T 내 픽셀 좌표 (x,y)는 복소수 $z=(x-x_c)+i(y-y_c)$ 로 사상된다. 여기서 (x_c,y_c)는 영상의 크기에 따라 달리 결정되며, 각 타일이 영상의 경계에서 벗어나지 않으면서 빈 공간을 최소화 하는 중심점이다. 그레이스케일의 경우 각 타일 내부의 화소값을 효율적으로 표현하기 위해 ψ를 기저로 하는 2차원 웨이블릿 변환을 수행한다.

$$w_k[u,v]=\sum_{x,y\in T_k} I(x,y)\,\psi_u(x)\,\psi_v(y)$$

V 결론 및 향후 연구

본 논문에서는 $-1+i$을 기수로 하는 복소수 진법을 기반으로 ALU의 개념적 설계와 응용 가능성을 정리하였다. 수학적 관점에서 $-1+i$ 진법의 표현성(가우스 정수의 유한 표현), 자리수 변환 알고리즘, 그리고 덧셈·뺄셈·곱셈·나눗셈의 연산 규칙을 체계화하였다. 하드웨어 관점에서는 비트열을 직접 다루는 연산기의 설계 원리와 HDL 수준의 설계 흐름을 제시했으며, 프랙탈 구조(트윈드래곤)를 이용한 영상 프랙탈-웨이블릿 압축 파이프라인을 제안하였다.

- **표현의 단일성:** 실수부·허수부를 분리하지 않고 하나의 비트열로 복소수를 표현하므로 데이터 경로를 단순화할 수 있다. 이는 특히 여러 복소수 연산을 연속적으로 수행할 때 레지스터 접근과 데이터 이동 비용을 줄이는 이점을 제공한다.
- **연산 횟수 관점의 잠재적 이득:** 전통적 복소수 곱셈(4개의 실수 곱셈 + 2개의 덧셈)에 비해 복소수 진법 기반 구현은 동일한 연산을 하나의 복소수 승산 경로로 통합함으로써 이론적으로 계산량을 크게 줄일 여지가 있다.
- **응용성:** FFT·컨볼루션 기반 연산, 복소수 기반 CNN(예: 스펙트럼 레이어), 양자컴퓨팅 시뮬레이션(복소수 진폭 연산)에 복소수 진법 기반 ALU를 적용하면 전체 워크로드 수준에서 실질적 성능 향상을 기대할 수 있다. 또한 트윈드래곤 기반 프랙탈 타일링은 영상 압축에서 자기유사성을 이용한 고효율 표현 수단으로 유망하다.

향후 연구 과제(우선순위 및 검증 계획)는 다음과 같다.

- 각 연산(덧셈, 곱셈, 나눗셈)에 대해 캐리 전파 길이, 필요 소자 수(LUT/Slice) 추정식을 유도한다.
- Xilinx 계열에서 4/8/16/32비트 CBNS 연산기의 합성, 타이밍 분석, 전력 추정 수행한다.
- FFT(길이 N)에서 복소수 진법 적용 시 총 실수 곱셈·덧셈 수의 감소율을 엄밀히 보이고, 이를 통해 이론적 시간 복잡도와 실제 사

이클 수(클럭 주파수 · 파이프라인 깊이 고려)를 연관 짓는다. 특히 FFT 기반 컨볼루션 레이어(FFT-Conv) 혹은 복소수 가중치를 쓰는 AI 레이어에 복소수 진법 ALU를 삽입해 학습 시 성능을 비교한다.

- 복소수 진법 ALU 기반의 상태벡터(State-vector) 기반 양자 컴퓨팅 시뮬레이션을 구현하여, 게이트 적용 시 소요 시간 · 메모리 접근 횟수를 얼마나 줄이는지 실험적으로 입증한다.
- 다른 기수로 일반화하여 어떤 기수가 하드웨어 유닛 개발 및 프랙털 활용에 있어 최적인지 비교 · 평가한다.

본 논문은 국문으로 복소수 진법에 관한 연구를 종합적으로 정리하여 국내 석학들에게 있어 새로운 연구의 지평을 여는 출발점이다. 다음 단계는 위에서 제시한 수학적 엄밀화와 FPGA/ASIC 수준의 프로토타이핑을 통해 이론적 예측을 실험적으로 검증하는 것이다. 특히 컨볼루션(FFT) 기반 AI 워크로드와 복소수 연산이 많은 양자 컴퓨팅 시뮬레이션에서 복소수 진법 기반 ALU가 실질적 이득을 제공할 가능성이 크므로 관련 분야에 대한 많은 관심을 부탁한다.

VI 참고문헌

[1] T. Jamil, M. Awadalla, and I. Mohammed, "Complex Binary Adder Designs and their Hardware Implementations," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 7, 2019. doi: 10.14569/IJACSA.2019.0100734.

[2] D. E. Knuth, "A imaginary number system," *Commun. ACM*, vol. 3, no. 4, pp. 245–247, Apr. 1960. doi: 10.1145/367177.367233.

[3] W. Penney, "A "Binary" System for Complex Numbers," *J. ACM*, vol. 12, no. 2, pp. 247–248, Apr. 1965. doi: 10.1145/321264.321274.

[4] C. Frougny and A. Surarerks, "On-line multiplication in real and complex base," in *Proc. 16th IEEE Symposium on Computer Arithmetic*, 2003, pp. 212–219. doi: 10.1109/ARITH.2003.1207681.

[5] A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Process.*, vol. 1, no. 1, pp. 18–30, 1992. doi: 10.1109/83.128028.

[6] T. Jamil, *Complex Binary Number System: Algorithms and Circuits*, 2013. doi: 10.1007/978-81-322-0854-9.

[7] W. J. Gilbert, "Fractal geometry derived from complex bases," *The Mathematical Intelligencer*, vol. 4, no. 2, pp. 78–86, Jun. 1982. doi: 10.1007/BF03023486.

[8] M. F. Barnsley and L. P. Hurd, *Fractal image compression*, A. K. Peters, Ltd., USA, 1993.

[9] J. Duda, "Complex base numeral systems," arXiv:0712.1309 [math.DS], 2008. `https://arxiv.org/abs/0712.1309`.

[10] W. J. Gilbert, "Arithmetic in Complex Bases," *Mathematics Magazine*, vol. 57, no. 2, pp. 77–81, 1984. doi: 10.1080/0025570X.1984.11977081.

[11] T. Jamil, M. Awadalla, and I. Mohammed, "Complex Binary Adder Designs and their Hardware Implementations," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, 2019. doi: 10.14569/IJACSA.2019.0100734.

[12] T. Jamil, M. Awadalla, and I. Mohammed, "Complex Binary Subtractor Designs and their Hardware Implementations," *Int. J. Adv. Res. Eng. Technol.*, vol. 12, no. 5, pp. 111–125, 2021. `https://iaeme.com/Home/article_id/IJARET_12_05_011`.

[13] T. Jamil, M. Awadalla, and I. Mohammed, "Nibble-size Multiplier Circuit Designs and their FPGA Implementations for Complex Binary Number System," *International Journal of Electrical Engineering and Technology (IJEET)*, vol. 12, no. 6, pp. 105–121, 2021. `https://iaeme.com/Home/article_id/IJEET_12_06_012`.

[14] D. C. Blest and T. Jamil, "Division in a binary representation for complex numbers," *Int. J. Math. Educ. Sci. Technol.*, vol. 34, no. 4, pp. 561–574, 2003. doi: 10.1080/0020739031000108592.

[15] A. Shadap and P. Saha, "Discrete Fourier transformation processor based on complex radix $(-1 + j)$ number system," *Eng. Sci. Technol., Int. J.*, vol. 20, no. 1, pp. 80–88, 2017. doi: 10.1016/j.jestch.2016.08.020.