논문 2012-49-9-21

이중 경로 십진 부동소수점 가산기 설계

(Design of Dual-Path Decimal Floating-Point Adder)

이 창 호*, 김 지 원*, 황 인 국*, 최 상 방**

(Chang-Ho Lee, Ji-Won Kim, In-Guk Hwang, and Sang-Bang Choi)

요 약

본 논문에서는 동일한 크기의 지수를 갖는 십진 부동소수점 오퍼랜드의 가산 및 감산연산을 빠르게 하기 위해, 두 개의 데이터 경로를 가지는 십진 부동소수점 가산기를 제안한다. 제안된 십진 부동소수점 가산기는 L. K. Wang의 오퍼랜드 정렬 계획을 사용하지만 오퍼랜드의 지수 크기가 같을 경우 정밀도를 보장하는 범위 내에서 속도 향상을 위해 고속의 데이터 경로를 통해 연산한다. 제안된 가산기의 성능 평가를 위해 Design Compiler에서 SMIC사의 0.18㎛ CMOS 공정 테크놀로지 라이브러리를 이용하여 합성하였다. 합성 결과 면적은 L. K. Wang의 가산기와 비교하여 8.26% 증가하였지만 전체 임계경로의 지연시간이 10.54% 감소하였다. 또한 같은 크기의 지수를 가지는 오퍼랜드를 연산할 때는 임계경로보다 13.65% 단축된 경로에서 연산을 수행하는 것을 확인하였다. 제안한 십진 부동소수점 가산기 구조는 동일 크기의 지수를 가지는 오퍼랜드의 비중이 2% 이상일 때 L. K. Wang의 가산기 구조 대비 효용성이 높다.

Abstract

We propose a variable-latency Decimal Floating Point(DFP) adder which adopts the dual data path scheme. It is to speed addition and subtraction of operand that has identical exponents. The proposed DFP adder makes use of L. K. Wang's operand alignment algorithm, but operates through high speed data-path in guaranteed accuracy range. Synthesis results show that the area of the proposed DFP adder is increased by 8.26% compared to the L. K. Wang's DFP adder, though critical path delay is reduced by 10.54%. It also operates at 13.65% reduced path than critical path in case of an operation which has two DFP operands with identical exponents. We prove that the proposed DFP adder shows higher efficiency than L. K. Wang's DFP adder when the ratio of identical exponents is larger than 2%.

Keywords : Decimal Floating-Point Adder, IEEE 754-2008

Ⅰ. 서 론

이진 부동소수점 연산은 과학, 공학 어플리케이션(application)에 있어서 이미 충분한 기능을 제공하고 있다. 그러나 상업, 금융 어플리케이션과 같이 금액을 주 연산 대상으로 하는 시스템에서는 십진 데이터들의 정밀도가 중요하므로 이에 적합하지 않다. 이는 이진 부동소수점 연산이 대부분의 부동소수점 수를 정확한 값으로 표현할 수 없고, 십진수에 대한 적절한 라운딩(rounding)을 제공하지 못하기 때문이다. 예를 들어 십진수 0.1(d)을 IEEE 754-1985의 단정도(single precision)에서 roundToNearestEven 라운딩 모드를 적용시켜 변환하면 그 값은 16진수로 3DCCCCCD(h)이고 이때 유효수부는 1.10011001100110011001101(b)로 표현된다. 이 값을 다시 십진수로 변환하면 0.10000000149011612(d)이므로 기존의 이진 부동소수점은 수치를 정확하게 표현하는데 한계가 있음을 알 수 있다. 이러한 이유에서

* 학생회원, ** 평생회원, 인하대학교 전자공학과
(Dept. of Electronic Engineering Inha University)
※ 이 논문은 인하대학교의 지원에 의하여 수행된 연구임
접수일자:2012년4월18일, 수정완료일:2012년9월7일

환전, 보험금 산출, 세금 계산, 그리고 전자상거래 및 은행 업무 등의 기능을 수행하는 상업, 금융 어플리케이션에 있어서, 십진수를 근사값이 아닌 정확한 수치로 표현할 수 있는 십진 부동소수점(Decimal Floating Point, DFP) 연산에 대한 많은 연구가 이루어지고 있다. 십진 부동소수점 연산 방식은 이진수의 변환 오차가 없어 보다 정확한 수의 표현이 가능하다. 하지만, 컴퓨터 하드웨어(hardware)는 0과 1의 이진 구조이므로 BCD(Binary Coded Decimal)와 같이 이진수를 십진수로 변환하는 방법이 필요하다. IEEE 754-2008[1] 표준안에서는 DPD(Densely Packed Decimal) 코드와 BID(Binary Integer Decimal) 코드를 제안하였는데, 일반적으로 DPD는 하드웨어에 적합하며 BID는 소프트웨어(software)에 적합한 것으로 알려져 있다. IEEE 754-2008 표준안의 십진 부동소수점 연산을 수행하는 소프트웨어 패키지(package)로는 IBM의 decNumber library, Java의 BigDecimal library, 그리고 Intel의 DFP Math library가 있다[2~4]. 이러한 소프트웨어 패키지들은 높은 성능을 요구하지 않는 어플리케이션에 적합하다. 연산에 대한 보다 높은 성능을 얻기 위해서는 하드웨어적인 측면에서 십진 부동소수점 연산기를 설계하여 이를 마이크로프로세서에 내장하는 것이 좋다. IBM에서는 POWER6, POWER7, z9, 그리고 z10과 같은 자사의 워크스테이션용 마이크로프로세서 아키텍처에 십진 부동소수점 연산에 대한 기능을 추가하여 하드웨어적으로 십진 부동소수점 연산을 수행하고 있다[5~8].

최근 들어 십진 부동소수점 어플리케이션의 벤치마킹 및 성능 분석에 대한 연구가 진행되고 있으며, 오퍼랜드의 특성 연구 결과 가산 및 감산 연산에서의 두 입력 오퍼랜드가 같은 크기의 지수를 갖는 확률이 상당 부분을 차지하고 있다[9]. 같은 크기의 지수를 갖는 두 오퍼랜드의 연산은 별도의 빠른 처리를 위한 경로를 구성하여 전반적인 성능향상을 기대할 수 있다[10~13]. 본 논문에서는 이중경로 방법을 십진 부동소수점에 맞게 적용하여, L. K. Wang의 십진 부동소수점 가산기에 가변시간을 갖는 가산기를 설계하였다. 오퍼랜드 정렬 계산 및 교환 유닛에 EQ신호와 LSE신호를 추가하여 같은 크기의 지수를 갖는 두 오퍼랜드의 연산이 배럴 시프터 유닛을 거치지 않고 빠르게 처리될 수 있도록 하였다. 또한 두 지수차의 절대값을 구하기 위해 사용된 순환 캐리 가산기의 캐리 전달에 따른 속도 저하를 해결하기 위해 K-S(Kogge-Stone) 가산기로 대체하였다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 IEEE 754-2008 십진 부동 소수점에 대해 알아본 후, Ⅲ장에서 L. K. Wang의 십진 부동소수점 가산기에 대해 설명한다. Ⅳ장에서는 제안하는 십진 부동소수점 가산기에 대해 살펴보고, Ⅴ장에서 합성한 결과를 바탕으로 L. K. Wang의 십진 부동소수점 가산기와 비교 분석한 뒤, Ⅵ장의 결론으로 본 논문을 마무리한다.

Ⅱ. 십진 부동 소수점

이 장에서는 십진 부동소수점 가산기에 대해 설명하기에 앞서 IEEE 754-2008 표준안에 명시된 십진 부동소수점의 포맷과 매개변수 및 특성, 그리고 연산에 대해 설명한다.

2.1 IEEE 754-2008

IEEE 754-2008 표준안은 이진 부동소수점을 정의한 IEEE 754-1985 표준안의 개정판으로써, 이미 기존 표준안을 통해 정의된 이진 부동소수점 포맷(format) 외에도 십진 부동소수점에 관한 포맷과 이에 관련된 연산을 추가적으로 정의하고 있다. 표준안에 새롭게 추가된 십진 부동소수점 포맷으로는 decimal32, decimal64, 그리고 decimal128이 있으며, 각각 32비트, 64비트, 128비트의 저장 공간을 필요로 한다. 이들 포맷은 저장 공간의 효율성을 보장하면서도 빠른 연산을 만족시키기 위해 고안되었다.

본 논문에서 구현한 십진 부동소수점 가산기는 IEEE 754-2008 표준안에 기반하여 설계되었기 때문에, 이번 장에서는 IEEE 754-2008 표준안에 대한 개요와 십진 부동소수점의 기본 형식과 특성, 그리고 관련 연산에 대해서 설명한다.

2.1.1 십진 부동소수점 포맷

IEEE 754-2008 표준안에서 십진 부동소수점 v는 식 (1)로 정의된다.

$$v = (-1)^S \times C \times 10^q \quad (1)$$

여기에서 S는 부호(sign) 비트, q는 편향 지수(biased exponent), 그리고 C는 유효수(significand)를

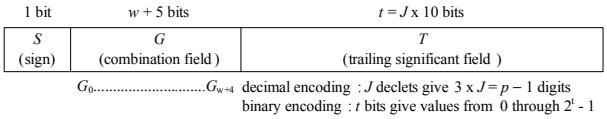

그림 1. 십진 부동소수점 포맷
Fig. 1. Decimal floating-point format.

나타낸다. 유효수 C가 정밀도(precision) p를 가질 때에는 벡터 $c_0c_1c_2...c_{p-1}$로 표현되며, 이때 c_i는 $\{0, 1, ..., 9\}$중 하나의 값을 갖는 각 자릿수(digit)이다. 십진 부동소수점 포맷인 decimal32, decimal64, decimal128에서 정밀도 p는 각각 7, 16, 34이다.

십진 부동소수점의 포맷은 그림 1과 같이 3개의 필드(field)로 구성되어 있는데, 먼저 각 필드의 기능에 대해 살펴보도록 한다. S는 부호비트로써, 이진 부동소수점과 마찬가지로 부호비트가 0일 때에는 양수를, 1일 때에는 음수를 나타낸다. G는 조합필드(combination field)라고 하며 w+5비트의 크기를 가진다. 조합필드에서 $G_{5:w+4}$의 하위 w비트는 단순히 지수의 일부분을 표현하는데 사용되고, $G_{0:4}$의 상위 5비트는 NaN(Not a Number), 무한대(∞), 0과 같은 특정 값을 구분하는데 사용된다. 하지만 유한수(finite number)를 표현하는 경우에는 지수의 상위 2비트와 유효수의 상위 4비트를 포함하기 때문에 조합필드라고 한다. T는 부호화된 유효수 필드(trailing significant field)라 하고, 유효수의 상위 4비트를 제외한 나머지 부분을 표현하는데 사용한다. 이진 고정점(binary fixed-point)을 사용하는 이진 부동소수점의 유효수와는 다르게, 십진 부동소수점에서는 십진 유효수를 표현하기 위해 3자리의 십진수를 디클렛(declet) J라는 단위를 사용하여 저장한다. BCD 코드로 한 개의 디클렛을 표현하면 12비트가 사용되는데, 저장 공간을 절약하기 위해 DPD(Densely Packed Decimal)나 BID(Binary Integer Decimal) 인코딩을 취하여 10비트에 저장한다. 따라서 J개의 디클렛은 $J\times10$비트를 사용하여 저장할 수 있다. 각 십진 부동소수점 포맷에 따른 매개변수는 표 1에 자세히 나타나 있다.

IEEE 754-2008 표준안에서는 부호화된 유효수 필드에서 사용되는 DPD와 BID 인코딩을 모두 지원한다. DPD는 일종의 허프만 인코딩(Huffman encoding) 방식을 적용한 것이고, BID는 10비트로 나타낼 수 있는 수의 범위인 $\{0, 1, ..., 1023\}$중에서 $\{0, 1, ..., 999\}$만을 사용하여 표현하는 방법이다. DPD와 BID를 사용할 때의 십진 부동소수점의 범위는 동일하기 때문에 IEEE 754-2008 표준안에서는 두 방식 중 하나를 자유롭게 사용하도록 하고 있다. 이중에 BID는 하드웨어적으로 라운딩 로직(rounding logic)을 구현하기가 어렵기 때문에 주로 소프트웨어 패키지에서 사용되고 있다.

2.1.2 DPD 포맷

앞서 설명한 바와 같이 조합필드에서 $G_{0:4}$의 상위 5비트는 특정 값을 구분하거나 지수와 유효수의 일부분을 포함하는데 사용되는데, 십진 부동소수점 포맷은 DPD와 BID 중에 어떤 포맷을 사용하는지에 따라 상이한 비트 사용 패턴을 보인다. 본 논문에서는 DPD 방식을 이용한 십진 부동소수점 가산기를 설계 및 구현하였으므로 DPD 인코딩에 대해 자세히 살펴보도록 한다. DPD 인코딩을 사용하는 경우에는 $G_{0:4}$가 11111(b)이면 십진 부동소수점 수가 NaN임을 나타낸다. 이때, 추가적으로 G_5가 1이면 sNaN(signaling NaN), 0이면 qNaN(quiet NaN)을 나타내고, 조합필드의 나머지 비트들은 무시된다. $G_{0:4}$가 11110(b)인 경우에는 조합필드와 부호화된 유효수 필드의 나머지 비트들은 무시되고, 부호비트에 따라 $\pm\infty$를 나타낸다. 나머지의 경우에는 유한수를 정의하게 되는데, $G_{0:4}$가 110XX(b) 또는 1110X(b)이면 유효수의 선두 자릿수(leading

표 1. 십진 부동소수점 형식의 매개변수
Table 1. Parameters of DFP formats.

Parameter	decimal 32	decimal 64	decimal 128
storage bits (k)	32	64	128
sign bit	1	1	1
combination field bits ($w+5$)	11	13	17
trailing significand field bits (t)	20	50	110
precision (p)	7	16	34
maximum exponent ($emax$)	+96	+384	+6144
minimum exponent ($emin$)	-95	-383	-6143
exponent bias ($bias$)	101	398	6176

표 2. DPD 인코딩 법칙
Table 2. DPD encoding rules.

DPD										BCD			Digit pattern
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	$B_{11:8}$	$B_{7:4}$	$B_{3:0}$	
a	b	c	d	e	f	0	g	h	i	0abc	0def	0ghi	3 smalls
a	b	c	d	e	f	1	0	0	i	0abc	0def	100i	1 big, 2 smalls
a	b	c	d	e	f	1	0	1	i	0abc	100f	0dei	
a	b	c	d	e	f	1	1	0	i	100c	0def	0abi	
a	b	c	1	0	f	1	1	1	i	0abc	100f	100i	2 bigs, 1 small
a	b	c	0	1	f	1	1	1	i	100c	0abf	100i	
a	b	c	0	0	f	1	1	1	i	100c	100f	0abi	
x	x	c	1	1	f	1	1	1	i	100c	100f	100i	3 bigs

significand digit) c_0는 $8+G_4$로 계산되며 $\{8,9\}$중의 한 값을 가지고, 편향 지수의 상위 2비트는 $2G_2+G_3$로 $\{0,1,2\}$중의 한 값을 가진다. $G_{0:4}$가 0XXXX(b) 또는 10XXX(b)이면 유효수의 선두 자릿수 c_0는 $4G_2+2G_3+G_4$로 계산되며 $\{0,1,...,7\}$중의 한 값을 가지고, 편향 지수의 상위 2비트는 $2G_0+G_1$로 계산되어 $\{0,1,2\}$중의 한 값을 가진다. 그리고, 유효수의 선두 자릿수를 제외한 나머지 유효수는 부호화된 유효수 필드에서 DPD 포맷으로 인코딩되어 표현된다. DPD 인코딩 법칙은 표 2와 같다.

2.2 십진 부동소수점의 특성

십진 부동소수점은 금융 어플리케이션을 주목적으로 하는데, 대부분 통화에 대한 연산을 다루기 때문에 이진 부동소수점과는 구분되는 특성이 있다. 가령 금융 어플리케이션에서는 통화의 단위를 정의하기 위해 천, 백만, 10억을 나타내는 “K”, “M”, “B”의 기호를 사용하거나 백분율을 표시하기 위해 “%”를 사용한다. 그러므로 데이터베이스에 특정 단위에 맞추어 부동소수점을 저장하기 위해서는 추가적인 연산이 필요하다. 또한 어플리케이션에서 데이터베이스로부터 읽은 십진 부동소수점이 같은 단위를 가지고 있는지 여부를 판별하고 필요하다면 같은 단위로 맞추어 주는 작업 또한 요구되고 있다. 따라서 IEEE 754-2008 표준안에서는 SameQuantum과 Quantize의 2가지 십진 부동소수점 연산을 추가적으로 정의하고 있다. SameQuantum의 경우에는 두 십진 부동소수점 수가 같은 지수를 가지고 있는지를 여부를 판별하는 연산이고, Quantize는 한 부동소수점을 다른 부동소수점의 지수에 맞추는 연산이다. 이러한 이진 부동소수점과는 다른 십진부동소수점의 특성때문에 IEEE 754-2008 표준안에서는 한 부동소수점이 다수의 표현방법을 가지는 것을 허용하고 있으며 이를 집단(cohort)라 한다. 예를 들어, $+1.2345\times10^6$과 $+12.345\times10^5$은 집단이다. 결론적으로 보면 십진 부동소수점은 정규화(normalization)가 필요하지 않다. 이러한 특성때문에 IEEE 754-2008 표준안에서는 선호지수(preferred exponent)라는 용어를 정의하고 있다. $x+y$의 가산연산에서 만약 결과값이 정확하게 표현될 수 없다면 선호지수는 최소한으로 가능한 지수가 선택되는데, 이는 유효수의 정밀도를 최대한으로 보장하기 위한 것이다. 예를 들어 정밀도 p가 7인 decimal32 포맷의 x가 215×10^2이고 y가 145×10^{-3}인 가산연산에서 라운딩 모드가 roundTiesToAway라고 가정하면 결과값은 $2{,}150{,}015\times10^{-2}$이 된다. 이것은 두 유효수를 더한 결과는 21,500,145 이지만 정해진 정밀도 내에서 표현할 수 없기 때문에 가능한 작은 선호지수가 선택된 것이다. 만약 결과값이 정밀도 내에서 정확하게 표현 가능하다면 선호지수는 두 오퍼랜드의 지수 중에서 작은 지수를 취한다.

Ⅲ. L. K. Wang의 십진 부동소수점 가산기

이 장에서는 본 논문에서 제안한 부동소수점 가산기의 비교대상으로 하는 L. K. Wang의 십진 부동소수점

가산기에 대해 설명한다[14]. 설명에 사용되는 기호는 표 3에 정리 되어 있고, 제안한 십진 부동소수점 가산기에도 동일하게 사용된다.

3.1 L. K. Wang의 가산기 개요

L. K. Wang의 가산기는 Thompson의 가산기[15]를 기반으로 유효수 보전을 위해 최적화된 오퍼랜드 정렬 알고리즘을 적용하고 십진 주입기반 라운딩(injection-based rounding)과 플래그된 프리픽스 가산기(flagged prefix adder)를 이용한다. 오퍼랜드 정렬 알고리즘은 정해진 정밀도 내에서 정확한 결과값과 최대의 효율을 얻을 수 있도록 고안되었다. 십진 주입기반 라운딩이란 이진 주입기반 라운딩을 십진 부동소수점에 맞게 적용한 것이다[16]. 이진 주입기반 라운딩에서는 가드(guard), 라운드(round), 스티키(sticky) 비트를 검사하지 않고, 최하위 자릿수까지 캐리가 발생하여 자리올림이 되도록 정해진 값을 주입해 연산하며, 라운딩된 값은 절삭을 통해 얻는다. 두 유효수의 가산은 K-S 가산기를 통해 이루어지는데, 값을 보정해야 하는 경우에 사용되는 플래그를 발생시킨다.

L. K. Wang의 가산기는 두 십진 부동소수점 오퍼랜드 OpA와 OpB를 입력받아 연산 명령 $Operation$과 라운딩 모드 $Rounding$에 따라 연산하고 결과값 $Result$를 출력하는 구조로 이루어져 있다. 가산기의 각 유닛들은 포맷 변환, 오퍼랜드 정렬, 연산 및 보정, 제어신호 발생 등의 역할을 담당하며, 다음 절에서 각 유닛들의 기능에 대해 간략하게 설명한다.

3.2 세부 유닛 설명

포워드 포맷 변환 유닛(Forward Format Conversion Unit)은 두 개의 IEEE 754-2008 형식의 오퍼랜드 A와 B를 입력받아 부호 비트 SA_1과 SB_1, BCD 형식의 유효수부 CA_1과 CB_1, 그리고 편향 지수 EA_1, EB_1으로 변환한다. 또한 연산 $Operation$을 입력받아 유효연산 EOP를 결정한다.

오퍼랜드 정렬 계산 및 교환 유닛(Operand Alignment Calculation and Swapping Unit, OACSU)은 포워드 포맷 변환 유닛에서 결정된 값을 통해 임시 지수 ER_1과 좌우 시프트 값인 LSA, RSA를 계산한다. 또한 더 큰 지수를 갖는 오퍼랜드의 유효수부가 CA_S가 되고 작은 지수를 갖는 오퍼랜드의 유효수부가 CB_S가 되도록 CA_1과 CB_1을 교환한다. 오퍼랜드의 지수가 다를 경우 십진 배럴 시프터(Decimal Barrel Shifters)에서 CA_S와 CB_S를 RSA와 LSA에 따라 시프트하여 정렬하며, 스티키 비트를 계산한다. 사전교정 및 오퍼랜드 배치 유닛(Precorrection and Operand Placement Unit)에서는 EOP와 라운딩 모드에 따라 BCD로 부호화된 유효수부를 사전 교정하고 라운딩에 필요한 십진 값들을 주입(injection)한다. 앞의 단계를

표 3. 십진 부동소수점 가산기에 사용된 기호
Table 3. Definition of Symbols used in DFP Adder.

Symbol	Definition
SA_1	Sign bit of operand A
SB_1	Sign bit of operand B
CA_1	Significand of operand A
CB_1	Significand of operand B
EA_1	Biased exponent of operand A
EB_1	Biased exponent of operand B
EOP	Effective operation
LA_S	The number of leading 0s in CA_S
RSA	Right shift amount
LSA	Left shift amount
CA_S	Significand with larger exponent
CB_S	Significand with smaller exponent
ER_1	Temporary result exponent
CA_2	Significand after alignment of CA_S
CB_2	Significand after alignment of CB_S
$CA_2^{'}$	Shifted and injected 19 digit significand from CA_2
$CB_2^{'}$	Shifted 19 digit significand from CB_2
CA_3	Precorrected significand from $CA_2^{'}$
CB_3	Precorrected significand from $CB_2^{'}$
C_1	Carry-out bits
UCR	Uncorrected result
F_1	Flag bits used in the postcorrection step
F_2	Flag bits indicate trailing-nine
CR_1	Corrected result
SR_1	Sign bit of the result
$overflow$	Overflow flag
ER_2	Final result exponent
CR_2	Final result significand
R_1	Final DPD-encoded result

통해 연산을 위한 오퍼랜드가 준비 완료되면 K-S 네트워크에서는 미정정 결과 값(uncorrected result) UCR과 디짓(digit) 캐리벡터 C_1, 그리고 후교정에서 사용되는 플래그 비트 벡터 F_1과 시프트 및 라운딩 유닛에서 사용되는 플래그 비트 벡터 F_2를 발생시킨다. 후교정 유닛(Postcorrection Unit)에서는 미정정 결과 값 UCR과 EOP, C_1, F_1으로부터 BCD로 부호화된 정정된 결과 값(corrected result) CR_1을 생성한다. 시프트 및 임시 지수 ER_1으로부터 결과 지수 ER_2를 생성한다. 만라운드 유닛(Shift and Round Unit)에서는 CR_1을 시프트하고 라운드하여 결과 유효수 CR_2를 생성하고, 임약 CR_1의 최상위 자릿수가 0이면, 후교정 유닛에서의 정정된 결과값 CR_1을 유효 자릿수만큼 절삭한다. 반면, CR_1의 최상위 자릿수가 0이 아닌 수이면 주입 정정값(injection correction value)이 초기주입값(initial injection value)을 조정하기 위해 CR_1에 더해진다. 부호 유닛(Sign Unit)과 오버플로우 유닛(Overflow Unit)에서는 결과 값의 부호 비트 SR_1을 계산하고 오버플로우 신호를 발생시키며, 백워드 포맷 변환 유닛(Backward Format Conversion Unit)은 SR_1, ER_2, CR_2와 같은 최종 결과 값들을 IEEE 754-2008의 DPD 형식의 R_1으로 변환한다. 마지막으로, 후처리 유닛(Postprocessing Unit)은 입력 오퍼랜드와 연산 결과 값을 통해 특이 결과(special result)의 여부를 판단한다. 입력 오퍼랜드의 둘 중 하나가 NaN이거나 $\pm\infty$일 경우 결과 값은 해당 값으로 고정된다. 이와는 별도로, 오버플로우 플래그, 결과 값의 부호, 그리고 라운딩 모드에 따라 결과 값은 $\pm\infty$, $\pm MAXFLOAT$으로 고정될 수 있다.

Ⅳ. 제안한 십진 부동소수점 가산기

4.1 제안한 십진 부동소수점 가산기 구현

본 논문에서는 두 입력 오퍼랜드의 지수가 같을 때 L. K. Wang의 복잡한 오퍼랜드 정렬 알고리즘을 거치지 않고 유효수를 고정된 자리에 배치하여 연산함으로써 처리 속도를 향상시킬 수 있는 십진 부동소수점 가산기를 제안한다. L. K. Wang의 십진 부동소수점 가산기는 이미 최적화된 오퍼랜드 정렬 및 배치 알고리즘과 주입기반 라운딩 방식을 적용함으로써 고속으로 동작하도록 고안된 가산기이다. 본 논문에서 제안한 가산기는 이를 개선하여 두 입력 오퍼랜드의 지수 크기가 같을 때에는 고속의 데이터 경로인 근거리 경로(close path)에서 연산을 수행하고, 다를 때에는 원거리 경로(far path)에서 연산을 수행한다. 즉, 입력 오퍼랜드의 특성에 따라 데이터의 처리가 각각 다른 경로에서 이루어지

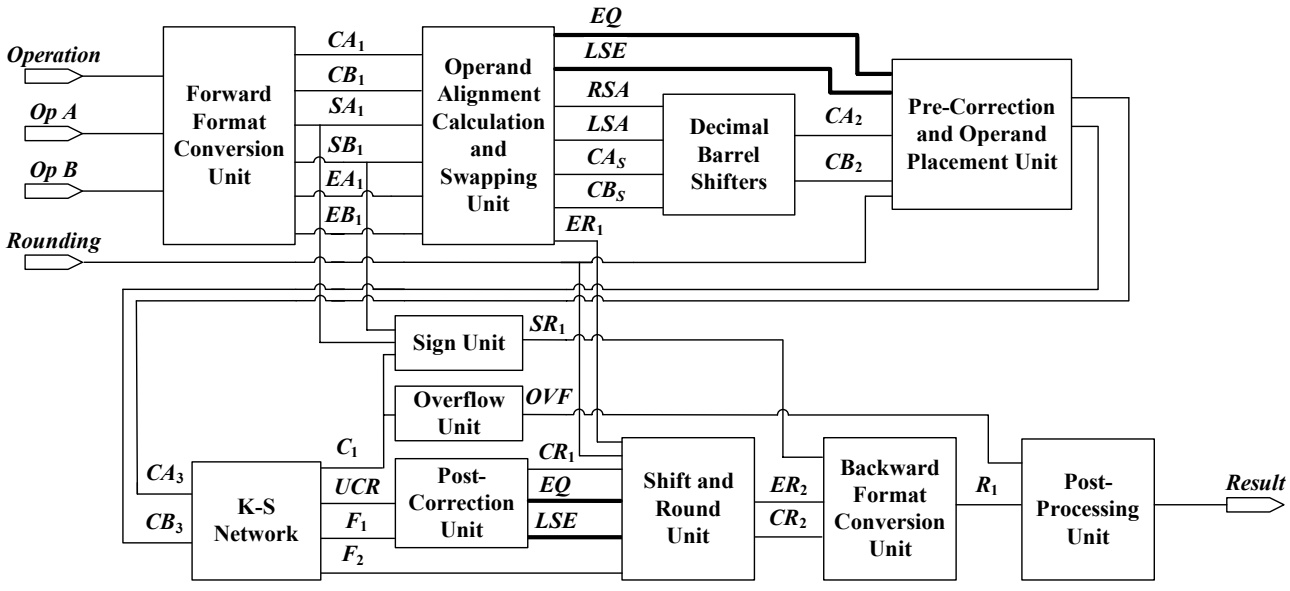

그림 2. 제안한 십진 부동소수점 가산기
Fig. 2. Proposed DFP adder.

므로 제안한 가산기는 가변시간(variable-latency)의 특성을 가진다. 그림 2는 제안한 십진 부동소수점 가산기의 구조를 나타낸다. L. K. Wang의 십진 부동소수점 가산기의 구조에 두 오퍼랜드의 지수가 같을 때 연산을 빠르게 처리할 수 있도록 EQ(*EQual*), LSE(*Left Shift Enable*) 신호가 추가된 형태를 가진다.

L. K. Wang의 십진 부동소수점 가산기에서는 순환 캐리 가산기를 이용하여 오퍼랜드에서 두 지수차의 절대값인 $|EA_1 - EB_1|$을 구한다. 이 방식은 면적과 전력 효율면에서는 뛰어나지만, 빠른 성능을 요구하는 시스템에는 적합하지 않다. 순환 캐리 가산기는 로직의 면적을 줄일 수는 있지만, 최상위 비트에서 생긴 순환 캐리(end around carry)가 최하위 비트로 들어가서 다시 최상위비트로 캐리가 전달되기까지 속도 저하가 발생하게 된다. 따라서 제안한 가산기는 그림 3과 같이 *CKS*(Conditional Kogge-Stone network) 모듈에서 $EA_1 - EB_1$과 $EB_1 - EA_1$을 K-S 가산기를 이용하여 계산하고, 이중에서 양의 값을 $|EA_1 - EB_1|$으로 출력하는 방식을 사용하였다. 뿐만 아니라, EA_1과 EB_1이 같을 때 고속의 데이터 경로인 근거리 경로로의 빠른 진입이 중요하므로 병렬로 처리되는 비교연산기(Comparator)가 EA_1, EB_1의 일치여부를 확인하여 EQ 신호를 발생시킨다. 좌우 시프트값과 임시 지수값이 계산되기 전에 EQ가 발생하므로, 연산 속도를 향상시킬 수 있다. EQ신호가 활성화되면 CA_S, CB_S, RSA, LSA, ER_1과 같은 신호들은 다른 유닛에서 돈케어(don't care)로 처리된다. 그림 4는 제안한 십진 부동소수점 가산기에 사용한 오퍼랜드 정렬 계산 및 교환 유닛이다. 이 유닛에서는 두 입력 오퍼랜드의 지수가 같같음을 의미하는 EQ신호를 발생시키고, 두 가수의 최상위 자릿수를 검사하여 둘 다 0일 경우 가수부를 좌

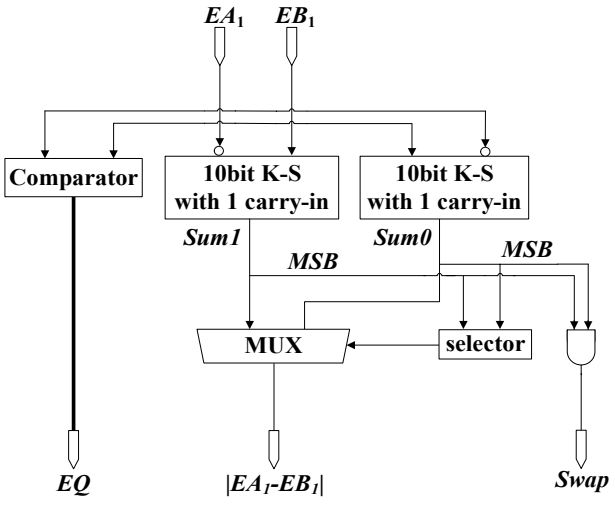

그림 3. EQ 신호 생성을 위한 CKS 모듈
Fig. 3. CKS module configuration for EQ generation.

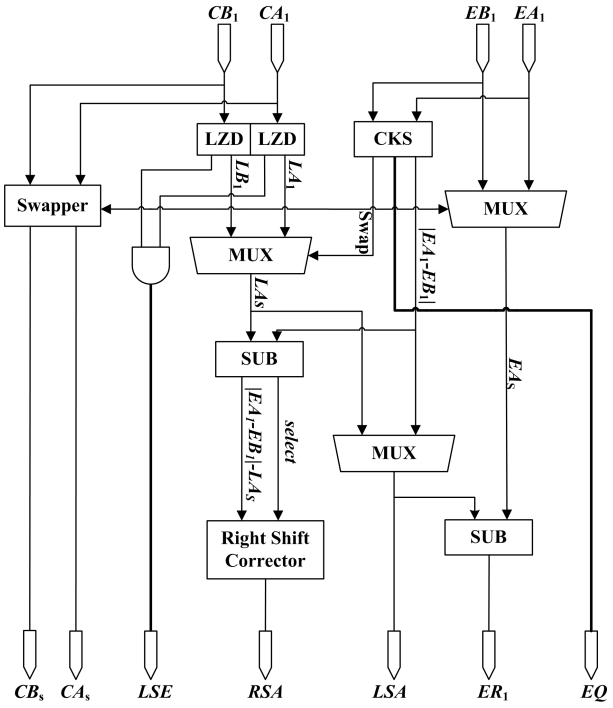

그림 4. 제안한 오퍼랜드 정렬 계산 및 교환 유닛
Fig. 4. Proposed OACSU.

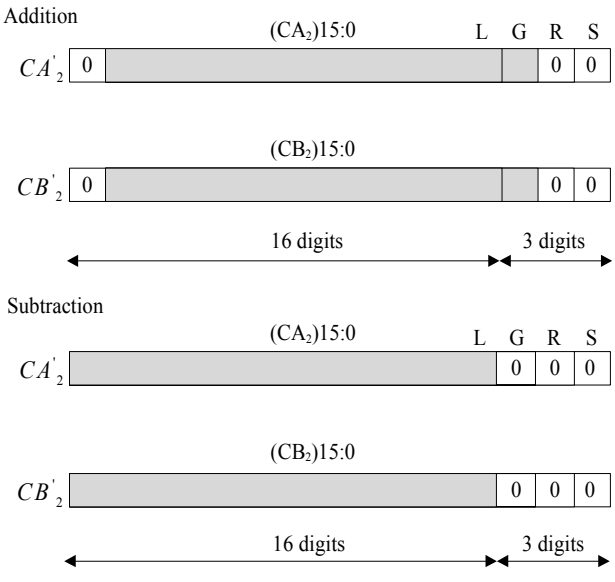

그림 5. 지수가 같을 경우의 오퍼랜드 배치
Fig. 5. Operand placement when the exponents are identical.

로 시프트 할 수 있도록 LSE신호를 발생시킨다. 본 논문에서는 이를 이용하여 지수가 같은 경우 이를 빠르게 처리하기 위한 근거리 경로와 원거리경로를 구성하여 가변시간 가산기를 구현하였다. 식 (2)와 (3)은 EQ와 LSE신호의 발생조건을 수식으로 정의한 것이다.

$$EQ=\begin{cases}1 & \text{if } EA_1 = EB_1 \\ 0 & \text{if } EA_1 \neq EB_1\end{cases} \tag{2}$$

$$LSE=\begin{cases}1 & \text{if } MSBs\ of\ CA_1 \\ & \quad \text{and } CB_1\ are\ both\ 0 \\ 0 & otherwise\end{cases} \tag{3}$$

만약 EQ가 1이면 CA_2와 CB_2는 배럴시프터 유닛을 거치지 않고 각각 CA_1과 CB_1에서 구할 수 있는데, 유효수를 최대한 보전하기 위해서 LSE가 1이면, 두 유효수를 좌로 시프트한다. 이에 대한 수식은 식 (4)와 같다.

$$CA_2=\begin{cases}left\_shift(CA_1,1) & \text{if } LSE\ is\ 1 \\ CA_1 & otherwise\end{cases} \tag{4}$$
$$CB_2=\begin{cases}left\_shift(CB_1,1) & \text{if } LSE\ is\ 1 \\ CB_1 & otherwise\end{cases}$$

유효수 정밀도 p가 16일 때, 그림 5와 같이 가산과 감산 연산에서의 유효수는 고정위치에 정렬될 수 있다. 가산 연산에서는 오버플로우가 일어날 수 있기 때문에 기존의 p+3 자릿수에서 최상위 자릿수를 비우고 우로 시프트하여 배치한다. 반면, 감산 연산의 경우에는 최상위 자릿수를 기준으로 정렬한다. 각 연산에서 유효수부를 제외한 G, R, S에는 주입값이 삽입되지 않고 각 자릿수는 0으로 채워진다.

L. K. Wang의 사전교정 및 오퍼랜드 배치 유닛의 주입기반 라운딩에서는 EOP가 감산 연산이고 RSA가 0일 때, 즉 십진 배럴 시프터를 통해서 시프트가 발생하지 않으면, 라운딩을 하지 않으므로 $flushing$신호를 발생시켜 주입값을 초기화 한다. 그러나 개선된 오퍼랜드 정렬 계산 및 교환 유닛에서는 두 오퍼랜드의 지수가 같을 경우 EQ 신호를 발생시키고 이 경우 십진 배럴 시프터에서 시프트 연산은 이루어지지 않는다.
이를 고려한 $flushing$신호의 발생 조건은 식 (5)와 같으며, 식에 사용된 연산 기호 $\wedge$는 AND를, $\vee$는 OR를 나타낸다.

$$flushing=(EOP\wedge\overline{RSA})\vee EQ \tag{5}$$

시프트 및 라운딩 유닛에서도 주입 수정값이 필요치 않을 경우 식 (6)과 같이 $flushing_{COR}$신호를 발생시켜 주입값을 초기화한다.

$$flushing_{COR}=EQ\wedge(EOP\vee LSE) \tag{6}$$

4.2 십진 부동소수점 연산의 예

4.2.1 가산 연산의 예

그림 6은 십진 부동소수점 가산 연산의 예이다. 예시에서 적용한 라운딩 모드는 부동소수점의 유효 자릿수 아래의 숫자가 중간값을 가질때 표현 가능한 가장 가까운 짝수를 취하는 roundTiesToEven이다. 두 지수의 크기가 같으므로 EQ는 1이 되고, 두 유효수의 최상위 자릿수가 0이므로 LSE 또한 1이 된다. EQ가 1이기 때문에 RSA, LSA는 돈케어가 되고, CA_S와 CB_S는 각각 CA_1과 CB_1이 된다. LSE가 1이므로, CA_2와 CB_2는 $left\_shift(CA_S,1)$, $left_shift(CB_S,1)$을 하여 배치되는데, 결과적으로$CA_2^{'}$, $CB_2^{'}$에 배치될 때는 최상위 자릿수를 비우고 우로 시프트하므로 원래 자리에 배치된다. 이때 $CA_2^{'}$의 R과 S에 주입되는 값은

	$CA_1\times10^{EA_1}$	=	0	000000013007524	$\times10^{75}$		
+	$CB_1\times10^{EB_1}$	=	0	000490000022566	$\times10^{75}$		
	$CA_S\times10^{EA_S}$	=	0	000000013007524	$\times10^{75}$		
+	$CB_S\times10^{EB_S}$	=	0	000490000022566	$\times10^{75}$		
	EQ	=	1				
	LSE	=	1				
	RSA	=	X				
	LSA	=	X	L	G	R	S
	$CA_2^{'}$	=	0	000000013007524	0	0	0
+	$CB_2^{'}$	=	0	000490000022566	0	0	0
	CA_3	=	6	666666679666DB8A	6	6	6
+	CB_3	=	0	000490000022566	0	0	0
	UCR	=	6	666AF66796900F0	6	6	6
	C_1	=	0	000000000001101	0	0	0
	F_1	=	0	000000000000000	0	0	0
	F_2	=	0	000000000000000			
	CR_1	=	0	000490013030090	0	0	0
+	INJ_COR	=			0	0	0
	$CR_2\times10^{EB_S}$	=	0	000490013030090	$\times10^{75}$		

그림 6. 십진 부동소수점 가산 연산의 예
Fig. 6. Example of DFP addition.

EQ가 1이므로 식 (6)에 의해 발생되는 $flushing$이 1이 되어 0으로 초기화된다. 그러므로 $CA_2^{'}$와 $CB_2^{'}$는 CA_S, CB_S와 같게 된다. $CA_2^{'}$는 사전교정 단계에서 각 자릿수에 6을 더해 교정되어 CA_3가 되고, CB_3와 더해져 K-S 네트워크에서 UCR, C_1, F_1, F_2를 생성한다. 후교정 단계에서는 $(C_1)_{i+1}$이 0인 UCR의 모든 자릿수에 대해 (-6)만큼 교정하여 CR_1을 구한다. 그리고 EQ와 LSE가 1이므로 식 (5)에 의해 $flushing_{COR}$은 1이 되고 INJ_COR(*Injection_Correction*)은 0으로 초기화된다. 이후 절삭되어 라운딩된 최종적인 유효수 결과값 CR_2는 0_000490013030090이 된다.

4.2.2 감산 연산의 예

그림 7은 십진 부동소수점 감산 연산의 예를 나타낸다. 예시에서 적용된 라운딩 모드는 부동소수점의 유효자릿수 아래의 숫자가 중간값을 가질 때 표현 가능한 가장 가까운 큰 수를 취하는 roundTiesToAway이다. 포워드 포맷 변환 유닛에 의해 BCD로 변환된 유효수 CA_1과 CB_1중 더 큰 지수를 가지는 유효수는 CB_1이다. 따라서 CB_1은 CA_S로, CA_1은 CB_S로 교환이 일어난다. 더 큰 지수를 갖는 유효수 CB_1의 선두 영의 개수 LA_S는 4이고, LSA는 $min(|EA_1-EB_1|, LA_S)$로 계산이 되는데 두 지수 차이인 $|EA_1-EB_1|$보다 LA_S가 작으므로 LA_S가 선택되어 4의 값을 가진다. RSA는 $min(max(|EA_1-EB_1|-LA_S, 0), p+3)$으로 계산되며, 11의 값을 가진다. ER_1은 EA_S-LSA로 계산되며, EA_S는 100이고 계산된 LSA는 4이므로 96의 값을 가진다. $CB_2^{'}$는 CB_S에서 RSA만큼 우로 시프트되고, $CA_2^{'}$는 CA_S에서 LSA만큼 좌로 시프트되며 주입값이 추가되어 배치가 완료된다. 여기에서 주입값 R과 S는 라운딩 모드가 roundTiesToAway이므로 (5, 0)이 선택된다. EOP는 감산 연산이므로 사전교정 단계에서 $CB_2^{'}$의 모든 비트는 반전되어 CB_3가 되고, K-S 네트워크를 통해 CA_3와 더해져 UCR, C_1, F_1, F_2를 생성한다. UCR은 EOP가 감산 연산이고, C_1의 최상위 비트가 1이기 때문에 양수이다. 따라서 F_1이 1인 모든 비트가 반전되고, $(C_1)_{i+1} \oplus (F_1)_i^3$가 0인 모든 자릿수가 (-6)만큼 보정되어 CR_1이 된다. CR_1의 최상위 자릿수가 2이기 때문에 이전에 삽입된 주입값은 더 이상 유효하지 않다. 따라서 주입 수정값을 G, R 자리에 삽입하여 합산하고 절삭하여 라운딩한다. 최종 결과값의 유효수 CR_2는 2_000000000010000과 같다.

	$CA_1 \times 10^{EA_1}$	=	9	000050000000000	$\times 10^{85}$		
−	$CB_1 \times 10^{EB_1}$	=	0	000200000000010	$\times 10^{100}$		
	$CA_S \times 10^{EA_S}$	=	0	000200000000010	$\times 10^{100}$		
−	$CB_S \times 10^{EB_S}$	=	9	000050000000000	$\times 10^{85}$		
	RSA	=	11				
	LSA	=	4	L	G	R	S
	$CA_2^{'}$	=	2	000000000100000	0	5	0
−	$CB_2^{'}$	=	0	000000000090000	5	0	0
	CA_3	=	2	000000000100000	0	5	0
+	CB_3	=	F	FFFFFFFFFF6FFFF	A	F	F
	UCR	=	2	00000000006FFFF	B	4	F
	C_1	=	1	111111111100000	0	1	0
	F_1	=	0	000000000000000	0	1	F
	F_2	=	0	000000000011111			
	CR_1	=	2	000000000009999	5	5	0
+	INJ_COR	=			4	5	0
	$CR_2 \times 10^{EB_S}$	=	2	000000000010000	$\times 10^{96}$		

그림 7. 십진 부동소수점 감산 연산의 예
Fig. 7. Example of DFP subtraction.

V. 실 험

5.1. 검증 및 시뮬레이션

본 논문에서 제안한 십진 부동소수점 가산기의 동작 검증을 위해 1000개의 특이값, 9000개의 지수가 같은 유한수, 90000개의 지수가 다른 유한수 오퍼랜드를 갖는 총 십만개의 테스트 벡터를 이용하여 올바른 동작을 검증하였다. Mentor Graphics사의 ModelSim 을 이용하여 RTL 레벨 시뮬레이션을 수행 하였고, Xilinx사의 Spartan3계열 FPGA 칩인 XC3S2000 FG456에서 구현하여 검증하였다[17].

5.2 실험 결과

정확한 성능 평가를 위해 동일한 환경에서 L. K. Wang의 십진 부동소수점 가산기와 오퍼랜드 정렬 계산 및 교환 유닛의 순환 캐리 가산기를 개선한 가산기,

그리고 본 논문에서 제안한 가산기를 ASIC 환경에서 합성하여 면적과 임계경로를 비교하였다. 합성툴은 Synopsys사의 Design Compiler를 이용하였고, 논리 합성을 위해 SMIC사의 0.18㎛ CMOS 공정 테크놀로지 라이브러리를 사용하였다. 표 2는 십진 부동소수점 가산기의 지연시간과 면적을 비교한 것이다. 임계경로상의 지연시간은 각각 ns단위와 FO4를 이용하여 계산하였는데, FO4는 디지털 CMOS 기술에서 공정에 독립적인 지연시간 단위(delay metric)이다. 면적은 ㎟단위를 사용하여 나타냈으며 NAND2 등가 게이트(equivalent gate)의 수로 나타내었다.

표 4는 비교대상과 제안한 가산기의 합성결과를 나타낸 표이다. L. K. Wang의 십진 부동소수점 가산기의 임계경로 지연시간은 4.27ns로 측정되었으며, FO4는 0.084ns로 계산한 결과 50.84 FO4와 같다. 또한 면적은 0.102949㎟이며 이는 9380개의 NAND2 등가 게이트와 같았다. L. K. Wang의 십진 부동소수점 가산기에서 오퍼랜드 정렬 계산 및 교환 유닛을 개선한 십진 부동소수점 가산기는 임계경로 지연시간이 3.81ns이고 면적은 0.110631㎟로 측정되었다. 이는 L. K. Wang의 십진 부동소수점 가산기와 비교하였을 때 임계경로 지연시간이 약 10.77% 감소되고 면적은 약 7.46% 증가된 결과를 보여준다. 마지막으로 본 논문에서 제안한 가변시간 십진 부동소수점 가산기의 임계경로 지연시간은 3.82ns이고, 면적은 0.111456㎟이다. 이는 L. K. Wang의 십진 부동소수점 가산기와 비교하였을 때 임계경로 지연시간이 약 10.54% 감소하고 면적은 8.26% 증가된 결과이다.

본 논문에서 제안한 가변시간 부동소수점 가산기에서 두 오퍼랜드의 지수가 다를 경우, 원거리 경로를 통해 연산이 이루어지며 결과값이 나올 때까지의 지연시간은 임계경로 지연시간인 3.82ns와 같다. $flushing$ 신호와 $flushing_{COR}$ 신호를 생성하기 위한 조건에 EQ, LSE 신호가 추가되어 0.01ns의 지연 시간이 증가하였다. 반면, 두 오퍼랜드가 같은 경우 오퍼랜드 정렬 계산 및 교환 유닛의 EQ 신호에 의해 활성화된 근거리 경로의 지연시간은 3.29ns임을 경로 분석을 통해 얻을 수 있었다.

그림 8은 L. K. Wang의 십진 부동소수점 가산기와, 순환캐리 가산기를 개선한 L. K. Wang의 가산기, 그리고 본 논문에서 제안한 가산기의 면적에 대해서 비교한 그래프이다. 가장 좌측의 막대그래프는 세 부동소수점 가산기의 각각의 면적 합계를 나타내며, 순환캐리 가산기를 개선한 L. K. Wang의 십진 부동소수점 가산기와 본 논문에서 제안한 가산기의 면적이 증가하였음을 나타낸다. 제안한 가산기의 각 유닛에서 전반적인 면적증가를 확인할 수 있으며, 크게 오퍼랜드 정렬 계

표 4. 십진 부동소수점 가산기의 지연시간과 면적 비교
Table 4. DFP adder comparison with respect to delay and area.

Designs	Delay (ns)	Delay (FO4)	Area (㎟)	Area (NAND2 Eq. Gates)
L. K. Wang's Adder	4.27	50.84	0.102949	9380
L. K. Wang's Adder with Refined OACSU	3.81	45.37	0.110631	10079
Proposed Variable-Latency Adder	3.82	45.48	0.111456	10155

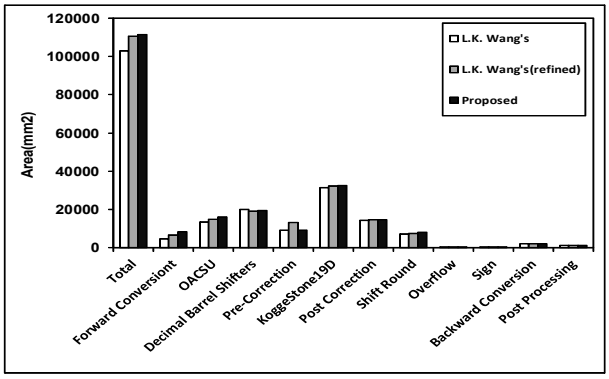

그림 8. 십진 부동소수점 가산기의 면적 비교
Fig. 8. Area comparison with DFP adder.

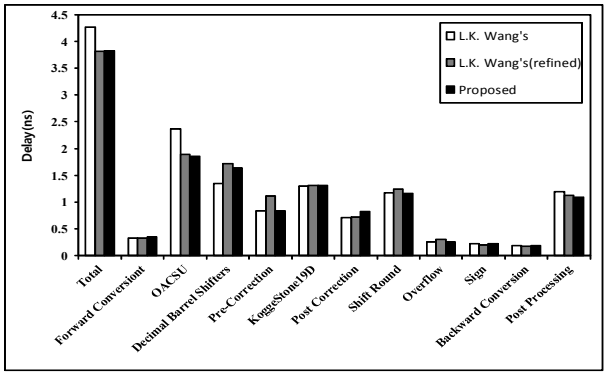

그림 9. 십진 부동소수점 가산기의 임계경로 지연시간 비교
Fig. 9. Critical path delay comparison with DFP adder.

산 및 교환 유닛과 사전교정 유닛에서 면적이 증가하였다. 포워드 포맷 변환 유닛과 같이 수정하지 않은 유닛의 면적량 변화는 로직의 계층구조를 변형하거나 필요하다면 일부분을 복제하는 합성툴의 지연시간 최적화 옵션에 의한 것이다.

그림 9는 L. K. Wang의 십진 부동소수점 가산기와, 순환캐리 가산기를 개선한 가산기, 그리고 본 논문에서 제안한 가산기의 임계경로상의 지연시간을 비교한 것이다. 가장 좌측의 막대그래프는 세 가산기의 총 임계경로 지연시간을 나타내며, 제안한 십진 부동소수점 가산기의 임계경로가 가장 짧은 것을 확인할 수 있다. 오퍼랜드 정렬 계산 및 교환 유닛의 순환캐리 가산기를 최적화함으로써 얻는 지연시간상의 이득이 가장 크고, 각 유닛의 개별적인 임계경로를 통해 추가된 로직만큼 임계경로가 증가한 것을 확인하였다.

그림 10은 십진 부동소수점에 입력되는 두 오퍼랜드에서 동일 크기 지수의 분포에 따른 스피드업 비율을 나타낸다. 스피드업은 식 (7)과 같이 이중 데이터 경로를 갖는 십진 부동소수점 가산기의 총 연산시간에 대한 단일 데이터 경로를 갖는 가산기의 총 연산 시간의 비율로 계산한다.

$$Speedup = \frac{Performance_{dual}}{Performance_{single}} = \frac{Total\ Execution\ Time_{single}}{Total\ Execution\ Time_{dual}} \quad (7)$$

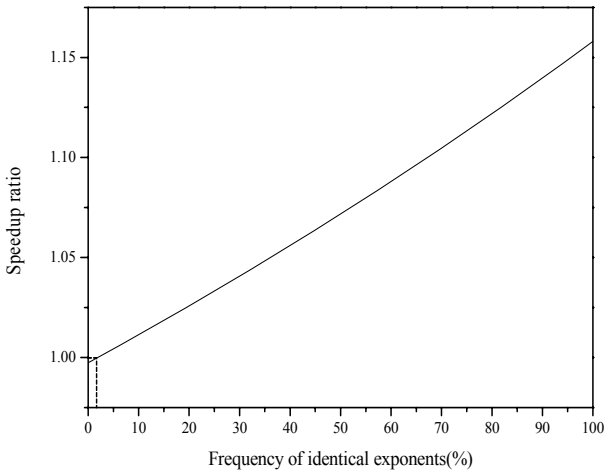

그림 10. 동일 크기의 지수 분포에 따른 스피드업 비율
Fig. 10. Speedup ratio with respect to frequency of identical exponents.

전체 오퍼랜드 중에서 동일 크기의 지수를 가지는 오퍼랜드의 비중이 높아질수록 연산 효율이 증가하는 것을 확인할 수 있고, 제안한 십진 부동소수점 가산기 구조는 동일 크기의 지수를 갖는 오퍼랜드의 비중이 2% 이상일 때 L. K. Wang의 가산기 대비 효용성이 높아지는 것을 알 수 있다.

VI. 결 론

이진 부동소수점은 충분한 기능과 성능을 제공하지만, 십진수를 정확하게 표현하지 못하는 근본적인 한계점이 있다. 십진 부동소수점은 정밀도내의 모든 십진수를 오차 없이 표현할 수 있고 십진기반 라운딩을 적용할 수 있기 때문에 기존의 이진 부동소수점이 가진 한계를 극복할 수 있으며, 오차없이 정확한 수치를 요구하는 시스템에서 사용된다.

본 논문에서는 L. K. Wang의 십진 부동소수점 가산기를 최적화하고, 오퍼랜드의 특성을 고려하여 동일한 크기의 지수를 갖는 오퍼랜드의 가산 및 감산 연산을 보다 빠르게 수행할 수 있는 가산기를 제안하였다. 제안한 십진 부동소수점 가산기는 L. K. Wang의 오퍼랜드 정렬 계획을 사용하지만 오퍼랜드의 지수가 같을 경우 정밀도를 보장하는 범위 내에서 고속의 데이터 경로를 통해 빠르게 연산을 수행한다. 합성 결과, 면적은 L. K. Wang의 가산기와 비교하여 8.26% 증가하였지만 전체 임계경로의 지연시간이 10.54% 감소하였다. 또한 같은 크기의 지수를 갖는 오퍼랜드를 연산할 때는 임계경로보다 13.65% 단축된 경로에서 연산이 이루어지며, 동일 크기의 지수를 가지는 오퍼랜드의 비중이 2% 이상일 때 L. K. Wang의 가산기 대비 효용성이 높아지는 것을 확인하였다.

참 고 문 헌

[1] IEEE, “IEEE 754-2008 Standard for Floating-Point Arithmetic”, 2008.
[2] M. F. Cowlishaw, The decNumber C Library v3.68, IBM Corportation, http://speleotrove.com/decimal/decnumber.pdf, 2011.
[3] Sun Microsystems, BigDecimal class, java 2 plaform standard edition 5.0, API specification, http://java.sun.com/j2se/1.5.0/docs/api/java/math/

BigDecimal.html, 2011.
[4] Intel Corportation, Intel Decimal Floating-Point Math Library, http://software.intel.com /en-us/articles/intel-decimal-floating-point-math-library/, 2011.
[5] L. Eisen, J. W. Ward III, H. -W. Tast, N. Mading, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, "IBM POWER6 Accelerators: VMX and DFU," *IBM J. Research and Development*, vol. 51, no. 6, pp. 663-684, 2007.
[6] B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, and R. Cargnoni, "IBM POWER7 multicore server processor," *IBM J. Research and Developement*, vol 55, no. 3, pp. 1-29, 2011.
[7] A. Y. Duale, M. H. Decker, H. -G. Zipperer, M. Aharoni, and T. J. Bohizic, "Decimal Floating-Point in z9: An Implementation and Testing Perspective," *IBM J. Research and Development*, vol. 51, no. 1/2, pp. 217-228, 2007.
[8] C. F. Webb, "IBM z10: The Next-Generation Mainframe Micro-processor," *IEEE Micro*, vol. 28, no. 2, pp. 19-29, Mar-Apr. 2008.
[9] L. K. Wang, C. Tsen, M. J. Schulte, and D. Jhalani, "Benchmarks and Performance Analysis of Decimal Floating-Point Applications," *Proc. 25th IEEE Int'l Conf. Computer Design*, pp. 164-170, 2007.
[10] P. Farmwald, "On the Design of High Performance Digital Arithmetic Units", PhD thesis, *Stanford Univ.*, Aug. 1981.
[11] S. Oberman, Design Issues in High-Performance Floating-Point Arithmetic Units. Ph.D. thesis, *Stanford University*, November 1996.
[12] S. Oberman, H. Al-Twaijry, M. Flynn, A SNAP project: design of floating-point arithmetic units, in: *Proceedings of 13th IEEE Symposium on Computer Arithmetic*, 1997, pp. 156 - 165.
[13] A. Akkas, Dual-mode floating-point adder architectures, *Journal of Systems Architecture*, 2008, 54(12): pp. 1129-1142.
[14] L. K. Wang, M. J. Schulte, J. D. Thompson, and N. Jairam, "Hardware Designs for Decimal Floating-Point Addition and Related Operations," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 322-355, Mar. 2009.
[15] J. Thompson, M.J. Schulte, and N. Karra, "A 64-Bit Decimal Floating-Point Adder," *Proc. IEEE CS Ann. Symp. VLSI (ISVLSI '04)*, pp. 297-298, Feb. 2004.
[16] G. Even and P. M. Seidel, "A Comparison of Three Rounding Algorithms for IEEE Floating-Point Multiplication," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 638-650, July 2000.
[17] Xilinx, Spartan-3 FPGA Family Data Sheet, http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf, 2009.

저 자 소 개

이 창 호(학생회원)
2008년 청주대학교 전자공학과 학사 졸업.
2010년 인하대학교 전자공학과 석사 졸업.
2010년～현재 인하대학교 전자공학과 박사과정.
<주관심분야 : 병렬 및 분산 처리 시스템, 컴퓨터 구조, Faulttolerant computing>

황 인 국(학생회원)
2011년 인하대학교 전자공학과 학사 졸업.
2011년～현재 인하대학교 전자공학과 석사과정
<주관심분야 : 컴퓨터 아키텍처, SoC 설계>

김 지 원(학생회원)
2007년 건양대학교 전자정보 공학과 학사 졸업.
2009년 인하대학교 전자공학과 석사 졸업.
2010년～현재 인하대학교 전자공학과 박사과정.
<주관심분야 : 멀티미디어 통신, 무선 통신, 센서 네트워크, 컴퓨터 네트워크>

최 상 방(평생회원)
1981년 한양대학교 전자공학과 학사 졸업.
1981～1986 LG 정보통신(주)
1988년 University of washinton 석사 졸업
1990년 University of washinton 박사 졸업.
1991년～현재 인하대학교 전자공학과 교수
<주관심분야 : 컴퓨터 아키텍처, 컴퓨터 네트워크, 무선 통신, 병렬 및 분산 처리 시스템>