JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 2 : 407~413 Feb. 2015

선행 제로 예측기를 이용한 고속 연산 십진 부동소수점 가산기 설계

윤형기 · 문대철*

Design of Decimal Floating-Point Adder for High Speed Operation with Leading Zero Anticipator

Hyoung-Kie Yun · Dai-Tchul Moon*

Department of Information and Communication Engineering, Hoseo University, Asan 336-795, Korea

요 약

본 논문에서 제안된 십진 부동소수점 가산기(decimal floating-point adder, DFPA)는 선행 제로 예측기(leading zero anticipator, LZA)를 이용해 임계 경로 단축을 통해 지연시간을 줄임으로서 연산 처리 속도를 향상시키는 파이프라인 구조로 설계하였다. 제안된 십진 부동소수점 가산기의 성능 평가 및 검증 환경은 시뮬레이션에 Flowrian 툴을 사용하였으며, 합성에는 QuartusII 툴 상에서 Cyclone III FPGA를 대상으로 지정하였다. 제안된 방식은 동일한 입력 데이터를 이용하여 기존에 제안된 설계 방식들과 시뮬레이션을 통해 비교 검증한 결과, L.K.Wang이 제안한 방식 및 기존 제안된 방식들보다 각각 11.2%, 5.9%의 성능이 향상되었다. 또한 연산 처리 속도 향상 및 임계 경로 상의 지연 소자의 수가 감소됨을 확인하였다.

ABSTRACT

In this paper, a DFPA(decimal floating-point adder) designed a pipeline structure that uses a LZA(leading zero anticipator) to reduce critical route to shorten delay to improve the speed of operation processing. The evaluation and verification of performance of proposed DFPA applied the Flowrian tool with simulation and Cyclone III FPGA was set as the target on the Quartus II tool for the synthesis. The proposed method compared and verified to proposed the other method using same input data. As a result, the performance of proposed method is improved 11.2% and 5.9% more than L.K.Wang's method and etc.. Also, it is confirmed that improvement of operation processing speed and reduction of the number of delay elements on critical path.

키워드 : 십진 부동소수점 가산기, 선행 제로 예측기, 파이프라인 구조, 고속 연산

Key word : Decimal Floating-Point Adder, Leading Zero Anticipator, Pipeline Processing Structure, High-speed Operation

접수일자 : 2014. 12. 12 심사완료일자 : 2014. 12. 17 게재확정일자 : 2014. 12. 31

* **Corresponding Author** Dai-Tchul Moon(E-mail:dtmoon@hoseo.edu, Tel:+82-41-540-5683)
Department of Information and Communication Engineering, Hoseo University, Asan 336-795, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.2.407 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

현대의 모든 시스템은 부동소수점 연산을 하드웨어 혹은 소프트웨어 형태로 지원하고 있다. 특히 이진 부동소수점 연산은 과학, 공학 등 정수 연산에 비하여 보다 높은 정밀도를 요구하는 영역에서 매우 중요한 역할을 하고 있다. 그러나 이진 부동소수점 연산은 소수점 이하의 계산 시 오차가 발생하는 단점이 있다. 이에 따라 IEEE에서는 기존의 이진 부동소수점 IEEE 754 표준에 십진 부동소수점 연산에 관한 내용과 누산기(fused multiply-add, FMA)에 관한 내용들을 새로이 추가하여 IEEE 754-2008을 발표하였다[1,2].

십진 부동 소수점 연산 방식은 인간이 주로 사용하는 십진수 체계를 그대로 사용함으로써 이진 체계로의 변환 오차를 줄인 정확한 연산 방식이므로 정밀도를 요구하는 분야에는 십진 부동소수점 연산이 반드시 필요하다. 그러나 십진 부동소수점 연산은 이진 부동소수점 연산에 비해 속도가 느리고 하드웨어 구현이 복잡하다는 단점이 존재한다. 그렇기 때문에 연산의 속도보다 정확도를 요구하는 시스템에서는 이진 부동소수점으로 계산된 근사값이 아니라 십진 부동소수점 연산으로 도출된 정확한 결과 값을 중요시하므로 십진 부동소수점 연산에 대한 중요성이 커진다. 특히 이진 부동소수점 연산은 금융, 세금, 상거래나 군사 무기와 같이 보다 높은 정밀도를 요구하는 영역에서의 계산 처리 시 소수점 이하 연산에서 오차가 발생하므로 적합하지 않다[3].

최근에는 십진 부동소수점 연산과 같은 정밀도를 중시하는 시스템이라 하여도 연산 처리 시 속도 문제를 무시할 수는 없기 때문에 다양한 연구가 이루어지고 있다. 그 중 가산기 분야는 가장 기본이 되면서 중요시 되는 모듈로서 많은 연구가 필요한 분야이다. 본 논문에서는 기존 십진 부동소수점 가산기에 선행 제로 예측기를 이용한 파이프라인 구조를 적용하여 기존 십진 부동소수점 연산 장치보다 고속의 연산이 가능하도록 설계하였다.

Ⅱ. L.K.Wang이 제안한 십진 부동소수점 가산기

L.K.Wang이 제안한 십진 부동소수점 가산기는 그림 1과 같이 구성된다.

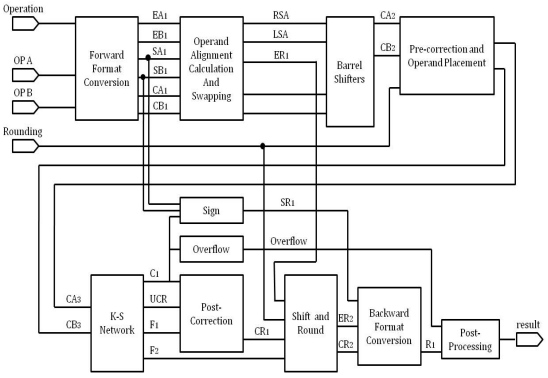

그림 1. L.K.Wang의 십진 부동소수점 가산기
Fig. 1 Decimal floating-point adder of L.K.Wang

먼저 전위 형식 변환 (forward format conversion) 장치에서는 피연산자 신호 OP A와 OP B를 받아서 부호 비트, 지수 비트, 가수 비트의 영역으로 나누어 연산 처리한다. 계산된 값을 피연산자 정렬 계산 및 교환 (operand alignment calculation and swapping) 장치를 통해 OP A의 가수와 OP B의 가수의 값을 비교 후 작은 값을 CA_S로, 큰 값을 CB_S로 지정해주고 이동 방향을 결정하는 RSA 및 LSA 신호를 발생한다. 그리고 십진 배럴 시프터 (barrel shifter)를 통해 유효 자리 수 이외의 신호가 제거된 후 출력된 CA_2 신호 및 CB_2와 라운딩 신호를 사전교정 및 피연산자 배치 (pre-correction and operand placement) 장치에서 받아 데이터 전달 시에 CA_2의 모든 비트에 6을 더하여 6초과 코드로 값을 교정해준다. 계산된 CA_3과 CB_3은 K-S 네트워크 (kogge-stone network) 장치를 통해 특이 값, 즉, NaN (not a number) 값이나 $\pm\infty$일 경우에는 특정한 값으로 출력된다. 부호 (sign) 장치에서는 두 피연산자의 부호 비트를 받아 K-S 네트워크를 통해 출력된 자리 올림 값 C_1과의 연산을 통해 최종 부호 값이 결정되며, 오버플로우 (overflow) 장치에서는 자리 올림 값 C_1을 받아 오버플로우를 발생시켜 후위 포맷 변환 (backward format conversion) 장치와 후처리 (post- processing) 장치를 통해 두 피연산자 OP A 및 OP B에 대한 가감산 결과 값이 결정된다[4-6].

Ⅲ. 제안된 십진 부동소수점 가산기

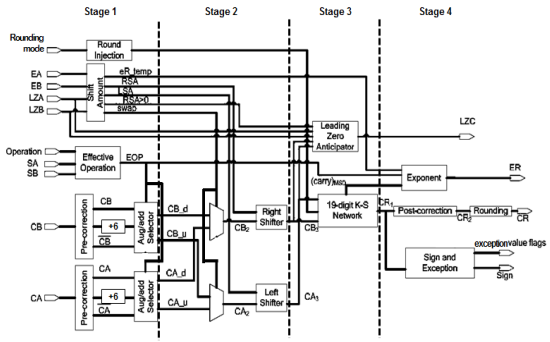

그림 2. 제안된 십진 부동소수점 가산기
Fig. 2 Proposed decimal floating-point adder

본 논문에서 제안된 십진 부동소수점 가산기는 L.K.Wang이 제안한 방식에 파이프라인 구조를 적용하여 기존 시스템보다 성능 향상을 통한 고속의 연산 처리가 가능하도록 설계하였다. 그림 2는 제안된 십진 부동소수점 가산기의 블록도이다. 제안된 연산 장치의 가산 연산 과정은 다음과 같이 이루어진다. 제안된 방식은 기존 L.K.Wang의 십진 부동소수점 가산기에서 전위 형식 변환 장치 및 후위 형식 변환 장치, 후처리 장치를 제거하고 사전교정 및 피연산자 배치 장치를 십진 배럴 시프터 앞으로 배치하여 사전교정 및 시프트를 결정하는 장치로 설계하였다.

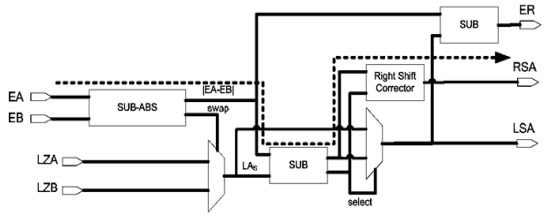

그림 3. 제안된 시프트 량 결정 장치의 블록도
Fig. 3 Block diagram of proposed shift amount unit

파이프라인 1단의 시프트 량 결정(shift amount) 장치는 피연산자 정렬 계산 및 교환 장치에서 교환(swapping) 기능만 더하여 시프트 방향을 결정하는 RSA 신호 및 LSA 신호, EA와 EB의 크기를 비교하여 교환 여부를 결정하는 swap 신호, 오류 유무에 대한 flag 신호인 eR_temp 신호, 정규화 시 발생하는 오류 교정을 위한 RSA>0 신호를 출력한다. 가수 영역 신호인 CA 와 CB는 사전교정(pre-correction) 장치를 통해 CA 및 CB를 6초과 코드로 변환한 신호와 원 신호인 CA와 CB 및 비트 반전 신호인 $\overline{CA}$ 및 $\overline{CB}$가 출력된다. 라운드 주입(round injection) 장치는 라운드 주입 값을 발생시켜 19자리 K-S 네트워크 장치로 전송한다. 효율 연산(effective operation) 장치는 부호 신호 및 연산(operation) 신호를 입력 받아 EOP 신호를 발생시키며 EOP 신호는 식 (1)을 통해 얻어 진다.

$$EOP = SA \oplus SB \oplus operation \qquad (1)$$

파이프라인 2단에서는 피가수 및 가수 선택 장치(augend/addend selector)를 통해 CA 및 CB 각각 상위 25비트의 CA_u, CB_u와 하위 25비트 CA_d, CB_d로 영역을 분할하여 가수 및 피가수로의 연산 처리를 위해 시프터로 전달된다. 이 때 EA≥EB 일 경우 우측 시프터로 CA_u와 CB_u의 값이 입력되며, EA<EB 일 경우 좌측 시프터로 CA_u 및 CB_u 값이 입력된다. 이 값들은 파이프라인 3단의 0의 숫자를 카운트하여 정규화 시 발생하는 오류를 정정하는 선행 제로 예측기로 입력되며, 시프터를 통해 출력된 가수 값 CA_3 와 CB_3가 K-S 네트워크 장치로 입력되어 가산 연산 처리된다. 파이프라인 4단에서는 선행 제로 예측기를 통해 발생된 교정 값 LZC 신호, eR_temp 신호 및 EOP 신호와 K-S 네크워크를 통해 발생된 자리 올림 값을 통해 최종 지수 값 ER 신호, 후교정과 라운딩을 통해 얻어진 최종 가수 값 CR 신호와 부호 및 ±∞, NaN, 0과 같은 특이 값의 발생 여부를 알 수 있는 플래그(flag) 신호를 부호 및 예외(sign and exception) 장치를 통해 얻을 수 있다[5,6].

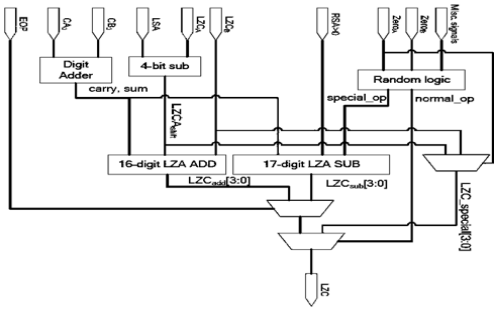

그림 4. 제안된 선행 제로 예측기
Fig. 4 Proposed leading zero anticipator(LZA)

그림 4는 제안된 선행 제로 예측기의 구조를 보여준다. 입력된 CA_3와 CB_3는 자릿수 가산기(digit adder)를 통해 합과 자리 올림 수가 발생하며 이 값들은 각각 16자리 선행 제로 예측 가산(LZA ADD) 장치와 17자리 선행 제로 예측 감산(LZA SUB) 장치로 입력된다. 4비트 감산(4-bit sub) 장치를 통해 발생된 시프트 수행된 CA에 대한 선행 제로 카운터($LZCA_{shift}$) 값과 B에 대한 선행 제로 카운터(LZCB) 값은 16자리 선행 제로 가산 장치로 입력되어 4비트 크기의 선행 제로 카운터(LZC_{add}[3:0]) 값이 출력된다. RSA>0 신호에서 1이 출력되면 17자리 선행 제로 감산 장치가 동작을 하며 랜덤 회로 (random logic) 에서 발생된 특이 피연산자(special_op) 신호를 통해 빌림 숫자로써 사용하고 마찬가지로 4비트 크기의 선행 제로 카운터(LZCsub[3:0]) 신호가 출력된다. EOP 신호가 0이면 LZC add 신호가, 1이면 LZC sub 신호가 출력되며, 최종적으로 정규화(normal_op) 신호를 통해 교정된 선행 제로 카운터(LZC) 값이 결정된다[7].

본 논문에서 사용된 피연산자 신호는 식 (2)와 같이 구성된다. 부동소수점 연산에 필요한 하나의 피연산자는 부호 및 지수, 가수 영역으로 나뉜다.

$$Value = -1^{(Sign)} \times 2^{(\mathrm{Exponent})} \times 1.(Significand) \quad (2)$$

표 1은 IEEE 754-2008 표준안에서 정의된 십진 부동소수점의 매개변수를 나타낸 것이다. decimal32, decimal64, decimal128의 저장 공간 비트수는 각각 32, 64, 128 비트이며 본 논문에서는 decimal64를 기준으로 피연산자 형식을 설정하였다[1].

표 1. 십진 부동소수점의 매개변수
Table. 1 Parameters of decimal floating point

Parameter	decimal32	decimal64	decimal128
Sign bit	1	1	1
Combination bit	5	5	5
Exponent bits	6	8	12
Significand bits	20	50	110
Precision	7	16	34
Maximum exponent	+96	+384	+6144
Minimum exponent	-95	-383	-6143
Exponent bias	-101	-398	-6176

본 논문에서 입력 데이터로 사용되는 피연산자는 표 2와 같이 구성된다. 63번 비트는 sign 비트, 62번 비트부터 55번 비트 영역은 지수(exponent)의 일부 영역을 나타내며 54번 비트부터 50번 비트 영역은 NaN, ± ∞ 또는 0과 같은 특이 값을 구분하는 데 사용된다. 가수(significand) 영역은 소수점 이하의 영역으로서 50비트의 크기의 영역을 갖는다.

표 2. 부호 및 지수, 가수 비트의 구성
Table. 2 Composition of sign, exponent and significand bits

Sign	Exponent	Exception	Significand
63	62……55	54……50	49……0

특이 값의 경우 표 3과 같이 값이 지정된다. ±∞나 NaN의 경우 인코딩의 다른 모든 비트는 무시되므로 단일 바이트 값으로 채워서 무한 수 또는 NaN의 배열을 초기화 할 수 있다.

표 3. 예외 값에 대한 지수, 가수 비트의 패킷
Table. 3 Packet of exponent, significand bits with exception value

Combination	Exponent Begins	Coefficient Digit	Other
00xxx	00	0xxx	-
01xxx	01	0xxx	-
10xxx	10	0xxx	-
1100x	00	100x	-
1101x	11	100x	-
1110x	00	100x	-
11110	-	-	±∞
11111	-	-	NaN. Sign bit ignored. First bit of exponent field determines if NaN is signaling.

그림 5는 제안된 십진 부동소수점 연산 처리의 알고리듬에 대한 흐름도를 나타낸다. 두 피연산자의 지수 값을 비교 후 시프트 방향 결정 시 RSA 신호가 0보다 크다면 선행 제로 예측기로 신호를 전달하여 0을 카운트하여 최종 결과값을 정정해준다. 가수의 상위 비트들과 하위 비트들의 가감산 후 결과 값에서 ±∞나 NaN과 같은 특이 값이 발생한다면 예외 장치에 저장된 결과 값을 출력하게 되며, 특이 값에 해당하지 않는다면 후

교정을 통해 가수 값 정정 후 라운딩 된 결과값을 얻을 수 있다[2].

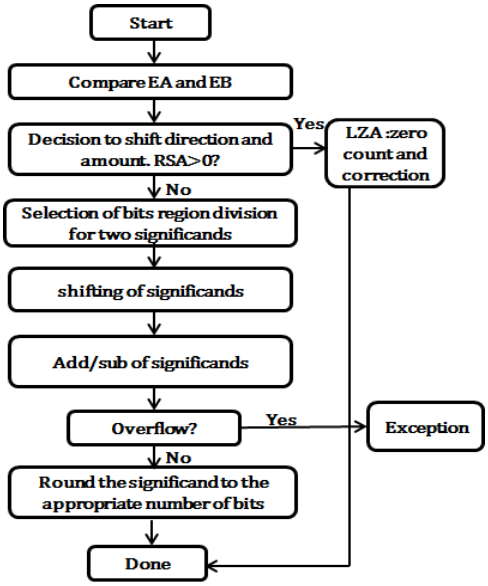

그림 5. 제안된 방식의 알고리듬 흐름도
Fig. 5 Flow chart of algorithm for proposed method

Ⅳ. 시뮬레이션 검증 및 성능 평가

본 논문에서 제안한 십진 부동소수점 가산기의 시뮬레이션 구현은 flowrian 툴을 이용하여 진행하였으며 QuartusII 툴을 이용하여 Cyclone III EP3C16F256CB 타겟 디바이스로 합성하였다. 십진 부동소수점 연산 장치 및 L.K.Wang이 제안한 방식과 참고문헌 [6]과 참고문헌 [8]에서 제안한 방식에 대한 시뮬레이션 결과를 비교 검증하였다[2,5,6,8].

```
script> runvsimsa -do run
Simulation Time : 425    ns <- start  time
Simulation Time : 116358 ns <- finish time
# Aldec, Inc. Riviera-PRO version 2009.10.7
```

(a)

```
script> runvsimsa -do run
Simulation Time : 380    ns <- start  time
Simulation Time : 109860 ns <- finish time
# Aldec, Inc. Riviera-PRO version 2009.10.7
```

(b)

```
script> runvsimsa -do run
Simulation Time : 378    ns <- start  time
Simulation Time : 106564 ns <- finish time
# Aldec, Inc. Riviera-PRO version 2009.10.7
```

(c)

```
script> runvsima -do run
Simulation Time : 354    ns <- start  time
Simulation Time : 103382 ns <- finish time
# Aldec, Inc. Riviera-PRO version 2009.10.7
```

(d)

그림 6. 제안된 방식 및 기존 방식들의 연산 처리 시간 비교 (a) L.K.Wang이 제안한 방식에 대한 스크립트 로그 (b) Reference [8]이 제안한 방식에 대한 스크립트 로그 (c) Reference [6]이 제안한 방식에 대한 스크립트 로그 (d) 본 논문에서 제안된 방식에 대한 스크립트 로그
Fig. 6 Comparison of operation processing time for proposed method and existing methods (a) Script log for proposed method of L.K.Wang (b) Script log for proposed method of Reference [7] (c) Script log for proposed method of Reference [5] (d) Script log for proposed method in this paper

그림 6은 본 논문에서 제안된 방식과 L.K.Wang이 제안한 십진 부동소수점 연산 장치 및 참고문헌 [6]과 [8]에서 동일한 2,000개의 샘플 데이터를 입력하였을 때 입력 지연 시간 및 최종 연산 처리 시간에 대한 스크립트 로그를 나타내었다. 세 방식의 입력 지연 시간을 포함한 데이터 연산 처리 시간을 비교해 본 결과 기존 L.K.Wang의 십진 부동소수점 연산 장치 보다 11.2%의 연산 처리 속도가 향상되었고, 참고문헌 [8] 보다 5.9%의 처리 속도 향상, 참고문헌 [6]보다 3%의 처리 속도가 향상된 것을 확인하였다.

그림 7은 각 상태에 맞는 동작 상태에 대한 시뮬레이션 결과이다. 기본 주파수 133MHz로 동작하도록 설계되었으며 ready 신호는 flag를 나타낸다. rmode 및 fpu_op 신호에 따라 가감산 및 오버플로우나 언더플로우 발생 시 가감산 동작을 수행한다. 또한 시뮬레이션 결과에서는 예외 상황 발생 (exception) 신호를 주어 발생 시 부정(inexact) 신호에 따라 NaN 또는 ∞값이 발생하도록 하였다. enable 단자로 1이 출력되면 fpu_op에 입력되는 신호에 맞는 연산 동작을 상승 클럭에 동기화하여 연산 처리하게 된다. 표 4는 제안된 십진 부동소수점 가산기의 합성 시 회로의 디바이스 사용량을 나타낸다.

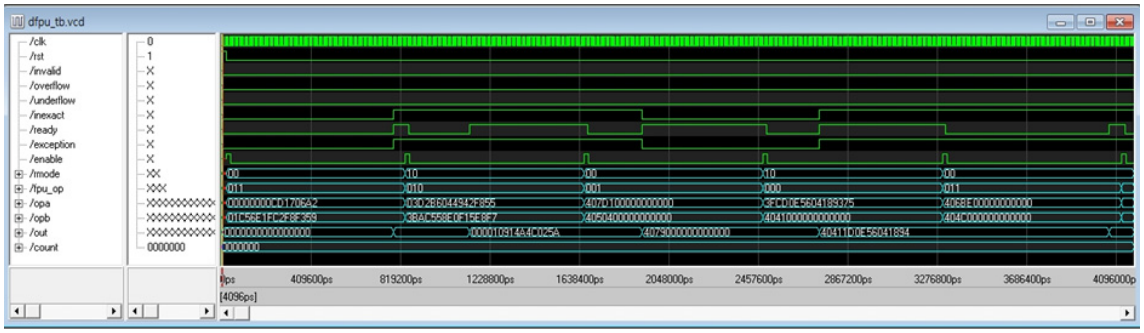

그림 7. 제안된 방식의 시뮬레이션 결과
Fig. 7 Results of Simulation for proposed method

표 4. 합성된 로직 디바이스 이용률
Table. 4 Utilization of the synthesized logic device

Logic Utilization	Used	Available	Utilization
Number of Slices	8837	33280	26%
Number of Slice Flip Flops	5035	66560	7%
Number of 4 input LUTs	16868	66560	25%
Number of bonded IOBs	206	633	32%
Number of MULT18X18s	14	104	13%
Number of GCLKs	1	8	12%

표 5는 그림 5에서 보여주는 바와 같이 500개, 1000개, 2000개의 동일한 샘플 데이터를 입력하여 시뮬레이션 검증한 결과에 대한 각각의 연산 시간을 비교한 결과를 나타내었다[5,6,8].

표 5. 샘플 수에 따른 연산 처리 시간 비교
Table. 5 Comparison of operation processing time as the number of samples

sample	L.K.Wang's method	Reference [8]	Reference [6]	Proposed method
500	29408 ns	27750 ns	26641 ns	25846ns
1000	58392 ns	55120 ns	53282 ns	51691 ns
2000	116358 ns	109860 ns	106564 ns	103382 ns

Ⅴ. 결 론

본 논문에서는 L.K.Wang이 제안한 십진 부동소수점 연산 장치를 기반으로 선행 제로 예측기를 이용한 파이프라인 구조를 적용하여 기존 십진 부동소수점 가산기보다 고속 연산이 가능하도록 구조를 제안하고 설계하였다. 시뮬레이션 결과는 L.K.Wang이 제안한 방식과 참고문헌 [6]과 [8]에서 제안한 방식 및 본 논문에서 제안된 방식의 입력으로 각각 입력 피연산자 신호 OP A와 OP B 에 500개, 1000개, 2000개의 동일한 입력 데이터를 이용하여 검증하였다. 제안된 병렬 십진 부동소수점 가산기의 시뮬레이션 결과 103,382 ns의 시간이 소요되었으며 같은 환경에서 L.K.Wang이 제안한 방식의 경우 116,358 ns의 시간이 소요되어 11.2% 정도의 처리 시간이 감소하였으며, 참고문헌 [8]에서 제안한 방식의 경우 109,860 ns의 시간이 소요됨으로 5.9% 정도의 처리 시간이 감소되었다. 또한 참고문헌 [6]에서 제안한 방식의 경우 106,564 ns의 시간이 소요됨으로 3% 정도의 처리 시간이 향상되었다. 또한 L.K.Wang의 십진 부동소수점 가산기와 비교하여 임계 경로 상의 지연 소자의 수는 14% 감소하였다.

추후 선행 제로 예측기의 대한 임계 경로에 걸리는 지연 시간 및 면적을 줄이는데 있어서 효율적인 연구를 통해, 보다 고속 연산 및 저면적의 십진 부동소수점 연산 장치에 대해 연구할 예정이다.

REFERENCES

[1] IEEE, *IEEE 754-2008 Standard for Floating-Point Arithmetic*, 2008.
[2] Lipsa Sahu, Ruby Dev, "An Efficient IEEE 754 Compliant

FPU using verilog", Department of Computer Science and Engineering National Institute of Technology Rourkela, 2012.

[3] M. Cowlishaw, "Decimal Floating-Point: Algorithm for Computers," *IEEE Symp. on Computer Arithmetic,* pp. 104-111, 2003.

[4] Jong-Hyeon Kim, *Parallel Computer Architecture,* New ed. Saeng-Neung Publish, 2010. 10.

[5] L. K. Wang, C. Tsen, M. J. Schulte, and D. Jhalani, "Benchmarks and Performance Analysis of Decimal Floating-Point Applications", *Computer Design, ICCD 2007. 25th International Conference on,* pp. 164-170, 2007.

[6] Hyoung-Kie Yun, Dai-Tchul Moon, "Design of Parallel Decimal Floating-Point Arithmetic Unit for High-speed Operations", *JKIICE Journal*, No. 17(12), pp. 2921-2926, 2013. 12.

[7] Mohamed H. Amin, Ahmed M. Eltantawy, Alhassan F. Khedr, Hossam A. H. Fahmy, Ahmed A. Naguib, "Efficient Decimal Leading Zero Anticipator Designs", *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, p. 139-143, 2011. 11.

[8] Steven Carlough, Adam Collura, Silvia Mueller, Michael Kroener, "The IBM zEnterprise-196 Decimal Floating-Point Accelerator", *Computer Arithmetic (ARITH), 2011 20th IEEE Symposium on,* PP. 139 - 146, 2011.

윤형기(Hyoung-Kie Yun)

2007 호서대학교 정보통신공학과 공학사
2009 호서대학교 정보통신공학과 공학석사
2010~현재 호서대학교 정보통신공학과 박사과정 재학
※관심분야 : DSP 및 영상신호처리 칩 설계, SoC 설계, 임베디드 시스템 설계

문대철(Dai-Tchul Moon)

1987 고려대학교 전자공학과 공학박사
1994 North Carolina State Univ. 연구교수
2005 Minnesota State Univ. Duluth 객원교수
1984~현재 호서대 정보통신공학과 교수
※관심분야 : DSP 및 영상신호처리 칩 설계, SoC 설계, 임베디드 시스템 설계, VLSI 신호처리