고속 연산을 위한 병렬 구조의 십진 부동소수점 연산 장치 설계

윤형기 · 문대철*

Design of Parallel Decimal Floating-Point Arithmetic Unit for High-speed Operations

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 연산 장치(decimal floating-point arithmetic unit, DFP)는 L.K.Wang에 의해 제안된 십진 부동소수점 유닛을 기반으로 하여 데이터의 병렬 처리를 통해 동일한 크기의 지수를 갖는 두 오퍼랜드의 가수 영역의 고속 연산을 지원하도록 재설계 하였다. 제안된 십진 부동소수점 연산 장치는 Xilinx ISE를 이용하여 xc2vp30-7ff896 타겟 디바이스로 합성하였으며 (주)시스템센트로이드의 Flowrian을 통해 시뮬레이션 검증하였다. 제안된 방식은 L.K.Wang에 의해 제안된 설계 방식 및 참고문헌 [6]의 설계 방식과 비교하여 동일한 입력 데이터를 이용하여 시뮬레이션 검증한 결과, L.K.Wang 방식보다 약 8.4%, 참고문헌 [6]의 방식보다 약 3% 정도의 처리 속도가 향상되었다.

ABSTRACT

In this paper, a decimal floating-point arithmetic unit(DFP) was proposed and redesigned to support high speed arithmetic operation employed parallel processing technique. The basic architecture of the proposed DFP was based on the L.K.Wang's DFP and improved it enabling high speed operation by parallel processing for two operands with same size of exponent. The proposed DFP was synthesized as a target device of xc2vp30-7ff896 using Xilinx ISE and verified by simulation using Flowrian tool of System Centroid co. Compared to L.K.Wang's DFP and reference [6]'s method, the proposed DFP improved data processing speed about 8.4% and 3% respectively in case of same input data.

키워드 : 십진 부동소수점 장치, DFP, 병렬 처리 구조, 고속 연산

Key word : Decimal Floating-Point Unit, DFP, Parallel Processing Structure, High-speed Arithmetic

접수일자 : 2013. 10. 01 심사완료일자 : 2013. 11. 12 게재확정일자 : 2013. 11. 25

* **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/jkice.2013.17.12.2921>

print ISSN: 2234-4772 online ISSN: 2288-4165

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

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. 그러므로 금융, 세금, 상거래와 같이 보다 높은 정밀도를 요구하는 영역에서의 계산 처리 시 적합하지 않다. 특히 금융권이나 군사 무기에서는 데이터들의 정확도가 매우 중요하기 때문에 이진 부동소수점 연산으로는 소수점 이하의 계산 시 오차로 인하여 결과가 정확히 표현되지 않는다. 이에 따라 IEEE에서는 기존의 이진 부동소수점 754 표준에 십진 부동소수점 연산에 관한 내용과 fused multiply-add에 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 방식이다. 그렇기 때문에 정밀도를 요구하는 분야에는 십진 부동소수점 연산이 반드시 필요하다. 십진 부동소수점 연산은 이진 부동소수점 연산에 비해 속도가 느리고 하드웨어 구현이 복잡하다는 단점이 존재한다. 그러나 연산의 정확도를 요구하는 시스템에서는 속도 문제보다 정확한 계산 값을 요구하므로 이진 부동소수점으로 계산된 근사값이 아니라 십진 부동소수점 연산으로 도출된 정확한 결과 값을 중요시하므로 십진 부동소수점 연산은 반드시 필요하다.

십진 부동소수점 연산과 같은 정밀도를 중시하는 시스템이라 하여도 연산 처리 시 속도 문제를 무시할 수는 없다. 그러므로 본 논문에서는 십진 부동소수점 연산 장치의 병렬 구조를 통한 속도 개선을 목표로 하여 기존 십진 부동소수점 연산 장치보다 고속의 연산이 가능하도록 설계하였다.

II. L.K.Wang의 십진 부동소수점 연산 장치

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

그림 1. L.K.Wang의 십진 부동소수점 연산 장치
Fig. 1 Decimal floating-point arithmetic unit of L.K.Wang

먼저 전위 포맷 변환 (forward format conversion) 유닛에서는 오퍼랜드 신호 및 부동소수점인 두 신호 OP A와 OP B를 받아서 부호비트, 지수비트, 가수 비트의 영역으로 나누어 연산 처리한다. 계산된 값을 오퍼랜드 정렬 계산 및 교환 (operand alignment calculation and swapping) 유닛을 통해 OP A의 가수와 OP B의 가수의 값을 비교 후 작은 값을 CA_5 로, 큰 값을 CB_5 로 지정해 주고 시프트 방향을 결정하는 RSA 및 LSA 신호를 발생한다. 그리고 배럴 시프터 (barrel shifter)를 통해 유효 자리 수 이외의 신호가 제거된 후 출력된 CA_2 신호 및 CB_2 와 라운딩 신호를 사전교정 및 오퍼랜드 배치 (pre-correction and operand placement) 유닛에서 받아 초기 값 정정 시 CA_2 의 모든 비트에 6을 더하여 값을 정정해준다. 계산된 CA_3 과 CB_3 은 k-s 네트워크 (kogge-stone network) 유닛을 통해 특이 결과의 경우, 즉, NaN (not a number) 값이나 $\pm \infty$ 일 경우에는 특정한 값으로 고정되어 출력된다. 부호 (sign) 유닛에서는 부호, 오버플로우 (overflow) 유닛에서는 발생하는 오버플로우 값을 각각 계산하여 후위 포맷 변환 (backward format conversion) 유닛과 후처리 (post-processing) 유닛을 통해 두 오퍼랜드 OP A 및 OP B에 대한 가산 결과 값이 결정된다[3].

III. 제안된 십진 부동소수점 연산 장치

본 논문에서 제안된 십진 부동소수점 연산 장치는 L.K.Wang이 제안한 방식보다 고속의 연산 처리가 가능하도록 재설계하였다.

그림 2. 제안된 십진 부동소수점 연산 장치
Fig. 2 Proposed decimal floating-point arithmetic unit

그림 2는 제안된 십진 부동소수점 연산 장치의 블록도이다. 제안된 방식은 오퍼랜드 정렬 계산 및 교환 유닛의 병렬 처리 방식으로, OP A와 OP B에 입력된 두 부동소수점의 가수 (exponent) 영역을 분할 처리하여 고속 연산이 가능하도록 설계되었다. 먼저 전위 포맷 변환 유닛에서 출력된 가수 영역 CA₁과 CB₁의 20비트 데이터를 상위 10비트와 하위 10비트로 나누어 각각 오퍼랜드 정렬 비교 및 교환 (operand alignment comparing and swapping) 유닛과 단순 가수 비트 계산을 수행하는 가수 계산 (significand calculation) 유닛으로 전송한다. CA₁의 하위 10비트와 CB₁의 하위 10비트의 가산 연산에서 발생하는 오버플로우는 상위 10비트끼리 연산된 결과와 함께 가수 오버플로우 가산기 및 DPD 인코더 (significand overflow adder and DPD encoder) 유닛에 입력된다. CA₁의 상위 10비트와 CB₁의 상위 10비트는 서로 연산 후 오버플로우 발생 시 오버플로우 된 비트를 버린다. 이 데이터들은 가수 오버플로우 가산기 및 DPD 인코더에서 상위 10비트에 하위 10비트 영역과 합산하여 IEEE 754-2008 표준 DPD (densely packed decimal) 인코딩 과정을 거친 후 CA₂ 및 CB₂를 얻을 수 있으며 두 오퍼랜드의 연산을 영역을 나누어 병렬로 처리함으로써 연산 처리 속도를 높였다[3-4].

본 논문에서 사용된 오퍼랜드는 식 (1)과 같이 구성된다. 부동소수점 연산에 필요한 하나의 오퍼랜드는 부호 및 지수, 가수 영역으로 나뉜다.

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

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 decimal32를 기준으로 오퍼랜드를 설정하였다[1].

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

Parameter	decimal32	decimal64	decimal128
Storage bit	32	64	128
Sign bit	1	1	1
Exponent bits	11	13	17
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 오퍼랜드는 표 2와 같이 구성된다. 31번 비트는 부호(sign) 비트, 30번 비트부터 25번 비트 영역은 지수(exponent)의 일부 영역을 나타내며 24번 비트부터 20번 비트 영역은 NaN, $\pm \infty$ 또는 0과 같은 특수 값을 구분하는 데 사용된다. 가수 (significand) 영역은 소수점 이하의 영역으로서 20비트의 크기를 갖는다.

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

Sign	Exponent	Significand
31	30.....20	19.....0

예외 값의 경우 표 3과 같이 값이 지정된다. $+\infty$ 일 경우 부호 비트는 0, 지수 영역의 모든 비트 값은 1, 가수 영역의 값은 모두 0으로 출력되며, $-\infty$ 일 경우 부호 비트 값은 1, 지수 영역의 모든 비트 값은 1, 가수 영역의 값은 모두 0으로 출력된다. sNaN (signaling not a number) 일 경우에는 가수 영역의 첫 번째 비트 값이 0으로 출력되거나 qNaN (quiet not a number) 일 경우 가수 영역의 첫 번째 비트 값은 1로 출력되며 지수 영역을 제외한 나머지 비트는 모두 don't care 처리된다.

표 3. 예외 값의 경우에 따른 부호, 지수, 가수 비트의 패킷
Table. 3 Packet of sign, exponent, significand bits with exception value

Sign	Exponent	Significand
31	30.....20	19.....0
0	1 1	1 1 1

Sign	Exponent	Significand
31	30.....20	19.....0
1	1 1	1 1 1

Sign	Exponent	Significand
31	30.....20	19.....0
x	1 1	0 x x

Sign	Exponent	Significand
31	30.....20	19.....0
x	1 1	1 x x

그림 3은 제안된 십진 부동소수점 가감산 연산의 알고리즘에 대한 흐름도를 나타낸다. 두 오퍼랜드의 지수 값을 비교 후 지수의 정밀도가 다를 경우 작은 지수 값이 큰 지수 값과 일치할 때까지 지수 영역을 우측 시프트를 취해주고 가수끼리 가산 또는 감산 연산을 한다. 우측 시프트를 통해 지수를 증가 시키거나 좌측 시프트로 지수를 감소시켜 정규화 된 결과 값을 얻는데 오버플로우가 발생하면 후처리 유닛으로 오버플로우를 넘겨준다. 그 후 결정된 비트 수만큼 가수를 라운딩 해준 뒤 정규화 된 결과 값을 도출하게 된다[2].

십진 부동소수점 연산 장치의 가수 영역 곱셈 연산은 각각 가수 영역을 두 부분으로 나누어 계산할 수 있도록 설계하여 처리 속도를 높였다. 가수의 곱셈 연산 처리의 경우는 입력 mul_a와 mul_b에 대하여 세 가지 영역으로 나누어 결과 값을 배열로 처리한 뒤, 예외 (exceptions) 유닛을 통해 합산 (summation) 연산한다. 이 때 결과 값의 최상위 비트 값이 1이 아니라면 왼쪽 시프트 처리를 해준다. 그 후 레지스터에 저장된 최상위 비트로부터 신호가 1이 나올 때까지의 0의 개수를 카운트하고 0들의 비트 수를 감소시켜 가수를 출력으로 내보낸다. 지수 영역은 입력 신호 OP A와 OP B의 합으로부터 연산된 OP A와 OP B 신호의 합이 31이 될 때까지 덧셈의 연산을 하여 두 연산의 뺄셈 값을 구한다. 만일 지수 연산 결과가 0보다 작게 되면 결과 값을 오른쪽

시프트 처리를 하여 exponent_under 레지스터에 저장한다. 오버플로우가 발생하는 경우 레지스터에 저장된 10번 비트의 값을 11번 비트로 시프트하여 연산 처리한다. 나눗셈 연산 처리도 곱셈 연산 처리와 연산 방식은 유사하다. 입력받는 두 신호에 대해 오른쪽 쉬프트 연산을 수행한다. 오버플로우 발생 시에는 레지스터에 저장된 20번 비트 값을 19번 비트로 시프트 연산 처리하게 된다[5-6].

그림 3. 제안된 방식의 알고리즘 흐름도

Fig. 3 Flow chart of algorithm for proposed method

IV. 시뮬레이션 검증 및 성능 평가

본 논문에서 제안한 병렬 십진 부동소수점 연산 장치의 하드웨어 시뮬레이션 구현은 xilinx ISE를 이용하여 xc2vp30-7ff896 타겟 디바이스로 합성하였으며 (주)시스템센트로이드의 flowrion 툴을 이용하여 제안된 병렬 십진 부동소수점 연산 장치 및 L.K.Wang이 제안한 방식과 참고문헌 [6]에서 제안한 방식에 대한 시뮬레이션 결과를 비교 검증하였다[2,4,7].

(a)

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

(b)

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

(c)

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

(d)

그림 4. 제안된 방식들의 연산 처리 시간 비교 (a) 제안된 방식과 기존 방식들의 데이터 처리 량에 따른 연산 처리 시간 비교 (b) L.K.Wang이 제안한 방식에 대한 script log (c) reference [6]이 제안한 방식에 대한 script log (d) 본 논문에서 제안된 방식에 대한 script log

Fig. 4 Comparison of operation processing time for proposed methods (a) Comparison of operation processing time for operation processing time of proposed method and existing methods (b) Script log for proposed method of L.K.Wang (c) Script log for proposed method of Reference [6] (d) Script log for proposed method in this paper

그림 4는 본 논문에서 제안된 방식과 L.K. Wang이 제안한 십진 부동소수점 연산 장치 및 참고문헌 [6]에서 동일한 2,000개의 샘플 데이터를 입력하였을 때 최종 연산 처리 시간에 대한 script log 를 나타내었다. 세 방식의 데이터 연산 처리 시간을 비교해 본 결과 기존 L.K.Wang의 십진 부동소수점 연산 장치 보다 약 8.4%의 연산 처리 속도가 향상되었고, 참고문헌 [6] 보다 약 3%의 처리 속도가 향상 된 것을 확인하였다.

그림 5는 각 상태에 맞는 동작 상태에 대한 시뮬레이션 결과이다. 제안된 방식은 다음과 같이 동작한다. fpu_op 신호가 000 인 경우 가산 연산, 001 인 경우 감산 연산, 010 인 경우 곱셈 연산, 011 인 경우 나눗셈 연산을 수행한다. ine 단자로 1이 출력되면 fpu_op에 입력되는 신호에 맞는 연산 동작을 상승 클럭에 동기화하여 연산 처리하게 된다.

표 4는 그림 4에서 보여주는 바와 같이 500개, 1000개, 2000개의 동일한 샘플 데이터를 입력하여 시뮬레이션 검증한 결과에 대한 각각의 테스트 시간을 비교한 결과를 나타내었다[4].

표 4. 샘플 수에 따른 연산 처리 시간 비교

Table. 4 Comparison of operation processing time as the number of samples

methods 샘플 수	L.K.Wang's method	Reference [6]	Proposed method
500	29408 ns	27750 ns	26641 ns
1000	58392 ns	55120 ns	53282 ns
2000	116358 ns	109860 ns	106564 ns

그림 5. 제안된 방식의 시뮬레이션 결과

Fig. 5 Results of Simulation for proposed method

V. 결 론

본 논문에서는 L.K.Wang이 제안한 십진 부동소수점 연산 장치를 기반으로 오퍼랜드 정렬 계산 및 교환 장치를 병렬 구조 형태로 설계하여 처리 속도를 향상시킬 수 있는 구조를 제안하였다. 시뮬레이션 결과는 L.K.Wang이 제안한 방식과 참고문헌 [6]에서 제안한 방식 및 본 논문에서 제안된 방식의 입력으로 각각 입력 오퍼랜드 OP A 와 OP B 에 500개, 1000개, 2000개의 동일한 입력 데이터를 이용하여 검증하였다. 제안된 병렬 십진 부동소수점 연산 장치의 시뮬레이션 결과 106,186 ns의 시간이 소요되었으며 같은 환경에서 L.K.Wang이 제안한 방식의 경우 115,933 ns의 시간이 소요되어 8.4% 정도의 처리 시간이 감소하였다. 또한 참고문헌 [6]에서 제안한 방식의 경우 109,480 ns의 시간이 소요됨으로 3% 정도의 처리 시간이 감소된 것을 확인하였다. 그러나 제안된 십진 부동소수점 연산 장치의 구현 시 가수 오버플로우 가산기 및 DPD 인코더의 추가로 하드웨어의 복잡도가 증가하는 단점이 있다. 추후 하드웨어 복잡도를 개선할 수 있도록 연구를 진행할 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] Jong-Hyeon Kim, *Parallel Computer Architecture*, New ed. Saeng-Neung Publish. 2010. 10.
- [4] 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.
- [5] Hickmann, B., Krioukov A., Schulte M., Erle M., "A parallel IEEE P754 decimal floating-point multiplier", *Computer Design, ICCD 2007. 25th International Conference on*, pp. 296-303, 2007. 10.
- [6] 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.
- [7] Chang-Ho Lee, Ji-Won Kim, In-Guk Hwang, and Sang-Bang Choi, "Design of Dual-Path Decimal Floating-Point Adder", *IEEK Journal*, No. 49(9), pp. 183-195, 2012. 9.

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