논문 2011-48SD-10-12

IEEE754 단정도·배정도를 지원하는 부동 소수점 변환기 설계

(Floating Point Converter Design Supporting Double/Single Precision of IEEE754)

박 상 수*, 김 현 필*, 이 용 석**

(Sang-su Park, Hyun-pil Kim, and Yong-surk Lee)

요 약

본 논문에서는 IEEE754 표준의 단정도 및 배정도를 지원하는 새로운 부동소수점 변환기를 제안하고 설계하였다. 제안된 변환기는 부호 있는 정수(32비트/64비트)와 부동소수점(단정도/배정도) 간 변환, 부호 없는 정수(32비트/64비트)를 부동소수점(단정도/배정도)으로의 변환, 부동소수점 단정도와 배정도 간 변환뿐만 아니라 부호 있는 고정소수점(32비트·64비트)과 부동소수점(단정도·배정도) 간 변환을 지원한다. 모든 입력 형태를 하나의 형태로 만드는 새로운 내부 형태를 정의함으로써 출력 형태의 표현 범위에 따른 오버플로우 검사를 쉽게 하도록 하였다. 내부 형태는 IEEE754 2008 표준에서 정의된 부동소수점 배정도의 확장된 형태(extended format)와 유사하다. 이 표준에서는 부동소수점 배정도의 확장된 형태(extended format)의 최소 지수부 비트폭은 15비트라고 명시하지만 제안된 컨버터를 구현하는데 11비트만으로도 충분하다. 또한 덧셈기가 대신 +1 증가기를 사용하면서 라운딩 연산과 음수의 정확한 표현이 가능하도록 변환기의 라운딩 스테이지를 최적화하였다. 단일 클럭 사이클 데이터패스와 5단 파이프라인 데이터패스를 설계하였다. 변환기의 두 데이터패스에 대한 HDL 모델을 기술한 후에 Synopsys design compiler를 사용하여 TSMC 180nm 공정 라이브러리로 합성하였다. 합성 결과의 셀 면적은 12,886 게이트(2입력 NAND 게이트 기준)이고 최대 동작 주파수는 411MHz이다.

Abstract

In this paper, we proposed and designed a novel floating point converter which supports single and double precisions of IEEE754 standard. The proposed convertor supports conversions between floating point number single/double precision and signed fixed point number(32bits/64bits) as well as conversions between signed integer(32bits/64bits) and floating point number single/double precision and conversions between floating point number single and double precisions. We defined a new internal format to convert various input types into one type so that overflow checking could be conducted easily according to range of output types. The internal format is similar to the extended format of floating point double precision defined in IEEE754 2008 standard. This standard specifies that minimum exponent bit-width of the extended format of floating point double precision is 15bits, but 11bits are enough to implement the proposed converting unit. Also, we optimized rounding stage of the convertor unit so that we could make it possible to operate rounding and represent correct negative numbers using an incrementer instead an adder. We designed single cycle data path and 5 cycles data path. After describing the HDL model for two data paths of the convertor, we synthesized them with TSMC 180nm technology library using Synopsys design compiler. Cell area of synthesis result occupies 12,886 gates(2 input NAND gate), and maximum operating frequency is 411MHz.

Keywords: Convertor, Internal format, Floating point number, Fixed pooint number, IEEE 754 standard

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electric & Electronics Engineering, Yonsei University)

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2011-0017625).

접수일자: 2011년4월25일, 수정완료일: 2011년10월18일

Ⅰ. 서 론

PC 및 TV, 개인 모바일 장치에서 3D 그래픽에 대한 요구가 증대되고, GPS나 이더넷에서의 네트워크 동기에 보다 정밀한 계산을 위해 부동소수점에 대한 중요성

이 증가하고 있다. 이런 디지털 신호 처리와 커뮤니케이션 알고리즘의 대부분은 부동소수점 연산에서 전개된 것이지만 좀 더 비용 효율적인 하드웨어 구현을 위하여 부동소수점을 정수나 고정 소수점으로 변환한다. 특히, 하나의 단말이 여러 통신 모듈을 수용하는 SDR (Software Defined Radio)과 H.264와 같은 영상 코덱에서 부동 소수점 연산을 정수로 변환하거나, 정수를 부동 소수점으로 변환, 또는 고정 소수점과 부동 소수점 간의 변환, 부동 소수점내에서의 단정도·배정도 간의 변환이 중요한 문제로 떠오르게 되었다[1~2]. 또한 IEEE754 2008에서는 기존 부동소수점 표준에 변환 수행(conversion operation)에 관련된 부분을 보완·강화하였다[3~4].

일반적으로, 부동 소수점 연산은 정수 연산에 비해 느리다. 만약 지수 승에 의한 넓은 범위의 수치를 다루는 경우가 아니라면 고정소수점을 사용하는 것이 유리하다. 제한된 범위의 수치를 다루는 응용 분야에서 대부분 고속 프로세싱을 위해 부동 소수점 연산을 고정 소수점이나 정수 연산으로 변환하여 처리한다. 따라서 부동소수점 연산기를 사용하지 않아도 되므로 하드웨어 비용 최소화와 속도 향상을 꾀할 수 있다.

실수를 표현하는 방법에는 부동소수점과 고정소수점 두 가지가 있다. 부동소수점은 대표적으로 IEEE754 표준에서 정의한 단정도·배정도 형식으로 표현된다. 이에 비해 고정소수점은 정수 부분과 소수 부분에 각각 고정된 비트수를 할당하여 실수를 표현하는 방식이다. 고정소수점은 일반적으로 정수와 동일한 비트 표현을 갖기 때문에 정수연산과 동일한 레벨에서 처리할 수 있다.

변환(conversion) 경우, 부동소수점과 정수 간 변환은 널리 사용되고 또한 상업용 IP로 제공하고 있다. 하지만 부동소수점과 고정소수점 간 변환은 대부분 소프트웨어를 통해 변환한다. 부동소수점과 고정소수점 간 변환기에 대한 연구는 소프트웨어를 사용하여 변환하거나 소프트웨어와 하드웨어를 결합하여 변환해 주는 방법이 제안되었다[2, 5~7]. 하지만 이런 변환 소프트웨어는 오버플로우가 발생하지 않으면서 출력 에러에 대한 오차를 만족시키는 변환을 해야 하므로 시간이 많이 걸린다.

본 논문에서는 부동소수점과 고정소수점 변환 시, 고정소수점의 소수 부분의 비트 폭 정보를 받아 변환하는 컨버팅 유닛을 제안한다. 또한 각 변환마다 변환 가능한 범위가 다르므로 여러 입력 형태를 하나의 형태로 만드는 내부 형태를 정의하고 출력 형태의 표현 범위에 따라 오버플로우 검사를 쉽게 처리하는 컨버팅 유닛을 제안한다. 제안된 컨버팅 유닛은 부호 있는 정수와 부동소수점 간 변환, 부호 없는 정수와 부동소수점 간 변환 그리고 부동소수점 단정도와 배정도 간 변환을 지원한다.

Ⅱ. 본 론

1. 관련 연구

그동안 부동 소수점의 하드웨어 관련 연구는 대부분 고속의 덧셈이나, 곱셈, MAC(Multiply and ACcumulation) 이나 MAF(Multiplier Adder Fused) 연산에 치중되어 왔다. 변환에 관련된 연구는 2008년 이전에 없다가 표준이 다시 재정된 후인 2009년에 Lance Saldanha의 하드웨어와 소프트웨어를 병행하여 처리하는 연구가 거의 유일하다[7].

Lance Saldanha의 변환 과정은 C나 C++의 고급 언어로 작성된 프로그램을 응용 프로그램의 성격에 맞게 자료 수집을 하여 적절한 고정 소수점의 범위를 결정한 후, 고정 소수점 라이브러리를 이용해 해당 프로그램의 부동 소수점을 고정 소수점으로 변환한다. 그 이후 고정 소수점으로 변환된 수는 전용의 하드웨어 엔진을 이용하여 빠르게 처리하는 방식을 사용한다. 하지만 이 방식은 우선 고정 소수점과 부동 소수점간의 변환만 적용가능하고, 둘째 사용을 위해 소프트웨어 라이브러리를 가지고 있어야 하며, 셋째 소프트웨어 설계자가 전용 coprocessor의 동작을 알고 있어야 한다는 단점이 있다.

여러 입력 형태를 여러 출력 형태로 변환을 지원하기 위해 각 변환에 따라 데이터 경로가 달라질 수 있다. 따라서 정수와 부동소수점 상호 변환 시, 정수에서 부동소수점으로 변환 블록, 부동소수점에서 정수로 변환 블록을 따로 분리하여 설계할 수도 있다[8]. 기능 블록을 따로 분리하면 기능이 간단하게 됨으로써 설계를 좀 더 쉽게 할 수 있다. 하지만 하드웨어로 구현했을 때, 따로 분리된 유닛으로 인해 하드웨어 면적이 늘어나고 사용되지 않는 리소스가 증가하게 된다.

본 논문에서는 부동소수점과 고정소수점 간 변환 시, 소수 부분의 비트 폭 알려주면 변환이 가능한 컨버터를 제안한다. 또한 여러 입력 형태를 하나의 내부 형태(internal format)로 정의함으로써 하나의 컨버팅 유닛을 통해 여러 변환을 수행하도록 설계한다.

2. 내부 형태 변환

일반적으로 변환에서 어떤 값이 부동소수점 표현, 정수·고정소수점 표현이 모두 가능하다면, 두 표현 간의 변환은 정확하게 이루어진다. 만약 변환하는 어떤 값이 원하는 표현에서 정확하게 표현할 수 없다면, 라운딩에 의해 결과값이 결정되고 그에 대응하는 플래그가 결정된다. 본 논문의 컨버팅 유닛에서 지원하는 변환 타입은 표 1과 같이 총 22가지 경우가 있다.

표 1의 22가지 변환을 수행하기 위해 각각의 경우에 대하여 하드웨어로 설계하거나 또는 부동소수점 단정도·배정도 간 변환, 부동소수점을 정수·고정소수점으로 변환, 정수·고정소수점을 부동소수점으로 변환 등 크게 3개의 기능으로 따로 분리하여 설계하는 것은 하드웨어 면적이 늘어나고 무엇보다 사용되지 않는 리소스가 늘어난다는 단점이 있다.

여러 변환을 효율적으로 실행하기 위해 내부적으로 일관되게 변환 형식을 처리하는 새로운 형태의 내부 형식(internal format)을 정의하였다. 입력으로 들어오는 값은 부동소수점(단정도·배정도), 부호 있는 고정소수점(32비트·64비트), 부호 있는 정수(32비트·64비트), 부호 없는 정수(32비트·64비트)로 8가지가 된다.

표 1. 제안하는 컨버팅 유닛에서 지원하는 변환
Table 1. Conversions supported by the proposed coverting unit.

변환전	변환후
float single precision	float double precision
	signed fixed point 32/64bits
	signed integer 32/64bits
float double precision	float single precision
	signed fixed point 32/64bits
	signed integer 32/64bits
signed fixed 32bits	float single/double precision
signed fixed 64bits	float single/double precision
singed integer 32bits	float single/double precision
signed integer 64bits	float single/double precision
unsigned integer 32bits	float single/double precision
unsigned integer 64bits	float single/double precision

가. 내부 형태(internal format) 정의

새로 정의된 내부 형태는 8가지 형태를 하나의 형태를 표현할 수 있어야 한다. 8가지 입력 수에서 표현 범위가 가장 넓은 것은 부동소수점 배정도이다. 그러나 부동소수점(배정도)과 정수(64비트) 간 변환, 부동소수점(배정도)와 고정소수점(64비트) 간 변환에서 64비트의 정수, 고정소수점을 부동소수점 배정도의 가수 부분(significant) 53비트에 넣을 수 없으므로, 가수 부분을 64비트로 확장해야 한다. 이는 IEEE754-2008에서 정의하는 부동소수점 배정도에 대한 확장된 형태(extended format) 가수 부분의 최소 비트 폭과 같다[4]. 확장된 형태(extended format) 지수 부분의 최소 비트 폭은 15비트로 정의하고 있지만 11비트만으로도 충분하다. 따라서 새로 정의된 내부 형태는 부동소수점 배정도와 같은 바이어스(bias)와 11비트의 지수(exponent)를 갖고 가수 부분은 64비트로 확장된다. 그림 1는 본 논문에서 제안하는 내부 형태를 보여준다.

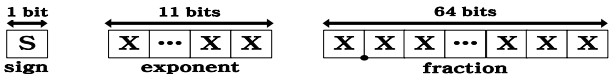

그림 1. 제안된 내부 형태(internal format)
Fig. 1. The form of the proposed internal format.

나. 내부 형태(internal format) 변환

IEEE 표준의 8가지 타입의 입력된 형식은 변환 형태에 따라 2가지의 의미를 지닐 수 있다. 정수, 고정소수점, 부동소수점을 부동소수점으로 변환할 때에는 내부 형태의 지수는 1023의 바이어스가 걸린 지수가 되고, 내부형태의 소수점은 가수 부분의 MSB(Most Significant Bit) 오른쪽에 위치한다. 반면에 부동소수점을 정수로 변환할 때에는 내부 형태의 지수는 오른쪽으로 시프트 하는 양이 되고, 이 내부 형태의 소수점은 가수 부분 LSB(Least Significant Bit)의 오른쪽에 위치한다. 그림 2는 데이터 형태 별 내부 형태 변환 및 출력 형태 변환을 보여준다. 입력 형태별로 내부형태 변환과정을 살펴보면 다음과 같다.

(1)정수를 내부형태로 변환과정

A.다음 수식처럼 정수 타입에 따라 64비트로 부호 확장한다.

$$I_{64bitext} = \begin{cases} f_{signextended}(32bit\ signed\ integer) \\ f_{zeroextended}(32bit\ unsigned\ integer) \\ 64bit\ signed\ integer \\ 64bit\ unsigned\ integer \end{cases} \quad (1)$$

B.내부 형태는 sign-magnitude 방식으로 표현되므로

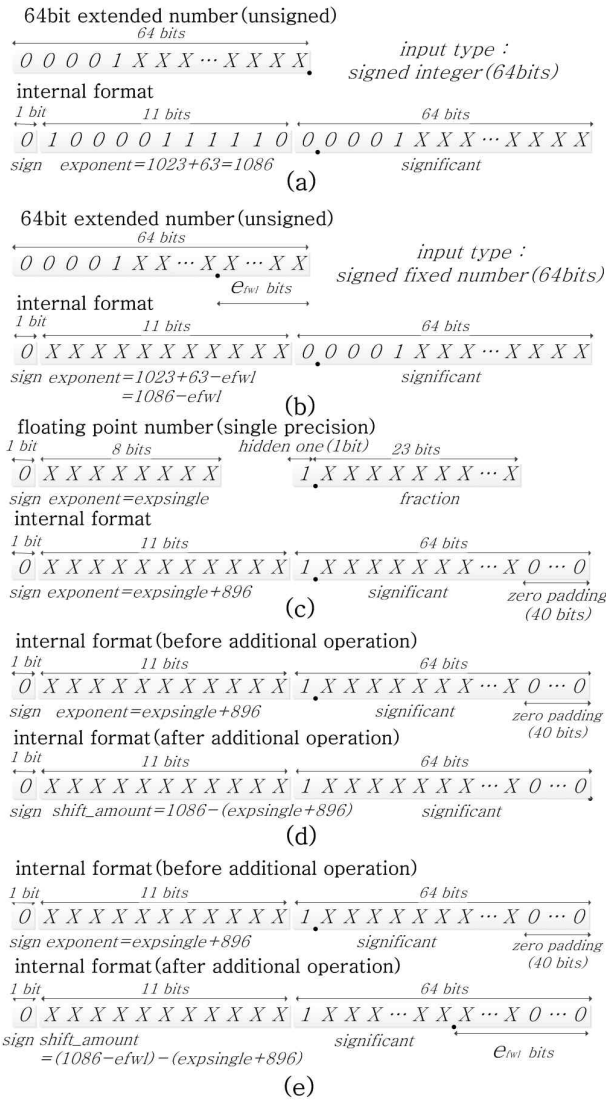

그림 2. 입력 형태에 따른 내부 형태 변환
Fig. 2. Converting into the internal format according to the input forms.

부호 없는 정수로 바꾼다.

$$I_{abs} = \begin{cases} I_{64bit\,ext} & \text{if } sign\ is\ positive \\ \overline{I_{64bit\,ext}} + 1 & \text{if } sign\ is\ negative \end{cases} \quad (2)$$

C.부호 없는 정수의 소수점을 내부 형태의 significant의 소수점 위치와 같은 위치에 놓고 이에 대한 보상을 해준다.

$$I_{abs} = I_{abs} \cdot 2^{-63} \cdot 2^{+63} = (I_{abs} \cdot 2^{-63}) \cdot 2^{+63} = S_{int} \cdot 2^{+63} \quad (3)$$

D.수식 (3)의 결과에 내부 형태의 바이어스를 가하고 내부 형태로 바꾸면 다음의 식과 같다.

$$e_{int} = e_{true} + bias = 63 + 1023 = 1089$$
$$v_{int} = (-1)^{Sign} \cdot 2^{e_{int} - 1023} \cdot S_{int} \quad (4)$$

그림 2 (a)는 64비트로 부호 확장된 정수를 내부 형태로의 변환 결과를 보여주고 있다.

(2)고정소수점을 내부형태로 변환과정

고정소수점은 소수 부분의 비트 폭을 알려주는 정보가 추가적으로 필요하다. 여기서, 소수 부분의 비트 폭 값을 efwl이라고 표기한다.

A.다음 수식처럼 고정소수점 타입에 따라 64비트로 부호 확장한다.

$$X_{64bit\,ext} = \begin{cases} f_{sign\,extend}(32bit\ signed\ fixed\ number) \\ 64bit\ signed\ fixed\ number \end{cases} \quad (5)$$

B.내부 형태는 sign-magnitude 방식으로 표현되므로 부호 없는 정수로 바꾼다.

$$X_{abs} = \begin{cases} X_{64bit\,ext} & \text{if } sign\ is\ positive \\ \overline{X_{64bit\,ext}} + 1 & \text{if } sign\ is\ negative \end{cases} \quad (6)$$

C.부호 없는 고정소수점 수의 소수점을 내부 형태의 significant의 소수점 위치와 같은 위치에 놓고 다음의 수식과 같이 이에 대한 보상을 해준다.

$$\begin{aligned} X_{abs} &= X_{abs} \cdot 2^{+efwl} \cdot 2^{-efwl} = (X_{abs} \cdot 2^{+efwl}) \cdot 2^{-efwl} \\ &= I_{abs} \cdot 2^{-efwl} = I_{abs} \cdot 2^{-63} \cdot 2^{+63} \cdot 2^{-efwl} \\ &= (I_{abs} \cdot 2^{-63}) \cdot 2^{63-efwl} = S_{int} \cdot 2^{63-efwl} \end{aligned} \quad (7)$$

D.수식 (7)의 결과에 내부 형태의 바이어스를 가하고 내부 형태로 바꾸면 다음의 식과 같다.

$$e_{int} = e_{true} + bias = 63 - efwl + 1023 = 1086 - efwl$$
$$v_{int} = (-1)^{Sign} \cdot 2^{e_{int} - 1023} \cdot S_{int} \quad (8)$$

그림 2 (b)는 64비트로 부호 확장된 고정소수점을 내부 형태로의 변환을 보여주고 있다.

(3)부동소수점을 내부 형태로 변환과정

배정도의 경우 내부 형태의 바이어스와 동일하고 배정도 significant를 내부 형태의 significant에 맞게 정렬하기만 하면 되므로 단정도의 경우만 살펴본다. 부동소수점 단정도는 다음과 같이 비정규화 된 수와 정규화 된 수로 구분할 수 있다.

$$v_{single}=\begin{cases}\text{if } e_{single}=0 \text{ and } f_{single}\neq 0 \\ \quad (-1)^{Sign}\cdot 2^{e_{single}-126}\cdot(0.f_{single}) \\ \text{if } 0<e_{single}<255 \\ \quad (-1)^{Sign}\cdot 2^{e_{single}-127}\cdot(1.f_{single})\end{cases} \tag{9}$$

부동소수점 단정도의 바이어스를 다음의 수식을 이용하여 내부 형태의 바이어스로 변경한다.

$$e_{int}=\begin{cases}1+896 & (\text{if } e_{single}=0 \text{ and } f_{single}\neq 0) \\ e_{single}+896 & (\text{if } 0<e_{single}<255)\end{cases} \tag{10}$$

$$v_{int}=\begin{cases}\text{if } e_{single}=0 \text{ and } f_{single}\neq 0 \\ (-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(0.f_{single}0\cdots 0) \\ \text{if } 0<e_{single}<255 \\ (-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{single}0\cdots 0)\end{cases} \tag{11}$$

그림 2 (c)는 부동소수점 단정도 정규화 된 수를 내부 형태로의 변환 결과를 보여주고 있다.

(4)부가적 연산

부동소수점을 정수나 고정소수점으로 변환하는 경우, 부동소수점 입력을 내부 형태에 맞게 정렬한 후, 추가적인 연산이 요구된다. 이는 소수점 위치 변경과 바이어스가 제거된 실제 지수 값에 따른 보상을 위한 쉬프트 양을 결정하는 것이다.

A.정수형으로 변환하는 경우

정수형으로 변환하기 위하여 내부 형태의 significant의 소수점을 LSB의 오른쪽에 위치시키고 지수부에 이에 대한 보상을 한다. 이에 대한 과정이 수식 (12)에 나타나 있다.

$$\begin{aligned}v_{int}&=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{int}) \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{int})\cdot 2^{+63}\cdot 2^{-63} \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{int}\cdot 2^{+63})\cdot 2^{-63} \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1f_{int})\cdot 2^{-63} \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1086}\cdot(1f_{int})\end{aligned} \tag{12}$$

그 다음 바이어스를 제거함으로써 실제 지수 값을 구하고 그에 해당하는 값만큼 쉬프트를 수행한다. 그림2 (d)는 내부 형태에서 정수형으로 변환하기 위하여 추가적인 연산이 적용된 내부 형태를 결과를 보여주고 있다. 여기서 최종 내부형태의 지수부는 양의 쉬프트 값이 된다는 점에 주의한다.

B.고정소수점 수로 변환하는 경우

고정소수점으로 변환하기 위하여 내부 형태 significant의 소수점을 고정소수점의 소수점 위치에 위치시키고 지수부에 이에 대한 보상을 한다. 이에 대한 과정이 수식 (13)에 나타나 있다. 여기서, efwl는 출력하고자 하는 고정소수점의 소수 부분 비트 폭 값이다.

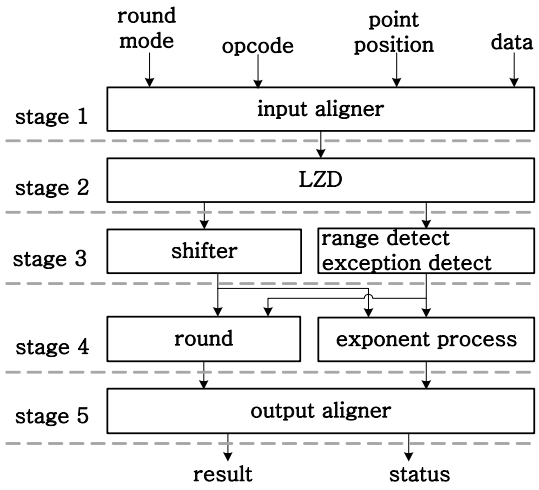

그림 4. 제안하는 컨버팅 블록
Fig. 4. The proposed converting block.

$$\begin{aligned}v_{int}&=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{int}) \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1023}\cdot(1.f_{int})\cdot 2^{+63}\cdot 2^{-63} \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1086}\cdot(1.f_{int}\cdot 2^{+63}) \\ &=(-1)^{Sign}\cdot 2^{e_{int}-1086}\cdot(1f_{int})\cdot 2^{+efwl}\cdot 2^{-efwl} \\ &=(-1)^{Sign}\cdot 2^{e_{int}+efwl-1086}\cdot(1f_{int}\cdot 2^{-efwl})\end{aligned} \tag{13}$$

그 다음 바이어스를 제거함으로써 실제 지수 값을 구하고 그에 해당하는 값만큼 쉬프트를 수행한다. 그림2 (e)는 내부 형태에서 고정소수점 수로 변환하기 위하여 추가적인 연산이 적용된 내부 형태를 결과를 보여주고 있다.

그림 3은 8가지 입력 형태를 내부 형태로 정리한 그래프이다. 내부 형태의 가수는 정규화 된 값(1≤ significant<2)이 된다. 그리고 x축은 1023바이어스가 걸린 내부 형태의 바이어스 된 지수이다. 따라서 8가지 입력 형태를 하나의 내부 형태로 바꿈으로써 언더플로·오버플로 검사를 쉽게 할 수 있다. 만약 내부 형태를 사용하지 않으면 22가지 변환에 대하여 표현 가능한 수의 범위인지 모두 확인해야 하고 이것은 설계를 복잡하게 만들 것이다.

3. 컨버팅 하드웨어의 구조

그림 4는 제안된 부동소수점 변환기 전체 블록을 보여준다. 첫 번째 단계에서는 8가지 입력 형태를 내부 형태로 변환하는 일을 수행한다. 즉, 부동소수점 단정

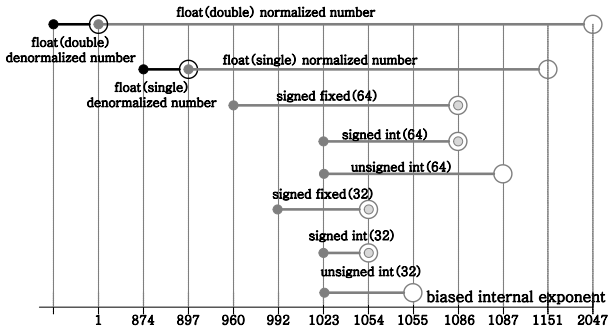

그림 3. 입력 형태에 대한 내부 형태 변환 범위 그래프
Fig. 3. The range graph of internal format for each input form.

도·배정도를 내부 형태로 변환하기 위해 바이어스를 가한다. 그리고 2의 보수(2's complement)로 표현하는 정수, 고정소수점은 부호크기(sign magnitude) 방식으로 표현되는 내부 형태로 변환해야 한다. 정수·고정소수점이 음수라면 절대값을 구하기 위한 2의 보수 연산이 필요하다. 또한 입력이 부동소수점인 경우 부동소수점 입력이 QNaN, SNaN, infinity, zero, 비정규화 된 수인지 체크한다. 특히, 부동소수점 단정도의 비정규화 된 수와 부동소수점 배정도 정규화 된 수의 상호 변환 시, 정확한 정렬을 위하여 바이어스 된 지수(biased exponent)가 0일 때와 1일 때의 소수점 위치가 같다는 것에 주의해야 한다. 비정규화 된 수를 정규화 된 수와 호환되도록 표현하기 위하여 비정규화 된 수의 바이어스 된 지수는 1로 세트해야 한다[9]. 따라서 부동소수점 단정도 비정규화 된 수가 내부 형태의 지수는 897이 된다.

두 번째 단계에서 leading zero detection을 수행한다[10]. 부동소수점에서 부호 있는 정수 또는 고정소수점으로 변환하는 경우, 세 번째 단계에서 가수를 오른쪽으로 쉬프트한다. 이 때 쉬프트 아웃되는 비트들이 발생할 수 있으며 정확한 라운딩 처리를 위하여 sticky 비트를 결정하는데 필요한 trailing zeros 카운트를 두 번째 단계에서 수행한다. 또한 비정규화 된 내부 형태 가수를 정규화하기 위하여 leading zero detection 수행한다.

세 번째 단계에서는 쉬프트를 수행한다. 양방향 배럴 시프터(barrel shifter)로 구성되어 있으며 LZD(Leading Zero Detector)에서 나온 왼쪽 쉬프트 양을 토대로 쉬프트를 수행한다. 또한 첫 번째 단계에서 결정된 오른쪽 쉬프트 양을 토대로 쉬프트를 수행한다. 동시에 LZD(Leading Zero Detector)에서 나온 trailing zeros count 수와 오른쪽 쉬프트 양을 비교하여 sticky비트를

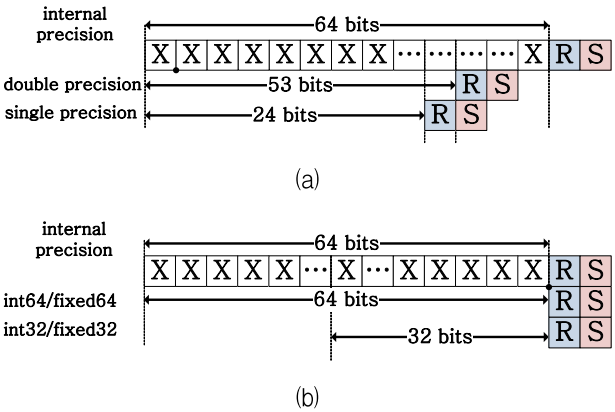

그림 5. 출력 형태에 대한 반올림 위치
(a) 부동소수점 단정도배정도에 대한 반올림 위치 (b) 정수고정소수점에 대한 반올림 위치
Fig. 5. Round-up Position for output forms.
(a) Round-up position for floating point single/double precison (b) Round-up Position for integer/fixed point number

결정한다.

range exception detect 블록에서는 내부 형태의 변환범위를 토대로 언더플로·오버플로가 발생하는지 미리 검사한다. 만약 오버플로가 발생하면 이 블록은 네 번째 단계에 오버플로 발생 여부를 알려주고 그에 해당하는 예외처리 정보를 전달해야 한다. 이는 변환할 때 범위를 크게 벗어나는 경우로서, 네 번째 단계에서 라운딩 모드에 따라 +1을 수행하는 라운드 처리만으로는 올바른 디폴트값을 출력할 수 없기 때문이다.

네 번째 단계에서는 라운딩 모드에 따라 라운딩을 수행한다. 만약, 세 번째 단계의 range exception detect 블록에서 미리 오버플로를 감지하면 라운딩 모드에 따라 디폴트값을 출력한다[3~4].

Exponent Process 블록은 바이어스를 변경하기 위하여 사용한다. 즉, 내부형태는 1023 만큼 바이어스가 걸린 지수를 가지므로 부동소수점 단정도 출력을 하는 경우, 바이어스를 127로 바꾸어 주기 위해 896을 빼주는 역할을 수행한다.

다섯 번째 단계는 라운딩에서 발생하는 가수의 오버플로에 대하여 지수에 1을 더한다. 또한 지수의 오버플로우가 발생했는지 체크한다. 그리고 출력 형태에 맞게 정렬하여 결과값을 출력한다. 그리고 여러 단계에 체크된 예외 상태(exception status)를 출력하게 된다.

4. 구현

파이프라인 5단계로 구성된 컨버팅 유닛에서 critical

path는 네 번째 단계이다. 네 번째 단계에서는 라운딩, post normalization, 그리고 2의 보수를 취할 수 있어야 한다.

4.1 덧셈기를 이용한 구현

1개의 덧셈기를 이용하여 라운딩, post normalization, 그리고 2의 보수 기능을 한 기능 블록에서 처리하도록 설계하였다. 이 단계에서는 세 번째 스테이지에서 시프트 된 가수와 round bit, sticky bit를 받는다.

Rounding decision 블록은 출력 형태와 라운딩 모드, round bit과 sticky bit을 고려하여 반올림 할 지, 버림 할 지 결정한다. 하지만 부동소수점 형태뿐만 아니라 부호 있는 정수 또는 고정소수점 등 여러 형태가 존재하므로 출력 형태에 따라 더해주는 값이 달라진다. 따라서 출력 형태에 따라 정확한 위치에서 반올림 할 수 있도록 constant biasing 블록이 필요하다. 그림 5 (a)에 나타낸 것처럼 출력이 배정도 부동소수점인 경우 반올림 할 때 0x800을 더해주면 된다. 그리고 출력이 단정도 부동소수점인 경우 반올림 할 때 0x10000000000을 더해주면 된다.

출력 형태가 정수 또는 고정소수점인 경우, 출력이 음수이면 2의 보수를 취해야 한다. 동시에 라운딩 모드에 따라 반올림해야 할 경우도 발생한다. 1의 보수 블록은 출력이 부호 있는 정수 또는 고정소수점이면서 부호가 음수인 경우에 1의 보수된 값을 출력한다. 1's complement = 2's complement -1이므로 1의 보수를 취함으로써 round toward negative infinity가 취해진 상태로, 절대값으로는 무조건 반올림 된 상태가 된다. 따라서 rounding decision 블록에서 반올림을 하도록 결정하면 그냥 1의 보수 값을 출력하면 된다. 만약 rounding decision 블록에서 버림을 하도록 결정하면 반드시 +1을 해주어야 한다.

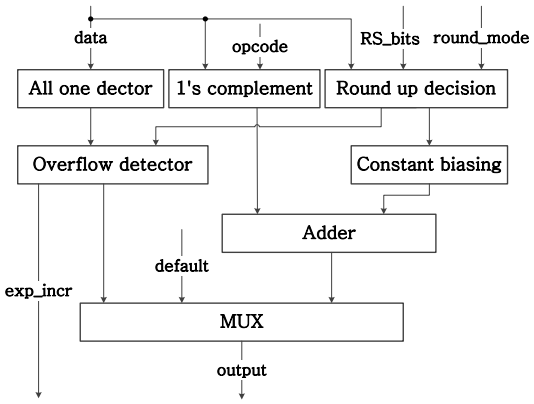

그림 6. 덧셈기를 이용한 라운딩 블록
Fig. 6. The rounding block using an adder.

그림 6은 덧셈기를 이용한 라운딩 블록을 나타낸다. 이 구조는 출력 형태에 따라 반올림 위치가 다르기 때문에 위치 정보를 테이블로 가지고 있어야 한다. 또한 라운딩 연산은 +1 연산이면 충분하지만 반올림 위치가 고정되지 않아 덧셈기를 필요로 하는 구조이다.

4.2 인크리멘터를 이용한 구현

덧셈기 대신 인크리멘터를 사용하기 위해서는 우선 반올림 위치를 고정시켜야 한다. 즉, 인크리멘터를 통과하기 전에 출력 형태에 따라 미리 정렬시키는 것이다. 해당하는 출력 형태의 LSB를 인트리멘터의 입력의 최하위 비트에 오도록 함으로써 라운딩에 따라 반올림을 할 수 있다. 따라서 덧셈기 대신 인크리멘터를 사용하여 구현할 수 있다. 그림 7은 인크리멘터를 사용하여 구현된 라운딩 블록을 나타낸다.

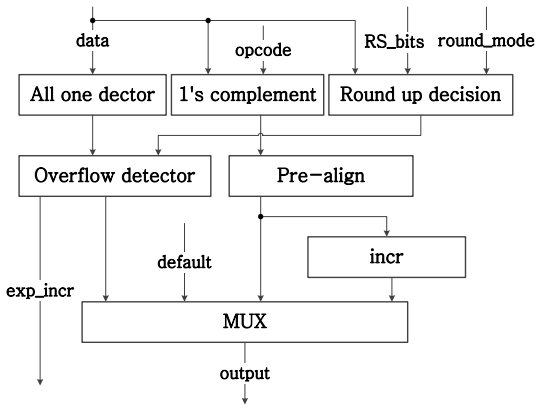

그림 7. 인크리멘터를 이용한 라운딩 블록
Fig. 7. The rounding block using an incrementor.

Ⅲ. 실 험

1. 검증 및 시뮬레이션

랜덤 벡터로 추출한 데이터는 전체 검증 구간 중 평균에 치중된 정규분포로 나타나기 때문에 라운딩 모드에 따라 오버플로우·언더플로우가 발생하거나 발생하지 않을 수도 있는 근처의 값들을 입력벡터로 생성하지 못할 수도 있다. 따라서 본 논문에서는 랜덤 벡터 100만 개뿐만 아니라 의도적으로 각 입력 데이터 형식 별로 오버플로·언더플로가 발생할 수 있는 최대값과 최소값의 경계 부근의 값들을 샘플링하여 검증 벡터로 사용하였다. 이에 대한 결과값과 예외 처리 플래그 결과를

SoftFloat이라는 소프트웨어의 결과와 비교하여 검증을 실시하였다[11].

2. 합성

본 논문에서 제안한 구조를 구현하기 위해 Verilog HDL로 설계하여 tsmc 180nm(slow)공정에서 Synopsys 합성 툴을 사용하여 합성하였다. 최근의 고성능 FPU는 여러 단의 파이프라인으로 구성되기 때문에 단일 클록 사이클로 동작하는 구조와 5단 파이프라인을 가진 구조의 두 가지 datapath로 나누어 합성을 진행하였다. 합성 결과는 표 2와 같다. 또한 5단 파이프라인 각 스테이지에서의 면적과 data arrive time을 표 3에서 보여준다.

컨버팅 유닛에 대한 하드웨어 구현 연구로 컨버팅 유닛을 FPGA 합성 툴을 사용하여 파이프라인 구조로 구현한 논문이 있다[8]. 하지만 제안된 컨버터와 비교하기에는 너무 제한된 변환을 지원하며 또한 파이프라인 스테이지 수도 다르기 때문에 직접적인 비교가 어렵다. 그래서 내부 형태를 사용했을 때 얼마만큼 하드웨어 면적을 줄일 수 있는지 확인하기 위하여 Synopsys DesignWare Building Block IP에서 제공하는 Floating point Converter data path를 합성하여 하드웨어 면적 결과를 비교하였다[12]. Synopsys IP를 사용한 총 12가지 변환에 대한 합성 결과를 표 4에서 보여준다. Synopsys DesignWare에서 제공하는 Converter datapath는 IEEE754 호환성 여부, 입력 형태, 출력 형태를 설정함으로써 각 변환마다 분리된 여러 블록을 합성하게 된다. Synopsys DesignWare Coverter datapath를 합성할 때 timing constraint를 제안된 단일 클럭 사이클 컨버팅 유닛의 data arrival time으로 설정함으로써, area 최적화를 수행할 수 있도록 하였다. 이를 제외한 다른 constraints, environment, optimization level 설정은 같다.

두 설계에 대한 합성 면적 비교 결과를 표 5에 나타내었다. 제안된 컨버팅 유닛이 더 많은 변환을 지원하면서도 게이트 면적이 더 작다는 것을 알 수 있다. 이는 여러 입력 형태를 하나의 내부 형태로 변환함으로써 컨

표 2. 단일 클럭 사이클 데이터패스와 5단 파이프라인 데이터패스의 합성 결과 비교
Table 2. Comparison of area and data arrival time for single clock cycle datapath and 5 stage pipelined datapath.

datapath	area [2-input NAND gate]	data arrival time[ns]
single cycle	8,965.97	7.03
5 cycles	12,878.50	2.25

표 3. 5단 파이프라인 데이터패스의 각 스테이지 별 합성 결과
Table 3. The synthesis result for each stage of 5 stage pipelined datapath.

stage	area [2-input NAND gate]	data arrival time[ns]
1 stage input aligner	2,394.34	2.03
2 stage leading zero detect	1,713.76	1.87
3 stage shifter/range & exception detect	4,240.18	1.87
4 stage rounding/exp process	3,415.54	2.25
5 stage output aligner	1,114.67	2.12

표 4. Synopsys IP Converter datapath 합성 결과
Table 4. The synthesis results for Synoptys IP Converter datapath.

Synopsys DW IP name	Converting Type	area [2input NAND gate]	data arrival time [ns]
DW_fp_flt2i	float(single)→signed int(32bits)	726.82	6.91
DW_fp_flt2i	float(single)→signed int(64bits)	1215.47	7.02
DW_fp_flt2i	float(double)→signed int(32bits)	1051.48	7.03
DW_fp_flt2i	float(double)→signed int(64bits)	1549.77	7.03
DW_fp_i2flt	signed int(32bits)→float(single)	716.51	7.03
DW_fp_i2flt	signed int(32bits)→float(double)	688.56	7.02
DW_fp_i2flt	signed int(64bits)→float(single)	1801.58	7.03
DW_fp_i2flt	signed int(64bits)→float(double)	1589.02	7.03
DW_fp_i2flt	unsigned int(32bits)→float(single)	548.86	7.03
DW_fp_i2flt	unsigned int(32bits)→float(double)	333.31	7.01
DW_fp_i2flt	unsigned int(64bits)→float(single)	993.26	7.03
DW_fp_i2flt	unsigned int(64bits)→float(double)	1160.91	7.02
Total		12,375.55	7.03

표 5. 합성 면적 결과 비교
Table 5. Comparison of the area for two designs.

design	the number of functions [#]	data arrival time [ns]	area [2-input NAND gate]	relative area
the proposed converting unit	22	7.03	8,965.97	0.72
Synopsys DW IP	12	7.03	12,375.55	1

버팅 유닛 내의 리소스들을 최대한 활용하기 때문이다. 제안된 파이프라인 컨버팅 유닛의 data arrival time에 플롭플롭의 셋업 시간(0.18ns)을 고려하여 최대 동작 주파수를 구하면 411MHz가 된다.

Ⅳ. 결 론

최근 임베디드 시스템에서도 다양한 애플리케이션의 사용이 늘어남에 따라 그 연산의 복잡도가 증가하고 있다. 그로 인해 부동소수점에 대한 중요성이 증가하고 있다. 부동 소수점 연산은 정수 연산에 비해 느리므로 넓은 범위의 수치를 다루는 경우가 아니라면 고정소수점을 사용하는 것이 효율적이다. 이를 통하여 면적 최소화, 처리속도 향상을 꾀할 수 있다.

본 논문에서는 부동소수점간 변환, 부동소수점과 부호 있는 정수 변환뿐만 아니라 고정소수점 부동소수점 변환을 지원하는 컨버터를 제안하고 설계하였다. 또한 하나의 컨버팅 유닛을 통하여 여러 변환을 수행하기 위하여 내부 형태를 정의하였다.

백만개 이상의 테스트 벡터에서 에러가 발생하지 않았으며 TSMC 180nm 공정에서 합성한 결과 면적은 12,878.50 게이트(2-입력 NAND 게이트 기준)이고 최대 동작 주파수는 411MHz이다.

참 고 문 헌

[1] K. Hyun-pil, et al., “The Design of a Structure of Network Co-processor for SDR(Software Defined Radio),” The Journal of The Korean Institute of Communication Sciences, vol. 32, pp. 188-194, 2007.

[2] Z. Linsheng, et al., “Floating-point to Fixed-point Transformation Using Extreme Value Theory,” in Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on, 2009, pp. 271-276.

[3] “IEEE Standard for Binary Floating-Point Arithmetic,” ANSI/IEEE Std 754-1985, p. 0_1, 1985.

[4] “IEEE Standard for Floating-Point Arithmetic,” IEEE Std 754-2008, pp. 1-58, 2008.

[5] K. Ki-Il, et al., “A floating-point to integer C converter with shift reduction for fixed-point digital signal processors,” in Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on, 1999, pp. 2163-2166 vol.4.

[6] K. Ki-Il, et al., “AUTOSCALER for C: an optimizing floating-point to integer C program converter for fixed-point digital signal processors,” Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 47, pp. 840-848, 2000.

[7] L. Saldanha and R. Lysecky, “Float-to-fixed and fixed-to-float hardware converters for rapid hardware/software partitioning of floating point software applications to static and dynamic fixed point coprocessors,” Design Automation for Embedded Systems, vol. 13, pp. 139-157, 2009.

[8] H. Zheng-wei, et al., “Pipeline Design of Transformation between Floating Point Numbers Based on IEEE754 Standard and 32-bit Integer Numbers,” in Intelligent Information Technology and Security Informatics, 2009. IITSI '09. Second International Symposium on, 2009, pp. 92-96.

[9] H. Hu, et al., “Multiply-add fused float point unit with on-fly denormalized number processing,” in Circuits and Systems, 2005. 48th Midwest Symposium on, 2005, pp. 1466-1468 Vol. 2.

[10] V. G. Oklobdzija, “An algorithmic and novel design of a leading zero detector circuit: comparison with logic synthesis,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 2, pp. 124-128, 1994.

[11] John Hauser, SoftFloat, http://www.jhauser.us/arithmetic/SoftFloat.html

[12] DesignWare Building Block IP Documentation Overview, Synopsys, Inc., Mountain View, CA, Feb. 2009.

저 자 소 개

박 상 수(학생회원)

2008년 충남대학교 전기정보통신 공학부 학사 졸업.

2010년 3월～현재 연세대학교 전기전자공학과 석사과정.

<주관심분야 : 마이크로 프로세서, 네트워크 프로세서, 고성능 연산기 설계, SOC>

이 용 석(평생회원)

1973년 연세대학교 전기공학과 학사 졸업.

1977년 University of Michigan, Ann Arbor 석사 졸업.

1981년 University of Michigan, Ann Arbor 박사 졸업.

1993년～현재 연세대학교 전기전자공학과 정교수.

<주관심분야 : 마이크로 프로세서, 네트워크 프로세서, 고성능 연산기 설계, SOC>

김 현 필(학생회원)

2005년 연세대학교 전기전자 공학과 학사 졸업.

2005년 3월～현재 연세대학교 전기전자공학과 석박사 통합과정.

<주관심분야 : 네트워크 프로세서, SVC, 컴퓨터 아키텍쳐, 멀티미디어 프로세서>