論文2000-37TE-5-2

고속 지수 선택기를 이용한 여분 부동 소수점 이진수의 제산/스퀘어-루트 설계 및 구현

(A Design and Implementation of the Division/Square-Root for a Redundant Floating Point Binary Number using High-Speed Quotient Selector)

金 鍾 燮 *, 趙 相 福 **
(Jong-Seop Kim and Sang-Bok Cho)

요 약

본 논문은 고속 지수 선택기를 이용한 여분 부동 소수점 이진수의 제산/스퀘어-루트 설계 및 구현에 관하여 기술하였다. 본 제산/스퀘어-루트는 처리 속도 25MHz를 갖는 여분 이진수의 가산 방식을 사용하여 올림수 지연을 제거함으로써 비트 크기에 관계없이 일정한 시간으로 가산을 수행한다. 각각의 반복 단계에 널리 사용된 제산과 스퀘어-루트에 대해 16-비트 VLSI 회로를 설계하였다. 이것은 매번 16개 클럭마다 시프트된 이진수를 여분 가산하여 제산 및 스퀘어-루트를 실행한다. 또한 이 회로는 비복원 방법을 사용하여 지수 비트를 얻는다. 지수 선택 논리의 간단한 회로를 구현하기 위하여 나머지 비트의 주요 세 자리를 사용하였다. 결과적으로, 이 회로의 성능은 새로운 지수 선택 가산 논리를 적용하여 지수 결정 영역을 병렬 처리함으로써 한층 더 연산 처리 속도를 높인 것이다. 이전에 동일한 알고리즘을 사용하여 제안된 설계보다 13% 빠른 속도 증가를 보였다.

Abstract

This paper described a design and implementation of the division/square-root for a redundant floating point binary number using high-speed quotient selector. This division/square-root used the method of a redundant binary addition with 25MHz clock speed. The addition of two numbers can be performed in a constant time independent of the word length since carry propagation can be eliminated. We have developed a 16-bit VLSI circuit for division and square-root operations used extensively in each iterative step. It performed the division and square-root by a redundant binary addition to the shifted binary number every 16 cycles. Also the circuit uses the nonrestoring method to obtain a quotient. The quotient selection logic used a leading three digits of partial remainders in order to be implemented in a simple circuit. As a result, the performance of the proposed scheme is further enhanced in the speed of operation process by applying new quotient selection addition logic which can be parallelly process the quotient decision field. It showed the speed-up of 13% faster than previously presented schemes used the same algorithms.

* 正會員, 徐羅伐大學 電氣電子電算學部
(School of Electricity, Electronics & Computer Science, Sorabol College)
** 從信會員, 蔚山大學校 電氣電子 및 自動化 工學部
(School of Electrical and Automation Engineering, Ulsan University)
※ 본 연구는 IC Design Education Center 지원에 의해 수행되었음.
接受日字:2000年 10月16日, 수정완료일 : 2000年11月24日

Ⅰ. 서 론

컴퓨터 시스템에서 제산(division)과 스퀘어-루트(square-root)는 매우 중요한 산술 연산이다. 이 연산은 디지털 신호 처리(DSP)뿐만 아니라 부동 소수점(floating point)에 널리 사용한다. 본 논문은 제산과 스퀘어-루트에 대한 산술 계산을 위해 동일한 형태의

알고리즘을 사용하여 공유된 조합 논리 회로를 설계하였다. 이 회로는 여분 가산 배열(redundant addition array)과 지수 선택 논리(quotient selection logic), 그리고 진행 변환(on-the-fly conversion)의 세 가지 주요한 요소로 구성된다. 여분 가산 배열은 일반적인 여분 가산 셀을 배열하여 피연산자와 지수 값을 병렬 가산 처리하여 나머지를 얻는다. 그리고 지수 선택 논리는 지수 숫자를 선택하여 여분 가산 배열과 진행 변환으로 의하여 지수를 결정한다.

부동소수점에 대한 고속 제산 알고리즘은 여분 이진수 체계를 사용한다. 제산 알고리즘을 기초로 한 스퀘어-루트 알고리즘은 여분 이진수 중에 가장 탁월한 성능을 갖는다. 이와 같은 두 개의 알고리즘은 비복원(non-restoring) 방법을 사용하여 정상적인 피연산자를 지수 숫자와 나머지 부분의 여분 표현으로 만들고, 또한 나머지 부분의 최상위 숫자의 정수에 의해 지수 숫자 선택을 수행한다.

이 회로의 주요 요소 중에 하나인 지수 선택 논리는 피연산자의 개수에 독립적인 모든 반복 연산에서 나머지 부분의 몇 개의 최상위 비트를 사용하여 지수를 선택한다. 만약 지수 선택에 사용되는 나머지 부분의 최상위 비트의 값을 x라 하면, 지수 선택은 x-숫자 지수를 선택하여 지시한다. 본 논문에서는 지수 선택에 필요한 나머지 부분의 최상위 비트에 대해 3개의 부호화된 형태를 사용하였으며, 이 것은 제산/스퀘어-루트의 지수 최대 값 x를 줄임으로써 연산 반복 시간을 단축하는데 중요한 역할을 한다. 이러한 관점에서, 제안된 설계는 공유한 제산과 스퀘어-루트에 대하여 $x=2$(Radix-2)[6]을 갖는다. 지수 숫자는 최대 여분 숫자인 {−1, 0, +1}에 속한 제안된 알고리즘에 의해 만들어진다. 또한, 만들어진 지수 숫자는 진행 변환 알고리즘을 사용하여 이진수 형태로 변환된다.

본 논문은 다음과 같이 세 개의 절로 구성하였다. 1장은 서론 부분이고, 2장 본론에서는 7개의 절로 나누어지는데 1절은 제산 알고리즘에 대하여, 2절은 스퀘어-루트 알고리즘에 대해 기술한다. 3절은 진행 여분 이진수에 대한 이진수 변환과 4절은 여분 이진수 표현의 가산을 논하였다. 그리고 5절은 지수 선택 논리 회로의 설계에 대하여, 6절은 제산과 스퀘어-루트 R_j+1 연산의 선택에 관하여 제시하였다. 그리고 7절은 전체적인 제산과 스퀘어-루트의 구조를 나타내었다. 마지막으로, 3장 결론 부분에는 제산/스퀘어-루트의 최종 구성과 시뮬레이션 결과 및 다른 설계에 대한 연산 처리 속도를 비교하였다.

Ⅱ. 본 론

1. 제산 알고리즘

제산 알고리즘[4]은 나머지 부분과 제수의 값을 판단하여 비복원 제산 방법과 4개의 지수를 기초로 한다. 비복원 제산은 시프트, 감산, 그리고 비교 동작에 의해 실현된 전통적인 제산 방법으로 널리 사용되는 알고리즘이다. 4개의 지수 숫자 q_j는 {−1, −0, +0, +1}을 갖는다. 지수 선택 숫자 q_j를 기초로 한 제산의 연산은 (1)식과 같은 관계식을 만족한다.

$$R_{j+1}=rR_j-q_jD,\ j=0,1,2,\cdots,n-1 \qquad (1)$$

여기서,

$R_j=j$번째 지수 숫자의 선택 후 나머지 부분
$R_0=$피제수(강제조건 $|R_0|<|D|$)
r =기수(이진수=2)
$q_j\in\{-1, 0, 1\}$, j번째 이진 지수 부호 숫자(BSD)
$D=$제수

이고, j가 0일 때, n−비트 피제수와 제수를 갖는 지수 q_0과 나머지 R_1을 구하면 다음과 같은 초기치를 갖는다.

$$q_o=1$$
$$R_1=R_0-D$$

그리고, 피제수 R_0은 구하면 다음과 같다.

$$R_0(\text{피제수})=R^{t}_{0}R^{0}_{0}.R^{1}_{0}R^{2}_{0}\cdots R^{n-1}_{0}$$
여기서 $R^{t}_{0}=0$

비복원 제산[8]에서 다음 관계식에 의해 $n-1$번을 반복적으로 지수 선택하여 나머지 부분에 대한 결과를 계산한다. 여기서 지수 q_j는 시프트된 j번째의 나머지 부분 R_j 여분 이진 형태에 대한 세 개 주요 숫자인 $(R^{t}_{j}R^{0}_{j}.R^{1}_{j})_{BCD}$의 크기 범위에 따라 네 가지 값을 갖는다.

$$q_j=\begin{cases}1\\0\\-0\\-1\end{cases},\quad R_{j+1}=\begin{cases}2(R_j-D)\\2(R_j+0_{\bar{2}})\\2(R_j+0)\\2(R_j+D)\end{cases}$$

$$\text{if}\begin{cases}1\le(R_j^tR_j^0.R_j^1)_{BSD}\\0\le(R_j^tR_j^0.R_j^1)_{BSD}<1\\-1<(R_j^tR_j^0.R_j^1)_{BSD}<1\\(R_j^tR_j^0.R_j^1)_{BSD}\le-1\end{cases}$$

여기서

$1\le j\le n-1$
$0\equiv 00.00\cdots0$
$0_{\bar{2}}\equiv 11.11\cdots1$ 초기 올림수 입력 $=1$(2의 보수)

이고, 그리고 나머지 부분의 연산 속도를 높이기 위하여, R_j+1 나머지 부분에 대한 갱신을 이진 부호 숫자를 갖는 여분 이진수로 표현한다. 반복 형태의 연산 방식은 이진수에 대한 여분 이진수의 가산이다. 여분 R_j+1의 오버플로우(overflow) 방지 표현으로 0에 대한 두 개의 역이 있다. 이것은 조건 표준 오차보다도 더 작은 나머지 값, 또는 주어진 반복의 수 동안 연산을 계속한다. 지수의 j번째 최상위 비트는 만약 R_j의 부호가 제수의 부호와 일치하면 1이고, 그리고 만약 일치하지 않는 부호이면 0이다.

2. 스퀘어-루트 알고리즘

스퀘어-루트 알고리즘은 제산과 스퀘어-루트 반복 연산에 대한 지수 숫자 $q_j=\{-1, -0, 0, 1\}$을 동일하게 사용한다. 여기서 나머지 부분이 복원되지 않는 비복원 알고리즘으로 두 개의 비트를 한 개의 비트로 전환한다. 여기서 음(−)일 때도 다음 반복에 사용되어진 각각의 반복에서 나머지 부분이 발생한다. 이전의 나머지 부분이 복원되지 않는 이와 같은 알고리즘을 비복원 스퀘어-루트 알고리즘[5]이라 한다. 만약 마지막 반복에서 나머지 부분이 음(−)이 아니면, 최종 나머지가 된다. 만약 그렇지 않으면, 나머지 부분을 가산하여 정확한 최종 나머지를 얻을 수 있다.

여기서, j번째의 나머지 R_j의 유도식 (2)를 쓰면

$$R_j=2^{-1}R_{j+1}+q_j\left[\sum_{k=0}^{j-1}q_k2^{-k}+q_j2^{-j-2}\right] \tag{2}$$

이고, j번째의 스퀘어-루트 여분 이진 Q_j는 다음과 같은 식 (3)으로 나타낼 수 있다.

$$Q_j=Q_{j-1}+2^{-j}\ q_j=q_0.q_1q_2\cdots q_{j-1} \tag{3}$$

제산의 이진수에 대한 여분 이진수 변환(RB-B) 가산기를 스퀘어-루트 연산에 사용하기 위해 이진 형태 Q_j+2^{-j-2} 또는 Q_j-2^{-j-2}를 갖는다. 이것은 여분 이진 Q_j로부터 그것의 이진 등가를 나타내는 것으로 진행 변환(on-the-fly conversion)하기 때문에 형태 변환을 할 수 있다. 이것을 증명하면, 나머지 R_j의 유도식으로부터 나머지 R_1의 (4)식과 R_2의 (5)식을 다음과 같이 나타낼 수 있다.

$$R_1=2^{-1}R_2+q_1\left[\sum_{k=0}^{0}q_k2^{-k}+q_1\,2^{-3}\right] \tag{4}$$

$$R_2=2^{-1}R_3+q_2\left[\sum_{k=0}^{0}q_k2^{-k}+q_2\,2^{-4}\right] \tag{5}$$

위의 식 (4)에 식 (5)을 대입하여 R_1을 구하면 다음과 같이 계산할 수 있다.

$$\begin{aligned}R_1&=2^{-1}\left\{2^{-1}R_3+q_2\left[\sum_{k=0}^{1}q_k2^{-k}+q_2\,2^{-4}\right]\right.\\&\quad\left.+q_1\left[\sum_{k=0}^{0}q_k2^{-k}+q_1\,2^{-3}\right]\right\}\\&=2^{-2}R_3+2^{-1}q_2Q_1+q_1Q_0+2^{-4}q_2^2+2^{-2}q_1^2\\&=2\sum_{j=1}^{n-1}2^{-j}\,q_jQ_{j-1}+2^{-2j}q_j^2\\&=(Q_1^2-Q_0^2)+(Q_2^2-Q_1^2)\cdots+(Q_{n-1}^2-Q_{n-2}^2)\\&=R_0-1\quad(Q.E.D.)\end{aligned}$$

스퀘어-루트 초기치 Q_0는 다음과 같고 이 때 지수 초기치 q_0는 1이다.

$q_o=1,\ Q_0=01,\ R_1=R_0-1,$

여기서 $R_0=$ 스퀘어−루트의 연산자

스퀘어-루트 반복 단계에서도 제산 방법과 동일하게 시프트된 j번째의 나머지 부분 R_j 여분 이진 형태에 대한 세 개 주요 숫자인 $(R_j^tR_j^0.R_j^1)_{BCD}$의 크기 범위에 따라 지수 q_j의 네 가지 값과 스퀘어-루트 여분 이진 형태의 변화 값에 의해 나머지 갱신 부분 R_{j+1}의 연산 값이 결정된다.

$$q_j=\begin{cases}1\\0\\-0\\-1\end{cases},\quad R_{j+1}=\begin{cases}2(R_j-(Q_j+2^{-j-2}))\\2(R_j+0_{\bar{2}})\\2(R_j+0)\\2(R_j+(Q_j-2^{-j-2}))\end{cases}$$

$$\text{if}\begin{cases} 1\le(R_j^{\,t}R_j^{\,0}.R_j^{\,1})_{BSD} \\ 0\le(R_j^{\,t}R_j^{\,0}.R_j^{\,1})_{BSD}<1 \\ -1<(R_j^{\,t}R_j^{\,0}.R_j^{\,1})_{BSD}<0 \\ (R_j^{\,t}R_j^{\,0}.R_j^{\,1})_{BSD}\le -1 \end{cases}$$

결국, 여기서 사용된 스퀘어-루트 알고리즘은 음(−)일 때 조차도 다음 반복 연산에 사용되어 각각의 나머지 부분을 만든다. 그래서 나머지는 멀티플렉서를 통하여 복원 알고리즘을 실행한 이전의 나머지를 사용하기 때문에 복원되지 않는다. 이와 같은 점에서, 비복원 스퀘어-루트 알고리즘은 비복원 제산 알고리즘과 매우 동일한 형태[3]를 갖는다.

3. 진행 여분 이진수에 대한 이진수 변환

진행 변환 알고리즘은 규정 범위의 보수 표현에서 여분 이진수 부호로부터 이진수 부호를 변환한다. 여분 이진수의 루트 지수 부분은 각각 반복된 단계에서 이진 등가[1]로 변환한다. 진행 변환은 모든 루트 지수에 대해 반복 연산의 마지막을 얻은 후 변환 지연이 최소화된다. 다시 말해서, 스퀘어-루트 연산의 반복 모형[3]은 여분 이진과 이진 가산의 가수로서 이진 루트 부분이 필요하다.

여분 숫자 벡터 ℚ는 정상적인 분수 q일 때 다음과 같은 (6)식을 갖는다.

$$q=\sum_{k=0}^{i-1} q_k 2^{-k}, \quad q_k\in\{-1,\ 0,\ 1\} \tag{6}$$

만약 $q_j=-1$ 일 때, 올림수 지연에 대한 알고리즘이 필요하다. 이 올림수의 지연은 이전의 비트에 도달할 때까지 q가 1을 갖는다. 예를 들면,

$$q_i, q_{i+1}, \cdots, q_{j-2}, q_{j-1}=1,0,\cdots,0,0$$

여기서 -2^{-2k}를 더하면 다음과 같은 값을 얻는다.

$$q_i, q_{i+1}, \cdots, q_{j-2}, q_{j-1}, q_j=0,1,\cdots,1,1,1$$

다음과 같이 상응하는 두 개의 조건 형태에 의해 올림수 지연을 피할 수 있다. 그 하나는 $q_j=1$ 또는 $q_j=0$일 때 Q_j^{-1}의 형태를 선택하고, 다른 하나는 $q_j=-1$일 때 Q_{j-1}^{-}의 형태를 선택한다.

이것은 나타내면,

$$Q_j=\begin{cases} Q_{j-1}+2^{-j} \\ Q_{j-1} \\ Q_{j-1}^{-}+2^{-j} \end{cases}, \quad \text{if } q_j=\begin{cases} +1 \\ 0 \\ -1 \end{cases}$$

이고, 식 (7)과 식 (8)같은 필요한 결과를 얻을 수 있다.

$$Q_{j-1}=-q_0+\sum_{i=0}^{j-1} q_i 2^{-i}=q_{j-1} \tag{7}$$

$$Q_{j-1}^{-}=Q_{j-1}-2^{j-i} \tag{8}$$

따라서 변환 지수의 최종 결과는 다음과 같다.

$$q_c=Q[m]$$

Q_j와 Q_j^-의 형태로 다시 계산하여 놓으면, 여기서 q가 정상적일 때, 그 초기 상태는 다음과 같은 형태를 갖는다.

$$Q_1=\begin{cases} 0.1 \\ 0.0 \\ 1.1 \end{cases}, \quad Q_1^{-}=\begin{cases} 0.0 \\ 1.1 \\ 1.0 \end{cases}, \quad \text{if } q_1=\begin{cases} +1 \\ 0 \\ -1 \end{cases}$$

변환 알고리즘은 제산 반복 단계마다 이진 지수 부분 $Q_j=Q_j+2^{-j}$와 음수의 이진 지수 부분 $Q_j^-=Q_j-2^{-j}$를 찾는다. 그 변환 지수 쌍(Q_j+1과 Q_{j+1}^-)은 다음과 같이 q_j+1 지수 숫자의 값을 바탕으로 갱신된다.

$$Q_{j+1}=\begin{cases} Q_j+2^{-(j+1)} \\ Q_j \\ Q_j^{-}+2^{-(j+1)} \end{cases}, \quad \text{if } q_{j+1}=\begin{cases} +1 \\ 0 \\ -1 \end{cases}$$

$$Q_{j+1}^{-}=\begin{cases} Q_j \\ Q_j^{-}+2^{-(j+1)} \\ Q_j^{-} \end{cases}, \quad \text{if } q_{j+1}=\begin{cases} +1 \\ 0 \\ -1 \end{cases}$$

결과적으로, 갱신된 값은 변환의 조건을 만족한다. 예를 들면, $q=0.1\bar{1}001\bar{1}10\bar{1}1$에 대한 변환을 나타내면 표 1과 같이 구할 수 있다. 여기서 $\bar{1}=-1$이라 가정하자.

따라서, 지수 $q=0.1\bar{1}001\bar{1}10\bar{1}1$에 대한 변환 지수 $q_c=0.0100011111$로 변환된다. 이와 같은 진행 변환 알고리즘 구현을 위해 Q_j와 Q_j^- 지수를 저장하는 두 개의 레지스터가 필요하다. 각각 두 개의 레지스터는 q_j의 값에 의해 마지막 비트에 대한 삽입과 그 비트와 함께 왼쪽으로 한 비트씩 시프트 한다. 또한, 이것

은 Q_j와 Q_j^-를 교환할 수 있는 병렬 로드가 필요하다.

표 1. 여분 이진수의 변환
Table 1. Conversion of Redundant Binary Number.

j	q_j	Q_j	Q_j^-	q_{cj}
0	0			
1	1	0.1	0.0	0.1
2	-1	0.01	0.00	0.01
3	0	0.010	0.001	0.010
4	0	0.0100	0.0011	0.0100
5	1	0.01001	0.00110	0.01001
6	-1	0.010001	0.001100	0.010001
7	1	0.0100011	0.0011000	0.0100011
8	0	0.01000110	0.00110001	0.01000110
9	-1	0.010001111	0.001100010	0.010001111
10	1	0.0100011111	0.0011000100	0.0100011111

4. 여분 이진수 표현의 가산

두 개의 이진수 가산은 올림수의 지연 때문에 연산자의 크기에 비례하여 연산 처리 시간이 정해진다. 그러나 여분 이진수 방식에서는 올림수 지연이 제거되어 두 개의 연산자의 크기에 독립적으로 일정한 시간으로 가산을 수행한다. 여분 이진수 표현은 제산 및 스퀘어-루트를 위해 피가수 변수 숫자가 {−1, 0, 1}을 갖는다. 두 개의 n−비트 여분 이진수의 가산은 여분 사용에 대한 일정한 시간을 형성한다. 이 가산은 다음과 같이 두 단계로 실행된다.

첫 번째 단계는 다음 (9)식과 같이 피가수와 가수의 두 개의 숫자를 더한다.

$$a_i + b_i = 2c_i + s_i \tag{9}$$

여기서

$a_i \in \{-1, 0, 1\}$, 이진 부호 변수(BCD)
$b_i \in \{0, 1\}$, 가수 변수
$c_i \in \{0, 1\}$, 중간 숫자 올림 변수
$s_i \in \{-1, 0\}$, 중간 합 변수

이고, 두 번째 단계는 다음 하위 순서 위치로부터 si와 ci를 더하여 각 위치에 대한 합 z_i를 (10)식과 같이 얻을 수 있다. 첫 번째 단계의 가산 후에 두 번째 단계의 어떤 위치에서도 자리 올림수는 발생하지 않는다.

$$s_i + c_{i+1} = z_i \tag{10}$$

한 개의 여분 이진수와 이진수(RB-B) 가산기[2]에 대한 논리는 00 = 0, 01=1, 10 =−1, 11= 0이고, 이진 부호 숫자에 대한 두 개의 부호 비트는 $a_i \equiv a_i^l a_i^r$, $z_i \equiv z_i^l z_i^r$ 놓고 진리표를 나타내면 표 2와 같이 된다.

표 2. 여분 이진수 가산의 진리표
Table 2. Truth Table of Redundant Binary Addition.

a_i^l	a_i^r	b_i	c_{i+1}	z_i^l	z_i^r
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	1

표 2로부터 여분 이진 표현의 가산 논리식은 아래와 같이 유도된다.

$$c_i = \overline{a_i^l} a_i^r + \overline{a_i^l} b_i + a_i^r b_i$$

$$z_i^l = a_i^l \oplus a_i^r \oplus b_i$$

$$z_i^r = c_{i+1} = \overline{a_{i+1}^l} a_{i+1} + \overline{a_{i+1}^l} b_{i+1} + a_{i+1}^r b_{i+1}$$

5. 지수 선택 논리 회로

지수 숫자는 각 단계에서 $(R_j^t R_j^0. R_j^1)_{BCD}$를 바탕을 두고 있다. 네 개의 가능한 지수 값 $q_j = \{-1, -0, 0, 1\}$을 선택한다. 그래서 $(R_j^t R_j^0. R_j^1)_{BCD}$는 a(1), b(0), c(−0), d(−1)의 네 개의 경우로 디코드 된다. 이것은 표 3에 의해서 다음과 일치하는 논리 값 S_2, S_1, S_0을 갖는다.

제산과 스퀘어-루트의 지수 선택 논리는 $(R_j^t R_j^0. R_j^1)_{BCD}$ 값을 입력으로 하여 그 합을 그림 1의 조합 논리 회로의 가산기에서 구한다. 여분 이진 표현의 주요 숫자에 대한 합은 디코더에 입력되어 네 개의 가능한

지수 선택 값 a, b, c, d가 디코드 된다. 이와 같은 지수 선택 논리에 대한 구성을 그림 1에 나타내었다.

표 3. 지수 선택 q_j의 진리표
Table 3. Truth tabl q_je for Quotient Selection of q_j.

S_2	S_1	S_0	q_j	value
0	0	0	b	0
0	0	1	a	1
0	1	0	a	1
0	1	1	a	1
1	0	0	c	-0
1	0	1	d	-1
1	1	0	d	-1
1	1	1	d	-1

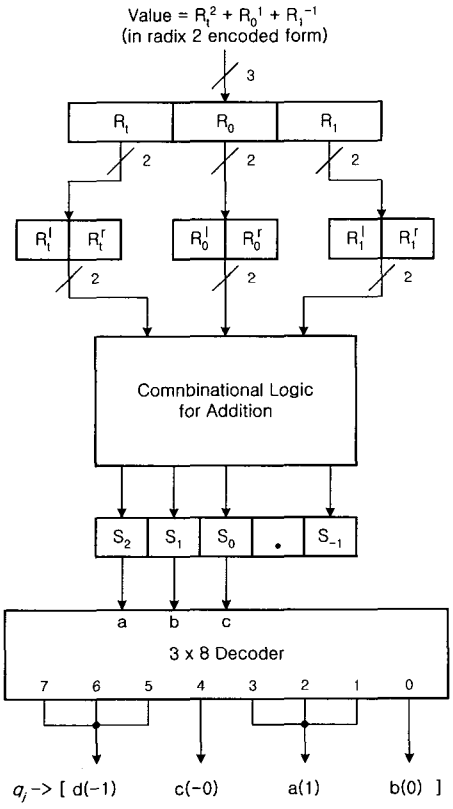


그림 1. 지수 선택 구성도
Fig. 1. Block Diagram of Quotient Selection.

조합 논리에 대한 가산 회로의 S_2, S_1, S_0을 구하기 위해, 먼저 $S_i=(R_t)^2+(R_0)^1$의 중간 가산 논리식 값을 진리표를 나타내면 표 4와 같다.

표 4에 의해서 S_i 가산에 대한 논리식 S_{i2}, S_{i1}, S_{i0}는 다음과 같이 유도된다.

$$S_{i2}=R_t^l R_t^r+\overline{R_t^r}\,R_0^l R_0^r$$
$$S_{i1}=R_t^l R_t^r R_0^l+\overline{R_t^l}\,R_t^r\,\overline{R_0^l}+R_t^r\,\overline{R_0^r}$$
$$S_{i0}=R_o^r$$

표 4. $S_i=(R_t)^2+(R_0)^1$의 진리표
Table 4. Truth table for $S_i=(R_t)^2+(R_0)^1$.

R_t^l	R_t^r	R_0^l	R_0^r	S_{i2}	S_{i1}	S_{i0}
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	0
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	1
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	0	1
1	1	1	0	1	1	0
1	1	1	1	1	1	1

따라서, S_{i2}, S_{i1}, S_{i0} 논리식을 이용하여 나머지 부분 R_j 여분 이진수의 $(R_j^t R_j^0 . R_j^1)_{BCD}$의 숫자에 대한 최종 가산 S_n의 조합 논리에 대한 S_2, S_1, S_0의 진리표 나타내면 표 5와 같이 표현된다.

표 5에 의해 S_n 가산에 대한 논리식 S_2, S_1, S_0은 다음과 같이 유도된다. 여기서 S_{-1}의 논리식은 지수 선택에 포함되지 않는다. 왜냐하면, S_2, S_1, S_0의 이진 값들만 사용하여도 지수 선택에 충분하기 때문이다.

$$S_2=S_{i2}S_{i1}+S_{i2}S_{i0}+\overline{S_{i2}}\,\overline{S_{i1}}\,\overline{S_{i0}}\;R_1^l R_1^r$$
$$S_1=S_{i1}$$
$$S_0=S_{i0}(S_{i1}+\overline{R_1^r}+\overline{S_{i2}\oplus R_1^l})$$

위에서 언급된 표 3에서 디코더의 출력 q_j의 지수 선택에서 알 수 있듯이 S_n 가산에 대한 논리식 S_2, S_1, S_0의 이진 값에 의해 지수 선택 논리는 $(R_j^t R_j^0 . R_j^1)_{BCD}$ 값을 네 개의 가능한 지수 값 a, b, c, d를 3×8 디코더를 사용하여 그림 1과 같이 선택할 수 있다.

표 5. $S_n=(R_l)^2+(R_0)^1+(R_1)^{-1}$의 진리표

Table 5. Truth table for $S_n=(R_l)^2+(R_0)^1+(R_1)^{-1}$.

S_{i2}	S_{i1}	S_{i0}	R_l^l	R_l^r	S_2	S_1	S_0	S_{-1}
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0
								1
0	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	1	1
0	0	1	1	0	0	0	1	0
								1
0	1	0	0	0	0	1	0	0
0	1	0	0	1	0	1	0	1
0	1	0	1	0	0	1	0	0
0	1	0	1	1	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	0	1	0	1	1	1
0	1	1	1	0	0	1	1	0
0	1	1	1	1	0	1	1	1
1	0	1	0	0	1	0	1	0
								1
1	0	1	1	0	1	0	1	0
1	0	1	1	1	1	0	1	1
1	1	0	0	0	1	1	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	0	0
1	1	0	1	1	1	1	0	1
1	1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1
					value S_n			

6. 제산 및 스퀘어-루트 R_{j+1} 연산의 선택

지수 선택 q_j를 사용하여 제수의 여분 이진수를 선택한다. 이 것은 모든 제수에 대하여 4×1 멀티플렉서를 연결하고, 제어 입력에 지수 선택 q_j를 입력하면 지수 값에 따라 제수의 여분 이진 값이 선택된다. 표 6은 여분 제산 R_{j+1}의 선택에 대한 네 가지 연산을 나타내었다.

위와 같은 방법으로 스퀘어-루트 R_{j+1}의 네 가지 연산을 수행하기 위해 지수 선택 q_j를 사용하여 스퀘어-루트의 여분 이진수 Q_j를 선택한다. 선택된 지수의 연산은 표 7과 같다.

표 6. 여분 이진수 제산 R_{j+1} 선택에 대한 연산

Table 6. Operation for Redundant Binary Division R_{j+1} Selection.

Quotient selection a	b	c	d	*R_{j+1} selection* A	B	*Operation*	*Selected D*
1	0	0	0	0	0	$2(R_j-D)$	D
0	1	0	0	0	1	$2(R_j+0_s)$	1
0	0	1	0	1	0	$2(R_j+0)$	0
0	0	0	1	1	1	$2(R_j+D)$	-D

표 7. 여분 이진수 스퀘어-루트 R_{j+1} 선택에 대한 연산

Table 7. Operation for Redundant Binary Square-Root R_{j+1} Selection.

Quotient select. a	b	c	d	*R_{j+1}select.* A	B	*Operation*	*Selected Q_j*
1	0	0	0	0	0	$2(R_j-(Q_j+2^{-j-2}))$	(Q_j+2^{-j-2})
0	1	0	0	0	1	$2(R_j+0_s)$	1
0	0	1	0	1	0	$2(R_j+0)$	0
0	0	0	1	1	1	$2(R_j+(Q_j-2^{-j-2}))$	$-(Q_j-2^{-j-2})$

제산 및 스퀘어-루트 R_{j+1} 연산의 선택[7] 회로를 나타내면 그림 2와 같다. 이 회로는 n-개의 제수와 스퀘어-루트 진행 변환에 대한 각각의 출력에 연결되어 지수 선택 a, b, c, d 값의 조합에 의하여 여분 이진수 D 또는 Q_j를 선택한다. 선택된 여분 이진수는 여분 이진 가산 셀로 보내어 가산을 처리한다.

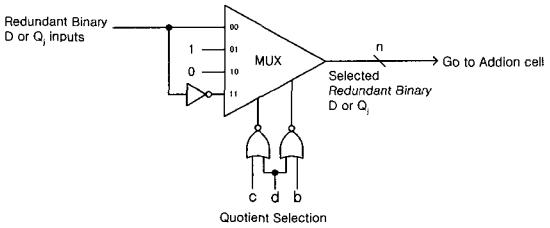


그림 2. 제산/스퀘어-루트 R_{j+1} 연산의 선택 회로

Fig. 2. Selection Circuit of Division/Square-Root R_{j+1} Operation.

7. 제산/스퀘어-루트의 전체 구조

위와 같이 기술된 제산과 스퀘어-루트 알고리즘은 그림 3과 같이 규격화된 셀로 구현할 수 있다. 이 회로는 여분 가산 배열과 지수 선택 논리 그리고 진행 변환의 세 가지 중요한 요소로 구성된다.

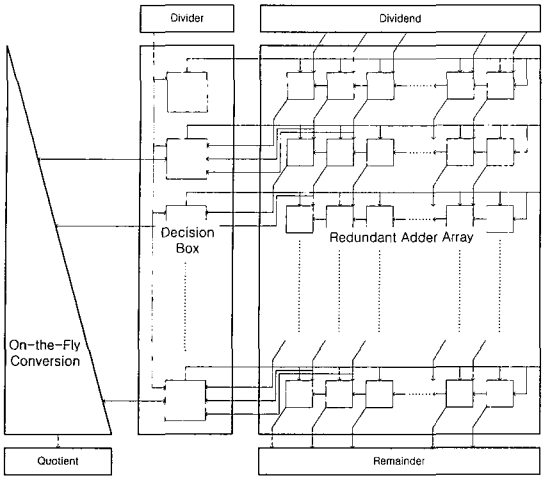


그림 3. 제산/스퀘어-루트의 전체 구조
Fig. 3. Overall Architecture of Division/Square-Root.

여분 가산 배열은 피제수를 입력으로 하는 가산 셀들로 구성되고, 그 결과가 나머지 값으로 출력된다. 그리고 진행 변환 회로는 이진수를 여분 이진수로 변환하여 지수를 결정하고, 지수 값을 지수 선택 논리에 보낸다. 마지막으로 지수 선택 논리는 이진 제수 값과 진행 여분 이진 스퀘어-루트 값을 입력으로 받아들인다. 그래서 R_{j+1} 연산을 처리하기 위해 여분 이진수 D 또는 Q_j를 선택하고, 그 값을 여분 가산 배열로 보낸다. 또한 시프트 된 R_j 여분 이진 형태의 $(R_j^t \;\; R_j^0 . R_j^1)_{BCD}$ 숫자를 지수 선택 논리 회로에 입력시켜, 그 가산 값을 진행 변환으로 보내어 지수를 결정한다.

Ⅲ. 결과 및 평가

본 논문에서 제안한 지수 선택 가산 회로를 VLSI 회로를 사용하여 설계하고, 0.6㎛ 공정, 전원 전압 5V, SPICE 모델에 대한 시뮬레이션 결과를 그림 4에 나타내었다. 이 그림에서 알 수 있듯이 지수 선택 가산 회로의 전달지연시간은 2ns를 갖는다. 시뮬레이션 결과에서 신호❶은 지수 선택 가산 회로에 대한 여분 이진 형태의 세 개의 주요 숫자 중 첫 번째 왼쪽 숫자인 R_t^l 입력이고, 신호❷는 지수 선택에 대한 가산 값 S_2의 출력이다.

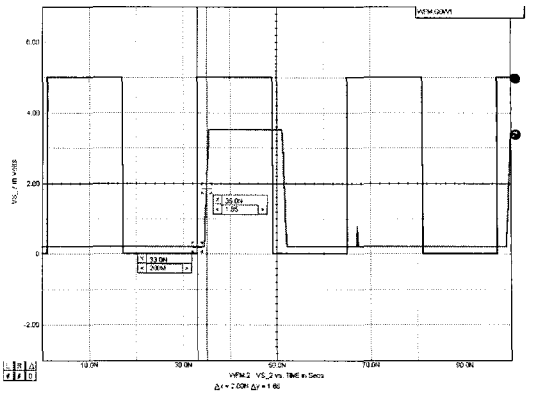


그림 4. 지수 선택 가산 회로의 전달 지연 시간에 대한 시뮬레이션 결과
Fig. 4. SPICE simulation result for Propagation Delay of the Quotient Select Addition Circuit.

그리고 나머지 여분 이진 형태의 세 개의 주요 숫자 $(R_j^t R_j^0 . R_j^1)_{BCD}$는 각각 두 개의 값, 즉 R_t^l, R_t^r, R_0^l, R_0^r, R_1^l, R_1^r의 입력 신호 6개를 갖는다. 그림 5는 이 입력 신호들에 대한 지수 선택 가산 S_2, S_1, S_0,의 출력 신호를 시뮬레이션 하여 나타낸 것이다. 이 세 개의 출력 신호를 디코더로 보내 지수 선택 값 a, b, c, d를 선택한다. 선택된 신호는 지수 선택 회로로 보내

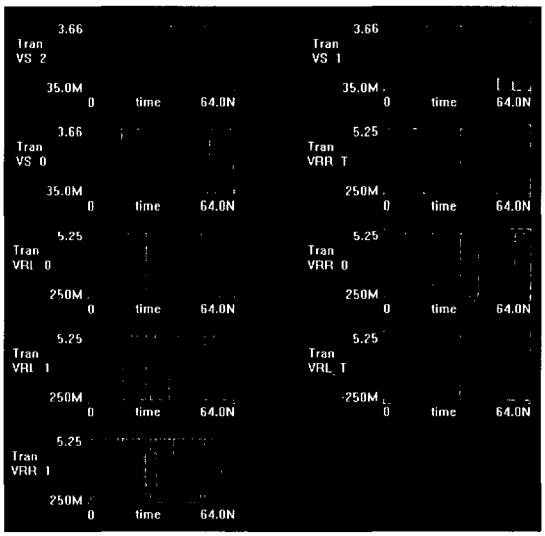


그림 5. 지수 선택 가산 회로의 SPICE 시뮬레이션 결과
Fig. 5. SPICE simulation result of the Quotient Select Addition Circuit.

어 제산 및 스퀘어-루트를 수행한다.

본 논문의 제산/스퀘어-루트는 16-비트 VLSI 회로로 설계하였으며, 총 트랜지스터 수는 약 17,000개이다. 이 회로에 대한 시뮬레이션 성능 결과는 제산의 연산 속도가 41ns이고, 스퀘어-루트의 연산 속도는 43ns를 갖는다. 이 것은 이전에 동일한 제산과 스퀘어-루트 알고리즘을 사용한 설계보다 약 13% 회로 속도가 향상된 것이다. 그 이유는 새로운 지수 선택 회로를 간략화하고 병렬 처리로 고속화하여 얻어진 결과이다. 그림 6은 제산 및 스퀘어-루트 모델[9,10]들의 속도를 비교한 것이다. 비교 모델이 동일하지 않는 비트 크기로 연산 결과를 비교하였지만, 본 논문이 제안한 제산 및 스퀘어-루트 연산은 올림수 지연을 제거함으로써 비트 크기에 상관없이 항상 일정한 지연 시간을 갖는다. 따라서 그림에서 알 수 있듯이 본 논문의 제산 및 스퀘어-루트 연산 속도가 다른 모델에 비해 성능이 우수하다는 것을 알 수 있다.

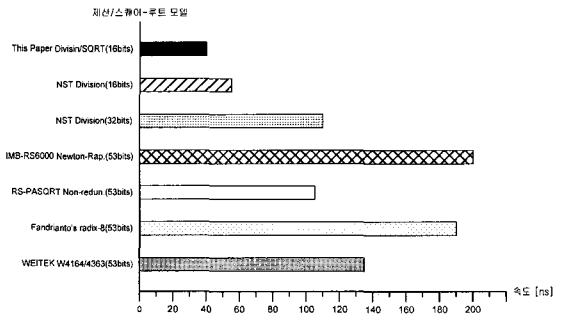


그림 6. 성능 비교
Fig. 6. Performance Comparison.

Ⅳ. 결 론

이 회로에서 구해진 루트 지수 비트는 비복원 방법을 사용한다. 각각의 반복 연산 단계에서의 루트 지수 값은 이진 형태를 유지하고, 이진수 변환으로 진행 여분 이진수(on-the-fly redundant binary)의 결과를 갖는다. 나머지 부분은 여분 이진 표현이다. 반복 연산 모형은 여분 이진과 자리 올림수 가산과 동일한 이진 감산 특징을 갖는 회로로 구현된다. 다시 말해서, 자리 올림수에 상관없는 지연 감산과 그리고 감수의 데이터 크기와 독립적인 속도를 갖는다. 루트 지수 선택 논리는 나머지 부분의 세 개의 주요 숫자를 입력으로 하여, 그림 4에 나타난 시뮬레이션 결과의 전달 지연 시간에서 알 수 있듯이 고속의 지수 선택 논리 회로를 구현하였다. 결과는 규칙적이고, VLSI 구현에 알맞은 모듈 구조를 갖는다.

참 고 문 헌

[1] Milos D. Ercegovac and Tomas Lang, "On-the-Fly Conversion of Redundant into Conventional Representations", *IEEE Trans. on Computers*, vol. C-36, no. 7, pp. 895-897, July 1987.

[2] Y. Harata, Y. Nakamura, H Nagase, M. Takigawa and N. Takagi, "A High-Speed Multiplier Using a Redundant Binary Adder Tree", *IEEE Journal Solid-State Circuits*, vol. SC-22, no. 1, pp. 28-34, February 1987.

[3] Milos D. Ercegovac and Tomas Lang, "On-the-Fly Rounding for Division and Square Root", *Proceeding of the 9th Symposium on Computer Arithmetic*, Santa Monica, CA, pp. 169-180, September 1989.

[4] D. E. Atkins, "Higher-Radix, Non- Restoring Division : History and Recent Development", *Proceeding of the 3rd Symposium on Computer Arithmetic*, Dallas, TX, pp. 158-160, November 1975.

[5] Y. Li and W. Chu, "A New Non- Restoring Square Root Algorithm and Its VLSI Implementations", *Proceeding of 1996 IEEE International Conference on Computer Design* : VLSI in Computer and Processor, Austin, TX, pp. 538-544, October 1996.

[6] P. Montuschi and L. Ciminiera, "Reducing Iteration Time When Result Digit is Zero for Radix 2 SRT Division and Square Root with Redundant Remainders", *IEEE Transactions on Computer Design*, vol. 42, pp. 239-246, February 1993.

[7] H. R. Srinivas, K. K. Parhi and L. A. Montalvo, "Radix-2 Division with Over-

Redundant Quotient Selection", *IEEE Transactions on Computer*, vol. 46, no. 1, January 1997.

[8] N. Burgess, "A Fast Division Algorithm for VLSI", Proceeding of *IEEE International Conference on Computer Design: VLSI in Computer and Processor*, Boston, MA, pp. 560-563, October 1991.

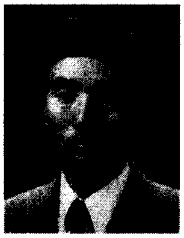
[9] Y. Li and W. Chu, "A Parallel-Array Implementations of A Non-Restoring Square Root Algorithm", *Proceeding of 1997 IEEE International Conference on Computer Design* : pp. 690-695, 1997.

[10] L. A. Montalvo, K. K. Parhi and A. Guyot, "New Svoboda-Tung Division", *IEEE Transactions on Computer*, vol. 47, no. 9, September 1998.

저 자 소 개

金 鍾 燮(正會員)

1961년 6월 28일생. 1990년 2월 숭실대학교 전자공학과 졸업(공학사). 1992년 1월 San Jose State Univ. 대학원 전자공학과 졸업(공학석사). 1997년 2월 울산대학교 대학원 전자공학과 수료(공학박사). 1993년 3월~현재 서라벌대학 전기전자전산학부 전임강사. 주관심분야는 VLSI&CAD 설계, 컴퓨터 및 데이터 통신



趙 相 福(從信會員)

1955년 6월 10일생. 1979년 2월 한양대학교 전자공학과 졸업(공학사). 1981년 2월 동 대학원 전자공학과 졸업(공학석사). 1985년 2월 동 대학원 전자공학과 졸업(공학박사). 1994년 8월~1995년 8월 Univ. of Texas, Austin 초빙학자. 1986년 3월~현재 울산대학교 전기전자 및 자동화공학부 교수. 자동차전자 연구센터 소장. 주관심분야는 ASIC 설계, 자동차 전자회로 설계, 비젼 시스템 개발, 테스트 및 테스트 용이한 설계 등임