전파 캐리의 선택에 의한 부호확장 오버헤드의 감소

이광철, 조경주, 박홍열, 정진균
전북대학교 전자정보공학부

Sign-Extension Reduction by Propagated-Carry Selection

Kwang-Chul Lee, Kyong-Ju Cho, Hong-Yul Park, Jin-Gyun Chung
Div. of Electronic & Information Engineering, Chonbuk National University
{kclee,kjcho,hypark,jgchung}@vlsidsp.chonbuk.ac.kr

요 약

고정 계수를 갖는 곱셈기의 구현 시 면적과 전력소모를 줄이기 위해서 곱셈계수를 CSD(Canonic Signed Digit) 형태로 표현 할 수 있다. CSD 계수의 1 또는 -1의 위치에 따라 부분곱들을 시프트 하여 더할 때 모든 부분곱들의 부호확장이 필요하며 이로 인해 하드웨어의 오버헤드가 증가하게된다. 본 논문에서는 부호확장 부분에서의 캐리 전파를 적절히 조절함으로써 부호확장으로 인한 오버헤드를 조절 할 수 있다는 사실을 이용하여 새로운 부호확장 오버헤드 감소방법을 제시한다. 제안한 방법과 기존의 방법을 다양한 시뮬레이션을 통해서 비교하고 기존의 방법에 비해 약 30%의 부호확장 오버헤드를 줄일 수 있음을 보인다.

Ⅰ. 서 론

신호처리 알고리즘의 하드웨어 구현 시 가장 큰 면적 및 전력소모를 차지하는 회로 중의 하나가 곱셈기이며, 곱셈기를 효율적으로 구현하기 위한 많은 연구가 진행되었다. 고정 계수를 갖는 곱셈기의 경우 계수를 CSD 형태로 표현함으로써 계수 중 0이 아닌 디지트의 개수를 감소시킬 수 있고 이에 따라 곱셈기의 면적 및 전력소모를 감소시킬 수 있다[1].

CSD 곱셈기가 디지털 필터에 사용될 경우 CSD 곱셈기의 0이 아닌 디지트의 개수를 최소화하기 위한 최적화방법들이 제안되었다[2]-[6]. 대부분의 최적화방법들은 계수의 0이 아닌 디지트 개수의 최대 값을 정한 후 그 범위 내에서 가능한 값들을 찾은 후 주파수응답을 구하고 최적의 주파수응답을 갖는 CSD 계수를 선택한다. 예로써, SSB 신호의 생성을 위한 힐버트변환 회로의 디자인에서는 0이 아닌 디지트의 개수를 4로 제한했을 때 -80dB의 저지대역대 통과대역의 에너지 비를 얻을 수 있다[5].

CSD 계수 중 1이나 -1의 위치에 따라 부분곱들을 더할 때 부호확장이 선행되어야 하며 이로 인해 하드웨어의 복잡도와 부호비트에 해당하는 데이터버스의 로드가 증가하게된다.

부호확장으로 인한 오버헤드를 줄이기 위해 compensation 벡터 방법을 사용할 수 있다[7]. Compensation vector 방법에서는 각 시프트된 데이터워드를 두 벡터의 합으로 표현한다. 예를 들어 $x_3.x_2x_1x_0\times 2^{-3}$은 다음과 같이 표현될 수 있다.

$$x_3.x_2x_1x_0\times 2^{-3}=0.00\overline{x_3}x_2x_1x_0+1.111 \qquad (1)$$

(1)에서 $\overline{x_3}$는 x_3의 보수를 나타낸다. 각 시프트된 데이터워드를 (1)과 같이 표현할 때 1로만 구성된 벡터들이 나타나는데 이러한 벡터들을 미리 더함으로써 하나의 벡터로 표현할 수 있다. 따라서 부호확장으로 인한 오버헤드는 하나의 compensation 벡터를 더하는데 필요한 오버헤드로 감소된다.

다음 계산에 대한 compensation 벡터의 적용 예를 그림 1에 보였다.

$$Y=x_9.x_8x_7x_6x_5x_4x_3x_2x_1x_0\times 1.0\bar{1}0000\bar{1}01 \qquad (2)$$

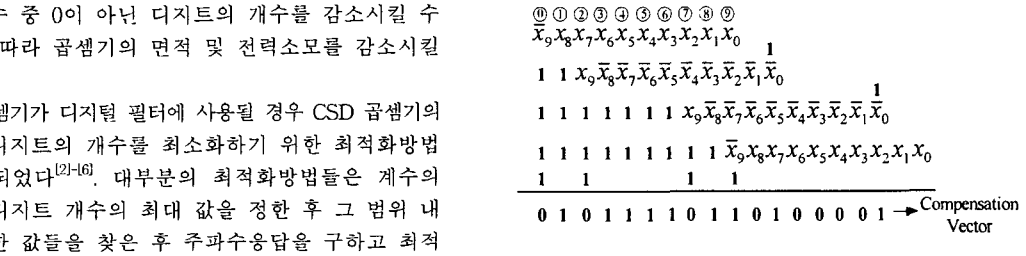

그림 1. 식 (2)에 대한 compensation 벡터 방법의 적용.

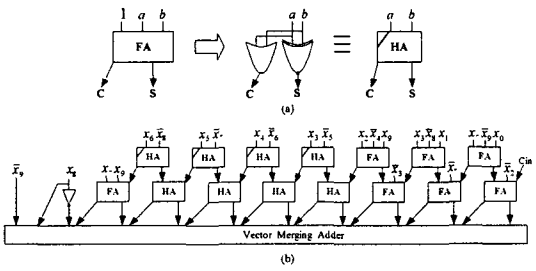

그림 2. 그림 1의 처음 10 컬럼에 해당하는 구현.

표 1. 예측 가능한 sum과 carry 비트

	Input bits	Sum	Carry
Case 1	$a\ a\ a$	a	a
Case 2	$a\ a\ \overline{a}$	$\overline{a}$	a
Case 3	$a\ a\ b$	b	a
Case 4	$a\ \overline{a}\ b$	$\overline{b}$	b

Compensation 벡터와의 덧셈 시 회로를 더욱 간단히 하기 위하여 다음 식을 사용 할 수 있다.

$$1+a+b \Rightarrow \text{Sum} = \overline{\text{Sum}(a+b)},\ \text{Carry} = a \vee b \quad (3)$$

(3)에서 $\vee$는 OR를 나타낸다. (3)은 그림 2(a)와 같이 구현될 수 있다. 그림 2(b)는 그림 1의 컬럼 0 ~ 9에 해당하는 구현 예를 보여준다.

본 논문에서는 compensation 벡터보다 더욱 효율적인 부호확장 오버헤드 제거방법을 제시한다. II절에서는 예측 가능한 덧셈에 대하여 고려하고, III절에서 새로운 부호확장 오버헤드 제거 방법을 제안한다. IV절에서 시뮬레이션 결과를 제시하고 V절에서 결론을 맺는다.

II. 예측 가능한 덧셈

전가산기는 세 비트를 입력으로 받아서 sum과 carry를 계산하여 출력하는 회로이다. 만일 세 입력 비트가 서로 독립적이지 않은 표 1과 같은 경우에는 sum과 carry를 계산 없이 바로 구할 수 있다.

식 (2)에 해당하는 부호확장된 부분곱들은 그림 3과 같다.

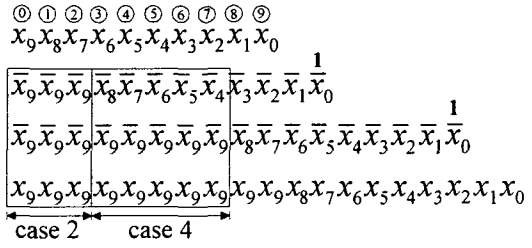

그림 3. 식 (2)에 해당하는 단순 부호확장된 부분곱.

원하는 곱셈결과를 얻기 위해서는 부호확장 부분도 덧셈에 포함되어야한다. 그러나 그림 3에서 알 수 있듯이 처음 3 컬럼은 표 1의 Case 2에 해당하고, 다음 5 컬럼은 Case 4에 해당하므로 이들 컬럼의 덧셈 결과는 간단하게 얻을 수 있다. 이러한 사실을 이용하여 새로운 부호확장 제거 방법을 다음 절에서 제시한다.

III. 부호확장 오버헤드 감소방법

이 절에서는 전파 carry의 선택절차를 제시하고 이를 이용한 부호확장 오버헤드감소방법을 제안한다.

3.1. 전파 Carry 선택

그림 3의 컬럼 6을 고려해보면 $\overline{x_5}+\overline{x_9}+x_9$은 Case 4에 해당하기 때문에 carry는 $\overline{x_5}$가 되어 컬럼 5로 전파된다. 그러나 컬럼 6에서 $x_3+\overline{x_9}+x_9$도 역시 Case 4에 해당함을 알 수 있고, 이 경우에는 x_3가 carry로 전파된다. 또한, 컬럼 7로부터 전파되어온 carry를 컬럼 5로 다시 전파 될 수 있도록 선택할 수 도 있다. 하드웨어를 감소시키기 위해서는 전달 가능한 모든 경우를 다 고려하여 전달 carry를 선택하여야한다. 이를 위해 다음의 전파 carry 선택절차를 사용한다.

전파 Carry 선택 절차

주어진 컬럼 i에 대해 가능한 모든 전파 carry 후보들을 결정하여 $p_0, p_1, \cdots p_M$이라고 한다. 각 전파 carry 후보 p_j에 대해 다음 조건을 만족하는 가장 큰 컬럼번호 n_j를 결정한다.

1. $n_j < i$,
2. p_j, 또는 $\overline{p_j}$가 컬럼 n_j에 나타난다.

가장 큰 컬럼번호 n_j를 갖는 p_j를 컬럼 i에 대한 전파 carry로 선택한다. 만일 어떠한 carry 후보도 위의 조건을 만족하는 컬럼을 갖지 못하면 임의로 전파 carry를 선택한다.

예제 1: 다음 부분곱들의 덧셈을 고려하자.

$$\begin{array}{ccc} \overline{x_4} & \overline{x_3} & \overline{x_2} \\ \overline{x_6} & \overline{x_5} & \overline{x_4} \\ x_9 & x_9 & x_9 \\ x_9 & x_9 & x_9 \\ & & c_{in} \end{array}$$

위의 부분곱들은 그림 3의 컬럼 5~7에 해당하며 c_{in}은 그 전 컬럼으로부터 전파되어온 carry를 나타낸다. 전파 carry 선택절차는 마지막 컬럼부터 시작하며, 마지막 컬럼에서 전파 carry의 후보는 x_2와 $\overline{x_4}$이다. (c_{in}은 미리 결정할 수 있는 값이 아니므로 전파 carry의 후보에서 제외한다.) x_2나 $\overline{x_2}$는 마지막 컬럼의 왼쪽에 있는 컬럼들에 나타나지 않는 반면,

x_4는 첫 번째 컬럼에 나타나므로 x_4를 전파 carry로 선택한다.

같은 방법으로 두 번째 컬럼에 대해서는 $\overline{x_4}$(마지막 컬럼으로부터 전파된 carry)를 전파 carry로 선택한다. 마지막으로 첫 번째 컬럼을 간략화 시키면 다음과 같은 단순화된 부분곱들을 얻는다.

$$\begin{array}{cccc} 1 & \overline{x_6} & x_3 & x_2 \\ & & x_5 & x_4 \\ & & x_4 & c_{in} \end{array}$$

그림 4(a)는 위의 부분곱들에 해당하는 회로구현의 예이다.

전달 carry 선택절차를 적용하지 않고 compensation 벡터 방법을 적용하여 간략화 시키면 다음과 같은 부분곱을 얻는다 (그림 1의 컬럼 5~7 참조).

$$\begin{array}{ccc} \overline{x_4} & \overline{x_3} & \overline{x_2} \\ x_6 & x_5 & x_4 \\ 1 & 1 & x_9 \\ & & c_{in} \end{array}$$

효율적인 구현을 위해서 식 (3)을 사용하면 그림 4(b)와 같이 구현된다. 그림 4로부터 제안한 방법을 이용하여 3개의 HA (1FA = 2HA)를 줄일 수 있음을 알 수 있다.

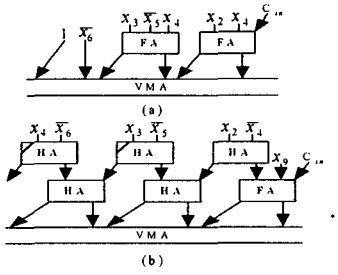

그림 4. 예제 1에 해당하는 구현:
(a) 전파 carry 선택절차에 의한 구현.
(b) compensation 벡터에 의한 구현.

3.2. 새로운 부호확장 오버헤드 감소 방법

전파 carry 선택절차를 더욱 효과적으로 적용하기 위하여 단순 부호확장과 compensation 벡터 방법을 혼합할 수 있다. 이를 이용한 새로운 부호확장 오버헤드 감소방법은 다음과 같다.

1. 단순 부호확장(그림 3 참조)을 모든 N 개의 부분곱에 적용한다. (N = 부분 곱의 개수)
2. 다음 과정을 i = 0 ~ N 동안 N+1 회 반복한다.
   - 마지막 i 개의 부분곱에만 compensation 벡터 방법을 적용하고 처음 N-i개의 부분곱은 변화시키지 않는다.
   - 새로운 N개의 부분곱에 전파 carry 선택절차를 적용한다.
   - 구현비용을 계산한다.
3. 구현비용이 가장 적은 구현을 선택한다.

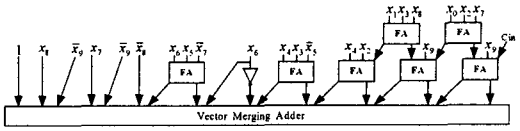

그림 5. 전파 carry 선택절차에 의한 그림 3의 처음 10 컬럼의 구현.

$$\begin{array}{l} \textcircled{0}\textcircled{1}\textcircled{2}\textcircled{3}\textcircled{4}\textcircled{5}\textcircled{6}\textcircled{7}\textcircled{8}\textcircled{9} \\ x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0 \\ \overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_8\overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4\overline{x}_3\overline{x}_2\overline{x}_1\overline{x}_0 \quad \mathbf{1} \\ \overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_9\overline{x}_8\overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4\overline{x}_3\overline{x}_2\overline{x}_1\overline{x}_0 \quad \mathbf{1} \\ \mathbf{1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1}\ \overline{x}_9x_8x_7x_6x_5x_4x_3x_2x_1x_0 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{1} \end{array}$$

그림 6. 마지막 부분곱에만 compensation 벡터 방법 적용 (i = 1).

예제 2: 제안한 방법에 의한 식 (2)의 구현을 고려하자.

먼저 단순 부호확장을 식 (2)에 적용하면 그림 3과 같은 부분곱을 얻고, 이 부분곱에 전파 carry 선택절차를 적용한 후 그 결과를 구현하면 그림 5와 같다.

다음, 그림 6과 같이 마지막 부분곱에만 compensation 벡터 방법을 적용하고 전파 carry 선택절차를 적용한 후 구현비용을 계산한다. 다음에는 마지막 두 개의 부분곱에만 compensation 벡터 방법을 적용하고 전파 carry 선택절차를 적용한 후 구현비용을 계산한다. 이러한 절차를 모든 4개의 부분곱이 compensation 벡터형태로 변환 될 때까지 반복한다. 마지막으로, 구현 비용이 가장 적은 구현을 선택한다.

본 예제에서는 그림 5의 구현이 가장 적은 비용을 차지함을 알 수 있으며 제안하는 방법에 의한 구조로 선택된다. 그림 5와 그림 2를 비교함으로써 제안하는 방법에 의해 8개의 HA를 절약할 수 있음을 알 수 있다.

IV. 시뮬레이션 결과

본 절에서는 표 2의 4가지 경우에 대한 시뮬레이션 결과를 소개한다. 곱셈기의 계수는 CSD 형태이고 입력은 2의 보수라고 가정한다. 경우 i과 ii에서는 0이 아닌 디지트 3개를 갖는 모든 11디지트 CSD 계수에 대해 시뮬레이션을 수행하고 경우 iii과 iv에서는 0이 아닌 디지트 4개를 갖는 모든 10 디지트 CSD 계수에 대해 시뮬레이션을 수행한다.

부호확장부분을 최소화하기 위해 각 경우에 대해 제안한 방법과 compensation 벡터 방법을 적용하여 시뮬레이션 한 후 사용되는 덧셈기 개수의 차이를 구한다. 그림 7~10는 각 경우에 대한 시뮬레이션 결과를 보인다.

시뮬레이션 결과로부터 제안한 방법에 의해 하나의 곱셈당 최고 5개까지의 FA를 절약할 수 있음을 알 수 있다. 전체적으로 제안한 방법에 의해 부호 확장에 의한 오버헤드를 약 30%정도 줄일 수 있다.

표 2. 시뮬레이션을 위한 4가지 경우

	CSD 계수의 0 아닌 디지트 수	CSD 계수의 워드 길이	input의 워드 길이
i	3	11	11
ii	3	11	13
iii	4	10	10
iv	4	10	11

V. 결 론

전파 carry를 적절히 선택함으로써 부호 확장에 의한 오버헤드를 줄일 수 있음을 보였다. 제안한 방법에 의해서 부호 확장에 의한 오버헤드를 기존의 방법보다 30%정도 더 감소시킬 수 있음을 다양한 시뮬레이션을 통해서 보였다.

참 고 문 헌

[1] S. W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, pp. 231-308, 1966.

[2] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with power-of-two coefficients," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 1044-1047, July 1989.

[3] D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proceedings of 1993 IEEE ISCAS*, (Chicago, IL), pp. 84-87, May 1993.

[4] C. L. Chen, K. Y. Khoo, and A. N. Willson, Jr., "An improved polynomial-time algorithm for designing digital filters with power-of-two coeffi-cients," in *Proceedings of 1995 IEEE ISCAS*, (Seattle, WA), pp. 223-226, May 1995.

[5] R. A. Hawley, T. J. Lin, and H. Samueli "A 300 MHz digital double-sidedband to single- sidedband converter in 1㎛ CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 4-10, Jan. 1995.

[6] J. G. Chung, Y. B. Kim, H. G. Jeong, K. K. Parhi, and Z. Wang, "Efficient parallel FIR filter implementations using frequency spectrum characteristics," in *Proceedings of 1998 IEEE ISCAS*, vol. V, (Monterey, CA), pp. 483-486, May 1998.

[7] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice-Hall International, Inc., 1993.

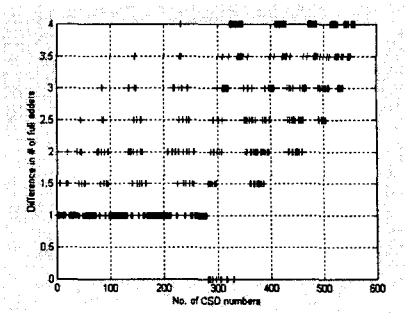

그림 7. 경우 i의 시뮬레이션 결과.

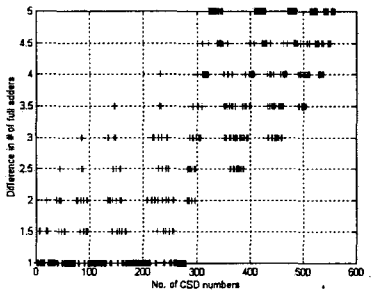

그림 8. 경우 ii의 시뮬레이션 결과.

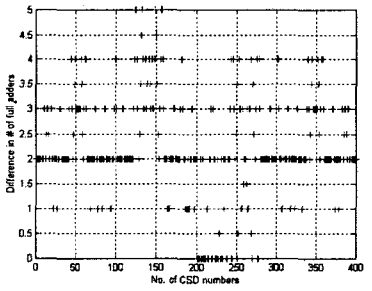

그림 9. 경우 iii의 시뮬레이션 결과.

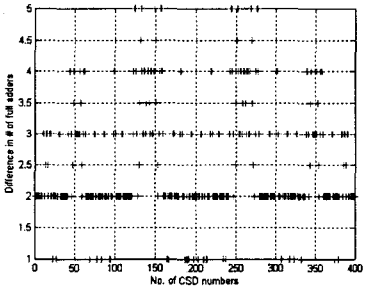

그림 10. 경우 iv의 시뮬레이션 결과