논문 2009-46SD-9-5

저오차 고정길이 그룹 CSD 곱셈기 설계

(Design of Low Error Fixed-Width Group CSD Multiplier)

김 용 은*, 조 경 주**, 정 진 균***

(Yong-Eun Kim, Kyung-Ju Cho, and Jin-Gyun Chung)

요 약

그룹 CSD 곱셈기는 프로그래머블 곱셈기에 사용되는 곱셈계수의 종류가 미리 정해져있고, 곱셈계수의 수가 많지 않은 FFT와 같은 응용에 효율적으로 사용하기 위해 최근 제안된 곱셈기이다. FFT를 비롯한 많은 DSP 응용의 VLSI 구현에서는 W비트 입력과 W비트 계수와의 곱셈 시 $(2W-1)$비트로 늘어나는 곱셈 출력 중 일부 비트만을 취하여 다음 연산에 사용한다. 본 논문에서는 워드길이가 W비트인 입력으로부터 W비트를 출력하는 고정길이 그룹 CSD 곱셈기 설계 방법을 제안한다. 양자화 오차를 효율적으로 보상하기 위해 그룹 CSD 곱셈기의 인코딩 신호를 이용하여 에러보상 바이어스를 생성한다. Synopsys 시뮬레이션을 통해 제안된 고정길이 그룹 CSD 곱셈기는 기존의 고정길이 modified Booth 곱셈기와 비교하여 전력소모에서 최대 84%, 면적에서 최대 79%까지 감소시킬 수 있음을 보인다.

Abstract

The group CSD (GCSD) multiplier was recently proposed based on the variation of canonic signed digit (CSD) encoding and partial product sharing. This multiplier provides an efficient design when the multiplications are performed only with a few predetermined coefficients (e.g., FFT). In many DSP applications such as FFT, the $(2W-1)$-bit product obtained from W-bit multiplicand and W-bit multiplier is quantized to W-bits by eliminating the $(W-1)$ least-significant bits. This paper presents an error compensation method for a fixed-width GCSD multiplier that receives a W-bit input and produces a W-bit product. To efficiently compensate for the quantization error, the encoded signals from the GCSD multiplier are used for the generation of error compensation bias. By Synopsys simulations, it is shown that the proposed method leads to up to 84% reduction in power consumption and up to 79% reduction in area compared with the fixed-width modified Booth multiplier.

Keywords : 그룹 CSD 곱셈기, modified Booth 곱셈기, 양자화 오차, 고정길이 곱셈기.

I. 서 론

GCSD (Group Canonic Signed Digit) 곱셈기는 프로그래머블 곱셈기에 사용되는 곱셈계수의 종류가 미리 정해져있고, 곱셈계수의 수가 많지 않은 FFT와 같은 응용에 효율적으로 사용하기 위해 최근 제안된 곱셈기이다[1]. GCSD 곱셈기는 미리 알려진 특정 그룹의 곱셈계수에 대해 CSD 코딩[2]과 부분곱 공유방법을 결합함으로써 생성되는 부분곱의 수를 감소시킨다. 곱셈계수의 워드길이가 W일 경우 modified Booth 곱셈기는 항상 $W/2$개의 부분곱을 생성하지만, GCSD 곱셈기는 곱셈기를 공유하는 곱셈계수들의 종류에 따라 $W/2$개, 또는 그 이하의 부분곱을 생성함으로써 효율적인 곱셈기의 구현이 가능하다.

많은 통신 및 DSP 응용의 VLSI 구현에서 승수와 피승수가 각각 W비트일 때 곱셈결과인 $(2W-1)$비트 중에서 상위 W비트를 포함한 일부 비트만을 취하여 다음 연산에 사용하는 경우가 흔히 발생하며, 이 경우 고정길이 곱셈기를 이용하여 효율적으로 구현할 수 있다. 고정길이 곱셈기에서 상위 W비트만을 취하는 경우 하

* 학생회원, *** 평생회원, 전북대학교 전자정보공학부
(Div. of Electronic & Information Engineering Chonbuk National University)
** 정회원, 항로표지기술협회
(Korea Association of Aids to Navigation)
접수일자: 2009년2월26일, 수정완료일: 2009년7월17일

위 LSB (Least Significant Bit)의 (W-1)비트에 해당하는 덧셈기 셀(adder cell)을 생략하고, 확률적인 추정에 근거하여 생성된 오차보상 바이어스를 더함으로써 양자화 오차를 보상한다.

Baugh-Wooley, modified Booth, CSD 곱셈기의 다양한 고정길이 곱셈기 설계 방법들이 제안되었다[3~8]. 오차보상 바이어스 생성에는 미리 설정한 인덱스 조건에 따라 바이어스를 생성하는 인덱스기반 오차보상[3~6] 방법과 부분곱 비트들의 양자화 오차에 미치는 영향에 따라 통계적 방법과 정확한 캐리발생 방법을 조합하여 오차발생 및 하드웨어를 감소시키는 방법들이 있다[7~8].

본 논문에서는 효율적인 고정길이 GCSD 곱셈기의 설계 방법을 제시한다. Ⅱ장에서 GCSD 곱셈기에 대해 소개하고, Ⅲ장에서는 GCSD 곱셈기의 오차보상 바이어스 회로 설계 방법을 제안한다. Ⅳ장에서는 제안한 고정길이 GCSD 곱셈기를 FFT 및 CDMA 펄스성형 필터에 응용한 결과를 제시하고, V장에서 결론을 맺는다.

Ⅱ. GCSD 곱셈기

그림 1의 radix-2^4 SDF (Single-path Delay Feedback) FFT 구조에서 첫 번째와 세 번째 프로그래머블 곱셈기는 곱셈계수로 {cos(π/8), cos(π/4), sin(π/8)}만을 갖는다. 일반적으로 이러한 곱셈은 modified Booth 곱셈기를 사용하여 구현할 수 있으며, modified Booth 곱셈기는 계수의 워드길이가 W일 때 $W/2$개의 부분곱을 생성한다.

GCSD 곱셈기에서는 부분곱의 수를 더욱 감소시키기 위해 먼저 2의 보수 형식의 모든 곱셈계수를 테이블 형태로 정리한 후 CSD 계수로 변환하고 처음 칼럼부터 시작하여 각 행에 포함된 nonzero 디지트의 개수가 1개 이하가 되면서 가능한 많은 칼럼이 포함되도록 그룹을 생성한다. 이 알고리즘에 의해 그룹을 생성할 경우, CSD 계수는 nonzero 디지트가 연속될 수 없다는 성질 때문에 생성된 그룹 수는 항상 $W/2$개 이하가 된다. 하나의 그룹은 하나의 부분곱을 생성하므로 GCSD 곱셈기에서 생성하는 부분곱의 수는 항상 $W/2$개 이하가 된다.

그림 1의 세 곱셈계수를 14비트로 표현하고 grouping 알고리즘을 적용한 결과는 표 1과 같다. 각 그룹 G_i는 부분곱 P_i를 생성하고 곱셈결과는 다음과 같이 얻어진다.

$$Y = P_4 + 2^{-2}P_3 + 2^{-4}P_2 + 2^{-6}P_1 + 2^{-8}P_0 \quad (1)$$

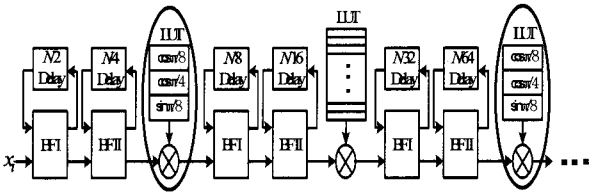


그림 1. Radix-24 SDF FFT 구조.
Fig. 1. Radix-24 SDF FFT architecture.

표 1. CSD 계수와 계수의 그룹핑
Table 1. CSD coefficients and its grouping.

Column	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cos(π/8)	1	0	0	0	-1	0	-1	0	0	1	0	0	0	0
cos(π/4)	1	0	-1	0	-1	0	1	0	1	0	0	0	0	0
sin(π/8)	0	1	0	-1	0	0	0	1	0	0	0	0	0	-1
Group	G_4		G_3		G_2		G_1		G_0					

표 2. 제어신호를 사용한 새로운 CSD 계수 표현
Table 2. New representation of CSD coefficients using control signals.

Group	G_4			G_3			G_2			G_1			G_0		
Control signals	S_4	N_4	Z_4	S_3	N_3	Z_3	S_2	N_2	Z_2	S_1	N_1	Z_1	S_0	N_0	Z_0
cos(π/8)	1	0	1	1	1	0	1	1	1	1	1	1	01	0	1
cos(π/4)	1	0	1	1	1	1	1	1	1	1	0	1	10	0	1
sin(π/8)	0	0	1	0	1	1	0	1	0	0	0	1	00	1	1

계수가 14비트이므로 modified Booth 곱셈기는 7개의 부분곱을 생성하나 GCSD는 5개의 부분곱만을 생성하므로 이 경우 약 29%의 부분곱을 감소시킬 수 있다.

부분곱 생성회로를 더욱 감소시키기 위하여 표 1의 CSD 계수에 부분곱 공유알고리즘을 적용하면 표 2와 같다. 표 2에서 S_i는 shift, N_i는 negation, Z_i는 zero control 신호를 나타낸다. GCSD 곱셈기는 계수를 저장하지 않고 표 2와 같이 코딩된 값을 저장한다. [1]에는 코딩 값 저장을 위한 look-up table 감소방법이 제시되어 있으며, 그림 1의 곱셈기를 GCSD 곱셈기로 구현하였을 때 modified Booth 곱셈기와 비교하여 면적, 전력소모, 지연시간 면에서 각각 41%, 45%, 12% 이득이 있음을 보이는 시뮬레이션 결과를 제시하고 있다.

Ⅲ. 고정길이 GCSD 곱셈기

GCSD 곱셈기의 경우 계수의 종류가 미리 정해져 있으므로 이런 특성을 이용하면 고정길이 곱셈기의 보상회로를 효율적으로 설계할 수 있다.

그림 2와 같이 입력과 계수가 각각 14비트인 GCSD

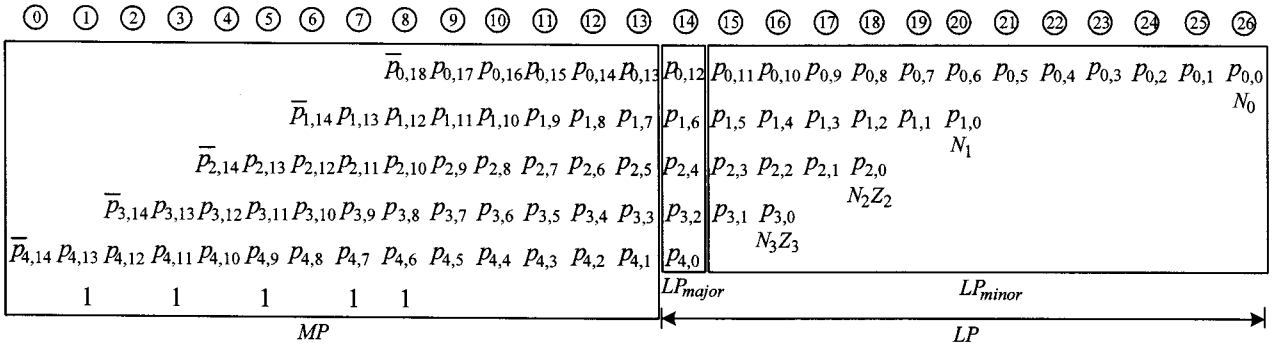


그림 2. 표 2에 대한 GCSD 부분곱 배열.
Fig. 2. Partial product array of GCSD multiplier for Table 2.

곱셈기의 부분곱 배열이 주어져 있을 때 출력에서 상위 14비트만을 계산하고자한다면 절사되는 부분을 *LP*라고 하고, *LP*중 weight가 가장 큰 부분을 LP_{major}, 나머지 부분을 LP_{minor}로 구분한다. LP_{minor}는 양자화 오차에 미치는 영향이 작기 때문에 근사적 방법으로 LP_{minor}에서 발생되는 carry를 구하고, LP_{major}에서 발생되는 carry는 정확하게 구하여 *MP*로 전달한다.

부분곱 P_i의 마지막 비트의 weight를 2^{-L_i}, P_i의 negation, 쉬프트 및 부분곱을 0으로 만드는 신호를 각각 N_i, M_i 및 Z_i로 정의한다. 그림 3에 보인바와 같이 $N_i=1$ 일 때 입력에 대해 negation을 수행하고, $Z_i=0$ 일 때 입력을 zero로 만들어 부분곱을 생성한다. 입력 비트 x_i의 기대값을 $E[x_i]=1/2$로 가정하면 각 부분곱 P_i의 LP_{minor}에 포함된 비트들의 기대값의 합을 식 (2)과 같이 구할 수 있다.

$$E[LP_{minor}(P_i)]=\begin{cases}0, & Z_i=0\\ 2^{-1}(1-2^{-f(L_i-M_i)}), & Z_i\neq 0,\ N_i=0\\ 2^{-1}(1+2^{-f(L_i-M_i)}), & Z_i\neq 0,\ N_i\neq 0\end{cases} \quad (2)$$

여기서 $f(L_i-M_i)=\begin{cases}L_i-M_i, & \text{for } L_i>M_i\\ 0, & \text{for } L_i\le M_i\end{cases}$

고정길이 modified Booth 곱셈기 설계[7]에도 유사한 방법이 사용되었으나, modified Booth 곱셈기에서는 부분곱을 0으로 만드는 신호 Z_i의 값만 고려한 반면, 본 논에서는 GCSD의 특성을 이용하여 L_i, N_i, M_i 및 Z_i 의 함수로 $E[LP_{minor}]$를 계산하기 때문에 보다 더 정확하게 계산할 수 있다.

그림 3은 표 2에 의해 GCSD 곱셈기를 구현할 때 그림 2의 LP_{minor} 영역에서 가능한 부분곱 형태를 보인다. 예를 들어 P_0는 L_0, N_0, M_0, Z_0가 *sel* 신호에 따라서 {12, 1, 0, 1}, {12, 0, 4, 1}, {12, 0, 5, 1}로 변하므로 이 변수 값을 식 (2)에 대입하면 각각에 대해 $2^{-1}(1+2^{-12})$, $2^{-1}(1-2^{-8})$, $2^{-1}(1-2^{-7})$로 부분곱 P_0의 기대 값을 계산할 수 있다.

P_0의 경우, $2^{-f(L_i-M_i)}$항들의 값이 작으므로 $E[LP_{minor}(P_0)]$의 반올림 값에 아무런 영향을 미치지 못하기 때문에 $E[LP_{minor}(P_0)]$의 반올림 값은 *sel* 신호에 무관하게 항상 $2^{-1}(1)$이 된다. 하지만 $2^{-f(L_i-M_i)}$의 값이 커지는 경우에는 반올림 값에 영향을 미칠 수 있다. 각 P_i

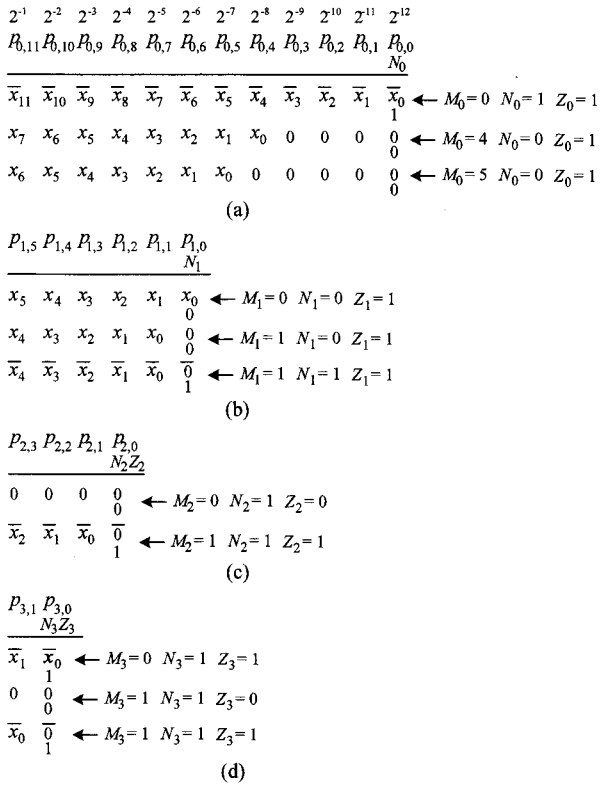


그림 3. 그림 2의 LP_{minor} 영역에서 코딩 신호에 따른 각 부분곱: (a)P0, (b)P1, (c)P2, (d)P3.
Fig. 3. Each partial product value depending on encoded signals in LP_{minor} of Fig. 2.

에 대해 계산된 $E[LP_{minor}(P_i)]$의 합으로부터 발생된 carry 신호는 LP_{major}로 전달된다.

LP_{minor}에 포함된 부분곱을 $P_L, P_{L-1}, P_{L-2}, \cdots, P_0$라고 할 때 다음관계가 성립함을 보일 수 있다.

$$0 \le f(L_L - M_L) < f(L_{L-1} - M_{L-1}) < \cdots < f(L_0 - M_0) \tag{3}$$

따라서 $E[LP_{minor}]$의 최대값은 다음과 같다.

$$\begin{aligned}
&\max\{E[LP_{minor}]\} \\
&= 2^{-1} \times \max\{(1+2^{-f(L_L - M_L)}) + (1+2^{-f(L_{L-1} - M_{L-1})}) \\
&\quad + \cdots + (1+2^{-f(L_0 - M_0)})\} \\
&= 2^{-1} \times (L+1+2^0+2^{-2}+2^{-4}+\cdots+2^{-2(L-1)}) \\
&= 2^{-1} \times (L+2) + 2^{-3} + \cdots + 2^{-2(L-1)-1}
\end{aligned} \tag{4}$$

그러므로 $f(L_{L-1} - M_{L-1})$, $f(L_{L-2} - M_{L-2})$,$\cdots$, $f(L_0 - M_0)$는 LP_{minor}에서 발생하는 carry에 영향을 줄 수 없으며 오직 $f(L_L - M_L)$만 carry에 영향을 줄 수 있다.

이상을 종합하면 $E[LP_{minor}]$는 다음과 같이 정의할 수 있다.

$$\begin{aligned}
E[LP_{minor}] = 2^{-1} \times \{ & g(L_L - M_L) \cdot 2N_L + \overline{g(L_L - M_L)} \cdot Z_L \\
& + Z_{L-1} + Z_{L-2} + \cdots + Z_0\}
\end{aligned} \tag{5}$$

여기서 $g(L_L - M_L) = \begin{cases} 1, & L_L - M_L \le 0 \\ 0, & L_L - M_L > 0 \end{cases}$

그림 4는 그림 2의 $E[LP_{minor}]$에 해당하는 바이어스를 식 (5)를 이용하여 구현한 회로로서 $E[LP_{minor}]$의 근사화 carry를 생성한다. 그림 5는 그림 4에서 발생한 carry가 더해지는 최종 부분곱 배열을 나타낸다. GCSD 곱셈기는 계수에 따라 코딩 값을 저장하고 있고, LP_{minor}의 기대 값을 미리 알고 있으므로 이러한 특징을 이용해 그림 4의 회로를 더욱 간단하게 만들 수 있다.

계수에 따라 출력되어야 하는 LP_{minor}의 carry 신호를 식 (5)를 이용하여 미리 계산하고, 계수 선택신호(sel)를

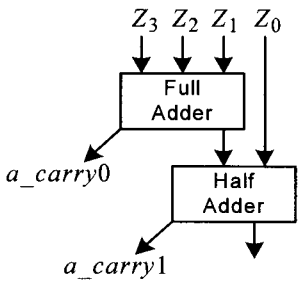


그림 4. 제안한 LP_{minor} 보상 바이어스 회로.
Fig. 4. Proposed LP_{minor} compensation bias circuit.

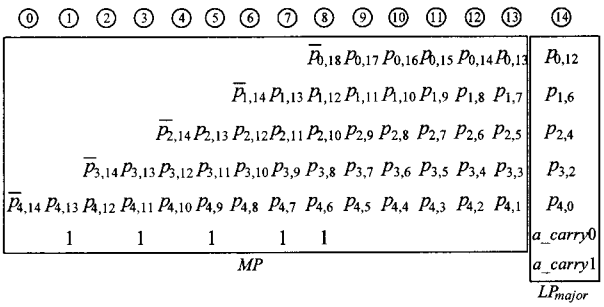
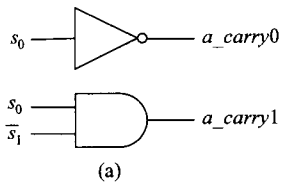


그림 5. 제안한 보상 캐리를 더한 부분곱 배열.
Fig. 5. Partial product array with proposed compensation carry.



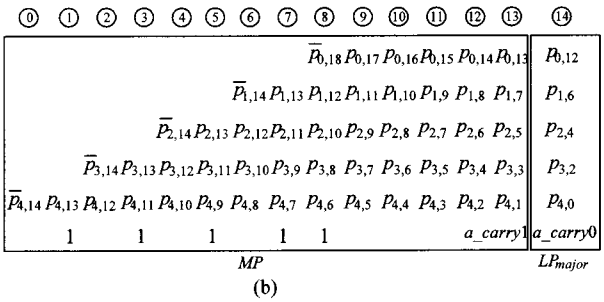


그림 6. Karnaugh 맵을 이용한 보상 바이어스 생성: (a)근사 캐리 생성회로, (b) 부분곱 배열.
Fig. 6. Compensation bias generation using Karnaugh map: (a) approximation carry generation circuit and (b) partial product array.

이용하여 Karnaugh-map을 작성하여 논리 회로를 설계하면 그림 4의 회로보다 간단하게 바이어스 회로를 구현할 수 있다.

예를 들어 그림 3의 예제에서는 계수가 3개이므로 sel 신호를 $s_1 s_0$으로 표현할 수 있으며 $s_1 s_0$이 {00, 01, 10}로 변할 때 LP_{minor}의 보상 carry 값이 식 (6)과 같이 {1, 2, 1}로 출력되어야 한다.

$$\begin{aligned}
&\bullet\ \cos(\pi/8) : s_1 s_0 = 00 \\
&\quad E[LP_{minor}] = 2^{-1}(1-2^{-8}) + 2^{-1}(1+2^{-5}) + 2^{-1}(1+2^{-3}) \\
&\quad \rightarrow a_carry = 1 \\
&\bullet\ \cos(\pi/4) : s_1 s_0 = 01 \\
&\quad E[LP_{minor}] = 2^{-1}(1-2^{-7}) + 2^{-1}(1-2^{-5}) + 2^{-1}(1+2^{-3}) \\
&\quad + 2^{-1}(1+2^{-1}) \\
&\quad \rightarrow a_carry = 2 \\
&\bullet\ \sin(\pi/8) : s_1 s_0 = 10 \\
&\quad E[LP_{minor}] = 2^{-1}(1+2^{-12}) + 2^{-1}(1-2^{-6}) + 2^{-1}(1+2^{-2}) \\
&\quad \rightarrow a_carry = 1
\end{aligned} \tag{6}$$

식 (6)과 같이 carry가 출력되어야 하는 경우 LP_{minor}의 carry 즉, *a_carry0*, *a_carry1*은 *sel* 신호에 따라 그림 6(a)와 같이 구현되며, *a_carry0*과 *a_carry1*은 그림 6(b)와 같이 부분곱 행렬에 더해 져야한다. 이 경우 그림 5와 비교하여 최대 부분곱의 높이가 더 작으므로 면적, 전력소모, 속도 면에서 효과적이다.

Ⅳ. 고정길이 Group CSD 곱셈기의 성능분석

표 3은 그림 2의 보상 회로를 (2*W*-1)비트의 이상적인 곱으로부터 *W*비트로 반올림하는 방법, 절사하는 방법, [7]의 방법, 제안한 방법을 사용하여 오차를 비교한 결과이다. [7]의 방법이 다른 고정 길이 곱셈기와 비교하여 양자화 오차가 제일 작은 것으로 알려져 있으므로 본 논문에서는 [7]에서 제안된 방법과 비교한다. GCSD_fixed는 고정길이 GCSD 곱셈기를 의미한다.

표 4는 표 3의 각 방법을 하이닉스 0.18μm공정을 이용하여 합성한 결과이다. FFT에 사용된 고정길이 GCSD 곱셈기의 경우 제안한 곱셈기의 절대 양자화 오차의 평균은 [7]의 절대 양자화 오차 평균의 약 90%이며, 면적, 전력소모, 지연시간은 [7]의 면적, 전력소모, 지연시간보다 각각 29%, 36%, 10% 정도 적다.

고정길이 GCSD 곱셈기의 또 다른 응용으로 CDMA에서 사용되는 CDMA 펄스성형 필터를 고려하자. 그림 7은 펄스성형 필터의 구조이며, ROM 1의 계수는 표 5와

표 3. FFT 응용에 대한 고정길이 곱셈기의 오차성능 비교

Table 3. Comparison of error performance in FFT application.

	최대오차	평균	분산
Rounding	6.1035×10^{-5}	3.0518×10^{-5}	3.1046×10^{-10}
Truncation	1.2206×10^{-4}	6.0913×10^{-5}	1.2418×10^{-9}
[6]의 방법	3.2139×10^{-4}	1.0635×10^{-4}	6.5125×10^{-9}
[7]의 방법	1.4734×10^{-4}	4.1126×10^{-5}	8.6590×10^{-10}
제안한 방법	1.2827×10^{-4}	3.6927×10^{-5}	6.6252×10^{-10}

표 4. FFT 응용에 대한 Synopsys 시뮬레이션 결과

Table 4. Synopsys simulation results of FFT application.

	[7]의 방법	GCSD	제안한 GCSD_fixed
면적(cell)	8159	7577	5779
전력소모(mW)	15.3	12.9	9.8
지연시간(ns)	10.25	11.89	9.32

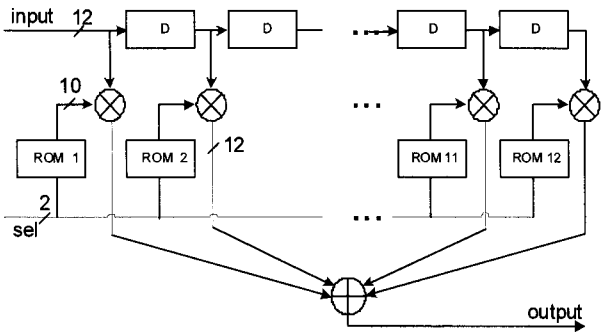


그림 7. CDMA 펄스 성형필터 구조.

Fig. 7. CDMA pulse shaping filter architecture.

표 5. 그림 7의 ROM1 계수

Table 5. Coefficients stored at ROM 1 in Fig. 7

sel	ROM 0
00	1 1 1 1 1 1 1 0 1 0
01	1 1 1 1 1 1 1 0 0 0
10	1 1 1 1 1 1 0 1 1 1
11	1 1 1 1 1 1 1 1 0 0

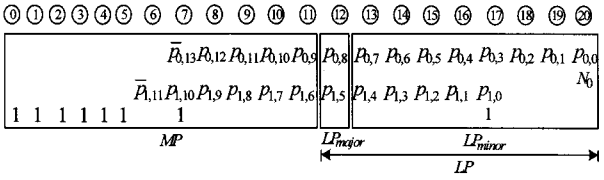


그림 8. ROM1 계수의 GCSD 곱셈기 부분곱 배열.

Fig. 8. Partial product array of GCSD multiplier in ROM 1 coefficients.

표 6. CDMA 펄스성형필터에 대한 고정길이 곱셈기의 성능 비교

Table 6. Comparison of performance in CDMA pulse shaping filter application.

	최대오차	평균	분산
Rounding	2.4414×10^{-4}	1.2207×10^{-4}	4.9703×10^{-9}
Truncation	4.8733×10^{-4}	2.2435×10^{-4}	1.9868×10^{-8}
[6]의 방법	10.8337×10^{-4}	4.9220×10^{-4}	9.6814×10^{-8}
[7]의 방법	4.8828×10^{-4}	1.3738×10^{-4}	8.4731×10^{-9}
제안한 방법	4.8086×10^{-4}	1.7785×10^{-4}	1.5477×10^{-8}

표 7. CDMA 펄스성형필터에서 고정길이 곱셈기의 Synopsys 시뮬레이션 결과

Table 7. Synopsys simulation results of CDMA pulse shaping filter application.

	[7]의 방법	GCSD	제안한 GCSD_fixed
면적(cell)	6116	2343	1338
전력소모(mW)	10.9	3.7	1.77
지연시간(ns)	9.56	8.14	4.46

같다. ROM 1에 대한 GCSD 곱셈기의 부분곱 배열은 그림 8과 같고, ROM 1과 같이 계수의 절대값이 작을 때 바이어스회로를 이용하여 LP_{minor}의 부분곱을 제거하면 더욱 효율적인 구현이 가능하다. 표 6은 오차성능의 비교 결과이고, 표 7은 그림 8을 구현한 결과이다.

표 6과 표 7에서 [7]의 방법과 오차는 동일하며 면적, 전력소모, 지연시간은 [7]에 비해 각각 78%, 84%, 54% 정도 적음을 알 수 있다. 그림 7에서 ROM 2~ROM 12의 계수에 대해서도 유사한 방법으로 구현될 수 있다.

V. 결 론

본 논문에서는 GCSD 곱셈기의 경우 계수의 종류가 미리 정해져 있다는 특성을 이용하여 고정길이 GCSD 곱셈기의 효율적인 보상회로 설계 방법을 제시하였다. 제안한 방법은 CSD 코딩 신호를 모두 고려하여 보다 정확한 기대 값을 계산함으로서 양자화 오차를 줄였으며, 계수 선택 신호를 이용하여 근사 carry 생성 회로를 더욱 효율적으로 설계 하였다.

Radix-2^4 FFT에서 사용되는 고정길이 계수 그룹 곱셈기 구현결과 제안한 방법이 [7]의 방법 보다 면적, 전력소모, 지연시간에서 29%, 36%, 10% 정도 효율적이었으며, CDMA 펄스성형 필터의 경우 78%, 84%, 54% 정도 효율적임을 보였다.

참 고 문 헌

[1] Y. E. Kim, S. H. Cho, and J. G. Chung, "Modified CSD group multiplier design for predetermined coefficient groups," in *Proc. IEEE ISCAS 2008*, pp. 3362-3365, May, 2008.

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

[3] J. M. Jou, S. R. Kuang, and R. D. Chen, "Design of a low-error fixed-width multipliers for DSP applications," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 6, pp. 836-842, June 1999.

[4] L. D. Van, S. S. Wang, and W. S. Feng, "Design of the lower error fixed-width multiplier and its application," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 1112-1118, Oct. 2000.

[5] L. D. Van, and C. C. Yang, "Generalized low-error area-efficient fixed-width multipliers," *IEEE Trans. Circuits Syst.* I, vol. 52, pp. 1608-1619, Aug. 2005.

[6] M. A. Song, L. D. Van, T. C. Huang, and S. Y. Kuo, "A generalized methodology for low-error and area-time efficient fixed-width Booth multipliers," in *Proc. IEEE Int. Midwest Symp. Circuits Systems*, vol. 1, pp. 9-12, Jul. 2004.

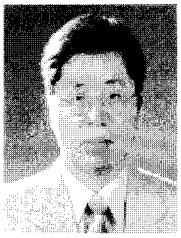
[7] K. J. Cho, K. C. Lee, J. G. Chung and K. K. Parhi, "Design of low-error fixed-width modified Booth multiplier," *IEEE Trans. VLSI Systems*, vol. 12. pp. 522-531. May 2004.

[8] S. M. Kim, J. G. Chung and K. K. Parhi, "Low error fixed-width CSD multiplier with efficient sign extension," *IEEE Trans. Circuits Syst. II*, vol. 50, pp. 984-993, Dec. 2003.

저 자 소 개

김 용 은(학생회원)
2005년 전북대학교 전자공학과 학사 졸업
2007년 전북대학교 정보통신공학과 석사 졸업
2007년~현재 전북대학교 전자정보공학부 박사
<주관심분야 : 통신, 신호처리, 반도체>

조 경 주(정회원)-교신저자
2000년 원광대학교 전자공학과 학사 졸업
2002년 전북대학교 정보통신학과 석사 졸업
2006년 전북대학교 정보통신공학과 박사 졸업
2006년~2009년 2월 : 전북대학교 Post-Doc
2009년~현재 항로표지기술협회 연구개발실 과장
< 주관심 분야 > VLSI 신호처리, 저전력 회로설계, SoC 설계



정 진 균(정회원)
1985년 전북대학교 전자공학 학사 졸업
1989년 미국 미네소타 주립대학 전기공학 석사 졸업
1991년 미국 미네소타 주립대학 전기공학 박사 졸업
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>