저전력 설계를 위한 면적 절약형 곱셈기 구조에 관한 연구

이 광 현°(李 光 鉉), 임 종 석(林 種 錫)

서강대학교 전자계산학과
서울시 마포구 신수동 1번지, 121-742
전화: 02-705-8490, Fax: 02-704-8273
E-mail: s197267@ccs.sogang.ac.kr, csrim@ccs.sogang.ac.kr

A Hardware Reduced Multiplier for Low Power Design

Lee, Kwang Hyun, Rim, Chong Suck

Dept. of Computer Science and Engineering, Sogang University.

Abstract

In this paper, we propose a hardware reduced multiplier for DSP applications. In many DSP application, all of multiplier products were not used, but only upper bits of product were used. Kidambi proposed truncated unsigned multiplier for this idea. In this paper, we adopt this scheme to Booth multiplier which can be used for real DSP systems. Also, zero input guarantees zero output that was not provided in the previous work.

1. 서론

곱셈기는 FIR filter, FFT, general DSP등 여러 부분에서 중요한 용도로 사용되는 대표적인 macro cell 이다. Unsigned array multiplier를 기본으로, 2의 보수 연산이 가능한 Baugh-Wooley array multiplier[3], Booth array multiplier[1][4]등 여러 가지 구조가 제안되었다.

실제로 많은 회로에서 곱셈기의 출력중 필요한 만큼 취하여 상위 비트 부분만 사용하는 경우가 많다. 이에, Kidambi는 truncated unsigned 곱셈기[2]를 제안하고, 필요하지 않은 하위 비트에 대한 연산 부분을 잘라 내어, 면적 이득을 얻을 수 있도록 하였다. 본 논문에서는 이런 개념을 고속의 2의 보수 연산이 가능하도록 Booth 곱셈기에 적용하였다.

2. Truncated unsigned 곱셈기

두 개의 k비트 unsigned 입력를 서로 곱해서 그 결과를 얻기 위해서는 $(k-1)k$개의 full adder를 k^2개의 AND gate와 조합해서 array 형태(그림 1)로 구현하면 된다. 그 결과 $2k$비트의 곱셈 결과를 얻게 된다. 하지만, 많은 DSP 응용회로에서는 $2k$비트의 곱셈 결과를 모두 사용하는 것이 아니라, 그중 상위 k비트만을 취해서 사용하는 경우가 많다. Kidambi는 이런 경우에 적용이 가능한 truncated unsigned 곱셈기[2](그림 2)를 제안하였다. 이 곱셈기는 상위 k비트 출력만을 얻기 위해서 하위 비트 계산에 사용되는 회로를 잘라 낸 구조이다. 이렇게 하면 대략 50% 정도의 면적을 줄일 수 있게 된다. 단, 잘려 나간 부분이 상위 비트에 전혀 영향을 주지 않는 것은 아니기 때문에, 이에 따른 오차가 생긴다. 오차의 계산은 확률적인 접근을 통해서 cut line(그림 1 참고)에 의해 잘리는 carry 출력에 대한 확률을 계산하여, 이를 오차 교정에 사용하였다.

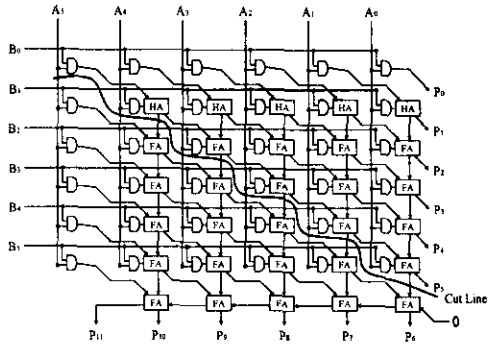

그림 1. 6x6 비트 unsigned 곱셈기

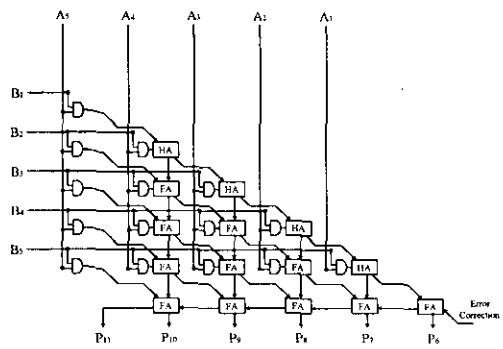

그림 2. 6x6 비트 truncated multiplier

3. Truncated Booth 곱셈기

본 논문에서는 truncation 개념을 Booth 곱셈기에 적용한 truncated Booth 곱셈기를 제안한다. Kidambi는 2의 보수 연산이 가능한 Baugh-Wooley 곱셈기[3]에 적용 가능함은 언급하였지만, 실제로 사용하기 위해서는 Booth 곱셈기에 적용될 필요가 있다.

3.1. 구조

일반적인 Booth 곱셈기(그림 3)은 Booth encoder, Booth selector, Booth adder, Booth first 등의 회로로 구성되어 있으며, 보다 자세한 구조는 참고문헌 [1]을 참고한다. 본 논문에서는 이런 구조의 Booth 곱셈기에 truncation 개념을 적용한다.

상위, 하위 비트 출력을 나누는 cut line(그림 3 참고)을 기준으로 하위 비트를 계산 하는 부분을 잘라 낸다. 중간 결과를 계산하는 Booth selector, Booth adder, Booth first가 row에 따라 필요한 부분만 남기고 잘려 진다. 예를 들어 첫 번째 row(j=0)인 경우, 0행부터 6행까지의 Booth selector와 Booth first가 잘리게 된다. 그리고, 분석을 통해서 얻은 결과를 이용해서, Booth encoder의 N출력을 쉬프트시켜 최종적으로 truncated Booth 곱셈기(그림 4)를 얻게 된다.

하위 비트를 계산하는 부분이 잘려 나갔기 때문에, 오차가 발생하며, 이는 3.2장에서 분석하고, 3.3장에서 오차를 교정한다.

3.2. 오차 분석

Kidambi의 truncated 곱셈기의 오차 분석할 때와 유사하게, 각 블록들의 입출력에 대한 진리표로부터 cut line에 의해 잘리는 carry 비트들에 대한 확률을 구해서, 예상 오차를 구하게 된다.

확률을 구하기 위해서 먼저 각 신호들에 대한 정의를 한다. 첨자 i, j는 행과 열을 나타낸다. 그림 3에서 보여지듯이 행은 제일 오른쪽이 0행이며, 왼쪽으로 가면서 1씩 증가한다. 열을 제일 위쪽이 0열이며, 아래쪽으로 가면서 1씩 증가한다. P_{X_i}, P_{Y_i}는 각각 승수(multiplicand), 피승수(multiplier)의 각 비트에 대한 확률이다. $P_{N_j}, P_{A_j}, P_{B_j}$는 j열의 Booth encoder의 출력에 대한 확률이다. $P_{T_{i,j}}$는 j열 i행의 Booth selector의 출력의 확률이다. $P_{S_{i,j}}, P_{C_{i,j}}$는 Adder row의 각 블록에서의 출력의 확률이다.

곱셈기의 입력 승수와 피승수는 k 비트이며, 각 비트가 1일 확률이 0.5라고 가정하고($P_{X_i}=P_{Y_i}=0.5$), 차례대로 확률을 계산할 수 있다.

먼저 Booth encoder의 출력에 대한 확률을 계산하는 데 Booth encoder의 진리표로부터 구한다.

$$P_{X_i}=P_{Y_i}=0.5 \qquad \text{(for all } 0\le i<k\text{, 가정)}$$
$$P_{N_j}=P_{Y_{2j+1}}=0.5$$
$$P_{A_j}=0.5$$
$$P_{B_j}=0.25 \qquad \text{(for all } 0\le j<k/2\text{)}$$

구해진 Booth encoder의 출력으로부터 Booth selector의 출력인 $T_{i,j}$의 확률을 구한다. 이 역시 Booth selector의 진리표로부터 얻어 진다.

$$\begin{aligned}P_{T_{i,j}} &= P_{N_j}\cdot(P_{A_j}\cdot P_{X_i}+P_{B_j}\cdot P_{X_{i-1}})\\ &\quad + Q_{N_j}\cdot[1-(P_{A_j}\cdot P_{X_i}+P_{B_j}\cdot P_{X_{i-1}})]\\ &= 0.5\cdot 0.375+0.5\cdot 0.625 = 0.5\end{aligned}$$

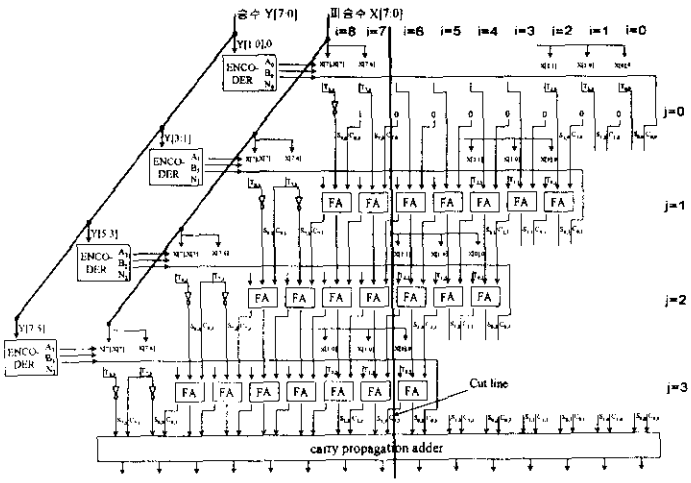

그림 3. 8x8 비트 Booth 곱셈기

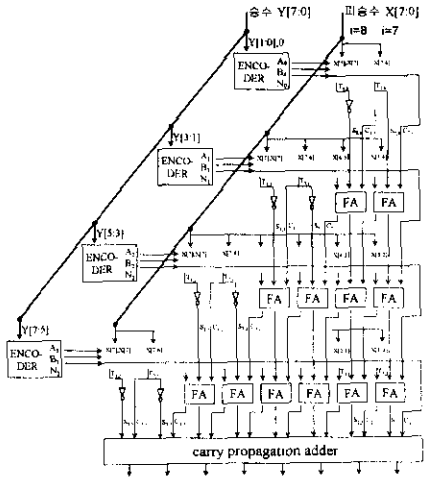

그림 4. 8x8 비트 truncated Booth 곱셈기

그리고, Booth first회로(j=0)의 출력에 대한 확률을 계산한다.

$$P_{S_{i,0}} = \begin{cases} 1 - P_{T_{i,0}} & (i=k) \\ P_{T_{i,0}} & (0 \le i \le k-1) \end{cases}$$

$$P_{C_{i,0}} = \begin{cases} 1 & (i=k) \\ 0 & (1 \le i \le k-1) \\ P_{N_0} = 0.5 & (i=0) \end{cases}$$

이제 각 adder row($j \ge 1$)에서의 출력에 대한 확률을 구한다.

$$P_{S_{i,j}} = 1 - P_{T_{i,j}} \qquad (k-1 \le i \le k)$$

$$\begin{aligned} P_{S_{i,j}} = {} & P_{T_{i,j}} \cdot P_{S_{i+1,j-1}} \cdot P_{C_{i,j-1}} \\ & + P_{T_{i,j}} \cdot Q_{S_{i+1,j-1}} \cdot Q_{C_{i,j-1}} \\ & + Q_{T_{i,j}} \cdot P_{S_{i-1,j-1}} \cdot Q_{C_{i,j-1}} \\ & + Q_{T_{i,j}} \cdot Q_{S_{i+1,j-1}} \cdot P_{C_{i,j-1}} \end{aligned} \qquad (0 \le i \le k-2)$$

$$P_{C_{i,j}} = P_{N_j} = 0.5 \qquad (i=k)$$

$$\begin{aligned} P_{C_{i,j}} = {} & P_{T_{i,j}} \cdot P_{S_{i+1,j-1}} \cdot P_{C_{i,j-1}} \\ & + P_{T_{i,j}} \cdot P_{S_{i+1,j-1}} \cdot Q_{C_{i,j-1}} \\ & + P_{T_{i,j}} \cdot Q_{S_{i+1,j-1}} \cdot P_{C_{i,j-1}} \\ & + Q_{T_{i,j}} \cdot P_{S_{i+1,j-1}} \cdot P_{C_{i,j-1}} \end{aligned} \qquad (1 \le i \le k-1)$$

$$P_{C_{i,j}} = P_{T_{i,j}} \qquad (i=0)$$

$$Q_{Z_{i,j}} = 1 - P_{Z_{i,j}} \qquad (\text{for all } Z,\ i,\ j)$$

차례대로 모든 row의 $P_{S_{i,j}}$와 $P_{C_{i,j}}$가 구해졌으면, 이로부터 cut line에 걸쳐져 있는 carry들의 확률을 모두 더해준다. 이 값이 carry로 인해서 생기는 오차의 예상 값이 된다.

$$E(C) = \sum_{\text{for each } C_{i,j} \text{ of Cut line}} P_{C_{i,j}}$$

Carry에 의한 오차 뿐만 아니라 하위 비트 출력이 곱셈이 결과에 미치는 영향도 고려해야 한다. CPA로 들어 가게 되는 $S_{i,j}$과 $C_{i,j}$들의 확률을 각각 구해서 자릿수에 맞게 더해 주어서 구한다.

$$\begin{aligned} E(LSB) = {} & (P_{C_{0,n/2}} + P_{S_{0,n/2}}) \cdot 2^{-1} \\ & + \sum_{j=0}^{n/2-1} [\,(P_{C_{0,j}} + P_{S_{0,j}}) \cdot 2^{2j-1-n} \\ & + (P_{C_{1,j}} + P_{S_{1,j}}) \cdot 2^{2j-2-n}] \end{aligned}$$

3.3. 오차 교정

Cut line을 기준으로 하여 회로를 잘라 내면, 발생하는 문제가 하나 있는 데, 입력 X 또는 Y 중 하나가 0 일 때, 출력이 0이 나오지 않을 수 있다는 점이다. 만약 승수 Y가 0 이라면, 이는 Booth encoder에 의해 항상 no operation을 수행하는 신호가 나가가 때문에, 0 출력이 쉽게 보장된다. 하지만, 피승수 X가 0이고, 승수 Y는 0이 아니라면, Booth selector에 의해 모두 반전된 값이 나갈 수 있다. N_j 값이 여기에 더해져서 모두 0 으로 다시 바뀌어야 하는데, truncation에 의해 이 회로 역시 잘리게 된다. N_j 값을 해당 row의 잘려진 비트수 만큼 쉬프트하여 cut line에 의해 잘리는 $C_{i,j}$에 연결하여서 문제를 해결하였다.

N_j 값으로 인한 영향을 제외한다면, truncated Booth 곱셈기는 그 오차가 음의 값으로 발생하는 데, N_j 를 쉬프트시켜서 더해줌에 따라, 이는 결과값을 0.5

비트수	이론적 계산치				실제 결과치	
	E(C)	E(LSB)	E(N)	E(Tr)	E(Tr)	Var
4	0.188	0.718	-1.0	-0.094	-0.075	0.232
8	0.954	0.924	-2.0	-0.122	-0.124	0.550
16	2.890	0.995	-4.0	-0.125	-0.129	1.166
24	4.876	0.999	-6.0	-0.125	-0.127	1.758
32	6.875	1.000	-8.0	-0.125	-0.129	2.182

표 1. Truncated Booth 곱셈기의
예상 오차 및 실제 오차

만큼 교정하는 효과를 가진다. N_j 에 의한 교정은 각 row 마다 이루워 지게 되며, N_j 로 인한 오차 교정 기대값은 0.5 이므로

$$E(N) = -0.5 \cdot (number\ of\ rows)$$

이 된다. 그러므로 전체 오차의 기대값 $E(Tr)$은 다음과 같이 표현될 수 있다.

$$E(Tr) = E(C) + E(LSB) + E(N)$$

이에 대한 통계 결과는 표 1에 나타나 있다.

오차는 unsigned truncated 곱셈기와는 달리, N_j로 인한 영향에 의해서, 비트수가 커질수록 오차 값이 증가하는 것이 아니라, 일정 값에 수렴함을 알 수 있다. 다만, 비트수가 커질수록 오차의 분산은 증가한다. 이는 row가 증가할수록 P_{C_j}가 0.5에 수렴하기 때문에, N_j에 의한 -0.5와 상쇄된다. 계산에 의해 나온 오차의 수렴 값은 -0.125이며, 이는 교정 없이 무시 가능하다.

4. 결과 비교

4.1. 정확도 비교

여러 가지 비트수에 대해 Booth 곱셈기의 평균 오차를 정리하여 표 1에 보였다. 잘려진 하위 비트 부분에 적절히 0를 덧붙혀 비트수를 같게 하여 비교한 것이다. 이론적으로 -0.125 만큼으로 오차가 발생하며, 실제 오차 역시 예상치와 근접함을 확인하였다. 이는 무시할 수 있을 정도로 작으며, 이 오차를 더욱 더 줄이는 방법에 대한 연구가 현재 진행중이다.

4.2. 면적 및 전력 소모에 대한 이득

면적에 대한 이득을 얻기 위해서 magic을 사용하여 LG 0.6 ㎛공정의 디자인 룰을 사용하여 곱셈기 회로를 구현하였다. 삼각형 형태의 곱셈기를 접어서 직사각형으로 만들어 면적을 최소화시킨 후의 면적이며, 그 결과는 표 2에 보였고, 단위는 ㎛이다. Truncated Booth

비트수	normal	truncation	면적이득
8	1051x219	627x233	37%
12	1479x327	837x342	41%
16	1834x435	1030x450	42%

표 2. 면적 계산표 (단위 ㎛)

비트수	normal	truncation	전력이득
8	3.404	1.899	44%
12	8.923	4.976	44%
16	18.036	10.031	44%

표 3. 전력소모계산표(단위 ㎽/operation)

곱셈기는 일반 Booth 곱셈기에 비해 약 37~42 %의 면적이 감소하였다.

회로가 잘려 나갔기 때문에, 소모되는 전력의 양도 당연히 줄어 든다. 전력 소모에 대한 이득은 표 3에 정리되어 있다. 무작위한 입력 값에 대한 평균 전력 소모량이며, 단위는 ㎽/operation이다. 새로운 곱셈기는 약 44%의 전력 소모 이득을 얻을 수 있었다.

5. 결론

본 논문에서 곱셈기 출력중 상위의 비트만을 내보내는 truncated Booth 곱셈기의 구조를 제안하였다. 하위 비트에 대한 출력을 계산하는 부분을 잘라 내 버린 후에, 이로 인해 발생할 수 있는 오차를 확률적인 접근을 통해서 계산하였고, 또한 0 입력에 대한 0 출력을 보장하기 위한 처리를 해 주었다. 이로써 오차가 0에 근접한 적은 면적과 전력 소모를 가지는 truncated Booth 곱셈기를 얻을 수 있었다. 이 곱셈기 구조는 곱셈 연산을 많이 하는 DSP 응용회로에서 면적과 전력 소모를 줄이기 위해서 사용되어 질수 있다.

참고문헌

[1] Neil H. E. Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design: A systems Perspective 2nd Ed.", Addison-Wesley, 1993.

[2] Sunder S. Kidambi, Fayez El-Guibaly, Andreas Antoniou, "Area-Efficient Multipliers for Digital Signal Processing Applications," *IEEE Trans. on C&S-II*, vol. 43, pp. 90-95, NO. 2, FEB, 1996.

[3] Baugh, C. R. and Wooley, B. A., "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Trans. on Computers*, Vol. C-22, No. 1-2, pp. 1045-1047, Dec. 1973.

[4] Booth, A. D., "A signed Binary Multiplication Algorithm," *Quart. J. Mech. Appl. Math.*, Vol. 4, Pt. 2, pp. 236-240, 1951.