논문 2008-45SD-2-7

팬 아웃이 고정된 carry increment 덧셈기 설계 방법

(The Design of carry increment Adder Fixed Fan-out)

김 용 은*, 정 진 균**

(Yong-Eun Kim and Jin-Gyun Chung)

요 약

가변 stage carry increment adder는 stage가 증가함에 따라 stage에서 계산되는 워드길이를 1비트씩 늘려줄 수 있으므로 속도는 $O(\sqrt{2n})$에 근접한다. 하지만 stage의 비트가 늘어남에 따라 stage에 입력되는 캐리의 팬 아웃이 증가하게 되고 이로 인하여 속도가 느려진다. 본 논문에서는 stage의 입력 비트를 증가하여도 팬 아웃이 stage에 관계없이 고정될 수 있는 알고리즘을 제안하고 37비트 덧셈기를 레이아웃하여 시뮬레이션 결과를 비교하였을 때 면적은 40% 늘어나는 것에 비해 덧셈기의 속도가 75% 향상되었다.

Abstract

According to increment of stage, the speed of changeable stage Carry-increment adder can be close to $O(\sqrt{2n})$ because the word length which is computed in stage can be lengthened by 1 bit. But the number of stage bits is increased, fan-out of carry which is inputted in stage is increased. So the speed can be slow. This paper presents a new carry-increment adder design method to fix the number of fan-outs regardless of the number of stages. By layout simulation of 37-bit adder, the area can be increased up to 40%, but speed improvement up to 75% can be achieved, by the proposed method, compared with a conventional method.

Keywords: Increment adder, Fan-out

Ⅰ. 서 론

덧셈기는 디지털 신호처리에서 대부분 사용되며 프로세서 및 많은 부분에서 이용된다. 따라서 많은 연구가 진행되었으며 특히 저전력 고속 덧셈기가 이슈가 되었다. 고속 덧셈기가 필요한 곳에서는 트리 덧셈기나 병렬 덧셈기와 같은 구조가 사용된다. 트리 덧셈기의 경우 이론적인 속도는 빠르지만 하드웨어가 매우 크다. 병렬 덧셈기는 트리 덧셈기보다 빠르지는 않지만 면적이 트리 덧셈기보다 작다. 따라서 면적 속도를 고려하여 carry lookahead, carry select adder, carry increment adder(CIA) 등 여러 가지 방식의 덧셈기를 선택하여 사용한다[1~6].

가변 stage CIA는 병렬로 덧셈이 수행되며 stage가 증가할수록 덧셈을 수행할 수 있는 워드길이가 증가한다. 가변 stage CIA의 이론적인 연산 속도는 $O(\sqrt{2n})$에 근접하지만 stage가 증가할수록 팬 아웃이 증가되어 속도가 느려지기 때문에 이론적인 연산 속도를 만족시키지 못한다[7].

본 논문에서는 가변 stage CIA에 제안한 알고리즘을 적용하여 stage가 증가하여도 팬 아웃이 고정되는 가변 stage CIA를 제안하였다.

본 논문의 Ⅱ절에서는 기존 CIA, Ⅲ절에서는 제안하는 CIA를 설명한다. Ⅳ절에서는 기존의 방법과 제안한 방법을 Synopsys사의 Design_analyzer를 이용하여 시뮬레이션하고 Ⅴ절에서 결론을 맺는다.

* 학생회원, ** 정회원, 전북대학교 전자정보공학부
(Div. of Electronic & Information Engineering Chonbuk University)

※ 이 연구에 참여한 연구자는 2단계 BK21사업의 지원비를 받았음, This work was supported by the second stage of Brain Korea 21 Project.
접수일자: 2007년11월3일, 수정완료일: 2008년1월7일

Ⅱ. Carry-Increment adder

덧셈기는 PG 네트워크를 통하여 면적 및 속도를 측정할 수 있다. 그림 1은 기존의 CIA PG 네트워크를 보여준다[7]. 첫 번째 stage에서는 $g(0)$을 이용하여 $g(1:0)$을 생성시키고 두 번째 stage에서는 $g(1:0)$을 이용하여 $g(2:0)$, $g(2:0)$을 생성시키고 세 번째 stage에서는 $g(3:0)$을 이용하여 $g(6:0)$ $g(5:0)$, $g(4:0)$을 생성시킨다. 따라서 stage가 증가할수록 전단의 특정 와어이의 팬 아웃이 증가하게 된다. 팬 아웃이 증가하면 이에 비례하여 속도가 느려지고 소비전력이 증가하게 된다. 그림 1을 수식으로 전개하면 표 1과 같다. 표 1에서 각 stage의 마지막 항들의 팬 아웃이 증가함을 알 수 있다.

Ⅲ. 팬 아웃이 고정된 Carry Increment adder

PG 네트워크에서 P그룹은 AND 게이트로 생성하기가 수월하다. 예를 들어 $p(5:2)$는 다음 식(1)과 같이 표현할 수 있다.

$$p(5:2)=p(5)\cdot p(4)\cdot p(3)\cdot p(2)\cdot p(1) \quad (1)$$

하지만 G 네트워크를 생성하려면 AND-OR 게이트의 조합으로서 생성하기가 까다롭다. 그룹 $g(m:n)$을 생성하기 위해서는 식(2)와 같은 항들이 필요하다(단 $m>j>n$). 따라서 다양한 G그룹을 생성하기 어렵기 때문에 생성된 G 그룹을 공유하여 PG 그룹을 생성시키 때문에 공유된 항은 팬 아웃이 발생한다.

$$g(m:n)=g(m:j)+p(m:j)\cdot g(m-1:n) \quad (2)$$

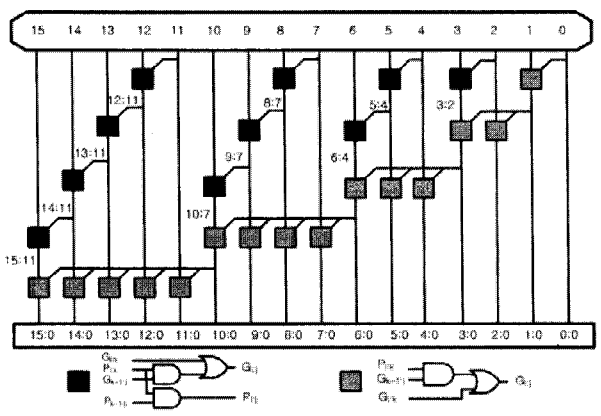


그림 1. 16-bit 가변 stage CIA PG 네트워크 다이어그램

Fig. 1. Multiplication block used in FFT.

표 1. 기존의 16 bit CIA expansion 수식 전개식 (단 OR은 + AND는 ·)

Table 1. Conventional 16 bit CIA computes the functions.

Stage1	g(1:0)=g(1)+ p(1)·g(0:0)
Stage2	g(2:0)=g(2:1)+ p(2:1)·g(0:0)
	g(3:0)=g(3:2)+ p(3:2)·g(1:0)
Stage3	g(4:0)=g(4)+ p(4)·g(3:0)
	g(5:0)=g(5:4)+ p(5:4)·g(3:0)
	g(6:0)=g(6:4)+ p(6:4)·g(3:0)
Stage4	g(7:0)=g(7)+ p(7)·g(6:0)
	g(8:0)=g(8:7)+ p(8:7)·g(6:0)
	g(9:0)=g(9:7)+ p(9:7)·g(6:0)
	g(10:0)=g(10:7)+ p(10:7)·g(6:0)
Stage5	g(11:0)=g(11)+ p(11)·g(10:0)
	g(12:0)=g(12:11)+ p(12:11)·g(10:0)
	g(13:0)=g(13:11)+ p(13:11)·g(10:0)
	g(14:0)=g(14:11)+ p(14:11)·g(10:0)
	g(15:0)=g(15:11)+ p(15:11)·g(10:0)

G그룹을 용이하게 생성 시킬 수 있도록 식(3)을 이용한다. (단 $i>j>m$)

$$g(i:m)=\overline{p(i:m)}\cdot g(i:j) \quad (3)$$

<식 (3)의 증명>

$\overline{p(i:m)}$를 드모르강 법칙에 의해서 전개하면 식(4)가 된다.

$$\overline{p(i:m)}=\{\overline{p_i}+\overline{p_{i-1}}+\overline{p_{i-2}}+\ldots+\overline{p_{m+1}}+\overline{p_m}\} \quad (4)$$

$$g(i:j)=g_i+p_i\cdot g_{i-1}+p_i\cdot p_{i-1}\cdot g_{i-2}+\ldots +p_i\cdot p_{i-1}\cdot p_{i-2}\cdot\ldots\cdot p_j \quad (5)$$

그러므로 식(4)와 식(5)를 식(3)에 대입하면 식(6)으로 전개된다.

$$\overline{p(i:m)} \cdot g(i:j) = \overline{p_i} \cdot g(i:j) + \overline{p_{i-1}} \cdot g(i:j) + \ldots + \overline{p_{m+1}} \cdot g(i:j) + \overline{p_m} \cdot g(i:j) \quad (6)$$

식 (6)의 우변 첫째 항 $\overline{p_i} \cdot g(i:j)$을 전개하면 아래와 같다.

$$\overline{p_i} \cdot g(i:j) = \overline{p_i} \cdot g_i + \overline{p_i} \cdot p_i \cdot g_{i-1} + \ldots = \overline{p_i} \cdot g_i = g_i \quad (7)$$

식(6) 우변 둘째 항$\overline{p_i} \cdot g(i:j)$을 전개하면 아래와 같다.

$$\overline{p_{i-1}} \cdot g(i:j) = \overline{p_{i-1}} \cdot g_i + \overline{p_{i-1}} \cdot p_i \cdot g_{i-1} + \overline{p_{i-1}} \cdot p_i \cdot p_{i-1} \cdot g_{i-2} + \overline{p_{i-1}} \cdot p_i \cdot p_{i-1} \cdot p_{i-2} \cdot g_{i-3} + \ldots.$$
$$\overline{p_{i-1}} \cdot g(i:j) = \overline{p_{i-1}} \cdot g_i + \overline{p_{i-1}} \cdot p_i \cdot g_{i-1} = \overline{p_{i-1}} \cdot g_i + p_i \cdot g_{i-1} \quad (8)$$

$\overline{p_m} \cdot g(i:j)$항까지 전개하여 항을 전부 OR 연산하고 각 항을 공통된 변수로 묶으면 식(9)과 같다. 식(9)에서 OR 연산항 중에 1이 존재하므로 간소화하면 결국 $g(i:m)$이 됨을 알 수 있다.

$$\begin{aligned}\overline{p(i.m)} \cdot g(i:j) &= g_i \cdot (1 + \overline{p_{i-1}} + \overline{p_{i-2}} + \cdots) \\ &+ p_i \cdot g_{i-1} \cdot (1 + \overline{p_{i-2}} + \overline{p_{i-3}} + \cdots) \\ &+ p_i \cdot p_{i-1} \cdot g_{i-2} \cdot (1 + \cdots) \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &+ p_i \cdot p_{i-1} \cdot p_{i-2} \cdot \ldots \cdot p_{m+1} \cdot g_m \\ &= g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot g_{i-2} + \cdots \\ &+ p_i \cdot p_{i-1} \cdot p_{i-2} \cdot \cdots \cdot p_{m+1} \cdot g_m \\ &= g(i:m)\end{aligned} \quad (9)$$

식(3)을 이용하여 표 1의 팬 아웃을 줄이기 위하여 마지막 G항과 연결되는 신호를 분배하여야 한다. 예를 들어 표 1의 stage 5에서$g(10:0)$은 $g(11:0)$, $g(12:0)$, $g(13:0)$, $g(14:0)$, $g(15:0)$과 연산된다. 식(3)을 이용하여 표 1의 stage 5에의 $g(10:0)$항을 표 2와 같이 $g(6:0)$, $g(7:0)$, $g(8:0)$, $g(9:0)$, $g(10:0)$로 바꿀 수 있으며 $g(10:0)$의 팬 아웃을 줄일 수 있다.

표 2에서 그룹 $g(i:0)$와 $\overline{p(i-m)}$을 AND 하여도 무방하므로 $\overline{p(i-m)}$ 을 AND 연산하여 표 2를 표 3으로 바꾸어 주면 멀티플렉서를 이용하여 CIA설계가 가능하다.

표 3을 이용하여 PG네트워크를 나타내면 그림 2와 같다. 그림 2에서 다음 stage로 넘어가는 신호들이 분배되어 팬 아웃이 감소되는 것을 알 수 있다. 설계를 용이하도록 하게 만들기 위해 그림 2에서와 같이 Block1, Block2, Block3 셀들이 규칙적으로 사용하여 빠른 설계가 용이하도록 하였다. 그림 3은 Block1, Block2, Block3을 이용하여 stage 4를 구성한 그림이다.

표 2. 제안한 16 bit CIA 전개식
Table 2. Proposed 16 bit CIA computes the functions.

Stage1	g(1:0)=g(1)+p(1)·g(0:0)
Stage2	g(2:0)=g(2:1)+p(2:1)·g(0:0)
	g(3:0)=g(3:2)+p(3:2)·g(1:0)
Stage3	g(4:0)=g(4:2)+p(4:2)·g(1:0)
	g(5:0)=g(5:3)+p(5:3)·g(2:0)
	g(6:0)=g(6:4)+p(6:4)·g(3:0)
Stage4	g(7:0)=g(7:4)+p(7:4)·g(3:0)
	g(8:0)=g(8:5)+p(8:5)·g(4:0)
	g(9:0)=g(9:6)+p(9:6)·g(5:0)
	g(10:0)=g(10:7)+p(10:7)·g(6:0)
Stage5	g(11:0)=g(11:7)+p(11:7)·g(6:0)
	g(12:0)=g(12:8)+p(12:8)·g(7:0)
	g(13:0)=g(13:9)+p(13:9)·g(8:0)
	g(14:0)=g(14:10)+p(14:10)·g(9:0)
	g(15:0)=g(15:11)+p(15:11)·g(10:0)

표 3. 표 2에 제안한 식(3)을 적용한 결과
Table 3. The result computed functions after appling expression (2).

Stage1	g(1:0)=g(1)+p(1)·g(0:0)
Stage2	g(2:0)=p(2:1)·g(2:0)+p(2:1)·g(0:0)
	g(3:0)=p(3:2)·g(3:0)+p(3:2)·g(1:0)
Stage3	g(4:0)=p(4:2)·g(4:0)+p(4:2)·g(1:0)
	g(5:0)=p(5:3)·g(5:0)+p(5:3)·g(2:0)
	g(6:0)=p(6:4)·g(6:0)+p(6:4)·g(3:0)
Stage4	g(7:0)=p(7:4)·g(7:0)+p(7:4)·g(3:0)
	g(8:0)=p(8:5)·g(8:0)+p(8:5)·g(4:0)
	g(9:0))=p(9:6)·g(9:0)+p(9:6)·g(5:0)
	g(10:0)=p(10:7)·g(10:0)+p(10:7)·g(6:0)
Stage5	g(11:0)=p(11:7)·g(11:0)+p(11:7)·g(6:0)
	g(12:0)=p(12:8)·g(12:0)+p(12:8)·g(7:0)
	g(13:0)=p(13:9)·g(13:0)+p(13:9)·g(8:0)
	g(14:0)=p(14:10)·g(14:0)+p(14:10)·g(9:0)
	g(15:0)=p(15:11)·g(15:0)+p(15:11)·g(10:0)

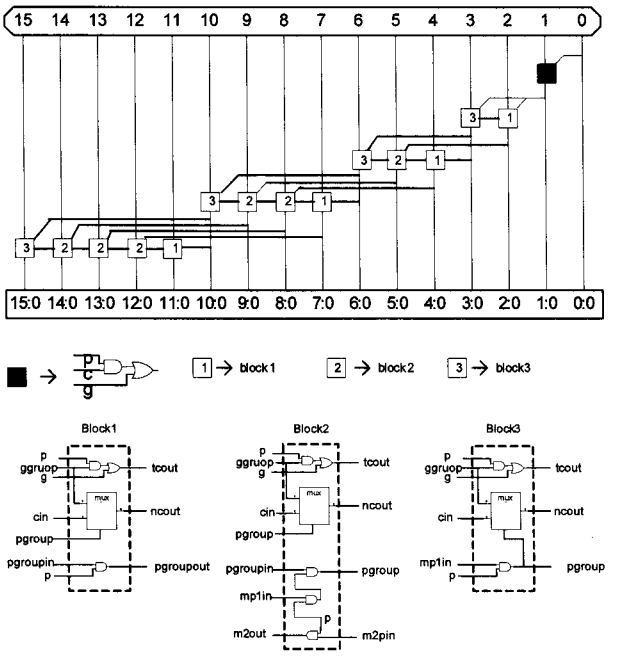


그림 2. 제안한 CIA의 PG 네트워크
Fig. 2. The proposed CLA network.

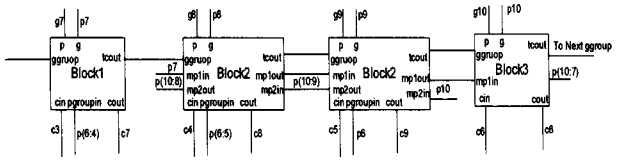


그림 3. Block1, Block2, Block3 셀로 구성된 stage 4의 구조
Fig. 3. The structure of stage 4 by Block1, Block2, Block3.

Ⅲ. 시뮬레이션 및 비교

기존의 CIA와 제안한 CIA를 설계하여 삼성 0.35μ 공정 라이브러리 셀을 이용하여 시뮬레이션 하였을 때 표 4와 같은 결과를 얻을 수 있었다. 속도 측정은 P그룹이 모두 1이고 전단의 carry-in을 0에서 1, 1에서 0으로 변화시키면서 carry-out이 출력되는 시간을 측정하였다.

표 4. 37 비트 CIA 설계 결과
Table 4. The synopsis of 37 bit CLA adder.

	기존 방법	제안한 방법	이 득
지연시간 (ps)	10646	2666	75%
면적 (cell)	156.5	219	-40%

Ⅳ. 결 론

기존의 CIA는 팬 아웃 문제로 속도가 느려지는 문제가 있었다. 이러한 문제를 해결하기 위하여 제안한 수식을 적용하여 stage를 증가시켜도 팬 아웃이 고정될 수 있는 CIA 설계 방법을 제안하였다. 또한 삼성 0.35μ 공정 라이브러리를 이용하여 기존의 CIA와 제안한 CIA를 설계하였을 때 면적에 대한 오버헤드에 비하여 속도가 향상됨을 알 수 있었다. 추후에는 Ling adder 알고리즘을 이용하여 더욱 빠르고 효율적인 CIA를 구현하도록 할 것이다.

참 고 문 헌

[1] Grad, J. and Stine, J.E., "Low power binary addition using carry increment adders," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium*, pp. 17-20, 2006.

[2] Chih-Jen Fang, Chung-Hsun Huang, Jinn-Shyan Wang, and Ching-Wei Yeh, "Fast and compact dynamic ripple carry adder design," *ASIC, 2002. Proceedings. 2002 IEEE Asia-Pacific Conference*, pp. 25-28, 2002.

[3] Wu-Tung Cheng and Patel, J. H, "A minimum test set for multiple fault detection in ripple carry adders," *Computers, IEEE Transactions*, pp. 891 -895, 1987.

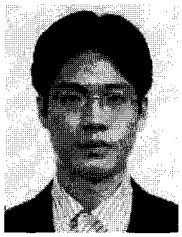
[4] Yu-Ting Pai and Yu-Kumg Chen, "The fastest carry look ahead adder," *Electronic Design, Test and Applications, 2004. DELTA 2004. Second IEEE International Workshop*, pp.434-436, 2004.

[5] Joonho Lim, Dong-Gyu Kim, and Soo-Ik Chae, "A 16-bit carry-look ahead adder using reversible energy recovery logic for ultra-low-energy systems," *Solid-State Circuits, IEEE Journal*, pp.898-903, 1999.

[6] Grad. J "New Algorithms for carry propagation," in *Proc. 15th ACM Great Lakes Symp. on VLSI*, pp. 396-399, 2005.

[7] Neil H. E. Weste and David Harris, "CMOS VLSI Design A Circuits and Systems Perspective," Addison-Wesley publisher, 2004.

저 자 소 개



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

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>