마이크로파이프라인 구조의 16bit 비동기 곱셈기

장미숙, 이유진, 김학윤, 이우석, 최호용
충북대학교 반도체공학과

Asynchronous 16bit Multiplier with micropipelined structure

Mi-Sook Jang, Eugene Lee, Hak-Yun Kim, Woo-Suk Lee, Ho-Yong Choi
Dept. of Semiconductor Eng., Chungbuk Nat'l Univ.
jms@mickey.chungbuk.ac.kr

Abstract - A 16bit asynchronous multiplier has been designed using micropipelind structure with 2 phase and data bundling. And 4-radix modified Booth algorithm, CPlatch(Cature-Pass latch) and modified 4-2 counters have adopted in this design. It is implemented in 0.65um double-poly/double-metal CMOS technology by using 12,074 transistors with core size of 1.4x1.8mm^2. And our design results in a computation rate 55MHz at a supply voltage of 3.3V.

1. 서 론

VLSI 제조 기술의 발달로 인해 소자의 속도는 더욱 고속으로 되고 회로의 규모가 커짐에 따라, 기존의 클럭을 갖는 동기식 시스템은 clock skew라는 물리적 제약에 의해 성능향상의 포화에 직면하고 있다[1,2]. 이 동기식 시스템의 한계를 극복하기 위해 최근 비동기식 설계 기법이 활발하게 연구되고 있다[1-4].

본 논문에서는 비동기식 시스템의 일부분으로 이용될 수 있고, 향후 멀티미디어·초고속 정보통신 분야의 고성능, 저전력 시스템 설계에 많이 사용되고 있는 서브시스템 중의 하나인 비동기식 곱셈기를 설계한다. 전체구조로는 비동기식의 마이크로파이프라인(micropipeline; 이하 마이크로파이프라인)구조[2]를 사용하고, 곱셈속도의 향상을 위해 4-radix modified Booth 알고리즘을 이용하여 설계한다. 마이크로파이프라인 구조는 기존의 동기식 설계에서의 로직 블럭을 그대로 사용할 수 있어 설계시간을 줄일 수 있고, 2상 프로토콜(protocol)을 기본적으로 사용할 수 있기 때문에 속도가 빠르다[4].

본 논문에서는 사용된 마이크로파이프라인 구조와 modified Booth 알고리즘을 설명하고, 이를 이용한 곱셈기 설계에 대해 기술하고 설계결과를 보인다.

※ 본 연구는 반도체설계교육센터의 부분적인 지원을 받아 이루어졌음.

2. 마이크로파이프라인과 Booth 알고리즘

(1) 마이크로파이프라인

마이크로파이프라인방식은 시스템의 제어구조에 있어서 기존의 동기식에 사용되는 전역 클럭에 의해 제어되는 구조가 아닌, 연산완료신호와 handshake 프로토콜에 의해 제어되는 구조를 갖는다[1,2]. 연산완료 신호를 생성하는 방법은 data coding방식과 data bundling방식으로 나누어진다[1]. data coding방식은 주로 dual-rail code방식을 이용하는데 로직블럭의 지연 변동율(데이타 입력에 따라 로직의 출력에 결과가 나오는 시간의 변동율)이 클 경우에는 고속이나, 이를 구성하기 위한 하드웨어 오버헤드가 크다. 한편 data bundling방식은 기존의 동기식으로 설계된 회로를 대부분 사용할 수 있기 때문에 동기와 비교했을 때 하드웨어의 부담이 적다는 장점이 있으나, 연산완료신호를 발생하기 위한 부가 지연회로가 필요하다.

Handshake 프로토콜에는 2상(phase) 프로토콜과 4상프로토콜로 나누어진다. 2상 프로토콜은 4상 프로토콜과는 달리 RTZ(Return To Zero)가 없기 때문에 속도가 빠르다는 장점이 있다[4].

본 논문에서는 속도향상을 위해 2상 프로토콜을 채택하고, 하드웨어의 오버헤드를 줄이기 위해 지연소자를

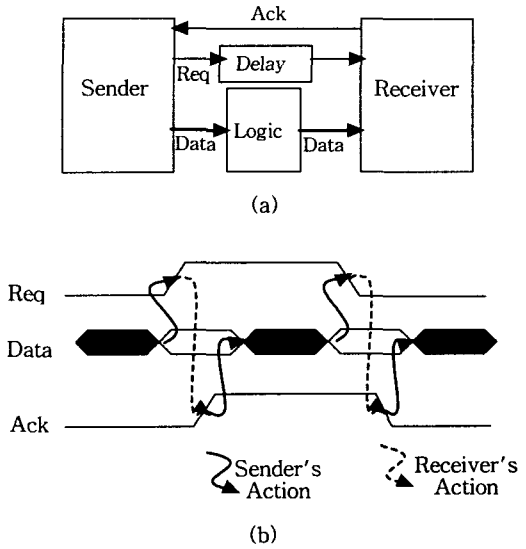


그림 1 2상 bundled data 프로토콜 (a) 2상 bundled data 인터페이스 (b) 2상 프로토콜

사용하는 data bundling방식을 이용하는 마이크로파이프라인 구조를 사용하여 곱셈기를 설계한다.

기본적으로 마이크로파이프라인 각 단은 그림 1(a)과 같은 2상 bundled data 인터페이스 구조를 가지며 데이터 전송에 있어 그림 1(b)의 2상 프로토콜을 사용한다. sender에서는 유효한 데이터를 전송하고 request신호를 천이 시킨다. 조합회로의 예측된 연산시간과 전달지연시간이 경과 한 후에 receiver는 연산 완료된 유효한 데이터를 받아들이고, acknowledge 신호를 sender에 발생시켜 데이터의 전송완료 및 다음 데이터 수신준비를 알린다. 그림 1(b)에서처럼 2상 프로토콜은 데이터가 request 신호의 rising edge와 falling edge 모두에서 전송되기 때문에 rising 혹은 falling edge에서만 데이터를 전송하는 4상 프로토콜보다 빠르다. 이때 "request 신호가 오기 전에 유효한 데이터가 먼저 와 있어야 한다"는 bundled constraint를 만족하기 위해 sender와 receiver에 지연소자를 삽입해야 하는데 이 지연소자의 값은 로직 단의 지연시간과 전달지연을 포함하는 값보다 큰 값으로 해야한다.

다단 마이크로파이프라인의 구조는 그림 2와 같이 C-gate와 지연소자를 사용하여 구성할 수 있다. 여기서 C-gate는 2상 프로토콜의 올바른 동작을 보증하기 위해 사용된다.

표 1 4-radix Booth 엔코딩 방법

Bit position			Action	Encoding		
i+1	i	i-1		N	S	Z
0	0	0	0	0	0	1
0	0	1	+M	0	0	0
0	1	0	+M	0	0	0
0	1	1	+2M	0	1	0
1	0	0	-2M	1	1	0
1	0	1	-M	1	0	0
1	1	0	-M	1	0	0
1	1	1	0	1	0	1

(2) Booth 알고리즘

본 논문에서는 곱셈수의 비트들을 3비트의 그룹으로 나누어 처리함으로써, 덧셈의 회수를 1/2로 감소시켜 곱셈의 속도를 향상시키는 4-radix modified Booth 알고리즘을 이용한다. 본 알고리즘에서는 표 1과 같이 곱셈수 3비트 그룹 입력에 대해 N, S, Z로 엔코딩한다. 이때 신호 Z는 부분곱(partial product)을 0으로 만들고 S는 좌 shift를 하고, N은 2의 보수를 만드는데 사용된다.

3. 본 곱셈기의 구조 및 회로 설계

(1) 구조 및 동작

본 곱셈기는 그림 2와 같이 3단의 마이크로파이프라인 구조로 구성되며, 크게 데이터패스부와 제어부(점선 내 부분)로 나누어진다. 데이터패스부는 8개의 부분곱을 생성하는 Booth logic, 이 부분곱들을 덧셈하는 adder array, 그리고 최종 sum과 carry를 덧셈하는 CPA의 3부분으로 구성되어 곱셈연산을 수행한다.

Booth logic은 곱셈수에 대해 표 1의 N, S, Z 제어신호를 발생하는 Booth encoder와 이 제어신호를 입력으로 하여 피곱셈수에 대해 표 1의 연산처리를 행하는 Booth mux로 나누어진다. adder array에서는 modified 4-2 counter, full adder, half adder를 사용하여 Booth logic에서 발생된 부분곱에 대해 최종 sum과 carry를 발생한다. CPA에서는 최종 sum과 carry의 가산 연산을 수행한다. 제어부는 외부 모듈에서 제어신호를 받아들여 데이터패스의 각 단의 latch에 필요한 제어신호와 외부 모듈과의 인터페이스를 위한 신호를 발생하기 위해 C-gate를 사용한다.

마이크로파이프라인구조에 기초한 본 곱셈기의 동작은 다음과 같다.

① 안정된 데이터(곱셈수, 피곱셈수)가 버스 Din에 실리고 곱셈요청신호인 Rin이 천이하면, 래치에 데이터가 capture되면서 곱셈이 시작된다. capture가 완

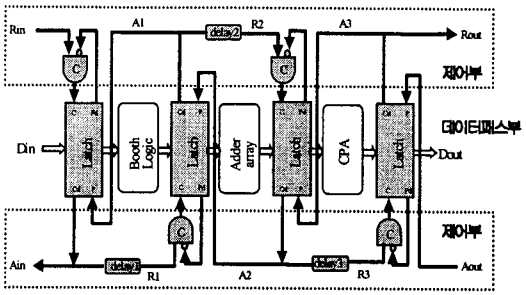


그림 2 마이크로파이프라인 구조의 곱셈기

료되면 Ain이 천이하여 데이터를 받았음을 전단에 알려, 전단이 새로운 데이터를 받을 수 있음을 알린다.

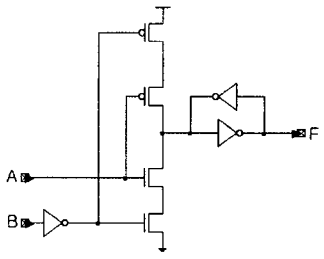
② 래치된 두 연산수는 Booth logic에서 8개의 부분합을 발생한다. Booth logic에서의 지연시간과 전달지연을 포함하는 값보다 큰 지연(delay1)후에 앞단으로의 R1이 천이한다.

③ R1이 천이되어 부분곱들이 래치에 저장되면 adder array에서 일괄 덧셈을 수행하여 최종 sum과 carry를 발생한다.

④ R2가 천이하면 CPA에서 sum과 carry의 최종 덧셈이 수행되어 곱셈연산이 종결된다. delay3후에 곱셈연산이 끝났음을 Rout신호를 통해 외부에 전달한다.

(2) 회로 설계

본 곱셈기는 2상 프로토콜을 실현하고자 그림 3의 C-gate와 그림 4의 CPlatch(Capture-Pass latch)를 사용한다. C-gate의 truth table은 그림 3(b)에 보인다. 한 쪽 입력이 반전을 가지는 구조로 입력이 서로 다를 때 0 또는 1이 되며 그 외의 경우는 전 값을 유지한다. CPlatch는 입력 게이트의



A	B	F_{n+}
0	0	F_n
0	1	0
1	0	1
1	1	F_n

(a) schemtic (b) truth table

그림 3 C-gate

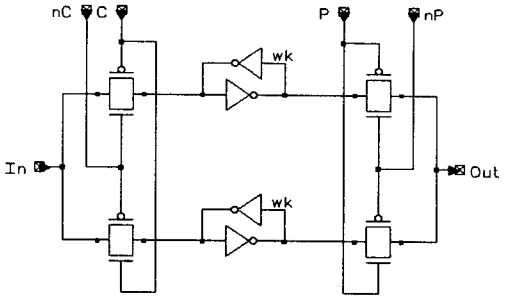


그림 4 capture-pass latch

캐패시턴스가 큰 단점이 있지만 이 래치를 사용함으로써 제어부의 회로를 그림 2처럼 C-gate를 사용하여 간소화 할 수 있다. 초기 CPlatch의 transparent한 상태에서 유효한 데이터가 오면 C가 천이하여 전단과의 연결을 끊고 weak inverter latch에 데이터를 저장한다. 이 래치의 출력에 이어지는 앞단은 이 데이터를 가지고 연산을 수행한다. 연산이 완료되어 P 신호가 천이하면 다시 transparent한 상태가 되어 다음 capture 신호를 기다린다. 이처럼 마이크로파이프라인은 래치를 통해 유효한 데이터의 흐름을 결정하기 때문에 hazard를 걸러낼 수 있다는 큰 장점이 있다.

본 곱셈기는 8개의 부분곱의 일괄 덧셈을 수행시 sign extension에 의한 하드웨어 낭비를 막고자, 부분곱들의 MSB를 반전시키고, 마지막 부분곱을 제외하고 MSB의 앞 비트에 '1'을 부가하는 부호비트보수화 방법을 사용한다(이때 첫번째 부분곱의 MSB 비트 위치에도 "1"을 부가해야 한다.). 부호비트보수화 방법을 구현하고자 Booth logic에서 MSB가 반전된 8개의 부분곱을 발생하고 adder array에서는 8개의 부분곱과 부호비트보수화에 필요한 '1'의 덧셈을 수행한다. 또한 CPA에서는 32비트 carry skip adder를 상위비트로부터 속도향상을 꾀하기 위해 2-8-7-6-5-4로 분할하여 설계한다.

4. 설계 결과

곱셈기의 각 회로에 대해 Hspice를 이용하여 회로 시뮬레이션을 하였으며, Cadence 툴을 사용하여 layout을 하였다. 그림 5은 그림 2에서 로직 블록을 제외하고 래치와 지연소자(지연은 13 ns이용)만을 사용하여(즉 FIFO회로) 마이크로파이프라인의 동작을 spice 시뮬레이션한 결과이다. 마이크로파이프라인 구조 각 단의 request 신호(Rin, R1, R2, R3)가 rising edge와 falling edge에서 천이함에 따라 데이터가 전송되는 것으로 2상 프로토콜을 실현함을 보여준다.

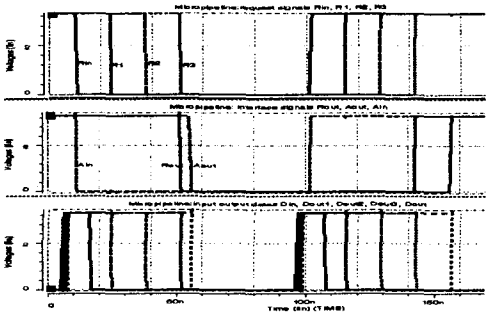


그림 5 마이크로파이프라인의 동작 시뮬레이션

표 2 각 로직 블럭의 지연시간

Blocks	Delay(ns)
Booth's logic	2.5
adder array	2.8
CPA	12.7

표 2는 래치를 포함한 데이터 패스부의 각 블록에 대한 post-layout 시뮬레이션 한 지연 시간에 20%의 여유도를 포함한 지연값을 나타낸다. 각 단의 지연시간은 latch에서 데이터가 capture된 후로부터 로직 단에서 연산을 거친 후, 다음 latch에서 데이터를 다시 capture할 때까지의 시간이다. 이 때의 곱셈기의 동작속도는 총 지연시간이 18ns로 되어 56MHz로 계산되었다. 마이크로파이프라인 단에 삽입되는 지연소자의 값으로는 표 2 의 각 블럭의 지연 값을 사용했다. 또한 새로이 CPA의 단수를 분할한 결과 pre-layout시뮬레이션 상에서 [5]의 6-7-6-5-4-4로 분할한 것보다 5%의 속도향상을 얻었다. 본 곱셈기는 2중 metal, 2중 poly의 0.65㎛ 공정을 이용한 결과 12,074개의 트랜지스터가 사용되었고, 면적은 1.4×1.8㎟로 구현되었다. 전체 레이아웃도는 그림 6 와 같다.

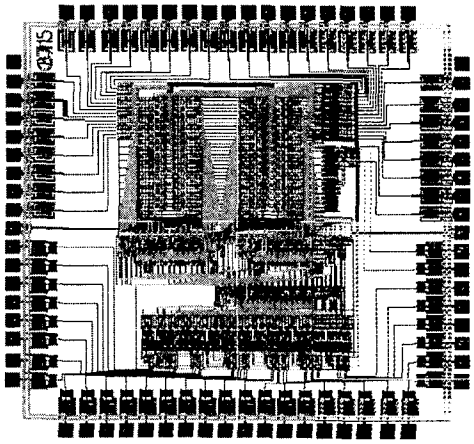


그림 6 전체 레이아웃도

5. 결 론

본 논문에서는 마이크로파이프라인구조와 Booth 알고리즘을 이용하여 16bit 비동기 곱셈기를 설계하였다. 설계결과 1.4×1.8㎟의 면적으로 구현되었으며, 56MHz의 동작결과를 얻었다. 마이크로파이프라인에 삽입되는 지연소자의 값으로 각 로직 블록의 최악의 지연을 가정한 값으로 했음에도 불구하고 고속의 성능을 얻었다. 따라서 본 곱셈기는 비동기 곱셈기 자체로의 이용 뿐 만 아니라, 비동기 시스템 설계의 고성능 곱셈 연산의 한 회로로 이용 될 수 있을 것이다

[참고문헌]

[1] Al Davis and Steven M. Nowick, "An introduction to asynchronous circuit design," Technical Report UUCS-97-013, Department of Computer Science, University of Utah, Sep. 1997.

[2] Ivan E. Sutherand, "Micropipelines," Communications of the ACM, vol. 32, no. 6, pp. 720-738, June 1988.

[3] P. Pay and J.V. Woods, "Investigation into micropipeline latch design styles," IEEE Transactions on Very Large Scale Integration Systems, vol. 3, no. 2, pp. 264-272, June 1995.

[4] Tin-Chak-Johnson Pang et al., "Self-timed Booth's multiplier," 2nd International Conference on ASIC, pp. 280-283, 1996.

[5] 김학윤, 이유진, 장미숙, 최호용, "비동기 시스템용 고성능 16비트 승산기 설계", 대한전자공학회 1999년도 추계종합학술대회 논문집, 제22권 제2호, pp. 356-359, 1999년 11월.