論文2000-37SD-11-8

차세대 ASIC 라이브러리를 위한 고속 저전력 조건 선택 덧셈기/뺄셈기의 설계

(Design of a Low Power High Speed Conditional Select Adder/Subtracter for Next Generation ASIC Library)

曺 基 善*, 宋 敏 圭*

(Ki Seon Cho and Min Kyu Song)

요 약

본 논문에서는 DSP에서 필수적인 고속 저 전력 조건 선택 덧셈기/뺄셈기의 마크로 셀 라이브러리를 설계, 구축하였다. 덧셈기의 Carry전달 지연 시간을 최소로 하기 위한 CLA 기법과 연산 가능한 모든 결과 값을 미리 계산한 후 선택하는 조건 선택 기법을 적용하였다. 또한 이러한 설계방법이 8비트에서 64비트까지 자동 생성될 수 있도록 전용 프로그램을 작성하고 셀 기반 설계기법을 도입하여 Auto P&R Tool과 연계하여 자동으로 레이아웃이 가능하도록 하였다. 제안된 덧셈기/뺄셈기는 0.25㎛, 1-Poly, 5-Metal, N-well CMOS 공정을 사용하여 제작되었으며, 2.5V 단일 공급전압에서 지연시간, 소모 전력을 측정하였다. 측정결과 32 비트 덧셈기/뺄셈기의 경우 3.43ns의 지연시간과 42.8 ㎼/㎒의 전력소비를 나타내었다.

Abstract

As multimedia applications become popular, computers increasingly require high-speed DSP for 3-DIM computer graphic. In this Paper, a Macro-cell Library of conditional select adder/subtracter is proposed for DSP within high speed and low power consumption. Using, this design method, we are able to obtain an auto generation of the adder or(and) subtracter from 8-bit to 64-bit. The proposed adder/subtracter has been fabricated with a 0.25㎛, single-poly, five-metal, N-well CMOS technology. From the experimental results, delay time is 3.43ns, and the power consumption is 42.8㎼/㎒ at the input frequency of 50㎒, at 2.5V single power supply, in case of the 32-bit adder/subtracter.

Ⅰ. 서 론

최근의 멀티미디의 경향은 컴퓨터에서 3차원 그래픽의 빠른 처리 속도를 요구한다. 따라서 디지털 신호 프로세서(DSP) 내에서 연산기능을 수행하는 경우, 고속 multi-bit 덧셈이나 곱셈 등을 위한 회로설계 기술의 중요성이 계속 증가되고 있다. 이 중 덧셈 연산은 마이크로 프로세서, ALU, 곱셈기 등의 필수적인 기능 중의 하나이며 시스템의 속도 성능에 큰 영향을 미치기 때문에 덧셈 연산의 속도를 향상시키기 위한 연구가 계속되고 있으며 덧셈기를 포함한 시스템을 단일 칩으로 설계하기 위한 설계자동화 기술 역시 크게 발전하고 있다.[1]

설계 자동화를 위한 구체적인 설계 방법에는 크게 두 가지가 있다. 첫 번째는 full-custom 설계로 bottom-up 방식이며, 다른 하나는 VHDL, Verilog-HDL과 같은 언어 기반의 설계로 top-down 방식이다. bottom-up 방식은 성능은 우수하나 개발 기간 및 비용이 높고, top-down 방법은 개발 기간 및 비용은 낮으나 성능이 상대적으로 떨어지는 단점이 있다. 그러나 어떠한 방식을 통하여 회로를 설계하더라도 지

* 正會員, 東國大學校 半導體科學科

(Dongguk Univ., Dept. of Semiconductor Science)

※ 본 논문은 한국과학재단 산학협력연구과제(과제번호 : 1999-30200-004-2)의 지원에 의해 수행되었습니다. 지원에 감사드립니다.

接受日字:1999年12月17日, 수정완료일:2000年9月1日

연시간, 면적, 및 전력소모와 같은 설계 지수가 시스템 요구사항을 만족 하여야한다.[2] 최근에는 생산공정의 VHDS(Very High Deep Submicron)로의 변화, Time-to-Market의 중요성 등으로 인해 거의 대부분의 시스템 설계가 top-down 방식으로 이루어지고 있다. top-down 방식은 이미 논한 바와 같이 성능이 bottom-up 방식보다는 떨어진다. 이러한 단점을 보완하기 위해 ASIC 부분의 주요 마크로 셀 라이브러리를 구축하여 개발 기간 및 비용을 절감하고 bottom-up 방식으로 설계할 때와 동일한 성능을 갖도록 개발하고 있다.[3,4] 그러나 기존의 마크로 셀 라이브러리는 Bottom-up 방식으로 각각 설계하고 있으므로 위에 논한 환경 변화에 대해 대응하기에는 부적합하다. 따라서 최근에는 bottom-up으로 설계된 기본 회로를 셀 베이스 설계 방식으로 전환한 후, 각 셀을 조합하는 전용 프로그램을 작성하고 자동화 툴을 이용하여 마크로 셀을 완성하는 새로운 방식이 사용되고 있다. 본 논문에서는 이와 같은 경향에 따라, 먼저 CMOS 로직과 패스트랜지스터 로직을 적절히 혼합하여 대표적인 마크로 셀인 덧셈기/뺄셈기의 새로운 구조를 제안하고, 이를 bottom-up 방식으로 설계하는 셀 베이스 설계 방식을 도입하고자 한다. 또한 별도의 전용 프로그램을 작성하고 Auto P&R 프로그램과 연계하여 8비트에서 64비트까지 자동으로 레이아웃이 생성될 수 있도록 하는 새로운 마크로 셀 라이브러리를 구축하고자 한다.

본 연구에서 제안한 CSAS(Conditional Select Adder/Subtracter) 덧셈기/뺄셈기는 CLA, CS기법 및 CSA기법[5]을 혼용하여 carry 발생블록과 sum 발생블록을 분리시켜 동시에 연산을 수행토록 한 고속 덧셈기이다. 본 논문의 순서는 다음과 같다. Ⅱ장에서는 제안된 64비트 선택 조합 덧셈기의 전체 구조를 나타냈으며, Ⅲ장에서는 제안하는 덧셈기의 세부 블록 회로 설계, 동작 원리 및 마크로 셀 라이브러리 구축을 설명하였고, Ⅳ장에서는 실험결과를 논하였으며, 마지막으로 Ⅴ장에서는 결론에 대해 기술하였다.

Ⅱ. 전체구조

본 연구에서 제안한 CSAS의 64 비트 전체 구조를 그림 1에 나타냈다. 설계된 64-bit CSAS는 그림에서처럼 8-bit을 기반으로 한 8개의 Conditional Selcet Adder Block(이하, CSAB)과 각 CSAB의 carry를 결정해주는 Block Carry Generation Block(이하, BCGB)로 구성되어 있다. 각각의 CSAB는 덧셈과 뺄셈을 결정하는 Add/Sub Block(이하, ASB), 입력 된 값을 분석하는 Pre carry & sum Generation Block (이하, PGB)과 carry값에 따른 두 상태의 합을 생성하는 Sum Generation Block(이하, SGB), 그리고 각 CSA모듈의 carry를 생성하는 Carry Generation Block(이하, CGB)으로 구성되어졌다. 각 블록은 PTL[4,6,7]을 이용한 Multiplexer(이하, MUX)를 기본으로 설계하였고, 저 전력과 고속 동작을 위해 MUX의 구동능력에 맞게 LRB[8]를 적절히 사용하였다.

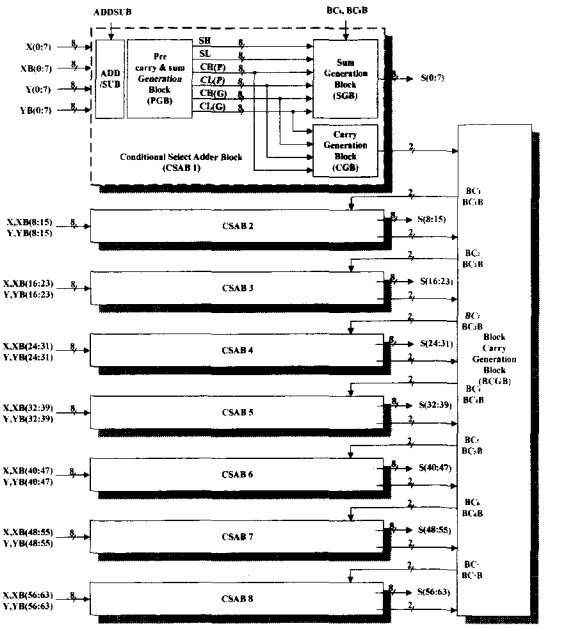


그림 1. 제안된 64-bit CSAS의 전체구조
Fig. 1. Architecture of the proposed 64-bit CSAS.

이러한 구조를 바탕으로 하여 설계된 64-bit CSAS의 동작원리를 살펴보면, ASB에서 선택된 신호는 PGB에 전달되며, PGB에서 분석된 신호(Conditional Sum, Propagation & Generation Carry)들은 SGB와 CGB로 보내지며, SGB는 carry 유무에 따른 두 조건의 합을 생성해 놓으며 이와 동시에 CGB는 각 CSAB의 Carry상태를 BCGB에 보내게 된다. BCGB는 각 CSAB에 의해 발생된 carry를 조합하여 carry 유무를 다음 단의 CSAB에 feed-back 함으로써 미리 계산되어진 SGB의 sum을 선택하는 방법이다.

Ⅲ. 블록 설계

1. Multiplexer의 설계

설계된 64-bit CSAS는 MUX를 기본으로 하여 구성되었으며 각 MUX의 회로와 symbol을 그림 2에 나타냈다. 이와 같이 PTL을 기본으로 한 MUX들은 용도(구동능력, rail의 수)에 따라 4가지 형태로 구분되며, 필요에 따라 선택적으로 사용하였다. 이 MUX들은 입력 조합에 따라 다양한 논리기능(예, AND, OR...)을 수행한다. MDL과 MSL의 SPICE 시뮬레이션 결과 MDL이 MSL보다 fan-out의 변화에 대한 지연시간이 더 짧다.[8]

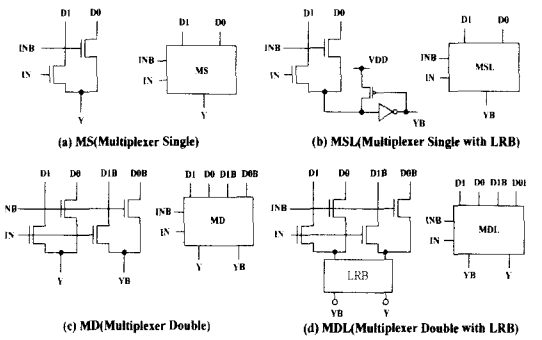


그림 2. 각 MUX의 회로와 symbol
Fig. 2. Circuit and symbol of each MUXs.

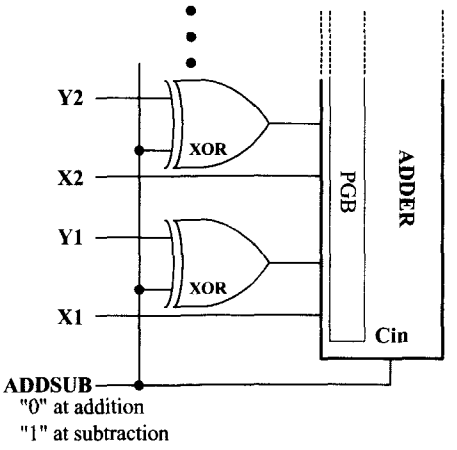


그림 3. 뺄셈연산을 위한 입력단 회로도
Fig. 3. Block Diagram of the input stage for addition and subtraction.

2. Adder/Subtracter Block의 설계

가산될 두 입력과 덧셈/뺄셈 기능을 결정하는 신호인 "ADDSUB"에 따라 입력 신호는 식 (1)에서와 같이 Y신호가 그대로 전달되거나(덧셈의 경우) 반전되어 전달된다(뺄셈의 경우). 이를 회로로 구현하면 그림 3과 같이 XOR와 같다.

$$\begin{aligned} SUM &= X_j + Y_j \quad (at\ Addition,\ cin=0) \\ SUM &= X_j + \overline{Y_j} \quad (at\ Subtraction,\ cin=1) \end{aligned} \tag{1}$$

3. Pre-carry & Sum Generation Block의 설계

가산될 두 입력으로부터 sum 및 carry의 발생경향을 분석하는 블록으로써 발생경향을 다음과 같은 식 (2)로 나타낼 수 있다.

$$\begin{aligned} SH &= X_j \cdot Y_j + \overline{X_j} \cdot \overline{Y_j} && (XNOR) \\ SL &= \overline{X_j} \cdot Y_j + X_j \cdot \overline{Y_j} && (XOR) \\ CP &= (X_j + Y_j) \cdot C_{j-1} = X_j + Y_j && (OR) \\ CG &= (X_j \cdot Y_j) + C_{j-1} = X_j \cdot Y_j && (AND) \end{aligned} \tag{2}$$

여기서 CP는 Carry Propagation Signal, CG는 Carry Generation Signal을 의미하다. 즉, j bit의 자리에서 carry가 발생될 수 도 있고 j-1 bit에서 발생된 carry가 j bit에 전달될 수 도 있다. 이 관계를 표 1에 정리하였다.

표 1. j bit에서의 sum 및 carry의 발생분석
Table 1. Analysis of the carry & sum generation at j-bit.

	SUMj	CARRYj	
Generation	SL (XOR)	CG (AND)	CGB (NAND)
Propagation	SH (XNOR)	CP (OR)	CPB (NOR)

따라서, sum 및 carry의 발생분석을 위한 논리기능들(XOR, XNOR, AND, NAND, OR, NOR)을 회로로 구현하기 위해 앞서 언급된 MUX를 사용하였다. PGB의 구성을 그림 4에 나타냈으며 각 신호들의 논리기능을 표시하였으며 PGB의 출력 신호는 높은 구동력이 필요하므로 MDL을 사용하였다. 구성된 PGB로부터 두 입력 X, Y에 대해 분석된 신호들은 각각 SGB와 CGB로 보내져 연산을 수행하게 된다.

4. Sum Generation Block의 설계

8-bit SGB는 그림 5에 나타낸 것과 같다. 즉, 각

SGB에서는 PGB에서 발생된 각 신호를 이용하여 전 단의 8-bit CSAB에서 carry가 전달될 때와 그렇지 않을 때를 가정하여 PGB를 제외하고 7MUX만에 두 가지 경우의 sum을 계산해 놓게 된다. 단위 블록들 간의 sum 계산은 Carry-Select Adder(CS)기법을 포함한 RCA를 적용하였다. 가장 느린 RCA를 사용한 이유는 제안한 CSAS에서의 critical path는 carry를 계산하는 CGB이기 때문이다. 이렇게 계산된 두 가지의 sum은 Block Carry를 발생하는 BCGB에 의해 Carry feed-back이 되면 선택적으로 하나의 sum만이 출력되도록 구성되어 있다.

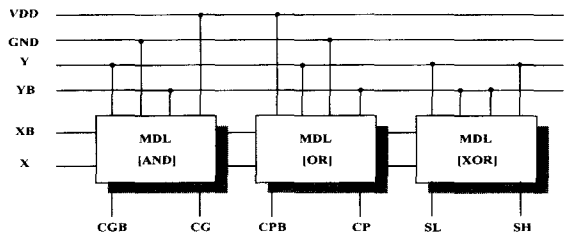


그림 4. Pre carry & sum Generation Block
Fig. 4. Block of the Pre carry & sum Generation.

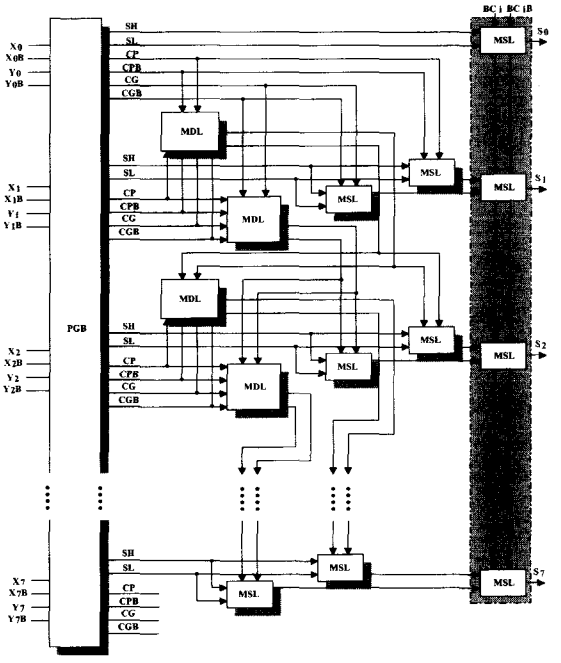


그림 5. 8-bit Sum Generation Block 구성
Fig. 5. Block of the 8bit Sum Generation.

5. Carry Generation Block의 설계

CGB는 SGB의 Sum이 연산되는 동안 8-bit CSAB의 carry를 연산하게 되며 CGB의 구성을 그림 6에 나타냈다. PGB에서 발생된 신호를 이용하여 j bit에서의 carry와 j-1 bit에서의 carry를 비교하여 j bit에서의 carry 발생여부를 결정하며 같은 방법으로 계산된 다음의 2-bit의 결과에 영향을 미쳐 carry를 결정해 가는 기법이다. 이렇게 하면 3 MUX만으로 8-bit 내의 carry의 유무가 계산되어질 수 있다. 이 결과가 BCGB에 전달되어 단위블록의 carry가 결정된다.

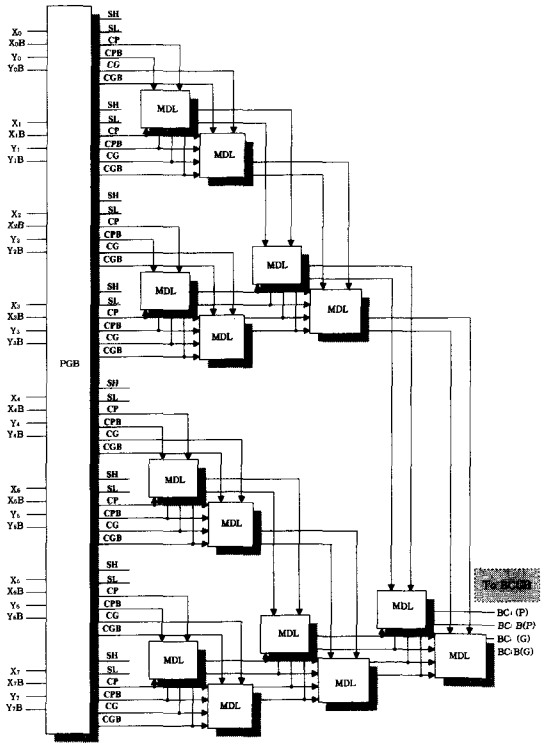


그림 6. 8-bit Carry Generation Block 구성
Fig. 6. Block of the 8bit Carry Generation.

6. Block Carry Generation Block의 설계

BCGB는 CGB와 동일한 기법으로 4 MUX만에 각 단위 블록의 SGB에 feed-back 되는 carry를 발생시키며 그 구성을 그림 7에 나타냈다. 이와 같은 carry 계산방법은 기존에 제안된 CCS(Conditional Carry Select)[9] carry 발생 구조를 개선한 것으로 기존의 5 MUX만에 계산되어지던 block carry가 4 MUX만에 계산되어진다. 또한 CMOS로 구현된[10] 경우 6 MUX의 지연시간이 걸리므로 설계된 구조가 2 MUX 정도

지연시간이 감소되었음을 알 수 있다.

7. CSAS의 지연시간 분석

설계된 64-bit CSAS(덧셈기/뺄셈기)는 전체 지연시간이 10 MUX 뿐이므로 기존에 제안된 덧셈기(CCS)[9]의 11 MUX보다 1 MUX 이상 지연시간이 감소한다. 또한 설계된 CSAS는 뺄셈기를 포함하고 LRB의 사용으로 다음 블록을 효과적으로 구동하므로 실제 지연시간은 더 감소하게 된다. 그림 8에는 제안하는 CSAS의 MUX단위 분석도를 나타내었다.

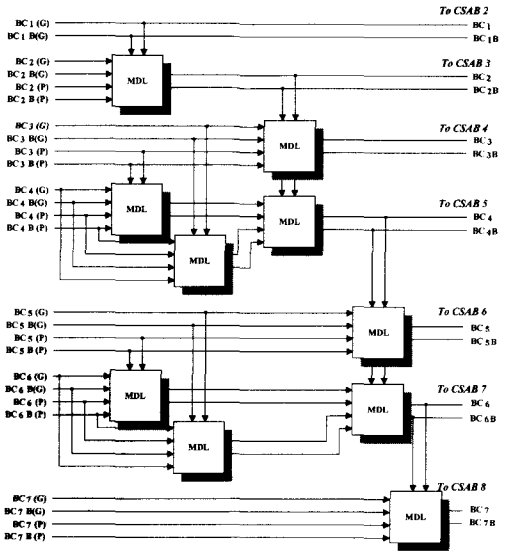


그림 7. Block Carry Generation Block
Fig. 7. Block of the Block Carry Generation.

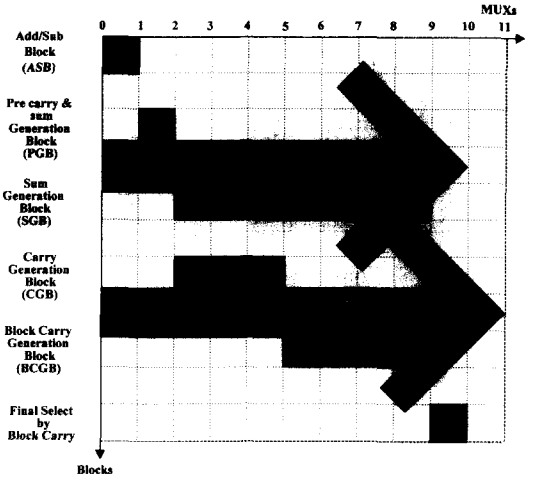


그림 8. 설계된 64-bit CSAS의 각 블럭 지연시간 분석
Fig. 8. Delay analysis of the designed 64-bit CSAS.

8. Macro-cell Library의 개발

위와 같은 기본 알고리즘을 이용하여 덧셈기/뺄셈기 Macro-cell Library를 개발하였다. 제안한 CSAS 라이브러리는 generator 프로그램에 의해 다음과 같이 구현되었다. 첫째, n-bit(8~64) 덧셈기 또는 덧셈기/뺄셈기, 둘째, 두 양수 또는 두 음수의 덧셈 시 overflow 검사, 셋째, schematic netlist와 layout의 자동생성, 넷째, 출력 구동 드라이버 선택가능 등이다.

이와 같은 기능을 만족하기 위해 generator가 Unix에서 C언어로 프로그래밍 되었다. Datapath 구성을 위해 각 회로들은 셀 단위로 재구성됐으며 기본셀로 나누어져 전체적인 architecture를 구성하게 된다. 이와 같이 나누어진 기본셀로 구성된 구성도를 32-bit을 예로 들어 그림 9에 나타냈으며 각 기본셀들간의 신호경로를 부분적으로 표시했다. Placement를 지정하고 각 기본셀의 port를 routing하는 역할은 generator로부터 수행된다. 기본 셀들은 그림 10에 나타내었다.

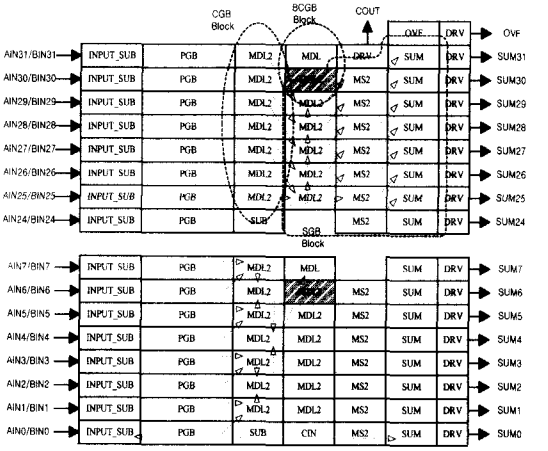


그림 9. 제안된 CSAS의 Datapath Architecture(32-bit)
Fig. 9. Datapath Architecture of the CSAS(32-bit).

먼저 입력단의 나누어진 기본셀을 보면, 조건(덧셈/뺄셈)에 따른 입력 선택을 위해 INPUT_SUB 및 INPUT_NOM, 입력신호를 분석하는 PGB, 두개의 MDL로 구성된 MDL2, 두개의 MS로 구성된 MS2, 그리고 MSL로 각각 나누어졌다. 마지막으로 추가적인 기능을 위한 나누어진 SUBEX, OVF, 그리고 출력단을 위한 DRV 등으로 각각 나누어진다. 이렇게 CSAS는 전체 10개의 기본셀로 나누어져 제안하는 CSAS를 구성하게 된다.

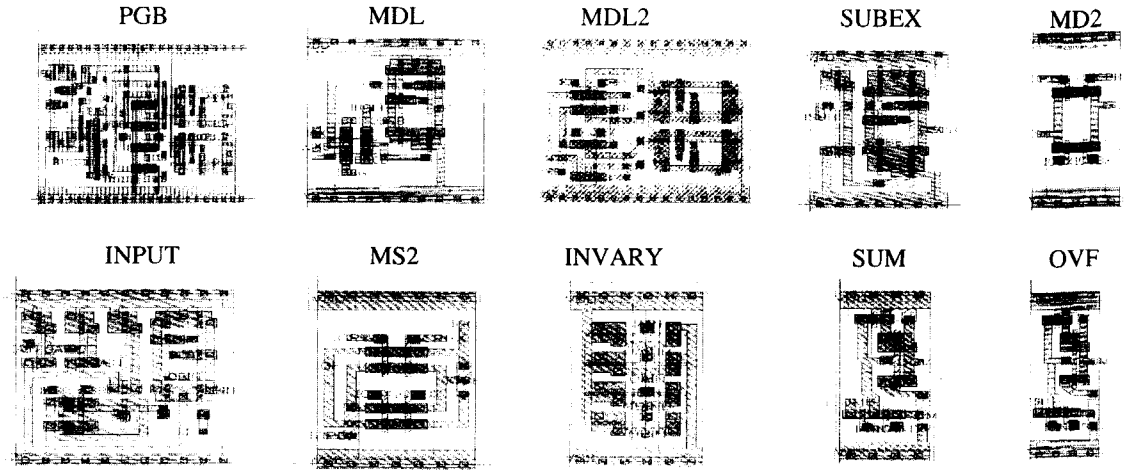


그림 10. 기본셀들의 나열(높이 12㎛)
Fig.10. Lists of primitive cell.

Ⅳ. 실험 결과

제안된 CSAS(덧셈기/뺄셈기)는 0.25㎛, 1-Poly, 5-Metal, N-well CMOS 공정을 사용하여 제작되었으며, 2.5V 단일 공급전압에서 지연시간, 소모 전력을 측정하였다. 먼저 SPICE 시뮬레이션 결과를 표 2에 나타내었다.

표 2. 제안된 덧셈기/뺄셈기의 시뮬레이션 결과
Table 2. Simulation Results of the proposed CSAS.

	Delay[ns]	Power[㎼/㎒]	P·D[fJ]
8 bit	1.74	6.75	11.75
16 bit	1.97	14.56	28.68
32 bit	2.22	31.51	69.95
64 bit	2.50	55.76	139.40

<VDD = 2.5V→Slop=0.170 CL=0.022 : at SUBTRACTION condition>

지연시간은 Subtraction 신호에 의해 입력된 Y신호가 반전되어 계산된 후 마지막 비트의 sum 출력을 측정한 결과이다. 8-bit의 결과가 다른 bit보다 지연시간이 특별히 낮지 못한 이유는 앞에서 언급한 것 같이 8-bit 내의 SUM 계산 방법을 RCA의 구조를 채택했기 때문이다. 이와 같은 결과를 토대로 32-bit CSAS(덧셈기/뺄셈기)를 사용하여 칩으로 구현하였다. 이를 그림 11에 나타내었다.

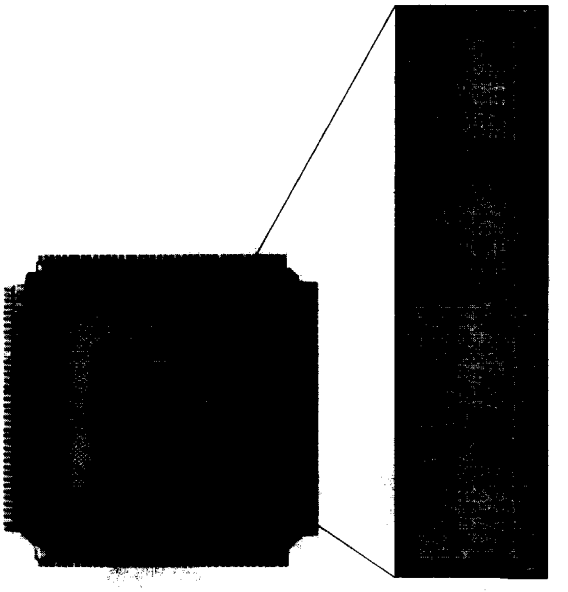


그림 11. 전체 Chip 사진 및 Layout
Fig. 11. Photograph of the full chip and layout.

이의 성능을 표 4에 나타내었으며 IMS(Integrated Measurement System)사의 XL DC Matrix와 Logic Master를 사용하여 Function 만족여부를 확인을 하였다. 슈뮤 플롯을 그림 12에, 오실로스코프의 측정 파형을 그림 13에 나타내었다.

표 4. 제안된 32 비트 덧셈기/뺄셈기의 성능
Table 4. Performance of the proposed 32-bit CSAS.

Process	0.25㎛ 1-Poly, 5-Metal, N-well CMOS
Chip	160 PIN, 8000㎛×8000㎛
Logic size	98.8㎛×398.0㎛
Power Supply	Single, 2.5V
Input/Output/Option	32-bit/32-bit/3-bit
Delay Time	3.43ns
Power Consumption(at 50㎒)	42.8㎼

<VDD =2.5V→Slop=0.170 CL=0.022 : at SUBTRACTION condition>

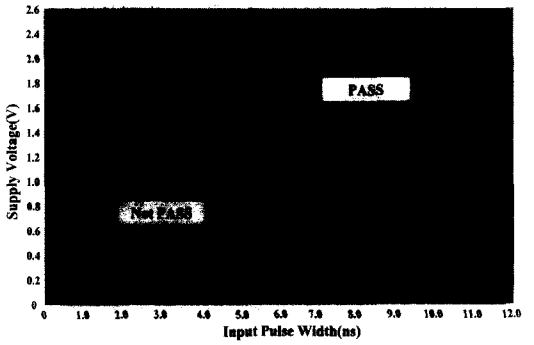


그림 12. 공급전압과 동작 주파수의 슈무 플롯
Fig. 12. Shmoo plots(supply voltage vs input pulse width).

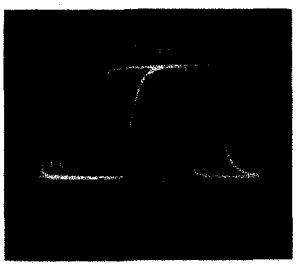


그림 13. 제안된 CSAS(덧셈기/뺄셈기)의 측정 파형
Fig. 13. Measured waveform of the proposed CSAS.

V. 결 론

본 논문에서는 덧셈기/뺄셈기, Cin, Cout, Over-flow등의 옵션을 모두 포함한 조건 선택 덧셈기/뺄셈기(CSAS;Conditional Select Adder/Subtracter)를 프로그램에 의해서 마크로 셀 라이브러리를 구축하였으며 Auto P&R Tool에 의해 8~64-bit 까지 자유롭게 자동 생성될 수 있도록 하였다. 이를 칩으로 구현하여 측정한 결과 32-bit CSAS의 경우 3.42ns의 지연시간과 42.8 ㎼/㎒의 저 전력소모를[11] 나타내었다.

참 고 문 헌

[1] Gensuke. Goto et.al., "A4.1-nsCompact 54×54-b Multiplier Utilizing Sign-Select Booth Encoders," *IEEE J. of Solid-state Circuits*, vol.32, no. 11, pp. 1676-1681, Nov. 1997.

[2] Neil H. E. weste, Kamran eshraghian, "*Principles of CMOS VLSI Design*," 2nd edition, Addison-wesley pub. co., pp. 381-563.

[3] Jim Lipman, "Not just your basic ASIC libraries," *in EDN Magazine*, pp. 53-60, Volumn 4, June 5, 1997.

[4] F. Moraes, et.al., "Pre-layout performance prediction for automatic macro-cell synthesis," *IEEE International Symposium on Circuits and Systems*, pp. 814-817, Volume 4, Mar 12, 1996.

[5] Abdellatif Bellaouar, et. al., *"Low-Power Digital VLSI Design Circuits and Systems,"* Kluwer Academic Publishers, pp. 409-450, Sep., 1995.

[6] T. Sakurai, et. al., "Low-Power Circuit Design for Multimedia VLSI," *in Proc. IEEE ICVC'95*, pp. 37-42, Oct., 1995.

[7] K.Yano, et. al., "Top-Down Pass-Transistor Logic Design," *IEEE J. of Solid-State Circuits*, Vol. 31, no. 6, pp. 792-803, Jun., 1996.

[8] M. Song, et. al., "Design Methodology for High Speed and Low Power Digital Circuits with Energy Economized Pass-transistor Logic (EEPL)," *in Proc. IEEE ESSCIRC '96*, pp.120-123, Sep., 1996.

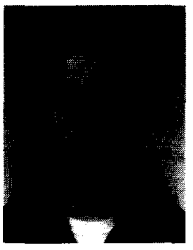
[9] N. Ohkubo, et. al., "A 4.4ns CMOS 54×54-b Multiplier Using Pass-Transistor Multiplexer," *IEEE J. of Solid-State Circuits*,

Vol. 30, no. 3, pp. 251-257, Mar., 1995.

[10] M. Song, et. al., "Power Optimization for Data Compressors of a Parallel Structured 54×54bit Multiplier," *in Proc. IEEE ECCTD'95*, pp. 427-430, Sep., 1995.

[11] Wei Hwang. et,al., "Implementation of a Self-Resetting CMOS 64-bit Parallel Adder with Enhanced Testability *"IEEE J. of Solid-State Circuits*, Vol. 34, no. 8, pp. 1108- 1117, Aug., 1999.

저 자 소 개



曺 基 善(正會員)

1998년 동국대학교 반도체과학과 석사. 1998년~현재 동국대학교 반도체과학과 석사과정 재학중, 주관심분야는 고집적 디지털 신호처리 및 CMOS 혼성모드 회로설계, 저전력 집적시스템설계

宋 敏 圭(終身會員)

서울대학교 전자공학과 학사(1986년), 석사(1998년), 박사(1993년). 1993년~1994년 일본 동경대학교 전자공학과 초빙연구원, 1995년~96년 삼성전자 ASIC 설계팀 선임연구원, 1997년~현재 동국대학교 반도체과학과 조교수, 주관심분야는 CMOS 혼성모드 회로설계, 저전력 집적시스템 설계