論文
3-2-3

광스위칭소자에 기반한 산술논리연산회로의 설계 (Design of An Arithmetic Logic Unit Based on Optical Switching Devices)

박 종 현* 이 원 주** 전 창 호***
(Jong-Hyun Park) (Won-Ju Lee) (Chang-Ho Jeon)

요 약

본 논문에서는 광컴퓨터의 개발에 이용될 수 있는 산술논리연산회로(ALU)를 설계하고 검증한다. 전자회로 기술의 접목이 용이하고 가장 상용화가 잘된 $LiNbO_3$ 광스위칭 소자에 기반한 이 ALU는 산술논리 동작을 실행하는 연산회로, 오퍼런드와 연산결과를 저장하는 메모리 소자 그리고 명령어 선택을 위한 부가회로로 구성되며, 비트 단위 직렬 방식으로 동작하는 것이다. 본 논문에서는 또한 설계한 ALU 회로의 정확성을 검증할 수 있는 시뮬레이터를 구현하고, 일련의 기본 명령어들을 순차적으로 실행하면서 메모리와 누산기에 저장된 값의 단계적 변화를 확인하는 시뮬레이션을 통하여 설계한 ALU가 정확함을 보인다.

ABSTRACT

This paper deals with design and verification of an arithmetic logic unit(ALU) to be used for development of optical computers. The ALU is based on optical switching device, $LiNbO_3$, which is easy to interface with electronic technology and most common in the market. It consists of an arithmetic/logic circuit performing logic operations, memory devices storing operands and the results of operations, and supplementary circuits to select instruction codes, and operates in bit-serial manner. In addition, a simulator is developed for verification of the design, and a set of basic instructions are executed in sequence and step-by-step changes in the accumulator and the memory are examined through simulations, to show that various operations are performed correctly.

1. 서론

컴퓨터의 성능 향상을 위한 노력은 빠르게 발전하는 반도체 제조기술에 의존하여 고성능 프로세서를 개발하는 방향으로 집중되어 왔다. 그러나 반도체 칩의 집적도가 높아지고 프로세서를 구성하는 소자의 수가 늘어남에 따라 소자들 사이의 연결구조가 복잡해지고, 높은 열과 과다한 소비전력, 그리고 전자파간섭 및 혼선 등의 문제점이 발생하고 있다. 이러한 문제점을 해결하기 위한 하나의 방법이 전자소자 대신 광소자를 사용하는 것이다. 광소자는 빛 특유의 병렬성과 고속도 그리고 비간섭 특성을 이용하

* 정회원 : (주)베리텍 선임연구원
** 정회원 : 두원공과대학 인터넷프로그래밍과 조교수
*** 정회원 : 한양대학교 전자컴퓨터공학부 교수

논문접수 : 2002. 1. 17.
심사완료 : 2002. 1. 31.

기 때문에 일반적으로 컴퓨터를 구성하는 전자소자에서 발생하는 문제점들이 발생하지 않는다. 또한, 광소자는 선형적 중첩성(linear superposition)과 2차원성 그리고 짧은 진폭 및 넓은 대역폭을 가지고 있어, 전자기 간섭에 강하고 전자소자에 비해 전력소모가 적으며, 기존의 컴퓨터 구조를 크게 변경하지 않고 적용할 수 있다는 장점이 있다[1].

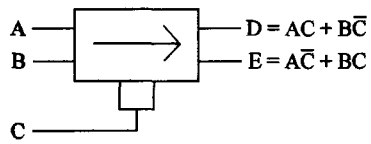
광컴퓨터에 대한 연구는 활발히 진행되고 있으며, 광컴퓨터 개발에 필수적인 이론과 기술들이 많이 개발되었다. n-비트 병렬 가산기의 설계[2]와 효율적인 산술연산[3], 병렬 산술연산 알고리즘[4]은 빛의 병렬성에 착안한 기호대치(symbolic substitution)이론을 적용한 것이다. 그리고 셀방식 논리배열(cellular logic array)은 최소단위의 논리연산회로(cell)를 조합하여 프로세서를 설계하는데 사용한다[5]. 또한, $LiNbO_3$(lithium niobate)와 같은 광스위칭소자를 사용하여 초보적인 비트단위 직렬방식(bit-serial)컴퓨터[6]와, 4-비트 카운터[7], SLM(spatial light modulator)을 사용한 비트 슬라이스 전가산기[8]등이 실험적으로 제작되었다. 광소자와 전자회로를 접목시켜 이진논리 벡터/행렬 승산이 가능한 프로세서로 DOCII(second-generation digital opticalelectronic computer)가 개발되었다[9]. 그리고 DOCII를 확장하여 단위 소비전력이 낮고, 성능이 향상된 HPOC(high performance optoelectronic computing)모듈이 개발되어 있다[10, 11]. 현재의 전자회로 기술과 쉽게 접목할 수 있으며, 상용화 되어 있는 $LiNbO_3$ 광스위칭소자를 사용하여 비트단위 직렬방식(bit-serial)으로 동작하는 SPOC(stored program optical computer) 광컴퓨터도 개발되어 있다[12].

본 논문에서는 $LiNbO_3$ 광스위칭소자를 사용하여 메모리, 멀티플렉서, 승산기 등의 필수적인 논리연산회로를 설계한다. 그리고 이 회로들을 이용하여 간단한 구조의 ALU를 설계하고, 이 ALU의 정확한 동작을 검증할 수 있는 시뮬레이터를 개발한다. 시뮬레이션에서는 11개의 기본적인 논리연산 명령어를 순서적으로 입력하여 실행한다. 각 명령어를 실행한 후 메모리와 누산기에 저장된 값의 변화를 비교함으로써 ALU를 구성하는 회로가 정확하게 동작하는지 검증한다. 본 논문의 제 2장에서는 $LiNbO_3$ 광스위칭소자의 논리적 동작을 설명하고 이를 이용한 기본 논리게이트와 메모리 소자에 대하여 설명한다. 그리고 제 3장에서는 $LiNbO_3$ 광스위칭소자를 이용하여 설계한 ALU의 구성과 그 동작에 대하여 상세히 설명한다. 제 4장에서는 시뮬레이터의 구성과 ALU의 동작 검증 과정을 설명하고 결과를 분석한다. 그리고 제 5장에서 결론을 맺는다.

2. $LiNbO_3$ 기반 회로성분

2.1 $LiNbO_3$ 광스위칭소자 논리게이트

$LiNbO_3$ 광스위칭소자는 2×2 입출력 스위치와 같이 동작하며, 소자의 입출력관계는 [그림 1]과 같다.



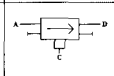
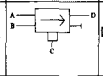
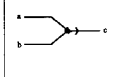
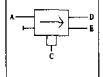
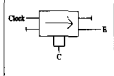
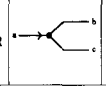
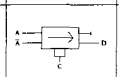
[그림 1] $LiNbO_3$ 광스위칭소자의 논리적 동작

[Fig. 1] Logical Operation of $LiNbO_3$ Optical Switching Device

[그림 1]에서 C단자의 입력이 1이면 A, B단자의 입력이 직진(=)하여 D, E단자에 출력되고, 0이면 교차(×)하여 E, D단자에 출력된다. A, B, C단자에 인가되고 D, E단자에서 생성되는 광신호는 밝으면 1, 어두우면 0로 인식된다. 이러한 동작 특성을 이용하여 구현할 수 있는 기본 논리게이트는 <표 1>과 같다.

<표 1> $LiNbO_3$를 이용한 기본논리게이트

<Table 1> Basic Logic Gate Using Optical Switching Device

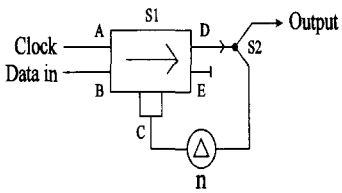
기능	회로구성	관계식	기능	회로구성	관계식
AND		D = AC	2-to-1 Mux		D = AC + BC̄
OR		c = a + b	1-to-2 DeMux		D = AC E = AC̄
NOT		E = C̄	SPLITTER		b = a c = a
XOR		D = A⊕C			

2.2 메모리 소자

$LiNbO_3$ 광스위칭소자와 광섬유를 이용하여 메모리을 구성하면 <그림 2>와 같다. <그림 2>에서 n은 저장할 비트의 수이며 △는 광섬유로 구성되는 지연소자를 의미한다. n은 지연소자의 길이와 클럭 속도에 따라 결정된다. n 비트의 데이터를 저장하기 위한 지연소자의 길이는 식(1)로 구한다.

$$\text{지연소자의 길이} = N_{clock} - S_{delay} \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

식(1)에서 N_{clock}은 n 클럭 동안 빛의 전송거리를 의미하고 S_{delay}는 스위치가 지연되는 시간 동안 빛의 전송거리를 의미한다. 지연소자의 길이를 조절하여 원하는 만큼의 데이터를 저장할 수 있기 때문에 이런 메모리를 DLM(delay line Memory)라 부른다 이것을 이용하여 카운터, 누산기, 레지스터 등을 설계할 수 있다.

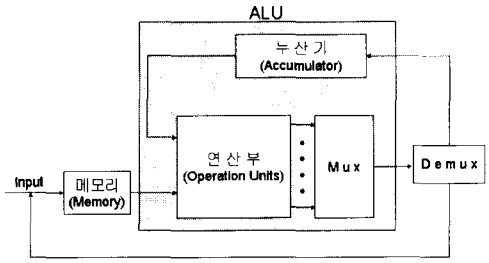


[그림 2] $LiNbO_3$와 광섬유를 사용한 DLM

[Fig. 2] DLM Using $LiNbO_3$ and Optical Fiber Device

3. 논리연산장치(ALU)의 구성

ALU는 [그림 3]과 같이 연산부(operation units), 멀티플렉서(multiplexer), 디멀티플렉서(demultiplexer), 누산기(accumulator) 등으로 구성된다.



[그림 3] ALU 블록 다이어그램

[Fig. 3] ALU Block Diagram

메모리에는 ALU에서 실행될 명령어와 오퍼런드가 저장된다. 본 연구에서는 <표 2>와 같은 기본 명령어와 오퍼런드를 처리하는 수준으로 ALU를 설계한다.

<표 2> 명령어 종류

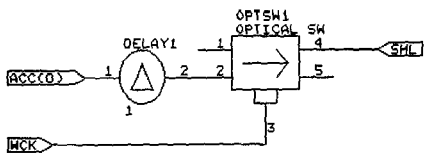
<Table 2> Type of Instruction

OPCODE	동작	표기	입력오퍼런드	결과출력
0	0으로 만듬	CLR	Acc|Mem	Acc|Mem
1	누산기 이동	MOVA		Mem
2	메모리 이동	MOVM		Acc
3	NOT	NOT	Acc	Acc|Mem
4	8비트 승산	MUL	Acc&Mem	Acc|Mem
5	가산	ADD	Acc&Mem	Acc|Mem
6	XOR	XOR	Acc&Mem	Acc|Mem
7	AND	AND	Acc&Mem	Acc|Mem
8	OR	OR	Acc&Mem	Acc|Mem
9	오른쪽 쉬프트	SHR	Acc	Acc|Mem
10	왼쪽 쉬프트	SHL	Acc	Acc|Mem
11	오른쪽 로테이트	ROR	Acc	Acc|Mem
12	왼쪽 로테이트	ROL	Acc	Acc|Mem
13	<없음>			
14	<없음>			

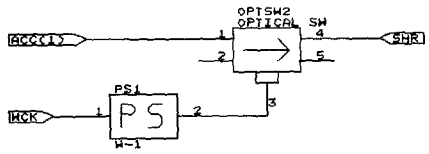
<표 2>에서 OPCODE는 명령어코드이고, Acc와 Mem은 누산기와 Mem를 의미한다. 예를 들어 OPCODE가 5인ADD 명령어가 연산부에서 실행된다면 입력 오퍼런드로 누산기와 메모리의 데이터가 제공된다. 그리고 ADD 명령어의 수행 결과는 Mux와 Demux를 통해 누산기 또는 메모리로 전송된다.

3.1 연산부

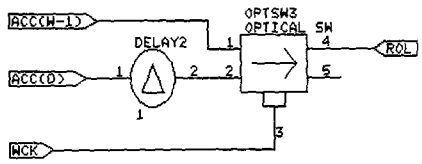
연산부는 전가산기(full adder), 승산기(multiplier), 로테이트(rotate)와 쉬프트(shift) 등의 기본 논리연산 회로를 사용하여 구성한다. 전가산기[6]는 기존에 설계된 것을 사용하기 때문에 회로도를 생략한다. 비트 단위의 직렬 방식으로 동작하는 로테이트와 쉬프트는 MSB나 LSB를 저장하기 위해 임시 메모리가 필요하다. [그림 2]의 DLM을 사용하면 별도의 임시 메모리를 사용하지 않고 [그림 4]와 같이 로테이트와 쉬프트 회로를 구성할 수 있다.



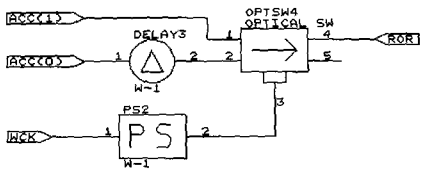
(a) Shift Left(SHL)



(b) Shift Right(SHR)



(c) Rotate Left(ROL)

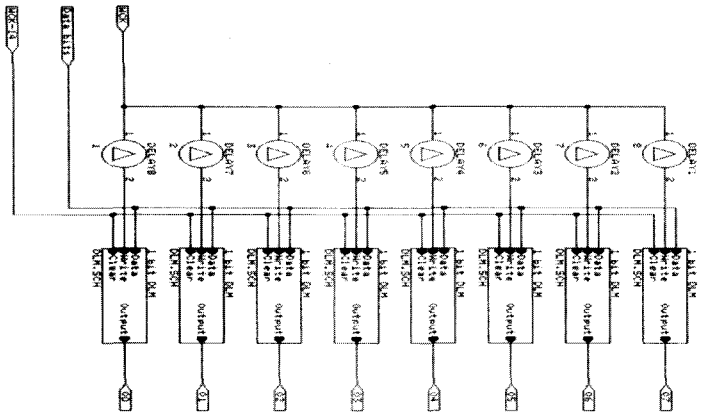


(d) Rotate Right(ROR)

[그림 4] 로테이트와 쉬프트

[Fig. 4] Rotate and Shift

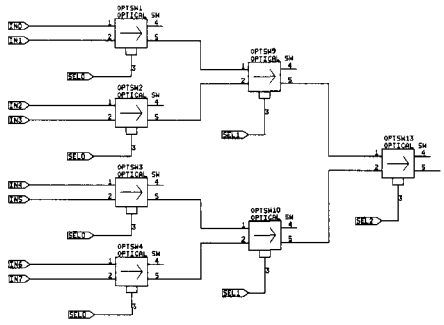
승산기로는 2 개의 오퍼런드가 모두 양수이고 16 비트 데이터를 처리할 수 있는 16 비트 양수 승산기를 구성한다. 16비트 승산기는 8비트 승산기 2개를 사용하여 구성한다. 16비트 승산기는 8비트 승산기와 회로 구성상의 차이는 없지만 사용하는 부품의 수가 8비트 승산기의 2배가 된다. 8비트 승산기는 큐 레지스터 A와 B와 5입력 가산기로 구성한다. 큐 레지스터는 곱셈연산을 수행하는 동안 입력 오퍼런드를 임시로 저장한다. 큐 레지스터 A의 구조는 [그림 5]와 같다. 큐 레지스터 B는 큐 레지스터 A와 비슷하지만 각각의 지연소자(Delay)들의 지연길이가 큐 레지스터 A보다 한 클럭 짧다. 5입력 가산기는 5 개의 입력 중 3 개와 2 개를 각각 더한 후 그 결과를 다시 합산하는 기능을 한다. 5입력 가산기는 기존의 전가산기를 사용하여 구성하며 회로의 동작도 전가산기와 유사하므로 5입력 가산기에 대한 회로도는 생략한다.



[그림 5] 큐 레지스터 A
[Fig. 5] Que register A

3.2 멀티플렉서(Mux)와 디멀티플렉서(Demux)

멀티플렉서는 연산부에서 실행된 여러 명령어의 실행 결과를 입력으로 받아 그 중에 하나를 출력으로 선택한다. 2-to-1 멀티플렉서로 동작하는 $LiNbO_3$ 광스위칭소자를 사용하면 4-to-1, 8-to-1, 16-to-1 멀티플렉서 등을 구성할 수 있다. 본 논문에서는 <표 2>의 16개 명령어를 사용한다. 멀티플렉서는 16개 명령어의 실행 결과들 중에 하나를 선택하여 출력해야 하기 때문에 16-to-1 멀티플렉서를 사용한다. 16-to-1 멀티플렉서는 [그림 6]과 같은 8-to-1 멀티플렉서 2개를 사용하여 구성한다.

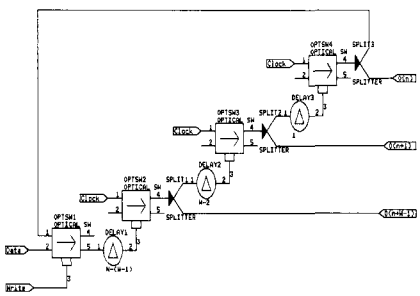


[그림 6] 8-to-1 멀티플렉서
[Fig. 6] 8-to-1 Multiplexer

연산부의 연산 결과는 IN_0~IN_7에 입력되고, 3 비트 선택신호는 SEL_0~SEL_2를 통해 각 스위치로 입력된다. 디멀티플렉서는 멀티플렉서의 출력을 누산기 또는 메모리로 전송하는 기능을 가진 소규모 1-to-2 디멀티플렉서이다. 디멀티플렉서의 제어 신호인 목적지선택(DESTSEL) 값이 0이면 누산기로, 1이면 메모리로 전송된다.

3.3 누산기

디멀티플렉서에서 전송된 연산 결과를 입력으로 받는 누산기는 DLM을 사용하여 [그림 7]과 같이 구성한다.



[그림 7] 누산기
[Fig. 7] Accumulator

본 연구에서는 데이터 저장 용량이 16비트인 누산기를 설계하였기 때문에 1 워드 클럭이 발생하면 DELAY3에는 LSB가 저장되고 DELAY1에는 MSB가 저장된다. 그리고 DELAY2에는 LSB와 MSB를 제외한 14비트가 저장된다.

4. 시뮬레이션 및 결과 분석

설계한 논리연산회로의 동작이 정확한지를 검증하기 위해 시뮬레이터를 개발하여 광스위칭소자로 구성된 개별 게이트 수준의 동작과 소자들간의 연결관계를 고려한 동작을 검증한다.

4.1 시뮬레이터의 구성

시뮬레이터 모듈은 입력과정과 동작 검증과정, 그리고 출력과정으로 구성된다. 입력과정에서는 광소자의 연결 관계와 입력 데이터를 묘사한 텍스트 파일이 작성된다. 시뮬레이션에 필요한 데이터들에 대한 표현형식은 <표 3>과 같다.

<표 3> 시뮬레이션 데이터 표현형식
<Table 3> Simulation Data Expression

항 목	표현형식
시뮬레이션 시간	period 100
입력단의 입력	input in1 0 1 0 1
출력단	output out1

<표 3>에서 시뮬레이션 시간은 시뮬레이션 하는 클럭 수를 의미하며 표현형식에서 'period 100'은 100 클럭 동안 시뮬레이션을 수행한다는 의미이다. 입력단의 입력은 'input' 키워드 뒤에 입력단 이름과 입력 데이터를 순차적으로 나열한다. 입력 데이터는 'period'에서 정한 클럭 수만큼 0과 1로 나타내주어야 한다.

<표 4> 소자들의 입력과정 표현형식
<표 4> Input Process Expression of Devices

소자	소자 그림	표현 형식	소자	소자 그림	표현 형식
스위치		Switch 1 2 3 4 5	OR		Or 1 2 3
분배기		Split 1 2 3	지연 소자		Delay 1 2 10
펄스 재생기		Ps 1 2 10			

시뮬레이터에 적용되는 소자의 표현형식은 <표 4>와 같이 소자의 이름 뒤에 입출력단자를 표기한다. 스위치의 경우 순서를 정확하게 지켜서 입력해야 하지만 OR는 두 입력단의 순서를 변경할 수 있다. 지연소자와 펄스재생기(pulse stretcher)는 입력단과 출력단의 순서를 변경할 수 있으며 입출력단자 이외에도 마지막에 하나의 수가 더 입력된다. 이것은 지연소자의 길이 및 펄스재생기가 만들어내는 연속된 클럭 수를 의미한다.

<표 5> 표 작성에 사용되는 구조체
<Table 5> Table Making Using Data Structure

테이블	소자	노드	I/O
구조체	struct dev_tble{ int type_num; BOOL operated ; int i1,i2,i3,o1,o2 ; int q_size ; int* queue ; } ;	struct node_tbl{ char* name ; BOOL ready ; int state ; } ;	struct io_table{ int num ; int* data ; } ;

동작 검증과정은 회로 분석단계와 표 작성단계, 그리고 동작 테스트단계로 구성된다. 회로 분석단계는 입력된 텍스트를 분석하여 연결 상태의 이상 여부를 검사하고 전체회로를 구성하고 있는 소자의 수와 같은 데이터를 얻는 단계이다. 표 작성단계는 분석된 회로의 데이터를 가지고 소자의 동작을 테스트하기 위해 <표 5>와 같은 구조체를 이용하여 소자별 노드정의, 그리고 그 각각의 입출력에 관계되는 데이터를 표로 작성한다. 동작 테스트단계는 구

성된 표를 이용하여 소자들의 동작을 시뮬레이션하고, 1 클럭 주기(clock period)로 입력이 전달되는 소자들만 동작시킨다. 모든 소자들의 동작이 끝나면 다음 클럭 주기를 반복하는 순환 구조를 갖는다. 출력과정에서는 입력과정에서 작성한 회로 데이터의 출력노드로부터 각 클럭별로 0과 1로 구성된 출력을 얻는다.

4.2 회로 검증

시뮬레이션에서는 ALU의 정확한 동작을 검증하기 위해 다음과 같이 두 가지 제한조건을 가정한다. 첫째, ALU가 명령어들을 수행하기 전에 필요한 데이터를 메모리에 입력하는 작업이 필요하다. 이 때 메모리의 용량이 너무 크면 많은 데이터들을 직접 입력해야 하는 부가적인 작업이 필요하기 때문에 메모리의 용량을 1 워드로 제한한다. 메모리에 입력되는 데이터는 명령어 실행에 필요한 오퍼런드이다. 둘째, 설계된 ALU의 메모리에는 명령어와 오퍼런드가 함께 저장되어 있지만, 명령어를 구분하는 별도의 제어회로를 본 논문에서는 고려하지 않기 때문에 실행할 명령어는 ALU의 명령어 선택단에 직접 입력한다. 일반적으로 오퍼런드가 필요한 명령어를 수행할 경우, 첫 번째 1워드 클럭에서 명령어를 입력하고, 두 번째 1워드 클럭에서 오퍼런드를 입력한다. 그러나 시뮬레이션에서는 오퍼런드가 필요한 명령어를 수행할 경우, 명령어와 오퍼런드가 동시에 입력되기 때문에 1 워드 클럭에 처리된다. 시뮬레이션에서는 <표 6>과 같이 11개의 기본 논리 연산 명령어를 순차적으로 입력하여 실행한다.

<표 6> 실행 명령어

<Table 6> Executing Instruction

명령어	설명
① MOVM	메모리의 내용을 누산기로 전송
② ADD	누산기의 내용과 메모리의 내용을 더함
③ OR	누산기의 내용과 메모리의 내용을 OR
④ AND	누산기의 내용과 메모리의 내용을 AND
⑤ XOR	누산기의 내용과 메모리의 내용을 XOR
⑥ ROL	누산기의 내용을 왼쪽으로 로테이트
⑦ ROR	누산기의 내용을 오른쪽으로 로테이트
⑧ SHL	누산기의 내용을 왼쪽으로 쉬프트
⑨ SHR	누산기의 내용을 오른쪽으로 쉬프트
⑩ AND	누산기의 내용과 메모리의 내용을 AND
⑪ MUL	누산기의 내용과 메모리의 내용을 곱함

<표 6>에서 오퍼런드가 필요한 명령어는 바로 앞의 명령어가 실행되는 동안에 미리 오퍼런드를 메모리에 입력한다. 예를 들어 ⑩ AND 명령어는 메모리의 값을 오퍼런드로 사용하기 때문에 ⑨번 SHR 명령어 수행시 메모리에 저장된 데이터들을 오퍼런드로 사용한다. 11개의 명령어를 실행하기 위해 필요한 시간은 11워드 클럭이지만 실제 시뮬레이션에는 13 워드 클럭이 필요하다. ① MOVM 명령어를 수행하기 전에 미리 데이터를 메모리에 입력하기 위해 1워드 클럭이 더 필요하고, ⑪ MUL 명령어를 수행한 결과가 다음 워드 클럭에서 출력되기 때문에 그 결과값을 얻기 위해 1워드 클럭이 더 필요하다.

4.3 결과 분석

<표 6>의 11개 명령어를 순차적으로 수행한 결과는 [그림 8]과 같다.

```
Filename: ALU.OLS
circuit analyzing...
period    input    output    switch    device  node
 208        8         2        227       468     665
simulating...
period
 208

output display
acc0:
0000000000000000 0000000000000000 1110000000000000
0010011000000000 0010011011111111 0010011010101010
1101100101010111 1110110010101011 1101100101010111
0110110010101011 1101100101010110 1101100100000000
1010011001011001
mem0:
0000000000000000 1110000000000000 1011101000000000
0000000011111111 1111111110101010 1111111111111101
1111111111111101 1111111111111101 1111111111111101
1111111111111101 1111111100000000 1111111100000000
1111111100000000
```

[그림 8] 실행 결과

[Fig. 8] The Result Of Simulation

[그림 8]에서 output display 키워드 다음에 누산기(acc0)와 메모리(mem0)의 데이터가 출력된다. 첫 번째 워드 클럭에서는 메모리에 데이터를 입력하는 단계이기 때문에 acc0와 mem0의 출력이 모두 0 이다. 두 번째 워드 클럭에서 mem0의 출력은 1110000 000000000 이다. 이 출력 결과는 LSB에서부터 MSB의 순서로 출력된 형태이기 때문에 mem0의 출력을 역순으로 변환한0007H(0000000000000111)가 실제값 이다. ①MOVM 명령어를 수행하면 mem0의 데이터를 누산기로 전송하기 때문에 mem0의 두 번째 출력값과 acc0의 세 번째 출력값이 같아야 한다. ①MOVM 명령어 실행후 acc0와 mem0의 출력 결과를 비교해 보면 그 값이 동일함을 알 수 있다. 이러한 방법으로 각 명령어들을 실행하고 메모리와 누산기에 저장된 값을 비교하여 설계한 회로가 정상적으로 동작하는지 검증한다.

5. 결론

본 논문에서는 $LiNbO_3$ 광스위칭소자를 이용한 기본 논리게이트와 광섬유를 사용하여 로테이트, 쉬프트 레지스터, 멀티플렉서, 승산기 등의 필수적인 논리연산회로를 설계하였다. 그리고 이 논리연산회로들을 조합하여 연산부, 16-to-1 멀티플렉서, 누산기로 ALU를 구성하였다. 또한 설계된 회로를 검증할 수 있는 시뮬레이터를 개발하여 개별 게이트 수준의 동작과 소자들간의 연결 관계를 고려한 동작을 검증하였다. 시뮬레이션에서는 11개의 기본적인 논리 연산 명령어를 순차적으로 입력하여 실행하였다. 각각의 명령어를 실행한 후 메모리와 누산기에 저장된 값의 변화를 비교하여 설계한 회로가 정상적으로 동작하고 있음을 검증하였다.

본 논문에서 설계한 ALU를 실제 광프로세서에 적용하기 위해서는 감산, 제산, 분기 및 조건분기 등과 같은 명령어를 늘려야 하며, 명령어 해석 및 선택, 그리고 연산에 필요한 제어신호를 생성하는 제어회로의 구현에 대한 연구가 추가적으로 진행되어야 한다.

※ 참고문헌

[1] W. T. Cathey, K. Wagner, and W. J. Micel i, "Digital Computing with Optics," Proc. IE EE, Vol. 77, pp. 1558-1572, Oct. 1989.

[2] J. L. Johnson, "Architectural Relationships I nvolving Symbolic Substitution," Appl. Opt., Vol. 27, No. 3, pp. 529-533, Feb. 1988.

[3] A. K. Cherri and M. A. Karim, "Modified-s igned Digit Arithmetic Using an Efficient S ymbolic Substitution," Appl. Opt., Vol. 27, No. 18, pp. 3824-3827, Sep. 1988.

[4] K. H. Brenner, M. Kufner, and S. Kufner, "Highly Parallel Arithmetic Algorithms for a Digital Optical Processor Using Symbolic S ubstitution Logic," Appl. Opt., Vol. 29, No. 11, pp. 1610-1618, Apr. 1990.

[5] J. Taboury, J. M. Wang, P. Chavel, and F. Devos, "Optical Cellular Processor Architect ure. 2: Illustration and System Consideration s," Appl. Opt., Vol. 28, No. 15, pp. 3138-3 147, Aug. 1989.

[6] W. P. Heuring, H. F. Jordan, and J. P. Prat t, "Bit-serial Architecture for Optical Compu ting," Appl. Opt., Vol. 31, No. 17, pp. 321 3-3224, Jun. 1992.

[7] A. F. Benner, J. Bowman, T. Erkkila, R. J. Feuerstein, V. P. Heuring, H. F. Jordan, J. Sauer, and T. Soukup, "Digital Optical Cou nter Using Directional Coupler Switches," A ppl. Opt., Vol. 30, No. 29, pp. 4179-4189, Oct. 1991.

[8] A. D. McAulay, J. Wang, and X. Xu, "Opti cal Adder that Uses Spatial Light Rebroadca sters," Appl. Opt., Vol. 31, No. 26, pp. 558 4-5591, Sep. 1992.

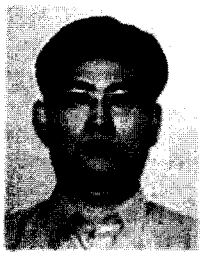
[9] P.S. Guilfoyle, "Digital Optical Computing Architectures for Compute Intensive Applicat ions ," Proc. 1994 Int'l Conf. Optical Comp uting, 1994.

[10] P.S. Guilfoyle et al., "Smart Optoeletronic Architecture for Data/Knowledge-Based Appl ications," Proc. Third Int'l Conf. on Electro nics Circuits and Systems, Vol. 2, pp. 566-569, 1996.

[11] P.S. Guilfoyle, John M. Hessenbruch, Richar d V. Stone, "Free-Space Interconnects for H igh-Performance Optoelectronic Switching," Computer, Vol. 31, No. 2, pp. 69-75, Feb. 1998.

[12] C.E. Love, H.F. Jordan, "SPOC - A Stored Program Optical Computer," IEEE Potentials Vol. 13, No. 4, Oct./Nov. 1994.

박 종 현



1993: 한양대학교
전기공학과 공학사
1995: 한양대학교
전자계산학과 공학석사
1995~1997: (주)큐닉스컴퓨터
선임연구원
1998~현재: (주)베리텍
선임연구원
관심분야: 광컴퓨터,
병렬구조시스템, 성능분석

이 원 주

1989: 한양대학교
전자계산학과 공학사
1991: 한양대학교
전자계산학과 공학석사
1998: 한양대학교
컴퓨터공학과 박사과정 수료
1999~현재: 두원공과대학
인터넷프로그래밍과 조교수
관심분야: 병렬구조시스템,
성능분석, 웹프로그래밍

전 창 호

1977: 한양대학교
전자공학과 공학사
1982: Cornell University
공학석사
1986: Cornell University
공학박사
1986~1989: 성균관대학교
전기공학과 조교수
1989~현재: 한양대학교
전자컴퓨터공학부 교수
관심분야: 병렬구조시스템,
성능분석, 컴퓨터구조