

컴퓨터 공학 기초 실험 2

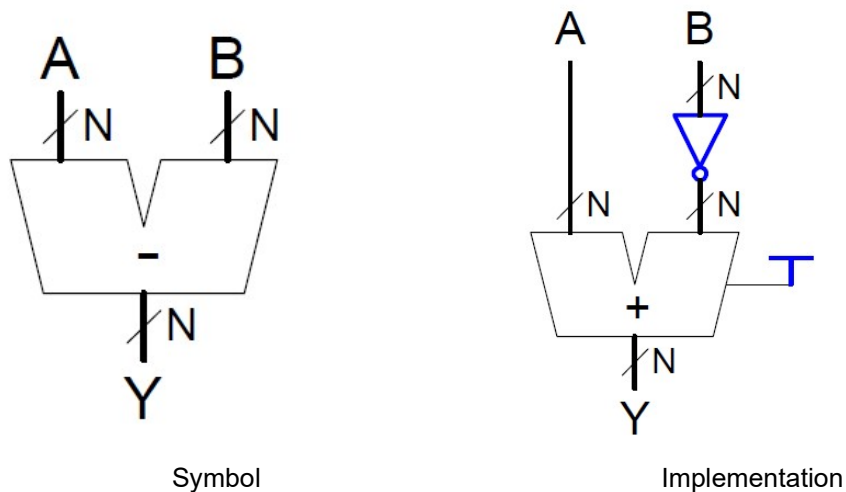
Assignment 4. Subtractor & Arithmetic Logic Unit (ALU)

1. Subtractor

➤ Functional Description

- ✓ 대부분의 digital system 에서 subtraction 을 위하여 subtrahend(감수)를 2's complement number(2의 보수)를 취해 더하게 된다.
- ✓ 2's complement number 를 구하는 방법은 해당 입력을 invert 한 후, 1을 더해주면 된다.
- ✓ 2's complement number 를 이용하여 subtraction 하는 방법
 - $A - B = A + (-B)$
 - 이때 $-B$ 를 2's complement number 로 변경

➤ B. Symbol & Implementation



결과적으로 하나의 adder 와 inverter 를 사용하여 덧셈, 뺄셈 연산 모두 수행 가능

2. Arithmetic Logic Unit (ALU)

➤ Functional Description

- ✓ Arithmetic logic unit(ALU, 산술 논리 장치)는 두 숫자의 산술연산(덧셈, 뺄셈 등등)과 논리 연산(AND, OR, XOR, 등등)을 계산하는 디지털 회로이다.
- ✓ 본 실습에서는 operator 인 3-bit opcode 에 따라 연산을 수행한다.
(실습에서는 아래의 표와 같은 8 개의 연산을 하는 ALU 를 구현한다.)

Opcode	Operation
3'b000	Not A
3'b001	Not B
3'b010	And
3'b011	Or
3'b100	Exclusive Or
3'b101	Exclusive Nor
3'b110	Addition
3'b111	Subtraction

✓ Flag

ALU 내에서 특정 조건이나 상황이 만족되었을 때, 이를 표시해주는 것을 flag 라 한다. 해당 실습에서는 carry, negative, zero, overflow 총 4 개의 flag 를 갖는다. 각각의 발생 조건은 다음과 같다.

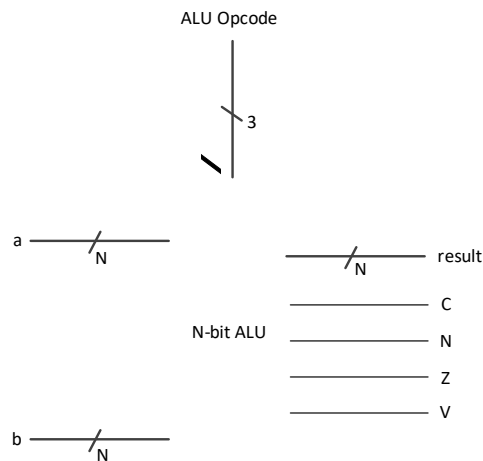
- C : Carry - 연산결과 carry 가 발생하는 경우
- N : Negative - 연산결과의 sign bit 가 1 인 경우
- Z : Zero - 연산결과가 0 인 경우
- V : Overflow - 연산결과 overflow 가 발생한 경우

Overflow 는 연산 과정에서 결과가 표현 가능한 최대 정수보다 큰 수가 입력되어 표현할 수 없는 경우를 의미한다.

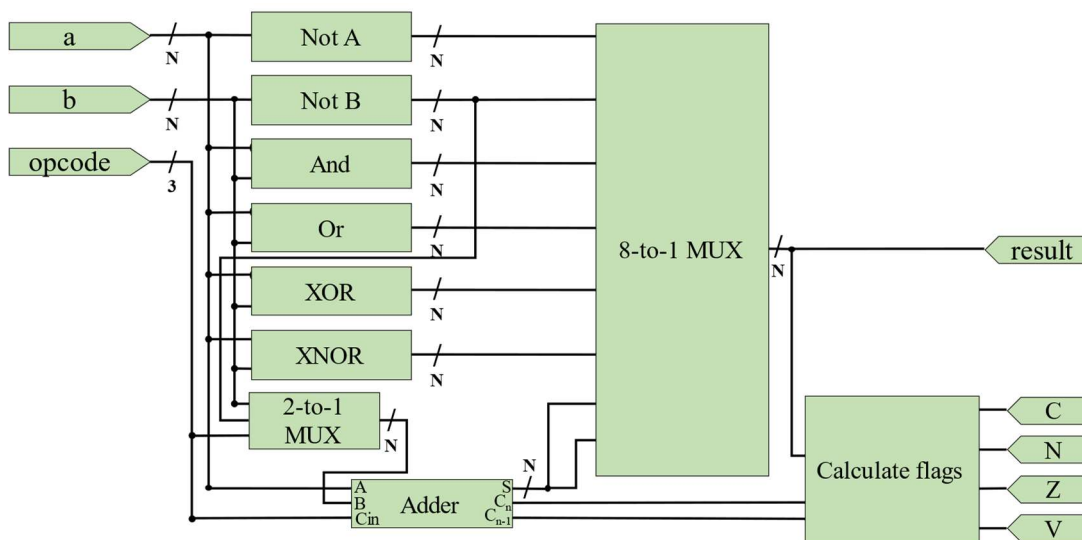
이러한 ALU 의 flag 는 비교 연산을 하는 데 사용될 수 있다.

➤ Symbol

N-bit ALU 의 symbol 은 다음과 같다.



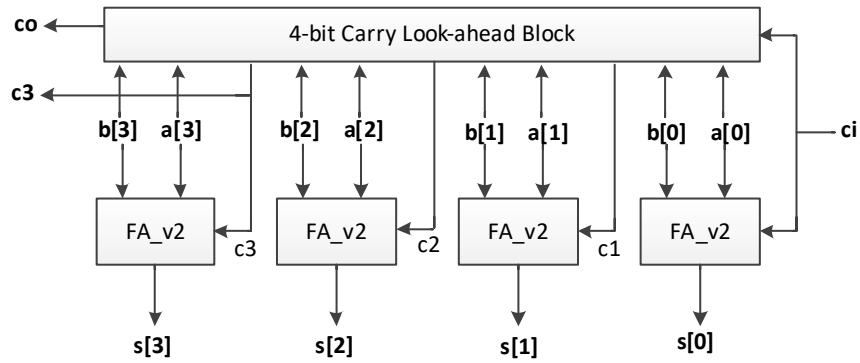
➤ Structural Description



➤ Modification of CLA

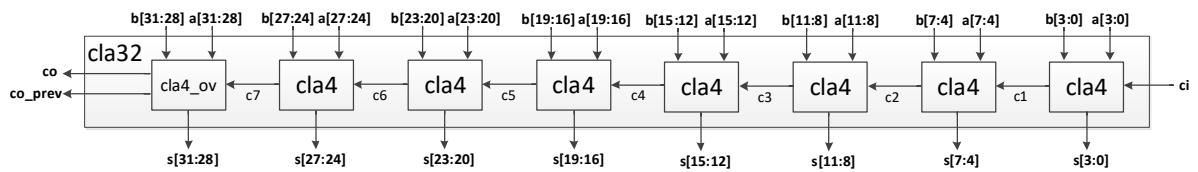
✓ 4-bit CLA to detect overflow

Overflow 를 검출하기 위하여 4-bit CLA 에서 carry out 과 carry[3]을 출력할 수 있도록 수정한다.



✓ 32-bit CLA

32-bit CLA 역시 Carry 의 최상위 bit 두 개를 출력해야 하므로, 앞선 실습에서 만들었던 32-bit CLA 의 마지막(8 번째) CLA 를 '4-bit CLA to detect overflow'로 변경한다.



3. Verilog 구현

➤ Design specification

✓ Module hierarchy description

- 기술된 top module 과 sub module 의 이름은 표와 **반드시 동일**해야 한다.
- 표에 나와 있지 않은 sub module 은 이전 과제를 참고한다.
- 아래의 표는 alu4 의 hierarchy 를 나타낸다.

구분	이름	설명
Top module	alu4	4 bits input arithmetic logic unit
Sub module	_inv_4bits	4 bits inverter module
Sub module	_and2_4bits	4 bits and logic module
Sub module	_or2_4bits	4 bits or logic module
Sub module	_xor2_4bits	4 bits xor logic module
Sub module	_xnor2_4bits	4 bits xnor logic module
Sub module	mx2_4bits	4 bits multiplexer for adder input B
Sub module	cla4_ov	4 bits CLA adder
Sub module	mx8_4bits	4 bits multiplexer for output
Sub module	cal_flags4	4 bits flag calculator

- 아래의 표는 alu32 의 hierarchy 를 나타낸다.

구분	이름	설명
Top module	alu32	32 bits input arithmetic logic unit
Sub module	_inv_32bits	32 bits inverter module
Sub module	_and2_32bits	32 bits and logic module
Sub module	_or2_32bits	32 bits or logic module
Sub module	_xor2_32bits	32 bits xor logic module
Sub module	_xnor2_32bits	32 bits xnor logic module
Sub module	mx2_32bits	32 bits multiplexer for adder input B
Sub module	cla32_ov	32 bits CLA adder
Sub module	mx8_32bits	32 bits multiplexer for output
Sub module	cal_flags32	32 bits flag calculator

- 그 외의 모듈은 강의자료 및 이전 강의자료 참고

3.1. Arithmetic logic unit 4 bits

➤ Module specification

- ✓ Module name: alu4
- ✓ I/O configuration
 - I/O 는 표와 **반드시 동일**해야 한다.
 - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	a	4 bits	Input data a
	b	4 bits	Input data b
	op	3 bits	Operation code
Output	result	4 bits	ALU result
	c	1 bit	Carry
	n	1 bit	Negative
	z	1 bit	Zero
	v	1 bit	Overflow

3.2. Flag calculator 4 bits

➤ Module specification

- ✓ Module name: cal_flags4
- ✓ I/O configuration
 - I/O 는 표와 **반드시 동일**해야 한다.
 - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	op	3 bits	Operation code
	result	4 bits	ALU result
	co_add	1 bit	Carry out bit
	c3_add	1 bit	[MSB -1] position bit
Output	c	1 bit	Carry
	n	1 bit	Negative
	z	1 bit	Zero
	v	1 bit	Overflow

3.3. Arithmetic logic unit 32 bits

➤ Module specification

- ✓ Module name: alu32
- ✓ I/O configuration
 - I/O 는 표와 **반드시 동일**해야 한다.
 - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	a	32 bits	Input data a
	b	32 bits	Input data b
	op	3 bits	Operation code
Output	result	32 bits	ALU result
	c	1 bit	Carry
	n	1 bit	Negative
	z	1 bit	Zero
	v	1 bit	Overflow

3.4. Flag calculator 32 bits

➤ Module specification

- ✓ Module name: cal_flags32
- ✓ I/O configuration
 - I/O 는 표와 **반드시 동일**해야 한다.
 - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	op	3 bits	Operation code
	result	32 bits	ALU result
	co_add	1 bit	Carry out bit
	co_prev_add	1 bit	[MSB - 1] position bit
Output	c	1 bit	Carry
	n	1 bit	Negative
	z	1 bit	Zero
	v	1 bit	Overflow

4. Report

- 레포트는 공지사항에 올린 보고서 양식에 맞추어 작성하고, 다음의 사항에 대하여도 추가적으로 작성한다.
- 4-bit ALU
 - ✓ 디지털 논리 시간에 배운 'using self-checking testbench with testvectors' 기법을 적용하여 검증할 것
 - ✓ RTL viewer, flow summary 를 포함할 것
- 32-bit ALU
 - ✓ 디지털 논리 시간에 배운 'using self-checking testbench with testvectors' 기법을 적용하여 검증할 것
 - ✓ RTL viewer, flow summary 를 포함할 것
- 원리(배경지식)에 4 개 flag 중 carry 와 overflow 의 차이에 대하여 설명한다.
- 원리(배경지식)에 Verilog 에서 blocking 과 non-blocking 에 대하여 설명하고, 아래의 Verilog file 을 compile 하여 blocking 과 non-blocking 의 차이를 설명하시오.
- ✓ Source code 는 report 에 포함하지 않는다.

```

module blocking_and_nonblocking(clk, a, b, c, d, e);
    input clk;
    input a;
    output b, c, d, e;

    blocking    U0_blocking    (.clk(clk), .a(a), .b(b), .c(c));
    nonblocking U1_nonblocking (.clk(clk), .a(a), .b(d), .c(e));
endmodule

module blocking(clk, a, b, c);
    input clk;
    input a;
    output reg b, c;

    always@(posedge clk)
    begin
        b = a;
        c = b;
    end
endmodule

module nonblocking(clk, a, b, c);
    input clk;
    input a;
    output reg b, c;

    always@(posedge clk)
    begin
        b <= a;
        c <= b;
    end
endmodule
    
```

➤ 채점기준

세부사항		점수	최상	상	중	하	최하
소스코드	Source code 가 잘 작성 되었는가? (Structural design 으로 작성되었는가?)	10	10	8	5	3	0
	주석을 적절히 달았는가? (반드시 영어로 주석 작성)	20	20	15	10	5	0
설계검증 (보고서)	보고서를 성실히 작성하였는가? (보고서 형식에 맞추어 작성)	30	30	20	10	5	0
	합성결과를 설명하였는가?	10	10	8	5	3	0
	검증을 제대로 수행하였는가? (모든 입력 조합, waveform 설명)	30	30	20	10	5	0
총점		100					

5. Submission

- 제출기한
 - 자세한 제출기한은 KLAS 와 일정을 참고
- 과제 업로드
 - ✓ Source code 와 report 를 같이 ZIP 파일로 압축하여 KLAS(종합정보서비스) 과제 제출에 해당 과제 upload
 - ✓ 업로드 파일명은 (요일#)_(학번)_Assignment_#.zip
 - 요일번호
 - 실습 미수강은 0
 - 월요일 0, 1, 2 교시 1
 - 화요일 0, 1, 2 교시 2
 - 수요일 5, 6, 7 교시 3
 - Ex) 월요일 반 수강, 2019110609, Assignment 1 제출 시
2_2019110609_Assignment_01.zip 으로 제출
 - ✓ Report 명은 (학번)_(요일#)_Assignment_#.pdf
 - 요일 번호는 위의 업로드 파일명과 동일하게 진행
 - ✓ Ex) 수요일 반 수강, 2019110609, Assignment 1 제출 시
2_2019110609_Assignment_01.pdf 으로 제출
 - ✓ Report 는 PDF 로 변환해 제출 (미수행시 감점)
- Source code 압축 시 db, incremental_db, simulation 폴더는 삭제 (미수행시 감점)
- Source code 압축 시 ~.bak 파일 삭제 (미수행시 감점)
- 제출 프로젝트
 - ✓ Project: alu4
File list: gates.v, fa_v2.v, clb.v, cla4_ov.v, mx2.v, mx2_4bits, mx8.v, mx8_4bits.v, cal_flags4.v, alu4.v, tb_alu4.v
 - ✓ B. Project: alu32
File list: gates.v, fa_v2.v, clb.v, cla4.v, cla4_ov.v, cla32_ov.v, mx2_32bits.v, mx8_32bits.v, cal_flags32.v, alu32.v, tb_alu32.v