컴퓨터 공학 기초 실험2 보고서

실험제목: Subtraction & ALU

실험일자: 2019년 09월 10일 (화)

제출일자: 2019년 09월 16일 (월)

학 과: 컴퓨터정보공학과

담당교수: 이준환 교수님

실습분반: 화요일 0, 1, 2

학 번: 2015722031

성 명: 박 태 성

1. 제목 및 목적

A. 제목

Subtractor & ALU

B. 목적

Arithmetic Logic Unit(ALU)를 설계를 해보면서 Multiplexer와 Subtractor의 원리와 구조에 대해 이해하는 것을 목적으로 한다. ALU의 status flags가 연산 결과에 따라 어떻게 변화하는지 알아보는데 목적을 둔다.

2. 원리(배경지식)

A. ALU

ALU는 두 숫자의 산술(덧셈, 뺄셈 등)과 논리(AND, OR, XOR 등)을 계산하는 디지털 회로이다. ALU는 컴퓨터 CPU 설계의 기본이 된다. Flag는 ALU의 연산 결과 값의 상태를 나타낸다. Carry, negative, zero, overflow 총 4개의 flag가 있다. ALU의 연산결과 carry가 발생하면 C(Carry)가 1이 된다. 연산결과 MSB가 1이 되면 N(Negative)가 1이 된다. 연산결과 0인 경우 Z(Zero)가 1이 된다. 연산결과가 해당 숫자의 bit들로 표현할 수 있는 범위를 벗어나면 V(Overflow)가 1이 된다. ALU의 flag는 비교 연산을 하는데 유용하다. ALU는 operator인 3-bits opcode에 따라 연산을 수행한다.

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

B. Carry와 Overflow의 차이점

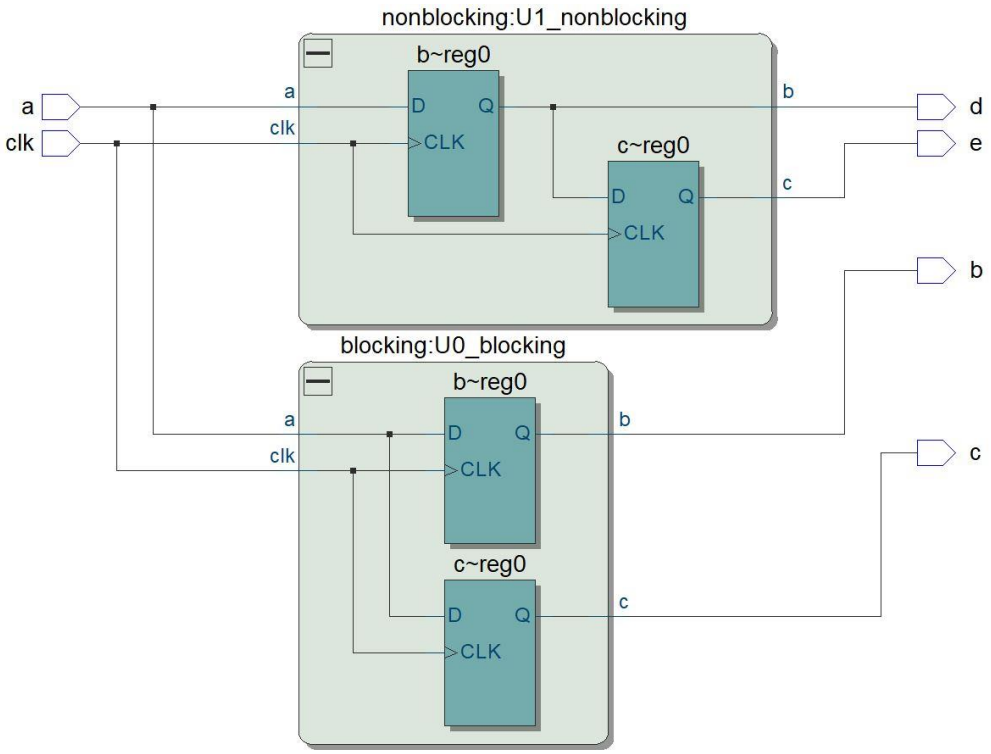
C(Carry)는 N bits의 두 수가 더해졌을 경우, 자릿수 올림이 발생하는 경우 1이 된다. V(Overflow)는 두 N bits의 양수, 두 N bits의 음수를 더했을 경우 MSB가 바뀌어 올바른 연산결과가 나오지 않았을 경우 1이 된다. 올바른 연산결과가 나오지 않았다는 것은 표현할 수 있는 범위를 벗어났다는 것과 같은 의미이다. V를 구하는 논리식은 co ⊕

co_prev이다.

C. Verilog에서 blocking과 non-blocking

Blocking은 코드가 순서대로 흘러가는 흐름을 막는다는 의미를 갖는다. Always 블록 안에서는 모든 문장들이 C언어와 같이 순차적으로 코드를 진행한다. 예로, a = b, c = a를 수행하면, b가 a에 저장되고 업데이트 된 a가 c값에 저장된다. 결과는 a = b = c이다. 반대로 non-blocking 할당은 right hand side(RHS)의 변수들을 전부 업데이트 한 후 left hand side(LHS)의 변수들을 업데이트한다. 다음과 같은 표현을 사용한다. Q <= D.

- RTL Viewer



Blocking은 clock이 변화할 때, a라는 값이 동시에 들어가지만 non-blocking은 첫 번째 flip-flop에 a가 들어가고 두 번째 flip-flop에는 알 수 없는 값이 들어간다. Blocking은 combinational circuit에서 non-blocking은 sequential circuit에서 주로 사용된다.

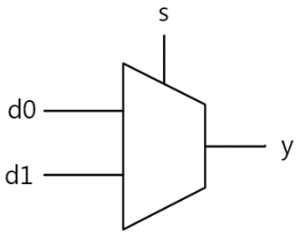
3. 설계 세부사항

A. Multiplexer

i. Functional Description

S의 값에 따라 두 input d0, d1 중 하나의 값을 output으로 출력한다. S가 0이면 d0가 출력되고 s가 1이면 d1이 출력된다.

ii. Symbol



iii. Truth Table

Input			Output
s	**d0**	**d1**	**y**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

iv. 1-bit 8-to-1 multiplexer 구현

1-bit 2-to-1 multiplexer 7개를 instanciation하여 구현한다. 이에 따라서 selection 신호가 3개로 늘어난다.

B. 4-bits ALU

i. Functional Description

두 input a, b를 입력 받아서 8가지의 논리, 산술 연산을 진행한다. 입력 받은 opcode에 따라서 multiplexer가 해당 결과값을 output result로 출력한다. Result와 다른 조건을 통합적으로 판단하여, flag를 발생한다.

ii. I/O Description

Port	Name	Bandwidth	Description
Input	a	4 bit	Operand A
	b	4 bit	Operand B
	op	3 bit	Opcode
Output	result	4 bit	ALU 연산 결과
	c	1 bit	Carry flag

	n	1 bit	Negative flag
	z	1 bit	Zero flag
	v	1 bit	Overflow flag

iii. Module Description

Classification	Name	Description
Module	alu4	4-bits ALU
Instance	U0_inv_4bits	4-bits inverter a
	U1_inv_4bits	4-bits inverter b
	U2_and2_4bits	4-bits a and b
	U3_or2_4bits	4-bits a or b
	U4_xor2_4bits	4-bits a xor b
	U5_xnor2_4bits	4-bits a xnor b
	U6_add	4-bits CLA(a add b)
	U7_sub	4-bits CLA(a sub b)
	U8_mx8_4bits	4-bits 8-to-1 multiplexer
	U9_cal_flags4	Calculate flags

C. 32-bits ALU

i. I/O Description

Port	Name	Bandwidth	Description
Input	a	32 bit	Operand A
	b	32 bit	Operand B
	op	3 bit	Opcode
Output	result	32 bit	ALU 연산 결과
	c	1 bit	Carry flag
	n	1 bit	Negative flag
	z	1 bit	Zero flag
	v	1 bit	Overflow flag

ii. Module Description

Classification	Name	Description
Module	alu32	32-bits ALU
Instance	U0_inv_32bits	32-bits inverter a
	U1_inv_32bits	32-bits inverter b
	U2_and2_32bits	32-bits a and b
	U3_or2_32bits	32-bits a or b

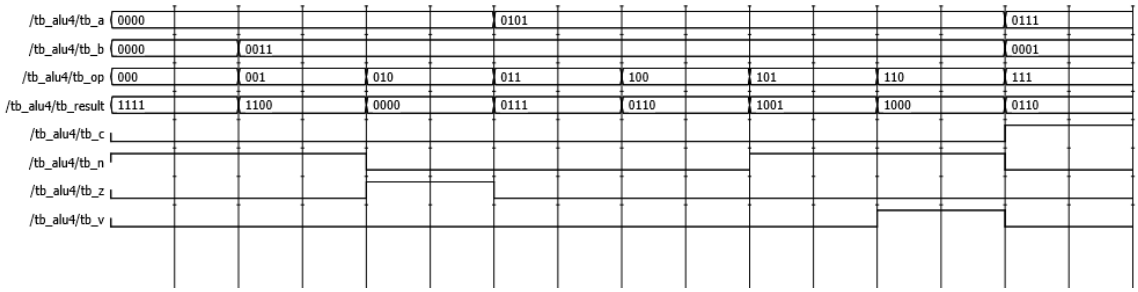
	U4_xor2_32bits	32-bits a xor b
	U5_xnor2_32bits	32-bits a xnor b
	U6_add	32-bits CLA(a add b)
	U7_sub	32-bits CLA(a sub b)
	U8_mx8_32bits	32-bits 8-to-1 multiplexer
	U9_cal_flags4	Calculate flags

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

i. 4-bits ALU

1. Waveform of Self-checking Testbench



Self-checking testbench는 예상 결과 값과 결과 값이 다르면 error를 출력한다.

2. Modelsim Transcript Window

```
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all

VSIM 2>
```

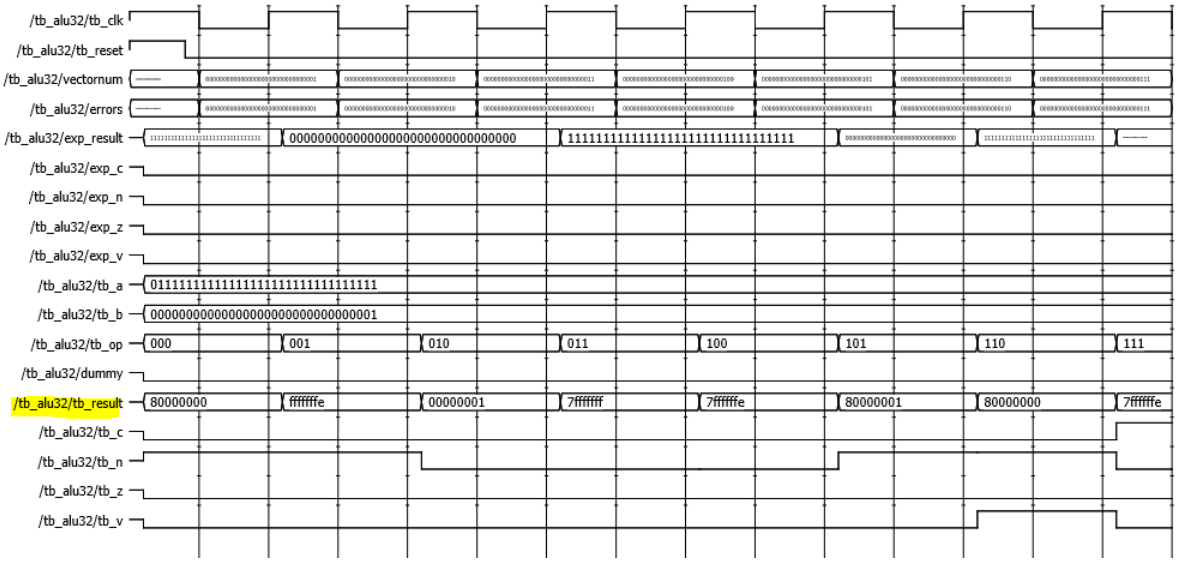
예상한 결과와 같은 결과가 출력되어 error message가 출력되지 않았다. Error message를 정상적으로 출력되는지 확인하기 위하여 testbench 코드를 수정해보았다.

```
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# opcode lll Succeed

VSIM 2>
```

특정한 조건이나 상황을 만족하면 transcript window에 error message가 출력된다.

ii. 32-bits ALU

1. Waveform of Self-checking testbench with testvectors



Self-checking testbench에 test vector를 추가로 사용한다. 입력과 예상 출력값을 testvector 파일로 양식에 맞게 작성한뒤 일정한 clock을 통해서 결과를 확인한다. Rising edge일 때, 파일로부터 입출력 값들을 변수들에 저장한다. Falling edge일 때, 예상 결과 값과 결과 값이 다르면 error를 출력한다.

2. Modelsim Transcript Window

```
# Error: inputs = a: 7fffffff, b: 00000001, op: 000
# outputs = result: 80000000, c: 0, n: 1, z: 0, v: 0 (ffffffff 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 001
# outputs = result: fffffffe, c: 0, n: 1, z: 0, v: 0 (00000000 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 010
# outputs = result: 00000001, c: 0, n: 0, z: 0, v: 0 (00000000 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 011
# outputs = result: 7fffffff, c: 0, n: 0, z: 0, v: 0 (ffffffff 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 100
# outputs = result: 7ffffffe, c: 0, n: 0, z: 0, v: 0 (ffffffff 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 101
# outputs = result: 80000001, c: 0, n: 1, z: 0, v: 0 (00000000 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 110
# outputs = result: 80000000, c: 0, n: 1, z: 0, v: 1 (ffffffff 0 0 0 0 expected)
# Error: inputs = a: 7fffffff, b: 00000001, op: 111
# outputs = result: 7ffffffe, c: 1, n: 0, z: 0, v: 0 (00000000 0 0 0 0 expected)
#          8 tests completed with          8 errors
# ** Note: $finish    : C:/Users/user/Desktop/ALU32/tb_alu32.v(58)
#    Time: 75 ns  Iteration: 1  Instance: /tb_alu32
# 1
# Break in Module tb_alu32 at C:/Users/user/Desktop/ALU32/tb_alu32.v line 58

VSIM 2>
```

example.tv 파일의 예상 결과 값과 실제 결과 값이 전부 다르게 나와 8개의 error가 발생하였다. Example.tv 파일의 예상 결과 값을 수정하여 error가 발생하지 않게 하였다.

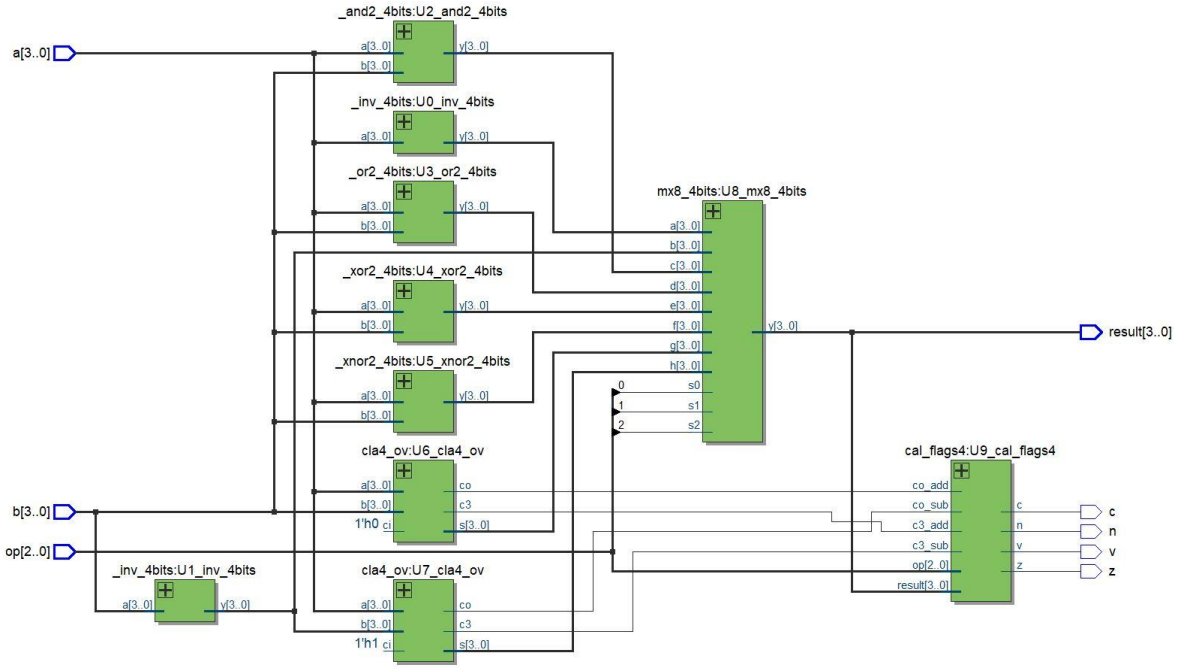
```
#          8 tests completed with          0 errors
# ** Note: $finish    : C:/Users/user/Desktop/ALU32/tb_alu32.v(58)
#    Time: 75 ns  Iteration: 1  Instance: /tb_alu32
# 1
# Break in Module tb_alu32 at C:/Users/user/Desktop/ALU32/tb_alu32.v line 58

VSIM 2>
```

'8 tests completed with 0 errors'를 확인하였다.

B. 합성(synthesis) 결과

i. 4-bits ALU

1. RTL Viewer



Adder인 cla4_ov를 subtractor로 사용하기 위하여 input b를 inverter를 통하여 1의 보수를 취해주고 ci에 1을 넣어 세 operands를 더해주면 감산기 역할을 수행한다.

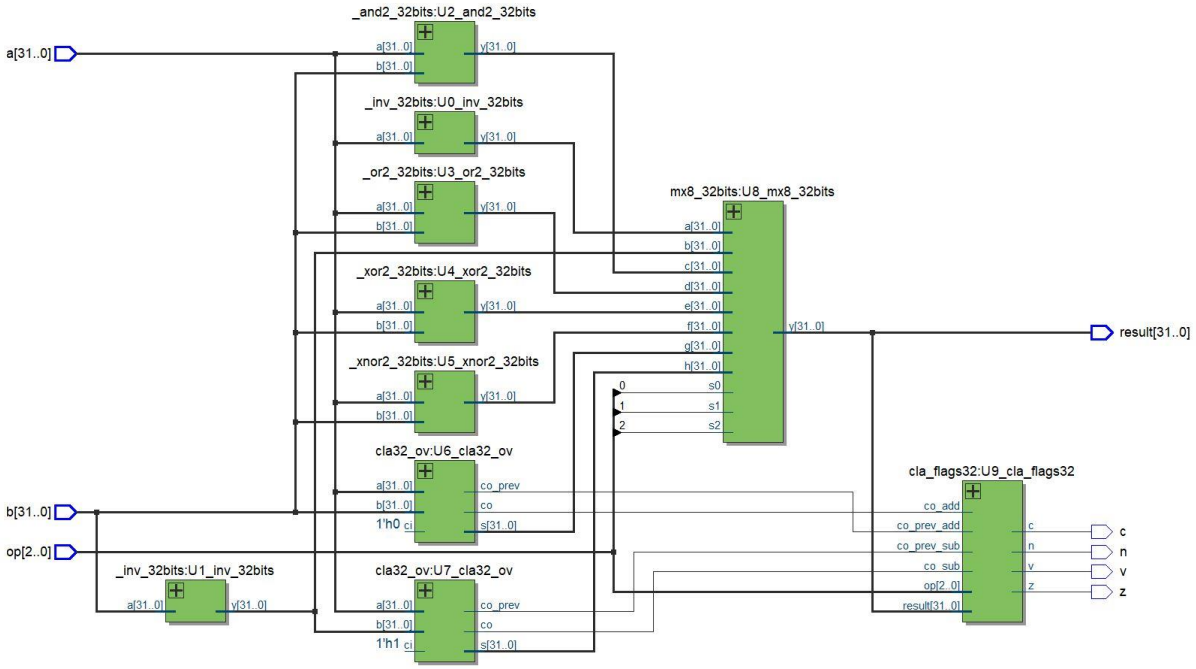
2. Flow Summary

Flow Summary	
Flow Status	Successful - Tue Sep 10 23:31:56 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	alu4
Top-level Entity Name	alu4
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	19
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

총 19개의 pin이 사용되었다.

ii. 32-bits ALU

1. RTL Viewer



4-bits ALU와 같은 구조이다.

2. Flow Summary

Flow Summary	
Flow Status	Successful - Tue Sep 10 23:38:53 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	alu32
Top-level Entity Name	alu32
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	103
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

총 pin의 개수와 총 port의 개수가 같다.

5. 고찰 및 결론

A. 고찰

tb_alu32를 modelsim에서 확인하는데 다음과 같은 오류가 지속적으로 발생하였다.

```
# Top level modules:
#       tb_alu32
# End time: 11:42:08 on Sep 09,2019, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cyclonev_ver -L cyclonev_hssi_ver -L cyclonev_pcie_hip_ver -L rtl_work -L work -voptarg
s="+acc"  tb_alu32
# Error loading design
# Error: Error loading design
#        Pausing macro execution
# MACRO ./alu32_run_msim_rtl_verilog.do PAUSED at line 21
```

Waveform이 나오지 않고 transcript에서 오류의 원인을 알려주지 않아 오류의 원인을 찾을 수 없었다. 조교님 컴퓨터로 원인을 찾을 수 있었다. 오류의 원인은 alu32 코드였다. Instanciation할 때 instance name을 작성하지 않았기 때문이다.

B. 결론

Opcode에 따라서 수행할 연산을 선택하는 줄 알았다. 그러나, 모든 연산을 수행하고 연산 결과 중 1가지를 multiplexer로 선택하는 기능을 구현하는 것이 목적이었다. 물론, opcode에 따라서 수행할 연산을 선택할 수 있다. 그러나, 설계 복잡도가 높아져 이번 과정에서는 연산 결과에 따라 flag를 출력하는 ALU를 구현하였다.

두 번째 피연산자의 1의 보수를 구하여 첫 번째 피연산자와 carry in을 1 입력으로 adder에 값을 주면 뺄셈기의 역할을 수행한다. 1의 보수는 inverter를 통해서 구한다. 그러므로 ALU의 구조에는 두 개의 adder가 아닌 한 개의 adder만을 사용하여 addition과 subtraction을 할 수 있다.

6. 참고문헌

David Money Harris & Sarah L. Harris / Digital Design and Computer Architecture / Elsevier Korea L.L.C / November 2013