디지털 시스템 설계 (CSED273) Lab 4. 이진수 연산 보고서

2023.04.27.

20220923 이윤혁

1. 개요

Lab4에서는 이진수 연산을 수행하는 Adder와 Subtractor, Multiplier를 다룬다. Lab4_1에서는 Half Adder와 Full Adder를 구현한다. Lab4_2에서는 5비트 리플 가산기를 구현한다. Lab4_3에서는 5비트 리플 감산기를 구현한다. Lab4_4에서는 5x3 이진 곱셈기를 구현한다.

2. 이론적 배경

2.1. 반가산기 (Half adder)

반가산기는 두 수 A, B를 입력 받아 A B의 합과 Carry out을 출력하는 연산기이다.

2.2. 전가산기 (Full adder)

전가산기는 두 수 A, B와 Carry in을 입력받아 A B, Carry in의 합과 Carry out을 출력하는 연산기이다. 반가산기 두 개와 OR 게이트를 이용해 구현할 수 있다.

2.3. N-bit 리플 가산기/감산기 (N-bit Ripple Adder/Subtractor)

N-bit 리플 가산기는 N개의 전가산기를 직렬로 이어 만들 수 있다. 각각의 전가산기는 Carry in으로 이전의 전가산기의 Carry out을 받는다. 각각의 전가산기는 입력된 두 수 A, B의 비트와 Carry in을 이용해 합과 Carry out을 구현하고, 합은 출력비트로, Carry out은 다음 전가산기의 Carry in으로 넘겨준다. 연산이 낮은 자리수 전가산기부터 높은 자리수 전가산기로 순차적으로 이루어지므로, 입력하는 숫자의 자리수가 길수록 연산처리가 오래 걸린다.

감산기가 A - B의 연산을 한다고 가정했을 때, 감산기는 A – B를 A + (-B)의 형태로 계산한다. -B를 계산하기 위해 2의 보수의 방식을 사용한다. 감산시에는 carry in으로 1을 무조건 입력하고, B의 모든 비트를 NOT게이트를 이용해 부정연산(0->1, 1->0)하면 -B를 2의 보수로 표현할 수 있다.

2.4. MxN 이진 곱셈기 (MxN Binary Multiplier)

M bit x N bit 이진 곱셈은 M bit Multiplicant를 N bit Multiplier와 bitwise AND 연산과 shifting으로 계산할 수 있다. M bit Multiplicant를 N bit의 k번째 bit와 bitwise and연산을 하고 한자리 shift한다. 이 값과 k-1번째 partial sum을 더하면 k번째 partial sum을 구할 수 있다. Partial sum은 M bit ripple adder로 구할 수 있으며, partial sum을 0번째(Multiplier의 최하위비트와 M bit Multiplicant를 bitwise and 연산한 수) 부터 N – 1번째 partial sum을 구하면 곱셈의 결과를 얻을 수 있다. 결과는 N - 1번째 partial sum과 carry를 더한 값이다.

3. 실험 준비

3.1. Lab4_1

반가산기의 진리표는 [Table 1]과 같다.

[Table 1] Truth table of Half adder

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

진리표로부터 C와 S의 식을 계산하면 다음과 같다.

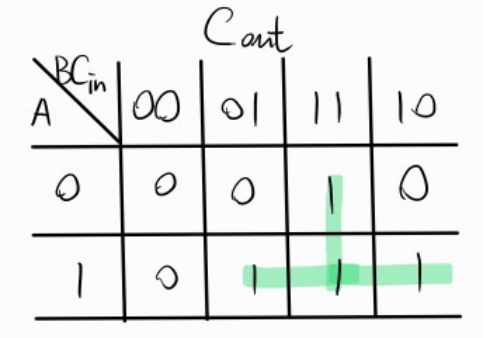
C = A AND B
S = A XOR B

전가산기의 진리표는 [Table 2]과 같다.

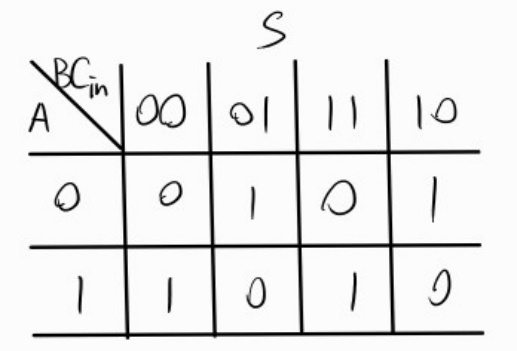
A	B	C_in	C_out	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_out과 S의 K-map은 다음과 같다.

[Figure 1] K-map of C_out



[Figure 2] K-map of S



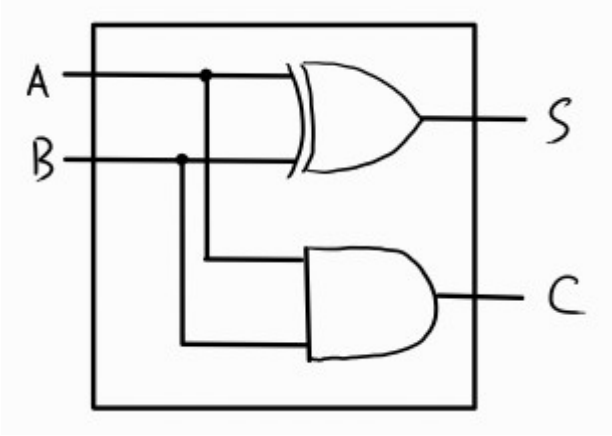
K-map으로부터 S와 C_out의 식을 구하면 다음과 같다.

S = A′B′C_in + A′BC_in′ + AB′C_in′ + ABC_in
  = (A′B′ + AB)C_in + (A′B + AB′)C_in′
  = (A′B + AB′)′C_in + (A′B + AB′)C_in′
  = A ⊕ B ⊕ C_in

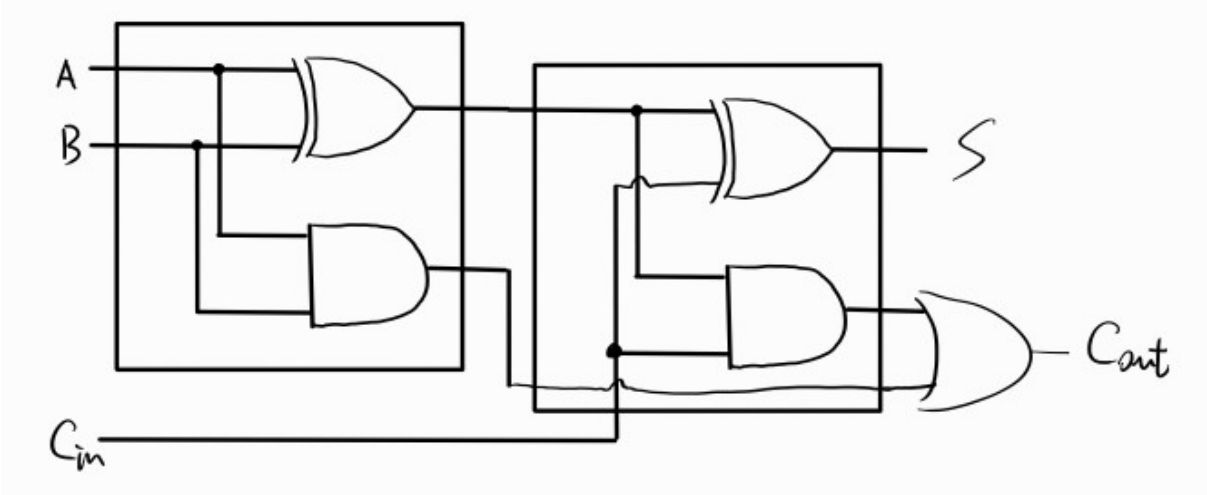
C_out = AB + A′BC_in + AB′C_in (half adder를 재사용하기 위해 Simplification 는 AB만 진행)
  = AB + (A ⊕ B)C_in

반가산기와 전가산기의 회로를 구현하면 [Figure 3], [Figure 4]와 같다.

[Figure 3] Circuit of Half Adder



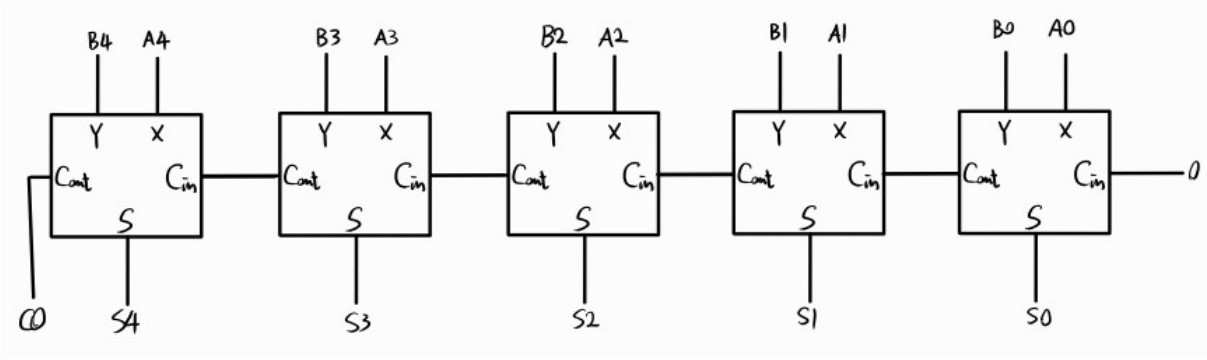
[Figure 4] Circuit of Full Adder



3.2. Lab4_2

5비트 리플가산기는 5개의 전가산기를 사용한다. 5비트 리플가산기의 회로도는 [Figure 5]와 같다.

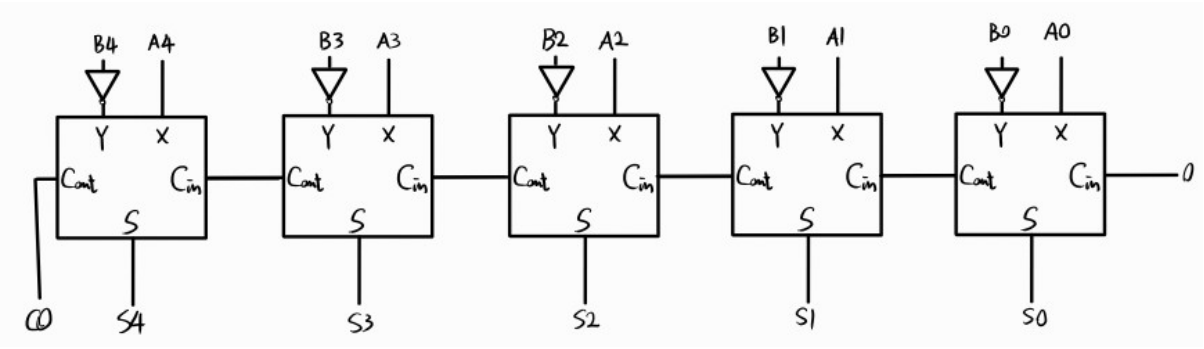
[Figure 5] Circuit of 5-bit Ripple adder



3.3. Lab4_3

5비트 리플 감산기의 회로도는 [Figure 6]와 같다.

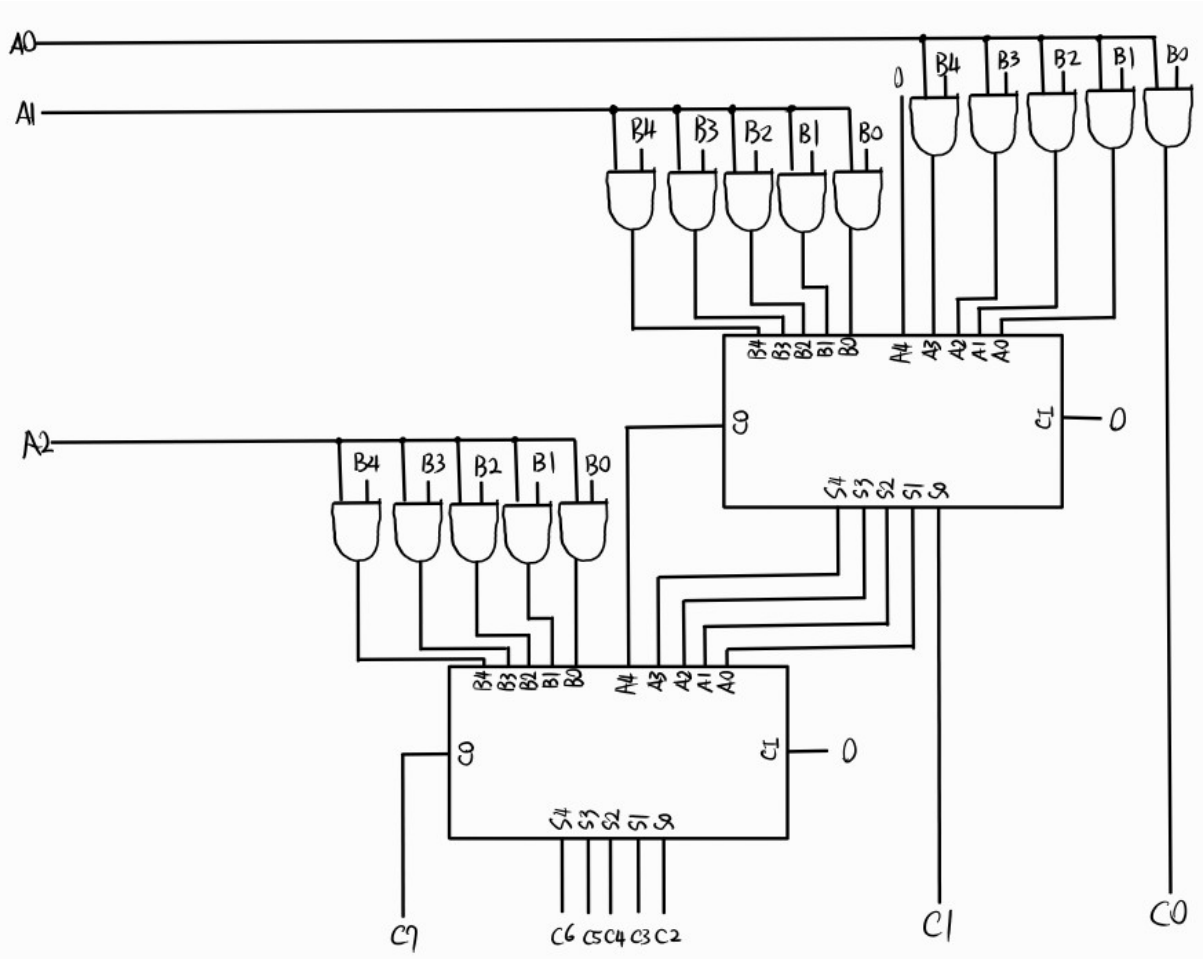
[Figure 6] Circuit of 5-bit Ripple subtractor



3.4. Lab4_4

5x3 이진 곱셈기의 회로도는 [Figure 7]과 같다.

[Figure 7] Circuit of 5x3 Binary multiplier

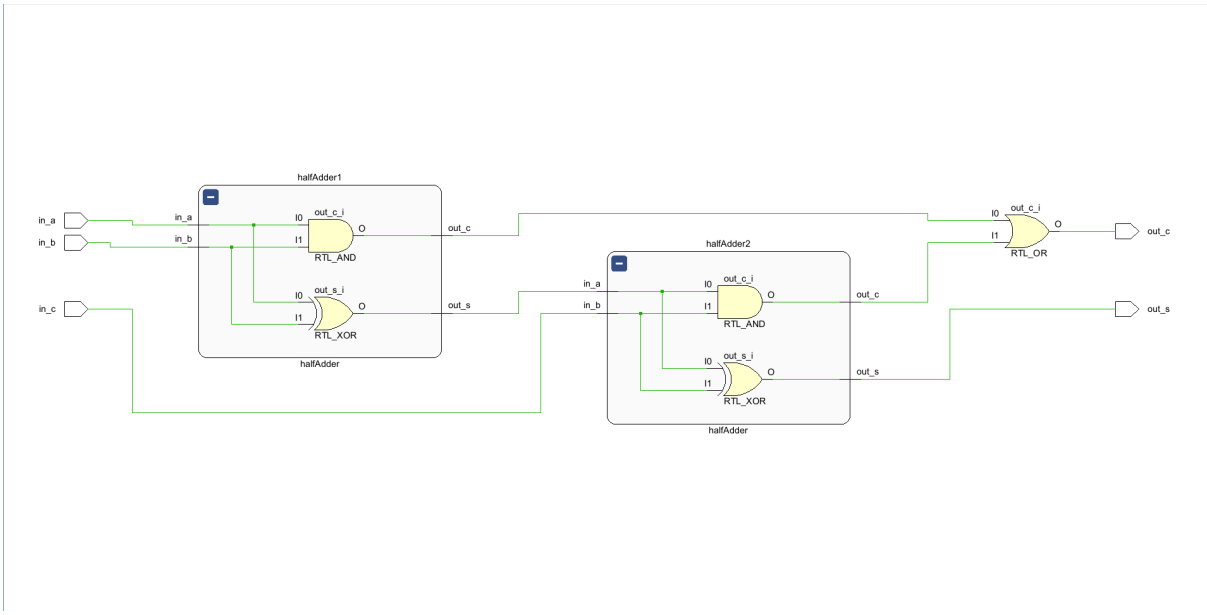


4. 결과

4.1. Lab4_1

[Figure 8]은 반가산기와 전가산기의 Schematic이다. 반가산기는 세부회로 보기를 통해 schematic으로 표현했다.

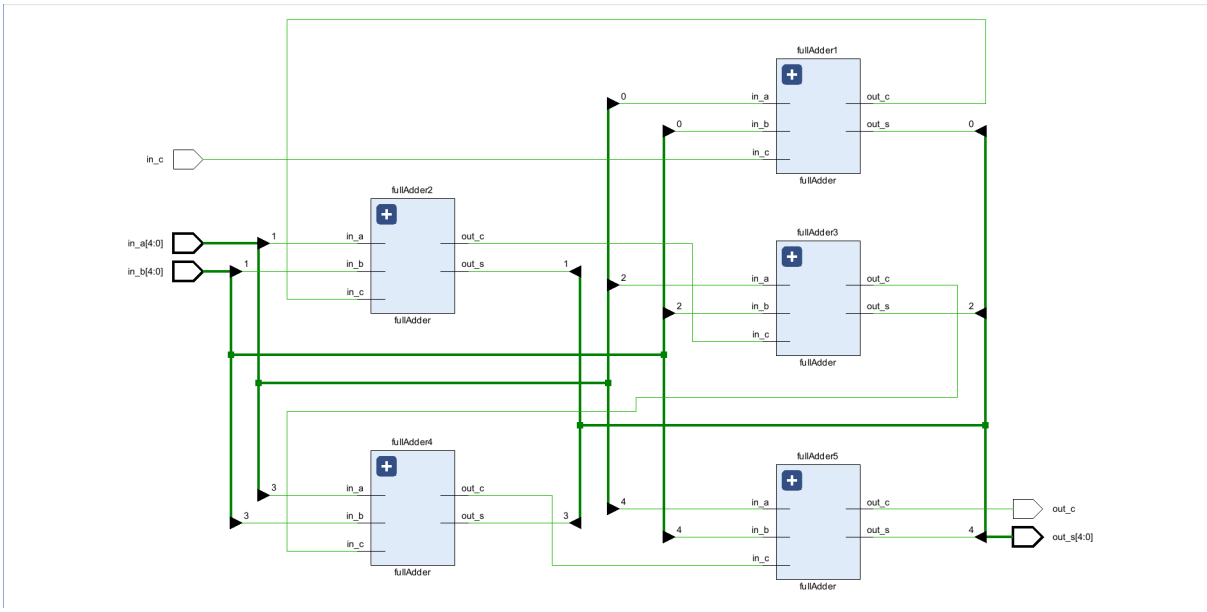
[Figure 8] Schematic of Half Adder and Full Adder



4.2. Lab4_2

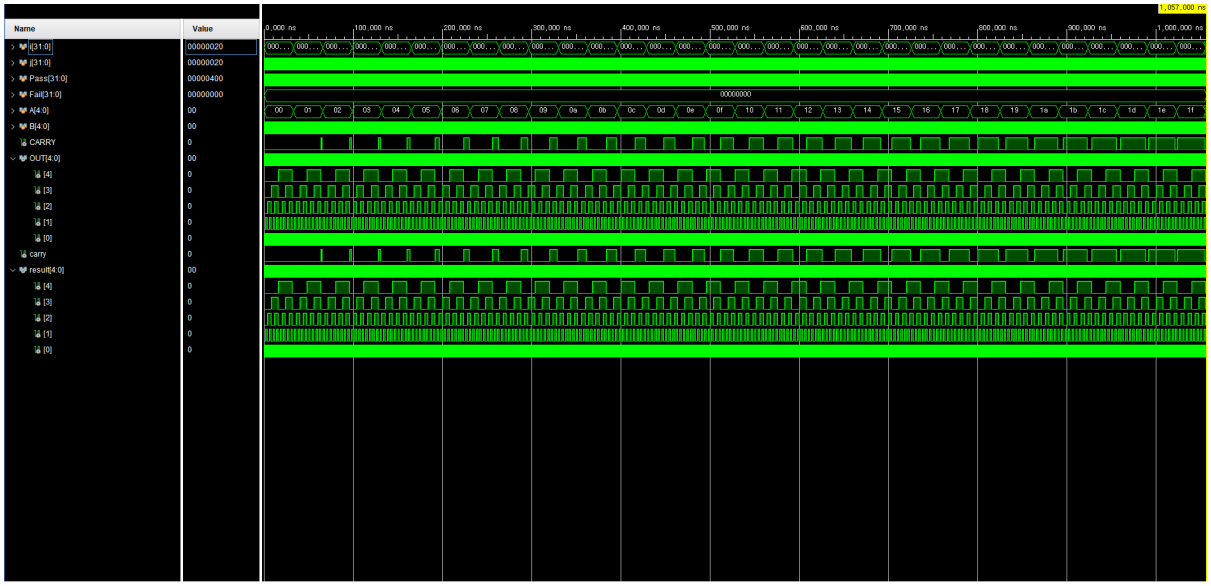
[Figure 9]는 Lab4_1의 전가산기를 이용해 5-bit Ripple Adder를 구현한 Schematic이다.

[Figure 9] Schematic of 5-bit Ripple Adder



[Figure 10]은 주어진 Testbench로 시뮬레이션한 파형이다.

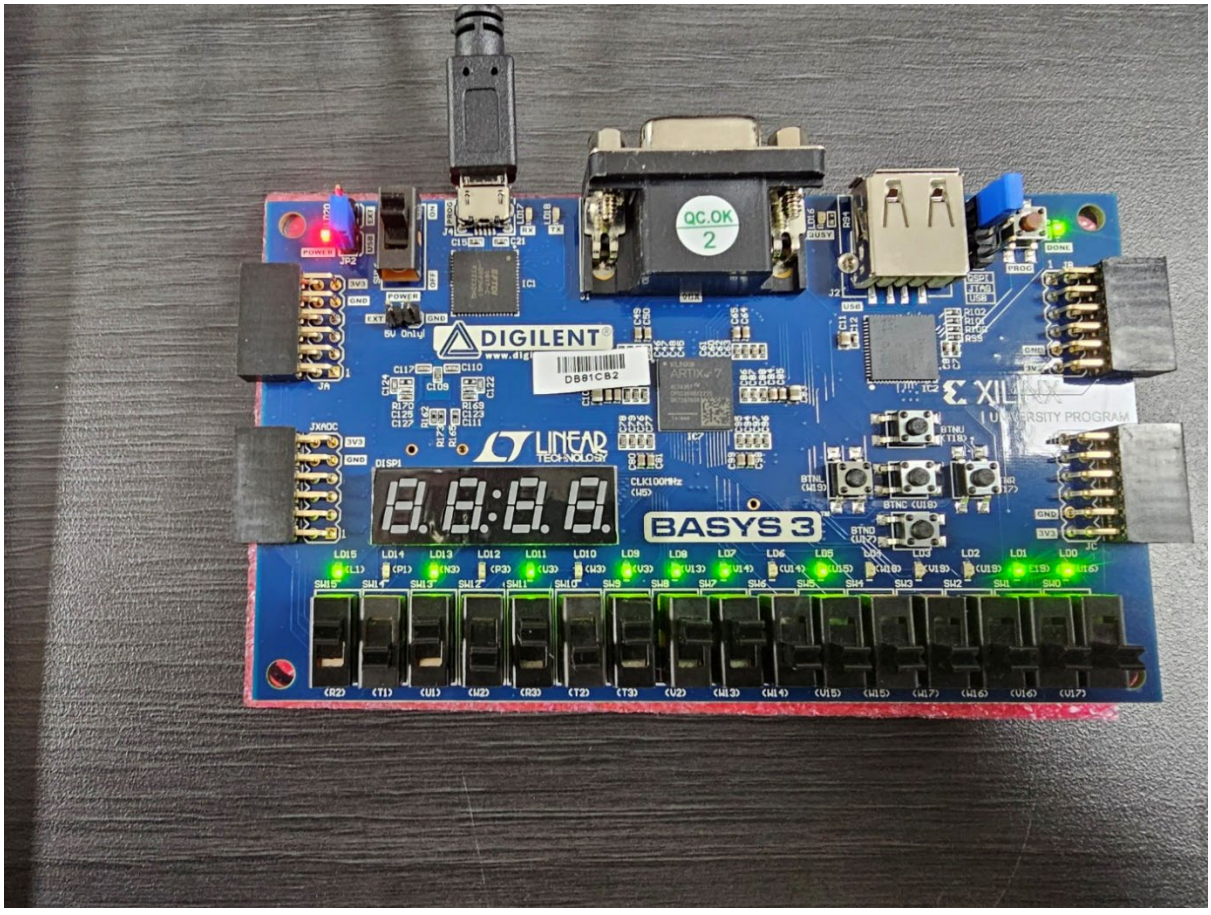
[Figure 10] Simulation waveform of 5-bit Ripple Adder



시뮬레이션 종료 후 Fail count가 0임을 보아 구현이 제대로 되었음을 알 수 있다.

[Figure 11]은 FPGA로 5-bit Ripple Adder를 구현한 사진이다.

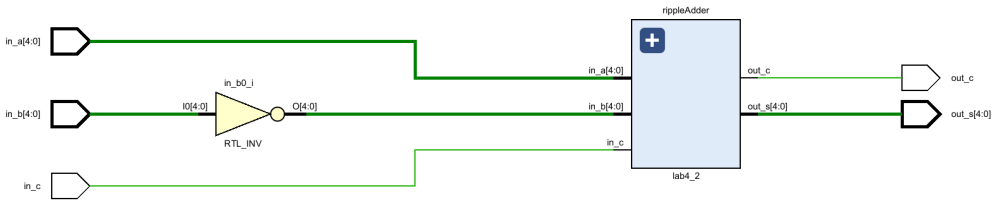
[Figure 11] 5-bit Ripple Adder implemented in FPGA



4.3. Lab4_3

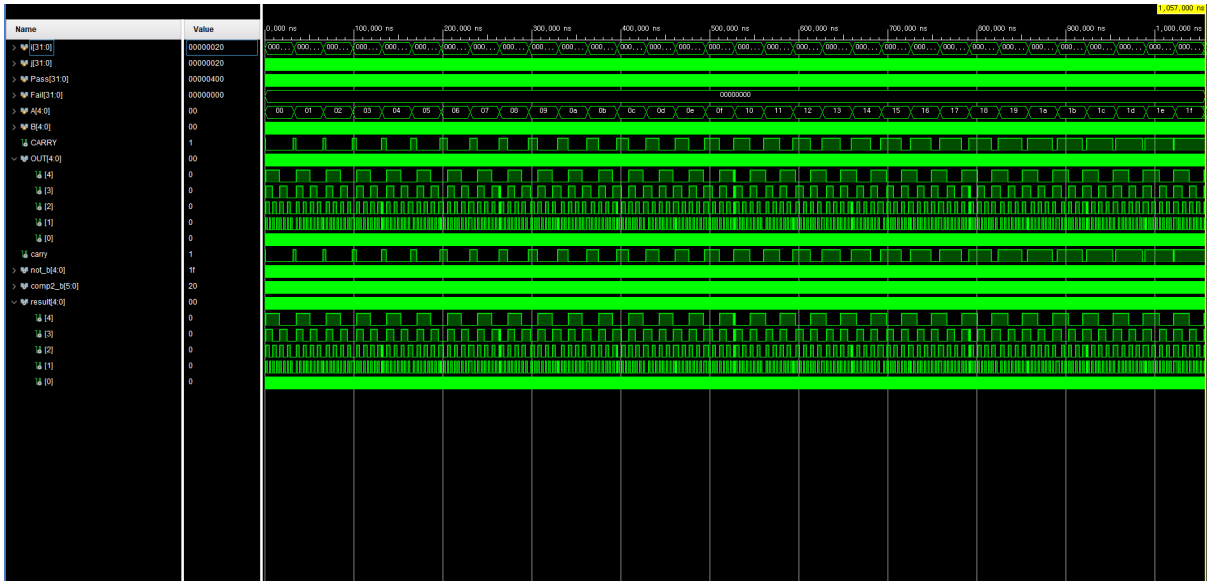
[Figure 12]는 Lab4_2의 5-bit Ripple Adder를 이용해 5-bit Ripple Subtractor를 구현한 Schematic이다.

[Figure 2] Schematic of 5-bit Ripple Subtractor



[Figure 13]는 주어진 Testbench로 시뮬레이션한 파형이다.

[Figure 13] Simulation waveform of 5-bit Ripple Subtractor

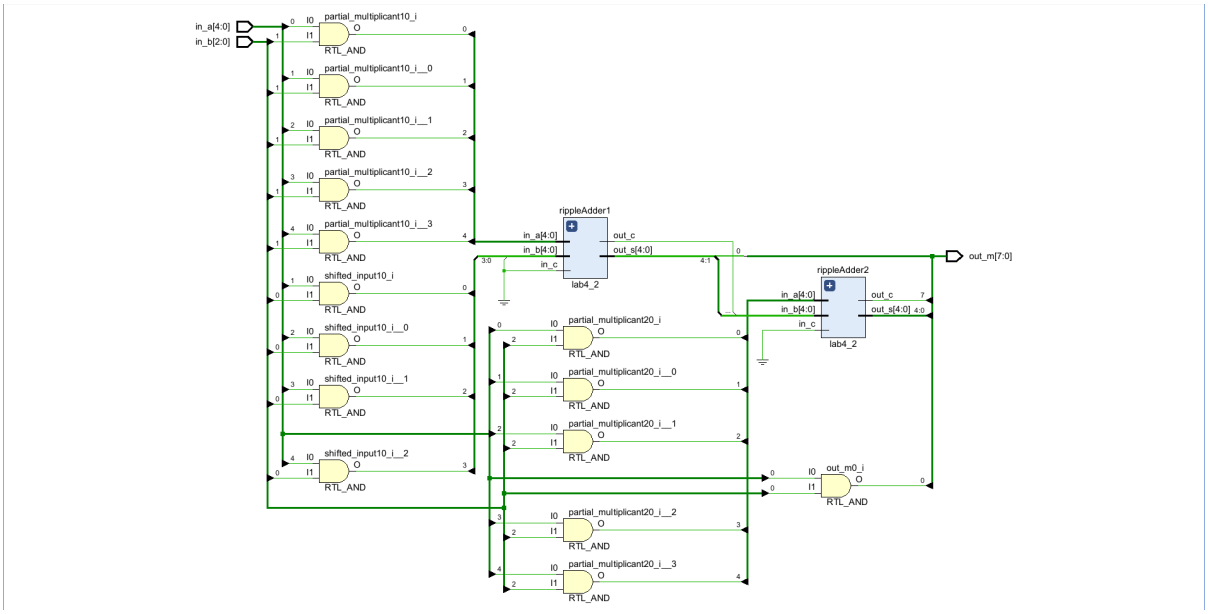


시뮬레이션 종료 후 Fail count가 0임을 보아 구현이 제대로 되었음을 알 수 있다.

4.4. Lab4_4

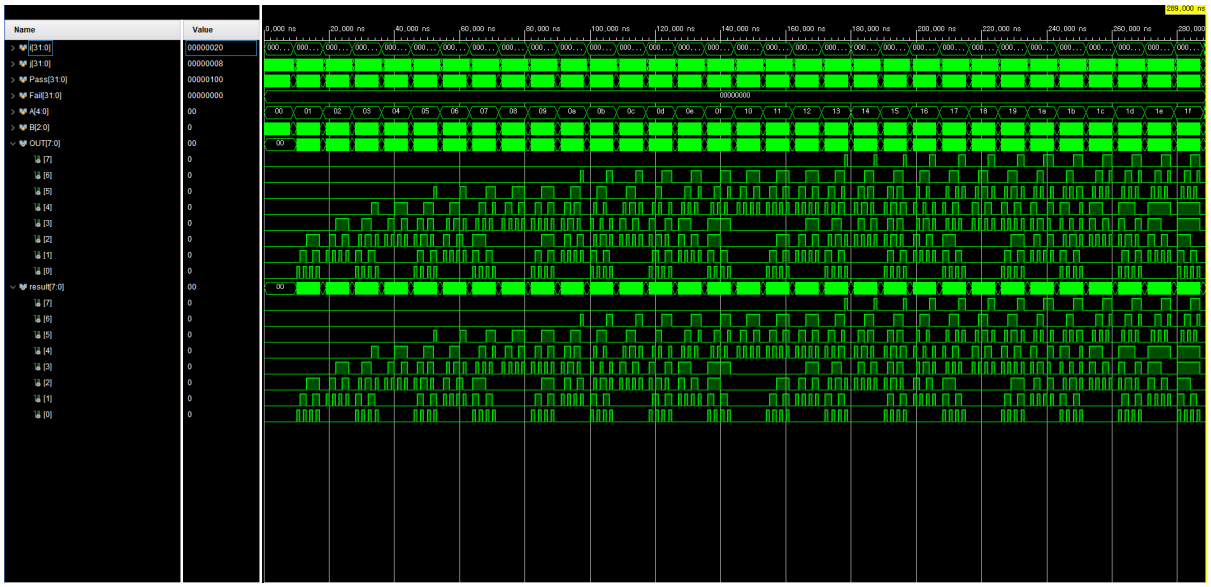
[Figure 14]은 Lab4_2의 5-bit Ripple Adder를 이용해 5x3 Binary Multiplier를 구현한 Schematic이다.

[Figure 14] Schematic of 5x3 Binary Multiplier



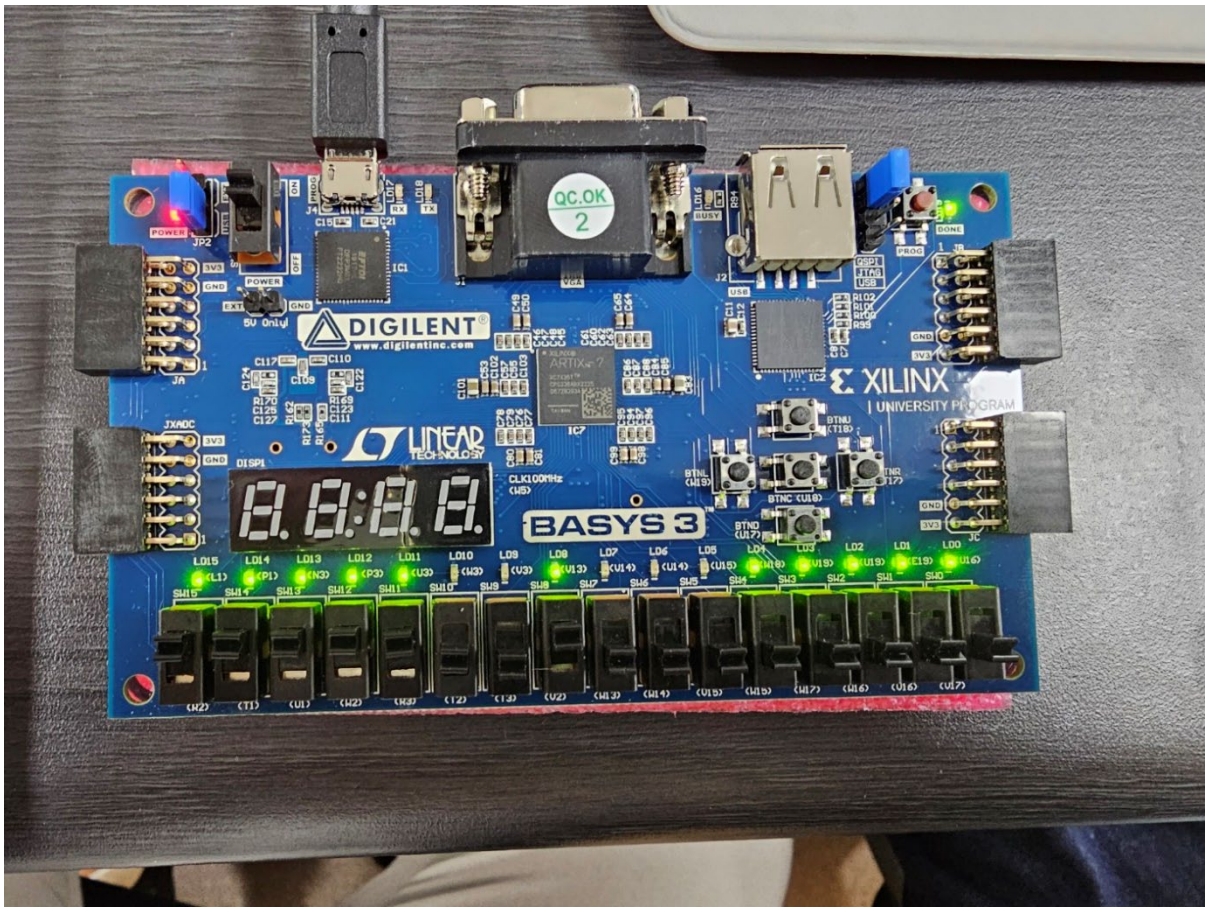
[Figure 15]는 주어진 Testbench로 시뮬레이션한 파형이다.

[Figure 15] Simulation waveform of 5x3 Binary Multiplier



[Figure 16]은 FPGA로 5x3 Binary Multiplier를 구현한 사진이다.

[Figure 16] 5x3 Binary Multiplier implemented in FPGA



시뮬레이션 종료 후 Fail count가 0임을 보아 구현이 제대로 되었음을 알 수 있다.

5. 논의

Lab4_1에서는 Truth Table을 이용해 반가산기와 전가산기의 식을 계산하고, 회로를 직접 구현하였다. Lab4_2에서는 Lab4_1의 전가산기를 이용해 Ripple Adder를 구현할 수 있었다. Lab4_3은 Lab4_2의 Ripple Adder를 이용해 Ripple Subtractor를 구현할 수 있었다. Lab4_4에서는 Lab4_2의 전가산기를 이용해 Binary Multiplier를 구현할 수 있었다. 이번 Lab4를 진행하면서, 간단한 기능을 하는 회로를 Truth Table과 Simplification을 통해 구현하고, 간단한 기능을 가진 회로들을 연결함으로써 좀 더 추상적인 기능을 할 수 있는 회로를 구현할 수 있다는 것을 크게 느낄 수 있었다.

또한 회로를 구현하면서 wire를 연결하는 작업이 조금 복잡했다. 실험을 진행하면서 시뮬레이션 결과를 확인하고, 코드를 고치면서 여러 개의 wire를 한번에 연산하는 Verilog 문법을 익힐 수 있었다.