6주차 결과 보고서

20191621 이민영

1. 실험 목적

전가산기와 반가산기, 전감산기와 반감산기, 그리고 부호 변환기(Code converter)에 대한 개념을 이해하고, 이해한 내용을 바탕으로 Verilog를 사용해서 이를 구현한다.

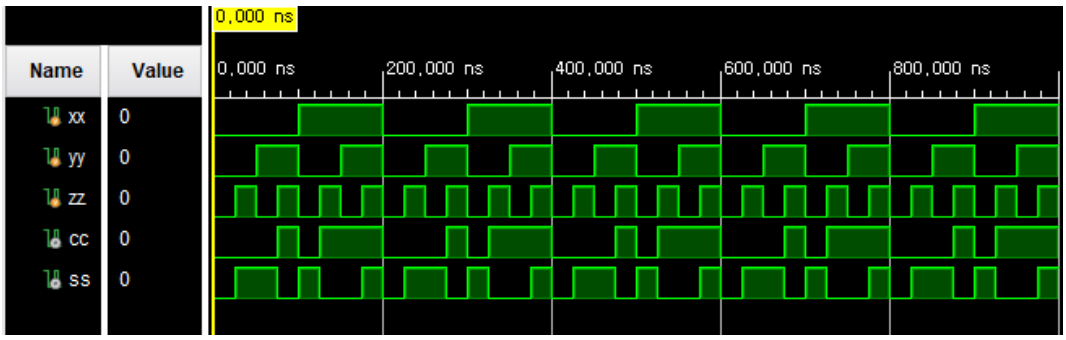
부호 변환기의 경우, 카르노 맵을 이용해서 SOP, POS 형태로 최소식을 나타내고, 이를 이용해서 부호 변환기를 Verilog를 통해서 살펴본다.

2. Full Adder 및 Half Adder의 Simulation 결과 및 과정에 대해서 설명하시오.

1) Full Adder(전가산기)

전가산기는 X,Y,Z가 입력으로 들어오고 출력값으로 S와 C가 나온다.

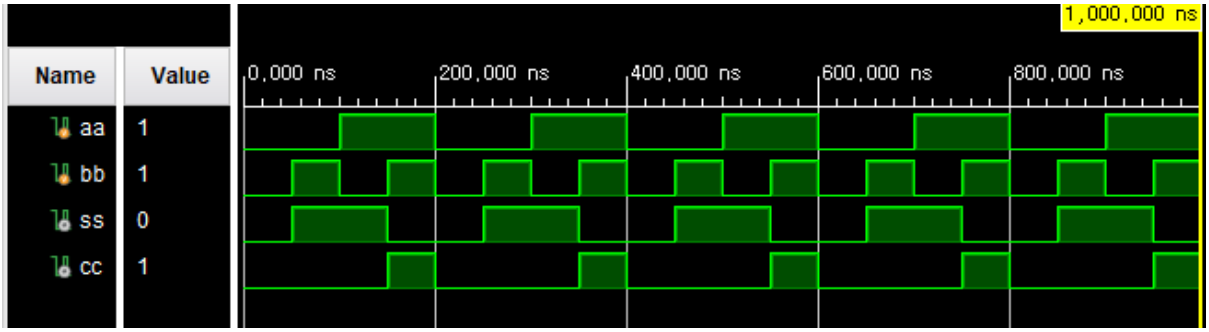
$$S = (A \oplus B) \oplus C_{in}$$
$$C_{out} = C_{in}(A \oplus B) + AB$$

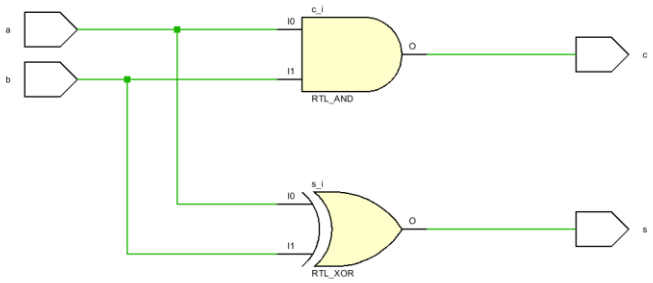


2) Half Adder(반가산기)

반 가산기는 A,B가 입력으로 들어오고 S와 C가 출력된다.

$$S = \overline{A}B + A\overline{B} = A \oplus B$$
$$C = AB$$





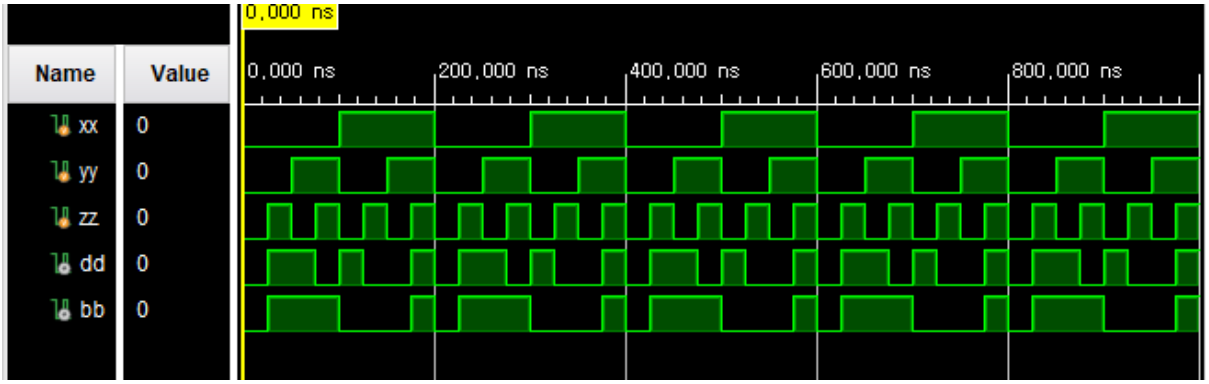
Simulation 결과, 전가산기와 반가산기 모두 진리표와 동일한 결과값을 얻을 수 있었다. 전 가산기의 경우 Carry의 역할을 눈으로 확인해 볼 수 있었다. 즉, 반가산기의 경우, 결과값인 S는 A와 B의 XOR값이며, 결과값으로 나오는 Carry는 A와 B의 and 결과와 같다는 것을 확인해 볼 수 있었다. 전가산기의 경우 결과값인 S는 A와 B의 XOR 값에 Cin을 XOR 시킨 것과 같고, Carry인 C는 A와 B의 XOR 값에 Cin을 and 시킨 결과와 A와 B의 and 시킨 연산을 or 연산 시킨 것과 같다는 것을 알 수 있다.

3. Full Subtracter 및 Half Subtracter의 simulation 결과 및 과정에 대해서 설명하시오.

1) Full Subtracter(전감산기)

전감산기는 X,Y,Z가 입력으로 들어오고 D와 B가 출력된다.

$$D_n = A_n \oplus B_n \oplus b_{n-1} \qquad b_n = \overline{(A_n \oplus B_n)} \bullet b_{n-1} + \overline{A_n} \bullet B_n$$

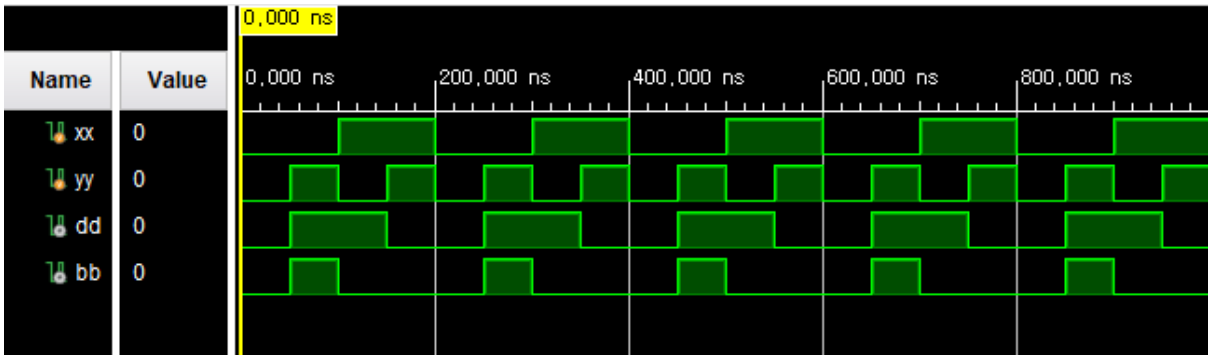


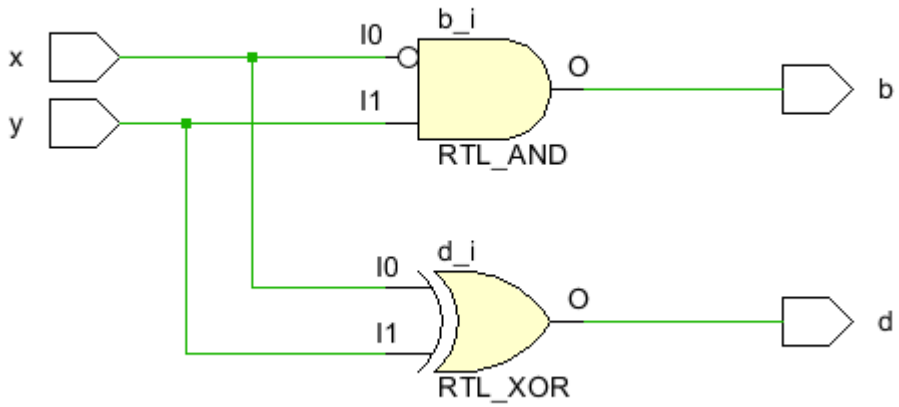
2) Half Subtracter(반감산기)

반감산기는 X,Y가 입력으로 들어오고, B와 D가 출력된다.

$$D = \overline{A} \bullet B + A \bullet \overline{B} = A \oplus B$$

$$b = \overline{A} \bullet B$$





전감산기와 반감산기의 Simulation결과 진리표와 동일한 값이 출력되는 것을 알 수 있으며 Borrow의 값을 눈으로 확인할 수 있다. 전감산기의 경우, Difference인 D는 A와B와 bn-1을 XOR 시킨 결과이며, bn은 A와 B의 XOR에 보수를 취한 결과값에 bn-1을 and 연산 취한것과, A의 보수 값에 B를 and 연산 시킨 결과값을 or 연산 시킨 결과와 같다는 것을 알 수 있다.

4. 8421(BCD)-2421 Code converter simulation 결과 및 과정에 대해서 설명하시오. (진리표 작성 및 카르노 맵 SOP form, POS form 포함)

<진리표>

	8421BCD				2421BCD			
	D	C	B	A	Output4	Output3	Output2	Output1
	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	1
	0	0	1	0	0	0	1	0
	0	0	1	1	0	0	1	1
	0	1	0	0	0	1	0	0
	0	1	0	1	1	0	1	1
	0	1	1	0	1	1	0	0
	0	1	1	1	1	1	0	1
	1	0	0	0	1	1	1	0
	1	0	0	1	1	1	1	1

<카르노 맵>

DC / BA	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	X	X
10	0	0	X	X

Output 1

DC / BA	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	1	0	X	X
10	1	0	X	X

Output 2

DC / BA	00	01	11	10
00	0	1	X	1
01	0	0	X	1
11	0	1	X	X
10	0	1	X	X

Output 3

DC / BA	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

Output4

<SOP>

BA \ DC	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	X	X
10	0	0	X	X

Output 1

BA \ DC	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	1	0	X	X
10	1	0	X	X

Output 2

BA \ DC	00	01	11	10
00	0	1	X	1
01	0	0	X	1
11	0	1	X	X
10	0	1	X	X

Output 3

BA \ DC	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

Output4

<POS>

BA \ DC	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	X	X
10	0	0	X	X

Output 1

BA \ DC	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	1	0	X	X
10	1	0	X	X

Output 2

BA \ DC	00	01	11	10
00	0	1	X	1
01	0	0	X	1
11	0	1	X	X
10	0	1	X	X

Output 3

BA \ DC	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

Output4

1) Output1

SOP 결과 : A

POS 결과 : A

2) Output2

SOP 결과 : D+CB'A+BC' = (D'(CB'A)'(BC')')'

POS 결과 : (A+B+D)(B+C+D)(B'+C') = ((A+B+D)' + (B+C+D)' + (B'+C')')'

= (A'B'D'+B'C'D'+BC)'

3) Output3

SOP 결과 : D+A'C+BC = (D'(A'C)'(BC)')'

POS 결과 : (C+D)(A'+B+D) = ((C+D)' + (A'+B+D)')' = ((C'D') + (AB'D'))'

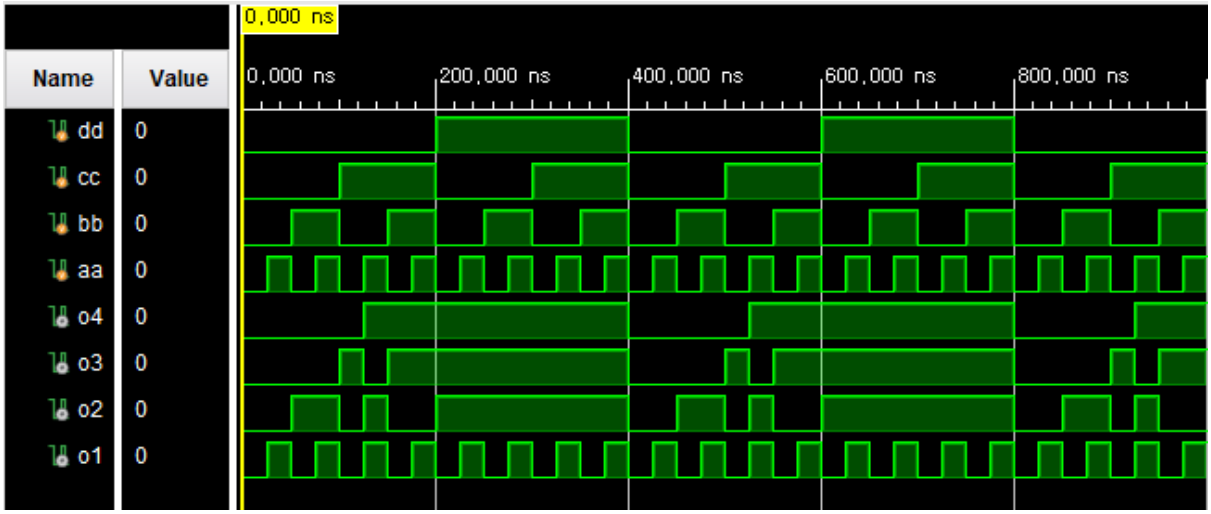
4) Output4

SOP 결과 : D+CB+CA = (D'(CB)'(CA)')'

POS 결과 : (C+D)(A+B+D) = ((C+D)' + (A+B+D)')' = ((C'D') + (A'B'D'))'

각각을 카르노맵을 이용해서 SOP, POS의 결과를 도출해낼 수 있었다. SOP의 경우에는 1을 묶어서 최대한 간단한 형태의 최소식으로 나타내었다. POS의 결과 0으로 묶어서 최대한 간단한 형태로 나타내었고 이 결과에 보수를 취해서 결과값을 얻어내었다.

또한 최대한 NAND와 NOR의 형태로 나타내기 위해서 부정을 취해서 식을 나타내었다.



Simulation 결과 진리표와 동일하게 나왔다는 것을 알 수 있었다. 각각의 값이 BCD converter의 값과 일치했다.

5. 결과 검토 및 논의 사항

이번 실험을 통해서 Full Adder, Half Adder, Full Subtractor, Half Subtractor을 구현해보았다. 가산기의 경우 Carry를 이용해서 덧셈을 시행하였고, 감산기의 경우 Borrow를 이용해서 뺄셈을 시행하였다. 각각 xor, and 등의 논리 회로를 이용해서 이를 표현할 수 있었고, Simulation 결과를 통해서 각각의 감산기와 가산기의 결과가 진리표와 동일하다는 것을 알아낼 수 있었다.

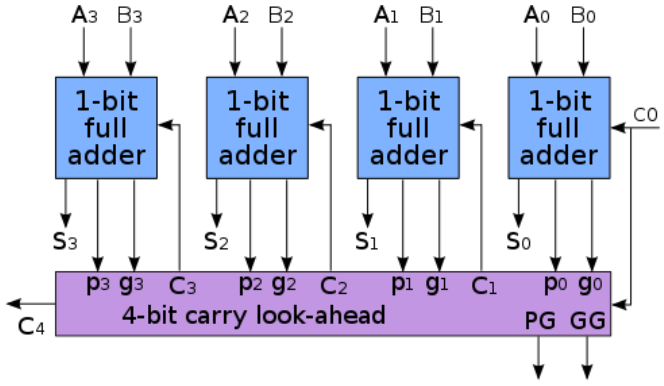
8421(BCD)-2421 Code Converter에서는 카르노 맵을 이용해서 각각의 output이 나오기 위한 최소식을 구할 수 있었다. 각각 SOP와 POS를 이용해서 구해볼 수 있었으며, NAND gate와 NOR gate를 이용한 형태로 변형시켜서 Simulation을 구현해볼 수 있었다. 또한 Simulation의 결과값이 진리표와 동일하다는 것을 확인할 수 있었다.

6. 추가 이론 조사 및 작성

예측 자리올림수 장치(lookahead carry unit, LCU)

예측 자리 올림수 장치는 가산기의 계산시간을 줄이기 위해서 사용되는 논리 장치이며, 보통 자리올림수 예측 가산기(CLA)와 연결해서 사용한다.

16비트 가산기는 4개의 4 비트 자리올림수 예측 가산기를 조합해서 만들어진다.



위의 그림은 4비트 자리올림수 예측 가산기이다. 예측 자리올림수 장치는 각각의 자리올림수 예측 가산기에 맞는 자리올림수 예측을 만들어 낸다.

64비트 가산기는 4개의 자리올림수 예측 가산기와 1개의 예측 자리올림수 장치가 합쳐져서 16비트 가산기를 만드는데, 이 16비트 가산기를 4개를 조합하면 64비트 가산기를 만들 수 있다.