**디지털회로실험 보고서**

-7주차-

전자공학과

2015104027

박정진

**실험 결과**

Quartus Simulation (top\_bdf.bdf, top\_bdf.vwf 파일이 메인 파일)

Schematic (schematic.pdf 에서 더 자세히 볼 수 있음)

A screenshot of a cell phone

Description automatically generated

A close up of a map

Description automatically generated

A screen shot of a computer

Description automatically generatedA close up of a map

Description automatically generated

동작 원리 :

버튼(input A[9:0])을 누르지 않았을 때 High, 눌렀을 때 Low로 동작하게 하였다. (BCD Encoder, ttl74147의 인풋이 activate low로 동작하는 특성을 고려하여 설계) 초기에 아무것도 누르지 않았을 때 A는 10'b111111111의 상태이다. 이 상태에서 버튼을 하나 누르면 ttl74174의 DFF가 인코딩 된 BCD 4bit를 저장한다. (nand - or의 조합으로 어떤 버튼을 누르면 pulse가 생성되게 하여 DFF의 CLK로 사용한다.) ttl74174는 Series로 연결되어 있어 버튼을 두 번 누르면 차례로 저장이 된다. (이하 저장된 두 개의 값을 A, B라 칭함) 들어온 A와 B는 xor와 4bit adder ttl74283로 만들어진 adder and subtractor는 input Addn\_Sub 의 값이 0일땐 add, 1 일땐 subtract를 수행한다. (표 1-1 을 통해 계산 결과를 확인 할 수 있다.) 계산 결과 상 add 일 땐 경우가 0~18, subtract일 땐 -9 ~ 9 의 경우가 나올 수 있다. 계산 결과를7segment decoder에 표시하기 위해서는 ttl7448에 들어갈 두 자리의 BCD코드에 의해 총 8bit가 필요한데 10의 자리는 add일 땐 10의 자리를, subtract일 땐 - 를 표시해야 한다. 이를 위해 K map 을 이용해 {Addn\_Sub, Cout, S} 총 6비트로 위의 경우를 알려 줄 수 있는 flag bit를 만들었다. 더해서 10의 자리를 넘어가면 멀티플렉서를 이용해 4'b0110 을 두 번째 4bit adder에서 첫 번째 4bit adder의 결과와 더해 일의 자리 BCD 4bit를 나타냈고, -를 표시해야 하는 상황에선 4'b0000 - (첫 번째 4bit adder의 결과) = 첫 번째 4bit adder의 결과 를 이용해 일의 자리 BCD 를 표현했다. 이런 계산을 통해 일의 자리 segment는 따로 컨트롤 하지 않아도 계산 결과를 표현하게 된다. 하지만 10의 자리 segment는 위에 언급한 두 가지의 경우를 구분해서 표현해야한다. add일 때 십의 자리가 나오는 경우는 특별히 따로 처리하지 않고 그대로 보여주면 맞는 경우가 되지만 subtract 에서 - 가 나오는 경우는 컨트롤을 필요로한다. 만약 그대로 내보내면 -9 ~ -1 의 경우 밖에 없으므로 segment는 0을 표시할 것이다. 하지만 위에서 언급 한 flag bit를 멀티플렉서의 select bit로 사용하면 - 경우 일때 -를 표시할 수 있게 된다. 0의 not 을 취하면 - 가 되므로, 따라서 - 경우인 flag bit를 받으면 멀티플렉서에서 not을 연결한 decoder값을 출력하게 하였다. 마지막으로 input EQUAL 신호를 넣어 EQUAL (DFF의 CLK)을 눌러야 계산 결과가 segment 에 나타나게 하였다. input START신호는 activate low로 설정하여 START를 누르면 리셋이 되고 계산을 새로 시작할 수 있게 하였다.

<표 1-1>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Addn\_Sub | Cout | S[3] | S[2] | S[1] | S[0] | unsigned | 계산 결과 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 2 | 2 |
| 0 | 0 | 0 | 0 | 1 | 1 | 3 | 3 |
| 0 | 0 | 0 | 1 | 0 | 0 | 4 | 4 |
| 0 | 0 | 0 | 1 | 0 | 1 | 5 | 5 |
| 0 | 0 | 0 | 1 | 1 | 0 | 6 | 6 |
| 0 | 0 | 0 | 1 | 1 | 1 | 7 | 7 |
| 0 | 0 | 1 | 0 | 0 | 0 | 8 | 8 |
| 0 | 0 | 1 | 0 | 0 | 1 | 9 | 9 |
| 0 | 0 | 1 | 0 | 1 | 0 | 10 | 10 |
| 0 | 0 | 1 | 0 | 1 | 1 | 11 | 11 |
| 0 | 0 | 1 | 1 | 0 | 0 | 12 | 12 |
| 0 | 0 | 1 | 1 | 0 | 1 | 13 | 13 |
| 0 | 0 | 1 | 1 | 1 | 0 | 14 | 14 |
| 0 | 0 | 1 | 1 | 1 | 1 | 15 | 15 |
| 0 | 1 | 0 | 0 | 0 | 0 | 16 | 16 |
| 0 | 1 | 0 | 0 | 0 | 1 | 17 | 17 |
| 0 | 1 | 0 | 0 | 1 | 0 | 18 | 18 |
| 0 | 1 | 0 | 0 | 1 | 1 | 19 | X |
| 0 | 1 | 0 | 1 | 0 | 0 | 20 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | 21 | X |
| 0 | 1 | 0 | 1 | 1 | 0 | 22 | X |
| 0 | 1 | 0 | 1 | 1 | 1 | 23 | X |
| 0 | 1 | 1 | 0 | 0 | 0 | 24 | X |
| 0 | 1 | 1 | 0 | 0 | 1 | 25 | X |
| 0 | 1 | 1 | 0 | 1 | 0 | 26 | X |
| 0 | 1 | 1 | 0 | 1 | 1 | 27 | X |
| 0 | 1 | 1 | 1 | 0 | 0 | 28 | X |
| 0 | 1 | 1 | 1 | 0 | 1 | 29 | X |
| 0 | 1 | 1 | 1 | 1 | 0 | 30 | X |
| 0 | 1 | 1 | 1 | 1 | 1 | 31 | X |
| 1 | 0 | 0 | 0 | 0 | 0 | 32 | X |
| 1 | 0 | 0 | 0 | 0 | 1 | 33 | X |
| 1 | 0 | 0 | 0 | 1 | 0 | 34 | X |
| 1 | 0 | 0 | 0 | 1 | 1 | 35 | X |
| 1 | 0 | 0 | 1 | 0 | 0 | 36 | X |
| 1 | 0 | 0 | 1 | 0 | 1 | 37 | X |
| 1 | 0 | 0 | 1 | 1 | 0 | 38 | X |
| 1 | 0 | 0 | 1 | 1 | 1 | 39 | -9 |
| 1 | 0 | 1 | 0 | 0 | 0 | 40 | -8 |
| 1 | 0 | 1 | 0 | 0 | 1 | 41 | -7 |
| 1 | 0 | 1 | 0 | 1 | 0 | 42 | -6 |
| 1 | 0 | 1 | 0 | 1 | 1 | 43 | -5 |
| 1 | 0 | 1 | 1 | 0 | 0 | 44 | -4 |
| 1 | 0 | 1 | 1 | 0 | 1 | 45 | -3 |
| 1 | 0 | 1 | 1 | 1 | 0 | 46 | -2 |
| 1 | 0 | 1 | 1 | 1 | 1 | 47 | -1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 48 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 49 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 50 | 2 |
| 1 | 1 | 0 | 0 | 1 | 1 | 51 | 3 |
| 1 | 1 | 0 | 1 | 0 | 0 | 52 | 4 |
| 1 | 1 | 0 | 1 | 0 | 1 | 53 | 5 |
| 1 | 1 | 0 | 1 | 1 | 0 | 54 | 6 |
| 1 | 1 | 0 | 1 | 1 | 1 | 55 | 7 |
| 1 | 1 | 1 | 0 | 0 | 0 | 56 | 8 |
| 1 | 1 | 1 | 0 | 0 | 1 | 57 | 9 |
| 1 | 1 | 1 | 0 | 1 | 0 | 58 | X |
| 1 | 1 | 1 | 0 | 1 | 1 | 59 | X |
| 1 | 1 | 1 | 1 | 0 | 0 | 60 | X |
| 1 | 1 | 1 | 1 | 0 | 1 | 61 | X |
| 1 | 1 | 1 | 1 | 1 | 0 | 62 | X |
| 1 | 1 | 1 | 1 | 1 | 1 | 63 | X |

Case 1)

A = 0

B = 8

Addn\_Sub = 1'b0

결과 예상 : 08

seg10(abcdefg) : 7'b1111110

seg1(abcdefg) : 7'b1111111

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A close up of a device

Description automatically generated

Case 2)

A = 7

B = 3

Addn\_Sub = 1'b0

결과 예상 : 10

seg10(abcdefg) : 7'b0110000

seg1(abcdefg) : 7'b1111110

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A picture containing monitor

Description automatically generated

Case 3)

A = 6

B = 9

Addn\_Sub = 1'b0

결과 예상 : 15

seg10(abcdefg) : 7'b0110000

seg1(abcdefg) : 7'b1011011

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A close up of a device

Description automatically generated

Case 4)

A = 0

B = 0

Addn\_Sub = 1'b0

결과 예상 : 00

seg10(abcdefg) : 7'b1111110

seg1(abcdefg) : 7'b1111110

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A picture containing building, sitting, monitor, red

Description automatically generated

Case 5)

A = 0

B = 8

Addn\_Sub = 1'b1

결과 예상 : -8

seg10(abcdefg) : 7'b0000001

seg1(abcdefg) : 7'b1111111

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A close up of a building

Description automatically generated

Case 5)

A = 7

B = 4

Addn\_Sub = 1'b1

결과 예상 : 30

seg10(abcdefg) : 7'b1111110

seg1(abcdefg) : 7'b1111001

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A close up of a computer

Description automatically generated

Case 7)

A = 2

B = 7

Addn\_Sub = 1'b1

결과 예상 : -5

seg10(abcdefg) : 7'b0000001

seg1(abcdefg) : 7'b1011011

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A close up of a computer

Description automatically generated

Case 8)

A = 0

B = 0

Addn\_Sub = 1'b1

결과 예상 : 00

seg10(abcdefg) : 7'b1111110

seg1(abcdefg) : 7'b1111110

EQUAL rising(0->1) 과 함께 계산 결과가 segment에 제대로 표시 되는 것을 확인 할 수 있다.

Start Falling(1->0) 과 함게 segment 가 00을 표시(리셋) 되는 것을 확인 할 수 있다.

A picture containing sitting, monitor, window, black

Description automatically generated

Bread Board Simulation

A close up of a device

Description automatically generated

Addn\_Sub

seg\_10

빨강 - a 파랑 - b 초록 - c 노랑 - d

하늘 - e 핑크 - f 주황 - g

seg\_1

equal

clk start

1 2 3 4 5 6 7 8 9 0

동작의 약간 차이가 있지만 동작 원리는 같다.

사용 방법 :

0. Add 는 스위치를 왼쪽에 Subtract는 스위치를 오른 쪽에 둔다.

1. A를 누른 상태로 둔다.

2. clk 스위치를 두번 클릭하여 펄스를 만든다.

3. A를 풀고, B를 누른 상태로 둔다.

4. clk 스위치를 두번 클릭하여 펄스를 만든다.

5. equal 버튼을 누른 상태로 둔다. (이 때 결과가 나옴)

6. start 를 눌러 리셋을 함.

7. 1 ~ 6 반복

Case 1)

A = 7

B = 8

Addn\_Sub = 1'b0

결과 예상 : 15

seg10(abcdefg) : 7'b0110000 (파랑 - b, 초록 -c)

seg1(abcdefg) : 7'b1011011 (5)

A picture containing clock

Description automatically generated

Case 2)

Start 누른 후

seg10(abcdefg) : 7'b1111110 (주황 - g 만 0)

seg1(abcdefg) : 7'b1111110 (0)

A picture containing clock

Description automatically generated

Case 3)

A = 3

B = 4

Addn\_Sub = 1'b0

결과 예상 : 07

seg10(abcdefg) : 7'b1111110 (주황 - g 만 0)

seg1(abcdefg) : 7'b1110000 (7)

A picture containing clock

Description automatically generated

Case 4)

A = 1

B = 2

Addn\_Sub = 1'b1

결과 예상 : -1

seg10(abcdefg) : 7'b0000001 (주황 - g 만 1)

seg1(abcdefg) : 7'b0110000 (1)

A picture containing clock

Description automatically generated