








Computer Architectrue

~ LAB 07 ~

2013210111 남세현

목표 :

4bit ALU를 만드는 과제입니다.

 /tb4alu/A	0011	0000	0101	1010	0101	1010	0111	0100	0101	0100	0111
 /tb4alu/B	0111	0000	1111		0000		0101	0010		0101	0110
 /tb4alu/CIN	1										
 /tb4alu/BINV	1										
 /tb4alu/OP	11	00			01		10				11
 /tb4alu/COUT	0										
 /tb4alu/Y	0000	0000	0101	1010	0101	1010	1100	0111	0011	1111	0000

실행화면

구현 방법

총 4가지의 연산이 있습니다.

b'00 : AND

b'01 : OR

b'10 : Addition

b'11 : SLT

4bit ALU 모듈을 만들기 위해, 1bit ALU 모듈을 만들었습니다.

1bit ALU는 Lab 07 조교 PPT의 맨 마지막 장 썸에 나와있는 예제를 참고하여 만들었습니다.

4bit ALU 모듈은 1bit ALU 모듈 4개를 가지고 연결하였습니다.

SLT의 경우,

$A == B : \text{Carry Out} = \text{Carry In}$

$A < B : \text{Carry Out} = 0$

$A > B : \text{Carry Out} = 1$

이 되도록 하였으며, 맨 첫 비트만 강제로 Result가 맨 마지막 비트의 Carry Out값이 되도록 하였습니다.

이외 과제 시 어려운 점은 없었고, 아래 코드를 첨부하도록 하겠습니다.

코드

[alu1bit.v]

```
module alu1bit(A, B, CarryIn, Binvert, Operation, Less, Result, CarryOut);
```

```
    input A, B, CarryIn, Binvert;
```

```
    input [1:0]Operation;
```

```
    input Less;
```

```
    output Result, CarryOut;
```

```
    reg tempResult, tempCarryOut;
```

```
    reg tempB;
```

```
    always @(A, B, Operation, CarryIn, Binvert, Less)
```

```
    begin
```

```
        tempCarryOut = 0;
```

```
        case (Binvert)
```

```
            1:    tempB = !B;
```

```
            0:    tempB = B;
```

```
        endcase
```

```
        case (Operation)
```

```
            2'b00: tempResult = A & tempB;
```

```
            2'b01: tempResult = A | tempB;
```

```
            2'b10: {tempCarryOut, tempResult} = A + tempB + CarryIn;
```

```
            2'b11:
```

```
                begin
```

```
                    tempCarryOut = ( ( A ~^ tempB ) & CarryIn | ( A ^ tempB )
```

```
                    & A );
```

```

        tempResult = Less;
    end
endcase
end
assign Result = tempResult;
assign CarryOut = tempCarryOut;
endmodule;

```

[alu4bit.v]

```

`include "alu1bit.v"

```

```

module alu4bit(A, B, CarryIn, Binvert, Operation, CarryOut, Result);
    input [3:0] A, B;
    input CarryIn, Binvert;
    input [1:0] Operation;
    output wire [3:0] Result;
    output wire CarryOut;

    wire LessWire;
    wire [2:0] CarryOutWire;

    alu1bit alu1_0 ( A[0], B[0], CarryIn, Binvert, Operation, LessWire, Result[0],
CarryOutWire[0]);
    alu1bit alu1_1 ( A[1], B[1], CarryOutWire[0], Binvert, Operation, 0, Result[1],
CarryOutWire[1]);
    alu1bit alu1_2 ( A[2], B[2], CarryOutWire[1], Binvert, Operation, 0, Result[2],
CarryOutWire[2]);
    alu1bit alu1_3 ( A[3], B[3], CarryOutWire[2], Binvert, Operation, 0, Result[3], LessWire);

    assign CarryOut = LessWire;
endmodule;

```