



디지털논리회로실습

Lab 8. 실습설명

2017. 11. 29

HANYANG University

Part II

8-bit Signed adder

- 두 개의 8-bit Signed Number를 더하는 회로 설계

- 조건

- SW15~8, SW7~0 으로 2개의 8bit 입력을 받음
- SW15, SW7은 부호를 의미함 (1은 음수, 0은 양수)
- HEX7~6, HEX5~4 각각은 입력 값에 따른 2자리 16진수를 표현하며, HEX3~1에 계산 결과 및 부호를 표시함
- KEY0을 누르면 모든 HEX를 0으로 초기화
- KEY1을 누르면 모든 HEX에 입력 값 및 결과 값이 표시됨
- Overflow가 발생 되면 LEDG7에 표시함
- 추가: sign-bit를 읽어 양수일 때 HEX3 Off, 음수일 때 - 표시가 나오도록 함

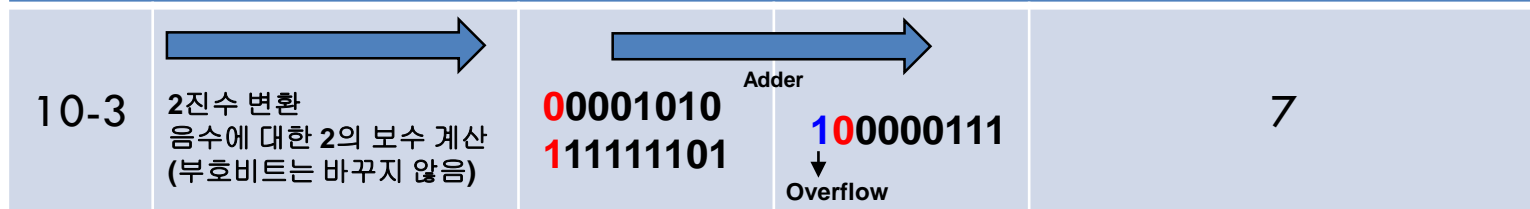
Part II

8-bit Signed adder

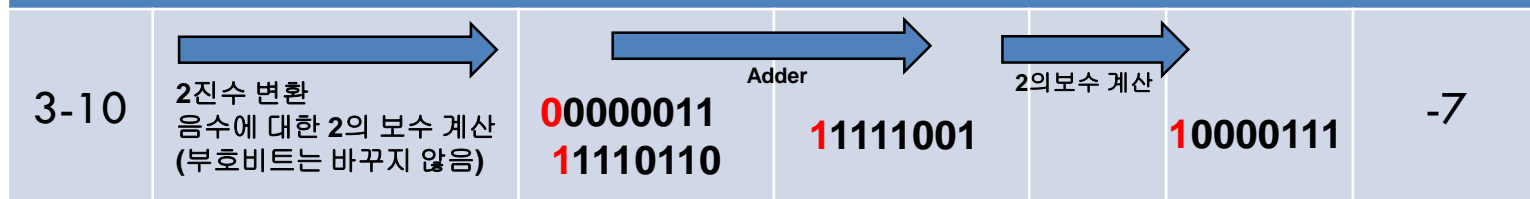
■ 감산기

- 1. 2의 보수를 계산한 결과를 Full Adder를 통해 더함
- 2. 최종 결과가 음수인 경우 최종결과의 2의 보수를 구함

최종결과가 양수인 예시



최종결과가 음수인 예시



Task and Function

- 같은 기능을 여러 번 작성할 경우에 Task 혹은 Function을 통해 같은 작업을 여러 번 호출해서 사용할 수 있음

Task	Function
<ul style="list-style-type: none">• 하나의 Task 내에서 다른 Task 와 Function 모두 호출 가능	<ul style="list-style-type: none">• 다른 Function 호출은 가능하지만 Task 호출은 불가능
<ul style="list-style-type: none">• Non-zero 시뮬레이션 타임에 실행 될 수 있음	<ul style="list-style-type: none">• 항상 0 시뮬레이션 타임에 실행
<ul style="list-style-type: none">• 하나 이상의 input, output 등의 인수를 가질 수 있으며 인수를 하나도 갖지 않을 수 있음	<ul style="list-style-type: none">• 하나 이상의 Input 인수를 가질 수 있으나 Output 인수를 가질 수 없음• (자기이름으로 된 하나의 return 값만 output 가능)
<ul style="list-style-type: none">• Local variables, reg, time variable, integer, real 등의 값을 사용 할 수 있음• Wire 는 가질 수 없음• Always, initial 구문을 가질 수 없음	

Task and Function

Task 와 Function의 사용

Task	Function
<ul style="list-style-type: none">• Delay, timing, event 등의 제어 구조가 있는 경우• Output이 없거나 하나 이상의 Output이 있는 경우• Input이 없는 경우	<ul style="list-style-type: none">• Delay, timing, event 등의 제어 구조가 없는 경우• 단 한 개의 값을 return을 하는 경우• 적어도 한 개의 Input이 있는 경우

Part II

Task and Function

Task 예제

```
// define task
task Task이름;
input 변수1,2...; //Input 변수 정의
output 변수1,2...; //Output 변수 정의;

Task의 동작을 정의

endtask
```

Syntax

module operation;

```
parameter delay = 10;
reg [15:0] A, B;
reg [15:0] AB_AND, AB_OR, AB_XOR;
```

```
always @(A or B) begin
    bitwise_oper(AB_AND, AB_OR, AB_XOR, A, B);
end
```

```
// define task
task bitwise_oper;
output [15:0] ab_and, ab_or, ab_xor;
input [15:0] a, b;
begin
    #delay ab_and = a & b;
    ab_or = a | b;
    ab_xor = a ^ b;
end
endtask
```

endmodule

Example

Task and Function

Function 예제

```
//define function  
function Function 이름;  
input 변수1,2...; //Input 변수 정의;  
  
Function 동작 정의  
Function 이름=return 값 전달;  
endfunction
```


Syntax

```
module notgate;  
    reg in_data;  
    reg out_data;  
  
    always @(in_data) begin  
        out_data=calc_not(in_data);  
    end  
  
    // define function  
    function calc_not;  
    input data;  
    begin  
        calc_not=~data;  
    end  
    endfunction  
  
    endmodule
```




Example

4 bit Multiplier

- 16진수 2개를 곱하는 4bit Multiplier

 $P = A \times B$


- 조건

-  SW11~8, SW3~0 으로 16진수 숫자 2개(A와 B) 입력
-  A는 HEX6에 B는 HEX4에 각각 출력
-  곱셈의 결과 P 는 HEX1~0 에 출력






Part VI

Sum of Multiplications

- $S = (A \times B) + (C \times D)$ 를 계산하는 모듈을 설계

-  카르노 맵에서 출력 Y 에 대한 식을 계산하는 방법 중 곱의합 (Sum Of Product:SOP)을 계산하는 모듈임

- 조건

-  SW15~8로 A와 C를 입력, SW7~0으로 B와 D를 입력(16진수 입력)
-  SW16은 A,C와 B,D 의 입력을 선택
-  SW17이 1일 때 실제 값이 입력됨, 0일때 값이 입력되지 않음
-  A,C는 HEX7~6에 표시, B,D는 HEX5~4에 표시, 계산 결과는 HEX3~0에 표시
-  만약 Carry 가 발생하면 LEDG8에 불이 들어 오도록 설계