설계과제 HW3

\* 각 문제는 divider에 학번을 적은 waveform을 저장한 뒤 해당 프로젝트 폴더 전체를 압축하여 제출하세요 (문서에 코드 첨부하지 말 것).

1. Full adder 모듈을 이용하여 4-bit adder/subtractor를 설계하고 15 + 15, 0 – 15, 4 + 11, 7 – 12의 결과를 시뮬레이션하여 waveform에 출력하세요. 모듈의 정의는 다음과 같습니다. 필요시 출력을 reg 타입으로 정의하세요.

module adder\_subtractor\_4bits (a, b, k, cout, s);

input [3:0] a, b;

input k;

output cout;

output [3:0] s;

…

endmodule

1. Verilog를 사용하는 가장 큰 이유 중 한가지는 동작적 수준(behavioral level)에서 회로의 동작을 기술하여도 합성을 통해 원하는 디지털 시스템을 구현할 수 있다는 점입니다. 예를 들어 1-bit의 adder는 다음과 같이 기술할 수 있습니다.

module adder (a, b, cout, s);

input a, b;

output cout, s;

reg [1:0] sum;

assign cout = sum[1];

assign s = sum[0];

always @(\*) sum = #1 a + b;

endmodule

module adder (a, b, cout, s);

input a, b;

output cout, s;

wire [1:0] sum;

assign cout = sum[1];

assign s = sum[0];

assign #1 sum = a + b;

endmodule

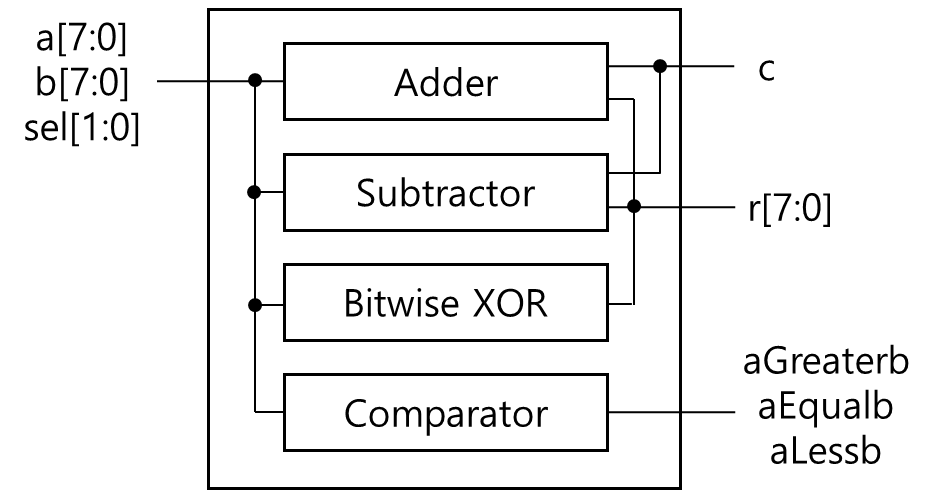
이와 같이 +/- 연산자를 이용하여 동작적 수준에서 Parameterized 8-bit adder/subtractor를 설계하고 128 + 200, 128 – 200, 157 + 32, 124 – 24에 대해 시뮬레이션 하세요. 모듈의 정의는 다음과 같습니다.

module adder\_subtractor\_8bits (a, b, k, cout, s);

…

endmodule

1. 8-bit의 입력 a, b와 2-bit의 입력 sel을 받고 sel 신호에 따라 다른 submodule의 연산결과를 출력하는 회로를 설계하세요. sel 신호가 2’b00이면 adder, 2’b01이면 subtractor, 2’b10이면 a와 b의 bitwise XOR 연산을 수행, 2’b11이면 comparator 연산을 합니다. 회로의 block diagram과 출력은 아래와 같습니다.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 출력 | sel[1:0] | | | |
| 00 | 01 | 10 | 11 |
| c | 연산결과 | 연산결과 | 0 | 0 |
| r[7:0] | 연산결과 | 연산결과 | 연산결과 | 8h’00 |
| aGreaterb | 0 | 0 | 0 | 연산결과 |
| aEqualb | 0 | 0 | 0 | 연산결과 |
| aLessb | 0 | 0 | 0 | 연산결과 |