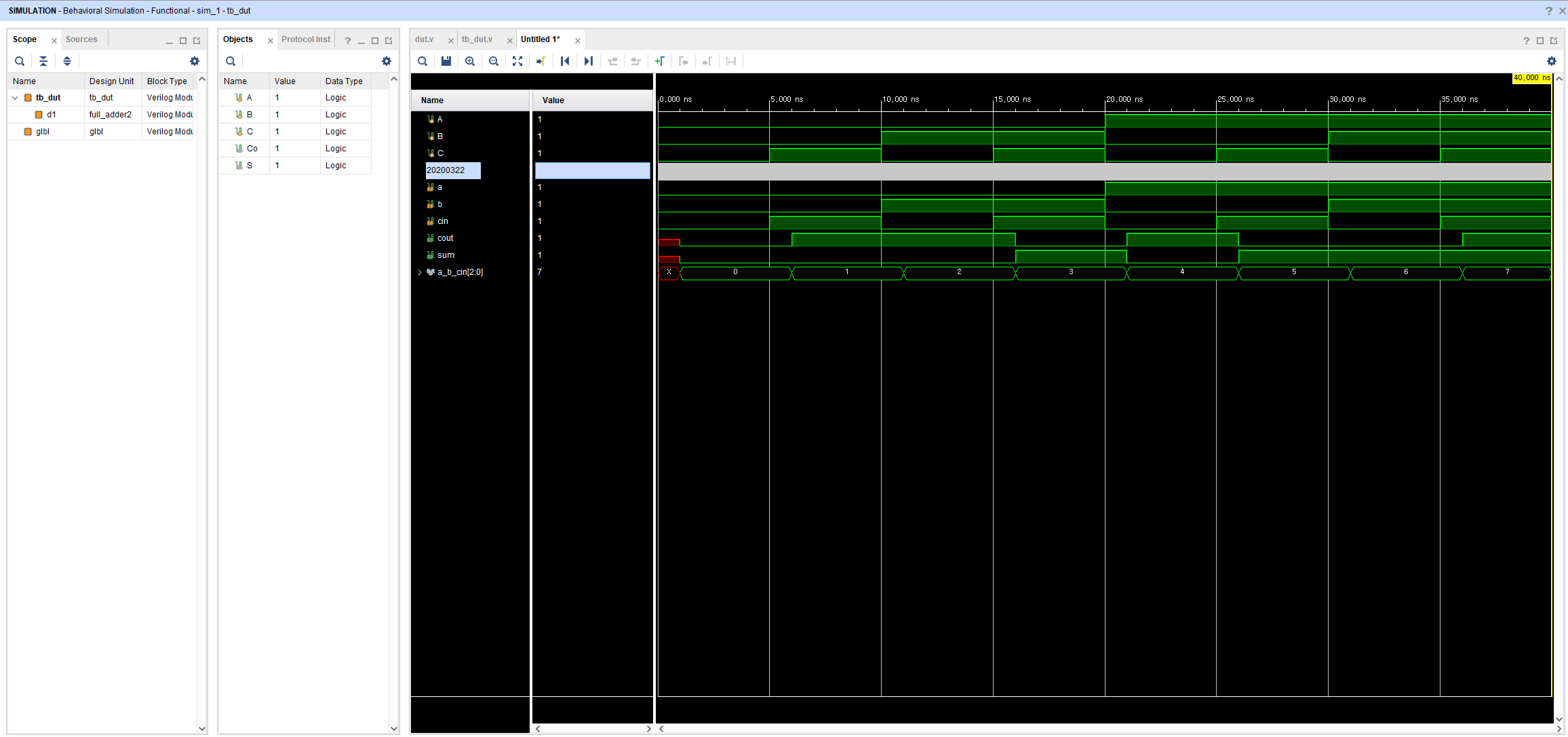
설계과제 HW2

학번 및 이름:

\* 문제 3, 4, 5번은 같은 testbench를 사용할 수 있습니다. 1, 2번의 답을 자필로 적어 스캔하여 제출하고, 나머지는 설계한 모듈과 testbench의 verilog 코드를 첨부하고 각 시뮬레이션 결과를 모든 입력 case를 커버할 수 있도록 capture하여 스캔본과 한 파일에 첨부하여 제출하세요. 이때, divider에 학번을 적으세요.



1. Full adder의 진리표를 완성하세요.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | Cout | Sum |
| 0 | 0 | 0 |  |  |
| 0 | 0 | 1 |  |  |
| 0 | 1 | 0 |  |  |
| 0 | 1 | 1 |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 |  |  |
| 1 | 1 | 0 |  |  |
| 1 | 1 | 1 |  |  |

1. Full adder의 Cout과 Sum에 대해 카르노 맵을 그리고 간소화된 논리식을 적으세요.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B,Cin  A | 00 | 01 | 11 | 10 |
| 0 |  |  |  |  |
| 1 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B,Cin  A | 00 | 01 | 11 | 10 |
| 0 |  |  |  |  |
| 1 |  |  |  |  |

Cout =

Sun =

1. Full adder의 논리식을 continuous assignment 통해 설계하고 모든 입력을 커버하는 testbench를 작성하여 시뮬레이션하세요. 단, 1 time unit delay를 줄 것.

module full\_adder1 (a, b, cin, cout, sum);

input a, b, cin;

output cout, sum;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

endmodule

1. 다음과 같이 A, B, Cin을 concatenate하여 a\_b\_cin이 정의했을 때 case 문을 이용하여 full adder module을 설계하고 모든 입력을 커버하는 testbench를 작성하여 시뮬레이션하세요.

module full\_adder2 (a, b, cin, cout, sum);

input a, b, cin;

output reg cout, sum;

wire [2:0] a\_b\_cin;

assign #1 a\_b\_cin = {a, b, cin};

always \_\_\_\_\_\_\_\_\_\_ begin

case (\_\_\_\_\_\_\_)

3’b000 : begin

\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_

end

…

endcase

end

endmodule

1. Hierarchical design은 submodule을 이용한 설계를 의미합니다. Half adder module을 submodule로 사용하는 full adder를 설계하고 모든 입력을 커버하는 testbench를 작성하여 시뮬레이션하세요. 이때 half adder는 procedural assignment를 이용하여 설계하고 full adder는 continuous assignment를 이용하여 설계하세요.

module full\_adder3 (a, b, cin, cout, sum);

input a, b, cin;

output cout, sum;

…

endmodule