6주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20221549 이름: 김효림

**1.**

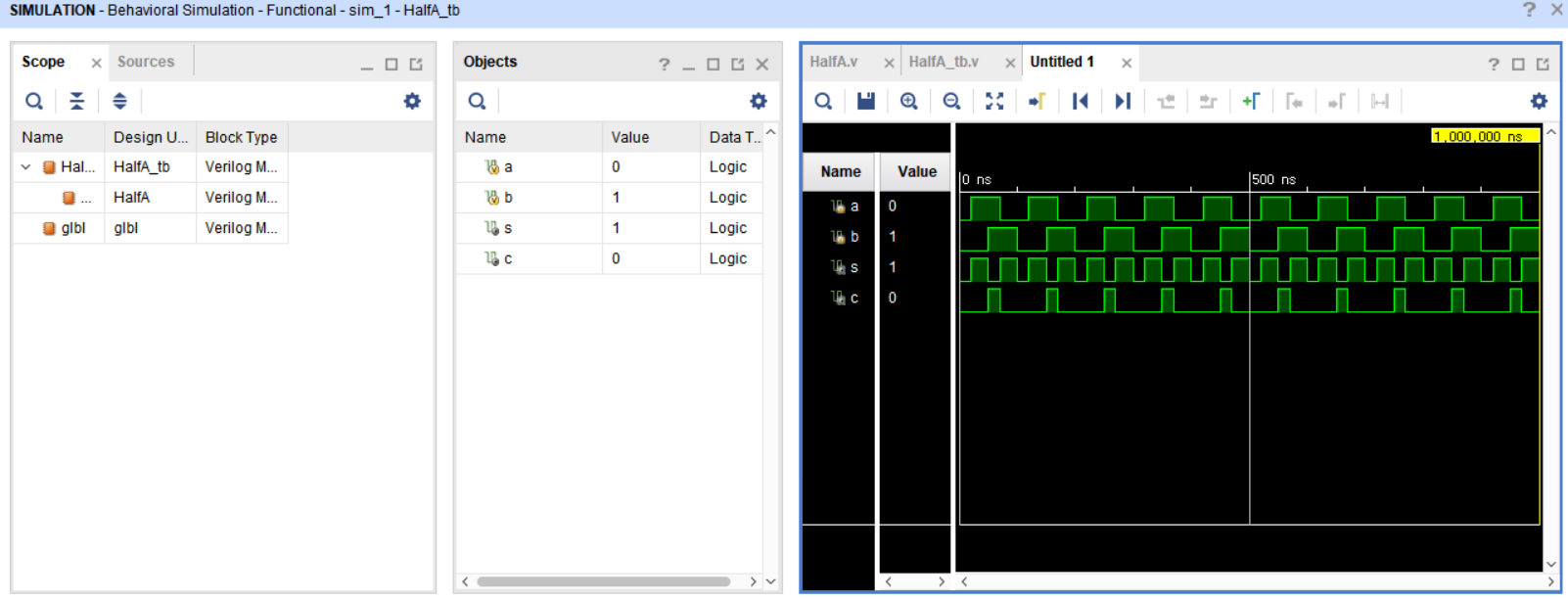
.................

전가산기, 반가산기, 전감산기, 반감산기 및 부호 변환기의 개념을 이해하고, Verilog를 사용하여 다양한 Adder와 Subtractor, Code converter를 구현하고 FPGA 보드를 활용하여 구현된 회로의 동작을 확인한다.

................

**2.**

.......................

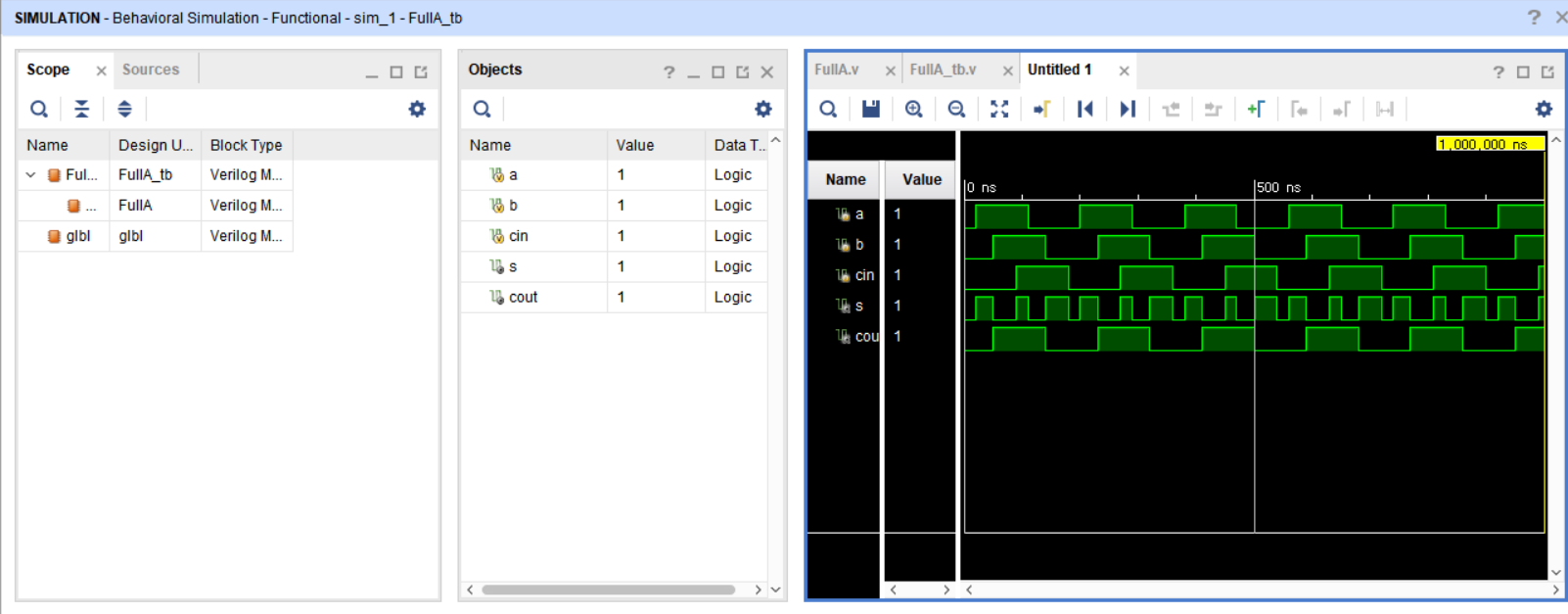


(반가산기 simulation)

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

(반가산기 진리표)

반가산기는 2개의 입력 신호를 받아, 그 두 개를 더한 결과값과 Carry bit를 설정하는 회로이다. 진리표를 확인해 보면, S의 값은 입력 신호가 다른 경우 참, 같은 경우 거짓이 나옴을 확인할 수 있다. 즉 S값은 두 입력 신호 A와 B의 XOR 연산 값이 들어간다. Carry bit의 경우 두 입력이 모두 참인 경우에만 참이 나옴을 확인할 수 있다. 즉, Carry bit는 A와 B의 AND연산 값이 들어간다는 걸 확인할 수 있다.



(전가산기 simulation)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| C\_in | B | A | S | C\_out |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

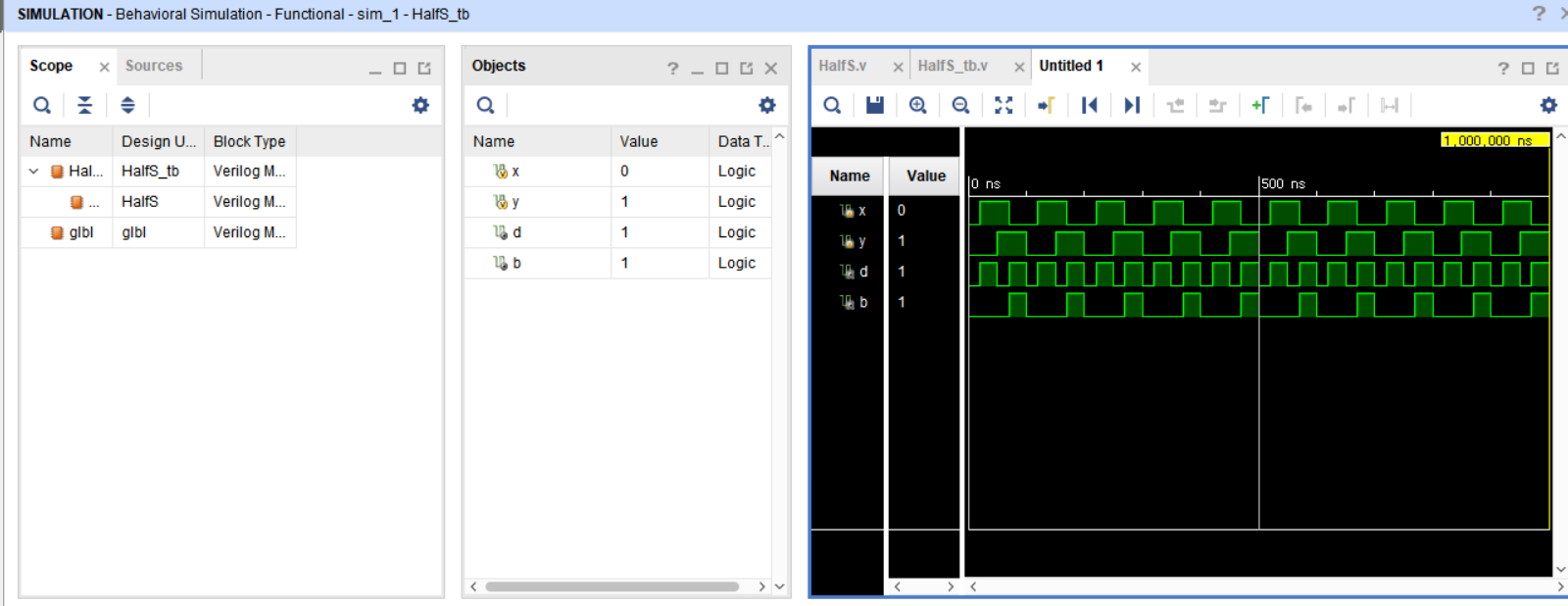
(전가산기 진리표)

전가산기는 두 입력 신호 A와 B 그리고 Carry\_in bit를 입력 받는다. 진리표를 확인해 보면, 결과 비트(S)는 입력에서 1 bit의 개수가 홀수 개일 때 참이 되고, Carry\_out bit는 세 입력 중 논리적 참의 개수가 2개 이상인 경우 1이 참이 됨을 알 수 있다. 즉 S는 A와 B와 C\_in의 XOR 연산 결과가, C\_out은 (A AND B) OR (A XOR B) AND C\_in 연산 결과가 들어가는 것을 확인할 수 있다.

.........................

**3.**

.......................

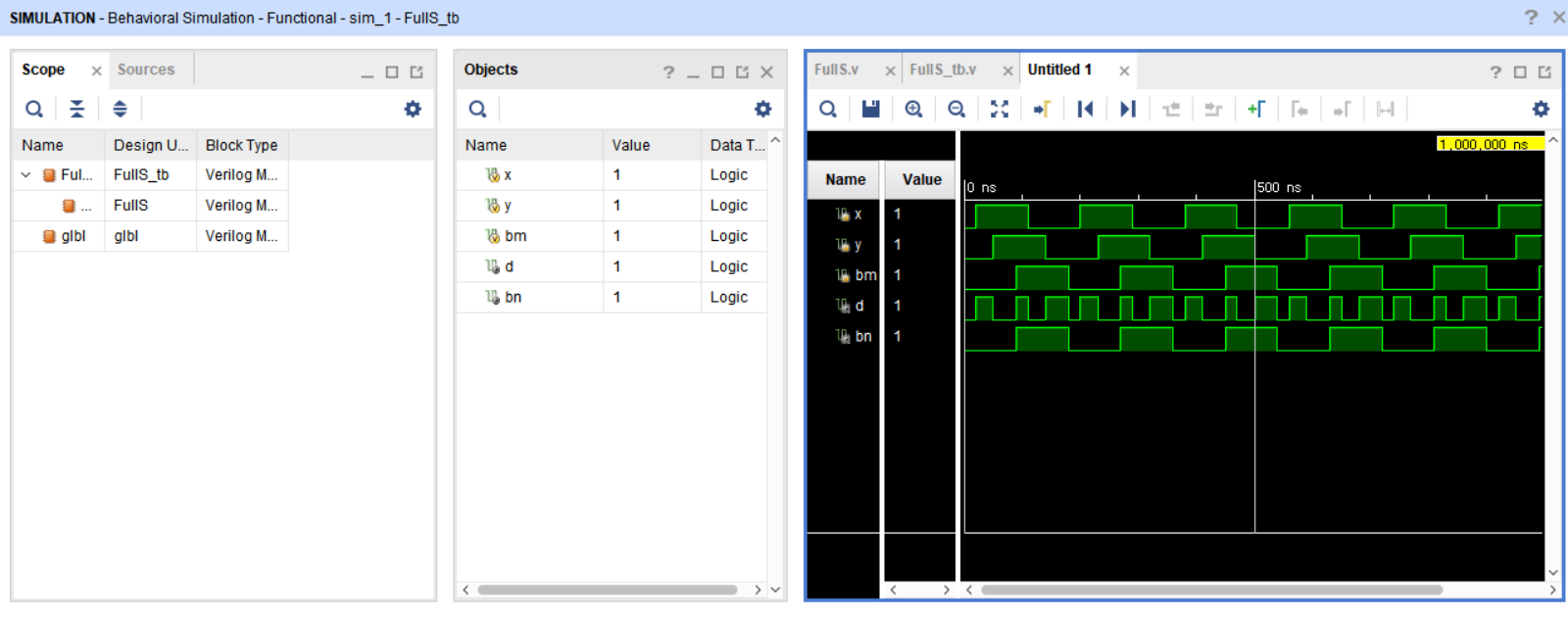


(반감산기 simulation)

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| A | B | b | D |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

(반감산기 진리표)

반감산기는 두 개의 입력 신호를 받아 결과값과 받아내림을 output으로 설정하는 회로이다. Difference(차이, 결과 bit)의 경우 두 입력 신호가 다를 경우에만 참이 나오고, Borrow(받아내림) 비트는 A가 0이고, B가 1인 경우에만 참이 나온다. D는 A와 B의 XOR 연산 결과가, b는 A’와 B의 AND연산 결과임을 알 수 있다.



(전감산기 simulation)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| A | B |  | D |  |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

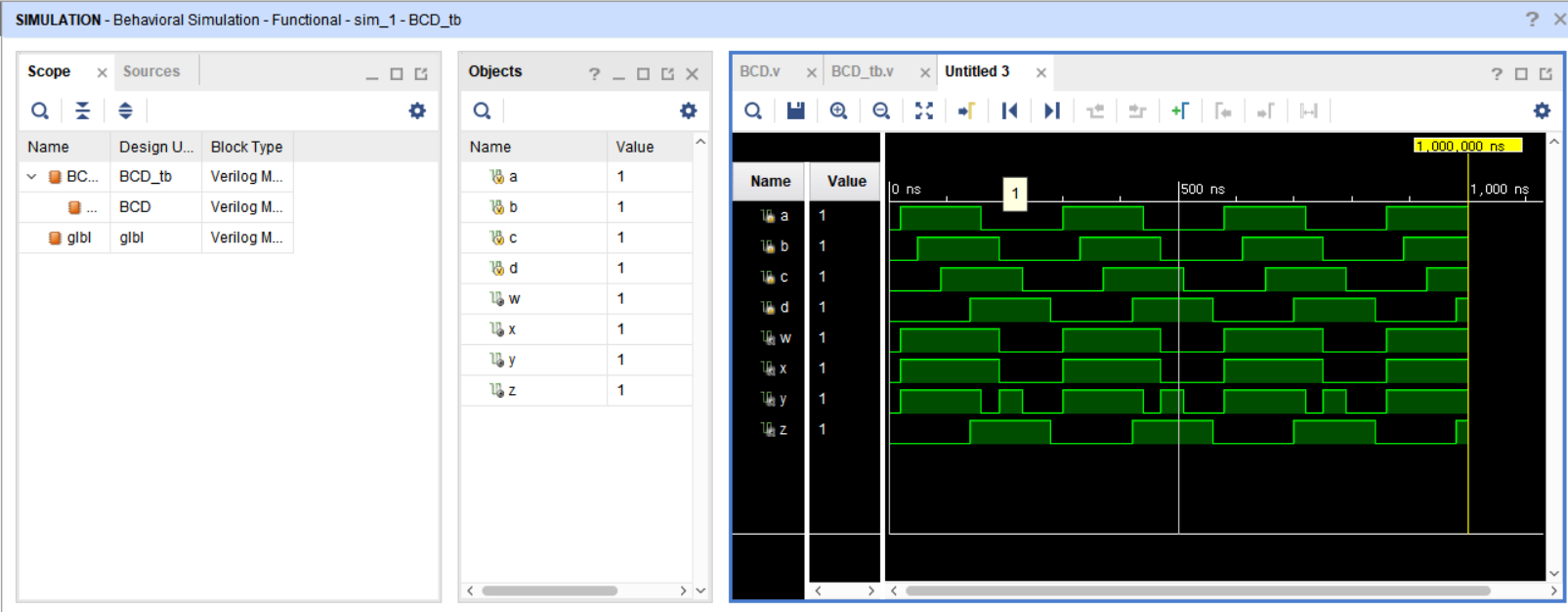
(전감산기 진리표)

전감산기는 두 입력 A, B 외에 받아내림 비트를 입력받아 결과가 계산된다. 진리표를 보면, D의 경우 입력 값 중 1의 개수가 홀수 개인 경우 참인 것을, 의 경우 A가 0이고 B와 중 하나가 1이거나 둘 다 1인 경우, A가 1이고 나머지 두 입력값이 모두 1인 경우 참이 된다. 즉, D = X XOR Y XOR B\_in, B\_out = A’ AND Y OR (X XOR Y)’ AND B\_in임을 확인할 수 있다.

.........................

**4.**

.......................



(Code Converter simulation)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| W | X | Y | Z | A | B | C | D |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

(Code Converter 진리표)

8421 표기법은 0부터 9까지 표기할 수 있다. 이를 바탕으로 카르노맵을 그려보면 다음과 같은 결과를 얻을 수 있다. 참고로 10~15의 결과값은 동작에 영향을 주지 않기 때문에, 카르노 맵을 그릴 때 don’t care임을 생각하며 그리면 된다.

텍스트, 도표, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

이를 바탕으로 각각의 결과들을 SOP, POS 형태로 나타내면 다음과 같다.

(POS, SOP 동일)

SOP 형태는 1로 표기된 것들을 그룹화, POS 형태는 0으로 표기된 것들을 그룹화한 후 드 모르간의 법칙을 활용하여 Boolean 식을 구할 수 있다.

.........................

**5.**

.......................

전가산기, 반가산기, 전감산기, 반감산기를 Verilog로 구현하면서 숫자를 계산하는 시스템을 만들 수 있었다. 외에도 이전 시간에 다뤘던 카르노 맵을 직접 그려보며 진리관계를 판단하고, 불 대수식을 쉽게 찾는 방법을 실습할 수 있었다.

.........................

**6.**

.......................

NAND gate를 이용하여 code converter를 구현하면 다음과 같은 시뮬레이션 결과를 얻을 수 있다. 즉, 회로를 NAND 게이트만을 사용하여 구현할 수 있음을 다시 한 번 확인해 볼 수 있다.

<BCD.v code>

module BCD(

input a,b,c,d,

output w,x,y,z

);

assign w = ~(~a & ~(b & c) & (b & d));

assign x = ~(~a & ~(b & ~c & ~d) & ~(b & c));

assign y = ~(~a & ~(~b & c) & ~(b & ~c & d));

assign z = d;

endmodule

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

.........................