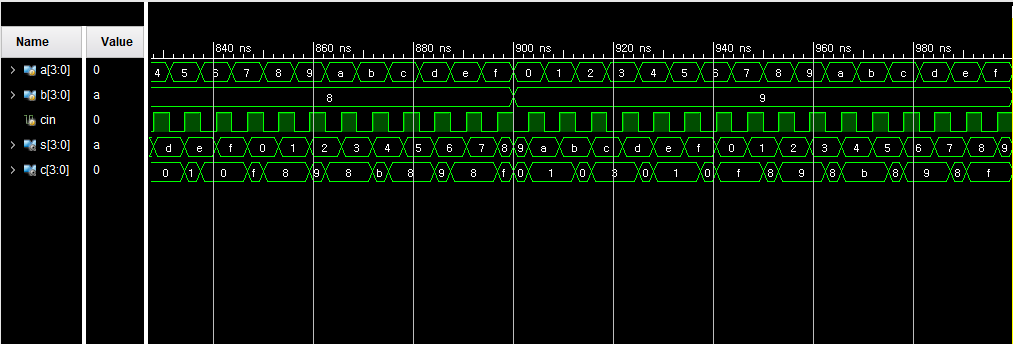
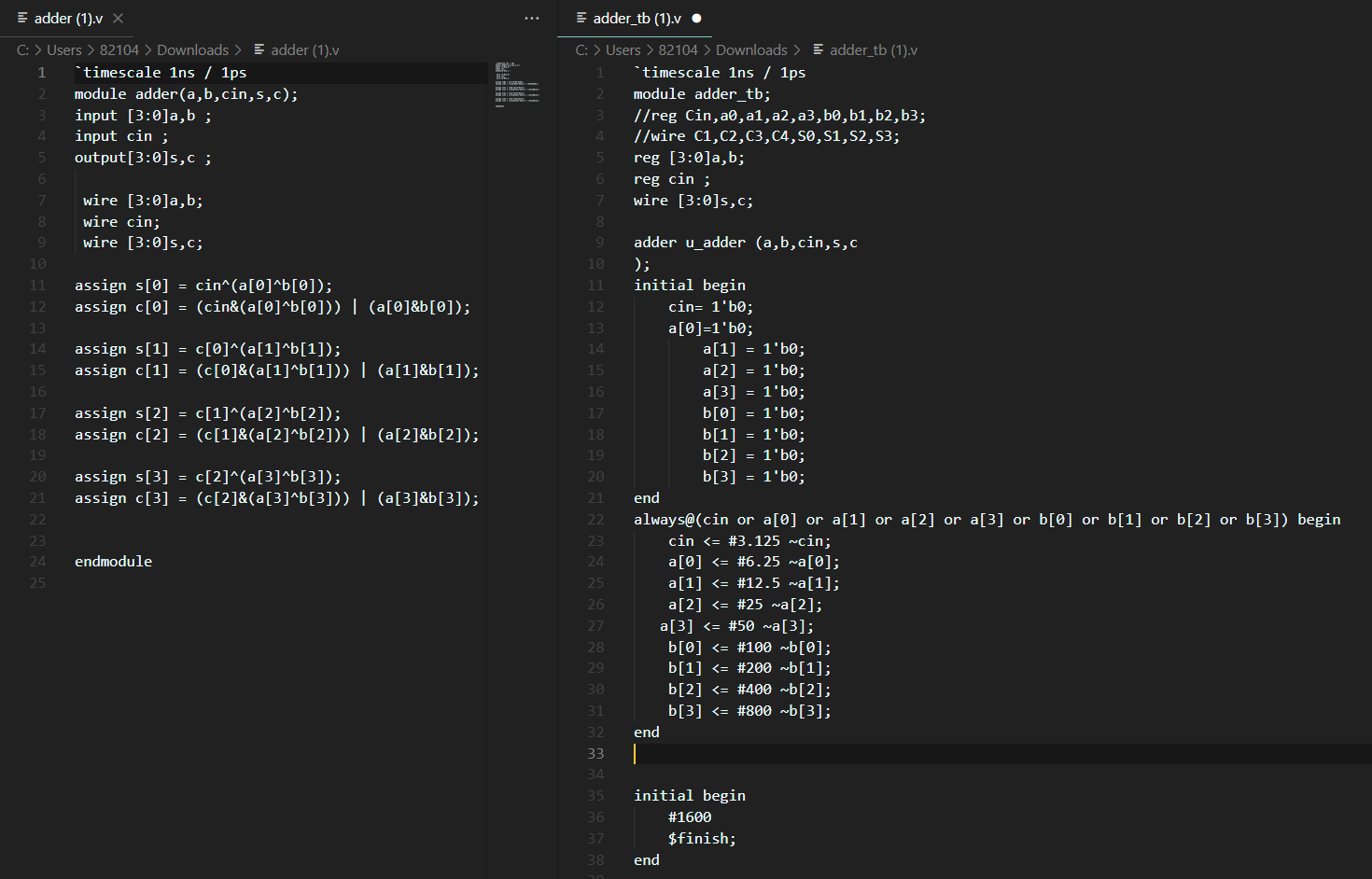
10주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191612 이름: 윤기웅

1.

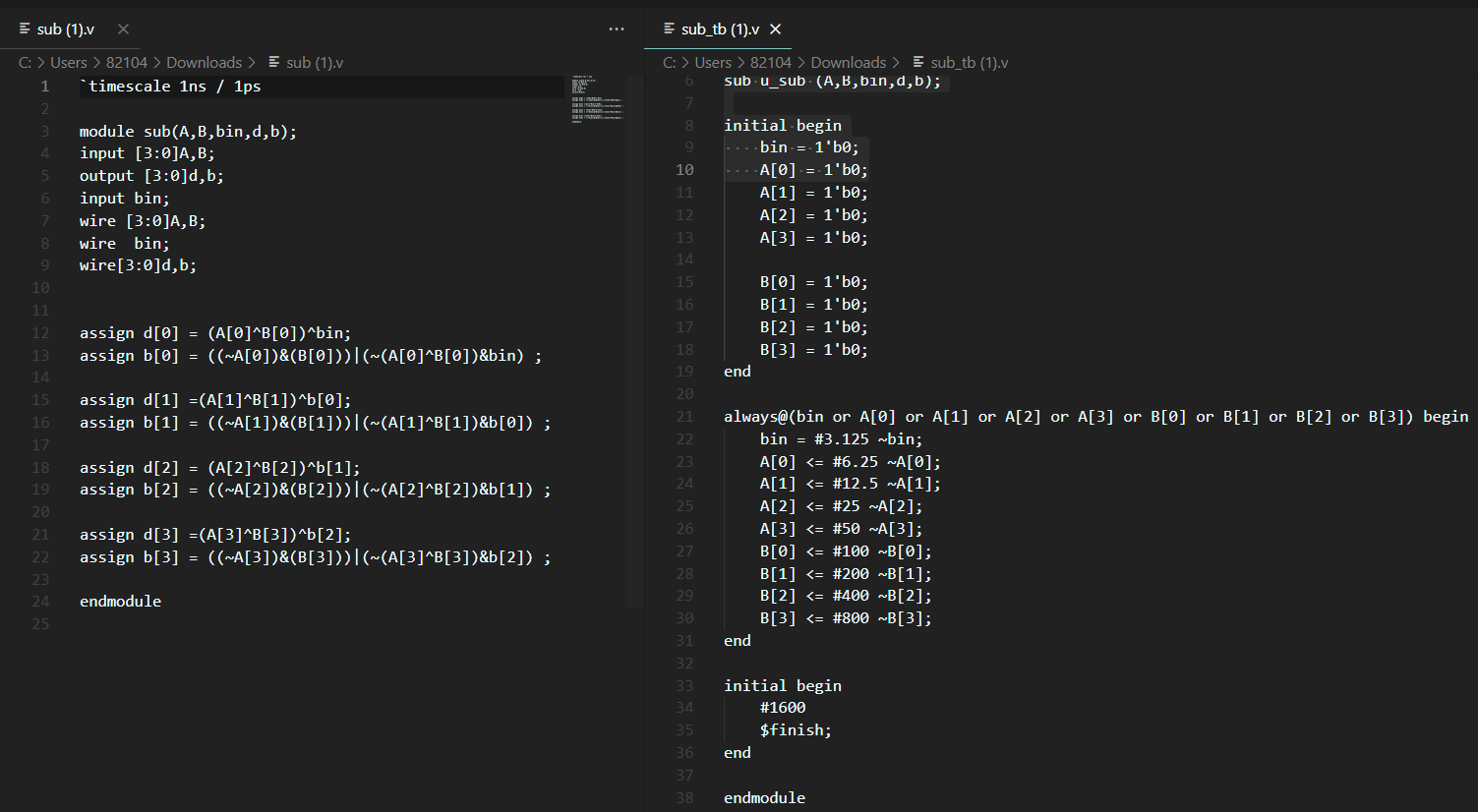
이는 1비트의 full adder를 4개를 연결한 것이라서 각 자리수마다 계산 시 발생하는 sum, carry를 다음 자리의 연산에 사용하면 된다. 즉, sum은 결과의 값을 갖고 carry는 다음 자리의 carry in 의 역할을 하게 되는 것이다. 덧셈을 a0, b0, carryin을 갖고 한다고 생각하면 3개의 입력 중에서 1의 개수가 1개이거나 3개인 경우에 결과 값이 1이 되고 그 나머지의 경우는 0이 된다. 그 다음 자리의 carry in으로 들어가는 carry는 a와 b가 모두 1인 경우나 아님 carry in이 1이고 a와 b의 xor연산이 1인 경우에 1이 된다. 주어지는 입력이 모든 경우의 수에 대해서 조사해보아야 하니 test bench의 주기를 이전보다 더 길게 설정한다.

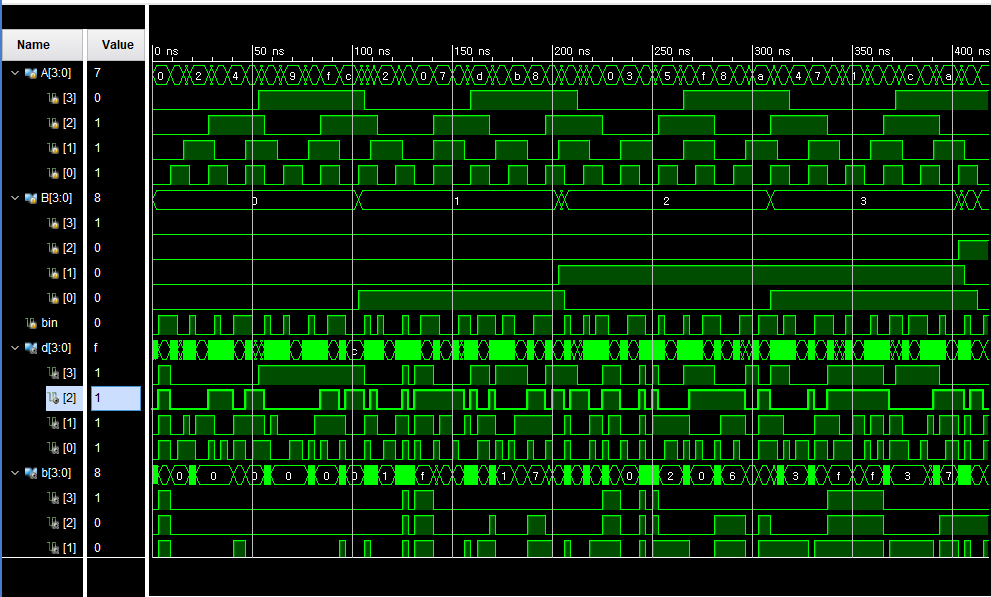




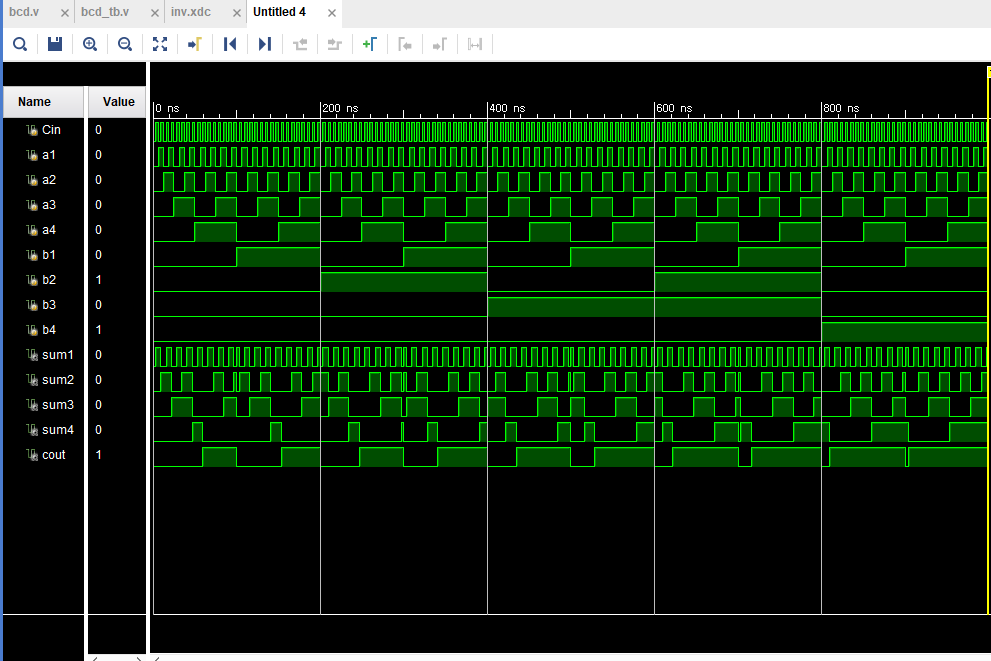
2.

4비트 binary parallel sub는 1비트의 full sub를 4개를 연결해서 만들 수 있다. 각 자리에서 발생하는 d는 결과값이 되고 borrow를 의미하는 b는 다음 자리 뺄셈의 borrow로 전달해준다. 결과값을 계산하는 경우 borrow in, a,b의 xor 연산을 하기 때문에 3개의 입력 중에서 1개의 1을 갖거나 3개의 1을 갖는 경우에 결과값이 1이 되고 그 이외의 경우에는 0의 결과를 갖게 된다. borrow를 구하는 경우에는 a'와 b가 모두 1인 경우 혹은 borrow in과 a xor b의 보수가 모두 1인 경우에 borrow가 1이 된다. 이 borrow는 또 다음 자리의 연산에 영향을 주게 된다. 이번에도 입력이 다른 경우보다 많기 때문에 testbench의 주기를 이전보다 길게 설정 해준다.





3. bcd의 연산을 4비트의 두 수 덧셈을 수행하면 된다. 4비트의 수 a와 b가 입력으로 들어오고 sum을 결과값으로 정해준다. 연산을 하고 결과가 9보다 큰 경우는 0110을 더해준다. 이를 보상회로를 통해서 만들어줘야 한다.



4.

처음에는 뺄셈 연산을 구현할 때 처음 borrow in을 어떻게 써야 하는지 잘 몰라서 이를 알아보는 데 많이 시간이 걸렸던 거 같다. 또한 이전의 실험보다 입력이 많이 들어와서 그런지 베릴로그 구현이 좀 복잡했다. 그래도 array를 사용해서 코드를 작성하는 방법을 알게 되어서 신기했다. 이번에 하면서 입력값과 결과값 그리고 wire의 변수들의 관계를 생각해보게 되었다. 결과값으로 확인할 필요가 없는 경우에 wire로 설정하면 된다는 것을 알았다. 이번에는 모듈을 여러개 사용하는 방법으로 코드를 구현하지 아니해서 그 방법은 잘 모르지만 c언어처럼 미리 만들어 놓은 다른 모듈을 갖고 와서 사용이 가능하다는 것이 신기했다.

5.

Verilog의 문법 : 베릴로그는 레지스터를 사용해서 데이터를 저장한다. 보통은 reg를 사용하고 reg이외에도 real, time, integer 등의 레지스터가 있다. 각각 실수 표현을 하거나 시간을 지정하거나 정수를 표현하는 레지스터이다. 벡터는 선언하는 경우에 그 순서가 중요하다. 먼저 데이터 타입을 써주고 제일 높은 비트 , 제일 낮은 비트를 적어주고 변수 명을 적어준다. wire[7:0] a로 쓰면 된다. 특이하게 2차원 이상의 배열은 만들 수 없다고 한다. 베릴로그의 파라미터는 상수를 정의하는 변수이다. 변수로는 사용이 불가능하다. 베릴로그에서 문자열도 사용이 가능한데 큰 따옴표를 넣어야 하고 한 라인에만 써야 한다. 문자열이 한 문자에 8비트를 차지하고 특수 문자는 역슬레시를 쓰면 출력 가능하다.