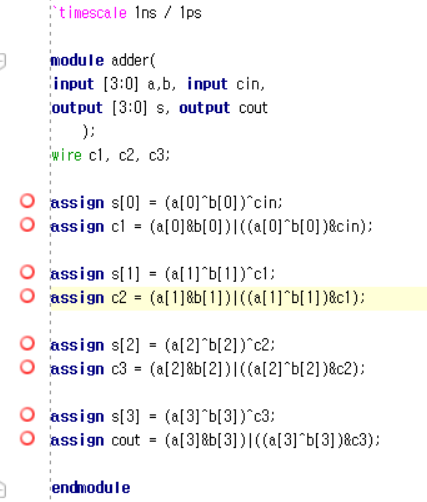
10주차 결과보고서

전공: 영미어문/컴퓨터공학과 학년: 3학년 학번: 20200185 이름: 박정주

**1.**

4bit binary parallel adder의 Design source를 다음과 같이 코딩하였다. 합할 4bit 수 2개를 의미하는 입력 a와 b를 배열로 선언하였다. a와 b를 합한 4bit 수를 의미하는 출력 s를 배열로 선언하였다. Carry in을 의미하는 입력 cin, carry out을 의미하는 출력 cout을 선언하고, 계산 과정에서 발생하지만 출력할 필요 없는 carry인 c1, c2, c3는 wire로 선언하였다.

덧셈을 수행할 두 수는 a[3]a[2]a[1]a[0]와 b[3]b[2]b[1]b[0]이다. 4개의 전가산기를 사용하는 방법으로 4bit adder를 구현하였다. s[0], c[1]은 a[0]과 b[0]을 입력으로 받는 첫번째 전가산기에서 출력하는 sum(s)과 carry(c)이다. s[3]과 cout은 마지막 전가산기, 즉 a[3]과 b[3]을 입력으로 받는 전가산기에서 발생한 sum(s)과 carry(c)이다. 따라서, 두 수의 뺄셈 덧셈 결과는 s[3]s[2]s[1]s[0]이다.



Simulation source를 다음과 같이 코딩하였다. Design source와 마찬가지로 입력 a, b와 출력 s는 배열로 선언하였다. a와 b는 0000으로 시작해 각각 10 시간단위, 20 시간단위가 지날 때마다 0001을 더하도록 하였다. cin은 0으로 시작하여 320 시간단위가 지나면 값을 부정하도록 하였다.

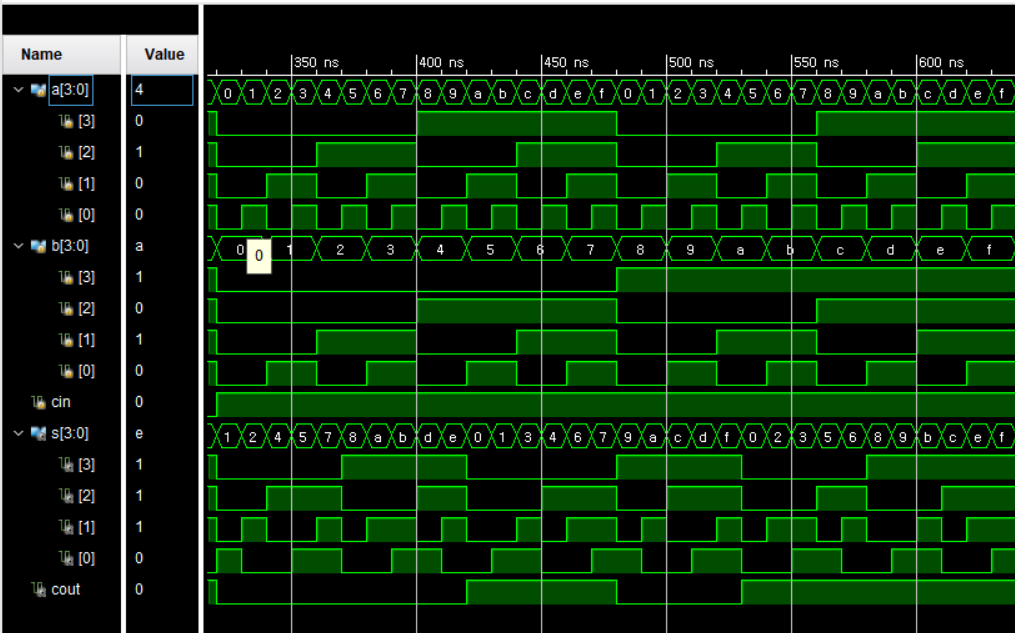
텍스트이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 다음과 같다. 예를 들어, a가 0100(4), b가 1010(10), cin이 0일 때, s는 1110(14)이고 cout은 0이다. a가 1001(9), b가 1100(12), cin이 0일 때, s는 0101(5), cout은 1이다. 즉 9+12=5+16=21이다.



다음은 cin이 1일 때의 시뮬레이션 결과이다. 예를 들어, a가 0000(0), b가 1000(8), cin이 1일 때, s는 1001(9), cout은 0이다.



**2.**

4bit binary parallel subtractor의 Design source를 다음과 같이 코딩하였다. 뺄셈을 수행할 4bit 수 2개를 의미하는 입력 a와 b를 배열로 선언하였다. a에서 b를 뺀 4bit 수를 의미하는 출력 d를 배열로 선언하였다. Borrow in을 의미하는 입력 bin, borrow out을 의미하는 출력 bout을 선언하고, 뺄셈 과정에서 발생하지만 출력할 필요 없는 borrow인 b1, b2, b3는 wire로 선언하였다.

뺄셈을 수행할 두 수는 a[3]a[2]a[1]a[0]와 b[3]b[2]b[1]b[0]이다. 4개의 전감산기를 사용하는 방법으로 4bit subtractor를 구현하였다. d[0], b[1]은 a[0]과 b[0]을 입력으로 받는 첫번째 전감산기에서 출력하는 difference(d)와 borrow(b)이다. d[3]과 bout은 마지막 전감산기, 즉 a[3]과 b[3]을 입력으로 받는 전감산기에서 발생한 difference(d)와 borrow(b)이다. 따라서, 두 수의 뺄셈 결과는 d[3]d[2]d[1]d[0]이다.

텍스트이(가) 표시된 사진

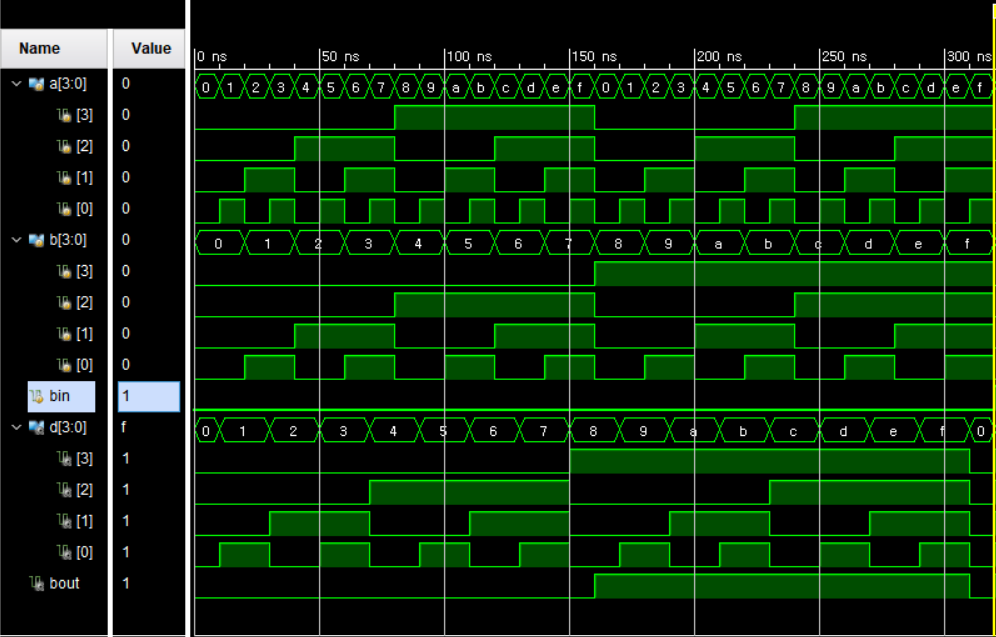
자동 생성된 설명

Simulation source를 다음과 같이 코딩하였다. Design source와 마찬가지로 입력 a, b와 출력 d는 배열로 선언하였다. a와 b는 0000으로 시작해 각각 10 시간단위, 20 시간단위가 지날 때마다 0001을 더하도록 하였다. bin은 0으로 시작하여 320 시간단위가 지나면 값을 부정하도록 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 다음과 같다. 예를 들어, a가 0010(2), b가 0001(1), bin이 0일 때, d는 0001(1)이고 bout은 0이다. A가 1000(8), b가 1100(12), bin이 0일 때, d는 1100, bout은 1이다. Bout이 1임은 값이 음수임을 나타내므로, d에 보수를 취하고(0011), 1을 더해주면(0100) 뺄셈 결과값(-4)을 구할 수 있다.



bin이 1일 때 a가 0000(0), b가 0000(0)이면 d는 1111이고 bout은 1이다. d에 보수를 취하고(0000) 1을 더한 뒤(0001) 마이너스 부호를 붙이면 십진수로 -1을 나타내는 것을 확인할 수 있다.



**3.**

BCD adder의 Design source를 다음과 같이 코딩하였다. 덧셈을 수행할 BCD 2개를 의미하는 입력 a와 b를 배열로 선언하였다. A와 b를 합한 BCD를 의미하는 출력 s를 배열로 선언하였다. Carry in을 의미하는 입력 bin, carry out을 의미하는 출력 bout을 선언하였다. 계산 과정에 팔요하지만 출력할 필요 없는 s0, s1, s2, s2와 k, c1, c2, c3, c4, c5, c6은 wire로 선언하였다.

덧셈을 수행할 두 BCD 수는 a[3]a[2]a[1]a[0]와 b[3]b[2]b[1]b[0]이다. 2개의 4bit 가산기를 사용하여 BCD adder를 구현하였다. 먼저, 첫 번째 4bit adder에서 s3, s2, s1, s0을 구하고, 이 때 발생한 cout을 k에 할당한다.

assign cout = (s2&s3) | (s1&s3) | k;

위 연산을 통해 최종 carry인 cout을 구한다.

다음으로, 0 cout cout 0과 s3 s2 s1 s0을 4bit adder에 넣는다. sum[0]은 0과 s0을 더한결과이고, c4는 둘을 더할 때 발생한 carry이다. 다음 자리에서는 c4를 carry in으로 받아 연산한다. sum[3]은 c6을 carry in으로 받아 0과 s3을 합한 결과이다. 이와 같은 과정을 통해, 두 BCD의 덧셈 결과인 sum[3]sum[2]sum[1]sum[0]을 얻을 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Simulation source를 다음과 같이 코딩하였다. Design source와 마찬가지로 입력 a, b와 출력 sum은 배열로 선언하였다. a와 b는 0000으로 시작해 각각 10 시간단위, 20 시간단위가 지날 때마다 0001을 더하도록 하였다. cin은 0으로 시작하여 320 시간단위가 지나면 값을 부정하도록 하였다.

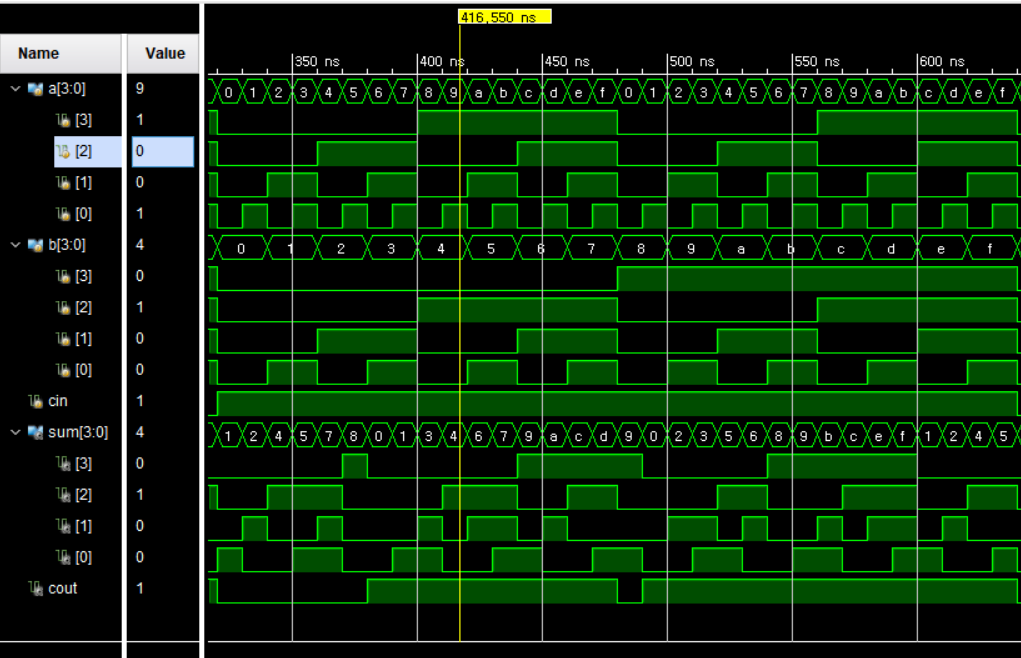
텍스트이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 다음과 같다. cin이 0, a가 1000(8), b가 0100(4)이면, sum은 0010(2), cout은 1이 출력된다. BCD adder에서 cout은 10을 의미하므로, 8+4 = 2+10 = 12로 덧셈이 정상적으로 수행된 것을 확인할 수 있다.



cin이 1, a가 1001(9), b가 0100(4)이면, sum은 0100(4), cout은 1이 출력되어, 합 14를 정상적으로 구하고 있다.



**4.**

4bit binary parallel adder와 subtracter는 각각 4bit 수의 덧셈과 뺄셈을 정상적으로 수행하였다. adder에서 덧셈 결과가 16을 넘으면 cout=1이 발생하였다. subtractor에서 뺄셈 결과가 음수가 되면 bout=1이 발생하였다. BCD adder에서는 덧셈 결과가 9를 넘으면 cout=1이 발생하였다. 세 회로가 모두 정상적으로 작동하는 것을 확인할 수 있었다.

**5.**