디지털논리회로 Verilog 과제

2018202046 이준휘

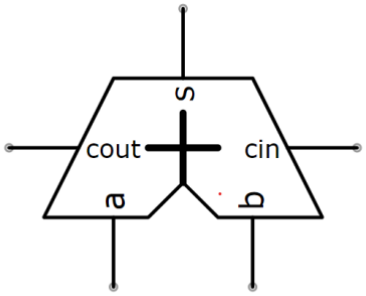
1. 문제 설명

해당 문제는 32bit의 Ripple Carry Adder(RCA), 즉 32비트끼리의 덧셈 가산기를 Verilog 상에서 구현하는 것을 목표로 한다. 모듈명(RCA32)과 모듈에 들어가는 기본적인 인자는 제시되고있으며, 필자는 해당 변수를 가지고 코드를 작성하여 RCA32를 완성하고, testbench를 통해 이를 시연해본다.

2. 해결과정

- module adder1bit(1-bit full adder)

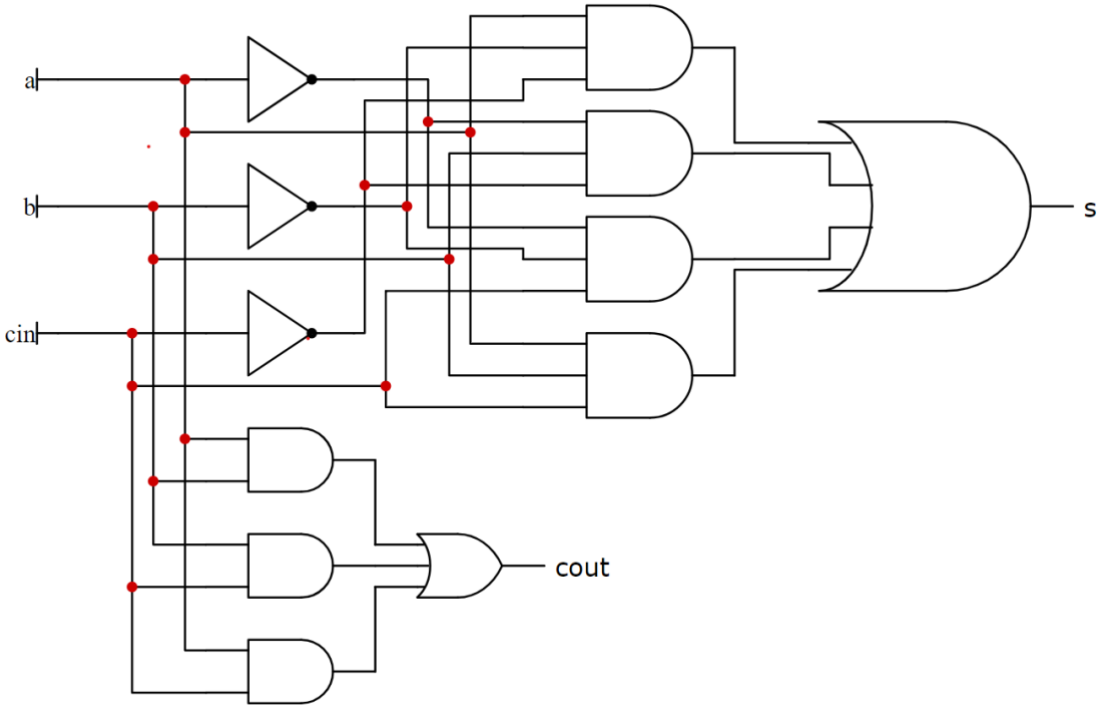
해당 모델의 경우 1-bit full adder의 역할을 수행한다. input a, b는 해당 자리에서 덧셈을 하는 인자다. 그리고 input cin은 이전 자리에서 올림으로 오는 인자다. output s는 해당 자리의 덧셈의 결과이고, output cout은 해당 자리에서 나오는 올림수다.



->full adder 기호

s의 값은 a, b, cin의 값이 홀수개만 true여야 true로 나와야 한다. 해당 부분의 논리식은 s = a ⊕ b ⊕ cin로 표현할 수 있다. 하지만 베릴로그에는 XOR gate 논리곱이 존재하지 않음으로 이를 s = a b′ cin′ + a′ b cin′ + a′ b′ cin + a b cin으로 풀어서 표현한다.

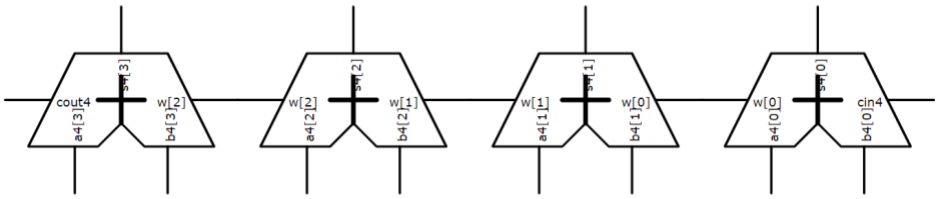
cout의 값은 a, b, cin의 값 중 2개 이상이 true일 경우 true여야 한다. 해당 부분의 논-리식은 cout = a b + b cin + cin a로 표현할 수 있다.



->1-bit full adder 회로도

- module adder4bit(4-bit RCA)

  해당 모델은 1bit의 full adder 4개를 병렬로 배치한 회로다. 해당 회로를 통해 4bit끼리의 덧셈 연산을 수행할 수 있다. 해당 회로는 input [3:0]으로 a4와, b4를 받는다. 그리고 이전 자리에서 올라오는 자리수인 output cin4를 받는다. output으로는 각 자리의 출력인 output [3:0] s4와 MSB자리에서 나오는 올림수인 output cout4가 있다. 마지막으로 MSB를 각각의 full adder를 연결할 wire [2:0] w가 있다.



->4-bit RCA 형태

우선 adder1bit인 first를 선언한다. 해당 모듈에서는 a4[0], b4[0], cin4를 인자로 받아 덧셈을 진행하여 s4[0]를 출력하고 올림수를 w[0]에 전달한다.

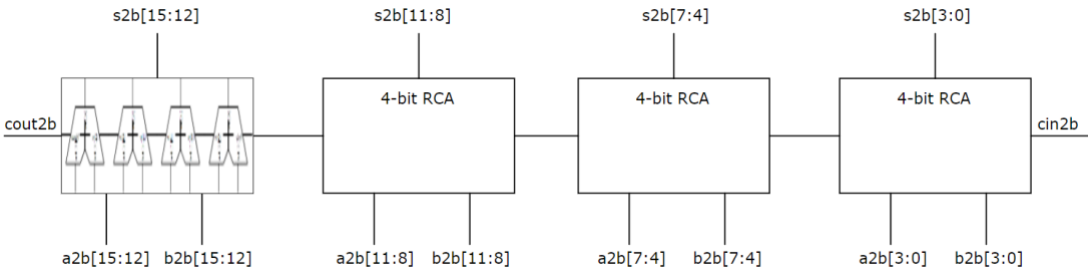
second는 해당 a4[1], b4[1], 그리고 올림수인 w[0]를 인자로 받아 덧셈을 진행하여 s4[1]을 출력하고 올림수를 w[1]에 전달한다.

third는 해당 a4[2], b4[2], 그리고 올림수인 w[1]를 인자로 받아 덧셈을 진행하여 s4[2]을 출력하고 올림수를 w[2]에 전달한다.

마지막으로 fourth에서는 해당 a4[3], b4[3], 그리고 올림수인 w[2]를 인자로 받아 덧셈을 진행하여 s4[3]을 출력하고 올림수 cout4를 출력한다.

- adder2byte(16-bit RCA)

해당 모델은 4-bit RCA 4개를 이용하여 16-bit RCA를 구현한다. 해당 회로를 통해 16비트끼리의 덧셈 연산을 수행할 수 있다. input [15:0] a2b, b2b를 받고 LSB에 더하는 인자인 input cin2b를 받는다. 이를 output [15:0] s2b과 다음 회로의 LSB에 넘길 output cout2b 값으로 반환한다. wire [2:0]를 통해 각 자리마다 넘어가는 올림수를 전달한다.



->16-bit RCA 형태

우선 adder4bit인 byte1를 선언한다. 해당 모듈에서는 a2b[3:0], b2b[3:0], cin2b를 인자로 받아 덧셈을 진행하여 s2b[3:0]를 출력하고 올림수를 wb[0]에 전달한다.

byte2는 해당 a2b[7:4], b2b[7:4], 그리고 올림수인 wb[0]를 인자로 받아 덧셈을 진행하여 s2b[7:4]를 출력하고 올림수를 wb[1]에 전달한다.

byte3는 해당 a2b[11:8], b2b[11:8], 그리고 올림수인 wb[1]를 인자로 받아 덧셈을 진행하여 s2b[11:8]를 출력하고 올림수를 wb[2]에 전달한다.

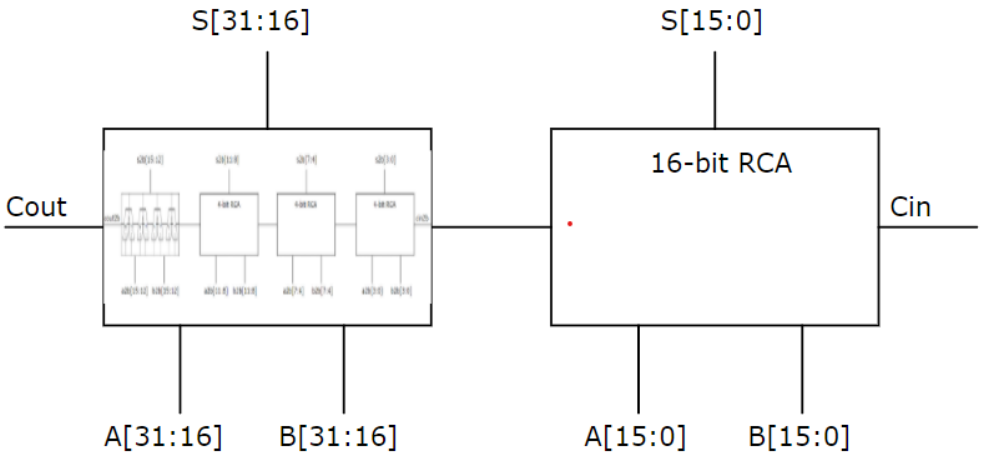
마지막으로 byte4에서는 해당 a2b[15:12], b2b[15:12], 그리고 올림수인 wb[2]를 인자로 받아 덧셈을 진행하여 s2b[15:12]를 출력하고 올림수를 wb[0]에 전달한다.

- model RCA32

해당 모델은 adder2byte 모델 2개를 이용하여 구현된다.

input [31:0] A와 B, 그리고 LSB에 더해지는 올림수인 input Cin이 있다. Cin이 존재하는 이유는 만약 Cin이 없도록 회로를 짤 경우 LSB를 더하는 과정은 half adder를 이용해야만 하기 때문에 회로도가 통일이 되지 않아 복잡해질 수 있다. 이에 full adder를 사용하는 대신 Cin에 0을 넣음으로써 half adder의 역할을 수행할 수 있다. output [31:0] S는 A, B의 합을 출력되는 곳이다. output Cout은 MSB의 연산 결과로 생기는 올림수를 나타내는 출력이다.

Cin과 Cout이 있음으로 인해 회로는 확장성에서 용이해진다. 만약 64비트의 RCA를 구현하고 싶은 경우 RCA32를 병렬로 연결함으로써 해당 회로를 구현할 수 있다.



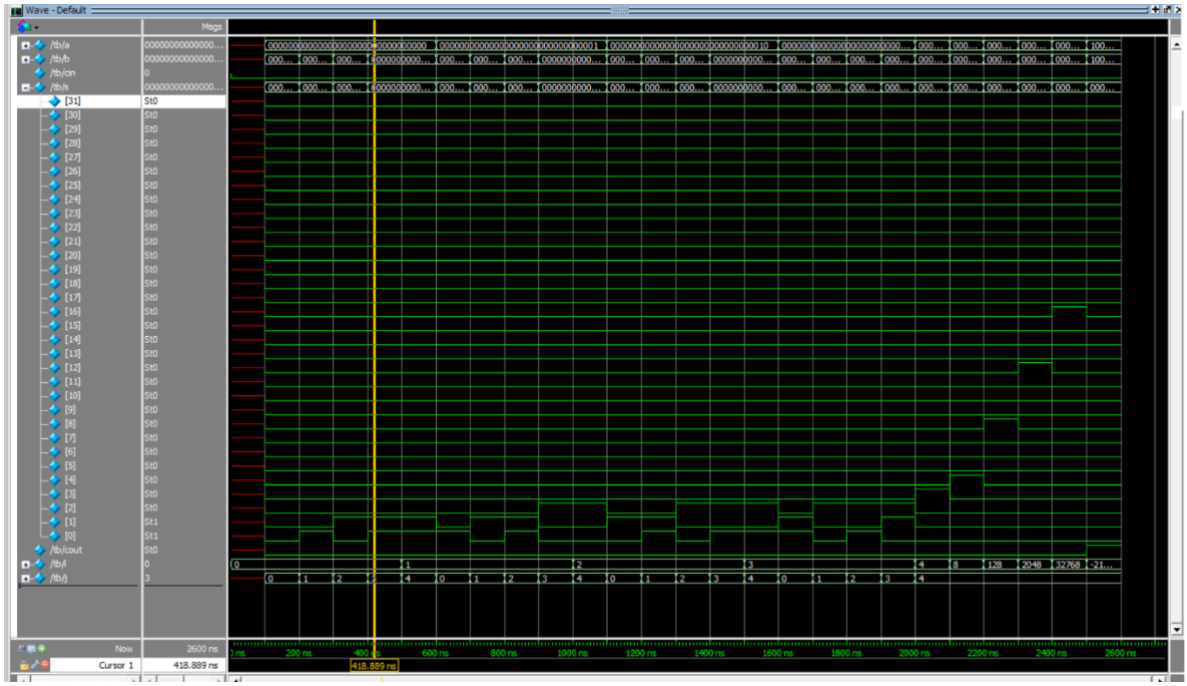
->RCA32의 블록 다이어그램

우선 adder2byte인 add1를 선언한다. 해당 모듈에서는 A[15:0], B[15:0], Cin를 인자로 받아 덧셈을 진행하여 S[15:0]를 출력하고 올림수를 w1(wire)에 전달한다.

byte2는 해당 A[31:16], B[31:16], 그리고 올림수인 w1를 인자로 받아 덧셈을 진행하여 S[31:16]를 출력하고 올림수를 Cout에 전달한다.

3. 결과 검증

test bench에서는 integer i와 j를 활용하여 검증을 진행한다.



->testbench 결과

우선 for문 2개와 i, j를 활용하여 0~4와 0~4와의 덧셈 16가지 경우가 잘 적용되었는지 확인하였다. 위의 사진을 보면 s의 값이 해당 값에 따라 적절하게 바뀌는 것을 볼 수 있다. 이는 가장 작은 모듈 full adder가 정상적으로 구동된다는 것을 의미한다.

그 후에 확인한 것은 RCA4bit module 안의 full adder 간의 자리수가 잘 전달되는지 확인하는 것이다. a, b가 4일 때와 8일 때 모두 잘 전달되어 각각 8, 16을 잘 표현하는 것을 볼 수 있다.

다음으로 확인한 것은 RCA16bit module 안의 RCA4bit끼리 값이 잘 전달되는지를 확인하는 것이다. a, b가 각각 2^7, 2^11, 2^15일 때 값이 2^8, 2^12, 2^16으로 모듈 간 자리수 전달이 제대로 이루어지는 것을 확인할 수 있다.

마지막으로 확인할 것은 만약 2^31 + 2^31를 진행할 경우 Cout에 1이 전달되는지 확인하는 것이다. 이에 위의 사진에서는 Cout 값이 1로 올라가 있는 것을 보아 정상임을 알 수 있다.