REPORT

[디지털 논리회로 – 32-Bit Ripple Carry Adder in Verilog]

디지털 논리회로 1

이준환 교수님

2020202077 신성민

2020 / 05 / 31 (월) 마감

INDEX

1. Introduce this problem

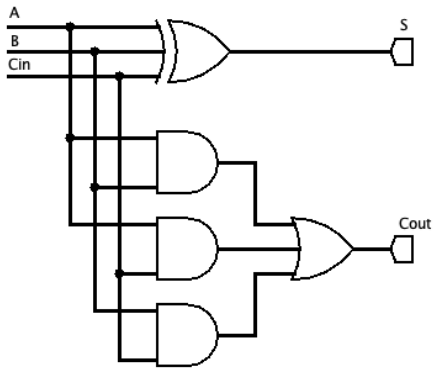
Ripple Carry Adder를 Verilog로 구현하는 문제로 32비트 RCA를 Verilog로 작성하고, 테스트 벤치를 작성하여 본인이 작성한 32-bit Ripple Carry Adder가 정상작동 하는지 확인해보는 과제입니다.

저의 경우에는 32-bit Ripple Carry Adder을 구현하기 위해 1-Bit Ripple Carry 연산을 하는 Adder를 모듈 RCA1으로, RCA1을 8개 묶어 8-Bit Ripple Carry Adder를 모듈 RCA8, RCA8를 4개를 묶에 32-bit Ripple Carry Adder을 시행하는 RCA32 모듈로 총 3개의 모듈로 나누어 구성하였습니다.

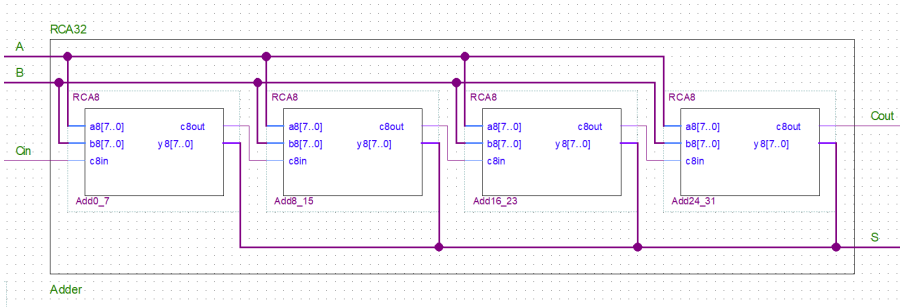
제가 구현한 Ripple Carry Adder에서는 input으로 A, B, Cin이 사용되고 Output으로 S, Cout이 사용되고 있습니다. Input에 사용되는 A와 B는 가산되는 값이며 32비트 버스로 구성되어 있고, Cin은 Carry 올림 비트입니다. Output에서 S의 경우에는 A와 B가 더해진 값을 반환하는 32비트 버스이며, Cout은 최대 비트에서 (즉, 32번째 RCA1에서) 자리 올림이 발생한 경우 1을 출력하는 비트입니다.

2. Gate-level design, Block diagram

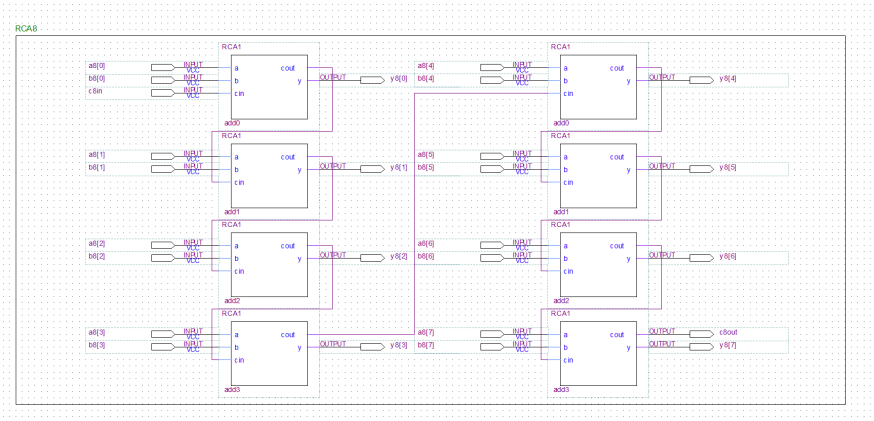
A. Gate-level design of a full adder



B. Block diagram of your RCA



<RCA32 Block diagram>



<RCA8 Block diagram>

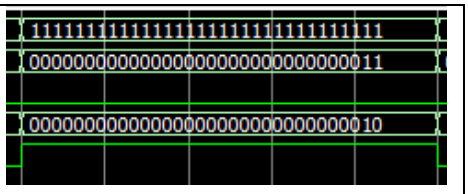
3. RCA Description with Examples

제가 작성한 RCA의 경우 1-Bit의 Ripple Carry Adder인 RCA1모듈과 이 RCA1 8개를 연결하여 8-Bit Ripple Carry Add 연산을 하는 RCA8 모듈, 그리고 RCA8을 4개 연결하여 32-Bit Ripple Carry Adder을 구성하였습니다. 작동 로직을 보면, 사실상 1번째 RCA1모듈에서 32번째 RCA1 모듈이 직렬로 Cin이 연결된 구성입니다.

A. About the Cin

- Why do you need Cin?

Cin이 필요한 이유는 아래 비트에서 자리 올림이 발생한 경우 이를 전달 받을 부분으로 사용하기 위해 존재합니다. 이진수에서는 산술 연산 1+1의 결과는 10입니다. 이를 이진수로 표현하기 위해 1 (XOR) 1을 진행하고, 자릿수 한자리가 올린 부분을 다음비트에서 Cin이 받는 것입니다.

A (0xFFFFFFFF)	
B (0x00000003)	
Cin (0)	
S (0x00000002)	
Cout (1)	

32비트의 최대값인 FFFFFFFF(16진수)에 3을 더한 상황에서는 첫번째 RCA1부터 자리 올림이 되어 다음 RCA1 모듈부터 cin이 1이 된다. 최종적으로 RCA32의 Cout이 1이 된 상황

- What is the value of Cin?

첫번째에 입력되는 RCA1 모듈에서는 0이 입력되게 하였습니다. (추가로, 감산 기능을 추가한다면 2의 보수를 구하기 위해 Cin에 감산 비트가 연결될 것 같다는 예상을 해 봅니다.) 이전에 연결되는 RCA가 없기 때문입니다. 이후 연결되는 RCA1의 모듈들은 이전 RCA1의 cout에 cin이 연결되도록 작성하였습니다.

4. Code Analysis

Code	Description
	1-Bit Ripple Carry Adder을 RCA1 모듈로, 8-Bit Ripple Carry Adder을 RCA8 모듈로, 32-Bit Ripple Carry Adder을 RCA32 모듈로 분리하여 구현

```
module RCA1(a, b, cin, cout, y);
    input a, b, cin;
    output cout, y;

    assign y = a ^ b ^ cin;
    assign cout = (a & b) | (a & cin) | (b & cin);
endmodule
```

RCA1 모듈은 1-bit Ripple Carry Adder

Input으로 Bit wire인 a, b, cin을 입력
Output으로 Bit wire인 cout, y를 출력
y는 a와 b와 cin의 XOR 연산
cout는 a와 b의 AND, a와 cin의 AND, b와 cin의 AND, 셋의 OR연산

```
module RCA8(a8, b8, c8in, c8out, y8); // 8bit로 분할한 RCA
    input [7:0] a8, b8;
    input c8in;
    output [7:0] y8;
    output c8out;

    RCA1 add0(a8[0], b8[0], c8in, i0, y8[0]);
    RCA1 add1(a8[1], b8[1], i0, i1, y8[1]);
    RCA1 add2(a8[2], b8[2], i1, i2, y8[2]);
    RCA1 add3(a8[3], b8[3], i2, i3, y8[3]);
    RCA1 add4(a8[4], b8[4], i3, i4, y8[4]);
    RCA1 add5(a8[5], b8[5], i4, i5, y8[5]);
    RCA1 add6(a8[6], b8[6], i5, i6, y8[6]);
    RCA1 add7(a8[7], b8[7], i6, c8out, y8[7]);
endmodule
```

RCA8 모듈은 RCA1 모듈 8개를 연결한 모듈

Input으로 8-Bit bus인 a8, b8과 Bit wire인 c8in을 입력
Output으로 8-Bit bus인 y8과 Bit wire인 c8out을 출력

add0~add7의 RCA1모듈을 선언 및 add0의 cin은 c8in에 연결, add7의 cout은 c8out에 연결, 나머지의 경우에는 현재 cout과 다음 cin을 연결함. 각 RCA1의 n번째 모듈은 a8, b8 bus의 n번째 비트를 받아와 각각의 a, b에 연결하고, 출력인 y의 경우 y8의 n번째 비트에 연결.

```
module RCA32 (S, Cout, A, B, Cin);
    input [31:0] A, B;
    input Cin;
    output [31:0] S;
    output Cout;
    // 8bit로 분할한 RCA 4개를 연결하여 32bit RCA 구현
    RCA8 Add0_7(A[7:0], B[7:0], Cin, w0, S[7:0]);
    RCA8 Add8_15(A[15:8], B[15:8], w0, w1, S[15:8]);
    RCA8 Add16_23(A[23:16], B[23:16], w1, w2, S[23:16]);
    RCA8 Add24_31(A[31:24], B[31:24], w2, Cout, S[31:24]);
endmodule
```

RCA32 모듈은 RCA8 모듈 8개를 연결한 모듈

Input으로 32-Bit bus인 A, B와 Bit wire인 Cin을 입력
Output으로 32-Bit bus인 S와 Bit wire인 Cout을 출력

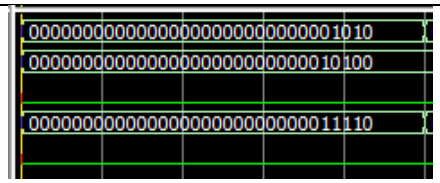
RCA8모듈 Add0_7, Add8_15, Add16_23, Add24_31 선언. 각각 A,B bus의 8비트씩 입력 받고, Add0_7의 경우 c8in으로 Cin을 입력, Add24_31의 경우 c8out을 Cout에 출력, 나머지의 RCA8 모듈의 경우 현재 c8out과 다음 c8in을 연결. 출력의 경우 Add0_7는 S[7:0], Add8_15는 S[15:8], Add16_23는 S[23:16], Add24_31는 S[31:24]에 출력하도록 연결.

5. Verification

1. A = (d)10, B = (d)20

Exp) S = 30, Cout = 0

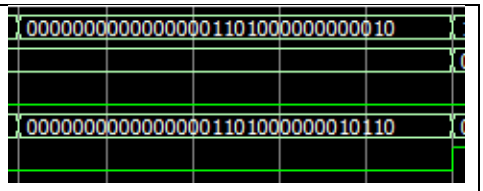
Real)

A (10)	
B (20)	
Cin (0)	
S (0b11110 = 30)	
Cout (0)	

2. A = (d)53250, B = (d)20

Exp) S = 53270, Cout = 0

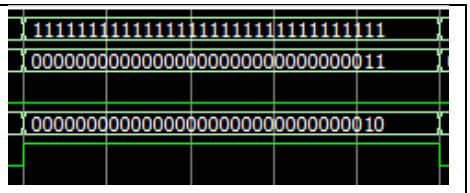
Real)

A (53250)	
B (20)	
Cin (0)	
S (53270)	
Cout (0)	

3. A = (h)FFFFFFFF, B = (h)00000003

Exp) S = 2, Cout = 1

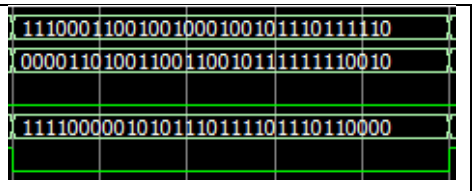
Real)

A (FFFFFFFF)	
B (00000003)	
Cin (0)	
S ((h)00000002)	
Cout (1)	

4. A = (h)E3244BBE, B = (h)0D332FF2

Exp) S = (h)F0577BB0, Cout = 0

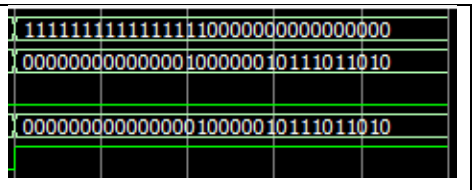
Real)

A (E3244BBE)	
B (0D332FF2)	
Cin (0)	
S ((h)F0577BB0)	
Cout (0)	

5. A = (h)FFFF0000, B = (h)000205DA

Exp) S = (h)000105DA, Cout = 1

Real)

A (FFFF0000)	
B (000205DA)	
Cin (0)	
S ((h)000105DA)	
Cout (1)	