4Bits-Adder 실험 결과 보고서

21700034 곽영혜

* 과제의 목적

Vivado 프로그램을 활용하여 verilog언어를 사용하는 방법을 익히기 위함과

강의 시간 때 배운 1bit-Adder를 활용하여 4bit 크기의 숫자 둘을 더할 수 있는

4bit-Adder를 만들어 FPGA에서 스위치와 LED를 통해 작동하고자 하였습니다.

* 1bit-Adder의 구조(adder.v)

module adder(A, B, C, Sum, Cout);

input A;

input B;

input C;

output Sum;

output Cout;

assign Sum=A^B^C;

assign Cout=(A&B)|(B&C)|(C&A);

endmodule

설명

: 1bit 숫자인 A와 B, C가 input 값으로 들어가 계산을 거쳐 Sum과 Cout이 output 값으로 도출됩니다.

\* 1bit-adder의 진리표

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | Cout | Sum |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Cout = A’BC + AB’C + ABC’

= AB + BC + CA

= (A&B)|(B&C)|(C&A)

Sum = A’B’C + A’BC’ + AB’C’ + ABC

= A ^ B ^ C

* 코드 설명

1. 소스코드(Add\_4.v)

module Add\_4(A, B, C, Sum, Cout); //4Bit-Adder의 모듈

input [3:0]A;

input [3:0]B;

//더하고자 하는 4bits의 두 숫자들(2진법)을 어레이에 저장하는 변수

input C;

//처음에는 0을 더하지만 1Bit-Adder의 출력결과인 Cout을

다음 Adder에 더할 수 있도록 마련한 Input 변수

wire C1, C2, C3;

//Adder에서 생성되는 Cout들을

다음 Adder로 연결하는 역할을 하는 와이어 변수

output [3:0]Sum;

//4Bits의 숫자들(2진법)을 합한 결과를 저장하는 어레이변수

output Cout;

//Adder에서 계산되어 나오는 출력값, 2진법 상으로는 1과 1이 더해져 자릿수가

한 칸 커졌을 때, 다음 칸의 2진법 수와 계산할 수 있도록 다음 Adder로 보내기

위해 존재함.

//assign Sum = A^B^C;

//assign Cout=(A&B)|(B&C)|(C&A);

//1Bit-Adder를 활용하기 때문에 Sum과 Cout에 대한 식이 필요 없음

adder u0 (A[0], B[0], 0, Sum[0], C1);

adder u1 (A[1], B[1], C1, Sum[1], C2);

adder u2 (A[2], B[2], C2, Sum[2], C3);

adder u3 (A[3], B[3], C3, Sum[3], Cout);

//1Bit-Adder를 이용하여 2진법의 동일한 자리수의 숫자끼리 더하여 합과 Cout을 구합니다.

특히, Cout을 나타내는 변수들은 wire로 다음 Adder의 input으로서 들어가 합과 Cout의 계산이 이어지도록 합니다.

endmodule

2. 테스트벤치 코드(testbench.v)

module testbench( ); //Adder가 제대로 계산되는지 확인하기위한 코드

reg [3:0]x;

reg [3:0]y;

//입력받는 숫자들을 논리식의 형태로 저장함.

wire [3:0]S;

//숫자들의 합을 와이어 어레이 변수에 저장하고

wire C1, C2, C3, C4;

//C1~4의 와이어 변수들을 통해 Cout의 이동경로를 만들어 줍니다.

adder u0 (x[0], y[0], 0, S[0], C1);

adder u1 (x[1], y[1], C1, S[1], C2);

adder u2 (x[2], y[2], C2, S[2], C3);

adder u3 (x[3], y[3], C3, S[3], C4);

//1Bit-Adder를 이용하여 2진법의 동일한 자리수의 숫자끼리 더하여 합(S[0~3])과 C4(Cout)를 구합니다.

initial begin

//테스트를 위해 변수 x, y값을 초기화 시킨 후,

임의의 숫자들을 넣음

x=0;

y=0;

#10 //10 ns(나노세컨드)가 지난 후 실행

x=1;

y=5;

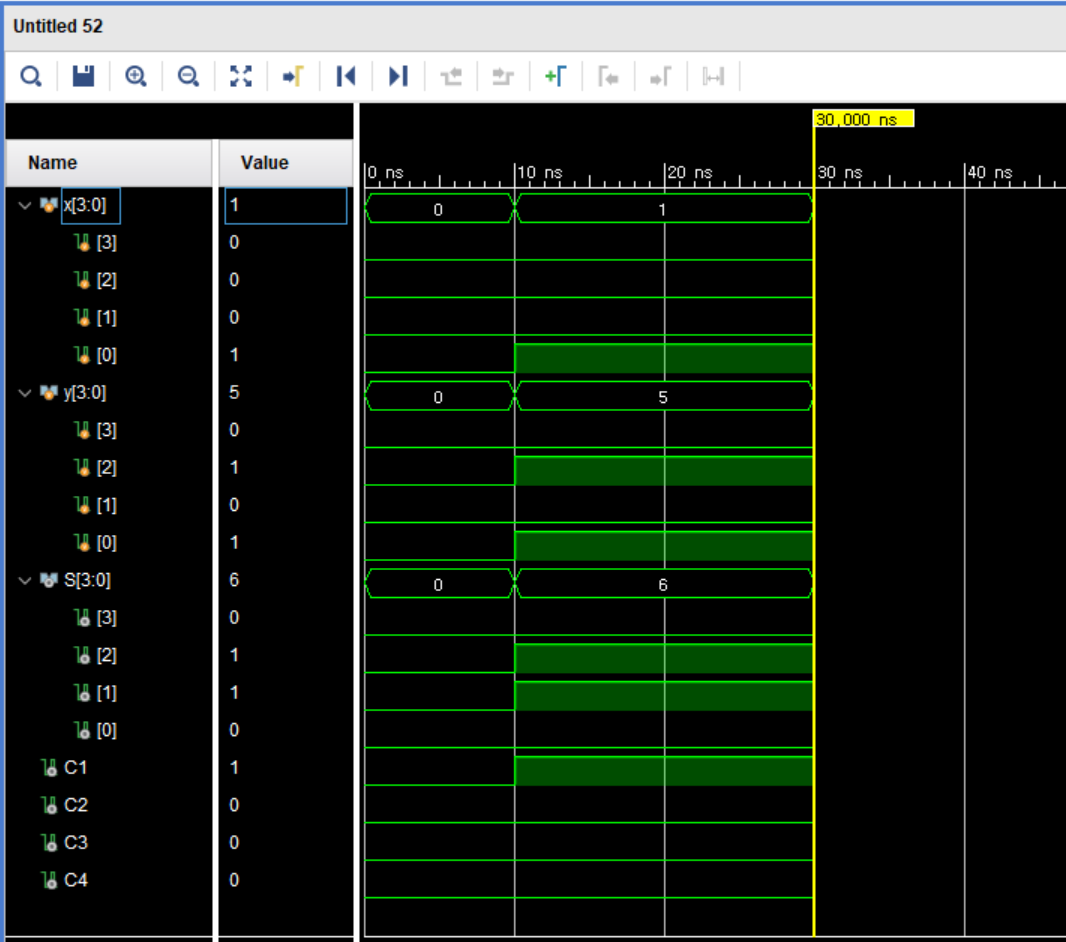
#20

$finish; //시간초 종료

End //테스트 종료

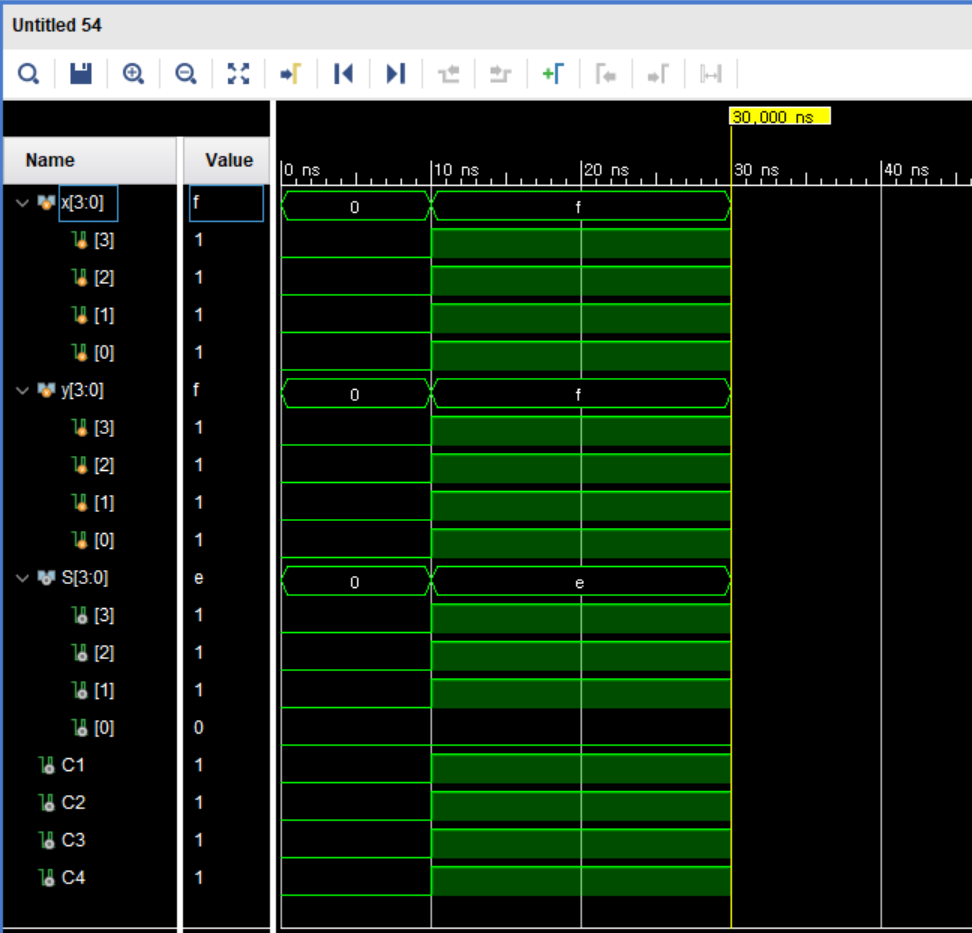
endmodule

1. 시물레이션 결과 waveform
   * X=4’b0001(1) / Y=4’b0101(5)의 simulation waveform



X[0]과 y[0], 0과의 Cout 계산에서 1이 나오면서 C1에 들어가고 C1은 x[1], y[1]과 함께 더해져서 Sum=4’b0110(6)의 결과가 나오는 것을 확인할 수 있습니다.

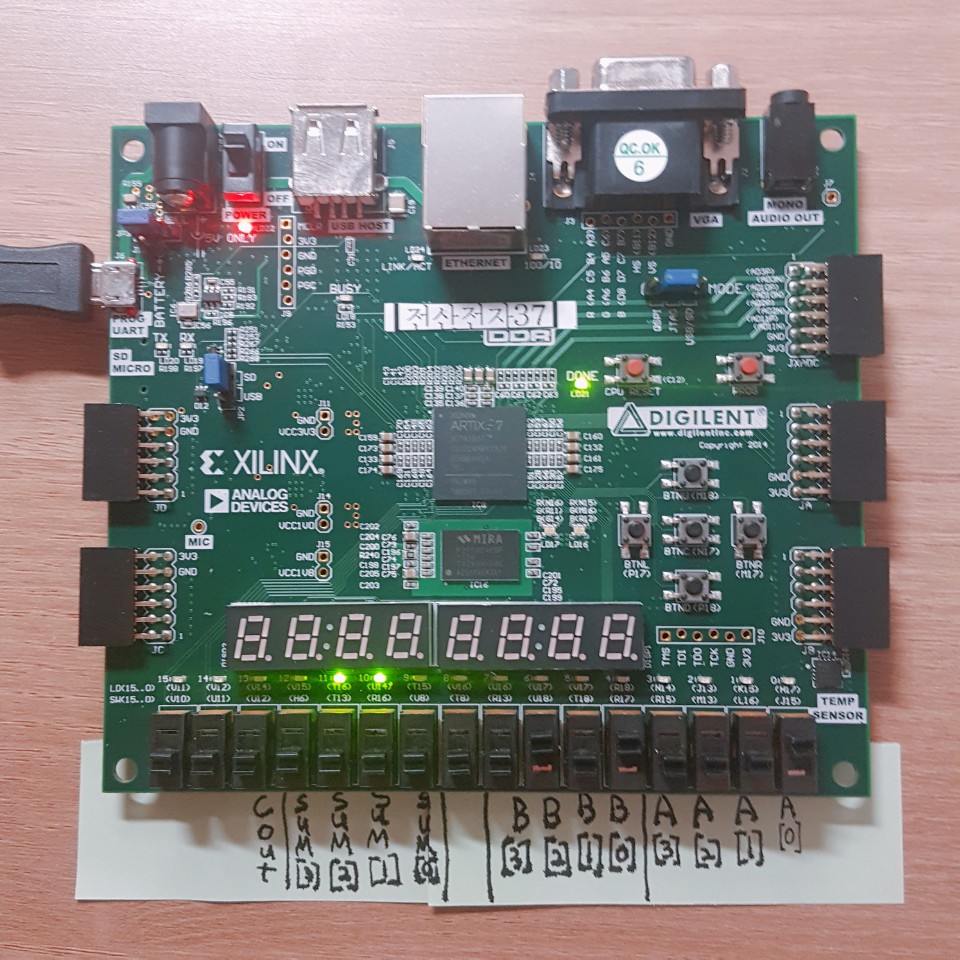
* + X=4’b1111(15) / Y=4’b1111(15)의 simulation waveform



모든 bit-adder에서 Cout의 값이 1로 생성되어 연결되며 최종 Cout이 1이며 Sum(30)이 4bit로 표현할 수 있는 값을 넘어간 것을 확인할 수 있습니다.

* Bitstream을 통한 FPGA에서의 실험 결과

1) X=4’b0001(1), Y=4’b0101(5)인 경우



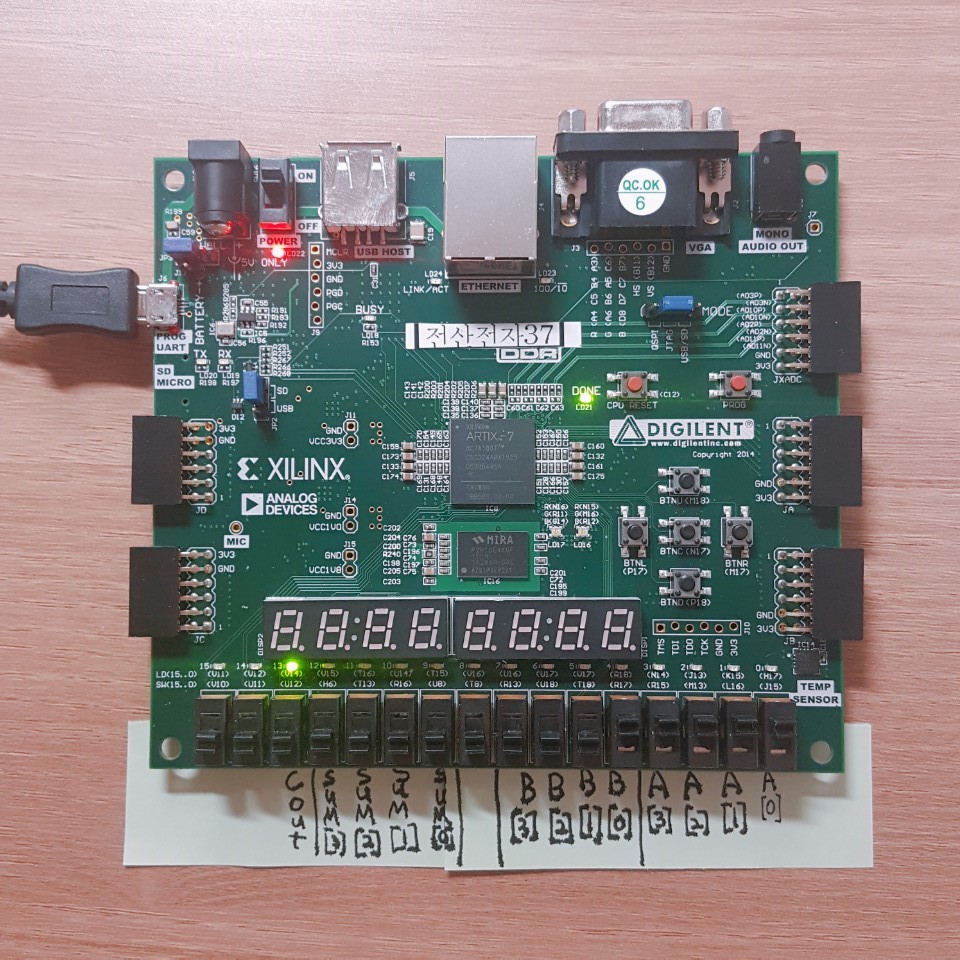
Sum=4’bit0110(6)의 결과가 위의 사진과 같이 Sum[2]와 Sum[1]의 LED에 불이 켜지는 것으로 확인할 수 있습니다.

2) X=4’b1111(15), Y=4b’1111(15)인 경우



Sum=11110(30)의 결과가 위의 사진과 같이 Cout과 Sum[0]을 제외한 모든 LED에 불이 켜지는 것으로 확인할 수 있습니다.

3) X=4’b0001(1), Y=4’b1111(15)인 경우



Sum=10000(16)의 결과가 위의 사진과 같이 Cout의 LED에 불이 켜지는 것으로 확인할 수 있습니다.

* 실험을 통해 느낀 점

저번 학기의 논리설계를 통해 이론적인 내용을 배웠다면 이 강의를 통해 실제 전자부품에 그 결과값을 나타나도록 설정하는 방법을 배울 수 있었습니다.

처음에는 vivado프로그램을 다루는 것이 어려웠는데 이번 과제를 통해 프로그램 Implementation과 Simulation을 실행하여 프로그램의 오류를 찾아내 바로잡는 과정을 반복하여 익힐 수 있었습니다.