10주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211533 이름: 남정연

**1.**

1. 4bit Binary Parallel Adder 의 결과 및 Simulation 과정에 대해서 설명하시오.

4 Bit binary Parallel Adder는 덧셈 연산을 수행하는 회로로, 각 단계에서 이전 단계의 Carry를 필요로 한다. 따라서 이전 단계의 Carry를 담을 wire 변수를 선언하여 아래와 같이 모듈을 구성하였다.

module inv(

input [3:0] A, [3:0] B,

input Cin,

output [3:0] S,

output Cout

);

wire [3:1] C;

assign S[0] = A[0]^B[0]^Cin;

assign C[1] = (A[0]^B[0])&Cin | A[0]&B[0];

assign S[1] = A[1]^B[1]^C[1];

assign C[2] = (A[1]^B[1])&C[1] | A[1]&B[1];

assign S[2] = A[2]^B[2]^C[2];

assign C[3] = (A[2]^B[2])&C[2] | A[2]&B[2];

assign S[3] = A[3]^B[3]^C[3];

assign Cout = (A[3]^B[3])&C[3] | A[3]&B[3];

endmodule

또한 테스트 벤치 코드의 경우 0000부터 시작하여 for 문을 이용하여 모든 경우를 고려할 수 있게 설계를 하였으며 이에 따라 아래와 같은 시뮬레이션 결과가 도출되었다.

Graphical user interface

Description automatically generated

**2.**

4 Bit binary Parallel Subtractor는 뺄셈 연산을 수행하는 회로로, 각 단계에서 이전 단계의 Borrow를 필요로 한다. 따라서 이전 단계의 Borrow를 담을 wire 변수를 선언하여 아래와 같이 모듈을 구성하였다.

module inv(

input [3:0] A, [3:0] B,

input Bin,

output [3:0] D,

output Bout

);

wire [3:1] wB;

assign D[0] = A[0]^B[0]^Bin;

assign wB[1] = (~A[0]&Bin) | (~A[0]&B[0]) | (B[0]&Bin);

assign D[1] = A[1]^B[1]^wB[1];

assign wB[2] = (~A[1]&wB[1]) | (~A[1]&B[1]) | (B[1]&wB[1]);

assign D[2] = A[2]^B[2]^wB[2];

assign wB[3] = (~A[2]&wB[2]) | (~A[2]&B[2]) | (B[2]&wB[2]);

assign D[3] = A[3]^B[3]^wB[3];

assign Bout = (~A[3]&wB[3]) | (~A[3]&B[3]) | (B[3]&wB[3]);

endmodule

또한 테스트 벤치 코드의 경우 0000부터 시작하여 for 문을 이용하여 모든 경우를 고려할 수 있게 설계를 하였으며 이에 따라 아래와 같은 시뮬레이션 결과가 도출되었다.

Graphical user interface

Description automatically generated

**3.**

BCD Adder는 Binary 값을 입력으로 받아 덧셈을 수행하여 BCD 코드로 이를 출력하는 회로로, 4 Bit Binary Adder에 보정값을 계산하는 방식으로 구성된다. 이에 따라 아래와 같이 모듈을 구성하였다.

module inv(

input [3:0] A, [3:0] B,

input Cin,

output [3:0] BcdS,

output OCout

);

wire [3:1] C;

wire [3:1] BcdC;

wire Cout;

wire [3:0] S;

assign S[0] = A[0]^B[0]^Cin;

assign C[1] = (A[0]^B[0])&Cin | A[0]&B[0];

assign S[1] = A[1]^B[1]^C[1];

assign C[2] = (A[1]^B[1])&C[1] | A[1]&B[1];

assign S[2] = A[2]^B[2]^C[2];

assign C[3] = (A[2]^B[2])&C[2] | A[2]&B[2];

assign S[3] = A[3]^B[3]^C[3];

assign Cout = (A[3]^B[3])&C[3] | A[3]&B[3];

assign OCout = Cout | (S[3]&S[2]) | (S[3]&S[1]);

wire [3:0] BcdA;

assign BcdA[0] = 0;

assign BcdA[1] = OCout;

assign BcdA[2] = OCout;

assign BcdA[3] = 0;

assign BcdS[0] = BcdA[0]^S[0];

assign BcdC[1] = BcdA[0]&S[0];

assign BcdS[1] = BcdA[1]^S[1]^BcdC[1];

assign BcdC[2] = (BcdA[1]^S[1])&BcdC[1] | BcdA[1]&S[1];

assign BcdS[2] = BcdA[2]^S[2]^BcdC[2];

assign BcdC[3] = (BcdA[2]^S[2])&BcdC[2] | BcdA[2]&S[2];

assign BcdS[3] = BcdA[3]^S[3]^BcdC[3];

endmodule

또한 테스트 벤치 코드의 경우 0000부터 시작하여 for 문을 이용하여 모든 경우를 고려할 수 있게 설계를 하였으며 이에 따라 아래와 같은 시뮬레이션 결과가 도출되었다.

Graphical user interface

Description automatically generated with medium confidence

**4.**

모든 값에 대하여 테스트 벤치 코드를 작성하여 시뮬레이션 결과를 알아보기에 어려움이 있었으나 Verilog으로 코딩한 것을 직접 FPGA에 실행시켜봄으로써 모든 실험에 대하여 정확하게 결과가 도출됨을 확인할 수 있었다. 이를 통해 사전에 설계한 논리회로와 Verilog를 이용하여 구성한 모듈이 같은 역할을 함을 확인할 수 있었다.

**5.**

논리회로는 크게 조합 논리회로와 순차 논리회로로 나눌 수 있다. 조합 논리 회로(Combinational Logic Circuit)은 입력 값들을 연산을 통해 출력을 하는 회로를 말하며 대수 연산이나 논리 연산을 하는 회로와 Mux, Demux, Encoder, Decoder와 같이 데이터 전송에 관련된 회로, BCD Converter나 7-Segment 출력과 같이 코드 변환을 하는 회로로 나눌 수 있다. 순차 논리 회로(Sequential Logic Circuit)은 출력이 이전 상태 값에 따라서 영향을 받는 회로로 기억 저장 장치가 여기에 포함된다. 다만 해당 출력이 외부 클럭에 의해서 트리거 되면 Synchornous, 그렇지 않으면 Asynchoronous한 회로로 구분할 수 있고 각각 플립플롭과 래치가 해당된다.