14주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231609 이름: 정희선

**1. Sequence Detector 1101 Moore machine**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input clk, rst, in,*  *output reg out*  *);*  *reg[3:0] seq; //최근 4개 입력 비트*  *reg[3:0] x = 4'b1101; //검출 시퀀스*  *initial out = 1'b0;*    *// 상승 엣지에서 동작*  *always @(posedge clk) begin*  *if (!rst) begin*  *// 리셋 시 초기화*  *seq[3] <= 1'b0;*  *seq[2] <= 1'b0;*  *seq[1] <= 1'b0;*  *seq[0] <= 1'b0;*  *out <= 1'b0;*  *end else begin*  *seq[3] <= seq[2];*  *seq[2] <= seq[1];*  *seq[1] <= seq[0];*  *seq[0] <= in;*  *end*  *if(seq == x) out <= 1'b1;*  *else out <= 1'b0;*  *end*  *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg clk, rst, in;*  *wire out;*  *boolean u\_boolean(*  *.clk(clk),*  *.rst(rst),*  *.in(in),*  *.out(out)*  *);*  *initial begin*  *clk = 1'b0;*  *forever #10 clk = ~clk; // 20ns 주기로 클럭 반전*  *end*  *initial begin*  *rst = 1'b0; // 초기에 리셋 활성화 (리셋 신호가 active high라고 가정)*  *in = 1'b0; // 초기 입력값은 0*  *#20 rst = 1'b1; // 20ns 후 리셋 비활성화*  *// 시퀀스 입력 시뮬레이션 시작*  *#30 in = 1'b1; #20; // 시간을 조금 더 주어서 시퀀스가 정확히 처리될 수 있도록 함*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20; // 첫 번째 '1101' 입력*  *#30 in = 1'b1; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20; // 두 번째 '1101' 입력*  *// 추가적으로 시퀀스를 더 입력하여 검출되는지 확인*  *#30 in = 1'b1; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20; // 세 번째 '1101' 입력*  *end*  *initial begin*  *#500*  *$finish;*  *end*  *endmodule* |

소프트웨어, 멀티미디어 소프트웨어, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

스케치, 그림, 화이트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

4비트 시퀀스 감지기를 구현한 것이다. 입력으로 클럭(clk), 리셋(rst), 그리고 입력 비트(in)을 받고 출력으로 out을 가진다. 레지스터 seq는 최근 4개의 입력 비트를 저장하며, x는 감지해야 할 시퀀스를 나타내는 4비트 상수로 4'b1101로 정의되어 있다. 초기 출력 out은 1'b0으로 설정되어 있으며 이는 회로가 처음 동작을 시작할 때 출력이 0으로 시작함을 의미한다.

클럭 신호의 상승 엣지에서 동작하며, 리셋 신호가 활성화되었을 때(seq 값은 0으로 초기화되고 out도 0으로 설정된다. 리셋이 비활성화되면 입력 비트 in이 클럭 상승 엣지마다 seq 레지스터에 시프트된다. seq[3]은 seq[2]의 값을 받고, seq[2]는 seq[1]의 값을, seq[1]은 seq[0]의 값을 받으며 seq[0]에는 새로 입력된 in 값이 저장된다. 이러한 방식은 입력 비트를 순차적으로 저장하는 시프트 레지스터의 동작과 동일하다.

seq의 상태가 감지하려는 시퀀스 x(1101)와 동일할 때 출력 out은 1로 설정된다. 그 외의 경우 out은 0으로 유지된다. 이 비교는 클럭 신호의 상승 엣지마다 이루어지며, seq가 새롭게 업데이트된 직후 상태를 기준으로 수행된다. seq의 업데이트와 출력 결정은 모두 non-blocking 할당을 사용하므로 동기적으로 동시에 업데이트된다. 따라서 레지스터 내 값이 안정적으로 유지되며 정확한 동작을 보장한다.

이 코드는 특정한 4비트 시퀀스를 감지하기 위한 간단한 회로로, 입력이 클럭에 맞춰 순차적으로 저장되며 지정된 시퀀스를 만족할 경우에만 출력이 1로 설정된다. 출력은 입력값과는 독립적으로 현재 상태(seq)에만 의존하여 결정되므로 Moore 머신의 특성을 가진다.

**2. Sequence Detector 10101**

**- mealy machine**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input clk, rst, in,*  *output reg out*  *);*  *reg[4:0] seq; //최근 4개 입력 비트*  *reg[4:0] x = 5'b10101; //검출 시퀀스*  *initial out = 1'b0;*    *// 상승 엣지에서 동작*  *always @(posedge clk) begin*  *if (!rst) begin*  *// 리셋 시 초기화*  *seq[4] <= 1'b0;*  *seq[3] <= 1'b0;*  *seq[2] <= 1'b0;*  *seq[1] <= 1'b0;*  *seq[0] <= 1'b0;*  *out <= 1'b0;*  *end else begin*  *seq[4] = seq[3];*  *seq[3] = seq[2];*  *seq[2] = seq[1];*  *seq[1] = seq[0];*  *seq[0] = in;*  *if(seq == x) out = 1'b1;*  *else out = 1'b0;*  *end*  *end*  *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg clk, rst, in;*  *wire out;*  *boolean u\_boolean(*  *.clk(clk),*  *.rst(rst),*  *.in(in),*  *.out(out)*  *);*  *initial begin*  *clk = 1'b0;*  *forever #10 clk = ~clk; // 20ns 주기로 클럭 반전*  *end*  *initial begin*  *rst = 1'b0; // 초기에 리셋 활성화 (리셋 신호가 active high라고 가정)*  *in = 1'b0; // 초기 입력값은 0*  *#20 rst = 1'b1; // 20ns 후 리셋 비활성화*  *// 시퀀스 입력 시뮬레이션 시작*  *#30 in = 1'b1; #20; // 시간을 조금 더 주어서 시퀀스가 정확히 처리될 수 있도록 함*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *#30 in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *end*  *initial begin*  *#500*  *$finish;*  *end*  *endmodule* |

스크린샷, 텍스트, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

그림, 스케치, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

5비트 시퀀스 감지기를 구현한 회로이다. 입력으로 클럭(clk), 리셋(rst), 그리고 입력 비트(in)을 받고 출력으로 out을 가진다. reg 타입의 레지스터 seq는 최근 5개의 입력 비트를 저장하기 위해 사용되며, reg 타입의 x는 감지할 시퀀스를 나타내는 5비트 상수로 5'b10101로 설정되어 있다. 초기 출력 out은 1'b0으로 설정되어 있으며, 이는 회로가 시작할 때 출력이 0임을 의미한다.

클럭 신호의 상승 엣지에서 동작하며, 리셋 신호(rst)가 비활성화되어 있을 때(seq와 out이 초기화된다. seq 레지스터의 모든 비트는 0으로 설정되며 out도 0으로 설정된다. 리셋이 비활성화되면 입력 비트 in이 클럭 신호 상승 엣지마다 seq 레지스터에 시프트된다. seq[4]는 seq[3]의 값을 받고, seq[3]은 seq[2]의 값을, seq[2]는 seq[1]의 값을, seq[1]은 seq[0]의 값을 받으며 seq[0]에는 새롭게 입력된 in 값이 저장된다. 이 동작은 시프트 레지스터와 동일한 동작을 수행하며 입력 신호를 순차적으로 저장한다.

seq 값이 감지하려는 시퀀스 x(10101)와 동일할 경우 출력 out은 1로 설정된다. 그렇지 않은 경우 출력은 0으로 설정된다. 이 비교는 클럭 신호 상승 엣지마다 이루어지며, seq 값이 업데이트된 이후의 상태를 기준으로 비교가 수행된다. seq의 업데이트는 non-blocking 할당을 사용하지 않고 blocking 할당을 사용하여 구현되었기 때문에 레지스터 값이 즉시 변경되며 이후의 출력 비교에 영향을 준다.

입력 신호가 주어질 때 seq 레지스터에 입력 값을 순차적으로 저장하며, 지정된 시퀀스를 만족할 경우에만 출력 out을 1로 설정한다. 출력 결정이 입력 신호 in과 seq 값의 조합에 따라 즉시 이루어지므로 이 회로는 Mealy 머신의 특성을 가진다.

**- moore machine**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input clk, rst, in,*  *output reg out*  *);*  *reg[4:0] seq; //최근 4개 입력 비트*  *reg[4:0] x = 5'b10101; //검출 시퀀스*  *initial out = 1'b0;*    *// 상승 엣지에서 동작*  *always @(posedge clk) begin*  *if (!rst) begin*  *// 리셋 시 초기화*  *seq[4] <= 1'b0;*  *seq[3] <= 1'b0;*  *seq[2] <= 1'b0;*  *seq[1] <= 1'b0;*  *seq[0] <= 1'b0;*  *out <= 1'b0;*  *end else begin*  *seq[4] <= seq[3];*  *seq[3] <= seq[2];*  *seq[2] <= seq[1];*  *seq[1] <= seq[0];*  *seq[0] <= in;*  *end*  *if(seq == x) out <= 1'b1;*  *else out <= 1'b0;*  *end*  *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg clk, rst, in;*  *wire out;*  *boolean u\_boolean(*  *.clk(clk),*  *.rst(rst),*  *.in(in),*  *.out(out)*  *);*  *initial begin*  *clk = 1'b0;*  *forever #10 clk = ~clk; // 20ns 주기로 클럭 반전*  *end*  *initial begin*  *rst = 1'b0; // 초기에 리셋 활성화 (리셋 신호가 active high라고 가정)*  *in = 1'b0; // 초기 입력값은 0*  *#20 rst = 1'b1; // 20ns 후 리셋 비활성화*  *// 시퀀스 입력 시뮬레이션 시작*  *#30 in = 1'b1; #20; // 시간을 조금 더 주어서 시퀀스가 정확히 처리될 수 있도록 함*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *#30 in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *in = 1'b0; #20;*  *in = 1'b1; #20;*  *end*  *initial begin*  *#500*  *$finish;*  *end*  *endmodule* |

**스크린샷, 소프트웨어, 텍스트, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

그림, 스케치, 라인 아트, 화이트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

5비트 시퀀스 감지기를 구현한 회로이다. 입력으로 클럭(clk), 리셋(rst), 그리고 단일 비트 입력(in)을 받고, 출력으로 out을 제공한다. 레지스터 seq는 최근 5개의 입력 비트를 저장하기 위해 사용되며, 레지스터 x는 감지해야 할 시퀀스를 나타내는 5비트 상수로 5'b10101로 설정되어 있다. 초기 출력 값 out은 1'b0으로 설정되어 있으며, 이는 회로가 처음 동작을 시작할 때 출력이 0으로 초기화됨을 의미한다.

클럭 신호의 상승 엣지에서 동작하며, 리셋 신호(rst)가 비활성화되었을 때(seq와 out이 초기화된다. seq의 모든 비트는 0으로 설정되며 출력 out도 0으로 설정된다. 리셋이 비활성화된 상태에서 클럭 신호가 상승하면, 입력 비트 in이 seq 레지스터에 시프트된다. seq[4]는 seq[3]의 값을 받고, seq[3]은 seq[2]의 값을, seq[2]는 seq[1]의 값을, seq[1]은 seq[0]의 값을 받으며 seq[0]에는 입력된 in 값이 저장된다. 이러한 방식은 시프트 레지스터의 동작과 유사하게 입력 신호를 순차적으로 저장하는 동작을 수행한다.

seq 값이 미리 정의된 감지 시퀀스 x(10101)와 동일하면 출력 out은 1로 설정된다. 그렇지 않은 경우 출력 out은 0으로 설정된다. 출력 결정은 seq의 현재 상태를 기준으로 이루어진다. seq 값이 매 클럭 상승 엣지마다 업데이트되기 때문에 출력도 동기적으로 결정된다. seq의 업데이트와 출력 결정은 모두 non-blocking 할당(<=)을 사용하여 구현되었기 때문에 레지스터와 출력이 안정적으로 업데이트된다. 입력 신호가 클럭에 따라 순차적으로 seq 레지스터에 저장되며, 지정된 시퀀스가 감지될 경우 출력 out이 1로 설정된다. 출력은 입력 신호 in과 현재 상태 seq의 조합에 따라 결정되지 않고, seq 값만을 기준으로 이루어진다. 따라서 이 회로는 Moore 머신의 특성을 가진다.