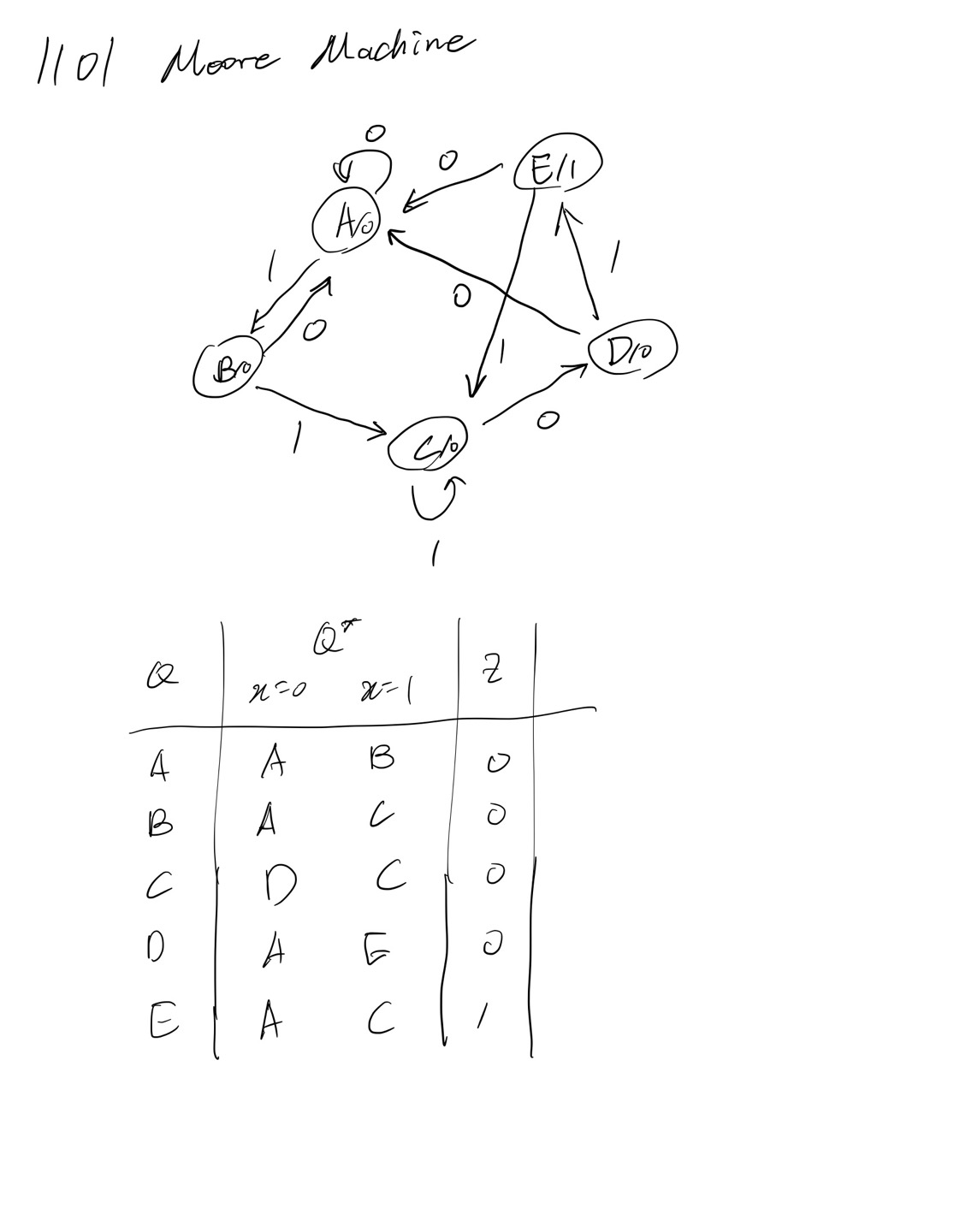
14주차 결과보고서

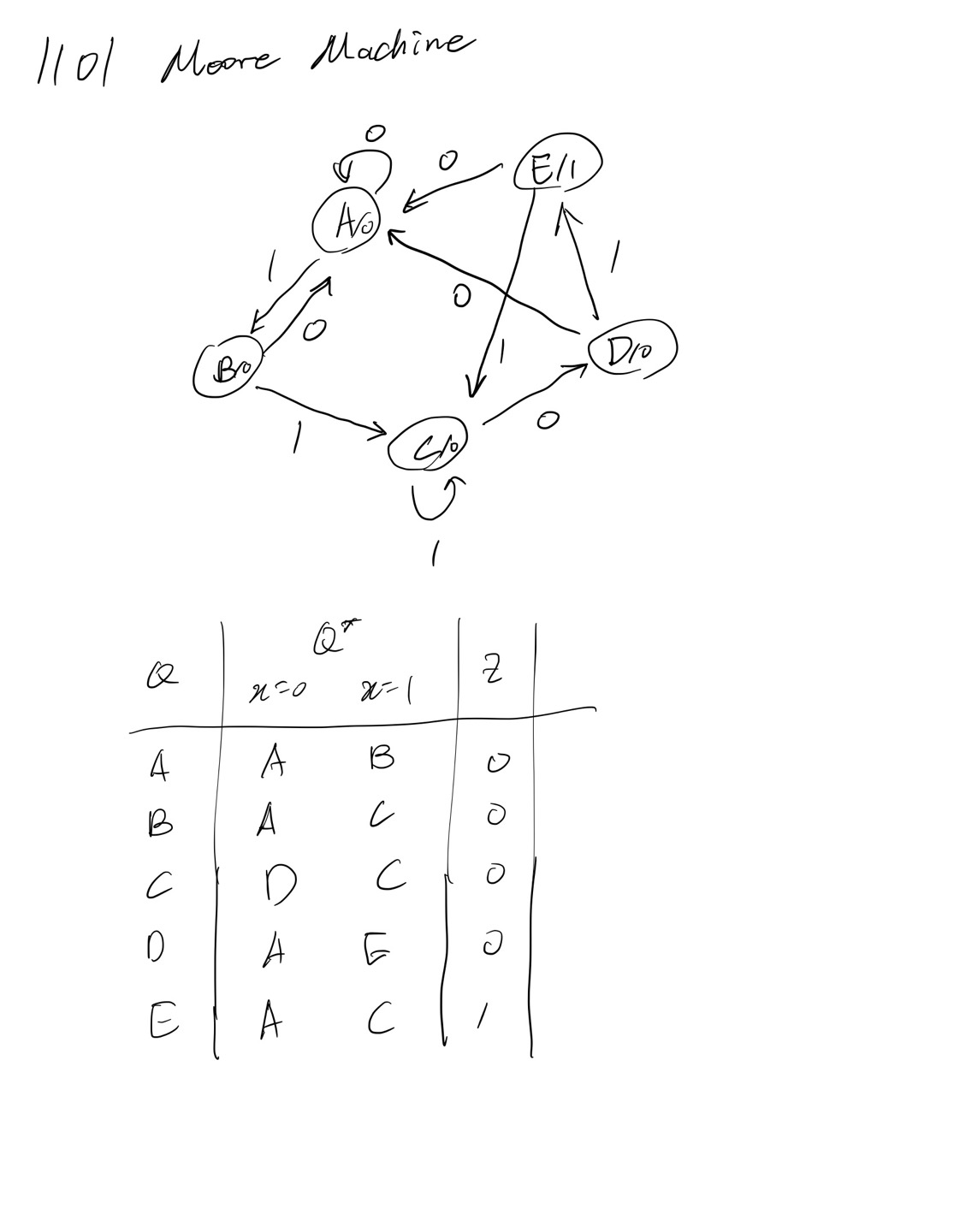
전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

Moore machine은 현재 상태로 출력 out을 결정한다. 1101을 감지하기 위한 moore machine의 상태도는 다음과 같다.



1101이 입력되는 것을 확인하기 위해서는 총 5개의 상태가 필요하다. 처음 상태를 A라고 하면, 처음 상태에서 0이 입력되면 다음 상태도 그대로 0이된다. A에서 1이 입력되면 다음 상태는 B가 되고, 이는 1101 중 처음 1이 입력되었다는 것을 의미한다. B에서 0이 입력되면 다시 처음 상태인 A로 돌아가 처음부터 다시 입력 받아야 하고, B에서 1이 입력되면 다음 상태는 C가 되는데, 이는 1101 중 11이 입력되었다는 것을 의미한다. C에서 1이 입력되면 다음 상태는 그대로 C가 되는데, 이는 지금까지 111이 입력되었기 때문에 다음 입력이 0이 되면 110이 입력되는 것이므로 그대로 C라는 상태를 유지하는 것이다. C에서 0이 입력되면 다음 상태는 D가 되는데, 이는 1101 중 110이 입력되었다는 것을 의미한다. D에서 다시 0이 입력되면 다음 상태는 A가 되는데, 1101이 아닌 1100이 입력되었으므로 다시 처음부터 입력 받아야 하기 때문이다. D에서 1이 입력되면 다음 상태는 E가 되고, 이때 1101이 입력되었다는 것을 의미하기 때문에 아웃풋이 1이 된다. E에서 0이 입력되면 처음 상태인 A로 돌아가 처음부터 입력받고, E에서 1이 입력되면 11011이 되기 때문에 11을 입력받은 상태와 동일하기 때문에 다음 상태는 C가 된다. 이를 상태표로 나타내면 다음과 같다.



이제, 이 상태표를 바탕으로 Verilog 코드를 작성한다. 먼저 디자인 코드는 다음과 같다.

module detector(

input Clk, rst, inx,

output out, state

);

reg[3:0] state = 4'b0000;

reg out = 1'b0;

always @(posedge Clk) begin

if (rst == 1)

begin

state = 4'b0000;

out = 0;

end

else

begin

if(state == 4'b1101)

out = 1;

else

out = 0;

state = state << 1;

state[0] = inx;

end

end

endmodule

먼저 입력에는 clock pulse인 Clk와 reset인 rst, 새로운 입력인 inx가 있고, 아웃풋에는 상태를 의미하는 state와 출력 z를 의미하는 out이 있다. State의 초기 값은 0000이고 out의 초기 값은 0이다. 그리고 Clk가 0에서 1로 상승하는 positive edge일 때 동작한다. Positive edge일 때 rst의 값이 1이면, 처음 상태로 돌아가기 때문에 state는 0000이 되고, out은 0이 된다. Rst의 값이 0이면, 먼저 현재 상태를 검사한다. 현재 상태가 1101이라면 out의 값이 1이 되고, 1101이 아니라면 out은 0이 된다. 현재 상태를 확인한 후에는 시프트가 일어난다. State에 저장되어 있는 데이터가 왼쪽으로 shift되고, state[0]에는 새로운 입력 inx가 저장된다. 이제 시뮬레이션 코드를 작성하면 다음과 같다.

module detector\_tb();

reg Clk, rst, inx;

wire out;

wire[3:0] state;

detector detector\_tb(

.Clk(Clk),

.rst(rst),

.inx(inx),

.out(out),

.state(state));

always #10 Clk = ~Clk;

always #30 inx = ~inx;

initial begin

Clk <= 0;

rst <= 0;

inx <= 0;

#20 rst <= 0;

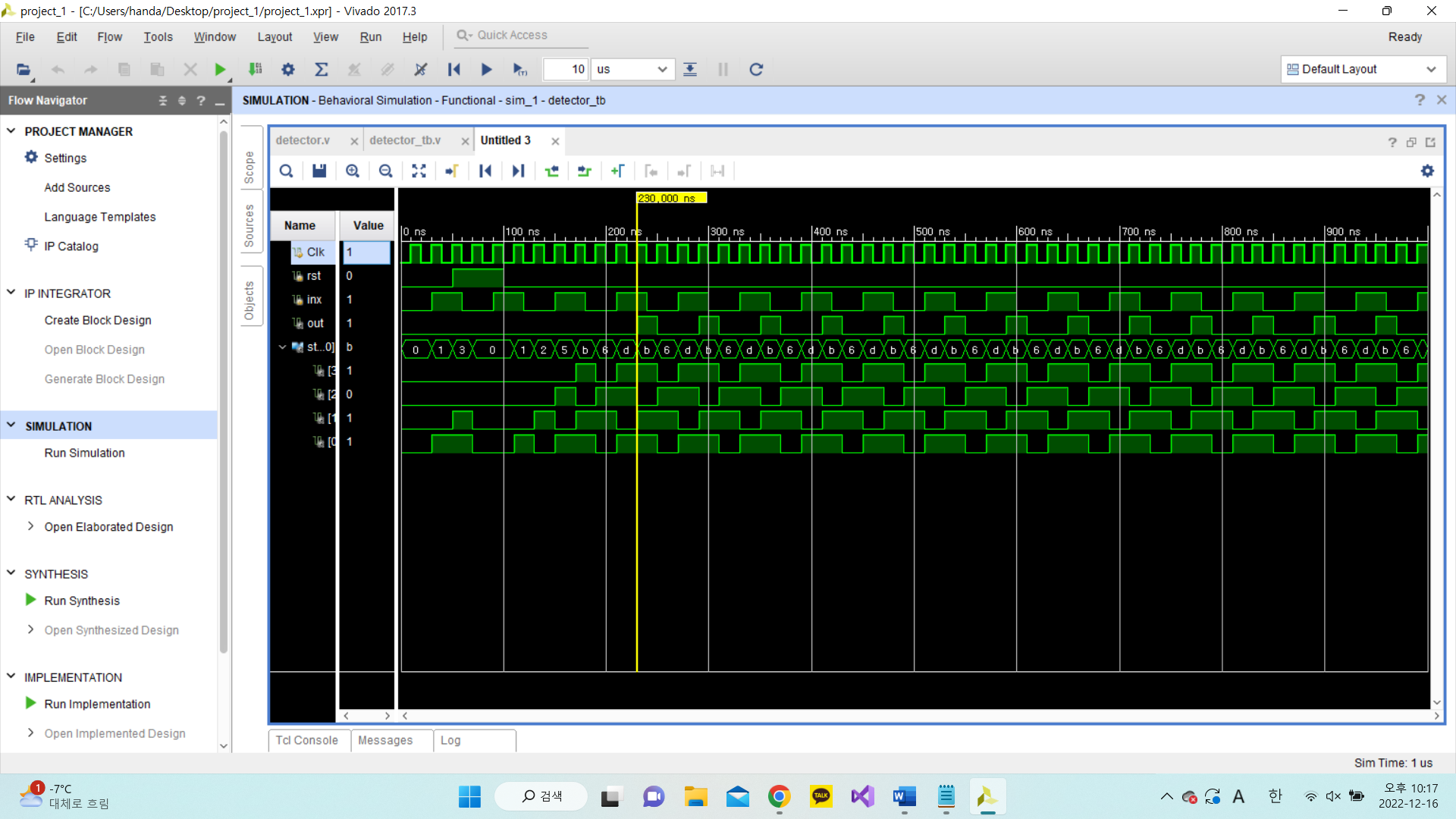
#30 rst <= 1;

#50 rst <= 0;

end

endmodule

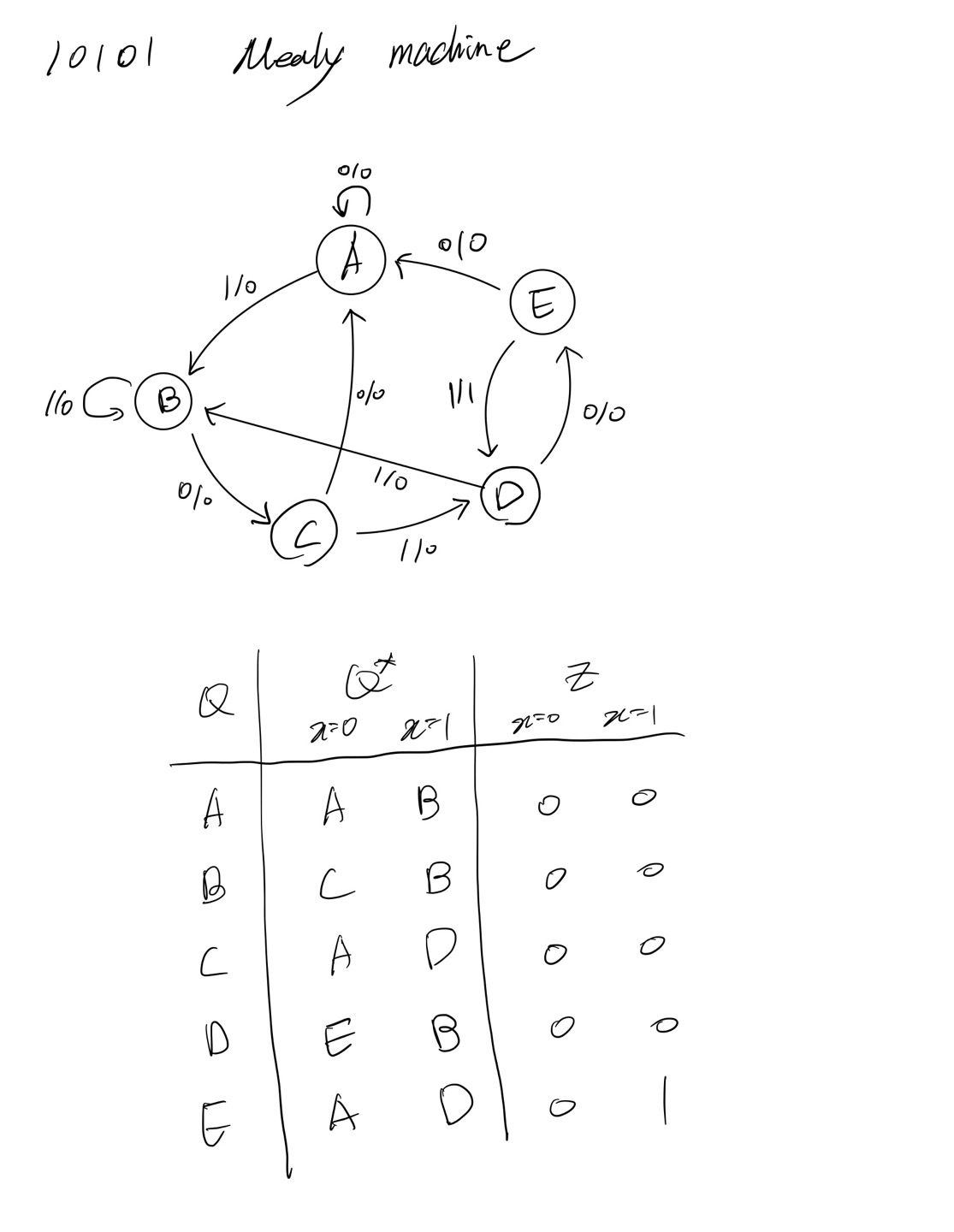
먼저 Clk, rst, inx, out, state를 모두 정의한 후 디자인 코드의 변수와 각각 연결시켜준다. Clk, rst, inx의 초기 값은 모두 0으로 설정하고, Clk는 10ns마다 반전되도록 하고, inx는 30ns마다 반전되도록 하며, rst는 20ns에 0, 30ns 후에 1로 변화하고 다시 50ns 후에 0으로 변화한다. 즉, 50ns 뒤에 0에서 1로 값이 변하고, 50ns 동안 1을 유지하며 이후로는 0의 값을 갖는다. 이 시뮬레이션 코드로 시뮬레이션을 작동시키면 그 결과는 다음과 같다.



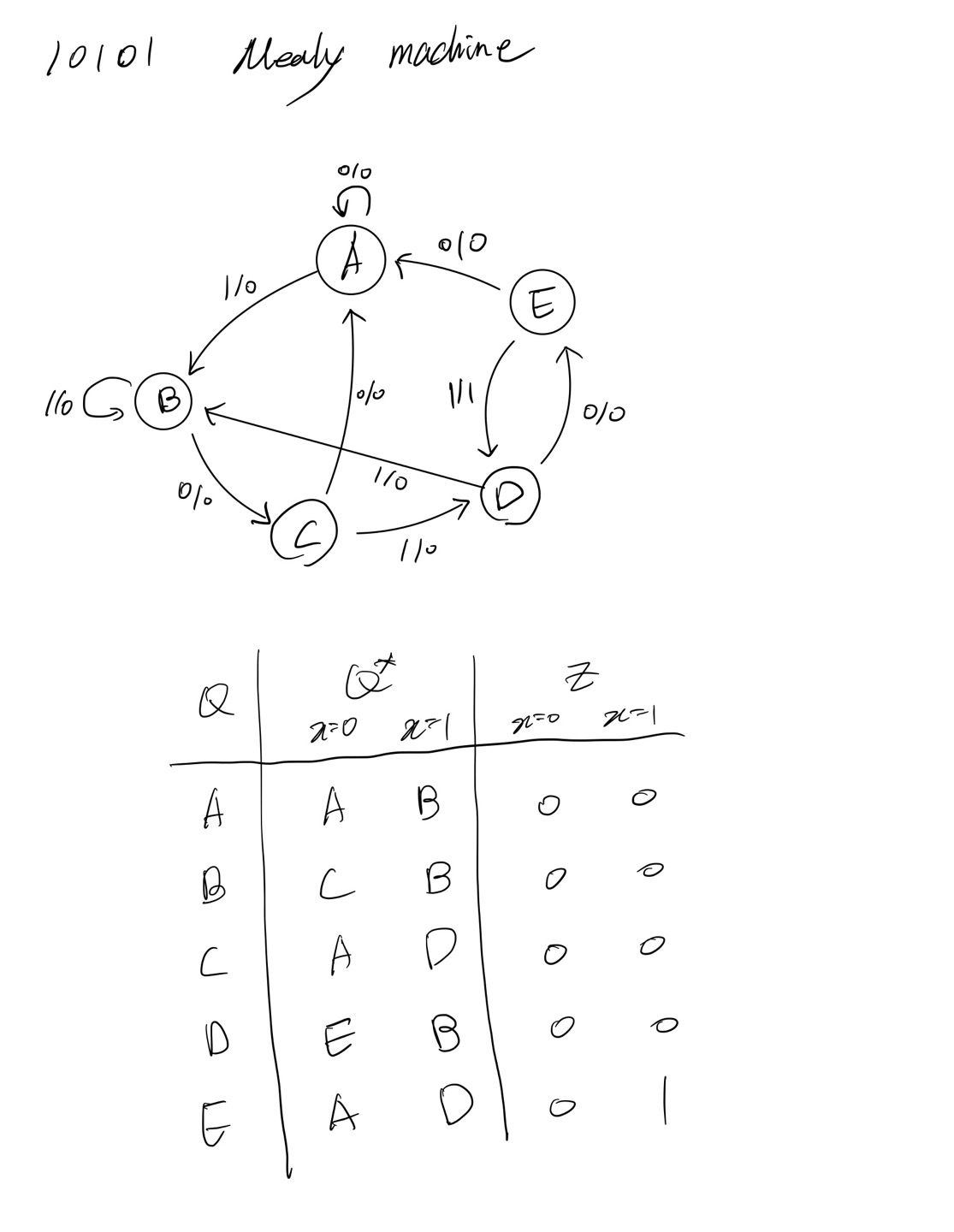
위의 나와있는 예시는 기존 상태가 1101이고, inx가 1, rst가 0인 positive edge 상황이다. Positive edge에서 동작을 실행하는데, rst의 값이 0이므로 reset되지 않는다. 기존 상태가 1101이므로 새로운 입력과 상관없이 out의 값은 1이 된다. 그리고 1101이었던 state가 왼쪽으로 시프트 되고, state[0]에 1이 입력되어 상태는 1011로 변화한다. 다음 positive edge에서는 기존 상태가 1011이므로 out의 값이 0이 되고, inx가 0이므로 상태는 0110으로 변화한다.

**2.**

Mealy machine은 현재 상태와 입력이 출력에 직접적인 영향을 미치는 방식이다. 10101을 감지하는 mealy machine의 상태도는 다음과 같다.



Mealy machine으로 설계할 때는 총 5개의 상태가 필요한데, 이는 후에 설계할 moore machine의 상태의 수보다 하나 적은 수이다. 초기 상태를 A라고 할 때, 1이 입력되면 다음 상태는 B가 되고, 0이 입력되면 다음 상태는 그대로 A가 된다. 현재 상태가 B인 상태에서 0이 입력되면 다음 상태는 C가 되고, 1이 입력되면 다음 상태는 그대로 B가 된다. C인 상태에서 1이 입력되면 다음 상태는 D가 되고 이는 101이 입력된 것을 의미하고, 0이 입력되면 다음 상태는 초기 상태인 A가 되어 처음부터 다시 입력 받는다. 현재 상태가 D인 상태에서 0이 입력되면 다음 상태는 E가 되고 이는 1010이 입력되었음을 의미하고, 1이 입력되면 다음 상태는 B가 되어 1이 입력된 상태에서 다시 입력을 받는다. 현재 상태가 E인 상태에서 1이 입력되면 10101이 입력되었다는 것을 의미하기 때문에 출력 Z의 값이 1이 되고, 101을 입력 받은 상태인 D가 다음 상태가 된다. 현재 상태가 E일 때 1이 아닌 0이 입력되면 다음 상태는 초기 상태인 A가 되어 처음부터 검사를 시작한다. 이를 상태표로 나타내면 다음과 같다.



위의 상태표를 보면 현재 상태와 X의 값 모두가 출력 Z을 결정한다는 것을 알 수 있다. 이제 이를 바탕으로 Verilog 코드를 작성한다. 10101을 검사하는 mealy machine의 디자인 코드는 다음과 같다.

module mealy(

input Clk, rst, inx,

output out, state

);

reg out = 0;

reg[3:0] state = 4'b0000;

always @(posedge Clk) begin

if (rst == 1)

begin

state = 4'b0000;

out = 0;

end

else

begin

if(state == 4'b1010 && inx == 1)

begin

out = 1;

state = 4'b0101;

end

else

begin

state = state << 1;

state[0] = inx;

out = 0;

end

end

end

endmodule

먼저 clock pulse인 Clk, reset인 rst, 새로운 입력인 inx, 출력 z와 상태 state를 선언한 후 state의 초기 값을 0000, out의 초기 값을 0으로 설정한다. Clk가 0에서 1로 상승하는 positive edge일 때 동작을 실행하는데, 이 때 rst가 1이라면 state를 초기 값이 0000, out을 초기 값인 0으로 돌아가도록 한다. Positive edge일 때 rst가 0이면 if문을 통해 동작을 구분하는데, 먼저 기존 state가 1010이고, 새로 입력 받은 데이터가 1이라면 10101이 입력되었다는 것을 의미하므로, 출력 out의 값이 1이 되고 state는 0101이 되어 다음 입력이 0과 1이 입력되면 10101이 될 수 있도록 한다. State가 1010이 아니거나 새로 입력 받은 데이터가 1이 아닐 경우에는 state의 값을 왼쪽으로 shift한다. 그리고 state[0]에 inx가 입력된다. 이렇게 하면 디자인 코드가 완성되고, 이제 시뮬레이션 코드를 작성한다. 시뮬레이션 코드는 다음과 같다.

module mealy\_tb();

reg Clk, rst, inx;

wire out;

wire[3:0] state;

mealy mealy\_tb(

.Clk(Clk),

.rst(rst),

.inx(inx),

.out(out),

.state(state));

always #10 Clk = ~Clk;

always #20 inx = ~inx;

initial begin

Clk <= 0;

rst <= 0;

inx <= 0;

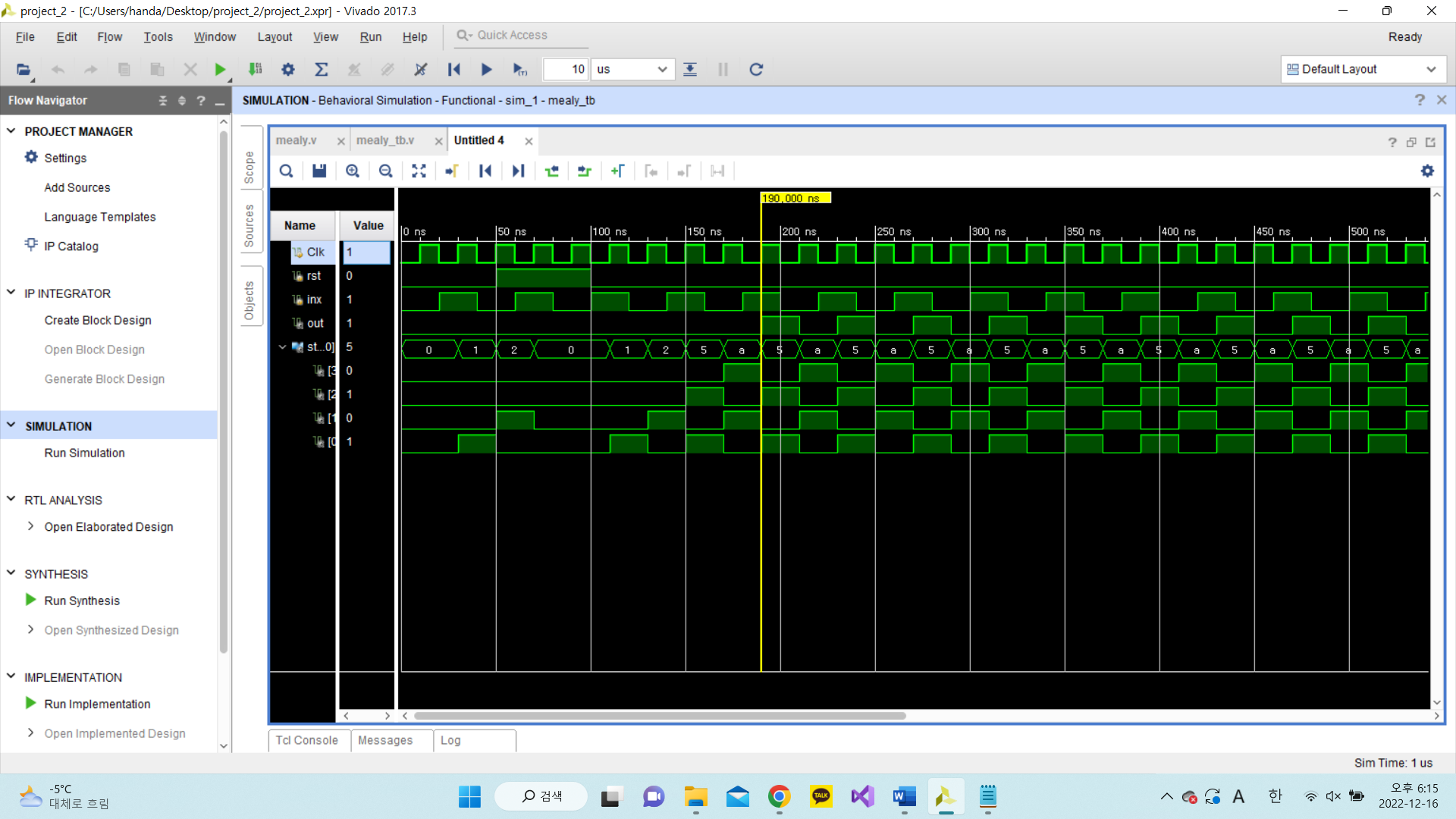
#50 rst <= 1;

#50 rst <= 0;

end

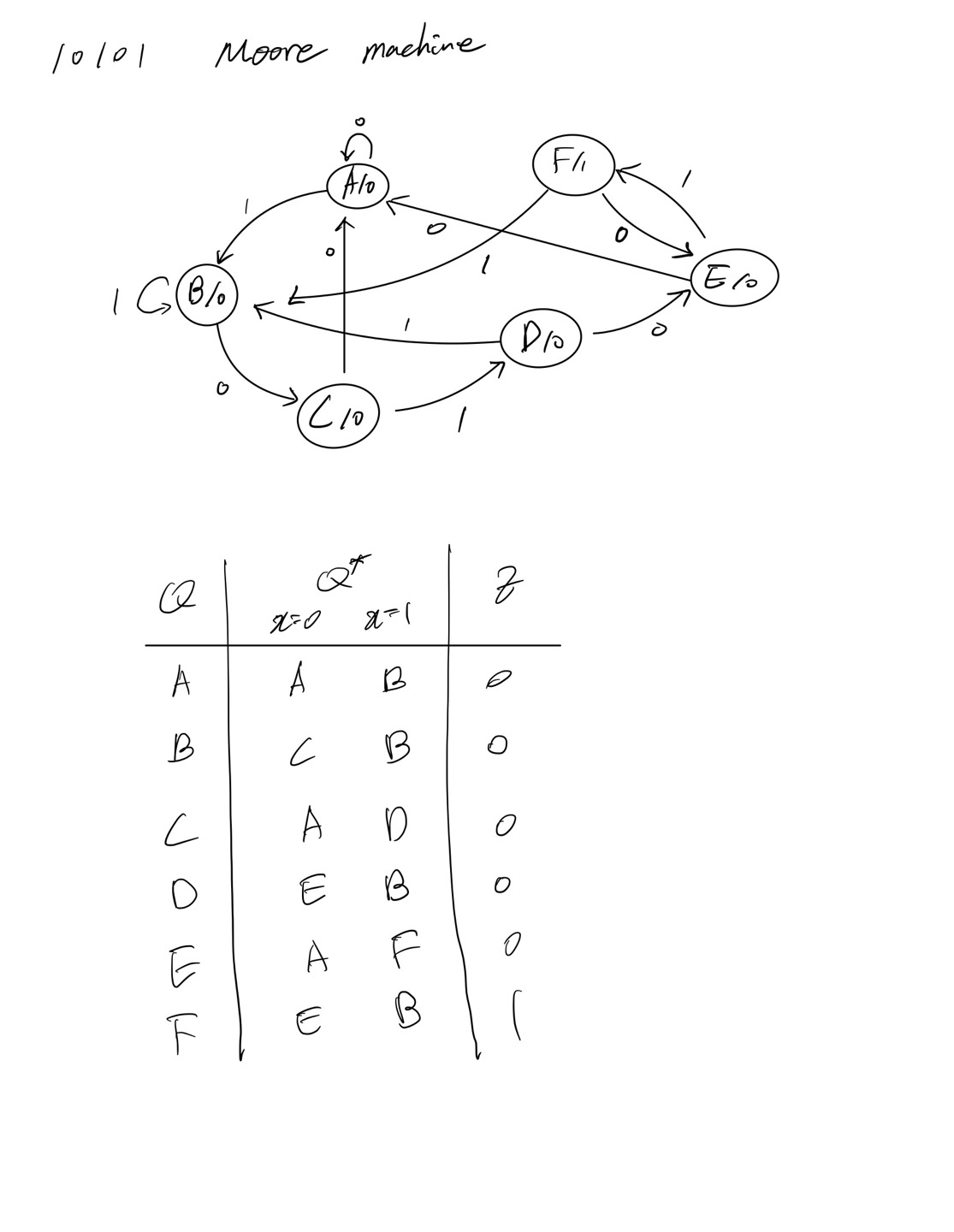
endmodule

먼저 Clk, rst, inx, state, out을 디자인 코드와 각각 연결시킨다. 이후 Clk는 10ns마다 반전되고, inx는 20ns마다 반전되도록 설계한다. 그리고 Clk, rst, inx의 초기 값을 모두 0으로 설정하고, rst는 50ns 뒤에 1이 되고, 다시 50ns 뒤에 0이 되도록 한다. 즉, rst는 50ns에서 100ns동안 1이고 나머지 시간 동안에는 0이 된다. 이렇게 시뮬레이션 코드를 작성한 후에는 시뮬레이션을 실행하여 결과를 확인한다. 시뮬레이션 결과는 다음과 같다.

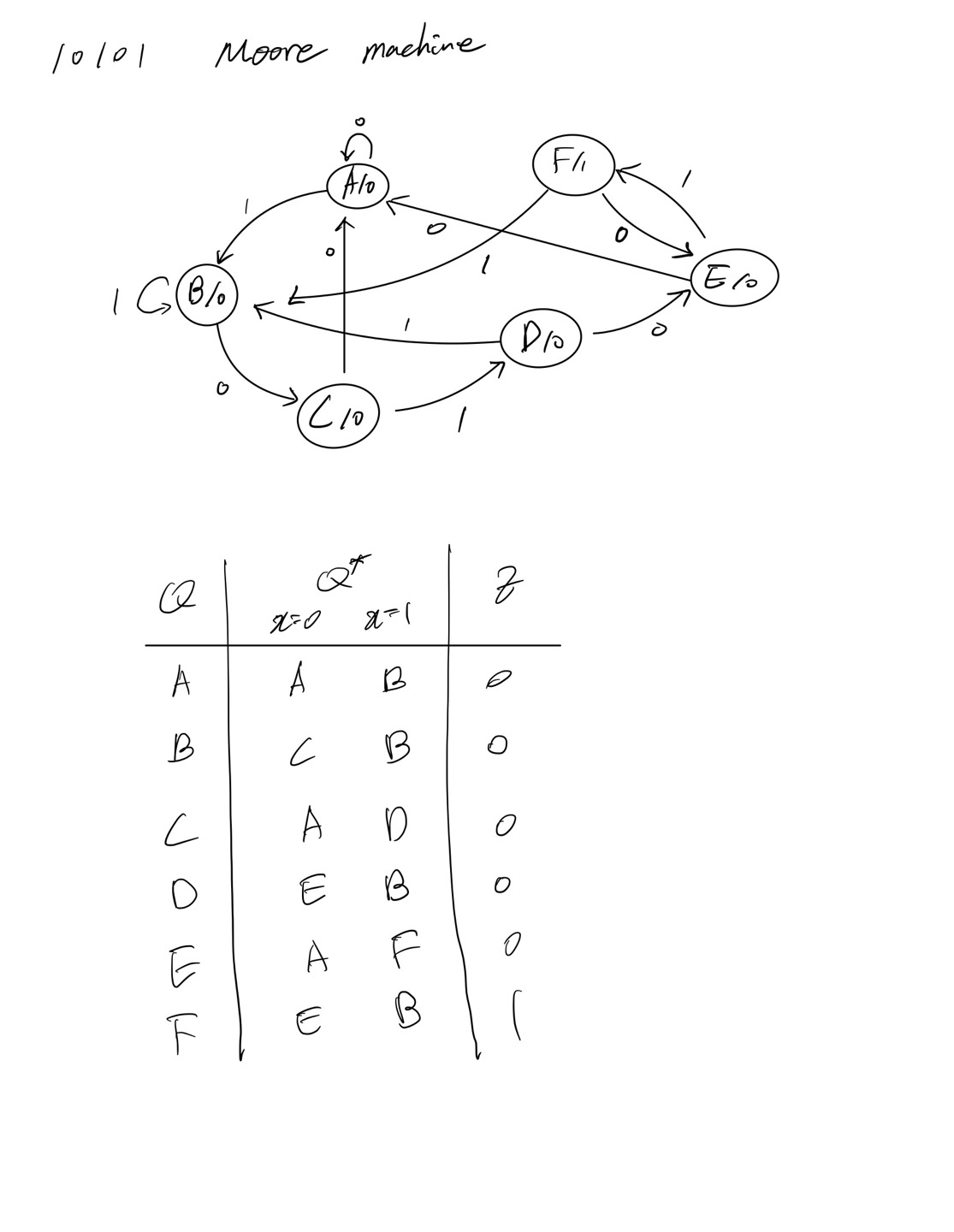


위 시뮬레이션 결과에서 보여지는 예시는 기존 state가 1010이고 새로운 입력 inx가 1인 상황이다. Clk가 0에서 1로 상승하는 positive edge이기 때문에 동작을 실행하고, rst의 값이 0이기 때문에 reset되지 않는다. 디자인 코드에서 state가 1010이고 inx가 1인 경우에는 out의 값이 1이 되도록 설계하였기 때문에 out이 1이 출력되었고, state는 디자인 코드에서 설계한 것과 같이 0101이 되었다. 다음 positive edge에서는 inx가 0이기 때문에 0101이 왼쪽으로 시프트 된 후 state[0]에 0이 입력된 1010이 되었고, 그 다음 positive edge에서는 현재 상태가 1010이고, inx가 1이기 때문에 out이 1이 출력되었다.

다음으로는 동일한 장치를 moore machine으로 설계하는 경우이다. 이 moore machine의 상태도는 다음과 같다.



처음 상태를 A라고 하면, 처음 상태에서 0이 입력되면 다음 상태도 그대로 0이된다. A에서 1이 입력되면 다음 상태는 B가 되고, 이는 처음 1이 입력되었다는 것을 의미한다. B에서 1이 입력되면 그대로 다음 상태는 B가 되고, B에서 0이 입력되면 다음 상태는 C가 되는데, 이는 10이 입력되었다는 것을 의미한다. C에서 0이 입력되면 다음 상태는 초기 상태인 A로 돌아가 다시 처음부터 입력 받기 시작한다. C에서 1이 입력되면 다음 상태는 D가 되는데, 이는 10101 중 101이 입력되었다는 것을 의미한다. D에서 1이 입력되면 다음 상태는 B가 되는데, 1010이 아닌 1011이 입력되었으므로 다시 B로 돌아가서 1 이후의 데이터를 입력 받아야 하기 때문이다. D에서 0이 입력되면 다음 상태는 E가 되고, 이때 1010이 입력되었다는 것을 의미한다. E에서 0이 입력되면 처음 상태인 A로 돌아가 처음부터 입력받고, E에서 1이 입력되면 다음 상태는 F가 된다. F는 10101이 입력된 상태이므로 상태가 F가 되면 1을 출력한다. F에서는 0을 입력 받으면 E로 돌아가고, 1을 입력 받으면 B로 돌아간다. 이를 상태표로 나타내면 다음과 같다.



앞서 나왔던 것처럼 mealy machine은 5개의 상태를 사용했지만 moore machine은 6개의 상태를 사용한 것을 알 수 있다. 이제 이를 바탕으로 Verilog 디자인 코드를 작성하면 다음과 같다.

module moore(

input Clk, rst, inx,

output out, state

);

reg[4:0] state = 5'b00000;

reg out = 1'b0;

always @(posedge Clk) begin

if (rst == 1)

begin

state = 4'b00000;

out = 0;

end

else

begin

if(state == 5'b10101)

out = 1;

else

out = 0;

state = state << 1;

state[0] = inx;

end

end

endmodule

moore machine에 사용된 입력과 출력은 Clk, rst, inx, state, out이고, 이 때 state는 5비트를 사용하였다. State의 초기 값을 00000으로, out의 초기 값을 0으로 설정한다. 이 후 clock pulse가 0에서 1로 상승하는 positive edge에서 동작을 실행하는데, rst의 값이 1이면 state와 out의 값을 초기 상태로 되돌리는 reset 동작을 실행한다. Positive edge에서 rst가 0이면, 먼저 state가 10101인지 확인한다. State가 10101이라는 것은 상태도의 F를 의미하는 것이므로 out의 값이 1이 된다. 만약 state가 10101이 아니라면 out의 값은 0이 된다. 현재 state를 확인한 후에는 state의 데이터를 왼쪽으로 shift하고 state[0]에는 새로운 입력 inx를 저장한다. 이제 시뮬레이션 코드를 작성하면 다음과 같다.

module moore\_tb();

reg Clk, rst, inx;

wire out;

wire[4:0] state;

moore moore\_tb(

.Clk(Clk),

.rst(rst),

.inx(inx),

.out(out),

.state(state));

always #10 Clk = ~Clk;

always #20 inx = ~inx;

initial begin

Clk <= 0;

rst <= 0;

inx <= 0;

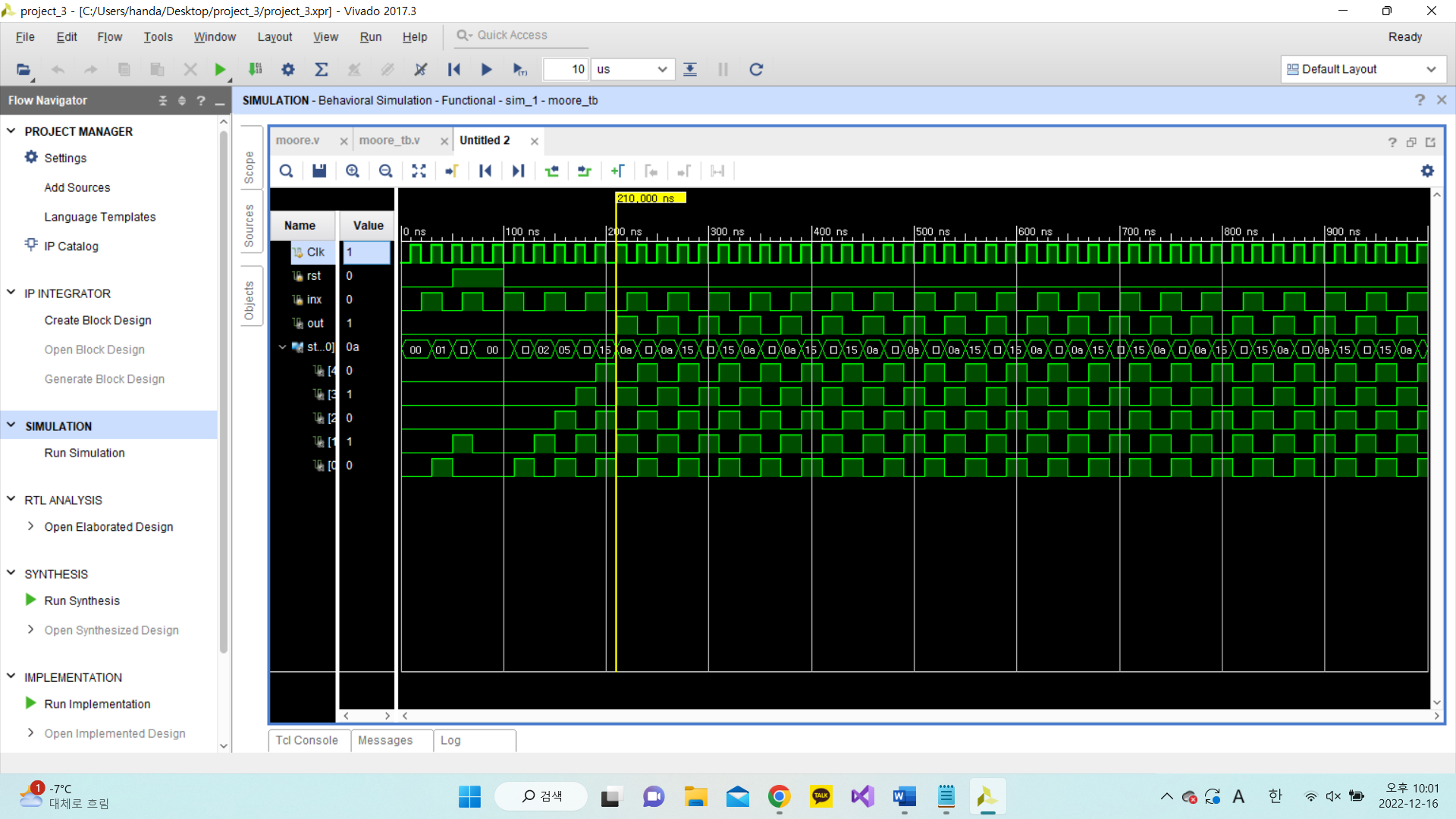
#50 rst <= 1;

#50 rst <= 0;

end

endmodule

먼저 Clk, rst, inx, state, out을 디자인 코드의 변수들과 각각 연결시킨다. 이 후 Clk는 10ns마다 반전되고, inx는 20ns마다 반전되도록 설정하고 Clk, rst, inx의 초기 값을 모두 0으로 설정한다. Rst는 초기 값인 0이 50ns동안 지속되다가 50ns에서 1로 변화하고 다시 50ns동안 1로 유지되다가 100ns에서 0으로 변화한다. 이제 디자인 코드와 시뮬레이션 코드를 바탕으로 시뮬레이션을 실행하면 그 결과는 다음과 같다.



위의 나와있는 예시는 기존 상태가 10101이고, inx가 0, rst가 0인 positive edge 상황이다. Positive edge에서 rst의 값이 0이므로 reset되지 않는다. 기존 상태가 10101이므로 이번 positive edge에서 새로운 입력과 상관없이 out의 값은 1이 된다. 새로운 입력이 0이기 때문에 10101이라는 데이터가 왼쪽으로 shift되고, state[0]에 0이 입력되어 상태는 01010이 된다. 다음 positive edge에서 다시 새로운 입력 1을 입력 받으면 시프트 과정을 통해 상태가 10101이 되고, 그러면 그 다음 positive edge의 현재 상태가 10101이므로 출력 out의 값이 1이 된다.