14주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

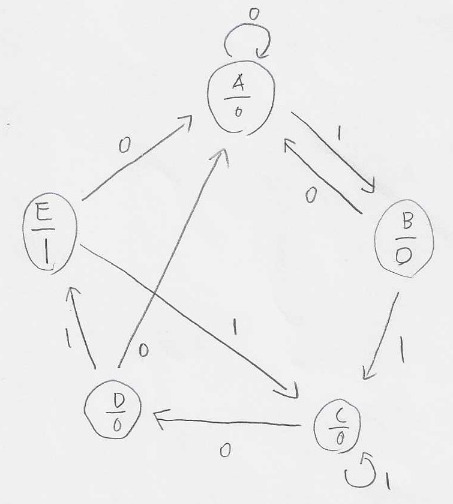
**1.**

Moore Machine은 실습시간에 구현한 Mealy Machine과 달리 현재 상태에 의해서만 출력값이 결정된다. 따라서 Sequence Detector를 Moore Machine으로 구현하게 되면, 같은 Sequence를 Detect하는 Mealy Machine에 비해 1개의 state를 더 가지게 된다.

1101 Moore machine을 설계하기 위해 state를 정의하고 각 state 간의 전이 관계를 state diagram과 state table로 나타낸 그림은 아래와 같다.



[1101 Moore machine의 State Table]

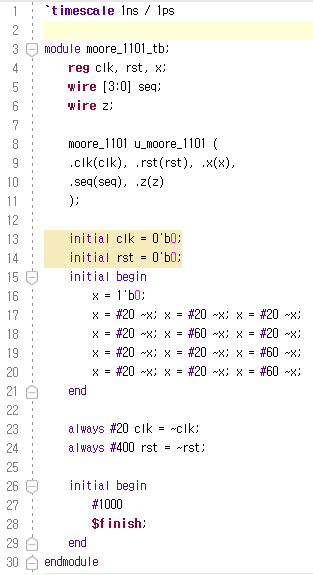
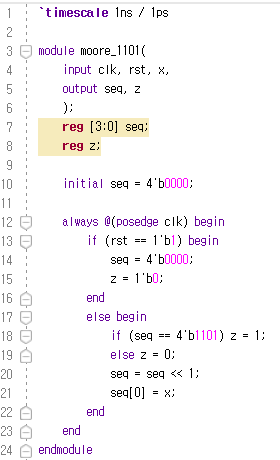


[1101 Moore machine의 State Diagram]

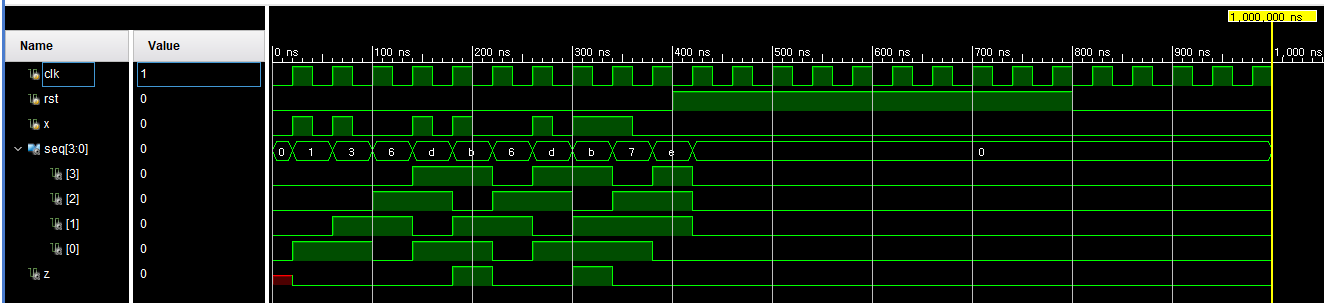
이때, A는 target sequence 중 들어온 입력이 없는 상태, B는 1이 들어온 상태, C는 11이 들어온 상태, D는 110이 들어온 상태, E는 1101이 들어온 상태로 정의했고, 상태 E일 때 target sequence가 완전히 들어온 상태이므로 출력값 1을 가지고, 나머지 상태들은 모두 출력값 0을 가지게 했다. 또한, 이 Sequence Detector는 Overlapping을 허용하므로

상태 E에서 1의 입력이 들어올 때 상태 B가 아닌 C로 전이되게 설계했다. (E에서 1의 입력이 들어오는 경우, 1101101과 같은 형태로 overlapping된 sequence도 detect 가능함)

이와 같은 1101 Moore Machine을 Verilog로 구현하면 아래와 같다.



[1101 Moore machine의 Verilog Code]



[1101 Moore machine의 Simulation 결과]

위 코드는 실습시간에 작성한 1101 Mealy Machine 코드를 응용하여 작성한 것이다.

Moore Machine은 현재 상태에 의해서만 출력값이 결정되므로 1101이 입력으로 들어오는 즉시 output이 1이 되지 않고, 한 clk 주기 늦게 입력값과 관계없이 출력값(z)이 1이 된다.

위 코드에서 seq는 입력된 sequence를, z는 출력값, x는 입력값을 의미한다.

위 코드에서 reset 신호가 들어오는 경우(rst=1) 입력된 sequence와 출력값을 0으로 초기화 하고, reset 신호가 없는 경우(rst=0) shift register처럼 seq를 왼쪽(<<, lshift 사용)으로 1칸씩 옮기고 입력값(x)을 seq의 LSB(seq[0])에 저장하게 했다.

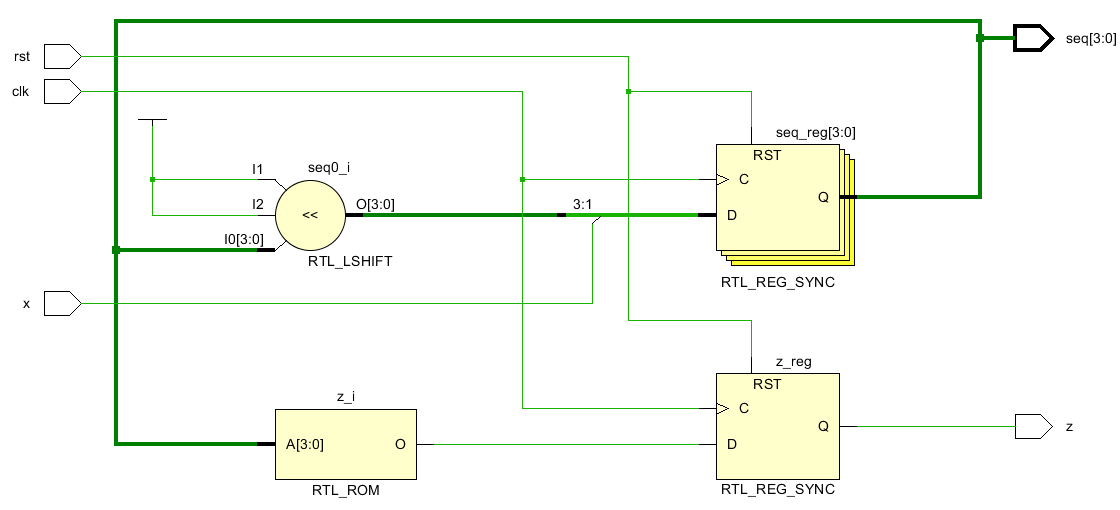
이때, Moore Machine은 현재 상태에서만 출력값이 결정되는 특징으로 인해 seq에 입력값 x가 들어와 1101이 되더라도 바로 출력값 z가 1이 되지 않는다.

따라서 seq에 1101이 저장된 후, 다음 주기에 출력값 z가 1이 될 수 있게 seq의 값을 갱신하는 코드보다 위에 seq의 값이 인지 검사하는 코드를 삽입해 이를 구현하였다.

(이전 주기의 seq가 이면 z가 1을, 다른 값을 가지면 z가 0을 가지게 함)

모든 동작은 clk가 posedge(rising edge)일 때 작동되게 했고, testbench 코드는 실습시간에 작성한 코드와 거의 동일하게 작성하여 입력되는 sequence에 따라 출력값의 변화를 볼 수 있도록 하였다.

아래의 사진은 위에서 구현한 1101 Moore machine의 Schematic이다.



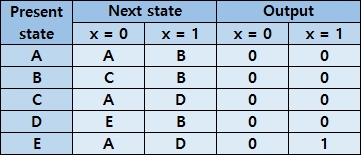
[1101 Moore machine의 Schematic]

**2.**

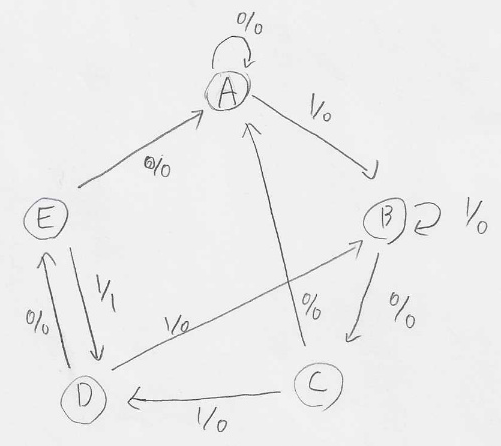
- 10101 Mealy machine

Mealy Machine은 실습시간에 구현한 코드와 같이 현재 상태와 입력값에 의해 출력값이 결정된다. 따라서 Sequence Detector를 Mealy Machine으로 구현하게 되면, 같은 Sequence를 Detect하는 Moore Machine에 비해 1개의 state를 덜 가지게 되고, target sequence가 감지되는 즉시 출력값이 1이 된다.

10101 Mealy machine을 설계하기 위해 state를 정의하고 각 state 간의 전이 관계를 state diagram과 state table로 나타낸 그림은 아래와 같다.



[10101 Mealy machine의 State Table]

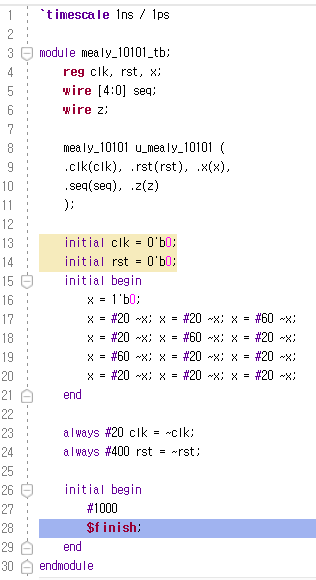
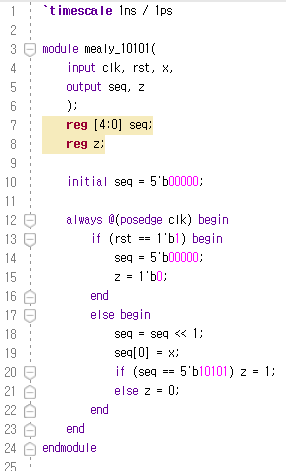


[10101 Mealy machine의 State Diagram]

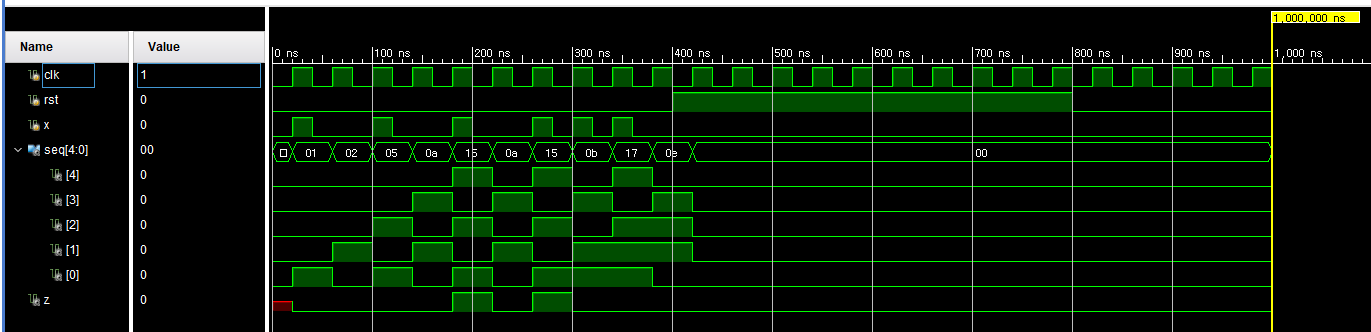
이때, A는 target sequence 중 들어온 입력이 없는 상태, B는 1이 들어온 상태, C는 10이 들어온 상태, D는 101이 들어온 상태, E는 1010이 들어온 상태로 정의했고, 상태 E일 때

1의 입력이 들어오면 10101이 되어 target sequence가 감지되므로 이 경우에만 출력값 1을 가지고, 다른 전이들은 모두 출력값 0을 가지게 했다. 또한, 이 Sequence Detector는 Overlapping을 허용하므로 상태 E에서 1의 입력이 들어올 때 상태 A가 아닌 D로 전이되게 설계했다. (E에서 1의 입력이 들어오는 경우, 1010101과 같은 형태로 overlapping된 sequence도 detect 가능함)

이와 같은 10101 Mealy Machine을 Verilog로 구현하면 아래와 같다.



[10101 Mealy machine의 Verilog Code]



[10101 Mealy machine의 Simulation 결과]

위 코드는 실습시간에 작성한 1101 Mealy Machine 코드를 응용하여 작성한 것이다.

Mealy Machine은 현재 상태와 입력값 모두에 의해 출력값이 결정되므로 10101이 입력으로 들어오는 즉시 output이 1이 된다.

위 코드에서 seq는 입력된 sequence를, z는 출력값, x는 입력값을 의미한다.

이때, 10101 Mealy machine은 감지할 target sequence가 5bit이므로 reg [4:0]로 선언하였다. (5비트의 크기를 가짐)

위 코드에서 reset 신호가 들어오는 경우(rst=1) 입력된 sequence와 출력값을 0으로 초기화 하고, reset 신호가 없는 경우(rst=0) shift register처럼 seq를 왼쪽(<<, lshift 사용)으로 1칸씩 옮기고 입력값(x)을 seq의 LSB(seq[0])에 저장하게 했다.

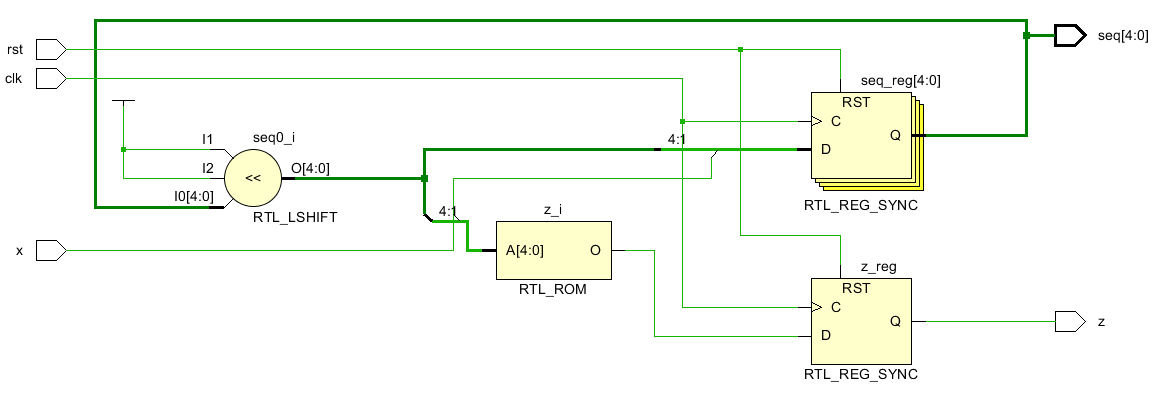
이때, Mealy Machine은 현재 상태와 입력값 모두에 의해 출력값이 결정되는 특징으로 인해 seq에 입력값 x가 들어와 10101이 되면 바로 출력값 z가 1이 된다.

따라서 seq에 10101이 저장된 후, 현재 주기에 출력값 z가 1이 될 수 있게 seq의 값을 갱신하는 코드보다 아래에 seq의 값이 인지 검사하는 코드를 삽입해 이를 구현하였다.

(현재 주기의 seq가 이면 z가 1을, 다른 값을 가지면 z가 0을 가지게 함)

모든 동작은 clk가 posedge(rising edge)일 때 작동되게 했고, testbench 코드는 실습시간에 작성한 코드와 거의 동일하게 작성하여 입력되는 sequence에 따라 출력값의 변화를 볼 수 있도록 하였다.

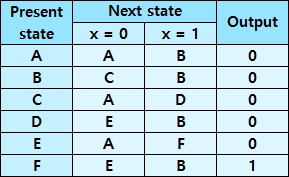
아래의 사진은 위에서 구현한 10101 Mealy machine의 Schematic이다.



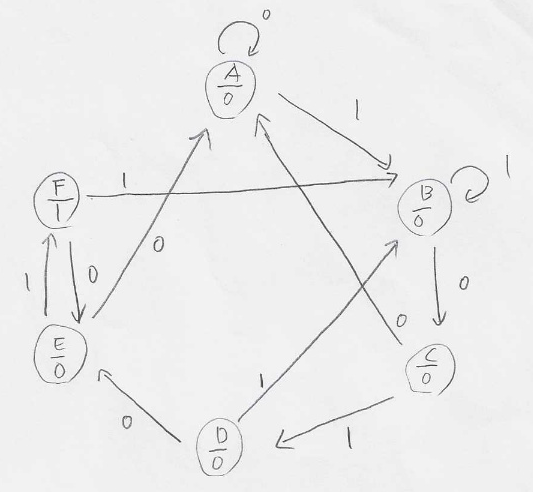
[10101 Mealy machine의 Schematic]

- 10101 Moore machine

10101 Moore machine을 설계하기 위해 state를 정의하고 각 state 간의 전이 관계를 state diagram과 state table로 나타낸 그림은 아래와 같다.



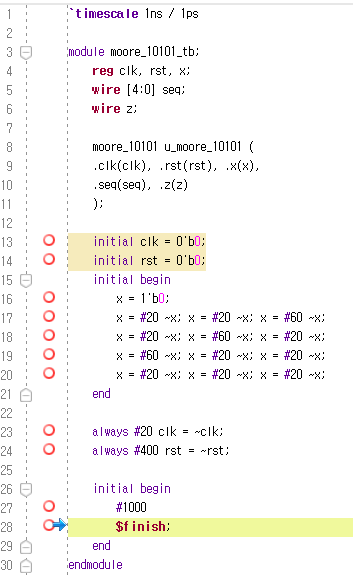
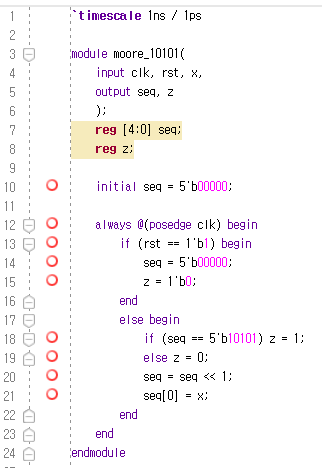
[10101 Moore machine의 State Table]



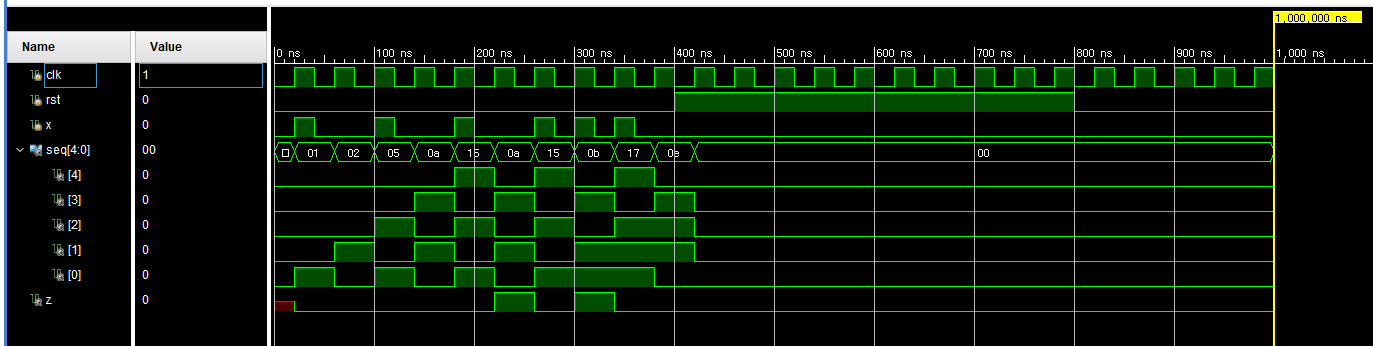
[10101 Moore machine의 State Diagram]

이때, A는 target sequence 중 들어온 입력이 없는 상태, B는 1이 들어온 상태, C는 10이 들어온 상태, D는 101이 들어온 상태, E는 1010이 들어온 상태, F는 10101이 들어온 상태로 정의했고, 상태 F일 때 target sequence가 완전히 들어온 상태이므로 출력값 1을 가지고, 나머지 상태들은 모두 출력값 0을 가지게 했다. 또한, 이 Sequence Detector는 Overlapping을 허용하므로 상태 F에서 0의 입력이 들어올 때 상태 A가 아닌 E로 전이되게 설계했다. (F에서 0의 입력이 들어오는 경우, 1010101과 같은 형태로 overlapping된 sequence도 detect 가능함)

이와 같은 10101 Moore Machine을 Verilog로 구현하면 아래와 같다.



[10101 Moore machine의 Verilog Code]



[10101 Moore machine의 Simulation 결과]

위 코드는 1번에서 작성한 1101 Moore Machine 코드를 응용하여 작성한 것이다.

Moore Machine은 현재 상태에 의해서만 출력값이 결정되므로 10101이 입력으로 들어오는 즉시 output이 1이 되지 않고, 한 clk 주기 늦게 입력값과 관계없이 출력값(z)이 1이 된다.

위 코드에서 seq는 입력된 sequence를, z는 출력값, x는 입력값을 의미한다.

이때, 10101 Moore machine은 감지할 target sequence가 5bit이므로 reg [4:0]로 선언하였다. (5비트의 크기를 가짐)

위 코드에서 reset 신호가 들어오는 경우(rst=1) 입력된 sequence와 출력값을 0으로 초기화 하고, reset 신호가 없는 경우(rst=0) shift register처럼 seq를 왼쪽(<<, lshift 사용)으로 1칸씩 옮기고 입력값(x)을 seq의 LSB(seq[0])에 저장하게 했다.

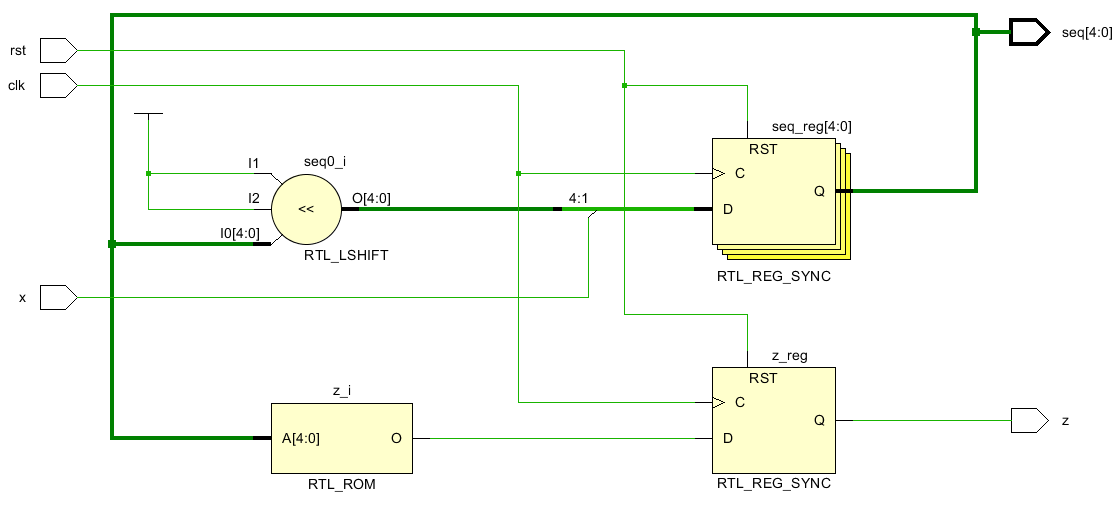
이때, Moore Machine은 현재 상태에서만 출력값이 결정되는 특징으로 인해 seq에 입력값 x가 들어와 10101이 되더라도 바로 출력값 z가 1이 되지 않는다.

따라서 seq에 10101이 저장된 후, 다음 주기에 출력값 z가 1이 될 수 있게 seq의 값을 갱신하는 코드보다 위에 seq의 값이 인지 검사하는 코드를 삽입해 이를 구현하였다.

(이전 주기의 seq가 이면 z가 1을, 다른 값을 가지면 z가 0을 가지게 함)

모든 동작은 clk가 posedge(rising edge)일 때 작동되게 했고, testbench 코드는 실습시간에 작성한 코드와 거의 동일하게 작성하여 입력되는 sequence에 따라 출력값의 변화를 볼 수 있도록 하였다.

아래의 사진은 위에서 구현한 10101 Moore machine의 Schematic이다.



[10101 Moore machine의 Schematic]