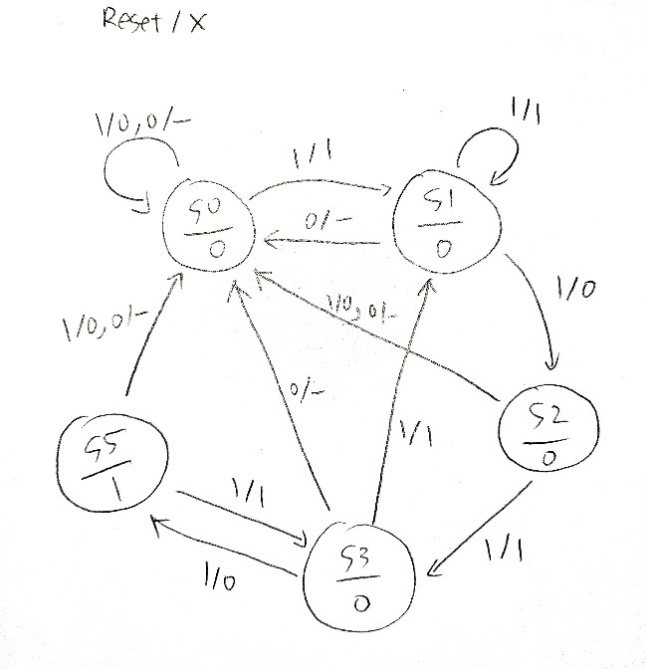
Copyright 2022. (Hyeonsu Lee) All rights reserved.

1. Seq\_Detector\_Moore

(1)설계의 핵심 / 설계 방법 및 시퀀스 검출기 동작 원리





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | 의미 | Next  State  (RESET=’1’일 때) | | Present output |
| X=’0’ | X=’1’ |
| s0 | Initial state | s0 | s1 | 0 |
| s1 | ‘1010’의 첫번째 1을 받은 상태 | s2 | s1 | 0 |
| s2 | ‘1010’의 앞 ‘10’ | s0 | s3 | 0 |
| s3 | ‘1010’에서 ‘101’ | s4 | s1 | 0 |
| s4 | ‘1010’ sequence detect | s0 | s3 | 1 |

\*설계의 핵심

- ‘101’에서 ‘0’의 입력을 받았을 때 s3에서 s2로 가지 못한다.(moore machine / 출력값이 state에 따라 바뀌기 때문)

-> ‘10’일 때와 ‘1010’일 때 출력값은 각각 0과 1로 달라야 한다.

- 입력 X가 순서대로 ‘1010’, ‘101010’ 이런식으로 들어와도 마지막에 X=’0’이 들어왔을 때 출력값이 1이 되게 설계해야 한다. (‘Z=1’이 출력되어도, 현재까지 들어온 입력은 모두 유효)

-> s4에서 X=’1’의 입력을 받았을 때, s1이 아니라 s3으로 다시 돌아가는 이유

-RESET=’0’일 때는 입력 X값에 상관없이 무조건 s0으로 돌아가고 출력값은 0이 되게 설계

\*설계 방법 및 시퀀스 검출기 동작 원리

- asynchronous reset이기 때문에 RESET이 0일 때는 state, X 값에 상관없이 항상 s0 state로 돌아간다.

- 출력은 ‘1010’일 때만 1이다.(X가 순서대로 1,0,1,0,1,0이어도 출력은 1이 되어야 한다.)

- RESET=’1’일 때

s0 -> s0 : 초기 상태에서 X=’0’의 입력을 받으면 ‘1010’ sequence를 시작할 수 없다.

s0 -> s1 : X=’1’의 입력을 받으면 ‘1010’의 sequence를 시작할 수 있는 가능성이 생긴다.

s1 -> s1 : ‘1010’의 첫번째 1을 받은 상태에서 X의 값이 1이 들어와도 첫번째 1이라는 사실은 변하지 않는다.

s1 -> s2 : ‘1010’에서 앞의 ‘10’

s2 -> s0 : ‘1010’에서 앞의 ‘10’이 들어온 상태에서 입력 X의 값이 0이 되어버리면 ‘100’이 되어서‘1010’의 sequence가 될 가능성이 없어지므로 초기 상태로 돌아간다.

s2 -> s3 : ‘10’에서 X=’1’의 입력을 받으면 ‘1010’에서 ‘101’이 만들어진다.

s3 -> s1 : ‘101’에서 X=’1’의 입력을 받으면 ‘1011’이 되어 ‘1010’이 될 수 없고 다시 ‘1010’에서 첫번째 1이 될 수 있는 상태, 즉 s1로 돌아간다.

s3 -> s4 : ‘101’에서 X=’0’의 입력을 받으면 ‘1010’의 sequence가 만들어지고 출력값은 1이 된다.

s4 -> s0 : ‘1010’에서 X=’0’의 입력을 받으면 ‘1010’이 될 가능성이 없어지고 초기 상태로 돌아간다.

s4 -> s3 : ‘1010’에서 X=’1’의 입력을 받으면 ‘101’이 만들어지므로 s3의 상태로 가고 s3에서 다시 X=’0’의 입력을 받으면 s4로 가서 전에 출력을 1로 만든다. 이는 ‘1010’이 처음 detect 됐을 때 그 입력값들을 무시하는 것이 아니라 ‘101010’이어도 1이 되게 해야 했기 때문에 이렇게 설계하였다.

(2) 설계한 VHDL 코드와 설명

\*sequence detector 코드

텍스트이(가) 표시된 사진

자동 생성된 설명

- 코드에서 상태를 직관적으로 파악할 수 있도록 states라는 type을 선언하여 s0, s1, s2, s3, s4을 정의하였다.

- state와 next state 모두 s0, s1, s2, s3, s4로 표현할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

-state transition

->RESET과 CLK의 변화에 따라 상태가 변화하기 때문에 sensitivity list에 이 두 입력을 포함시켰다.

->state transition은 state register에서 일어나기 때문에 sequential logic이다. 따라서 CLK에 따라서 상태가 변화한다.

->asynchronous RESET(active low)이기 때문에 CLK에 상관없이 ‘0’이 되기만 하면 초기 상태로 돌아가게 설계하였다.

->CLK의 상승 에지에서 state transition이 일어나도록 설계하였다.

-outputs

->moore machine에서 출력은 present state에 따라서만 변화하기 때문에 sensitivity list에는 state만 포함시켰다.

->앞의 state diagram을 통해 각 state에서의 출력값을 알 수 있고, 코드에서는 각각의 state에 따라 알맞은 출력값들을 인가했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

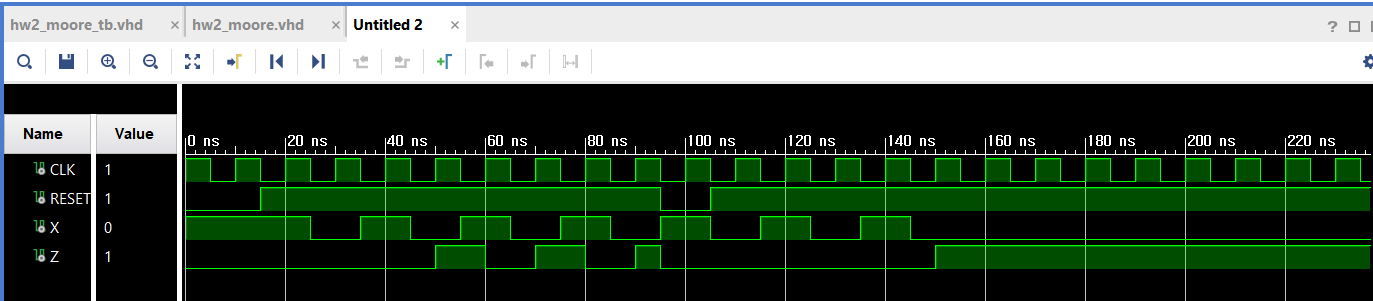
테이블이(가) 표시된 사진

자동 생성된 설명

-무어 머신이기 때문에 next state는 입력에 따라서만 변한다. 따라서 sensitivity list에 X를 포함시켰다.

- 특정 state에서 X에 따른 next state의 변화는 앞서 언급한 state diagram과 state table을 참고하여 코드를 작성하였다.

(2) waveform과 설명 및 분석



-Clk 상승 edge에서의 입력(RESET/X)과 출력(Z)을 확인해야 한다.

-Moore machine의 출력은 state에 따라서만 변한다.

-(RESET이 1일 때만) 클락의 상승 에지에서 X는 순서대로 1,0,1,**0**,1,**0**,1,**0**이고 초기화 후에 다시 0,1,0,1,**0**이므로 결과는 X에서 빨간색으로 표시한 입력이 들어왔을 때 출력(Z)의 값이 1이 되는지를 확인해야 한다. -> 입력의 값이 0으로 바뀌고 다음 CLK의 상승 에지에서 출력의 값이 1로 변한다(올바르게 동작함을 확인할 수 있다.)

-state transition은 CLK의 상승 에지에서 일어나기 때문에 출력 Z도 CLK의 상승 에지에서 변화한다.

-RESET이 0이 되면 바로 출력이 0으로 바뀌는 것을 확인할 수 있다.(Asynchronous reset이기 때문)

10ns RESET=’0’, X=’1’ => Z=’0’

->RESET이 0이기 때문에 s0 상태이고 출력값은 0이다.

20ns RESET=’1’, X=’1’ => Z=’0’

->RESET이 ‘1’이 되어서 활성화된다. s0에서 1/1의 입력을 받으면 s1이 되고 출력은 ‘0’이다.

30ns RESET=’1’, X=’0’ => Z=’0’

->s1에서 1/0의 입력을 받으면 s2가 되고 출력은 ‘0’이다.

40ns RESET=’1’, X=’1’ => Z=’0’

->s2에서 1/1의 입력을 받으면 s3이 되고 출력은 ‘0’이다.

50ns RESET=’1’, X=’0’ => Z=’1’

->s3에서 1/0의 입력을 받으면 s4가 되고 출력은 ‘1’이다.

60ns RESET=’1’, X=’1’ => Z=’0’

->s4에서 1/1의 입력을 받으면 s3이 되고 출력은 ‘0’이다.

70ns RESET=’1’, X=’0’ => Z=’1’

->s3에서 1/0의 입력을 받으면 다시 s4로 돌아가고 따라서 출력은 ‘1’이 된다.

80ns RESET=’1’, X=’1’ => Z=’0’

->s4에서 1/1의 입력을 받으면 s3이 되고 출력은 ‘0’이다.

90ns RESET=’1’, X=’0’ => Z=’1’

->s3에서 1/0의 입력을 받으면 다시 s4로 돌아가고 따라서 출력은 ‘1’이 된다.

100ns RESET=’0’, X=’1’ => Z=’0’

->RESET=’0’이므로 X의 값에 상관없이 s0으로 돌아가고 출력은 ‘0’이 된다.

110ns RESET=’1’, X=’0’ => Z=’0’

->s0에서 1/0의 입력을 받으면 여전히 s0이고 출력은 ‘0’이다.

120ns RESET=’1’, X=’1’ => Z=’0’

->s0에서 1/1의 입력을 받으면 s1이 되고 출력은 ‘0’이다.

130ns RESET=’1’, X=’0’ => Z=’0’

->s1에서 1/0의 입력을 받으면 s2가 되고 출력은 ‘0’이다.

140 ns RESET=’1’, X=’1’ => Z=’0’

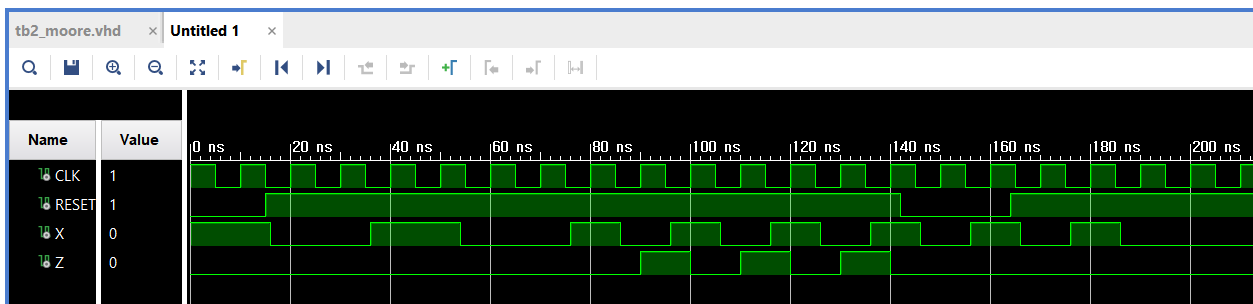
->s2에서 1/1의 입력을 받으면 s3이 되고 출력은 ‘0’이다.

150 ns RESET=’1’, X=’0’ => Z=’1’

->s3에서 1/0의 입력을 받으면 s4가 되고 출력은 ‘1’이다.

-위 결과를 state diagram(혹은 state table)과 비교해보면 SPEC에 맞게 동작한다는 것을 확인할 수 있다.

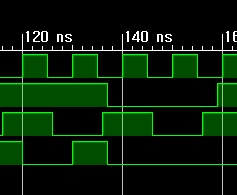
Extra verification



-결과를 분석하는 원리는 앞에서 한 것과 동일하므로 생략한다.

-무어 머신은 state에 의해서만 출력이 변화하기 때문에 state transition이 일어나는 클락의 상승 에지에서 출력이 바뀌는 지를 확인해야 한다.

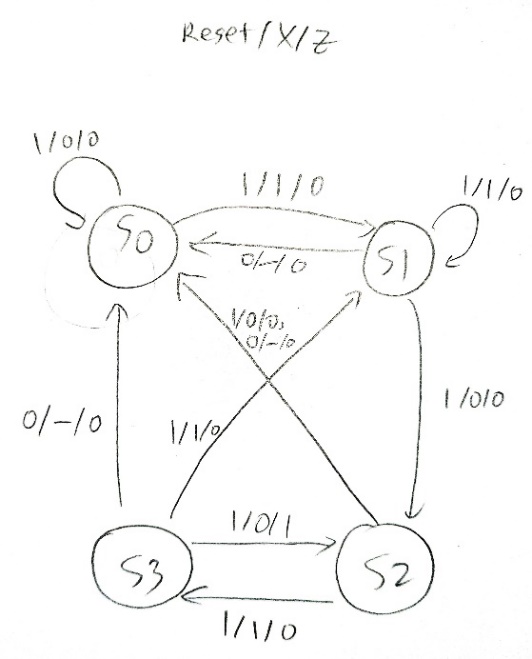
-90ns에서 state가 s4로 바뀌면서 출력도 1로 바뀌는 것을 확인할 수 있다. 마찬가지로 다음 상승 에지에서는 다시 s3으로 바뀌면서 출력도 동시에 0으로 바뀌며 다시 X=’1’의 입력을 받아 s4가 되면서 출력이 1로 변화하는 것을 알 수 있다. Sequence가 detect된 후에도 앞서 들어온 입력이 무효화되지 않고 유효하다는 것도 확인 가능하다.



-RESET=’0’이 되자마자 출력이 0으로 바뀌는지 확인하기 위해 코드를 조금 수정하여 다시 시뮬레이션 해보았다. Asynchronous reset이기 때문에 CLK과 상관없이 바로 Z가 0이 되는 것을 확인할 수 있었다.

2. Seq\_Detector\_Mealy

(1) 설계의 핵심 / 설계 방법 및 시퀀스 검출기 동작 원리



#RESET이 ‘1’일 때

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | 의미 | Next state | | output | |
| X=’0’ | X=’1’ | X=’0’ | X=’1’ |
| s0 | Initial state | s0 | s1 | 0 | 0 |
| s1 | ‘1010’에서 첫번째 1 | s2 | s1 | 0 | 0 |
| s2 | ‘1010’에서 2, 4번째 0 /  즉 연속된 입력이 ‘10’일 때와 ‘1010’일 때 | s0 | s3 | 0 | 0 |
| s3 | ‘1010’에서 ‘101’의 연속된 입력이 들어온 상태 | s2 | s1 | 1 | 0 |

\*설계의 핵심

-moore machine과 달리 입력과 state에 따라 출력이 결정된다.

->s3에서 새로운 state를 추가하지 않고 s2로 돌아갈 수 있는 이유

- 입력 X가 순서대로 ‘1010’, ‘101010’ 이런식으로 들어와도 마지막에 X=’0’이 들어왔을 때 출력값이 1이 되게 설계해야 한다. (‘Z=1’이 출력되어도, 현재까지 들어온 입력은 모두 유효)

-> 입력 X의 값에 따라 s2와 s3 사이를 왔다갔다 할 수 있도록 설계

-RESET=’0’일 때는 입력 X값에 상관없이 무조건 s0으로 돌아가고 출력값은 0이 되게 설계

\*설계 방법 및 시퀀스 검출기 동작 원리

- asynchronous reset이기 때문에 RESET이 0일 때는 state, X 값에 상관없이 항상 s0 state로 돌아간다.

- 출력은 ‘1010’일 때만 1이다.(‘1010’ detect 후에도 앞서 들어온 입력은 유효, X가 순서대로 1,0,1,0,1,0이어도 출력은 1이 되어야 한다.)

- RESET=’1’일 때

s0 -> s0 : 초기 상태에서 X=’0’의 입력을 받으면 ‘1010’ sequence를 시작할 수 없고, 그때의 출력은 0이다.

s0 -> s1 : X=’1’의 입력을 받으면 ‘1010’의 sequence를 시작할 수 있는 가능성이 생기지만 출력은 0이다.

s1 -> s1 : ‘1010’의 첫번째 1을 받은 상태에서 X의 값이 1이 들어와도 첫번째 1이라는 사실은 변하지 않는다. ‘1010’ sequence가 만들어지지 않으므로 출력은 0이다.

s1 -> s2 : ‘1010’에서 앞의 ‘10’이 만들어지며 출력은 0이다.

s2 -> s0 : ‘1010’에서 앞의 ‘10’이 들어온 상태에서 입력 X의 값이 0이 되어버리면 ‘100’이 되어서 ‘1010’의 sequence가 될 가능성이 없어지므로 초기 상태로 돌아가고 출력은 0이다. 또한 ‘1010’에서 X=’0’의 입력을 받으면 ‘1010’이 될 가능성이 없어지고 초기 상태로 돌아간다.

s2 -> s3 : ‘10’에서 X=’1’의 입력을 받으면 ‘1010’에서 ‘101’이 만들어지며 출력은 0이다. s3에서 다시 X=’0’의 입력을 받으면 s2로 돌아오며 출력을 1로 만든다. 이는 ‘1010’이 처음 detect 됐을 때 그 입력값들을 무효화 하는 것이 아니라 ‘101010’이어도 1이 되게 해야 했기 때문이다.

s3 -> s1 : ‘101’에서 X=’1’의 입력을 받으면 ‘1011’이 되어 ‘1010’이 될 수 없고 다시 ‘1010’에서 첫번째 1이 될 수 있는 상태, 즉 s1로 돌아간다.

s3 -> s2 : ‘101’에서 X=’0’의 입력을 받으면 ‘1010’의 sequence가 만들어지고 출력값은 1이 된다. 이때 무어 머신과 달리 s2에서 출력이 달라도 된다.

(2) 설계한 VHDL 코드와 설명

\*sequence detector code

텍스트이(가) 표시된 사진

자동 생성된 설명

-moore machine과 같이 입력은 클락, 리셋, X이고 출력은 Z이다.

-state를 직관적으로 파악할 수 있도록 type으로 s0, s1, s2, s3을 정의하였다.

-state와 next state를 s0, s1, s2, s3을 사용하여 표현할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

-state register는 sequential logic이기 때문에 CLK를 필요로 한다.

-RESET=’0’일 때는 항상 초기 상태로 돌아가게 하고 RESET=’1’이면서 CLK의 상승 에지일 때 state는 next state로 변화한다.

- RESET과 CLK에 따라 state의 변화가 나타나기 때문에 sensitivity list에 포함시켰다.

텍스트이(가) 표시된 사진

자동 생성된 설명

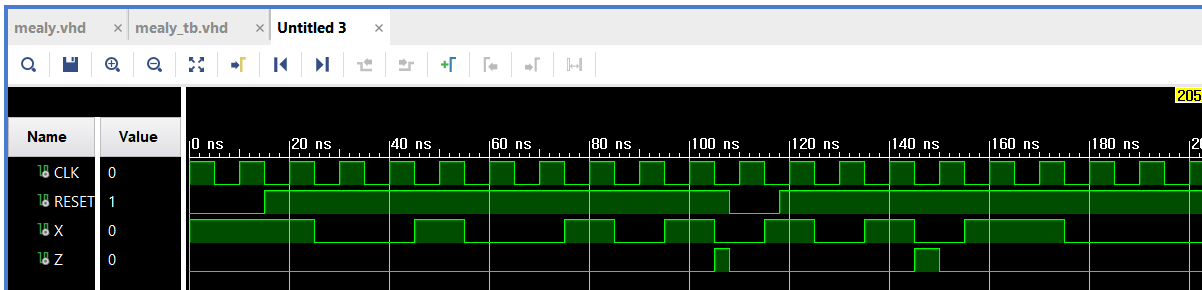
텍스트이(가) 표시된 사진

자동 생성된 설명

- State diagram, state table을 활용하여 코드를 작성하였다. 특정 state에서 어떤 입력을 받았는지에 따라 next state와 Z가 어떻게 달라지는 지를 표현하였다.

-밀리 머신의 출력은 입력과 state에 의해 변화하기 때문에 X, state를 sensitivity list에 포함하였다.

(2) waveform과 설명 및 분석



-입력 : RESET, X / 출력 : Z

-RESET/X/Z(아래에서 1/0/1 이런 식으로 작성하는 것 모두 RESET=’1’, X=’0’, Z=’1’이라는 의미이다.)

-밀리 머신에서 state는 다음 상승 에지에서 변화하지만 출력은 입력이 변함과 동시에 변화한다.

-출력 Z가 클락의 상승 에지에서 변하는 것이 아니라 입력 X가 변할 때 바로 변하는 것을 확인할 수 있다. 출력이 입력에 의해 변하는 next state에 따라 바뀌는 것이 아니라 입력과 현재 state에 따라서 변화하기 때문이다.

-RESET=’1’일 때(두번째 RESET=’0’ 전까지)의 state를 살펴보면 s0 s1 s2 s0 s1 s2 s0 s1 s2 s3의 순서로 변화한다. 105ns에서 일정 시간 동안 출력이 1인데 이는 100ns에서 state가 s3이 되었고, 1/1의 입력을 받았기 때문이다. 즉 밀리 머신은 출력이 state보다 먼저 바뀔 수 있음을 확인할 수 있다. 이와 유사하게 145ns에서도 1/0의 입력을 받자마자 출력이 1로 바뀌는 것을 확인할 수 있다.

\*CLK 상승 에지에서 state 확인

10ns RESET=’0’, X=’1’

-> 0/1/0이기 때문에 X의 값에 상관없이 s0 상태이다.

20ns RESET=’1’, X=’1’

->RESET이 ‘1’이 되어서 활성화된다. s0에서 1/1의 입력을 받고 s1이 된다.

30ns RESET=’1’, X=’0’

->s1에서 1/0의 입력을 받아 s2가 된다.

40ns RESET=’1’, X=’0’

->s2에서 1/0의 입력을 받아 s0이 된다.

50ns RESET=’1’, X=’1’

->s0에서 1/1의 입력을 받고 s1이 된다.

60ns RESET=’1’, X=’0’

->s1에서 1/0의 입력을 받고 s2가 된다.

70ns RESET=’1’, X=’0’

->s2에서 1/0의 입력을 받고 s0이 된다.

80ns RESET=’1’, X=’1’

->s0에서 1/1의 입력을 받고 s1이 된다.

90ns RESET=’1’, X=’0’

->s1에서 1/0의 입력을 받고 s2가 된다.

100ns RESET=’1’, X=’1’

-> s2에서 1/1의 입력을 받고 s3이 된다.

110ns RESET=’0’, X=’0’

->RESET=’0’이므로 X와 state에 관계없이 s0이 된다.

120ns RESET=’1’, X=’1’

-> s0에서 1/1의 입력을 받고 s1이 된다.

130ns RESET=’1’, X=’0’

->s1에서 1/0의 입력을 받아 s2가 된다.

140 ns RESET=’1’, X=’1’

-> s2에서 1/1의 입력을 받고 s3이 된다.

150 ns RESET=’1’, X=’0’

->s3에서 1/0의 입력을 받고 s2가 된다.

160 ns RESET=’1’, X=’1’

-> s2에서 1/1의 입력을 받고 s3이 된다.

170 ns RESET=’1’, X=’1’

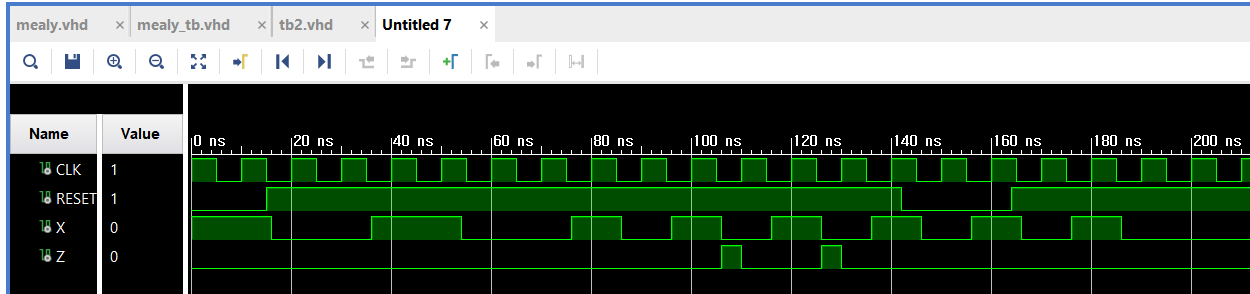
->s3에서 1/1의 입력을 받고 s1이 된다.

180 ns RESET=’1’, X=’0’

-> s1에서 1/0의 입력을 받고 s2가 된다.

-위 결과를 state diagram(혹은 state table)과 비교해보면 SPEC에 맞게 동작한다는 것을 확인할 수 있다.

\*waveform 및 결과 분석(extra)



-입력 RESET/X

-과제 pdf 파일에 있는 입력대로 인가했을 때의 결과를 확인해보고자 하였다.(일부 다름)

-CLK의 상승 에지에서의 state 변화에 따른 결과를 나타내면(20ns부터 10ns 간격으로)

->s0 s0 s1 s1 s2 s0 s1 s2 **s3(100ns)** s2 s3 s2 s3 …

->빨간색으로 표시한 s3에서 1/1의 입력을 받음과 동시에 출력이 1로 변하는 것을 확인할 수 있다.

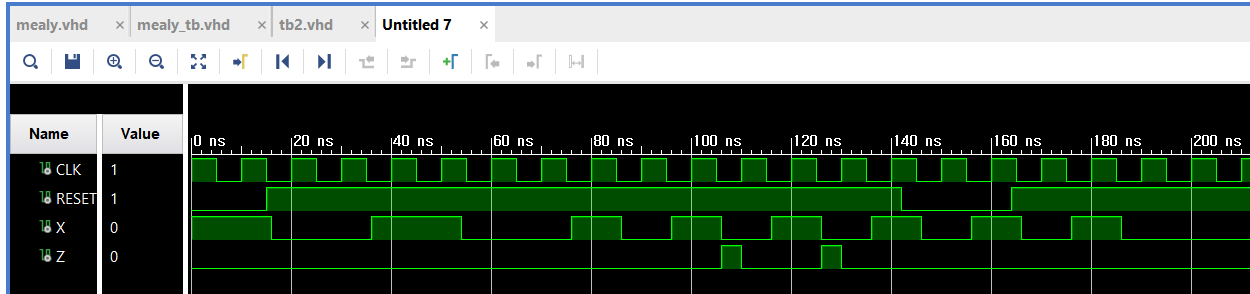
-RESET=’0’이 됐을 때는 초기 상태로 돌아가서 출력이 0으로 바뀐 것도 확인할 수 있다.

3. Mealy / Moore 결과의 차이 및 원인

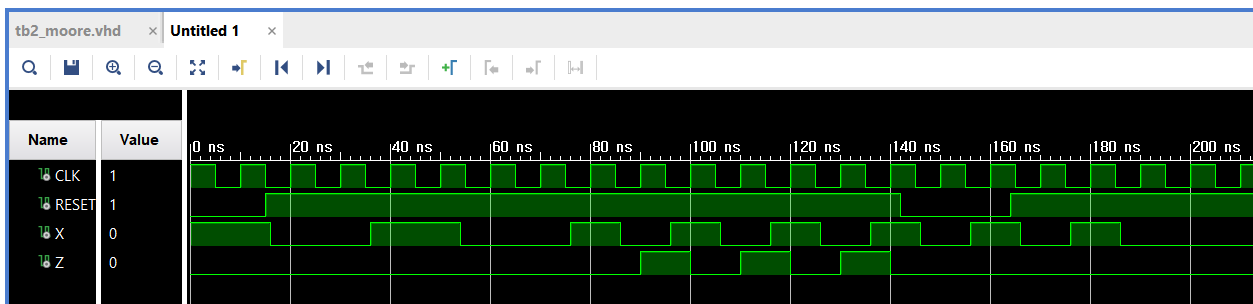
-결과가 어떻게 다른지 확인하기 위해 동일한 입력을 인가하였다.

-waveform만 보더라도 확실한 차이가 있다는 것을 알 수 있다.

\*mealy



\*moore



\*차이점

-moore machine에서의 결과에서 Z가 CLK의 상승에지에서 변하고 그 출력이 다음 상승에지까지 유지되는 반면, mealy machine에서의 waveform에서는 CLK의 상승에지가 아니라 입력의 값이 변함과 동시에 출력 Z가 바뀐다.

Ex) 무어 머신 waveform 90ns 상승에지에서 state가 s4로 변하는 동시에 출력 Z가 1로 변한다.

Ex) 밀리 머신 waveform 100ns에 s3에서 1/1의 입력을 받음과 동시에 출력이 1로 변하는 것을 확인할 수 있다.

\*원인

-mealy machine과 moore machine의 동작 원리 자체가 원인이다. Mealy machine은 state와 입력에 따라 출력이 결정되는 반면 moore machine은 오직 state에 의해서만 출력이 결정된다. 그렇기 때문에 출력이 나타나는 시점에 차이가 생기게 된다. State diagram에서도 이를 알 수 있는데, 무어 머신은 state와 출력을 같이 쓰고, 밀리 머신은 특정 state에서 다른 state로 갈 때 입력과 출력을 함께 쓴다.

-moore machine은 state에 의해서만 출력이 결정되기 때문에 state의 변화가 나타나는 CLK의 상승 에지에서만 state에 따라 출력이 나타나고 그 출력이 다음 CLK의 상승 에지까지 유지된다.

-mealy machine은 입력과 state에 의해 출력이 결정되기 때문에 다음 state로 변하기 전에 state와 입력에 따라 출력이 나타난다. 따라서 CLK의 상승 에지까지 기다릴 필요 없이 입력이 변하면 바로 출력이 변한다.