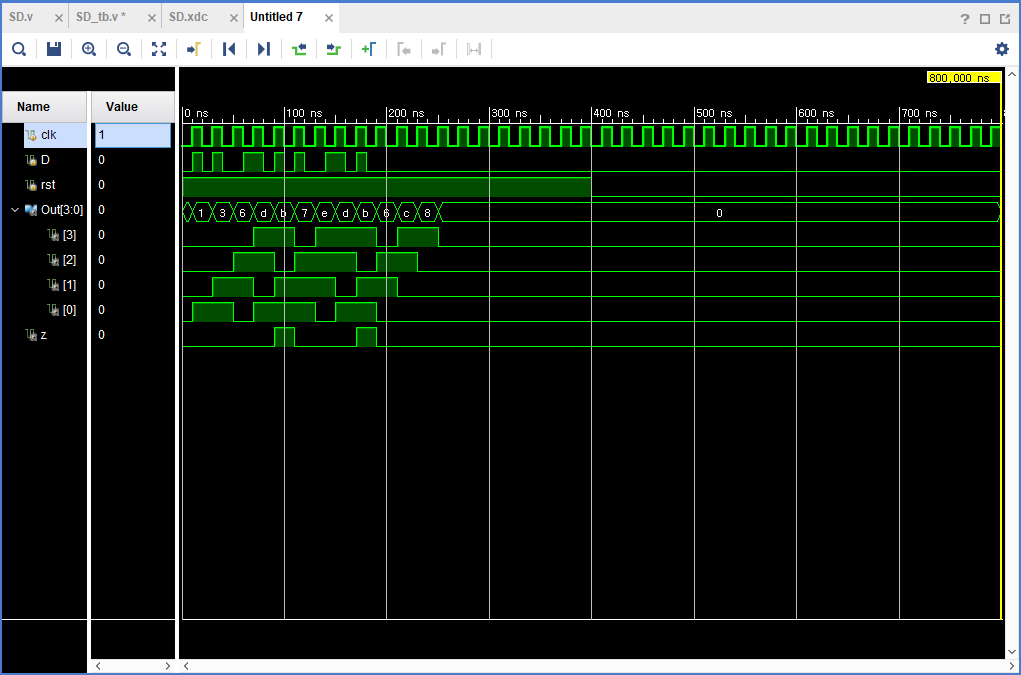
14주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**



\*Sequence Detector 1101 Moore Machine Simulation 결과

위의 결과는 Sequence Detector 1101을 구현하여 Simulation을 돌린 결과이다. 이번에

구현한 Detector는 Moore Machine으로 오직 현재 State에 의존하여 출력을 결정한다.

Out이 11012가 될 때는 Out이 01102일 때 입력 1이 들어오는 상황이기 때문에 현재

State만 신경 쓰는 z는 아직 Out을 01102으로 인식하고 있다. 따라서 11012가 되는

순간에는 z값이 변하지 않고, 다음 Cycle에서 현재 State가 11012에서 다른 값으로 변하는

순간에 z값이 11012을 인식하여 1을 출력한다.

clk는 Clock Signal이며, clk가 1이 될 때마다 모든 Flipflop이 그 신호를 받아 Shift

Resister처럼 저장하고 있는 데이터를 뒤에 있는 Flipflop으로 보낸다. D는 데이터를 뜻하며,

D가 1이 되면 Sequence Detector 의 첫번째 Flipflop의 입력으로 1이 들어간다. rst는

Reset기능을 하며, Active Low로 작동하기 때문에 rst가 0일 때 Reset기능이 활성화되어

Resister내에 있는 모든 데이터를 초기화하며, rst가 1일 때는 Reset기능이 비활성화된다.

z는 Out 데이터 11012를 감지하면 1을 출력으로 내고 그 이외의 경우는 0을 출력하는

Output이다. 위의 Simulation 결과를 보게 되면 clk이 1이 될 때마다 Sequence Detector 의

데이터가 한칸씩 옆으로 Shift하는 것을 볼 수 있고, D가 1일 때는 데이터가 Out[0] 위치에

추가되는 것을 볼 수 있다. 또한 Out 데이터가 11012이 되면 그 다음 Clock Signal에서

Output z가 1이 되는 것을 볼 수 있고, 그 외에는 z값이 계속 0인 것도 볼 수 있다.

마지막으로 rst가 0일 때 초기화가 정상적으로 되는 것을 볼 수 있다. 아래 사진은

Sequence Detector 의 Verilog 코드와 Simulation 코드이다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 , 텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 번호, 디스플레이이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 1101 Moore Machine Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. z와 Out [3:0] 배열은

output으로 선언해주었고, clk, D 그리고 rst는 input으로 선언해주었다. posedge를 이용해

clk가 1이 될 때마다 begin~end 문을 수행하도록 했다. Out의 데이터를 왼쪽으로

Shift하고 Out[0]에 입력된 D가 추가된다. 그리고 만약 rst가 0이면, Out은 0으로

초기화하며, 만약 Out이 11012이면 z는 1로 바꿔준다. 그 외에는 z는 0으로 출력을 낸다.

simulation 코드에서는 clk를 10클럭마다, rst는 400클럭마다 반전시켜주었다.

D의 경우 일정한 클럭마다 바뀌면 11012을 만들 수 없기 때문에 따로 initial begin ~

end문을 이용해서 11012이 될 수 있도록 만들어주었다.

상태표를 만들기 위해 처음의 State와 입력에 따라 바뀌는 그 이후의 State를 정해준다. 처음의 State를 S0라고 한다면, 입력 1이 들어온 xxx12을 S1, 다시 입력 1이 들어온

xx112을 S2,입력 0이 들어온 x1102을 S3, 마지막으로 입력 1이 들어온 11012을 S4이라고

했을 때 상태표는 아래와 같다. 여기서 x는 Don’t care이다.

|  |  |  |  |
| --- | --- | --- | --- |
| 현재 State | 다음 State | | 출력z |
| D = 0 | D = 1 |
| S0 | S0 | S1 | 0 |
| S1 | S0 | S2 | 0 |
| S2 | S3 | S2 | 0 |
| S3 | S0 | S4 | 0 |
| S4 | S0 | S2 | 1 |

\*Sequence Detector 1101 Moore Machine State Table

위의 상태표를 토대로 상태도를 그리면 아래와 같다.

도표, 원이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 1101 Moore Machine State Diagram

**2.**

**-Mealy Machine**

스크린샷이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Mealy Machine Simulation 결과

위의 결과는 Sequence Detector 10101을 구현하여 Simulation을 돌린 결과이다.

이번에 구현한 Detector는 Mealy Machine으로 현재 State와 현재 입력에 의존하여 출력을

결정한다. Out이 101012가 되는 순간은 Out이 010102일 때 입력 1이 들어오는 상황이기

때문에 현재 State와 현재 입력에 신경 쓰는 z는 Out의 010102 현재 입력 1을 인식한다.

따라서 101012가 되는 순간에 Moore Machine과 다르게 z값은 즉각적으로 1을 출력으로

낸다.

clk는 Clock Signal이며, clk가 1이 될 때마다 모든 Flipflop이 그 신호를 받아 Shift

Resister처럼 저장하고 있는 데이터를 뒤에 있는 Flipflop으로 보낸다. D는 데이터를 뜻하며,

D가 1이 되면 Sequence Detector 의 첫번째 Flipflop의 입력으로 1이 들어간다. rst는

Reset기능을 하며, Active High로 작동하기 때문에 rst가 1일 때 Reset기능이 활성화되어

Resister내에 있는 모든 데이터를 초기화하며, rst가 0일 때는 Reset기능이 비활성화된다.

z는 Out 데이터 101012를 감지하면 1을 출력으로 내고 그 이외의 경우는 0을 출력하는

Output이다. 위의 Simulation 결과를 보게 되면 clk이 1이 될 때마다 Sequence Detector 의

데이터가 한칸씩 옆으로 Shift하는 것을 볼 수 있고, D가 1일 때는 데이터가 Out[0] 위치에

추가되는 것을 볼 수 있다. 또한 Out 데이터가 101012이 되면 바로 Output z가 1이 되는

것을 볼 수 있고, 그 외에는 z값이 계속 0인 것도 볼 수 있다. 마지막으로 rst가 1일 때

초기화가 정상적으로 되는 것을 볼 수 있다. 아래 사진은 Sequence Detector 의 Verilog

코드와 Simulation 코드이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Mealy Machine Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. z와 Out [4:0] 배열은

output으로 선언해주었고, clk, D 그리고 rst는 input으로 선언해주었다. posedge를 이용해

clk가 1이 될 때마다 begin~end 문을 수행하도록 했다. Out의 데이터를 왼쪽으로

Shift하고 Out[0]에 입력된 D가 추가된다. 그리고 만약 rst가 1이면, Out은 0으로

초기화하며, 만약 Out이 101012이면 z는 1로 바꿔준다. 그 외에는 z는 0으로 출력을 낸다.

simulation 코드에서는 clk를 10클럭마다, rst는 200클럭마다 반전시켜주었다.

D의 경우 일정한 클럭마다 바뀌면 101012을 만들 수 없기 때문에 따로 initial begin ~

end문을 이용해서 101012이 될 수 있도록 만들어주었다.

상태표를 만들기 위해 현재 State와 현재 입력에 따라 바뀌는 그 이후의 State를 정해준다.

처음의 State를 S0라고 한다면, 입력 1이 들어온 xxxx12을 S1, 입력 0이 들어온

xxx102을 S2,입력 1이 들어온 xx1012을 S3, 마지막으로 입력 0이 들어온 x10102을

S4이라고 했을 때 상태표는 아래와 같다. 여기서 101012 상태를 따로 규정하지 않은 이유는

Mealy Machine이 현재 상태와 현재 입력에 따라서 출력이 바로 변하는 특징을 가지고 있기

때문이다. 따라서 마지막 101012 상태가 필요 없다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 현재 State | 다음 State | | 출력z | |
| D = 0 | D = 1 | D = 0 | D = 1 |
| S0 | S0 | S1 | 0 | 0 |
| S1 | S2 | S1 | 0 | 0 |
| S2 | S0 | S3 | 0 | 0 |
| S3 | S4 | S1 | 0 | 0 |
| S4 | S0 | S3 | 0 | 1 |

\*Sequence Detector 10101 Mealy Machine State Table

위의 상태표를 토대로 상태도를 그리면 아래와 같다.

도표, 원이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Mealy Machine State Diagram

**-Moore Machine**

스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Moore Machine Simulation 결과

위의 결과는 Sequence Detector 10101을 구현하여 Simulation을 돌린 결과이다.

이번에 구현한 Detector는 1번 1101 Detector와 마찬가지로 Moore Machine으로 구현했기

때문에 오직 현재 State만을 의존하여 출력을 결정한다. Out이 101012가 되는 순간은

Out이 010102일 때 입력 1이 들어오는 상황이기 때문에 현재 State만 신경 쓰는 z는

Out의 010102만을 인식한다. 따라서 101012가 되는 순간에는 z값이 반영하지 못하고, 한

Cycle이 지난 뒤에 z는 101012을 인식하고 1을 출력으로 낸다.

각종 input과 output은 위에서 설명한 Mealy Machine과 같은 기능을 하며 한가지

차이점이 있다면, Mealy Machine과 다르게 Out이 101012이 되고 난 뒤 그 다음 Clock Signal에 z값이 1이 되는 것을 볼 수 있다.

아래 사진은 Sequence Detector 의 Verilog 코드와 Simulation 코드이다.

텍스트, 스크린샷, 폰트, 영수증이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Mealy Machine Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. z와 Out [4:0] 배열은

output으로 선언해주었고, clk, D 그리고 rst는 input으로 선언해주었다. posedge를 이용해

clk가 1이 될 때마다 begin~end 문을 수행하도록 했다. Out의 데이터를 왼쪽으로

Shift하고 Out[0]에 입력된 D가 추가된다. 그리고 만약 rst가 1이면, Out은 0으로

초기화하며, 만약 Out이 101012이면 z는 1로 바꿔준다. 그 외에는 z는 0으로 출력을 낸다.

simulation 코드에서는 clk를 10클럭마다, rst는 200클럭마다 반전시켜주었다.

D의 경우 일정한 클럭마다 바뀌면 101012을 만들 수 없기 때문에 따로 initial begin ~

end문을 이용해서 101012이 될 수 있도록 만들어주었다.

상태표를 만들기 위해 현재 State에 따라 바뀌는 그 이후의 State을 정해준다.

Moore Machine과 다르게 오직 현재 State에 의해서만 다음 State가 정해지고, 그 결과가

한 Cycle뒤에 반영되기 때문에 위에서 설명한 Moore Machine의 State에서 마지막 101012

State를 추가해서 총 6개의 State가 사용된다. 처음의 State를 S0라고 한다면, 입력 1이

들어온 xxxx12을 S1, 입력 0이 들어온 xxx102을 S2,입력 1이 들어온 xx1012을 S3,

입력 0이 들어온 x10102을 S4, 마지막으로 입력 1이 들어온 101012을 S5 이라고 했을 때 상태표는 아래와 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| 현재 State | 다음 State | | 출력z |
| D = 0 | D = 1 |
| S0 | S0 | S1 | 0 |
| S1 | S2 | S1 | 0 |
| S2 | S0 | S3 | 0 |
| S3 | S4 | S1 | 0 |
| S4 | S0 | S5 | 0 |
| S5 | S4 | S1 | 1 |

\*Sequence Detector 10101 Moore Machine State Table

위의 상태표를 토대로 상태도를 그리면 아래와 같다.

도표, 원, 라인이(가) 표시된 사진

자동 생성된 설명

\*Sequence Detector 10101 Moore Machine State Diagram