14주차 예비보고서

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

**1.**

FSM은 Finite State Machine의 약자로 유한개의 각 State가 있고, State를 전이시키는

조건이 있을 때, 그것을 표현하는 방법이다. 다시 말해 유한개의 State를 가지고 주어지는

입력에 따라 출력이 정해지고 또는 다른 상태로 전이되는 장치를 나타낸 모델이다. 주로

컴퓨터 프로그램 및 전자 논리 회로 등을 설계하는데에 쓰이는 수학적 모델이다.

FSM의 구성요소는 State, Transition, Transition Condition(Event), Entry Action이 있다.

State는 말 그대로 Object의 상태를 나타내며, Transition은 현재 State에서 다른 State로의

전이 또는 사건을 의미한다. Transition Condition은 Transition이 발생하기 위한 조건을

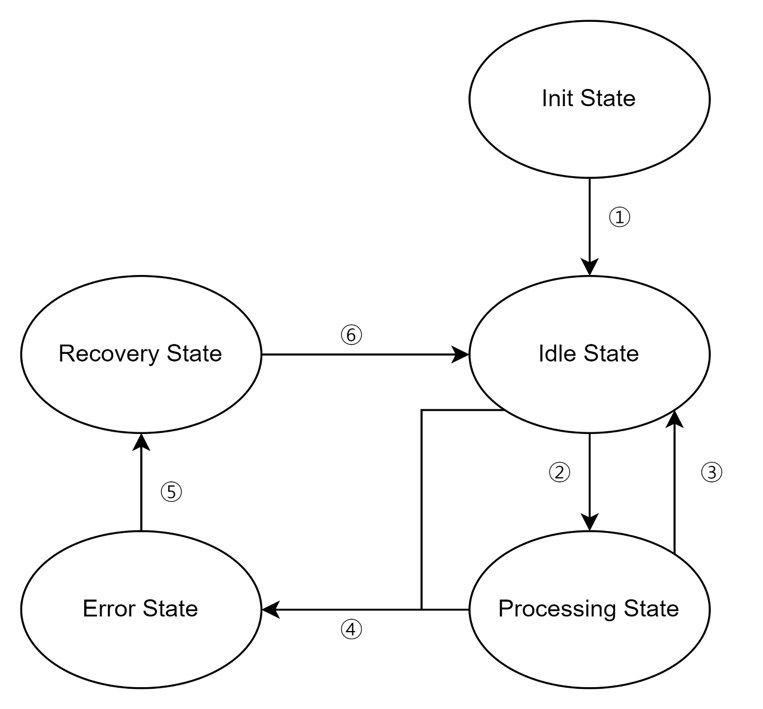
의미하고, Entry Action은 새로운 State로 Transition했을 때에 행해지는 어떠한 동작을

의미한다. FSM을 쓰는 이유는 바로 ’안전성’ 때문이다. FSM을 사용하여 설계를 할 경우

통제 가능한 변인을 가지고 원하는 Output을 낼 수 있어, FSM이 아닌 절차지향

프로그램으로 설계를 하면서 변수오염 또는 잘못된 플로우를 탈 수 있는 가능성을

현저히 낮춘다. 예를 들어 아래와 같은 프로그램을 설계할 때,



위의 그림을 보면 Init State → Idle State → Processing State → Error State → Recovery State 순으로 나뉘는 FSM임을 알 수 있다.

---------------------------------------------------------------------------------

A. Init State에서 장치의 각종 Init을 처리한다.

**① Init Process가 모두 종료된다면,**

B. Idle State에서 대기한다.

**② 센서나 외부 입력이 감지되면,**

C. Processing State 에서 입력에 대한 처리를 한다.

**③ 입력에 대한 처리가 모두 끝나면 이를 디스플레이에 표시하거나** **서버에 전송하고,**

D. Idle State로 돌아가서 대기한다.

**④ 만약 Idle State 또는 Processing State에서 오류가 난다면**

E. Error State 로 가서 에러를 분석한다.

**⑤ Error가 복구 가능한 수준이라면**

F. Recovery State로 가서 Error를 복구한다.

**⑥ 복구가 모두 끝나면,**

G. Idle State로 돌아가서 대기한다.

---------------------------------------------------------------------------------

위의 A부터 G까지의 과정을 보면, A~G의 문장은 State 및 Entry Action이며, 볼드체와 밑줄이 그어져 있는 문장은 Transition Condition(Event)이다.

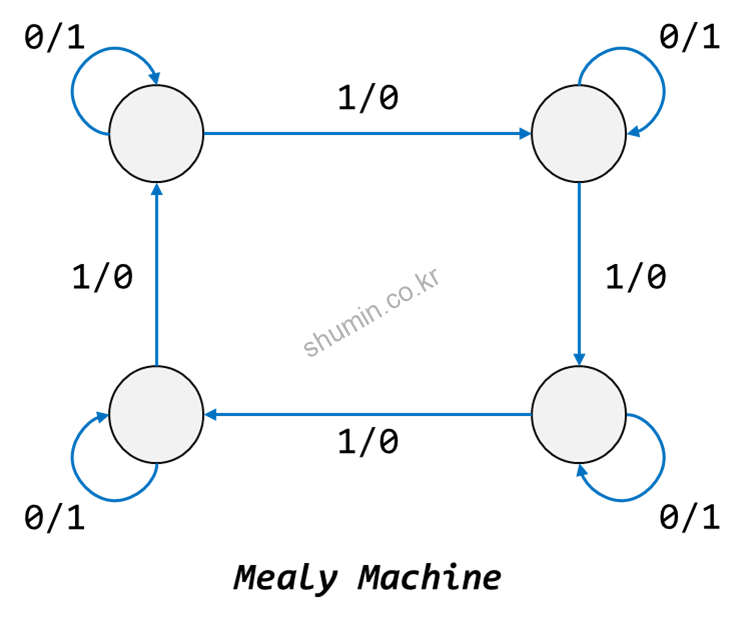
위의 과정을 보면 FSM의 장점이 보이는데, 먼저 반드시 설계한대로만 Flow가 흐르며,

정해진 조건에 의해서만 State가 Transition된다. 마지막으로 전체적인 과정이 유기적이면 명확하기 때문에 설계자 외의 사람들이 봐도 코드가 명확하고 한 눈에 들어온다.

**2.**

FSM에는 2 종류가 있는데, Mealy Machine과 Moore Machine이 있다.

Mealy Machine은 현재 State와 입력에 의존하여 Output을 출력한다.



\*Mealy Machine 예시

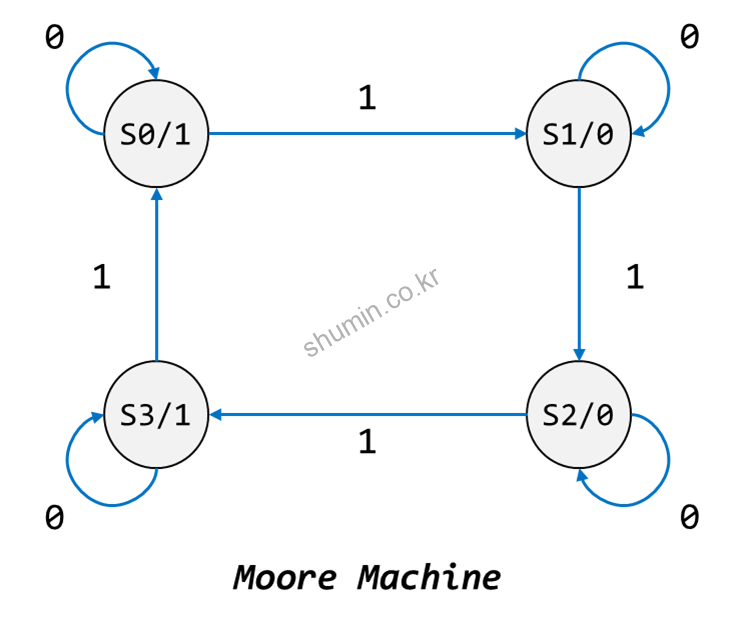
위의 예시를 보면 현재 State에서 입력이 들어오면 출력이 바로 반영되는 것을 볼 수 있다.

따라서 Mealy Machine은 입력에 대해 반응이 빠르고, State수가 Moore Machine에 비해

적게 필요하다. 하지만 입력이 바로 출력에 반영되기 때문에 2개 이상의 장치를 연동해서 사용할 경우 비동기화 문제가 생길 수 있다. 또한 구조가 직관적이지 못해 설계에 어려움이 있다.

**3.**

Moore Machine은 Mealy Machine과 달리 오직 현재 State에 의해서만 Output이 결정된다.



\*Moore Machine 예시

위의 예시를 보면, 입력이 들어오고 난 뒤 다음 State에서 출력이 변하는 것을 볼 수 있다.

따라서 Mealy Machine과 다르게 입력이 출력에 바로 반영되지 않고 한 cycle뒤에 반영된다.

입력이 들어오는 순간마다 출력에 반영되지 않고, 일정한 Clock Signal에 맞춰 출력이

바뀌기 때문에 안정성이 좋고, 그런 안정성으로 인해서 Mealy Machine에 비해 설계도

비교적 간단하다는 장점도 있다. 하지만 Mealy Machine에 비해 입력에 대한 반응이 느리고,

필요한 State수도 더 많다.

**4.**

Sequence Detector는 입력을 받을 때, 특정 순서의 입력을 받으면 출력 1을 반환하는

장치이다. 예를 들어 원하는 Sequence가 11011일 때, 마지막으로 받은 입력 5개가

11011이면 출력 1을 반환한다. Sequence Detector의 종류에는 Overlapping과

Non-Overlapping이 있다. Non-Overlapping은 말 그대로 겹침을 허용하지 않는다는 뜻으로

11011을 입력으로 받아 원하는 Sequence를 충족하면, 다음 입력은 Sequence의 첫번째의

1부터 다시 찾는다. 하지만 Overlapping은 반대로 겹침을 허용한다는 뜻으로 11011을

입력으로 받으면, 출력 1을 반환하고, 다음 Sequence의 11을 이전 입력에서 이미 받았다고

여겨서 11을 겹쳐서 Sequence의 세번째부터 시작해 011 입력을 기다린다. 아래의 예시를 보면 이해가 더 쉬울 것이다.

|  |  |
| --- | --- |
| 입력 X | 1 1 0 1 1 0 1 1 0 1 1 |
| Non-Overlapping 출력 | 0 0 0 0 1 0 0 0 0 0 1 |
| 입력 X | 1 1 0 1 1 0 1 1 0 1 1 |
| Overlapping 출력 | 0 0 0 0 1 0 0 1 0 0 1 |

\*11011 Sequence Detector

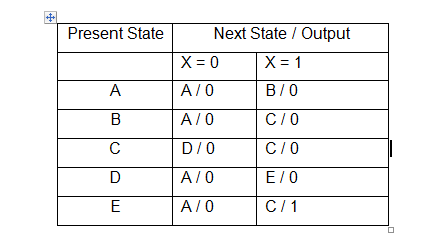
11011 Sequence Detector 설계 순서는 다음과 같다.

1. 회로의 State Diagram 그리기

도표, 원, 스케치, 그림이(가) 표시된 사진

자동 생성된 설명

2. State Diagram 토대로 State Table 작성



3. 회로에 필요한 Flipflop 종류와 수 알아보기

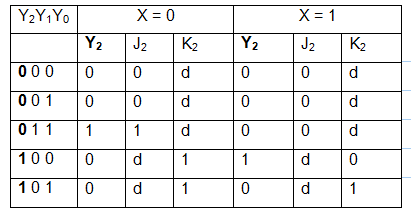
4. Excitation Table 작성

텍스트, 번호, 달력, 낱말맞추기 퍼즐이(가) 표시된 사진

자동 생성된 설명

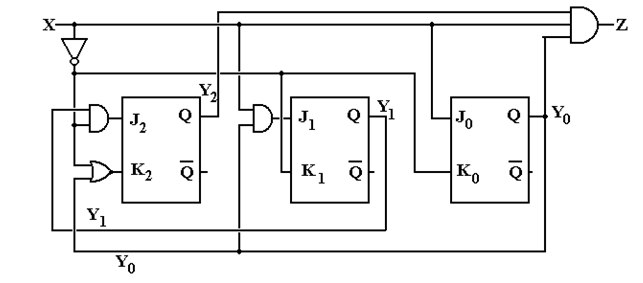
텍스트, 번호, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명



5. Excitation Table을 토대로 각 Flipflop input과 output Z’s Equation 구하기

6. 도출해낸 방정식으로 회로도 그리기



**5.**

Moore Machine과 Mealy Machine의 장점을 합친 것이 Synchronous Mealy Machine이다.

Flipflop을 사용하여 Clock에 맞춰 State를 동기화한다. 즉, Mealy Machine처럼 Output을

빠르게 계산하면서 Flipflop을 이용해 다음 State와 동기화 함으로써, Mealy Machine에서

발생할 수 있는 Glitch에 대한 문제를 피할 수 있다. 마찬가지로 Synchronous Moore Machine도 만들 수 있다. 현재 State로부터 Output계산을 하는데에 발생하는 Delay을

없애기 위해 현재 State값을 Flipflop에 넣기 전 Output 계산을 하고, Output과 현재 state

계산값들을 Flipflop에 넣어 동기화 할 수 있다.