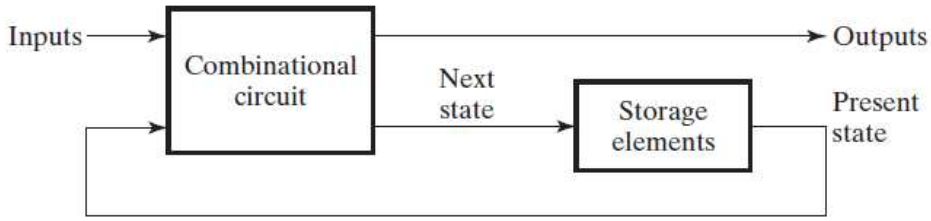
순차회로 기본

1. 순차 회로 개요

- **순차회로(Sequential Circuit)**
  - 현재의 입력과 이전의 출력상태에 의해 현재 출력이 결정되는 회로
    - 조합 회로와 데이터 저장 회로가 결합된 회로
  - 데이터 저장회로
    - 저장된 값은 특정 조건이 만족될 때까지 계속 유지.
  - 순차회로에서의 조합회로의 역할
    - 출력신호 생성
    - 저장회로의 입력신호를 생성



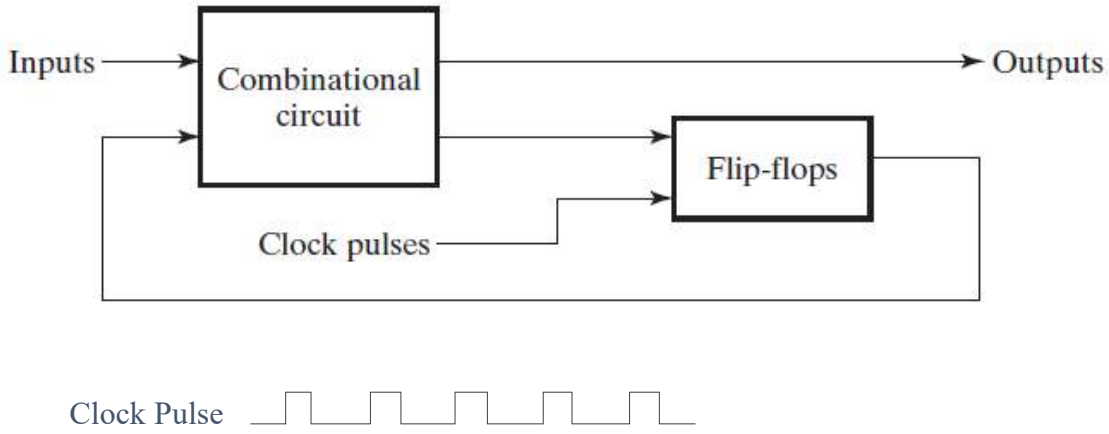
1.1 순차 회로 유형

- **동기식 순차회로**
  - 특정 신호에 동기되어 비연속적인 시간에만 동작하는 순차회로
  - 클럭과 같이 주기적으로 발생되기 신호에 반응.

- **비동기식 순차회로**
  - 입력신호 변화에 따라 연속적으로 동작하는 순차회로
  - 이벤트 발생과 같이 비주기적으로 발생하는 신호에 반응.

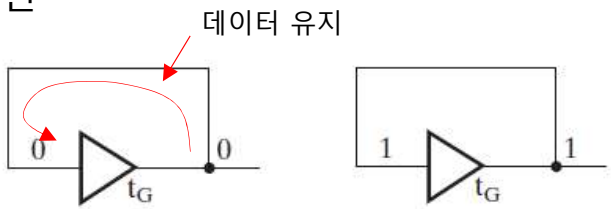
1.2 동기식 순차 회로

- **동기식 순차 회로**
  - 입력의 변화에 따른 출력신호의 변환 및 내부 상태 변화가 클럭에 동기되어 동작하는 회로

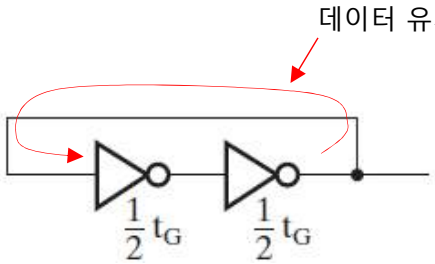


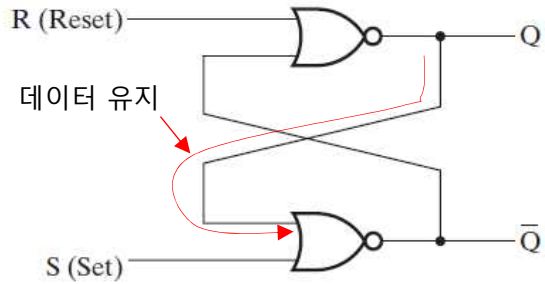
1.3 데이터 저장회로 구현

- **데이터 저장회로 구현방법**
  - 버퍼를 사용한 구현



  - 인버터를 사용한 구현

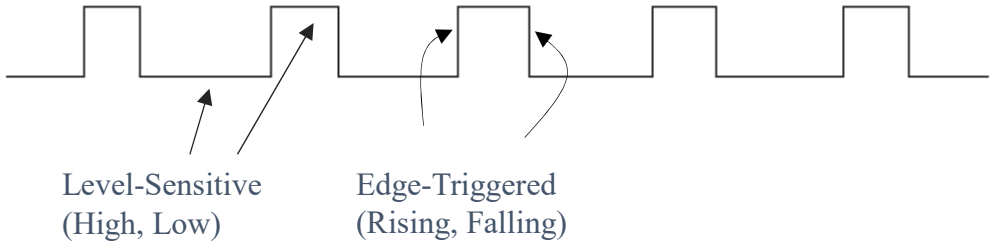




  - 저장된 데이터 값 변경을 위해 별도 신호 선 필요

2. 래치와 플립플럽

- **순차회로의 데이터 저장회로 : 래치 또는 플립플럽**
- **래치(Latch)**
  - 클럭의 레벨에 동기화 : Level-Sensitive
    - 클럭의 레벨이 High 또는 Low 로 유지되는 동안, 입력 신호의 변화가 그대로 출력에 반영 : Transparent
- **플립플럽(Flip-Flop)**
  - 클럭의 엣지에 동기화 : Edge-Triggered

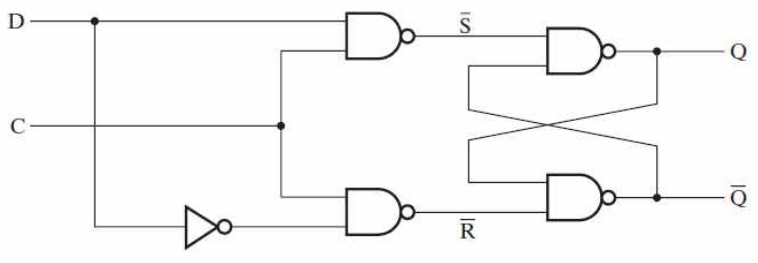


2.1 D-Latch

- **입력 값이 그대로 출력 값에 반영되는 회로**
  - 데이터 저장능력 보유

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

Function Table

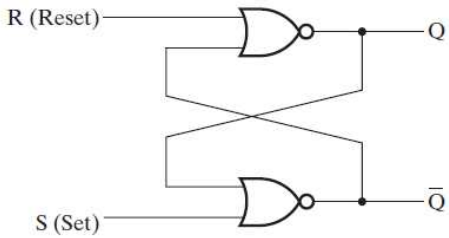


Logic Diagram

2.2 SR-Latch

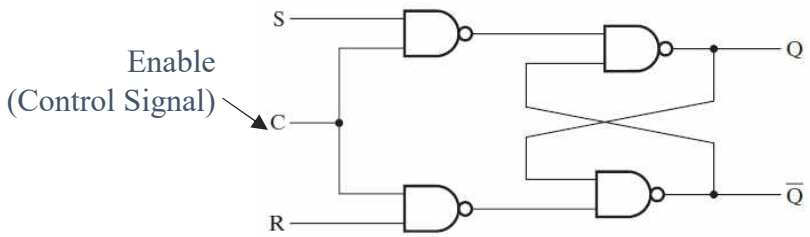
- **출력을 Set 하거나 Reset 하는 회로**
  - NAND 또는 NOR 게이트로 구현

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined



C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

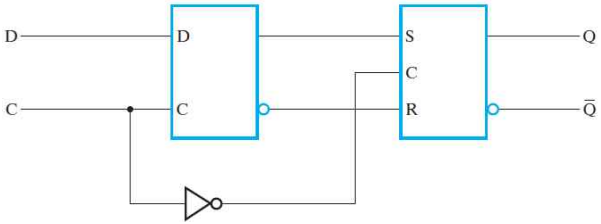
Function Table



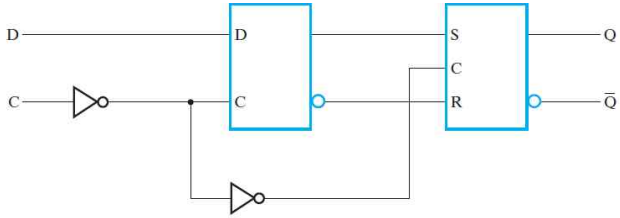
Logic Diagram

2.3 D-Flip Flop

- **입력 값이 그대로 출력 값에 반영되는 플립플럽**
  - 클럭의 엣지시점에서 입력이 출력에 반영
  - 2개의 래치(D, SR)로 구현



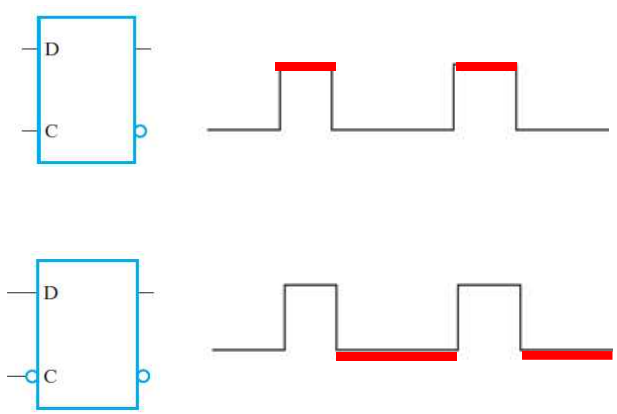
Negative-Edge Triggered D Flip Flop



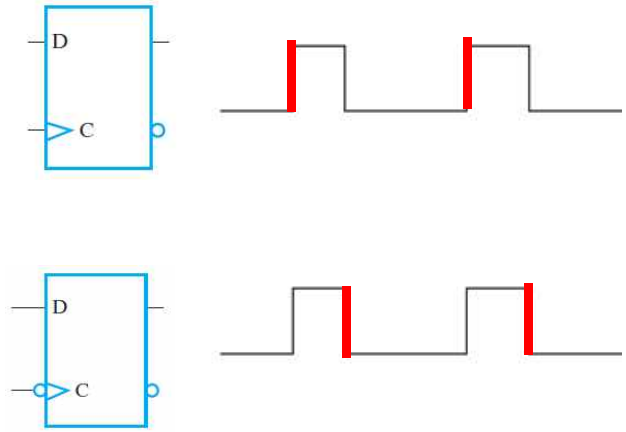
Positive-Edge Triggered D Flip Flop

2.4 표준 그래픽 심볼

- 래치와 플립플럽에 대한 그래픽 심볼



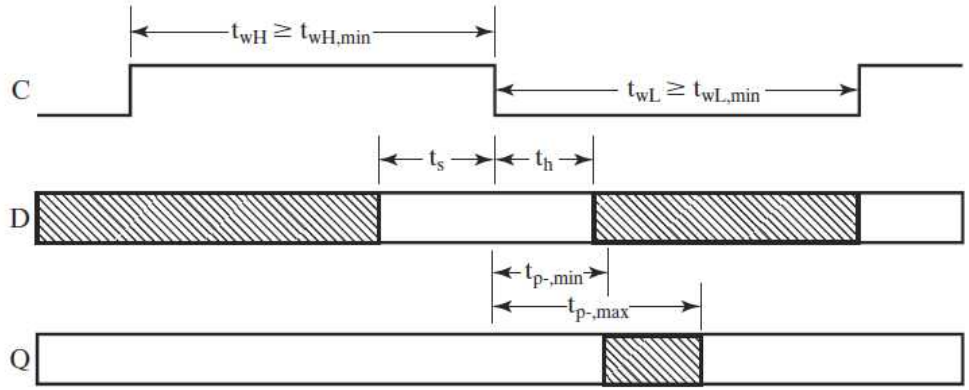
D-Latch



D-Flip Flop

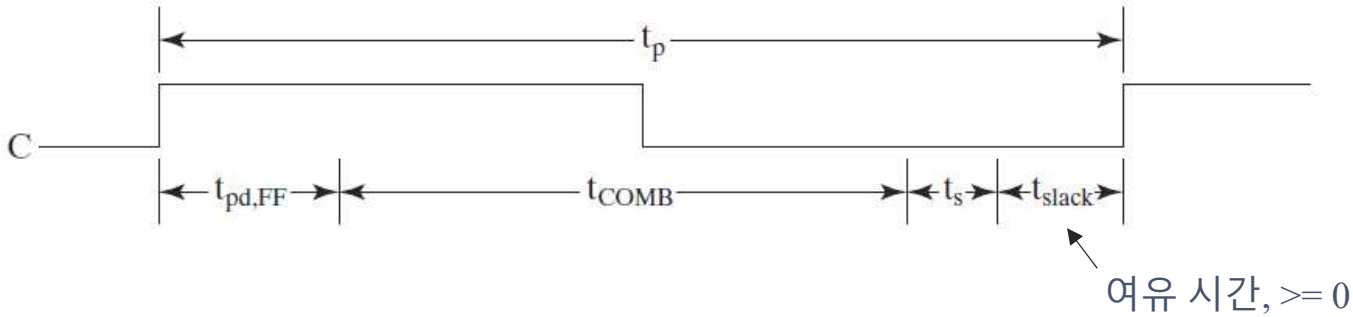
2.5 플립 플롭 타이밍

- **플립플럽의 정상적인 동작을 위해 입력신호와 클럭이 지켜야 할 시간제약 요소**
  - Setup Time (ts) : 클럭 트리거(Trigger)가 발생하기 전에 입력신호가 안정적으로 유지되어야 하는 최소 시간.
  - Hold Time (th) : 클럭 트리거 발생후, 입력시간이 안정적으로 유지되어야 하는 최소 시간.
  - Min. Clock Pulse Width (tw) : 입력신호를 정상적으로 인식하기 위해 필요한 최소 시간.
  - Propagation Delay (tp): 클럭 트리거 발생후, 안정적인 출력이 생성되는 때까지 소요되는 시간.



2.6 순차 회로 타이밍

- **순차회로의 동작속도 결정**
  - 플립플럽 지연시간 + 조합회로 전파지연 시간
    - 플립플럽 지연시간 : Setup, Hold, Propagation Delay
    - 조합회로 지연시간 : 조합회로의 최장 데이터 경로의 Propagation Delay



$$t_p = t_{slack} + (t_{pd,FF} + t_{pd,COMB} + t_s)$$

$$t_p \geq \max(t_{pd,FF} + t_{COMB} + t_s) = t_{p,min}$$

순차회로 설계

1. 순차회로 동작

- **순차회로의 상태**
  - 데이터 저장회로에 저장된 값에 따라 순차회로의 상태(State)를 정의할 수 있음.

- **순차회로의 동작**
  - 순차회로에 대한 입력, 출력, 상태 정보로 표현
    - 상태표(State Table)과 상태 천이도(State Transition Diagram)를 사용

1.1 플립플럽 입력 방정식

- **입력 방정식**
  - 플립플럽의 입력 신호를 발생시키는 조합회로를 표현한 부울방정식
  - 플립플럽의 다음상태를 결정
  - 사용하는 플립플럽의 유형에 따라 입력 방정식이 결정
    - 플립플럽의 입력 신호의 수에 따라 입력 방정식의 수가 결정
      - D-플립플럽 : 1개의 입력 방정식
      - SR-플립플럽 : 2개의 입력 방정식

1.2 상태표

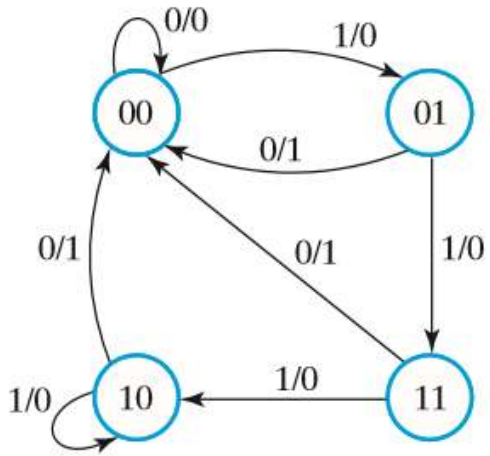
- **상태표**
  - 순차회로의 입력, 출력, 상태 정보를 기술한 표
    - 상태정보 : 현재 상태(Present State), 다음 상태 (Next State)
  - 출력과 다음 상태는 입력과 현재 상태로 부터 결정

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

1.3 상태 천이도

- **상태천이도**
  - 상태표를 그래픽으로 표현한 그림
    - 상태 : 원(Circle)로 표현
    - 상태천이 : 화살(Directed Line)로 표현
  - 순차회로의 상태변화를 그래픽으로 표현

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



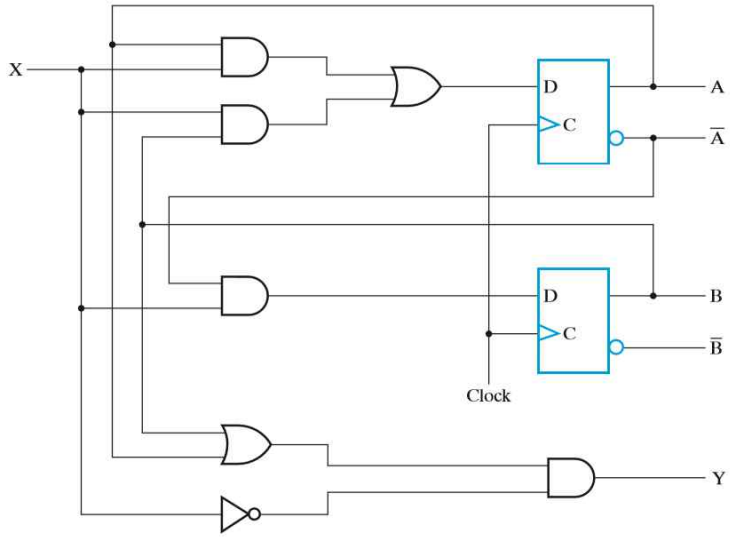
1.4 순차회로 구현 (예)

- **상태표로부터 플립플럽의 입력 부울함수와 출력 부울함수를 구하여 로직회로를 구현.**
  - 사용하는 플립플럽의 유형에 따라 입력 부울함수가 다름.

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

$$D_A = AX + BX$$
$$D_B = \overline{A}X$$

$$Y = (A + B)\overline{X}$$



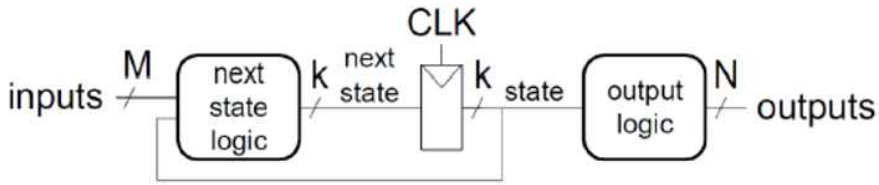
2. FSM

- **FSM(Finite State Machine)**
  - 상태의 수가 유한한 추상적 기계 모델
    - 상태천이도로 동작을 표현

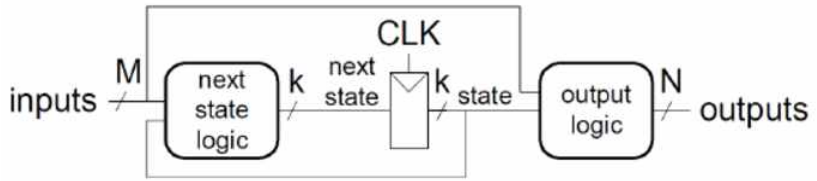
- **FSM 유형**
  - Moore Machine
  - Mealy Machine

2.1 FSM 유형

- **FSM 유형**
  - Moore Machine : 출력이 현재 상태에 의해서만 결정되는 FSM

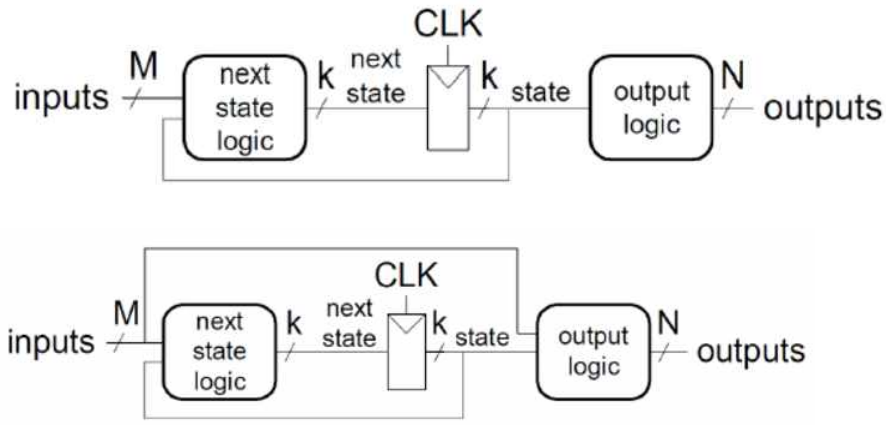


  - Mealy Machine : 출력이 현재 상태와 입력에 의해 결정되는 FSM



2.2 FSM 비교

- **Moore vs. Mealy Machine 비교**
  - Mealy Machine의 상태가 적은 경향이 있다.
  - Moore Machine이 더 안정적으로 동작한다.
    - Moore Machine에서는 출력이 항상 클럭에 동기되지만, Mealy Machine에서는 입력의 변화가 곧바로 출력에 반영된다.
  - Mealy Machine이 입력에 더 빠르게 반응한다.



3. 순차회로 설계절차

- **순차회로의 동작을 사양서(Specification)으로 정의**
- **상태표 또는 상태천이도를 사용하여 동작을 표현**
  - 상태표의 상태에 바이너리 코드를 할당
  - 플립플럽의 유형을 선택
- **상태표에서 플립플럽의 다음 상태 값을 결정하는 플립 플럽의 입력회로를 설계**
  - 조합회로 설계방식을 적용하여 입력회로를 설계
- **상태표에서 출력신호를 결정하는 출력회로를 설계**
- **플립플럽의 입력회로와 출력회로를 최적화**
- **플립플럽과 로직게이트를 사용하여 순차회로 완성**
- **동작을 검증 및 동작 주파수 결정**

3.1 상태 코드 할당

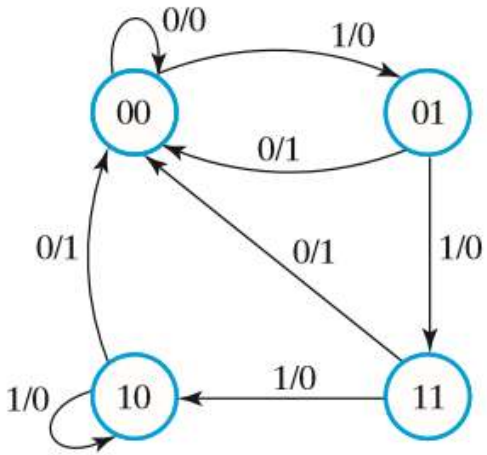
- **상태 코드 할당**
  - 상태 천이도에 정의된 개별 상태들에 이진 코드를 할당하는 것.
  - 할당된 이진 코드를 사용하여 상태 식별

- **상태 코드 할당시 고려사항**
  - 상태 천이도를 구현하는데 필요한 하드웨어 비용과 소비전력 등을 고려
  - 하드웨어 비용
    - 필요한 플립플럽의 수 + 입력신호 생성에 필요한 조합회로의 복잡도
  - 소비전력
    - 상태 천이(상태 코드 값 변화)에 따라 소비전력 차이
    - 상태 코드의 비트 변화에 비례하여 소비전력 증가

상태 코드 변화	00 → 01	11 → 00
비트 변화	1-비트	2-비트
소비 전력	1 배	2 배

3.2 대표적 상태 코드 할당 방법

- **Counting order 할당**
  - 각 상태에 카운팅 순서로 비트
    - 예) 00 → 01 → 10 → 11
  - 가장 간단한 할당 방법



3.2 상태 코드 할당 방법

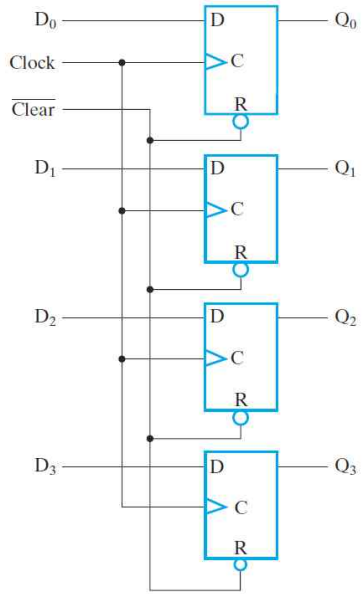
- **Gray Code 할당**
  - 각 상태 전이 순서에 따라 상태 코드의 비트 변동을 최소화
    - 예) 00 → 01 → 11 → 10
  - 소비전력 최소화
  - 상태 전이가 랜덤인 경우, Counting order와 유사

- **One-Hot Code 할당**
  - 각 상태당 하나의 플립플롭을 할당.
  - 상태 수가 n 이라면 n-bit 상태 코드 생성.
    - 예) 0001 → 0010 → 0100 → 1000
  - 상태 디코딩 회로 간단

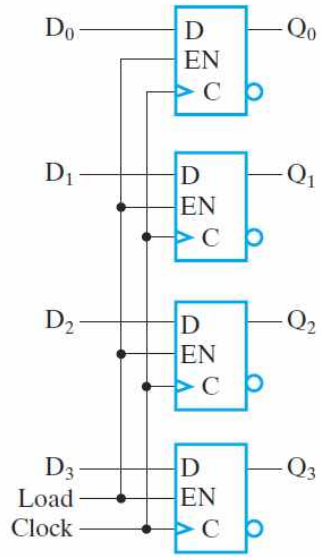
레지스터와 레지스터 전송수준

1. 레지스터 개요

- **레지스터(Register)**
  - 플립 플럽의 집합
    - N-bit 레지스터 : N-bit 데이터를 저장하기 위해 N 개의 플립플럽으로 구성.



4-bit Register with nClear



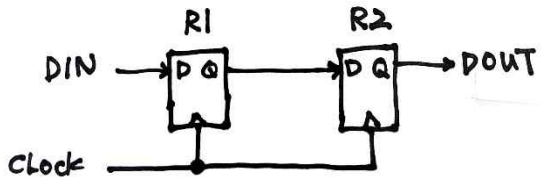
4-bit Register with LOAD

2. 레지스터 전송수준

- **레지스터 전송수준 동작(Register Transfer Level Operation)**
  - 레지스터 수준에서 데이터 흐름과 연산동작을 표현.
  - 데이터 흐름과 연산과정을 쉽게 파악

- **디지털 시스템에서의 레지스터 전송 동작 표현 요소**
  - 사용되는 레지스터들의 집합
  - 레지스터에 저장된 데이터의 전송 및 데이터 연산 작업
  - 작업을 제어하는 제어신호

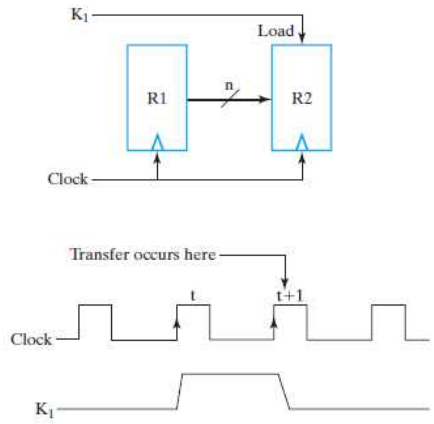
RTL Block Diagram



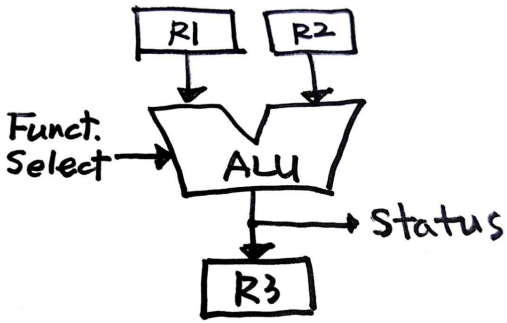
RTL Logic Diagram

2.1 Micro-Operation

- **Micro-operation(MOP)**
  - 레지스터 전송수준의 단위 동작
- **MOP 유형**
  - 데이터 전송
  - 산술/논리 연산



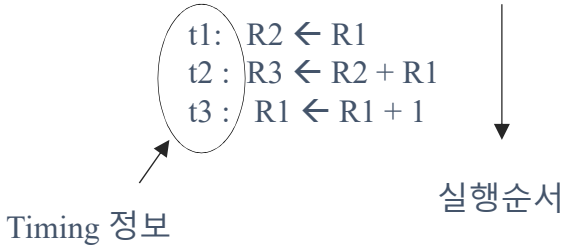
* 데이터 전송 : R2 ← R1



* 산술 연산 : R3 ← R1 + R2

2.2 MOP 실행 타이밍

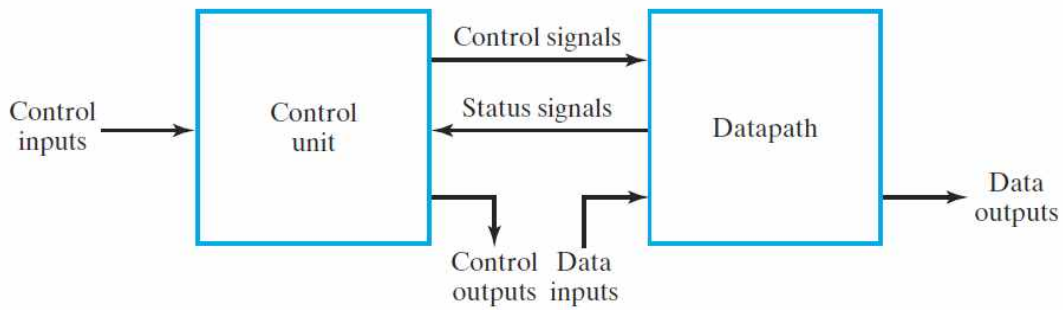
- **MOP이 실행되는 시점에 관한 정보**
  - 실행 순서에 관한 정보
  - 클럭단위로 정의
  - CPU 의 제어 유니트의 시퀀서(Sequencer)를 사용하여 타이밍 신호 발생



2.3 RTL 관점에서의 디지털 시스템

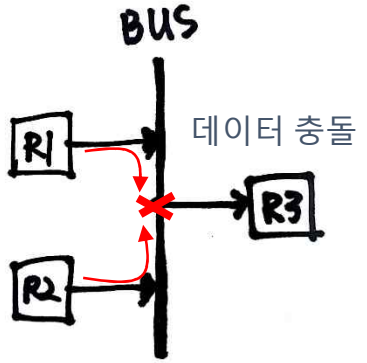
- **데이터 경로(Datapath)**
  - 데이터 처리 동작이 이루어지는 부분
  - 레지스터와 레지스터에 저장된 데이터에 대한 연산 수행

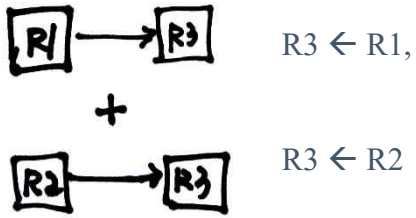
- **제어 유니트(Control Unit)**
  - 데이터 처리 동작을 제어하는 제어 신호를 발생하는 부분



3. 레지스터의 상호 연결

- **여러 개의 레지스터를 상호 연결하는 방법**
  - 멀티플렉서 기반(Multiplexer-based) 연결
    - 멀티플렉서 : N-to-1 연결 회로
    - 여러 데이터의 동시전송 가능
  - 버스 기반(Bus-based) 연결
    - 버스 : 배선의 집합 (Group of Wires)
    - 한번에 전송할 수 있는 데이터 = 버스 수

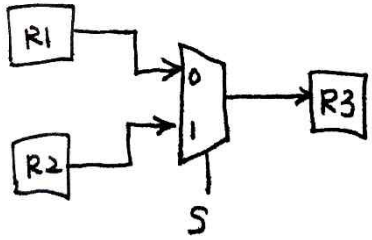




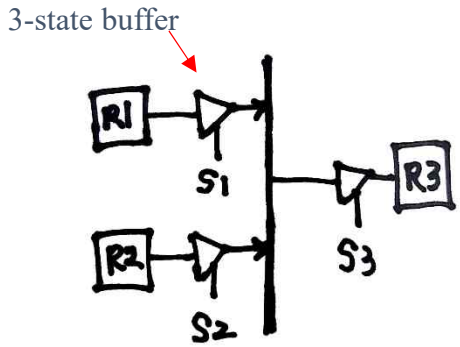
R3 ← R1,

R3 ← R2

예) R1, R2의 출력이 R3에 입력되는 경우



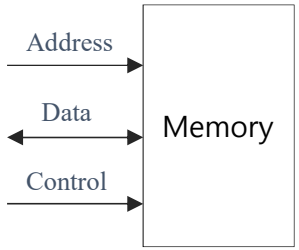
멀티플렉서 기반 연결



버스 기반 연결

4. 메모리 인터페이스

- **메모리에 데이터 읽기/쓰기**
  - 어드레스 버스 : 데이터 저장위치 지정
  - 데이터 버스: 데이터 입출력 경로
  - 제어 신호
    - 메모리 동작에 필요한 제어신호
      - 메모리 유형에 따라 제어신호 구성이 다름.
      - CE(Chip Enable), R/nW(Read/Write), OE(Output Enable), …



- **메모리 읽기 동작의 RTL 표현**
  - R1에 저장된 어드레스 값에 해당하는 메모리 위치에 저장된 데이터 값을 읽어서 R2 레지스터에 저장

t1 : R2 ← M[R1]

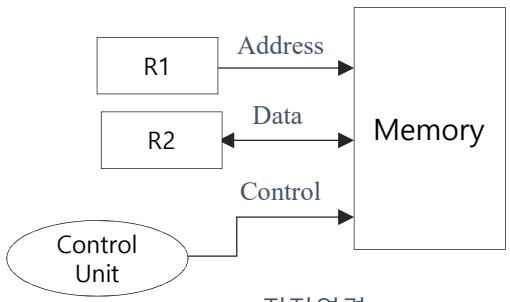
- **메모리 쓰기 동작의 RTL 표현**
  - R1에 저장된 데이터 값을 R2 레지스터에 저장된 어드레스 값에 해당하는 메모리 위치에 쓰기

t2 : M[R2] ← R1

4.1 메모리와 레지스터 연결

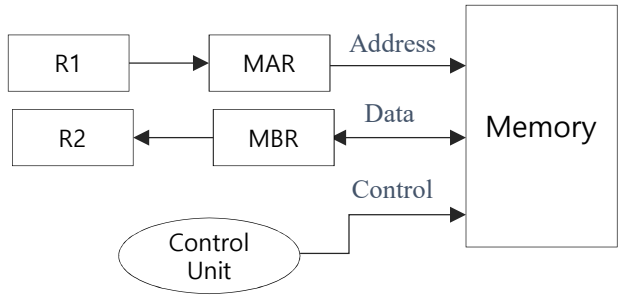
- **연결 방법**
  - 직접 연결
    - 어드레스/데이터 정보를 저장하고 있는 레지스터와 직접 연결
  - 간접 연결
    - 전용 레지스터 사용
      - MAR(Memory Address Register)
      - MBR(Memory Buffer Register)

t1: R2 ← M[R1]



직접연결

t1 : MAR ← R1
t2 : MBR ← M[MAR]
t3 : R2 ← MBR



간접연결

end