12주차 예비보고서

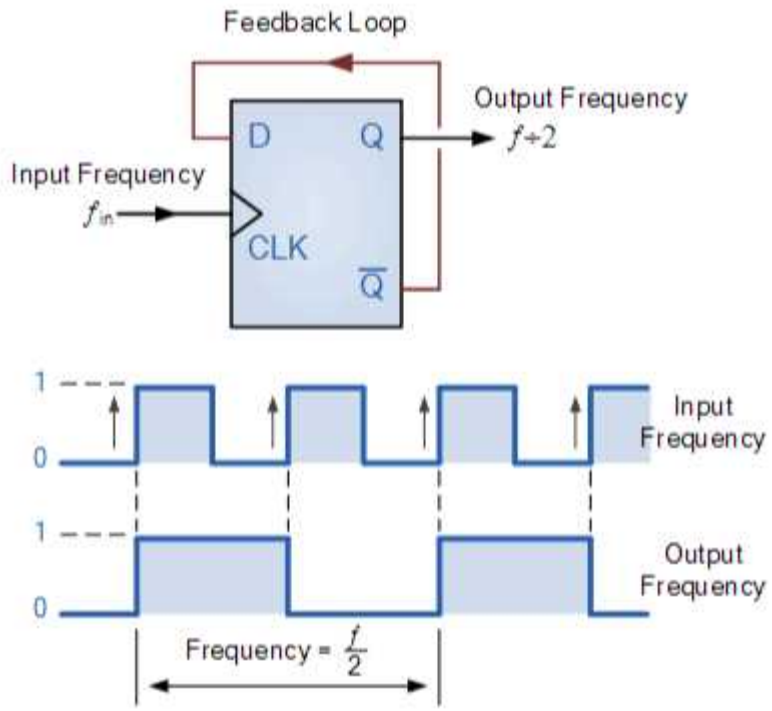
전공: 신문방송학과 학년: 4학년 학번: 20191150 이름: 전현길

1. Counter에 대해서 조사하시오. (예시 포함)

카운터란 데이터 입력이 없이 연속되는 클럭에 따라 정해진 일련의 상태를 반복하는 소자이다. 입력 신호의 주기(pulse)를 세서, 클럭 신호에 따라 값을 증가시키거나 감소하면서 정해진 출력값의 순서를 반복한다.

카운터는 up counter, down counter, up/down counter로 나눌 수 있으며, 각각 값이 증가하는 카운터, 감소하는 카운터, 증가/감소가 모두 가능한 카운터로 나뉜다.

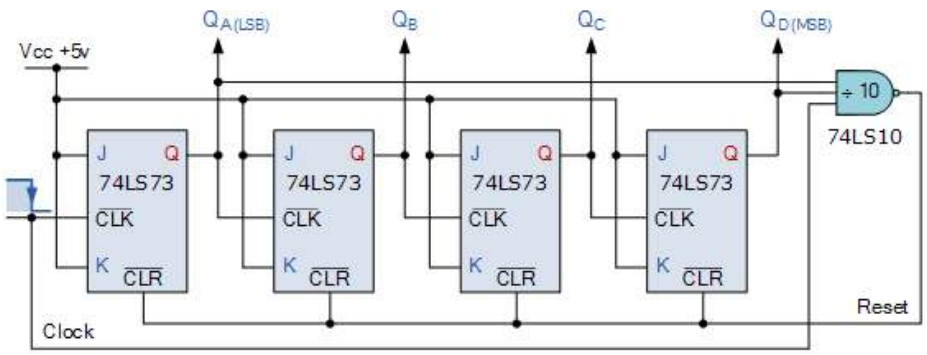
frequency division divide-by-2 counter



위의 예는 1개의 D flip-flop을 통해 주파수를 1/2로 만들 수 있는 매우 간단한 카운터이다. 위 카운터는 rising-edge 입력 펄스가 들어올 때마다 상태를 변화시키기 때문에, 입력 펄스의 정확히 2배의 주기를 갖게 된다. 또 정해진 출력값의 순서는 이 경우 0, 1이다.

2. Decade Counter 에 대해서 조사하시오.

asynchronous decade up counter



Decade 카운터는 주로 BCD(Binary-coded Decimal) 코드를 표현하는 데에 사용되는 카운터로, 클록 펄스를 입력받아 주로 4개의 JK 플립플롭을 연결하여 8421, 5421, 4221 code 등을 4bit 값으로 출력한다. 위 회로는 추후 설명할 비동기식 카운터(asynchronous counter)의 digital circuit diagram이다.

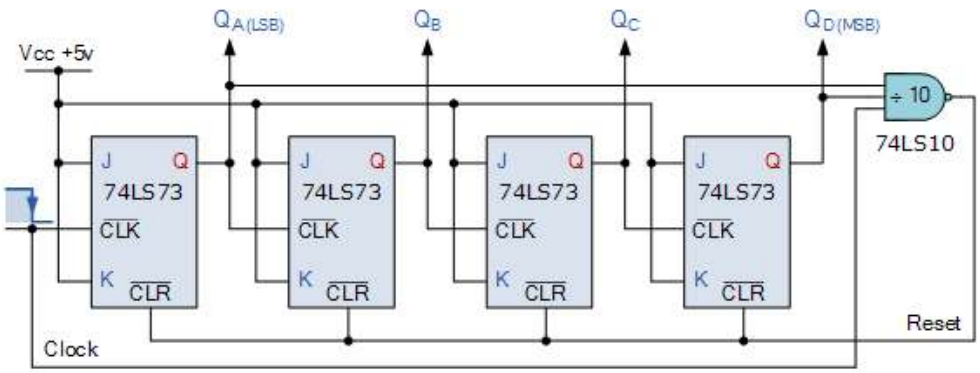
JK 플립플롭의 구현 방식을 고려하고, 위 플립플롭의 JK 회로가 모두 Vcc + 5v와 연결되어 있는 것을 통해 유추할 수 있겠지만, 각 플립플롭은 모두 J, K = 1인 상태로 입력이 고정되어 있고, 클록 펄스가 한 번 켜질 때마다 값을 변경한다. 또 $\overline{CLR}$ 신호에서 CLR 위에 윗줄이 그어져 있는 것을 보면 알 수 있듯이, 이 회로는 **Q_A, Q_D, clock = 1이 되어 NAND 게이트의 출력값이 0이 될 때 모든 JK 플립플롭에 저장되어 있는 값을 0으로 초기화**한다.

오른쪽은 8421 BCD code를 기준으로 구현한 decade counter의 state diagram이다. 일반적으로 **x = 0일 때 입력 신호를 유지하며, x = 1일 때 다음 상태로 이동**한다. 입력값이 1개뿐이기 때문에 Q(n+1)의 상태도를 2열만으로 작성할 수 있지만, 입력값이 2개 이상이라면 그에 따라 상태도를 2^n열만큼 작성해야 한다.

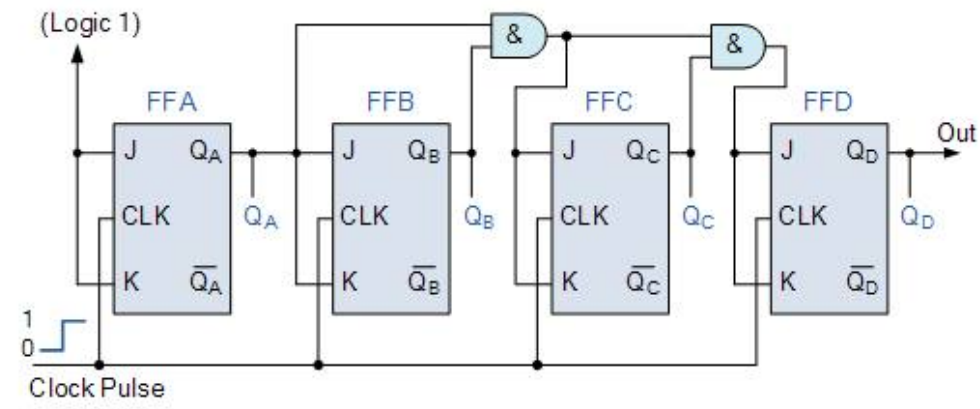
Q(n)	Q(n+1)	
	x = 0	x = 1
0000	0000	0001
0001	0001	0010
0010	0010	0011
0011	0011	0100
0100	0100	0101
0101	0101	0110
0110	0110	0111
0111	0111	1000
1000	1000	1001
1001	1001	0000

3. 비동기식 Counter 및 동기식 Counter에 대해서 조사하시오.

asynchronous decade up counter



synchronous decade up counter



동기식 카운터는 **모든 플립플롭이 동일한 클럭 신호로 동작하는 카운터**이며, **비동기식 카운터**는 **첫 번째 카운터만 외부 클럭 신호에 의해 동작하고 나머지 플립플롭들은 전부 이전 플립플롭의 출력에 의해 구동되는 카운터**를 의미한다. 앞서 살펴보았던 decade counter가 비동기식 카운터이다.

위의 양 카운터의 circuit diagram을 확인하면, 비동기식 카운터의 경우 clock signal이 첫 번째 신호에만 입력으로 그려져 있는 것을 확인할 수 있다. 또 비동기식 카운터의 경우 JK 플립플롭의 출력 신호가 다음 플립플롭의 **클록 신호**로 사용되고 있지만, 동기식 카운터의 경우 곧바로 다음 플립플롭의 J, K 입력이 되거나, 이전 입력들과 AND gate에 입력된 뒤에 J, K 입력이 되는 것을 알 수 있다. 이 때 각 출력값이 어떻게 연결될지는 실제 상태도를 살펴보며

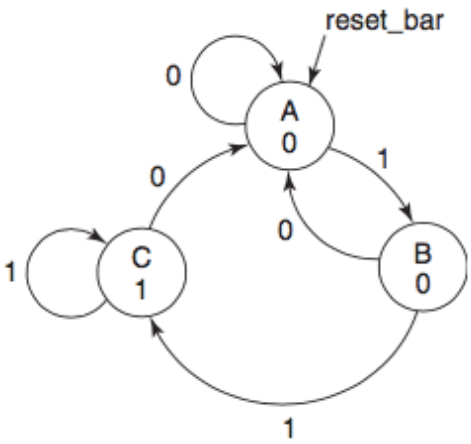
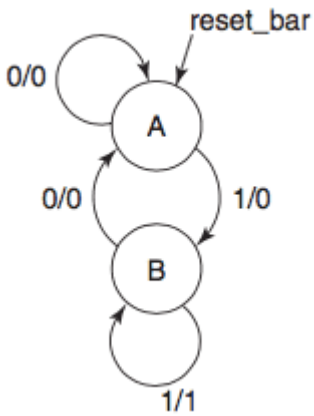
결정해야 한다.

비동기식 카운터의 경우 이전 플립플롭의 출력이 결정되기 전에는 다음 플립플롭의 출력을 결정할 수 없으므로 플립플롭 1개당 전파지연이 축적된다. decade counter의 경우 4bit만큼의 전파 지연만 발생하지만 bit 수가 클 경우 전파 지연은 점차 누적될 수 있다. 반대로 동기식 카운터의 경우 모든 플립플롭이 동시에 클럭되므로 비동기식 카운터에서 발생하는 전파 지연을 최소화할 수 있다.

4. FSM(finite-state machine) 에 대해서 조사하시오.

FSM(finite state machine; 유한상태기계)는 **유한한 상태(state)와, 이들 사이의 전환(transition)**을 다루는 수학적 모델이다. finite-state machine은 유한한 상태의 목록과 초기 상태, 각 전환을 일으키는 입력으로 정의된다.

이 때 FSM은 결정론적 FSM(deterministic FSM)과 비결정론적 FSM(non-deterministic FSM)으로 나뉜다. 이 때 결정론적 FSM은 각 전환이 이전 상태와 입력값 심볼을 통해 고유하게 결정되며, 상태 전환을 위해 입력이 필요한 FSM이다. 좀 더 간단히 설명하면, **현재 상태와 입력에 따른 다음 상태가 유일하고, 상태의 변화를 위해 입력이 반드시 필요**한 FSM이다. 비결정론적 FSM은 이러한 제약을 따르지 않는 FSM을 의미하므로, 넓은 의미로 모든 비결정론적 FSM은 결정론적 FSM에 포함된다. 단, 상황에 따라서 비결정론적 FSM이 결정론적 FSM이 아닌 모든 FSM을 가리키기도 한다.

moore machine	mealy machine
	

FSM은 이외에도 **Moore machine**과 **Mealy machine**으로 구분할 수 있다. Moore machine은 현재 상태만으로 출력이 결정되는 FSM을 가리키며, Meal

y machine은 현재 상태에 더해 추가적인 입력에 따라 출력이 결정되는 FSM을 의미한다.

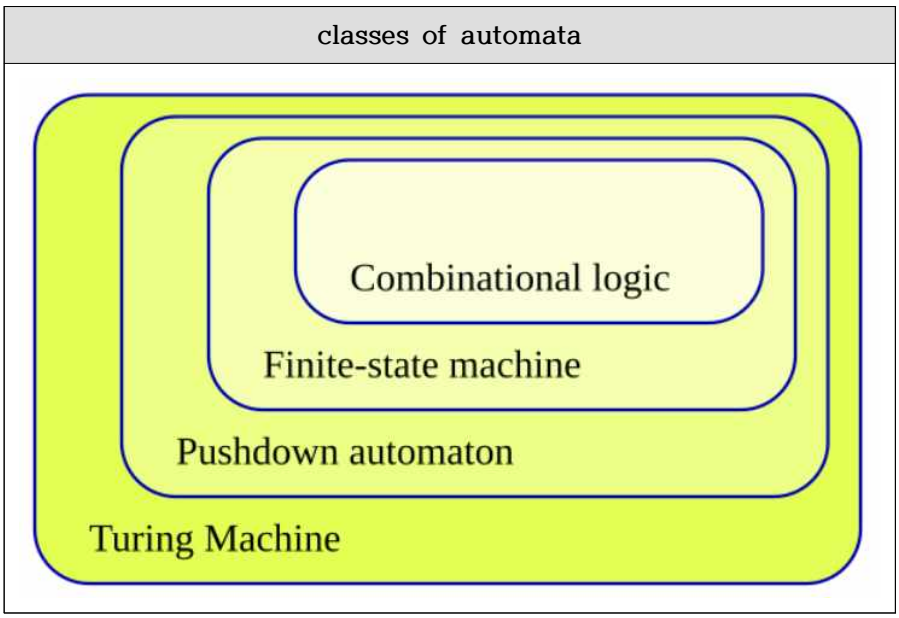
위는 간단한 moore machine과 mealy machine의 예로, 보는 것처럼 **둘 간에 state diagram을 그리는 방법이 조금씩 다르다.**

전형적인 state diagram에서 moore machine의 **동그라미 안의 글자들은 각각의 상태(=A, B, C)와 현재 상태에서의 출력(=0, 1)**을 의미한다. 또 **화살표에 할당되어 있는 글자들은 해당 상태 변화를 일으키는 입력**을 의미한다. 해당 moore machine은 입력이 1bit고, 상태가 3개 존재하므로 모든 입력에 대응되기 위해서 반드시 6개의 화살표가 존재해야 한다. (상태의 수(k) * 입력의 수(2^n) = 상태 변화의 수)

mealy machine의 경우 동그라미 안의 글자들은 현재 상태를 의미한다. 또 화살표에 할당되어 있는 글자들은 **해당 상태 변화를 일으키는 입력과, 현재 상태 + 해당 입력이 들어왔을 때의 출력**을 의미한다.

5. 기타 이론

오토마타 이론(automata theory)은 추상적인 기계 또는 오토마타를 연구하거나, 이를 응용해 해결할 수 있는 문제들을 다루는 학문이다. FSM이 유한한 상태의 목록, 초기 상태, 각 전환을 일으키는 입력으로 구성되듯이, 오토마타 역시 유한한 상태의 목록, 초기 상태, 각 전환을 일으키는 입력으로 구성된다. 이 때 무한한 상태의 목록을 가지 기계는 따로 무한 상태 기계(infinite state machine)로 호칭한다.



오토마타 이론에 따르면 오토마타는 위와 같이 **combinational logic, finite-state machine, push-down automaton, turing machine의 네 단계의 분류**로 나뉜다. 이 때, 가장 기본적인 combinational logic은 우리가 앞 주차에서 살펴보았듯, 상태가 존재하지 않고 정해진 입력에 따라 출력값을 변화시키는 논리 회로를 의미한다.

push-down automaton은 유한 상태 기계에 스택(stack)을 추가한 모델로, **입력값, 현재 상태에 더해 스택 메모리의 현재 top에 들어 있는 데이터에 따라 상태를 변화시키는** 오토마타이다. push-down automaton의 state diagram은 **현재 상태를 diagram의 동그라미에 표시하며, 입력값/스택의 top/현재 상태 변화에 따라 스택의 top의 변화**까지 모두 3가지 값들을 화살표에 할당한다. 이 때, 현재 상태 변화에 따른 스택의 top의 변화는 다시 3가지로 나뉘는데, 자료구조에서 배웠던 push/pop 연산과, top의 data를 바꾸는 연산이다.

turing machine은 **입력 테이프에 쓰여 있는 여러 기호를 일정한 규칙에 따라 변환하여, 출력과 상태를 변화시키면서 지정된 동작을 수행하는 장치**를 의미한다. 이 때 입력과 상태, 출력의 개수는 모두 유한해야 한다.

튜링 머신은 앨런 튜링이 1936년에 제시한 개념으로 정의가 비교적 단순하지 못하다. 그럼에도 불구하고 정의를 그대로 옮기면 다음과 같다. 튜링 머신은 **셀로 나눌 수 있는 테이프(a tape divided into cells), 현재 입력을 읽을 수 있는 헤드(head), 현재 상태를 저장하는 상태 레지스터(state register), 유한 명령어 표(a finite table of instruction)**로 구성되는 오토마타이다.

우리가 일반적으로 접하는 CPU가 튜링 머신의 정의에 부합한다. cell을 사용자 입력 또는 소스 코드로, head를 PC register로, state register는 레지스터, 메모리 등의 내부 저장장치부터, SSD나 HDD, USB 등의 외부 저장장치로 비유할 수 있을 것이다. 유한 명령어 표는 입력되는 값에 따라 어떻게 동작할지를 나타내는 동작표를 의미한다.

유한 명령어 표는 CPU로 비유하기 조금 까다로운데, CPU에 입력되는 테이프를 프로그래밍 언어(C, Java)라고 볼 수도, 어셈블리 언어라고 볼 수도, 기계어라고 볼 수도 있기 때문이다. 단 최소한 각각의 언어에 대해 각 CPU의 동작이 일대일 대응하므로, (유한한) 각각의 입력에 대한 CPU의 동작을 나타내 표로 만든 것이라고 생각할 수 있을 것이다.