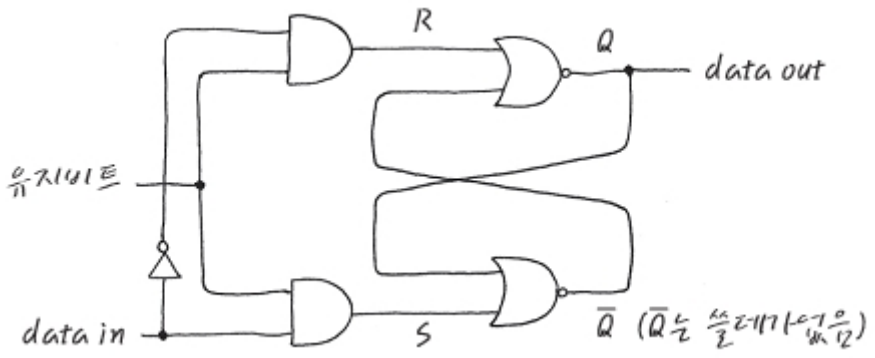
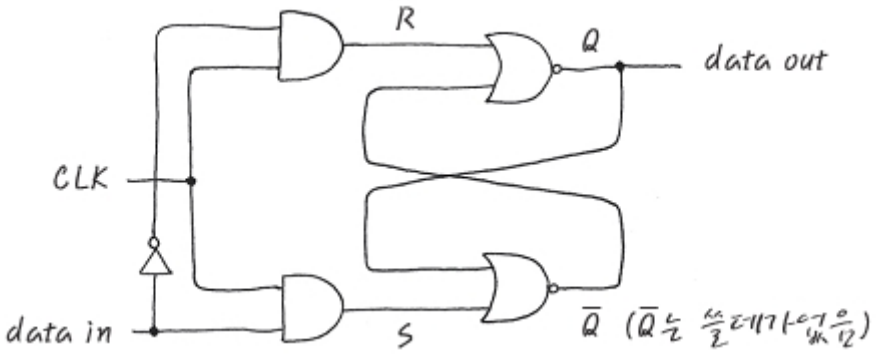
친절한 임베디드 시스템 개발자 되기 강좌 : Clock 이란?

Clock 이란?
Clock이라는 거 왜 있을까요. Embedded System이나, 일반 Computing System이나Clock 속도를 가지고 뭔가를 따지죠. 내 PC는 1GHz인데, 니꺼는 2GHz네? 니꺼가 더 빠르구나, 좋겠다. 뭐 이런식의 애들 대화는 심심치 않게 들을 수 있습니다.
제가 어렸을 때도 처음 XT PC를 장만하고서, 내 PC는 4.77MHz로 동작해라고 득의 양양하게친구한테 자랑 했더니, 내 친구가 내 PC는 AT야 21MHz에서 동작하지. 어디서 쨉도 안되는 PC가지고 나한테 덤벼 라는 반격을 받은 기억이 납니다.
집에 드러누워서 나는 언제나 386을 장만할 수 있을까. 386이면 33MHz 우와 별의 별걸 다 할 수 있겠지? 라는 어이 없는 상상을 하면서
돈 많이 벌어야지 라는 엉뚱한 생각을 한 적도 있죠.
Clock은 디지털회로에서 심장박동이라고 할 수 있습니다.주기적인 전기적 펄스이지요. Clock에 따라서 모든 것이 Synchronization을 맞추어 동작을 합니다. 박자를 맞추는거에요. 쿵짝쿵짝쿵짝.
동기화 (Synchonization)이라는 말을 많이 만날 것입니다. Synchonization이라는 말을 우리나라 말로 바꾸면 두개의 단어로 압축 할 수 있을 것 같아요.
1. 박자를 맞추다 (Hardware Level)2. 순서를 맞추다 (Software Level)
박자를 맞추는 건 말 그대로 작전 수행시에 다 같이 시계를 맞추고 작전을 수행하는것과 같은의미이고요. 그러니까 system 적으로는 Clock이 모든 행위의 단위가 된다는 의미입니다.또는 주기적인 2개의 사건을 자알~ 콘트롤하여 결합시켜 일정한 위상관계를 유지 시키는
행위이고요. 순서를 맞춘다는건 사건을 정해진 순서에 서로 엉크러짐 없이 수행할 수 있도록 한다는 의미로먼저 하겠다는 넘과 덩달아 하겠다는 넘의 순서를 잘 정리해 주는 것을 의미합니다.
Digital 세계에서는 이 Clock이라는 것이 진짜 시간이라는 겁니다. 이 Clock에 맞춰서 모든 것이 일사불란하게 움직이는 거에요.
어떻게 Clock이라는게 그런 역할을 하는건가요?
일단 Register (Latch)에서 Clock이 어떤 역할을 하는지 한번 보시죠. 앞에 Flip Flop에서 Write 신호가 기억나시겠죠. 이 부분을 유지 비트로 말을 바꾸어 보겠습니다.



유지비트로 말을 바꾸고 나니, Data In에 대해서 유지비트가 1일 때만 Data를 바꿀 수 있고 유지비트가 0일때는 마지막 유지비트가 1일 때의 Data가 유지 됩니다. 자, 이제는 이 유지비트라는 걸 Clock이라는 걸로 이름을 바꾸어 볼까요? 좀 더 근사한 말로 표현이 되겠습니다.



똑같은 말을 다시 써 보면 CLK가 High일 때 값을 저장 할 수 있고, CLK가 LOW일 때는 마지막 CLK가 High일 때의 값을 저장하고 있습니다. 자, 일단 임시저장소 인 Latch가 CLK과 Data를 주면 그에 맞추어서 Input 값을 저장하거나, Output으로 내놓거나 하는 일을 하지요.

Digital 회로는 논리회로와 이런 Latch회로로 구성됩니다. 문제는 모든 Digital회로를 논리회로로만 구성해서는 수지타산이 맞지 않다는게 문제입니다. 예를 들어, 이미 살펴 보았던 2 bit Adder를 이용하여, 자기 자신을 20번 더하는 논리회로를 만든다고 해보죠. 이런 경우라면 2 bit Adder를 20개 두어야겠죠? 또는 이런 20회의 계산을 하기 위해 어려운 진리표를 또 만들어서 그런 논리 회로를 만들어야 하는

수고로움이 있습니다. 또한 곱셈 회로라도 한다 치면, 자리수가 늘어날 수록 엄청 복잡해 지는 진리표를 만들기 졸라 머리 아플 겁니다.

하지만 중간에 Register를 하나 두고 한번 계산된 값을 Register에 넣어 둔 다음에 20번 Feed back하는 회로를 만들면 더욱 간단하겠죠.

이런 의미에서는 한 단계마다 Clcok이 똑닥똑닥 Synchronization을 맞추어 줘야 합니다. Synchronization을 맞춘다 - 라는 뜻은 박자를 맞추어 순서를 엉크러지지 않게 한다가 딱 알맞겠습니다. -

한번 두번 세번 이렇게 tick을 주어서 이번엔 한번 더해서 Register에 저장할 차례야 Register에서 가져와서 또 더할 차례야라는 순서를 알려줘야죠. 그 박자에 맞추어서 동작하는 원리이지요.

또는 이런 예도 있을 수 있겠군요. 서로 처리 속도가 다른 논리회로를 여러개 늘어놓고서 맨처음 논리회로부터 맨 마지막 논리회로까지

데이터가 지나간다고 할 때, 그리고 데이터가 연속적으로 마구잡이로 맨 처음 논리회로에입력될 때 맨 마지막에서의 출력은 어떻게 판단해야 할까요?그것도 Clock을 이용한 Synchronization을 이용하죠. 논리회로들 사이에 Latch를 두고서 Clock을 계속 주게 되면

한칸씩 데이터가 밀려나가면서 논리회로 개수후에, 즉 논리회로 개수 만큼의 clock후에지금의 데이터가 나타나겠죠? 그렇지 않으면 출력에서는 이미 가져가버린 데이터를 지금 데이터라고 판단할 수도 있고, 데이터를 못가져가는 일도 벌어질 것이지요.

이런 의미에서 모든 Digital회로는 Clock에 의해서 그 동작이 결정된다고 봐야 합니다. 그러면, Clock은 무조건 빠를 수록 좋겠네요? 으하하하. 하지만, 각 디바이스들은 자기가 동작할 수 있는 시간적인 동작 범위가 있으니까 무조건 빠르다고 될 일은 아닙니다. 각각 자기가 소화할 수 있는 속도가 있다는 말이죠. 이건 물리적인 내용이 다분하며, 예를 들면 TR이나, FET는 스위칭 특성이 있어서, 입력신호가 들어온 후 출력시간이 나오기까지 약간의 시간이 걸리며, 그 시간을 delay time (전달지연시간)이라고 부릅니다. 그 원인 중 하나는 완전한 High 신호를 줄때 10%~90%로 올라가기 까지상승시간이 걸리며, 모든 신호의 변화는 이런 rising 또는 falling delay를 갖게 됩니다.

요 부분에 대한 상세한 내용은 "Timing 그리고 spec 읽기"에 자세하게 다루게 되지요.그러니까 각각의 Device IC들에 대해서 동작 범위의 Clock을 feeding해 주는 것이 중요하고요. MCU*의 경우에도 자기가 동작 가능한 속도가 있으니 그 속도에 맞추어 전체 시스템 속도를

조절해 주는 것이 중요합니다.

보통 clcok은 다음 처럼 표시하며,

모든 순차 논리 회로나, 순서가 중요한 경우에는 system에 흐르는 맥박인 clock을 기준으로 움직이게 됩니다. clokc은 system이 하는 모든 일의 순서와 timing을 정해주니 얼마나 중요한 건지 말 안해도 아시겠죠.
째깍째깍.

더 이후에 Clock을 configuration 하는 방법에 대해서도 고고 할겁니다.

. 서로 처리 속도가 틀린 Digital 회로들끼리 연결되어 있는 경우에는 처리 속도가 가장 느린 Digital 회로에 clock을 맞춰야 전체 회로가 제대로 동작하겠죠. 이것이 바로, 한없이 clock을 올려서는 안되는 이유이고요, Device Spec에서의 Feeding Clock을 제대로 확인해야 하는 이유에요.

갑자기 MCU라는 용어가 나왔는데, ARM SoC (System On Chip) - ARM 그렇고 말고~ 편에서 보시면, MCU의 용어 설명이 되어 있으니 참고하세요.