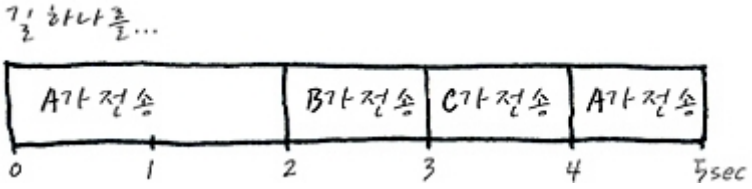
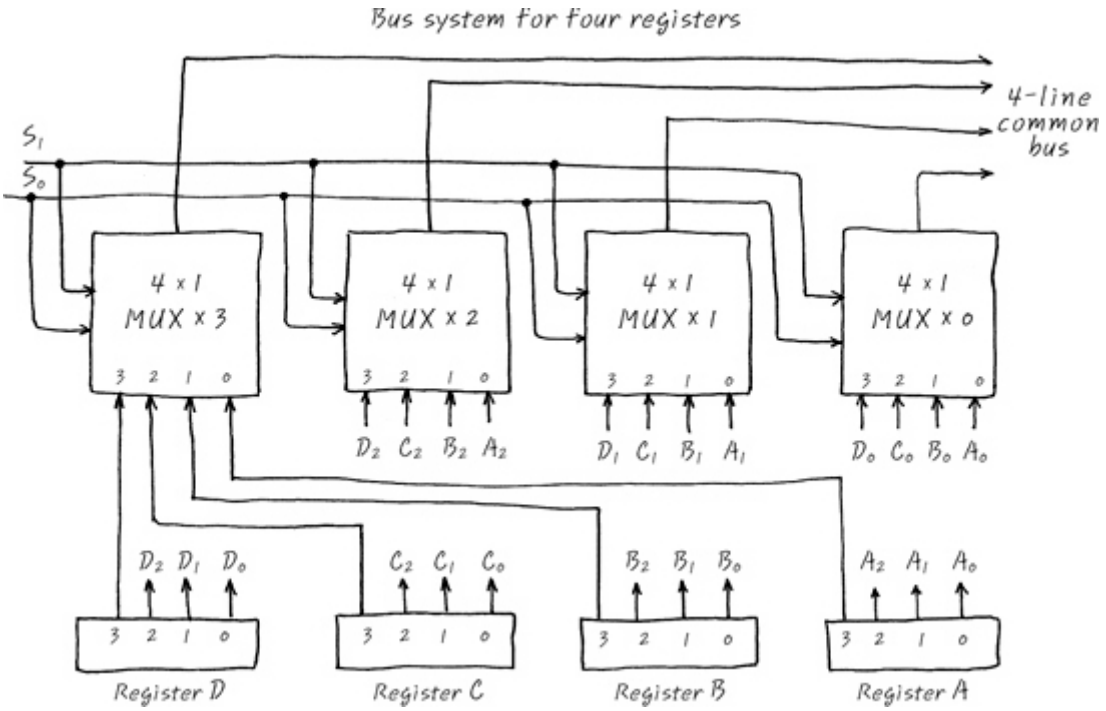
친절한 임베디드 시스템 개발자 되기 강좌 : Bus Transfer Mechanism

Bus Transfer MechanismBus

는 여러명의 승객을 태우고 운송을 하는 수단을 말한다고 합니다. 원래 omnibus라는 승합마차라는 말에서 왔다고 하는데, 결국엔 운송을 담당하는 수단을 의미합니다만, 실은 우리가 다룰 버스는 이런 버스라고 생각하기에는 조금 상상력이 필요하다는
의견입니다.
Data 측면에서 운송수단이 맞긴 맞습니다만, 제 개인 적인 의견으로는Bus라는 말보다는 Freeway 정도로 바꾸는게 좋지 않을 까 생각합니다만. Freeway이긴 하지만 한번에 한 녀석만 탈 수 있는 뭐 그런 길이죠.
원론적으로 Bus란 장치들이 정보 공유를 위해서 공유하는 선들의 집합이라고 합니다. 가끔 모든 Digital 신호들은 bus를 통해서 이동한다는 말이 여기저기 보이는데, 저는 이동한다는 말이 별로 마음에 안드는지라, 정확히 얘기 한다면, 어떤 한 특정 시점에서시간을 멈추어 Bus를 쳐다 본다면, Bus위에는 그 당시에 Bus를 쓸 수 있는 허가를 받은 장치들의 신호만이 보인다고 보면 됩니다. A와 B를 wire로 연결하여 A - B 로 만들어 놓은 다음에 A에서 B로 data를 전송하면, 전기 신호도 빛의 속도와 마찬가지의 속도로 이동을 하니, 이동이라기 보담은 wire - 위에는 동시에 A가 보낸 신호가 wire위에 떠 있다고 봐야 합니다.
결국, 마구 엉킨 bus system은 clock에 맞추어 한 clock의 시간을 쪼개어 봤을 때, 한가지 종류의 신호만이 존재하게 되는것이며, 계속 시간을 흘려 보내면서 관찰했을 때는각 clock 단위 시간마다 번쩍 번쩍 하면서 다른 종류의 신호들이 네온 사인 처럼나타났다가 사라지는 것입니다.
그러니까, bus로 통신할 필요가 있는 여러개의 회로들을 엮어 놓으면,clock에 따라 - 시간에 따라 - 서로 통신할 필요가 있는 것들끼리의 신호가 bus line에 떠 있는거죠. 번쩍 번쩍 나타났다 사라졌다.
그러니까, 길 하나를 다음 그림처럼..



짜잔, Bus Bandwidth는 단위시간당 얼마나 많은 data들이 번쩍 일 수 있느냐의 문제와 같습니다. (clock 주파수가 1Hz인 case가 되겠네요)
이런 뒤엉킨 Bus system의 미쟝센은 아비터 (Arbiter)가 맡고 있습니다. 전체적인 신호등 역할을 하는 건데, 현재 시점에서 어느 장치가 Bus를 사용할 지를결정해주는 역할을 합니다. 가스덩어리 아비터가 시공간 왜곡 기술의 스테세스 필드를 사용하는 것과는 다른 얘깁니다만.
전형적인 버스 시스템중 Register에서의 버스를 예로 고고레츠고 합니다. 아래의 그림 같은 경우는 Mux를 이용한 Register끼리의 통신 통로로서의 Bus입니다. 1 bit 씩도 transfer 가능하며, 공통 버스를 이용하는 시스템인 셈입니다. 각각의 register는 4bit latch이며, 4개의 MUX에 연결되어 있습니다.
S1, S0가 MUX control 신호이며, S1S0에 의하여, MUX의 enable port 번호가 결정됩니다. 예를 들어, S1 S0 가 0, 0 이면 0번 port가 enable S1 S0 가 0, 1 이면 1번 port가 enable S1 S0 가 1, 0 이면 2번 port가 enable S1 S0 가 1, 1 이면 3번 port가 enable 되는 형식인 셈입니다. 이 상황에서 예를 들어 S1, S0가 1, 0이면 각각의 MUX의 2번 port만이 input이 가능한데, MUX 각각의 2번 port는 왼쪽에서 부터 C3, C2, C1, C0가 연결되어 있어서 Register C가Bus를 번쩍 독점하게 되는 구조이지요.



이런식으로 따져서 진리표를 다시 그려보면 S1 S0 EnabledPort Register 0 0 0 A 0 1 1 B 1 0 2 C 1 1 3 D 로 표현할 수 있겠사옵니다. 이런 시스템이라면 S1, S0 신호선을 잘 control해서그때 그때 Bus 사용권을 각각의 register들에게 할당할 수 있겠지요?

보통은 이런 S1, S0를 Arbiter라는 신호등이 자신의 Algorithm Policy에 의거하여, 사용권을 부여 합니다 - 실은 CPU 내부에서는 이런 일을 CU*가 담당하고 있습니다만, 버스를 쓰고 싶어하는 주체가 Register가 아닌 경우인 CPU 외부에서 이런 버스 사용권 경쟁이

일어나는 경우라면 Arbiter가 일진 먹는거지요 -

버스에는 주소버스, 데이터버스, 제어 및 상태버스 등의 3가지 종류의 버스가 있습니다. 주소 버스에는 현재 MCU가 access하려는 주소가 번쩍거리고 그러니까 MCU에서 나오는 단방향신호입니다. 데이터버스는 양방향으로서 MCU가 data를 주거나 받을 수 있습니다. 제어버스는 단방향일수도 양방향일 수도 있는데, MPU가 참조하려는 상태에 관한 정보들이

번쩍 번쩍 합니다. 번쩍 번쩍.

CU라는 용어가 궁금하시면, "확장 to the CPU - How CPU works" 편을 애용해주세요.

by 히언 | 2009/05/27 20:11 | 하드웨어꼴라쥬 | 트랙백 | 핑백(2) | 덧글(2)