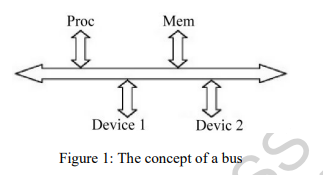
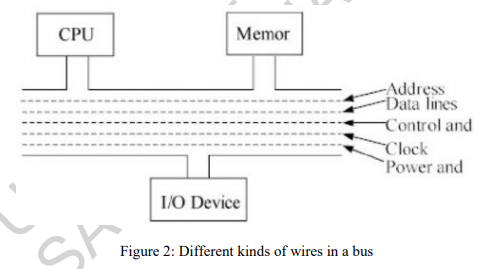
1. Introduction A typical computer system is composed of several components such as the Central Processing Unit (CPU), memory chips, and Input/Output (I/O) devices. A bus is a common pathway or a set of wires that interconnect these various subsystems. The bus thus allows the different components to communicate with each other. The concept of a bus is illustrated in Figure 1. A bus, in computer language, is a channel over which information flows between units or devices. It typically has access points, or places into which a device can tap to become part of the channel. Most buses are bidirectional and devices can send or receive information. A bus is a shared communication link between the different

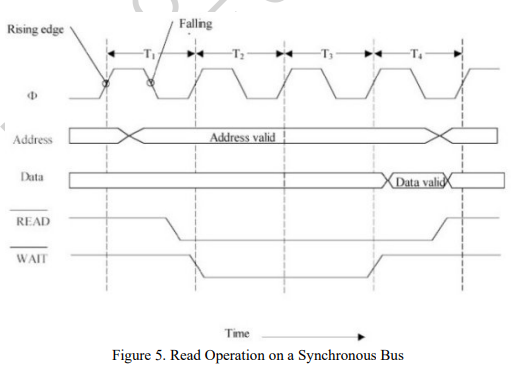
devices. It allows to add new devices easily and facilitates portability of peripheral devices between different computer systems. However, if too many devices are connected to the same bus, the bandwidth of the bus can become a bottleneck. Typically more than two devices or subsystems are involved in a bus, and channels connecting only two components are some times referred to as ports instead of buses.



Buses often include wires to carry signals for addresses, data, control, status, clock, power and ground as illustrated in Figure 2. The address lines indicate the source or destination of the data on the data lines. Control lines are used to implement the bus protocol. Often there are lines to request bus control, to handle interrupts, etc. Status lines indicate the progress of the current transaction. Clock signals are used in synchronous bus systems to synchronize bus operation.



1. Bus Protocols A bus is a communication channel shared by many devices and hence rules need to be established in order for the communication to happen correctly. These rules are called bus protocols. Design of a bus architecture involves several tradeoffs related to the width of the data bus, data transfer size, bus protocols, clocking, etc. Depending on whether the bus transactions are controlled by a clock or not, buses are classified into synchronous and asynchronous buses. Depending on whether the data bits are sent on parallel wires or multiplexed onto one single wire, there are parallel and serial buses. Control of the bus communication in the presence of multiple devices necessitates defined procedures called arbitration schemes. In this section, different kinds of buses and arbitration schemes are described. 2.1. Synchronous and Asynchronous Buses In a synchronous bus, bus operations are synchronized with reference to a clock signal. The bus clock is generally derived from the computer system clock, however, often it is slower than the master clock. For instance, 66MHz buses are used in systems with a processor clock of over 500MHz. Buses were traditionally slower than processors because memory access times are typically longer than processor clock cycles. A bus transaction often takes several clock cycles, although the cycles are collectively referred to by many as a bus cycle



A memory read transaction on the synchronous bus typically proceeds as illustrated in Fig. 5. During the first clock cycle the CPU places the address of the location it wants to read, on the address lines of the bus. Later during the same clock cycle, once the address lines have stabilized, the READ request is asserted by the CPU. Many times, some of these control signals are active low and asserting the signal means that they are pulled low. A few clock cycles are needed for the memory to perform accessing of the requested location. In a simple non-pipelined bus, these appear as wait states and the data is placed on the bus by the memory after the tow or three wait cycles. The CPU then releases the bus by deasserting the READ control signal. The write transaction is similar except that the processor is the data source and the WRITE signal is the one that is asserted. Different bus architectures synchronize bus operations with respect to the rising edge or falling edge or level of the clock signal. An asynchronous bus has no system clock. Handshaking is done to properly conduct the transmission of data between the sender and the receiver. The process is illustrated in Fig. 6. For example, in an asynchronous read operation, the bus master puts the address and control signals on the bus and then asserts a synchronization signal. The synchronization signal from the master prompts the slave to get synchronized and once it has accessed the data, it asserts its own synchronization signal. The slave's synchronization signal indicates to the processor that there is valid data on the bus, and it reads the data. The master then deasserts its synchronization signal, which indicates to the slave that the master has read the data. The slave then deasserts its synchronization signal. This method of synchronization is referred to as a full handshake. Note that there is no clock and that starting and ending of the data transfer are indicated by special synchronization signals. An asynchronous communication protocol can be considered as a pair of Finite State machines (FSMs) that operate in such a way that one FSM does not proceed until the other FSM has reached a certain state.

