ASYNCHRONOUS DATA TRANSFER

The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. Clock pulses are applied to all registers within a unit and all data transfers among internal registers occurs simultaneously during the occurance of clock pulse. Two units such as CPU and I/O interface, are designed independently of each other. If register in the interface share a common clock with CPU registers, the transfer b/w the two units is said to be synchronous.

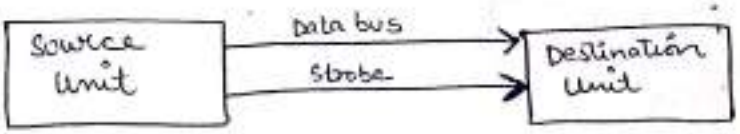
In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other.

Asynchronous data transfer between two independent units requires that control signals be transmitted b/w the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by Strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

Another method used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as Handshaking

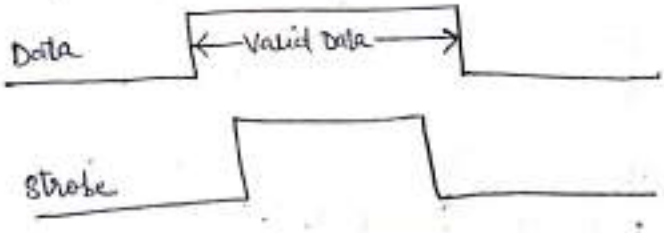
In general case we consider the transmitting unit as the source and receiving unit as the destination. For Eg: The CPU is the source unit during an output or a write transfer and it is the destination unit during an input or read transfer. It is necessary to specify the asynchronous transfer b/w two independent units by means of timing diagram that shows relationship that exist b/w control signals & the data in the buses.

STROBE CONTROL



(Source initiated transfer)

(a) BLOCK DIAGRAM



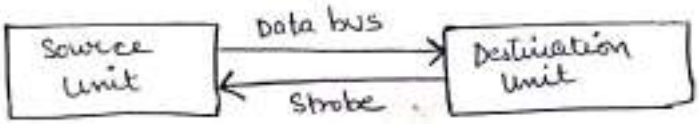
(b) TIMING DIAGRAM

The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.

The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

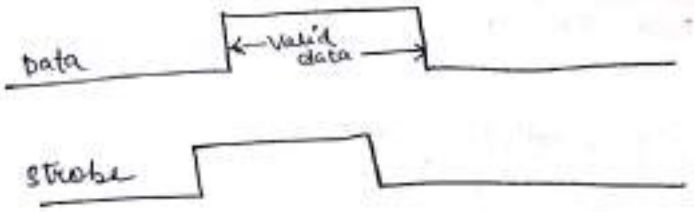
As shown in diagram, the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates strobe pulse. The information on data bus and strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data. Often the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers. The source removes the data from the bus a brief period after it disables its strobe pulse. The fact that the strobe signal is disabled indicates that the data bus doesnot contain valid data.

eg:- The strobe in this figure could be memory-write control signal from the CPU to a memory unit. The source being a CPU, places a word on the data bus and informs the memory unit, which is the destination, that this is a write operation



(destination initiated transfer)

(a) BLOCK DIAGRAM



In this case the destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid & remain in the bus long enough for the destination unit to accept it. The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disable the strobe. The source removes the data from the bus after a predetermined time interval.

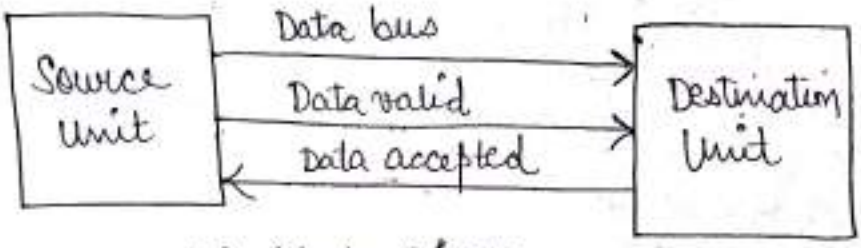
Eg:- The strobe in this figure could be memory-read control signal from the CPU to a memory unit. The destination, the CPU, initiates the read operations to inform the memory, which is the source, to place a selected word into the data bus.

HANDSHAKING

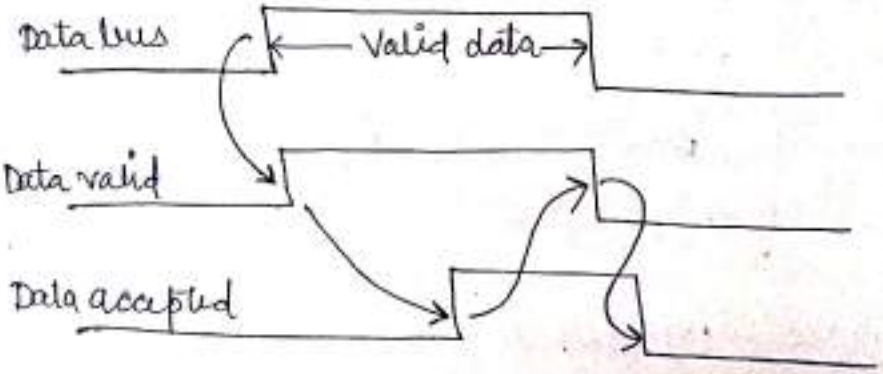
The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed data on the bus.

The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer. The basic principle of two-wire handshaking method of data transfer is as follow:

One control line is in the same direction as the data flow in the bus from the source to destination. It is used by the source unit to inform the destination unit whether there are valid data in the bus.
The other control line is in the other direction from the destination to the source. It is used by the destination unid to inform the source whether it can accept data. The sequence of control during the transfer depends on the unit that initiates the transfer.

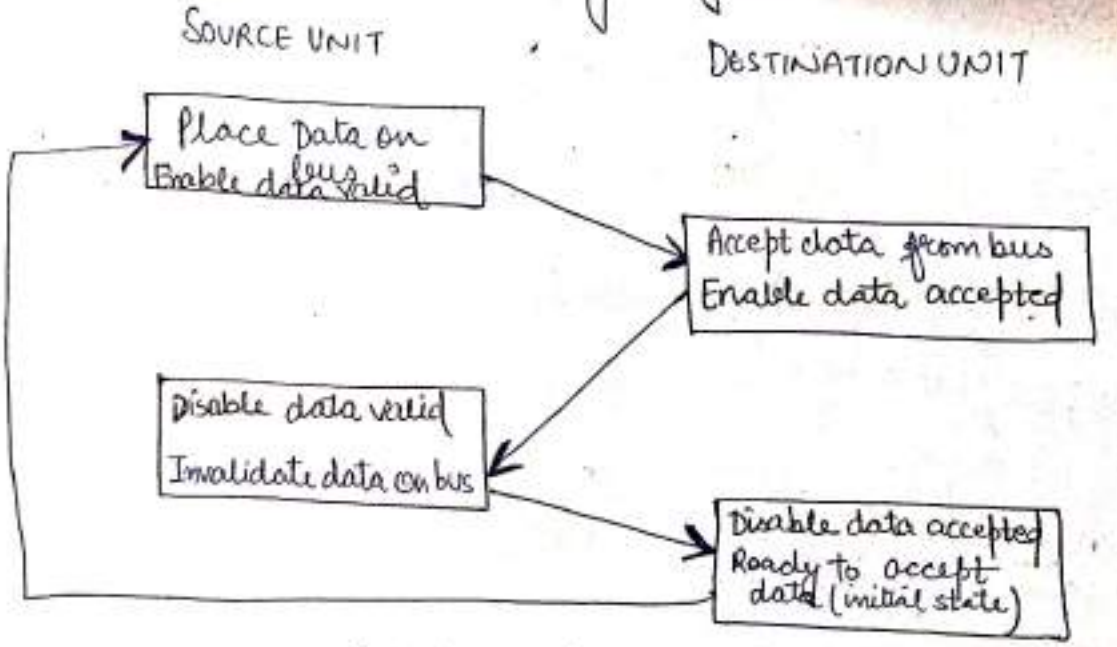


(a) Block Diagram



Data transfer initiated by source

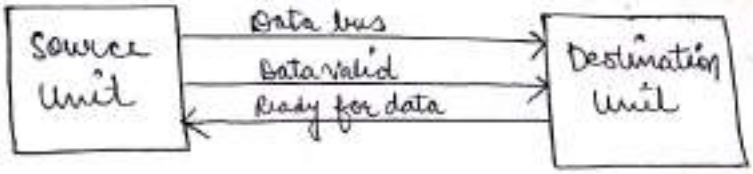
(b) Timing Diagram



(c) Sequence of events

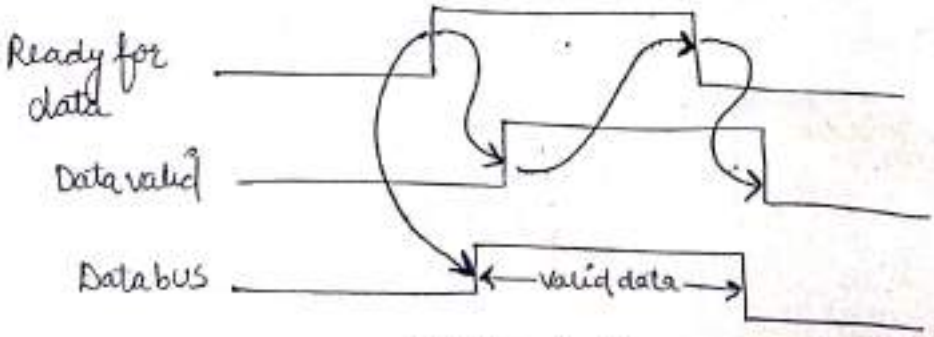
Two handshaking lines are data valid, which is generated by source unit, and data accepted, generated by the destination unit. The timing diagram shows the exchange of signals b/w the two units. The sequence of events listed shows four possible states that the system can be at any given time.

The SOURCE UNIT initiates the transfer by placing the data on the bus and enabling its data valid signal. The data accepted signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its data valid signal, which invalidates the data on the bus. The destination unit then disable its data accepted signal & the system goes into its initial state. The source doesnot send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.

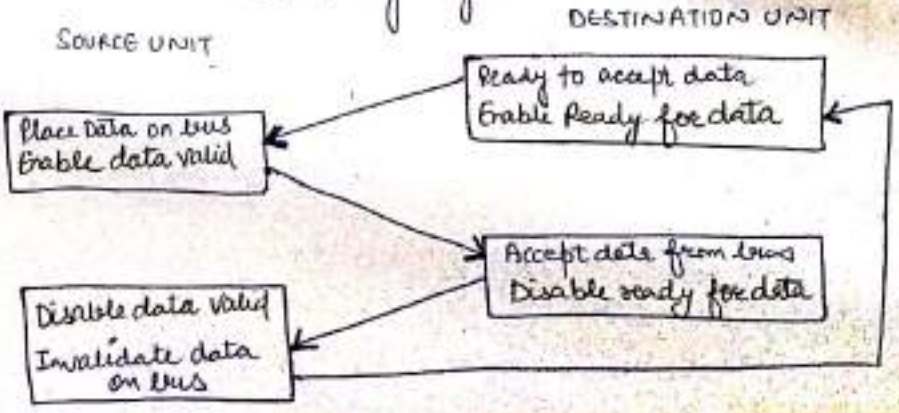


(a) Block diagram

Destination initiated transfer



(b) Timing diagram



(c) Sequence of Events

The Source Unit in this case doesnot place data on the bus until after it receives the ready for data signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source-initiated case.

The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation of both units. If one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a timeout mechanism which produces an ALARM if the data transfer is not completed within a predetermined time. The timeout is implemented by means of an internal clock that starts counting time when the unit enables one of its handshaking control signals.

Asynchronous Serial Transfer

The transfer of data between two units may be done parallel / serial

Parallel Data Transmission

1. In parallel data transmission, each bit of message has its own path and the total message is transmitted at the same time. This means that n-bit message must be transmitted through n seperate conductor paths.
2. It is fast but requires many wires.
3. It is used for short distances

Serial Data Transmission

1. In Serial data transmission, each bit in the message is sent in sequence once at a time. This method requires the use of one pair of conductor or one conductor.
2. It is slower but is less expensive.
3. It is used for long distances.

Serial Transmission can be — Synchronous / Asynchronous

In Synchronous transmission, the two units share a common clock frequency and bits are transmitted

continuously at the rate dictated by clock pulse.

In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.

A serial asynchronous data transmission technique used in many interactive terminals employs special bits that are inserted at both ends of character code. With this technique, each character consists of three parts: a start bit, character bits & stop bits. The convention is that the transmitter rests at the 1-state when no characters are transmitted. The first bit called the start bit, is always 0 and is used to indicate the beginning of character. The last bit called the stop bit is always 1.

(Asynchronous serial transmission)

A transmitted character can be detected by the receiver from knowledge of the transmission rules:

1. When a character is not being sent, the line is kept in 1-state
2. The initiation of a character transmission is detected from the start bit, which is always 0.
3. The character bits always follow the start bit.
4. After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for atleast one bit time.

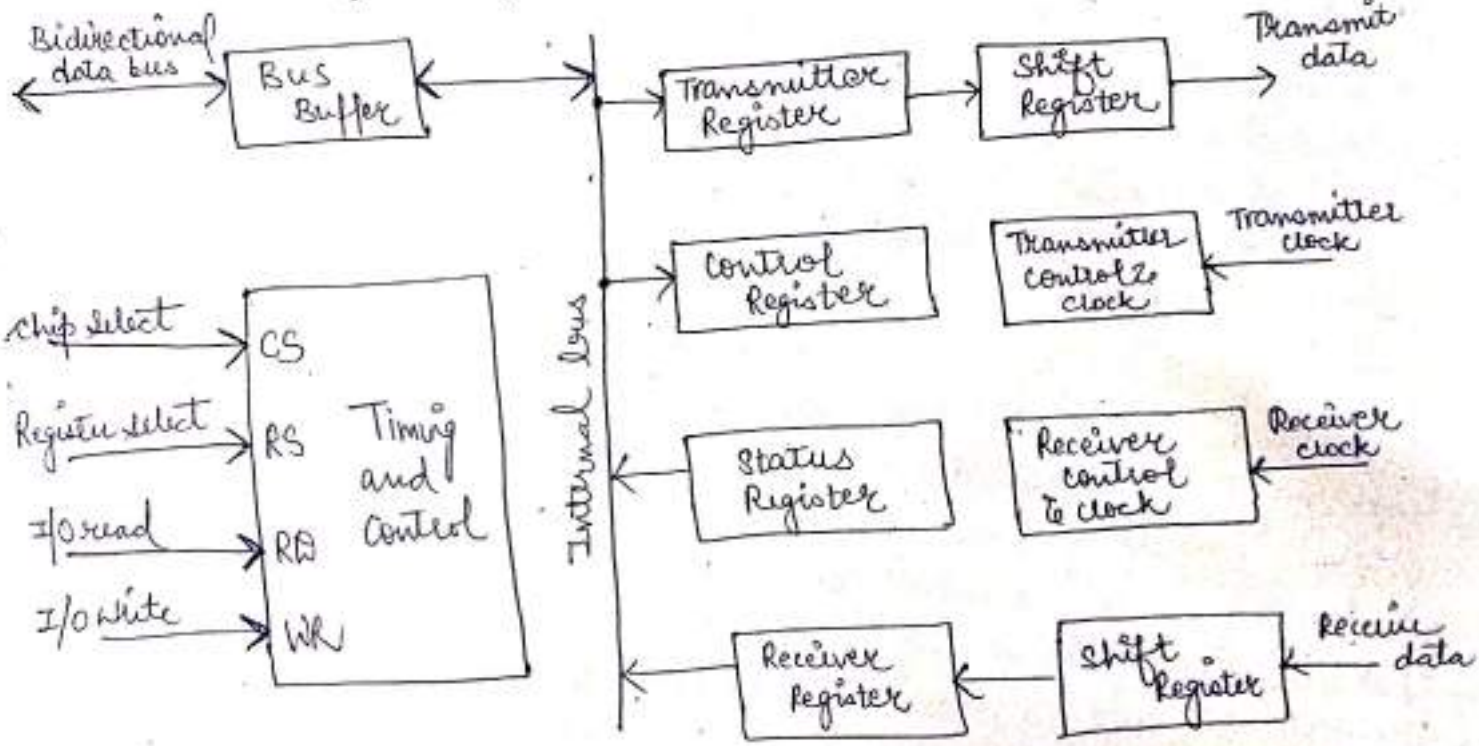
Consider the serial transmission of a terminal whose transfer rate is 10 character per second. Each transmitted character

consists of a start bit, eight information bits and two stop bits, for a total 11 bits. Ten character per second means that each character takes 0.1s for transfer. Since there are 11 bits to be transmitted, it follows that the bit time is 9.09ms.

The baud rate is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second. Ten character per second with an 11 bit format has a transfer rate of 110 baud.

ASYNCHRONOUS COMMUNICATION INTERFACE

The block diagram for asynchronous communication interface is :-



Block diagram of asynchronous communication interface

CS	RS	Operation	Register Selected
0	X	X	None: data bus in high impedence
1	0	WR	Transmitter Register
1	1	WR	Control Register
1	0	RD	Receiver Register
1	1	RD	Status Register

It functions as both a transmitter and a receiver. The interface is initialized for a particular mode of transfer by a means of control byte that is loaded into its control Register. The transmitter register accepts a data byte from the CPU through the databus. This byte is transferred to a shift register for serial transmission. The receiver portion receives serial information into another shift register, and when a complete data byte is accumulated it is transferred to the receiver register. The CPU can select the receiver register to read the byte through the data bus. The bits in the status register are used for input and output flags and for recording certain errors that may occur during the transmission. The chip select (CS) input is used to select the interface through the address bus. The Register Select (RS) is associated with the read (RD) and write (WR) controls. Two registers are write only and two are read-only. The register selected is a function of the RS value and the RD and WR status as listed in table.

The operation of the asynchronous communication interface is initialized by the CPU by sending a byte to the control Register. The initialization procedure places the interface in a specific mode of operation as it defines certain parameters such as baud rate to use, how many stop bits are appended to each character. Two bits in the status register are used as flags. One bit are used to indicate whether the transmitter register is empty and another bit is used to indicate whether the received register is full.

OPERATION OF TRANSMITTER PORTION OF INTERFACE:

The CPU reads the status register and checks the flag to see if the transmitter register is empty. If it is empty, the CPU transfer a character to the transmitter register and interface clears the flag to mark the register full. The first bit in the transmitter shift register is used to 0 to generate a start bit. The character is transferred in parallel from the transmitter register to the shift register and number of stop bits are appended into the shift register. The transmitter register is marked empty

The character can now be transmitted one bit at a time by shifting the data in the shift register at specified baud rate. The CPU can transfer another character to the transmitter register after checking the flag in the status register. The interface is said to be double buffered because a new character can be loaded as soon as previous one starts transmission.

OPERATION OF RECEIVER PORTION OF INTERFACE :

The receiver receive data input is in the 1-state when the line is idle. The receiver control monitors the receive-data line for a 0 signal to detect the occurance of a start bit. Once a start bit has been detected, the incoming bits of the character are shifted into the shift register at the prescribed baud rate. The character without the start and stop bits is then transferred in parallel from the shift register to the receiver register. The flag in status register is set to indicate that the receiver register is full. The CPU reads the status register and check the flag, and if set, it reads the data from the receiver register.

The interface checks for any possible errors during transmission and sets appropriate bit in the status register. Three possible errors that the interface checks during transmission are parity error, framing error, overrun error.

PARITY ERROR occurs if the number of 1's in the received data is not the correct parity.

FRAMING ERROR occurs if the right number of stop bits is not detected at the end of the received character.

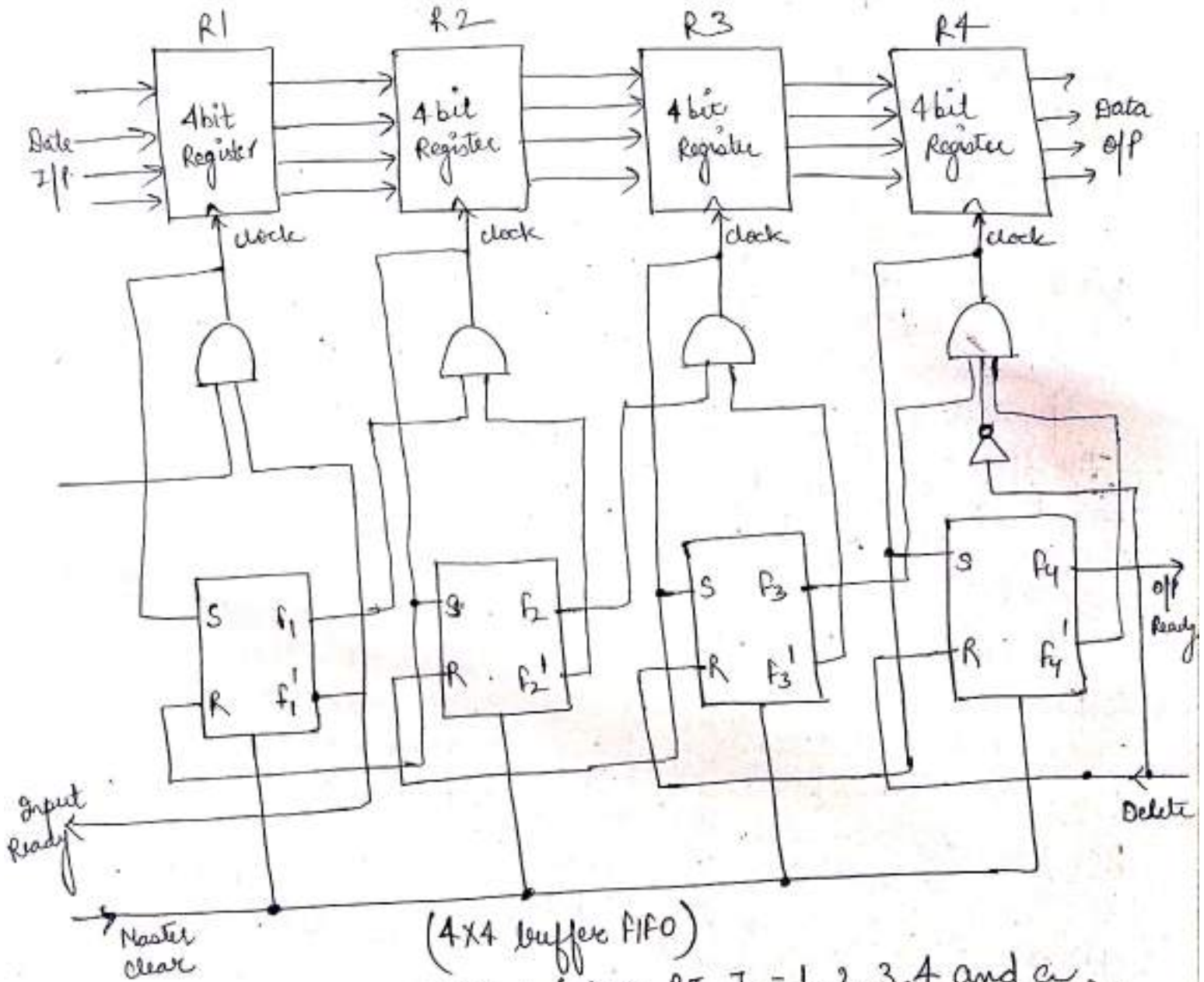
OVERRUN ERROR occurs if the CPU doesnot read the character from the receiver register before the next one becomes available in the shift register.

FIRST IN, FIRST OUT BUFFER

A first in, first-out (FIFO) buffer is a memory unit that stores information in such a manner that the item first in is the item first out. A FIFO buffer comes with seperate input and output terminals. The important feature of this buffer is that it

can input data and output data at two different rates.

- FIFO buffer can be useful in application when data are transferred asynchronously. It fills up data as they come in and gives them away in the same order when the data are needed.



(4X4 buffer FIFO)

It consists of four 4 bit registers R_I, $I = 1, 2, 3, 4$ and a Control Register with flipflops f_i, $i = 1, 2, 3, 4$ one for each register. The FIFO can store four words of four bits each. The number of bits per word can be increased by increasing the number of bits in each register and the no of words can be increased by increasing the no of registers.

A flip flop f_i in the control register that is set to 1 indicates that a 4 bit data word is stored in the corresponding register R_I. A 0 in f_i indicates that the corresponding register does not

contain valid data. The control register directs the movement of data through the registers. Whenever the F_i bit of the control register is set ($F_i = 1$) and the F_{i+1} bit is reset ($F_{i+1} = 1$), a clock is generated causing register $R(I+1)$ to accept the data from register RI. Data are inserted into the buffer provided that the input ready signal is enabled. This occurs when the first control flip-flop f_1 is reset, indicating that register R1 is empty. Data are loaded from the input lines by enabling the clock in R1 through the insert control line.

The data falling through the register stack up at the o/p end. The output ready control line is enabled when the last control flip-flop f_4 is set, indicating that there are valid data in the o/p register R4. The o/p data from R4 are accepted by a distination unit, which then enables the delete control signal. This reset f_4, causing o/p ready to disable, indicating the data on the o/p are no longer valid. Only after the delete signal goes back to 0 can the data from R3 move into R4. If FIFO is empty, there will be no data in R_3 and f_4 will remain in reset state.

MODES OF TRANSFER:

Binary information received from an external device is usually stored in memory. Information transferred from the central computer into an external device originates in the memory unit. The CPU merely executes the I/O instructions, but ultimate source or destination is the memory unit. Data transfer b/w the central computer and I/O device may be handled in variety of modes. Some modes use the CPU as an intermediate path; other transfers the data directly to and from the memory unit. Data transfer to and from peripherals may be handled in one of three possible modes:

1. Programmed I/O.
2. Interrupt - initiated I/O
3. Direct Memory access (DMA)