EC-252: COMPUTER ARCHITECTURE AND MICROPROCESSORS

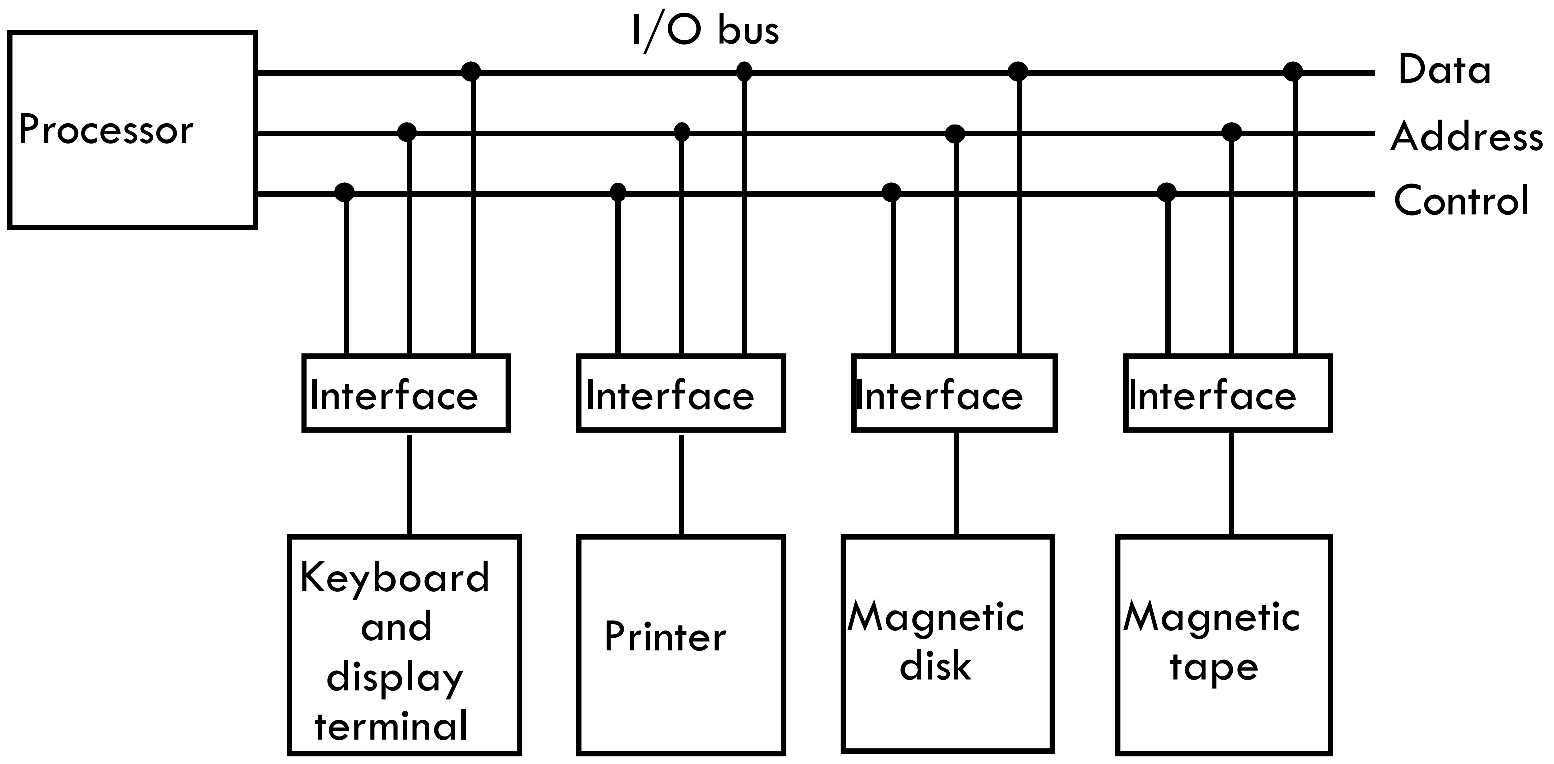
Vaskar Raychoudhury

Indian Institute of Technology Roorkee

I/O Bus and Interface Modules

- Each peripheral has an interface module associated with it along with its own controller
- Interface
  - Decodes the device address (device code)
  - Decodes the commands (operation)
  - Provides signals for the peripheral controller
  - Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory

I/O Bus and Interface Modules



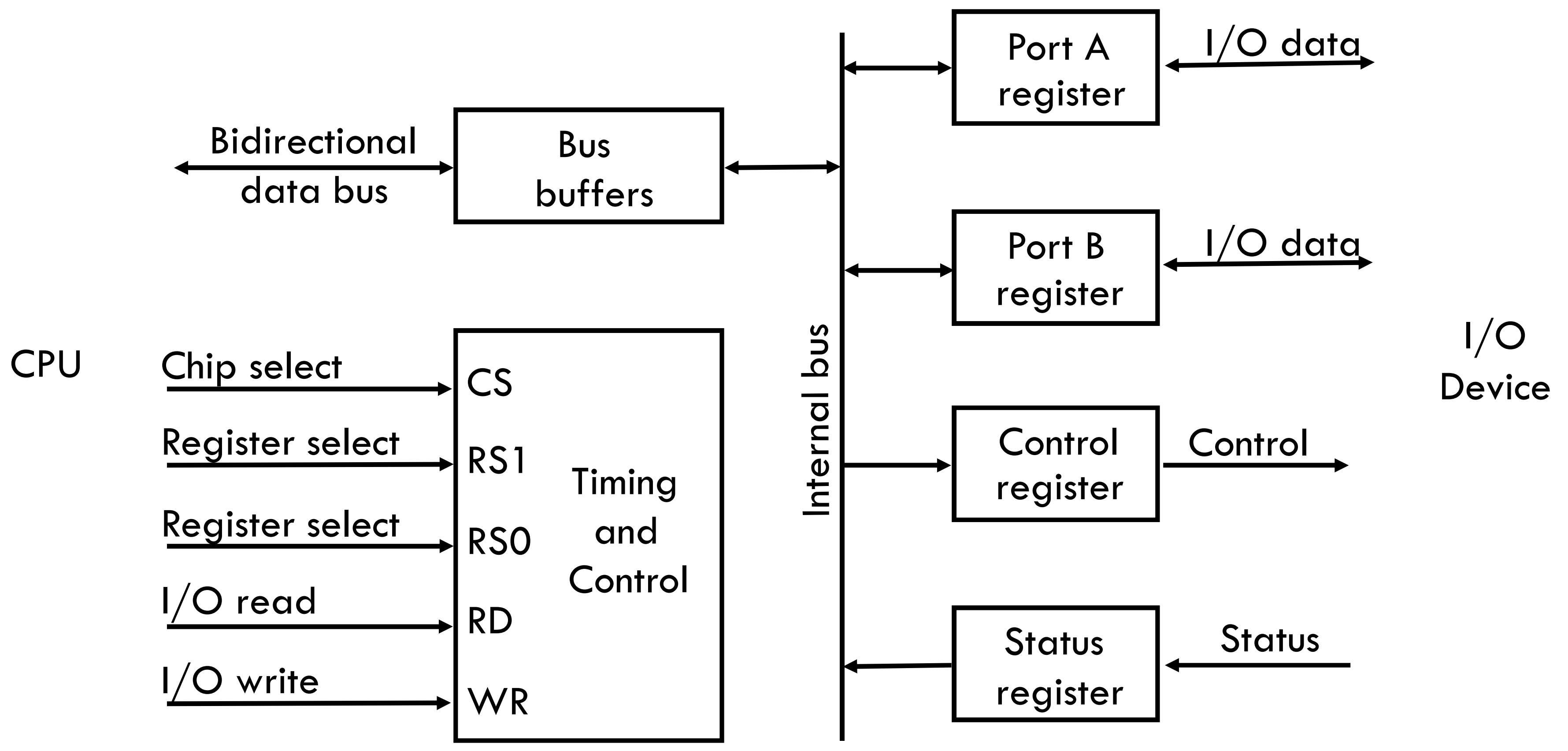
Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory-mapped I/O

- A single set of read/write control lines (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
  - reduces memory address range available
- No specific input or output instruction
  - The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

Example of I/O Interface Unit



Example of I/O Interface Unit

CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedence
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Programmable Interface

- Information in each port can be assigned a meaning depending on the mode of operation of the I/O device
  - Port A = Data; Port B = Command; Port C = Status
- CPU initializes (loads) each port by transferring a byte to the Control Register
  - Allows CPU to define the mode of operation of each port
  - *Programmable Port*: By changing the bits in the control register, it is possible to change the interface characteristics

Asynchronous Data Transfer

- Synchronous and Asynchronous Operations
  - Synchronous
    - All devices derive the timing information from common clock line
  - Asynchronous
    - No common clock
- Asynchronous Data Transfer
  - Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted

Asynchronous Data Transfer Methods

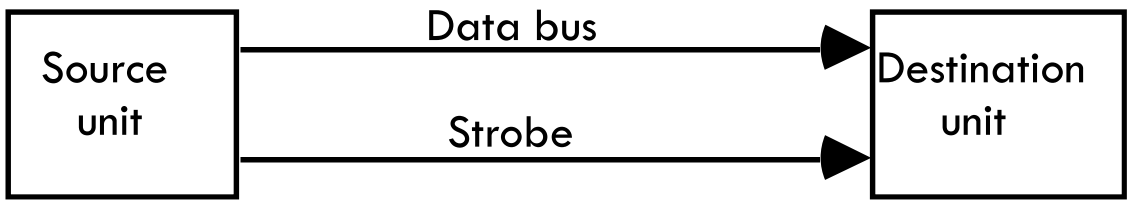
- Strobe pulse
  - A strobe pulse is supplied by one unit to indicate the other unit when the transfer has to occur
- Handshaking
  - A control signal is accompanied with each data being transmitted to indicate the presence of data
  - The receiving unit responds with another control signal to acknowledge receipt of the data

Strobe Control

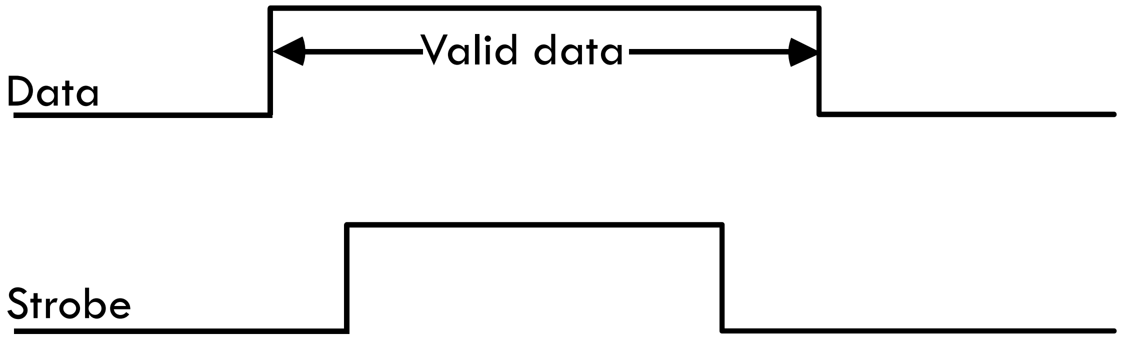
- Employs a single control line to time each transfer
- The strobe may be activated by either the source or the destination unit

Source-initiated Strobe for Data Transfer

Block Diagram

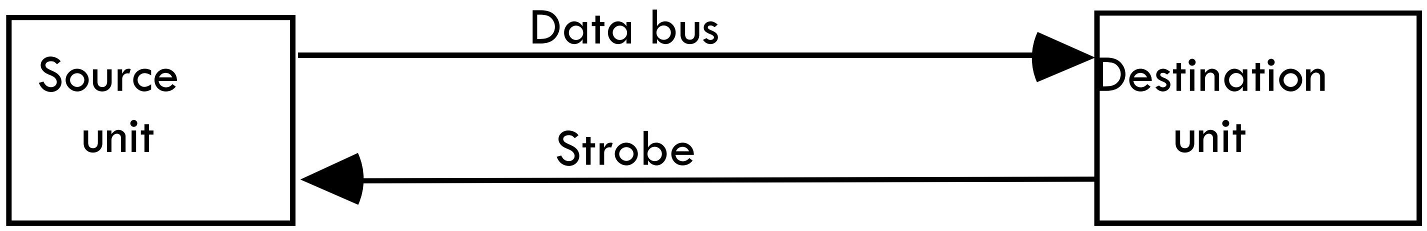


Timing Diagram

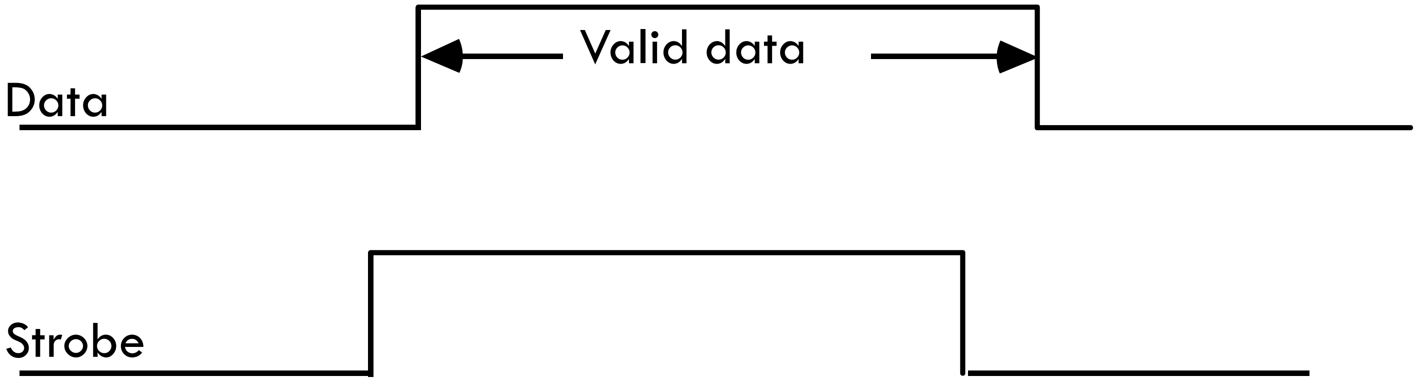


Destination-initiated Strobe for Data Transfer

Block Diagram



Timing Diagram



Problems of Strobe Methods

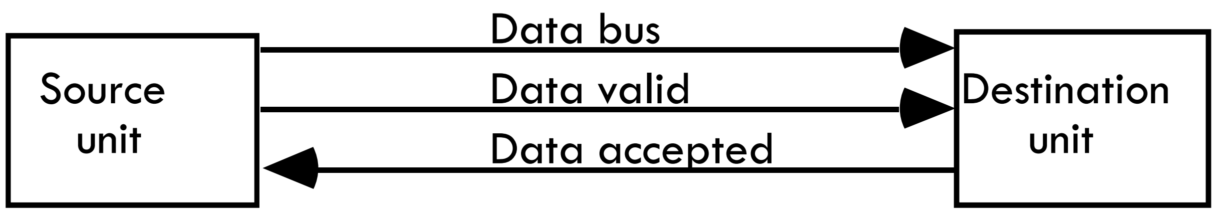
- Source-Initiated
  - The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data
- Destination-Initiated
  - The destination unit that initiates the transfer no way of knowing whether the source has actually placed the data on the bus

Handshaking

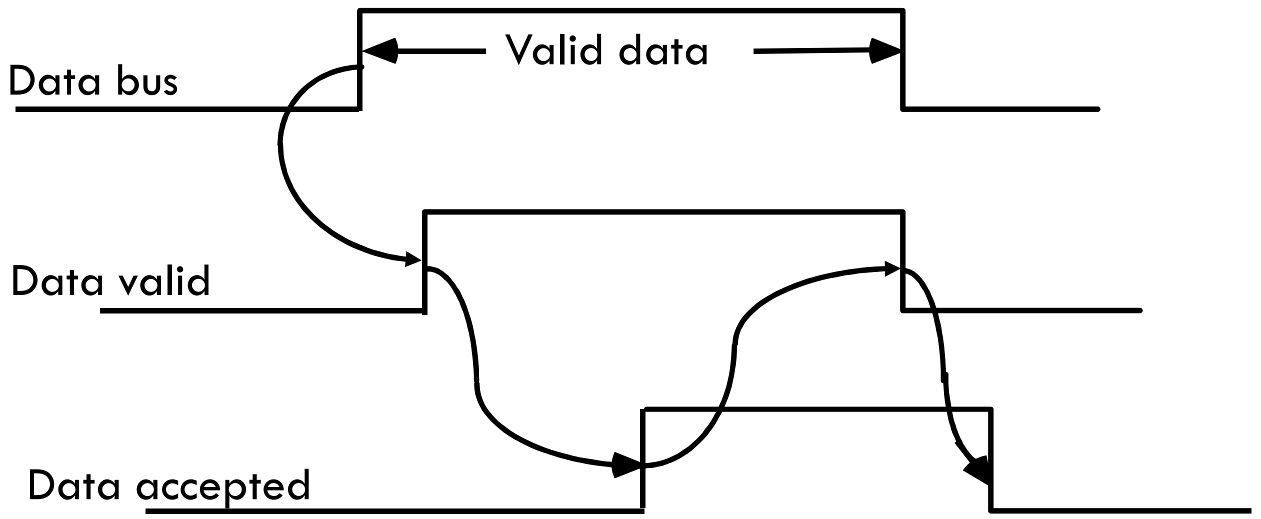
- To solve the strobe related problems
  - the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

Source-initiated Transfer Using Handshaking

Block Diagram

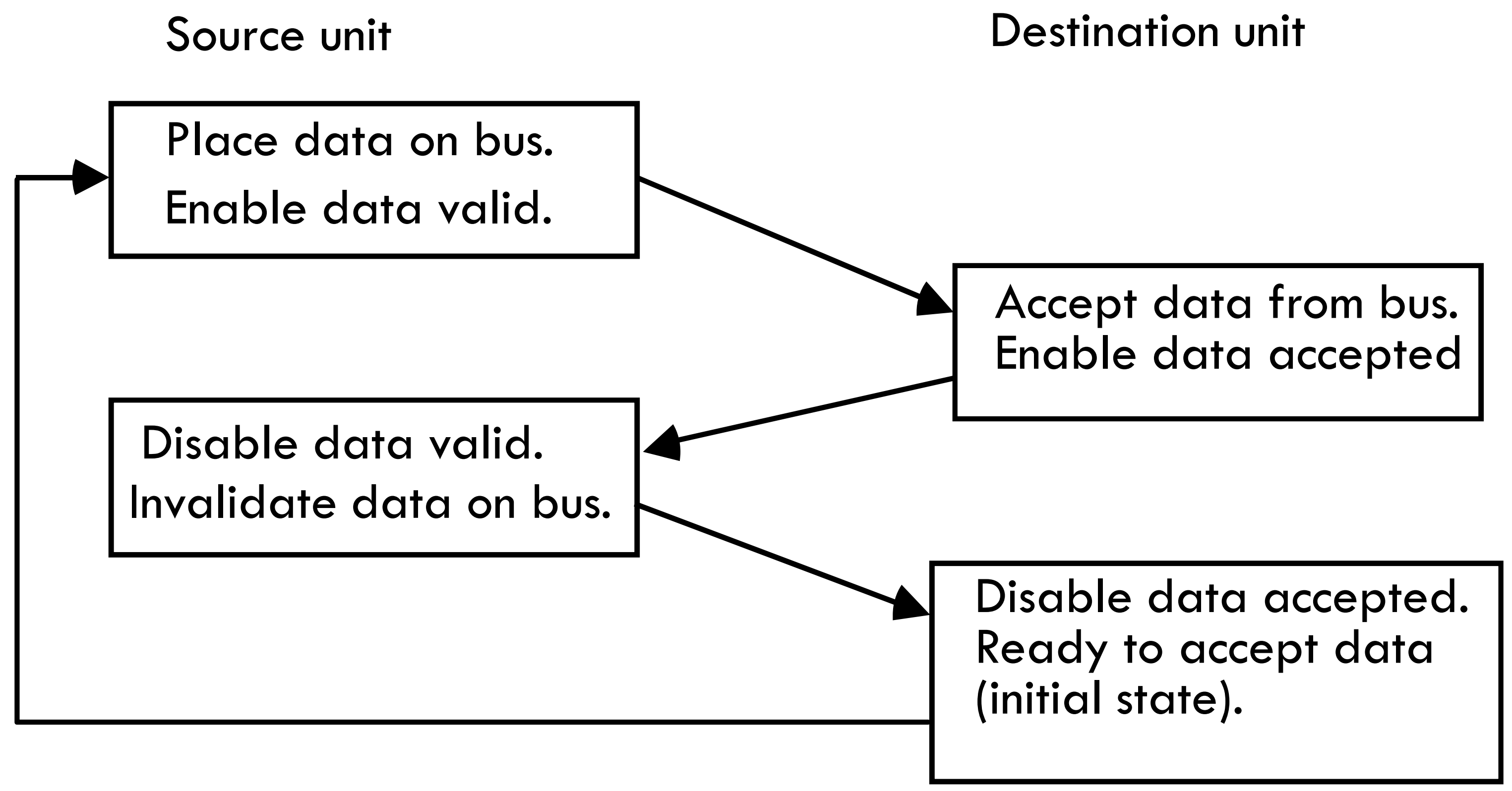


Timing Diagram



Source-initiated Transfer Using Handshaking (2)

Sequence of Events

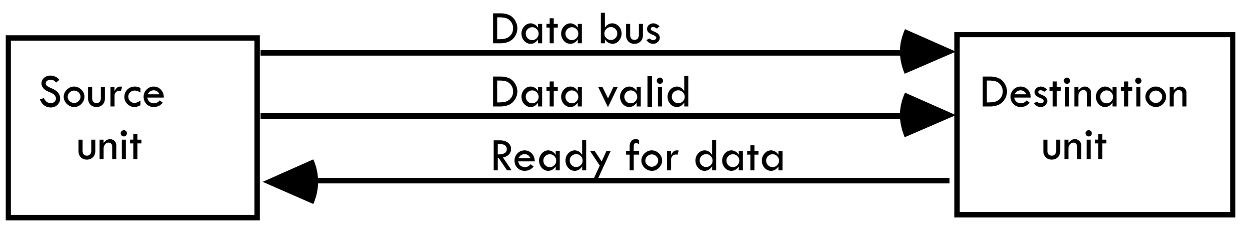


Source-initiated Transfer Using Handshaking (3)

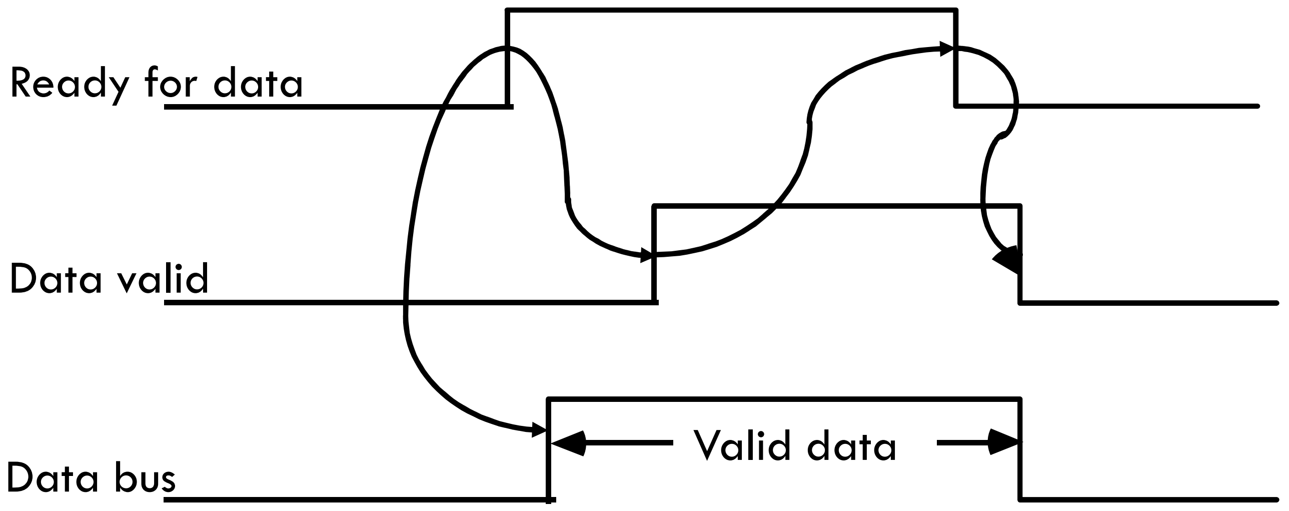
- Source-initiated transfer using handshaking
  - Allows arbitrary delays from one state to the next
  - Permits each unit to respond at its own data transfer rate
  - The rate of transfer is determined by the slower unit

Destination-initiated Transfer Using Handshaking

Block Diagram

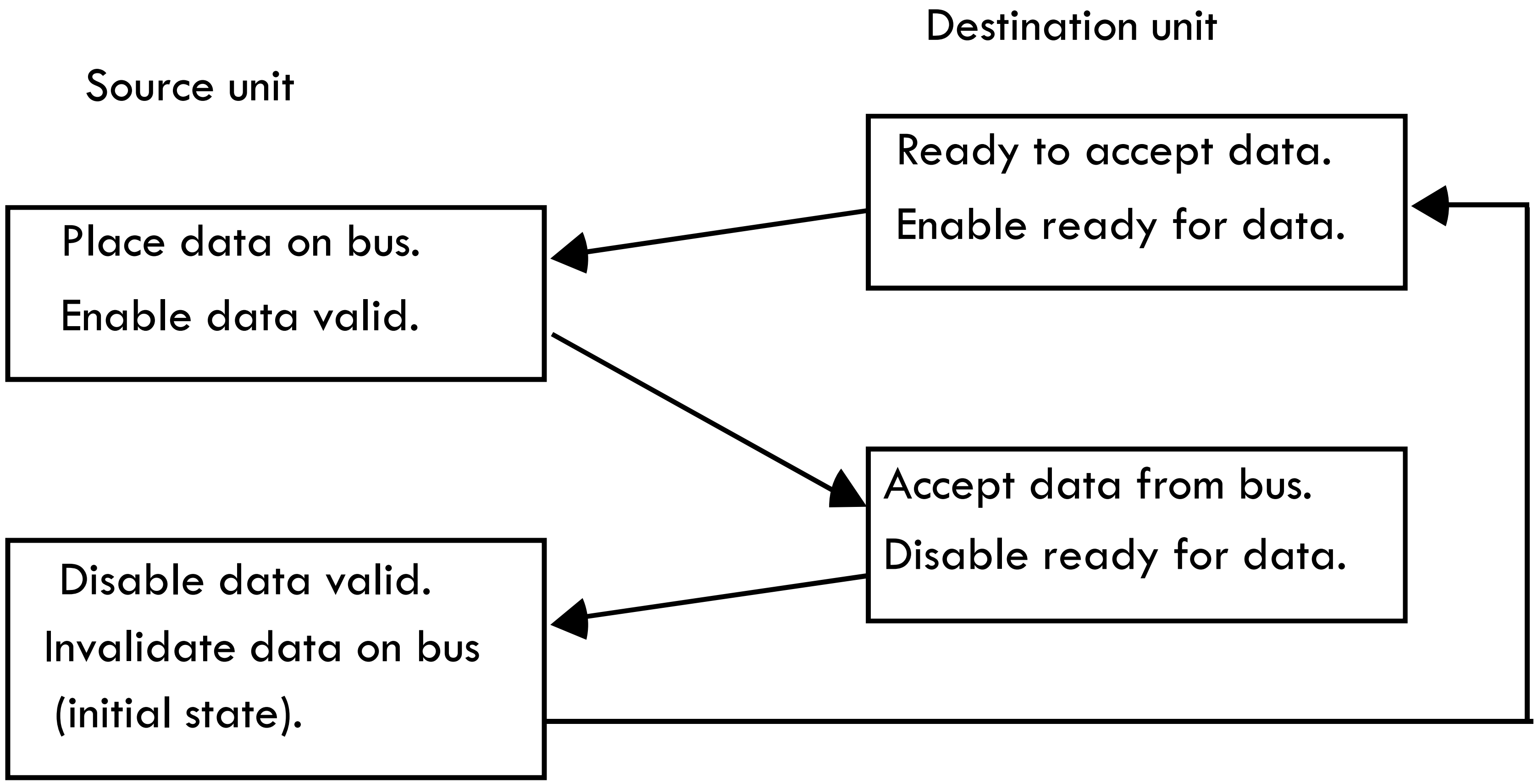


Timing Diagram



Destination-initiated Transfer Using Handshaking (2)

Sequence of Events



Reliability of Handshaking

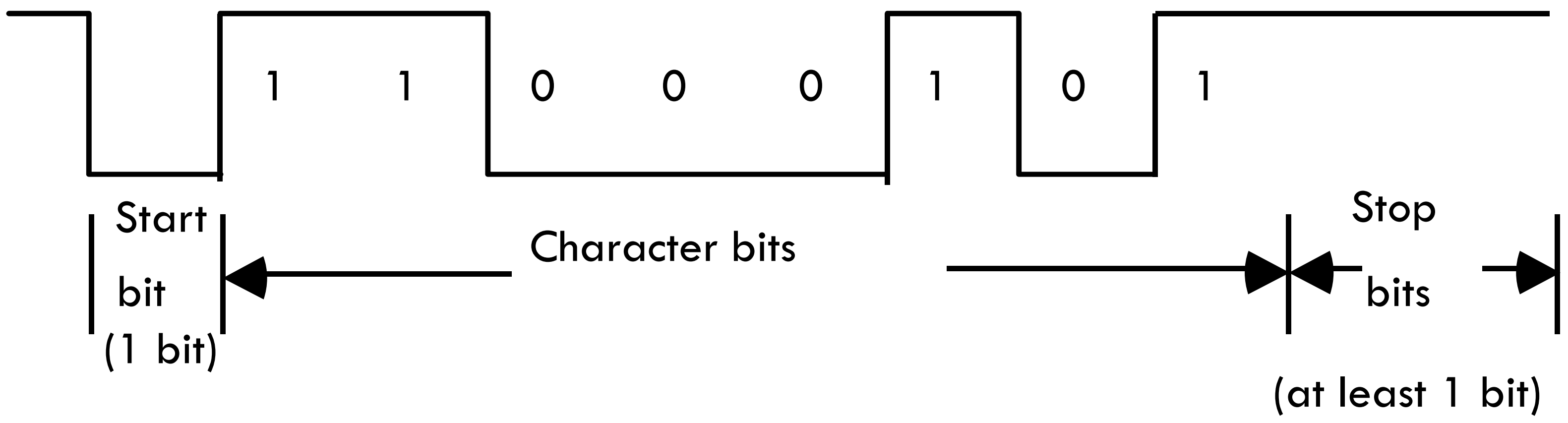
- Handshaking provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units
- If one unit is faulty, data transfer will not be completed
  - Can be detected by means of a *timeout* mechanism

Asynchronous Data Transfer Types

- Four Different Types of Transfer
  - Asynchronous serial transfer
  - Synchronous serial transfer
  - Asynchronous parallel transfer
  - Synchronous parallel transfer

Asynchronous Serial Transfer

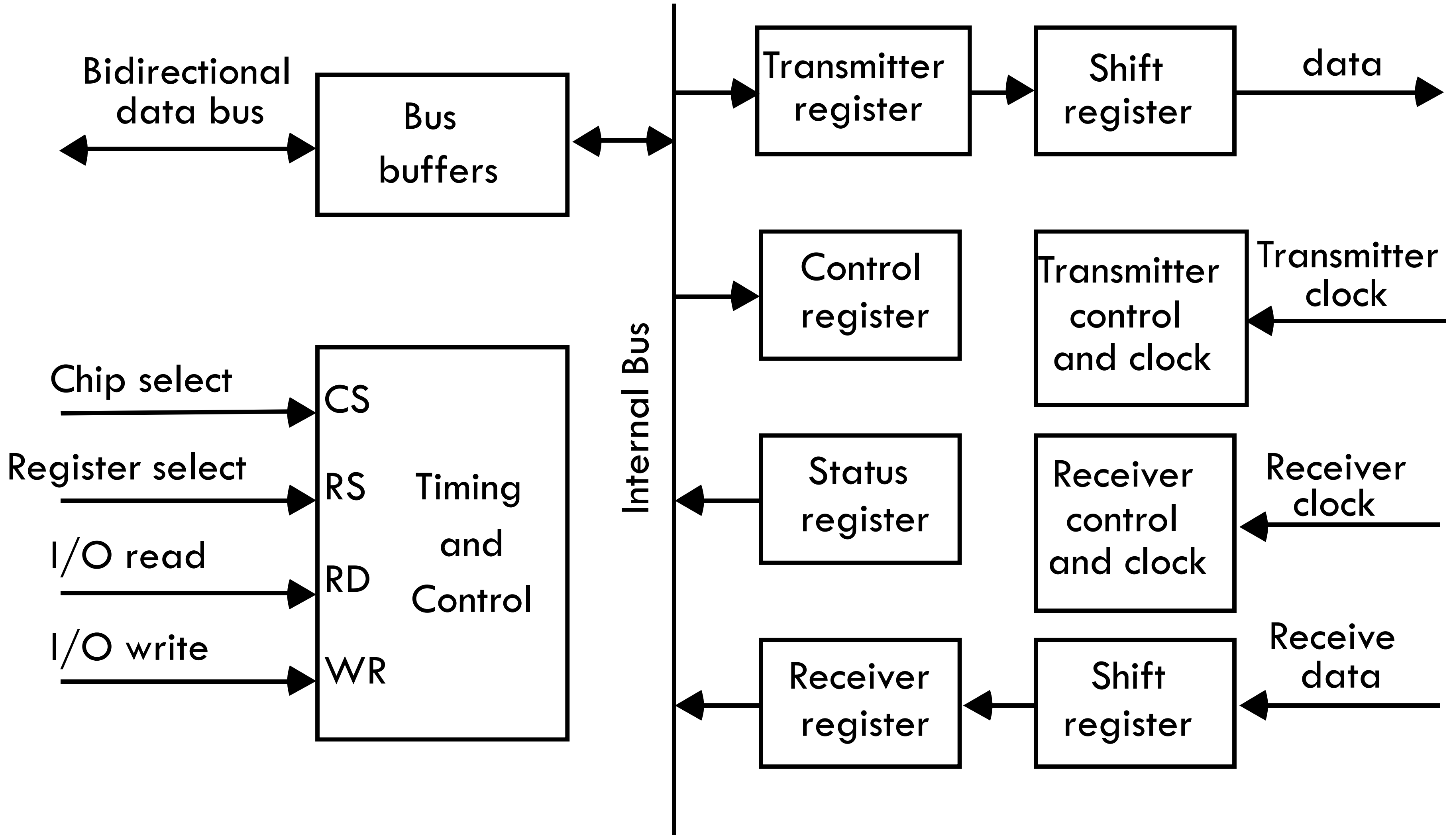
- Asynchronous Serial Transfer
  - Employs special bits which are inserted at both ends of the character code
  - Each character consists of three parts; Start bit; Data bits; Stop bits



Asynchronous Serial Transfer

- A character can be detected by the receiver from the knowledge of 4 rules
  - When data are not being sent, the line is kept in the 1-state (idle state)
  - The initiation of a character transmission is detected by a *Start Bit*, which is always a 0
  - The character bits always follow the *Start Bit*
  - After the last character, a *Stop Bit* is detected when the line returns to the 1-state for at least 1 bit time
- The receiver knows in advance the transfer rate of the bits and the no. of information bits to expect

Universal Asynchronous Receiver-transmitter (UART)



Universal Asynchronous Receiver-transmitter (UART)

CS	RS	Oper.	Register selected
0	x	x	None
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

Universal Asynchronous Receiver-transmitter (UART)

- Transmitter Register
  - Accepts a data byte(from CPU) through the data bus
  - Transferred to a shift register for serial transmission
- Receiver
  - Receives serial information into another shift register
  - Complete data byte is sent to the receiver register
- Status Register Bits
  - Used for I/O flags and for recording errors
- Control Register Bits
  - Define baud rate, no. of bits in each character, whether to generate and check parity, and no. of stop bits