DIRECT MEMORY ACCESS
and Computer Buses

Tutorial 6
Logic Design
Tutorial Contents

1. Concepts
2. Measurement
3. Operation
Concepts

- **Level 0**: Registers in CPU
- **Level 1**: Cache (SRAM)
- **Level 2**: Main memory (DRAM)
- **Level 3**: Disk storage
- **Level 4**: Tape units

Increases in Capacity and Access time:

Increases in cost per bit:

Capacity
A bus is a collection of wires and connectors through which the data is transmitted.

- Data bus: transfers actual data.
- Address bus: transfers information about data and where it should go.

Bus = address bus + data bus

- Data bus: transfers actual data.
- Address bus: transfers information about data and where it should go.
• **Bus protocol**: rules determining the format and transmission of data through bus.

• **Parallel bus**: data is transmitted in parallel.
  – Advantage: fast
  – Disadvantage: high cost for long distance transmission, interference between lines at high frequency.

• **Serial bus**: data is transmitted in serial.
  – Advantage: low cost for long distance transmission, no interference.
  – Disadvantage: slow

• **Bus master**: The device controls bus. Other devices are slaves.
Measurement

- Bus width: indicates the number of wires in the bus for transferring data.
- Bus bandwidth: refers to the total amount of data that can theoretically be transferred on the bus in a given unit of time.
## Width and Bandwidth of Some Typical Buses

<table>
<thead>
<tr>
<th>Bus</th>
<th>Width (bit)</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit ISA</td>
<td>16</td>
<td>15.9</td>
</tr>
<tr>
<td>EISA</td>
<td>32</td>
<td>31.8</td>
</tr>
<tr>
<td>VLB</td>
<td>32</td>
<td>127.2</td>
</tr>
<tr>
<td>PCI</td>
<td>32</td>
<td>127.2</td>
</tr>
<tr>
<td>64-bit PCI 2.1 (66 MHz)</td>
<td>64</td>
<td>508.6</td>
</tr>
<tr>
<td>AGP 8x</td>
<td>32</td>
<td>2,133</td>
</tr>
<tr>
<td>USB 2</td>
<td>1</td>
<td>Slow-Speed: 1.5 Mbit/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Full-Speed: 12 Mbit/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hi-Speed: 480 Mbit/s</td>
</tr>
<tr>
<td>Firewire 400</td>
<td>1</td>
<td>400 Mbit/s</td>
</tr>
<tr>
<td>PCI-Express 16x version 2</td>
<td>16</td>
<td>8,000</td>
</tr>
</tbody>
</table>
A bus can be classified as one of two types: synchronous and asynchronous.

**Synchronous bus**: There is a common clock that synchronizes bus operations.

**Asynchronous bus**: There is no common clock. Bus master and slaves have to “handshake” during transmission process.
A simple bus

Wires:
- Uni-directional or bi-directional
- One line may represent multiple wires

Bus
- Set of wires with a single function
  - Address bus, data bus
- Or, entire collection of wires
  - Address, data and control
  - Associated protocol: rules for communication
Timing Diagrams

• Most common method for describing a communication protocol
• Protocol may have subprotocols
  – Called bus cycle, e.g., read and write
  – Each may be several clock cycles
• Read example
  – $rd'/wr$ set low, address placed on $addr$ for at least $t_{\text{setup}}$ time before $enable$ asserted, enable triggers memory to place data on $data$ wires by time $t_{\text{read}}$
Bridge-based bus architectures

- System includes a lot of buses which are segregated by bridges.
- Advantage: buses can simultaneously operate.
- Intel architecture:

```
   Central Processing Unit
      |
      v
   Northbridge
      /|
     / | RAM (Memory)
      v  v
     v  v
     v  v
      v  v
  PCI Bus
      |
      v
  Southbridge
      /|
     / | Real Time Clock
      v  v
     v  v
      v  v
     v  v
      v  v
  APM (Power Management)
      |
      v
  USB
      |
      v
  Other Devices
```
Internal Communication Methodologies

- Programmed I/O (polling)
- Interrupt-drive I/O
- Direct Memory Access (DMA)
**Programmed I/O (polling)**

- CPU polls each device to see if it needs servicing.
- Drawback: The CPU wastes time for polling devices (busy-wait.)
Interrupt-Drive I/O (PIO)

- Device requests service through a special interrupt request line that goes directly to the CPU.
- No busy-wait. More efficient than PIO.
Direct Memory Access (DMA)

- Devices transfer data directly to and from memory bypasses the CPU.
- Very efficient mode. CPU is free to do other operations.

Figure 8-10
DMA transfer from disk to memory bypasses the CPU.
Figure 8-11
DMA flowchart for a disk transfer.
Direct Memory Access (DMA)

• Why is DMA an improvement over CPU programmed I/O?

• DMA is a mechanism that provides a device controller the ability to transfer data directly to or from the memory without involving the processor.

• This allows the CPU to perform arithmetic and other instructions while the DMA is going on in parallel.
Direct Memory Access (DMA)

• When would DMA transfer be a poor choice?
• DMA is not useful when the amount of data to transferred between memory and the I/O device is negligible.
• In this case the overhead of setting up the DMA transfer would outweigh the benefits of direct data transfer without the interference of the processor.
## Performance Example

### Programmed I/O modes in the ATA interface

<table>
<thead>
<tr>
<th>Mode</th>
<th>Maximum transfer rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode 0</td>
<td>3.3</td>
</tr>
<tr>
<td>mode 1</td>
<td>5.2</td>
</tr>
<tr>
<td>mode 2</td>
<td>8.3</td>
</tr>
<tr>
<td>mode 3</td>
<td>11.1</td>
</tr>
<tr>
<td>mode 4</td>
<td>16.7</td>
</tr>
<tr>
<td>mode 5</td>
<td>20</td>
</tr>
<tr>
<td>mode 6</td>
<td>25</td>
</tr>
</tbody>
</table>
## Performance Example

### DMA modes in the ATA interface

<table>
<thead>
<tr>
<th>Modes</th>
<th>Maximum transfer rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-word DMA 1</td>
<td>13.3</td>
</tr>
<tr>
<td>Multi-word DMA 2</td>
<td>16.6</td>
</tr>
<tr>
<td>Ultra DMA 0</td>
<td>16.7</td>
</tr>
<tr>
<td>Ultra DMA 1</td>
<td>25.0</td>
</tr>
<tr>
<td>Ultra DMA 2</td>
<td>33.3</td>
</tr>
<tr>
<td>Ultra DMA 3</td>
<td>44.4</td>
</tr>
<tr>
<td>Ultra DMA 4</td>
<td>66.7</td>
</tr>
<tr>
<td>Ultra DMA 5</td>
<td>100</td>
</tr>
<tr>
<td>Ultra DMA 6</td>
<td>133</td>
</tr>
</tbody>
</table>
Direct Memory Access (DMA)
In conclusion

• Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load.

• Q : How does the CPU interface with the device to coordinate the transfer?

• A : All devices have special hardware controllers. The device controllers have registers, counters and buffers to store arguments and results. The CPU first loads them, and then the device controller takes over.
In conclusion

- Q: How does the CPU know when the memory operations are complete?
- A: The device controller sends an interrupt to the CPU