**I/O Organization**

* Recall – circuit hardware
  + Propagation delay – delay between change in input & change in output
  + SR latch – most basic memory component
    - SR = 00 → no change
    - SR = 10 → set (Q = 1)
    - SR = 01 → reset (Q = 0)
    - SR = 11 → undefined
  + Gated latch – change in input only propagates to output during a clock pulse
  + Gated D latch
    - Clk = 0 → no change
    - D = 0 → reset (Q = 0)
    - D = 1 → set (Q = 1)
  + Flip-flop – output changes at an edge (positive/negative) of the clock signal
  + DFF
    - D = 0 → Q(t + 1) = 0
    - D = 1 → Q(t + 1) = 1
  + TFF
    - T = 0 → Q(t + 1) = Q(t)
    - T = 1 → Q(t + 1) = !Q(t)
  + JKFF
    - JK = 00 → Q(t)
    - JK = 01 → 0
    - JK = 10 → 1
    - JK = 11 → !Q(t)
  + Setup time – interval for which the input should be stable before the clock edge
  + Hold time – interval for which the input should be stable after the clock edge
  + Multiplexer – n select inputs, select one signal from 2n data signals
    - E.g. 4 data inputs, use 2 select inputs to choose 1 of the 4 signals to pass to output
  + Decoder – n inputs, 2n outputs (max)
    - E.g. 2 inputs, decode into 4 outputs with only one active (i.e. one-hot)
* **The bus**
  + The bus carries 3 signals – address, data, control
  + I/O devices connect to the bus via I/O interfaces
  + Bus address signal → address decoder
  + Bus control signal ↔ control circuitry
  + Bus data signal ↔ data, status, control registers
* Address decoder
  + Determines when a device should respond to a request from the processor
  + Produces a device enable signal when an address in the device’s range in provided on the address bus
  + Given a memory map, one can determine:
    - Which address signals uniquely identify a device
    - Which address signals are used by a device
    - The size of address range used by device
    - E.g. device start address = 0x8020
      * Ending address = 0x803F
      * Address range = 1000 0000 001X XXXX
      * Device enable (identifies the device)

= 1000 0000 001 = b15\*(!b14)\*…\*(!b6)\*b5

* + - * Address signals used

= X XXXX = b4, b3, … b0

* + - * Address range size

= 0x20 addresses = 25 bytes

* Bus operation
  + Bus protocol – set of rules that governs how the bus is used by devices
  + Bus control lines contain:
    - Read/write command
    - Data size of operation
    - Information about timing
  + Master – device initiating data transfers by issuing r/w commands to the bus
  + Slave – device addressed by the master (i.e. the device I/O interface)
  + Synchronous bus – all devices get timing information from the bus clock
    - E.g. read command:
    - t0 – master places device address → address lines & read command → control lines
    - t1 – slave returns data on data lines
    - t2 – master reads data on data lines and stores into registers; slave removes data from data lines
    - t1 – t0 = clock high pulse
      * Must be > propagation delay over bus + time for slave to decode address & control signals
    - t2 – t1 = clock low pulse
      * Must be > propagation delay over bus + setup time of registers
    - Propagation delay also exists between the clock, master & slave
      * E.g. delay between when master asserts signal & when slave sees it, etc.
    - With one clock cycle, t2 – t0 must accommodate the longest possible delay on the bus & the slowest device
    - Can use multiple clock cycles
      * Set ready signal = 1 when slave places data on data lines, indicates master can read data
      * Set read signal = 0 when slave removes data from data lines
  + Asynchronous bus – i.e. handshake protocol
    - Automatically accommodates for bus & device delays by not using a clock
    - E.g. read operation:
    - t0 – master places address & command on bus lines
    - t1 – master-ready line = 1
      * t1 – t0 delay ensures that ready signal arrives at slave after address & command (bus skew)
    - t2 – slave places data on bus, slave-ready line = 1
    - t3 – slave-ready = 1 arrives at master; master reads data; master-ready = 0
    - t4 – master de-asserts address & command
      * t4 – t3 ensures ready signal arrives before address & command are removed (bus skew)
    - t5 – master-ready = 0 arrives at slave; slave de-asserts data & slave-ready = 0
    - E.g. write operation:
    - t0 – master places address, command & data on bus
    - t1 – master-ready = 1
    - t2 – slave reads data from bus → register; slave-ready = 1
    - t3 – slave-ready arrives at master; master-ready = 0
    - t4 – master de-asserts address, command & data
    - t5 – master-ready arrives at slave; slave-ready = 0
* Bus arbitration
  + Multiple master devices may need access to a given slave device at the same time
  + Centralized arbitration – uses central arbiter circuit to process bus requests
    - Each master sends request (BR = 1) to arbiter
    - If unused, device is granted to master (BG = 1)
    - If used, master waits until device becomes unused & all other higher-priority requests have been granted (BR = 1 still); then arbiter grants access
    - When master is finished with device, BR = BG = 0
* Interface circuits
  + Contains:
    - Register for temporary storage
    - Status register (accessed by processor)
    - Control register (determines behaviour of interface)
    - Address decoder
    - Timing signal generator
    - Format converter (between processor & I/O)
  + Parallel port interface
    - Multiple bits are transferred simultaneously
    - High-speed, but costly
  + Serial port interface
    - Uses a single wire to transfer one bit at a time (uses shift register)
    - Slow, but cheap
    - Synchronous transmission
      * Receiver generates clock that is synchronized with transmitter clock
      * Enables high-speed transfers
    - Asynchronous transmission – i.e. start-stop transmission
      * When idle, bit = 1
      * Transmit start bit = 0 alerts the receiver that transmission is starting
      * Transmit 8 bits
      * Transmit 1 or 2 stop bits = 1
* Interconnection standards
  + Bridge – interconnects 2 different bus architectures
  + Universal Serial Bus (USB)
    - Plug-and-play – its presence is detected automatically by the system and accommodated for
    - Tree structure – every node/hub = transfer point between host computer & I/O
    - Root hub connects entire tree to host computer
    - Leaf nodes are I/O devices
    - Messages from root are broadcast to all child hubs
      * Only the addressed device receives the message
    - Addressed device transmits its response upstream
      * Only hubs listen to upstream messages (not I/O devices)
    - Does not require arbitration since root hub is bus master
  + Peripheral Component Interconnect Bus (PCI)
    - Plug-and-play
    - Supports 3 independent address spaces:
      * Memory address space (memory-mapped I/O)
      * I/O address space
      * Configuration address space
    - PCI bus connected to processor via bridge
      * Has a port to the main memory
    - Initiator = bus master
    - Target = bus slave
    - Transaction = transmission of a starting address & multiple words of data (i.e. burst mode)
      * Phase = transmission of a single word within a transaction
      * Length of transaction is indicated by the length of the FRAME signal
  + Small Computer System Interface Bus (SCSI)
    - SCSI devices are not part of the processor’s address space
    - E.g. read command
      * Processor assembles command in memory
      * Processor requests SCSI controller for bus access
      * Controller grants access (after arbitration), sends read command to disk controller
      * Disk controller requires delay to read from disk; indicates to SCSI controller their connection will be temporarily suspended
        + SCSI can be used by other devices
      * Disk controller moves disk drive to head of sector; reads data and stores in data buffer; requests for control of bus
      * Disk ↔ SCSI controllers connection re-established; data buffer contents sent to SCSI controller; connection suspended again
      * Process is repeated for second sector of disk
      * SCSI controller sends data to memory, sends interrupt to processor