Sequential Circuits, Bus

Online Ch 20.1-3 [Sta10]
Circuits with memory
Flip-Flop
S-R Latch
Registers, Counters

Ch 3 [Sta10]
What moves on Bus?
Bus characteristics
PCI-bus

Circuits with memory
Flip-Flop
S-R Latch
Registers, Counters

Sequential Circuits, Bus

Sequential Circuits, Bus

Flip-Flop (kiikku)

- William Eccles & F.W. Jordan
  - with vacuum tubes, 1919
- 2 states for Q (0 or 1, true or false)
  - 1-bit memory
  - Maintains state when input absent
- 2 outputs
  - complement values
  - both always available on different pins
- Need to be able to change the state (Q)

S-R Flip-Flop or S-R Latch (salpa)

Usually both 0
S = “SET” = “Write 1” = “set S=1 for a short time”
R = “RESET” = “Write 0” = “set R=1 for a short time”

Use NOR gates
nor (0, 0) = 1
nor (0, 1) = 0
nor (1, 0) = 0
nor (1, 1) = 0

Clocks Flip-Flops

- State change can only when clock is 1
- more control on state changes

Clocks S-R Flip-Flops

- D Flip-Flop
  - only one input D
  - D = 1 and CLOCK write 1
  - D = 0 and CLOCK write 0
- J-K Flip-Flop
  - Toggle Q when J=K=1

Registers

- Parallel registers
  - read/write
  - CPU user registers
  - additional internal registers
- Shift Registers
  - shifts data 1 bit to the right
  - serial to parallel?
  - ALU ops?
  - rotate?
Counters
- Add 1 to stored counter value
- Counter
  - parallel register plus increment circuits
- Ripple counter (aalto, viive)
  - asynchronous
  - increment least significant bit,
  - and handle ‘carry’ bit
  - as far as needed
- Synchronous counter
  - modify all counter
  - flip-flops simultaneously
  - faster, more complex,
  - more expensive
  ![Space Time Tradeoff]

Digital Logic Summary
- Boolean algebra
- Gates – not, nand, xor, and, or
- Circuits
  - Presentation: Boolean equations, Truth tables, Graphical Symbols
  - Simplification with Karnaugh Maps
  - Combination Circuits – output depends on input only
    - Set inputs, wait, output ready – no dynamic state memory
    - ROM
  - Sequential Circuits – output depends also on internal state
    - Flip-Flops, registers, counters, memory
  - Implement Computer
    - apply combination and sequential circuits smartly

Bus (Väylä)
- For communication with and between devices
- Broadcast (yleislähetys) - most common
- React to messages/signals to itself only
- Each device has its own control and status information
  - Device driver (OS) moves control data to device controller’s registers
    - memory address, device address, how much, direction
  - Device driver reads the status from the controller’s status register
    - Ready? Operation successful? …

Bus structure
- Control lines (Ohjausväylä, ~ johtimet)
  - Control and timing information
    - Operations: like memory read, memory write, I/O read
    - Interrupt
    - Clock
- Address lines (Osoiteväylä)
  - Source and destination ids
    - Memory address, device address (module, port)
    - For transfer source and destination
  - Width (number of parallel lines) determines directly
    - addressable memory address space (osoiteavaruuden koko)
    - For example: 32 b ⇒ 4 GB

Bus structure
- Data lines (dataväylä)
  - All processing information:
    - Instructions
    - Data
    - DMA-transfer contents
  - Width determines the maximum number of bits
    - that can be transferred at the same time
    - For example 38b wide line allows 32 bits
      data plus 6 Hamming-coded parity bits
Lecture 2: Digi Logic & Bus

What moves on the bus?

- Processor
- Memory
- I/O Controller

Bus = Bottleneck?

- von Neumann architecture
  - Instructions and data both in main memory
  - All memory content referred using address
  - Sequentially ordered instructions executed sequentially
    - unless order changed explicitly (jumps, branches)

Bus characteristics

- Width
  - 50 – 100 lines (johdin)
  - motherboard, cable, connectors
- Bus type
  - Dedicated, non-multiplexed (dedikoitu)
  - Address and data – separate lines
  - Time multiplexed (aikavuorottelu)
    - Address and data share lines
    - Address valid / data valid -line
- Arbitration (vuoron varaus)
  - Centralized
    - One bus controller, arbiter (väyläohjain)
  - Distributed
    - Controllers have necessary logic

Synchronous timing

- Based on clock
  - Control line has clock pulse (cycle 1-0)
  - All devices "hear" the same pulse
- Event takes one cycle (commonly)
  - Start at the beginning of the cycle (leading edge)
  - For example, reading data takes one cycle
- All devices in the bus work at the same pace
  - Slowest determines the speed of all
  - Each device knows the speed of the others
  - Each device knows, when the other is ready for next event
  - "Do this during the next cycle"
  - Device can count on the other one to do it!

Asynchronous timing

- Devices can use arbitrary speeds (variation allowed)
- Processing time depends on the device
- Device can determine, when the other one is ready
  - How long is the event going to last to perform?
- Synchronization using a special signal
  - Send synchronization signal, when work done and ready
    - Address and data on bus → send signal "write" (for example: change "write"-line to 1)
  - Data stored to memory → send signal "ack"
- Time of the next event depends on signals
  - "Do this when you have time, inform me when ready"
  - Wait until signal that "this" is done
**Timing diagrams (ajoituskaavio)**

- See Appendix 3a [Sta10, Ch 3]

**Synchronous Timing**

Initiator CPU (for example)

**Asynchronous timing - Read**

**Asynchronous timing - Write**

**Bus Events (väylätapahtumia)**

**Bus Configuration**

- All devices on one bus?
  - All must use the same technique
  - Long bus \(\rightarrow\) large propagation delay (etenemisviive)
  - Combined data rates of the devices may exceed the capacity of the bus
  - Collisions on the arbitration, extra wait
  - Synchronous? \(\Rightarrow\) slowest determines the speed of all

- Bus hierarchy
  - Isolate independent traffic from each other
  - Maximize the most important transfer pace
  - CPU \(\Rightarrow\) MEM
  - I/O can manage with lower speed
Bus Hierarchy
Typical Pentium 4

Computer Organization II

PCI-bus
[Sta10, Ch 3.5]

http://www.soe.ucsc.edu/classes/cmpe003/Spring02/motherboard.gif

PCI: Peripheral Component Interconnect
- 49 mandatory (+51 optional) signal lines
- Address data: 32b mandatory (optional allows 64b)
- Other signals: 17 mandatory (+ 19 optional)
- Centralized arbiter (keskitetty väylän varaus)
- Synchronous timing (synkroninen tahdistus)
  - own 33 or 66 MHz clock (PCI-X: 133/156/533 MHz)
  - Transfer rate 133, 266, 532 MB/s (PCI-X: 1 GB/s, 4 GB/s)
- Events on the bus
  - read, write, read block, write block (multiplexed)
- Max 16 devices

49 Mandatory Signal Lines (PCI)
- AD[32]: address or data, multiplexed (aikavuorottelu)
  + 1 parity
- C/BE[4]: bus command tai byte enable, multiplexed
  - For example: 0110/1111 = memory read/ all 4 Bytes
- CLK, RST#: clock, reset
  - 6 for interface control
  - FRAME#, IRDY#, TRDY#, STOP#, IDSEL, DEVSEL#
- 2 for arbitration (väylän varaus)
  - REQ# requires, GNT# granted
  - Dedicated lines for devices
- 2 error reporting pins (lines)
  - PERR#parity, SERR# system

51 Optional Signal Lines (PCI)
- 4 lines for interrupt requests (keskeytyspyyntö)
  - Each device has its own dedicated line(s)
- 2 lines for cache support
  - (on CPU or other devices)
  - snoopy cache
- 32 A/D extra lines
  - 32 mandatory + 32 optional -> 64 bit address/data lines
- 4 additional lines for C/BE bus command/byte enable
- 2 lines to negotiate 64b transfer
- 1 extra parity line
- 5 lines for testing

PCI Transactions
- Bus activity as transactions
  - New bus request for each new transaction
  - (1) Bus reservation
    - Central arbiter
    - send REQ, wait for GNT
  - (2) Bus transaction
    - Initiator or master (device who reserved the bus)
    - Begin by asserting FRAME (reserve of bus)
    - Stop by releasing FRAME (indicate free bus)
Bus arbitration: A and B want bus

- **A** wants bus
- **B** wants bus
- **C** granted bus
- A knows that it has bus and bus is available

Arbitration: A and B want bus

- A starts frame, requests also for next transaction
- B starts frame, no more B req.
- A marks last frame transfer, marks data ready
- B knows that it has bus

PCI Memory Read

- Start trans frame, set addr, set trans type
- Recognise address, find data
- Select bytes, indicate ready to receive
- Ack address, set data, indicate valid data

Discussion?

- Boolean Algebra → Gates → Circuits
- Combination circuits, sequential circuits
- Components for CPU design
  - ROM, adder, multiplexer, encoder/decoder
  - flip-flop, register, shift register, counter
- Bus
  - Structure, components, signals, arbitration
  - PCI bus example
Review Questions

- Main differences between synchronous and asynchronous timing?
- Benefits of bus hierarchy?
- Text book review questions
- Text book support page review questions

http://www.box.net/shared/4597aix1nm