# Clocks, Synchronous & Asynchronous Logic

prepared for Programmable Logic Lessons / 1.3 by ~skmp

Kindly hosted by hackerspace.gr

# Clocks

- Clocks are a way to measure time
- Clock signal is a signal with known time characteristics
  - Duty cycle, Period, jitter etc
  - As hardware works in parallel and continuously, clock signals enable it to **wait**
    - Work is considered complete when an “event” happens on the clock (low → hi, hi → low, both, etc)
  - Sources are usually quartz-piezoelectric mechanical tuned osculations
    - RC oscillators are used as a “cheap-and-dirty” replacement for low frequencies
- It is usual to have many clock signals on a circuit
  - Clocks can be generated from another clock (clock divider/multipliers // PLLs)\
    - This helps to keep the clocks synchronized
    - Phase locked clocks are also quite popular
  - Multiple crystals are possible
    - It is -nearly-impossible- to synchronize clocks from different sources
    - Time/Clock drift between the clocks is quite common
  - Usually circuits are split into “regions” that use different clocks – “clock domains”
    - A clock domain is defined as a group of flip-flops with a common clock
    - Clock domain crossings Is when we need to communicate between different clock domains

# Synchronization building blocks

- Latch
  - "Samples" an analog signal into a clock/discrete signal
- FIFO/Buffers
  - FIFO is like a "queue" of data, first-in-first-out
  - Buffers are used to hold data
    - Kind of single slot FIFO
  - FIFOs are very useful to buffer delays between different blocks

# Pipelines

- Break a long process into smaller stages
  - Process at the same time different commands at diffident stages
  - Much higher utilization of hardware resources
  - For reference, modern cpus have 15-30 stage pipeline, modern gpus even longer
  - Has costs when output must be used as input
    - Pipeline bubbles

# Asynchronous logic

- Instead of clocks uses data avaiable / processing finished
- Much more efficient (Around 50% of the power usage is spent on the clock)
- MUCH harder to design (all tools are optimized for synchronous design)
- (We won't really use this)

# Mixed logic

- Try to use best of both wrolds
- Globally asynchronous locally synchronous
  - Synchronous for smaller parts (simpler design)
  - Asynchronous for “bigger” blocks where it makes sense and doesn't have much overhead
  - Isolate clock domains and match them better

# Thanks !

Next week we'll talk more about cpus :)

Feel free to drop by #hsgr @ freenode

... or the hsgr mailing list

... and use the wiki !

(or, send direct feedback – skmp@emudev.org)