# Emulation

Vector Based Acceleration

* Input through text files (eg, for camera, RGB values)
* Static inputs
* Faster to validate functionality
* Combine individual bits
* Vector based monitoring

Signal Based Acceleration

* Signal changes are monitored by the software

TBA (Transaction Based Acceleration)

* Communication b/w hw and sw (when a change happens in 1 component, communicated to the other)
* TB will monitor HW only when there's a transaction

Embedded Testbench Based Acceleration

* TB is partly synthesizable and is run on HW as well

In-circuit Emulation

* Pure HW
* DUT on HW
* No software TB… testing on actual hardware components (eg peripherals, etc)
* Fastest validation method
* Monitoring not easy

Synthesis = HDL to gate level design (netlist creation)  
  
Setup  Time:  the  amount  of  time  the  data  at  the  synchronous  input  (D)  must  be  stable  before  the active edge of clock. This is so that the data can be stored successfully in the storage device.

Setup violations can be fixed by either slowing down the clock (increase the period) or by decreasing the delay of the data path logic

Hold Time: the amount of time the data at the synchronous input (D) must be stable after the active edge of clock.

Hold violations can be fixed by increasing the delay of the data path or by decreasing the clock uncertainty (skew) if specified in the design.

**Functional Verification**

2 main techniques

1. Logic Simulation
   * Pro: easy, accurate, flexible, and low cost
   * Con: Not fast enough for large designs
   * Executes RTL code serially
     + Can set breakpoints
     + Can stop mid-cycle
     + Can see any signal/memory contents at any time
2. Prototyping on FPGA
   * Pro: Fast, inexpensive
   * Con:
     + Changes to fix design flaws take a long time to implement and may require board wiring changes
     + Traditionally, little debug capability
   * Executes RTL code fully in parallel
     + User employs logic analyzer to examine signals, using pre-defined probes
     + each time the user changes the probes or trigger condition, they have to reset the environment and start again from the beginning.
3. Common Compromise: Use simulation early in verification process when bugs are frequent, and prototyping at the end

**Simulation Acceleration**

* Design is mapped into a hardware accelerator to run much faster
* Testbench (and any behavioral design code) continues to run on the simulator on the workstation
* A high-bandwidth, low latency channel connects the workstation to the accelerator to exchange signal data between testbench and design
* By [Amdahl's law](https://en.wikipedia.org/wiki/Amdahl%27s_law), the slowest device in the chain will determine the speed achievable.
  + Usually, the testbench
  + If TB very efficient (eg, transaction based), then it's the channel

**In-Circuit Emulation**

* Better implementation times than FPGA prototyping
* Provides comprehensive, efficient debugging
* At expense of higher cost

Acceleration/Emulation:

* Design executes simultaneously (as in silicon)
* same hardware is often used to provide both simulation acceleration and in-circuit emulation
  + blend of these two very different debugging styles
    - can set a breakpoint and stop emulation to inspect the design state, interact with the design, and resume emulation. The emulator always stops on cycle boundaries.
    - user has visibility to any signal or memory contents in the design without the need to set up probes before the run.
    - user can even back up time if checkpoints available