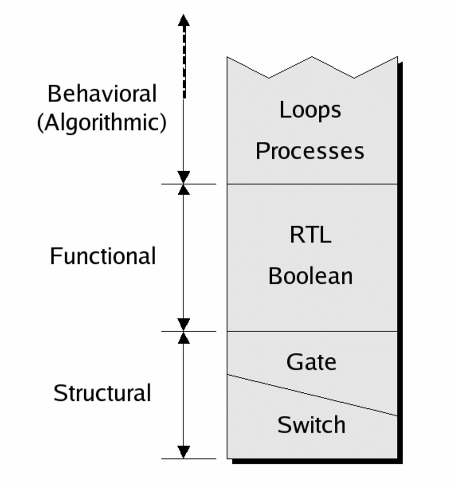
HDL-Based Design Flows: ASIC

Toward the end of the ’80s, it became difficult to use schematic-based **ASIC** flows to deal with the size and complexity of >5K or more gates.

HDLs were introduced to deal with this problem.

They can represent the functionality of a digital circuit at different levels of abstraction.



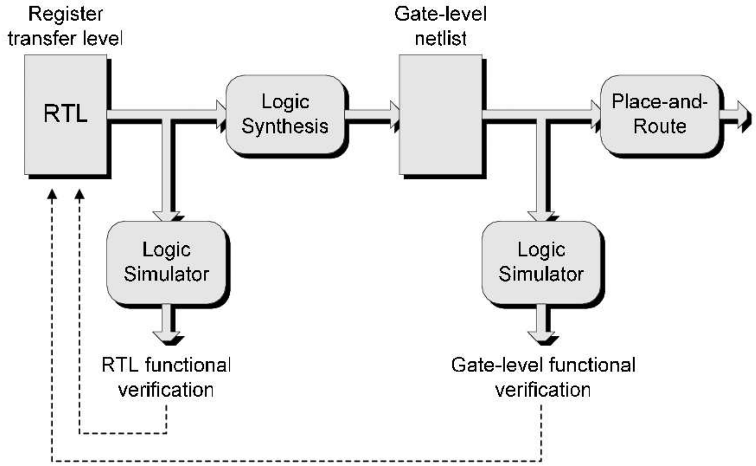
The lower levels, *gate* and *switch-level*, are the netlist formats described in relation to the schematic level tools.

As discussed w.r.t. Verilog, these are classified as ***structural*** representations.

HDL-Based Design Flows: ASIC

We've also discussed HDL support for *functional* representations (Boolean equations and RTL) and *behavior* representations.

An early (mid '80s) ASIC flow allowed RTL and timing constraints to be specified.



The key addition is *logic synthesis*, which

- Converts the RTL into registers and Boolean equations
- Performs a variety of minimizations & optimizations (for area and timing).
- Produces a gate-level netlist that meets the original timing constraints.

HDL-Based Design Flows

The *new flow* offered several advantages:

- Increased designer productivity significantly b/c it was easier to specify, understand and debug RTL level designs.
- Logic simulators could run RTL designs much more quickly than their gate-level counterparts.

Even though logic simulators could simulate *behavioral* level designs, early synthesis engines could only accept RTL level representations.

Therefore, designers were forced to work with a *synthesizable subset* of the HDL language (which still holds true today to some degree).

It took until the early ’90s before HDL-based flows, featuring logic synthesis, became fully available in the **FPGA** world.

In either case, once the netlist was generated by the logic synthesis tool, the flows were very similar to the schematic flows described earlier.

HDL-Based Design Flows: FPGA

The main problem with logic synthesis in the original HDL-based FPGA flows was that they were derived from ASIC flows.

The tools worked at the primitive logic gate level, and produced gate-level netlists which needed to be mapped, packed and P&R by the FPGA vendor.

In '94, synthesis tools became FPGA architecture *aware* and could perform *mapping* and some level of *packing* to produce a LUT/CLB-level netlist.

- The advantage is that synthesis tools had a better understanding of timing and area utilization.

Around 2000, the concept of *physically aware synthesis* began to take hold in the FPGA world, to address problems with obtaining timing closure.

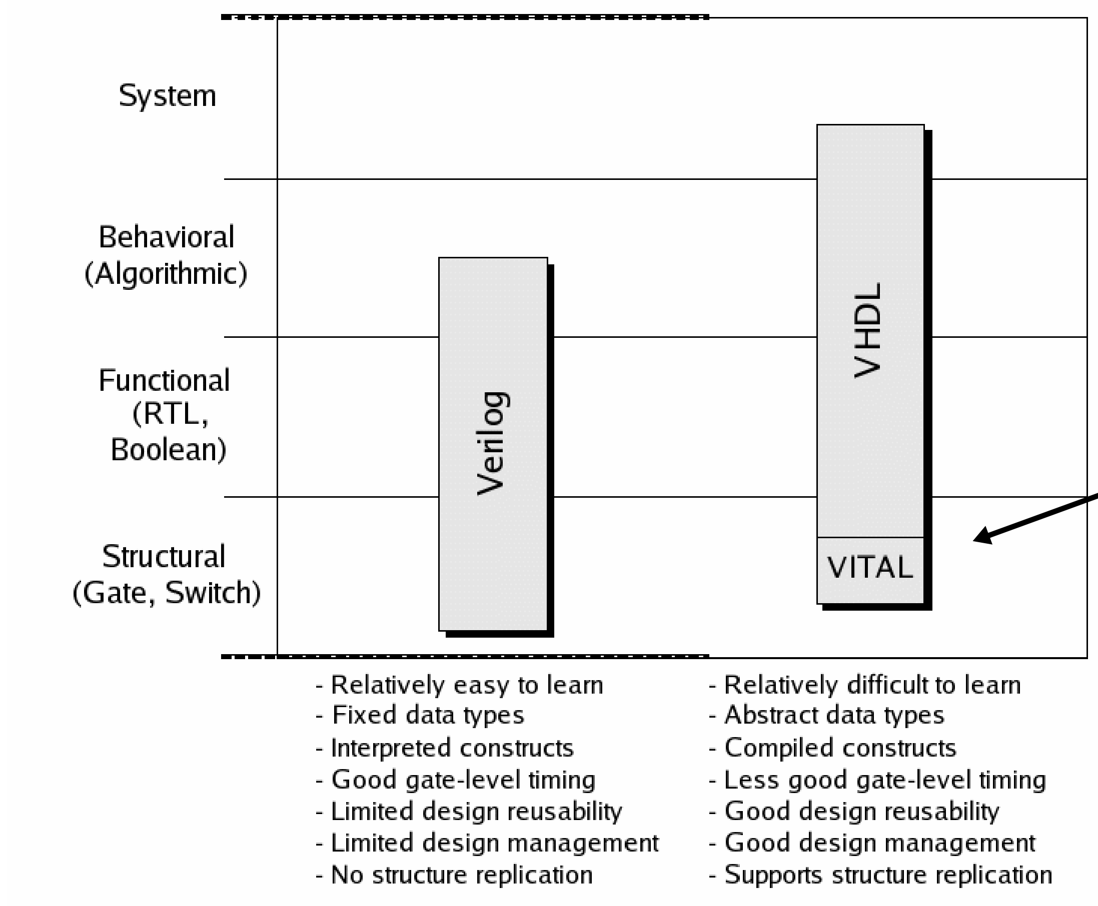
Here, the product of the synthesis engine was a *mapped*, *packed* and *placed* CLB-level netlist.

- The FPGA P&R tools started with the initial placement and performed *local* (fine-grained) placement optimizations and detailed routing.

Hardware Description Languages: Verilog and VHDL

The text gives a nice description of the *evolution* of Verilog and VHDL, as well as SystemC and SystemVerilog.

Support of each at the different levels of abstraction.



Lower levels not supported as well as in Verilog

The Design Warrior's Guide to FPGAs, ISBN 0750676043, Copyright(C) 2004 Mentor Graphics Corp