HDL Coding Rules and Guidelines

Gord Allan

September 3, 2003

Contents

1 Design Flow

ASIC design flows vary widely according to the current state of EDA (Electronic Design Automation) tools and company preferences. The current flow is based primarily on tools provided by Cadence Design Systems, but where appropriate, competing tools are mentioned.

In this document we will focus on the steps from RTL Design through to Global Routing, but for completeness the entire ASIC flow is described.

- *Specification* — The system design must meet any intended standards. Referencing the standard, the designer would typically create custom C models for their portion of the design. System-level verification is performed by integrating these models with reference designs and ensuring performance requirements are met. Typical tools for system level design and specification include Matlab/Simulink, Cadence's SPW, and Synopsys' Co-Centric. SystemC and other variants are also emerging to perform system level design and verification.

- *RTL Design* — With parameters from the system designer, the hardware engineer must efficiently implement the required algorithm. This is done at the Behavioural or Register-Transfer-Level (RTL) using constructs such as adders, multipliers, memories and finite state machines. The mapping from a system level algorithm to a hardware description is typically a manual process, though there are efforts to automate it. Verification of the RTL design is performed by comparing its I/O vectors with those applied to the system-level model. Simulation of RTL can be done using tools such as Cadence's NC-Verilog, Synopsys' VCS, or Mentor's Modelsim.

- *Generic Mapping* — This automated step takes the RTL description and attempts to map it to generic hardware components such as gates, flip-flops, and adders. If there are portions of the RTL which cannot be described by hardware (ie. unsynthesisable code) or other problems (eg. latch inferencing), they are often found at this stage. The mapping step is contained within the main synthesis tool where the available tools are Synopsys' Design Compiler(DC) and Cadence's Buildgates/PKS.

- *Constraints* — After mapping to generic hardware, the designer *could* immediately compile the design into digital library cells. Doing so, however, the tool will pick the smallest available architectures to do the job (eg. ripple-carry adders vs. carry look-ahead). This leads to slower designs. Most often, the design will be required to operate with a certain throughput, and thus, a certain clock frequency ($f_{critical}$). By constraining the design, the user guides the tool to optimize certain paths.

- *Floorplanning* — As technologies become smaller, delay due to interconnect resistance and capacitance becomes more significant than gate-delays. Therefore, if two cells are physically beside each other they will experience much less delay than if seperated by the length of the chip. Thus, in order to fully determine whether a design will meet timing and area requirements, it must be physically layed-out. During this step, the basic floorplan of the chip is described so that the interconnect delays can be estimated during compilation.

- *Power Planning* — Each cell must be connected to power and ground along its edges. To protect the chip wiring, the current through any particular wire must be limited below some threshold. Based on your design's speed, layout, and toggling activity, power rails must be distributed across the design so that this limit is not violated.
- *Compiling* — From the generic HW mapping, the tool picks elements from the digital library and logically arranges them to perform the required tasks within the timing constraints.
- *Scan Insertion* — If all of a design's flip-flops can be configured to form a long shift-register, manufacturing faults can be detected. Tools can automatically place multiplexors at the input to all flip-flops and link them together into a 'scan-chain.' During normal operation the circuit is unaffected, but when a test signal is asserted the scan-chain can be used to isolate manufacturing defects. Synopsys' DC, Cadence's PKS, and Mentor's FastScan can automatically insert the additional circuitry to allow scan-testing.
- *Clock Tree Insertion* — Ideally the clock signal will arrive to all flip-flops at the same time. Due to variations in buffering, loading, and interconnect lengths, however, the clock's arrival is skewed. A clock-tree insertion tool evaluates the loading and positioning of all clock related signals and places clock buffers in the appropriate spots to minimize skew to acceptable levels. Some clock tree insertion tools, all from Cadence, include CTSGen, ctgen, and CTPKS.
- *Optimization* — After placing the cells, adding scan circuitry and inserting a clock-tree, the design may no longer meet timing requirements. This optimization step can restructure logic, re-size cells, and vary cell placement in order to meet constraints.
- *Routing* — Up until this point, all timing estimates assume that signals can be routed without being detoured, as can be caused by wiring congestion. After initial optimization, the routing is actually performed in two steps:
    1. *Global Routing* creates a coarse routing map of the design. It evaluates areas which are highly congested and plans how signals should go around those area. After global routing, the design can be re-timed using more accurate interconnect data.
    2. *Final Routing* uses the plan from the global route and lays out the metal tracks and vias to physically connect the cells. Two final-routers are available - WarpRoute and NanoRoute.

- *Parasitic Extraction* — Once the detailed routing tracks are inserted, an extraction tool is used to more accurately determine the resistance and capacitance of each net. Two such extraction tools are 'Fire and Ice' and 'HyperExtract.' These tools can also be used to determine the cross-coupling capacitance between two signals which are important when evaluating signal integrity.

- *Post-Routing In-Place-Optimization* — After importing the parasitic information (usually in the form of a .rspf file), timing is re-evaluated to ensure it meets the constraints. At this stage limited changes can be performed, such as cell re-sizing and net re-routing in attempts to 'close timing'.

- *Signal Integrity Fixes* — If the cross-coupling capacitance between two signal lines is high, quick transitions on one net can affect the other. Within the EDA tools, these nets are referred to as 'victims' and 'agressors'. Agressors are characterized by large drivers and quick transistion times, whereas victims posess the opposite characteristics. Signal integrity violations can be divided into two categories:

  1. *Crosstalk* is caused when a victim and agressor pair transition at the same time. The victim may be either sped up (if both signals transition in the same direction), or delayed. This variation is then taken into account for either best or worst case timing analysis.
  2. *Glitching* is caused when a transition on the agressor net can cause a logical change (from 1-to-0 or 0-to-1) on the victim net.

  In either case, the signal integrity tool (Cadence's CeltIC) identifies the victim and agressor nets for repair. To fix such a violation, buffers can be inserted, nets can be re-routed, or shielding can be inserted between the offending nets. After any signal integrity fixes, extraction is re-done and timing closure must be verified.

- *Physical Checks* — Once timing closure has been assured, various physical checks are carried out. If any changes are made, extraction should be redone and timing re-evaluated:

  - *Antenna Check* — During manufacture, when a metal patch is being deposited charge builds upon it. If the charge builds faster than it can be dissipated than a large voltage can be developed. If a transistor's gate is exposed to this large voltage then it can be destroyed. This is referred to as an antenna violation. To prevent this, leakage diodes can be inserted to drain excess charge, or long metal traces on a single layer can be prevented.
  - *Layout vs. Schematic (LVS)* — The LVS tool extracts the connectivity information from the routed layout and compares it with the final logical netlist. An LVS match confirms that errors were not introduced during the physical layout of the design. Tools to perform

for LVS include Cadence's Assura (formerly Diva, formerly Dracula) and Mentor's Calibre.

– *Design Rule Checking (DRC)* — The design rule check validates that the spacing and geometry in the design meets the requirements of the foundry. The same tools used for LVS are used to perform DRC.