2.0 FPGA DESIGN GUIDELINES

You can utilize these sample guidelines or modify with your own.

2.1 Design Entry Guidelines

2.1.1 Top-level Design

The top-level of the hardware design should be presented as a block diagram or schematic, which modularizes the design into reasonably sized functional blocks. The top-level diagram should show all interconnects between the functional blocks. Once the top-level block diagram is designed, HDL can be generated and synthesized. The top-level block diagrams should be constructed to provide a complete overview of the hardware design.

2.1.2 Functional Block Design

After a top-level block diagram is completed, the content of each functional block is to be designed. Either a schematic capture tool or HDL may be used. Of the two methods, using a HDL enables a more portable design allowing for device independence (i.e. not tied to a gate-level library until logic synthesis) and portability. Some blocks will be more suited to using a HDL/synthesis tool and others will be more suited to a schematic capture approach. In general, the HDL method should be used for more complex designs.

2.1.3 Hierarchical Design

A complex programmable logic design is typically approached in a hierarchical fashion allowing the designer to focus on a manageable portion of the circuit at a time. In general, any hierarchical designs with more than four levels are difficult to manage and review. For that reason, the design hierarchy should be limited to no more than four levels.

2.1.4 Synchronous/Asynchronous Design

Synchronous design should be the preferred method over asynchronous design because asynchronous design often depends on element delays which are difficult to predict and verify. Synchronous logic design coupled with static timing analysis provides a reliable method to verify the hardware design timing. Synchronous design also provides assurance of robust timing, because simulation methods always face pragmatic limits regarding assurance that all functions are actually tested and all delay paths have been simulated. Synchronous logic design also provides a benefit regarding design reuse issues, including parts, obsolescence issues requiring migration of the design to a newer generation device or possibly to a different device family or manufacturer from the original target device.

2.1.5 Gated Clock

The gated clocks shall be avoided because they cause more skew between the clock and data and potentially generate glitches. Internally generated clock usage within the hardware design shall be also avoided for the same reason. If the design requires a clock divider within the hardware design, the designer shall use a PLL or a DLL to generate the divided clock.

2.1.6 Revision Label

Note, this is just providing an example. You can define your own revisioning format.

If the hardware design has a microprocessor interface, the design should have a register that contains a unique hexadecimal revision label for the build. The format should be MMmm, where MM is a major revision number and mm is a minor revision number. All

values are to be read as binary coded decimal (BCD) numbers.

The revision label shall be implemented in a HDL source code file that shall be updated when creating a new release and/or implementing/deleting major functionality. Therefore, the revision label will be compiled into the binary configuration data and readable through the device interface to the microprocessor. This feature will ease determination of the hardware binary configuration data in the system in the production/maintenance processes.

2.1.7 Source Readability

HDL source code shall conform to human readability. The review process for the detail design representation should ensure the human readability criteria are met. While the HDL source code shall largely avoid structural notation, the designer may make use of "logic blocks" created by the tools or written separately to render large standard structures. Detail design representations shall be formatted so as to make clear the structures that exist in the design.

HDL source code shall be indented according to structure levels to aid readability. Block diagrams shall be laid out to aid reading the diagram. The conceptual and detail design representation shall include comments and notations to enhance clarity of the design intent. Excessive comments can hinder clarity by obscuring structure inherent in the representation and should be avoided.

2.1.8 HDL Coding Standards

See Appendix A "VHDL CODING STANDARDS" for a complete list of standards that will guide the development of the HDL code for this design.

2.1.9 Static Timing Analysis

Most hardware Place and Route tools have some level of static timing analysis included. When hardware place & route is completed, the tool generates a report that can be used for static timing analysis. The designer shall check the frequency of the main clock against the maximum operating frequency from the report and ensure that the design has a minimum 10% margin. As explained earlier, synchronous logic design coupled with static timing analysis provides a reliable method to verify robust timing closure in the overall design.

2.2 Process-Related Guidelines

2.2.1 Device & Tool Selection

Some important considerations in selecting the hardware vendor are design support, device cost over time, vendor's financial stability, tool compatibility with the the platform on which it will be installed, and past performance of the vendor. Types of tools used may include:

- Schematic capture software
- Fitter software (for place and route)
- Synthesizer software
- Floor planner software
- Simulator software
- HDL synthesis software

These tools may be obtained from the hardware vendor or obtained from third party vendors. The target device should be selected based on size requirements, pin count, maximum gate count, complexity of the required logic, speed requirements, power consumption requirements, capability for "on board" programming, and forecasts of part

obsolescence. Whenever feasible, the specific piece part should be chosen from a preferred parts list.

2.2.2 Traceability & Archive

Once the detailed design with functional simulation step is completed, a baseline shall be created in the configuration control tool prior to executing synthesis. Design traceability (of one version to another) and archiving shall be done according to <doc-ref#>_HCMP.

2.2.3 Prototype Activity

The lifecycle for hardware development allows an iterative approach, beginning first with a prototyping phase. The prototype phase will be used to elicit requirements from our customers and higher design levels. The prototype serves to aid in the generation of requirements for the hardware design. The prototype phase also permits the ability to explore the requirements for both software and hardware. The prototype activity is considered an external process to the development effort. The issues identified in the prototyping phase may be incorporated into the system design description, which allocates requirements for both software and hardware.

After the prototype phase, the iteration of requirements capture can begin. The hardware requirements, conceptual design, and detailed design are cyclic or iterative. This approach allows the hardware design team to design, code, and integrate subsets of requirements at a time, keeping the task at hand small and manageable. The prototyping effort may implement many of the functions required for the official development process.

HDL source code generated during prototyping or other informal development efforts, as suitable, shall be subject to Hardware Design Standards. After the prototype phase is complete, formal development activities may begin.

2.2.4 Design Description

The Hardware Design Data (<doc-ref#>_HDD) should be captured during the detailed design with simulation step and may be used for hardware validation and verification. The design description should consist of the following level of details:

- Assignment of groups of functionalities to basic design blocks (conceptual design data) and HDL code.
- State Diagrams with transitions and I/O signal behavior, timing diagrams
- Data flow descriptions describing clock cycle timing and timing diagrams.
- Memory element descriptions, timing diagrams and connections
- Register map with register field descriptions

2.2.5 Previously Developed Modules

The project may utilize source code from previously developed projects. In the event that any part of the design is reused, it will be baselined and used as an entry point to begin development. The project will not take credit for previous certification of the reused design.