Do – 254 notes

This standard has basically three parts

1. Block diagrams flow chart that explains the design before coding.

The project documentation contain must of the drawing that were

Done prior the vhdl coding process.

1. The coding process, this part contain the d0 -254 requirement to

Design a reliable vhdl design. Below the list of main code policy

Requirement.

Cp2 – Avoid duplicate signal assignment.

Cp3 – Avoid hard coded numeric values. Constant or generics should

Be used.

Cp5 – Ensure consistent FMS state encoding style.

Cp6 - Ensure FMS transition.

1. An FMS should have a defined reset state.
2. All unused (illegal or undefined) states should transition

To a define recovery state. There should be no unreachable states

(i.e those without any incoming transition) and dead-end states.

(i.e those without any outgoing transition) in a FMS.

Cp7 - Avoid mismatching ranges.

Cp8 - Ensure complete sensitivity list.

Cp9 - Ensure proper sub – program body.

1. Have only one exit point.
2. Have no recursion.
3. Access only local variable/signals.

Cp10 – assign values to variables before using.

Every object (e.g signals, variables, ports) should be assigned a value

Using it.

* When used before assign a value a latch may be inferred during

Synthesis which most likely unindent function behavior of the design.

Cp11 – Avoid unconnected inputs. All inputs should be driven.

Cp12 – Avoid unconnected output ports.

Cp13 - define objects before using.

Cp14 - Avoid unused declaration. All declaration should be used.

# High level design concept.

Simple block diagram description.

Basic function and interface.

Drawing with power point.

Automatic checking of the design.

QuartusII compilation report ( basically warnings.)

Timing test (timquest) and file.SDC file.

Clock domain – problem of met stability which is very hard to detect.

Update in paralleled feedback may create race condition which means

Variables are not update as expected.

Cd1 – With multiple clocks a through clocks crossing (CDC) analysis should be done.

(met stability indeterminate) very hard to detect at hardware level.

Worse – This problem will not show during test, but might appear

During flight.

Ss2 - In case and state, always use “when others”.

Ss3 - Avoid combinational feedback.

Feedback cause race condition and unpredictable behavior.

Ss4 - Avoid latch inference. Carful to avoid latches in the design.

Ss5 - Avoid multiply waveform , only one waveform should exist at the right hand

Side a signal assignment.

Ss6 - Avoid multiple drivers, only one sequential block.

Ss7 - Avoid uninitialized VHDL constants. All VHDL constant must be initialized.

Ss8 - Avoid clocks used as a data. Clock signals should not be used in a logic path

That drive data to registers.

Ss9 - Avoid shared clock and reset signals.

Ss10 – Avoid gated clocks data.

signals should not be used in a logic path that drives clock input.

Ss12 – Avoid internal reset. Use for example Reg <= (others => ‘0’);.

Ss14 - Avoid non resettable registers. All registers should have reset control.

Ss15 - Avoid asynchronous reset release.

Reset signals should have a synchronous release.

Ss17 – Avoid un driven and unused logic. Every register must be used and driven in the

Design. No “dead code”.

Ss19 - Avoid snake path.

Combinational path should not exceed maximum allowable logic depth.

DESIN REVIEW DR

Dr1 – Case statements, process, block, initial block – should have labels.

Dr2 - Avoid mixed case naming. Names should not be differentiated by case alone.

Dr3 - Use unique names.

Dr5 – use separate statement style. Each statement should be placed on a separate

Line.

Dr8 – Avoid large design files. Avoid large flat code structure.

Dr11 – Code should be sufficiently documented via in line comments.

Dr12 – Comments should be placed in appropriate places to aid understanding.

SUMMARY OF DO – 254

A – schematic drawings, block diagrams, flaw chart and any additional information

In preparation for team review before starting of detail design.

B – Detail VHDL code requirement.

1. Define reset state.
2. Watch mismatch ranges.
3. Assign values to signals before using them. Uninitialized signal might create latch.
4. No unconnected inputs.
5. No unconnected outputs.
6. Define objects before using.
7. All declaration should be used.
8. No unused lines or process in the design.
9. Watch for clock domain. Example (sysclk and clk10mhz).
10. In case or state machine use “when others”.
11. Use a complete sensitivities list,
12. Watch not to use latch in the design.
13. Avoid gated clocks.
14. All registers should have reset function.
15. Reset signal should have synchronous release.
16. No unused code, no “dead logic” avoid logic that is not used.
17. N0 snake path, too long logic in a single process.
18. Avoid long files. No long flat code.
19. Code with sufficient documentation and in predetermined place.

1. TOOLS
2. Compilation with qurtusii – warnings
3. Time request test of the design.
4. RTL view of the design.

■ Clock-domain crossing analysis: Integrating multiple functions into a chip is commonplace today and usually involves multiple, asynchronous clocks. Clock signals that cross domains can lead to a condition called metastability, which is a leading cause of device failure. The problems associated with signals that cross clock domains are extremely difficult and expensive to debug and fix because they typically are not detected until a failure occurs in the lab or field. Mentor Graphics 0-In® Clock Domain Crossing (CDC) is an analysis tool based on formal methods that can help reduce the likelihood of metastability. For more information, see “Mitigating the Dangers of Multi-Clock Designs” (presented at the 2008 FAA National Conference) found at www.mentor. com/go/do-254.

■ Static timing analysis: Static timing analysis, STA, runs during synthesis and also place and route processes. STA analyzes paths to ensure they meet timing constraints. STA run during synthesis reports only estimated timing. STA run during place and route includes real delays as the design is being implemented into the real silicon. Unconstrained paths go unreported in STA, so it’s important to properly constrain the design, and have this independently reviewed to ensure it’s done properly. STA does not replace functional timing analysis (i.e., full timing simulation) and CDC analysis. The results of STA can be back-annotated onto the netlist for full timing simulation.

■ Netlist (full timing) simulation: The transformations of synthesis and also place and route must be verified. This is typically done by re-running the HDL tests on the resulting netlist, with timing information from STA back-annotated. Because a netlist with timing information has much more detail than the mere functional description of the HDL code, this process can take quite a bit longer. Still, the general DO-254 expectation is that full test suite will be run. If the design is very large and complex, and the test suite takes days (or weeks) to simulate, other methods, such as logical equivalency checking may be considered. Mentor Graphics ModelSim tool can run this full-timing simulation as well

■ Hardware testing: Verifying the device in its target system is the ultimate test as to whether the device performs its intended function. The device, after all functions as part of a system, and proper system function is what is critical for an aircraft. Prior to this phase all verification has tested the device either in isolation or within a model of the system. Testing the hardware (HW test) can occur either by 1) testing the silicon (programmed FPGA) in isolation or 2) in its target system. While the former is good practice, the latter is imperative. In support of No. 1, numerous new tools have entered the market to support integration of simulation and HW test. These tools reuse the simulation testbench on hardware, drive HW testing from the simulator, compare silicon to simulation results, and/or provide visibility into hardware for ease of debug. GateRocket (www.gaterocket.com) provides these sorts of tools, which can help ensure a very clean device to the final “in-target” (i.e., in-system) testing phase. However, they cannot take the place of the final system testing that DO-254 compliance requires. System level design and analysis can likewise support links to physical testing while also helping with the final system testing phase. The section Looking Beyond the Chip provides more information on this topic. The following verification methods are less commonly used today but are nonetheless very powerful. Interest in and acceptance of these methods continues to grow. ■ Formal verification (model checking): Listed as an acceptable method of advanced verification for level A/B devices in DO-254 appendix B, model checking is a formal methods technique that analyzes