HF Controls

FPGA Development Process

WORK INSTRUCTION-ENGINEERING DEPARTMENT

Work Instruction No: **WI-ENG-108** Revision: A

Effective Date: 4/16/18

Prepared By: _____

Reviewed By: _____

Approved By: _____

FPGA Development Process

Date	Rev.	Author	Changes
1/17/2018	A	J. Mott	Initial Release

TABLE OF CONTENTS

TABLE OF FIGURES

1.0 PURPOSE AND SCOPE

1.1. The purpose of this procedure is to describe the process for designing, implementing, and testing a FPGA that is designated to be used on a HFC FPGA based PCB assembly.

1.2. This procedure applies to safety and safety related HFC hardware products having functional characteristics intended for application in nuclear power plant control.

1.3. This procedure applies to design process for safety components designed by HFC using FPGA devices purchased from HFC qualified vendors.

1.4. This procedure applies to the implementation process for HFC designed safety components that use FPGA devices.

1.5. This procedure describes the test process for HFC designed safety components that use FPGA devices.

2.0 ORGANIZATIONAL RESPONSIBILITIES

2.1. The VP of Engineering / Designee is responsible for ensuring that an effective FPGA design/implementation/test procedure is developed for all new hardware assemblies to be included as part of an HFC product line.

2.2. The lead test engineer is responsible for developing and maintaining these test procedures.

3.0 REFERENCES

3.1. WI-ENG-001, “Design Verification and Reviews”

3.2. WI-ENG-102, "Printed Circuit Board Design”

3.3. WI-ENG-104, "Development of Hardware Requirements Specifications”

3.4. WI-ENG-106, "Development of Hardware Design Specifications”

3.5. WI-ENG-103, "HDL Coding Standard”

3.6. WI-ENG-202, "Development of Software & Firmware Requirement Specification”

3.7. WI-ENG-203, "Develop Software Design Specification”

3.8. WI-ENG-830, "Source Code Review"

3.9. NUREG CR-7006 Review Guidelines for FPGAs in Nuclear Power Plant Safety Systems.

3.10. IEEE 1364-2001 IEEE Standard Verilog® Hardware Description Language

4.0 GENERAL INFORMATION

4.1. FPGA based PCB assemblies used in nuclear I&C systems requires a design process that has been intentionally created to produce reliable, robust, defensive, synchronous, traceable and maintainable FPGA designs.

4.2. The implementation process for realizing an FPGA used in nuclear I&C systems requires a rigorous, well defined process that has systematic verification of each step in the process.

4.3. The test procedure for FPGA based PCB assemblies used in nuclear I&C systems requires a thorough, systematic, well defined test to verify the FPGA behaves as designed.

5.0 PROCEDURE

5.1. **HFC FPGA Hardware Design**

5.1.1. FPGA PCB Physical Implementation

FPGA's require a solid PCB hardware design to support the code that has been synthesized to use the FPGA resources. These details are captured in the hardware requirements and design specifications. This is the input to the schematic that represents the connectivity between devices on the PCB. The completed schematic is the input to the layout process which organizes the devices and implements the power and signal connectivity required

[

]

Required steps to complete when designing a PCB that contains a FPGA:

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

[

]

5.1.2. Robustness

Hardware designs that use FPGA's and operate under all conditions are very challenging. The following guidelines apply to using FPGA's in a robust hardware design:

- Design the architecture with redundancy to ensure error detection and if required correction of system output.
- Use Single Event Upset (SEU)-immune FPGA's. Use flash based FPGA's over SRAM based FPGA's. Beware that SEU immunity only applies to the configuration data in many cases.
- Employ watchdog timers to detect system failures and force failover to hot standby/redundant resources.

5.1.3. Traceability

Traceability in FPGA hardware design applies to the careful use of revision control and design storage facilities. These requirements are called out in WI-ENG-102.

Design documentation from the part vendor shall be controlled for all parts on the PCB. Datasheets and part documentation used in the design are required in the approval of the final design for Nuclear applications.

5.1.4. Maintainability

It is a requirement to be able to maintain the FPGA based over the 40-year lifespan of the assembly.

Some required steps to making the FPGA based assembly maintainable:

- Include all parts in the final assembly to support reprogramming of the FPGA by site or factory personnel. Make sure to add masking instructions for the connector(s) so that good connections can be made after conformal coating.
- Select vendors with a long-term presence in the FPGA market that have a history of supporting their legacy parts and design tools. Purchase extra FPGA devices with the same Die Revision as tested to be able to respond to customer requests with out re-testing.

[

]

5.2. **HFC FPGA Hardware RTL Design Entry Practices**

Refer to WI-ENG-103 for RTL design entry practice requirements.

5.3. **HFC FPGA Development Process**

The design process flow chart for FPGA development is shown in Figure 1 - FPGA Development Process . The input to the process is an approved software requirements specification. The work instruction for this document is WI-ENG-202.

FPGA Development Process

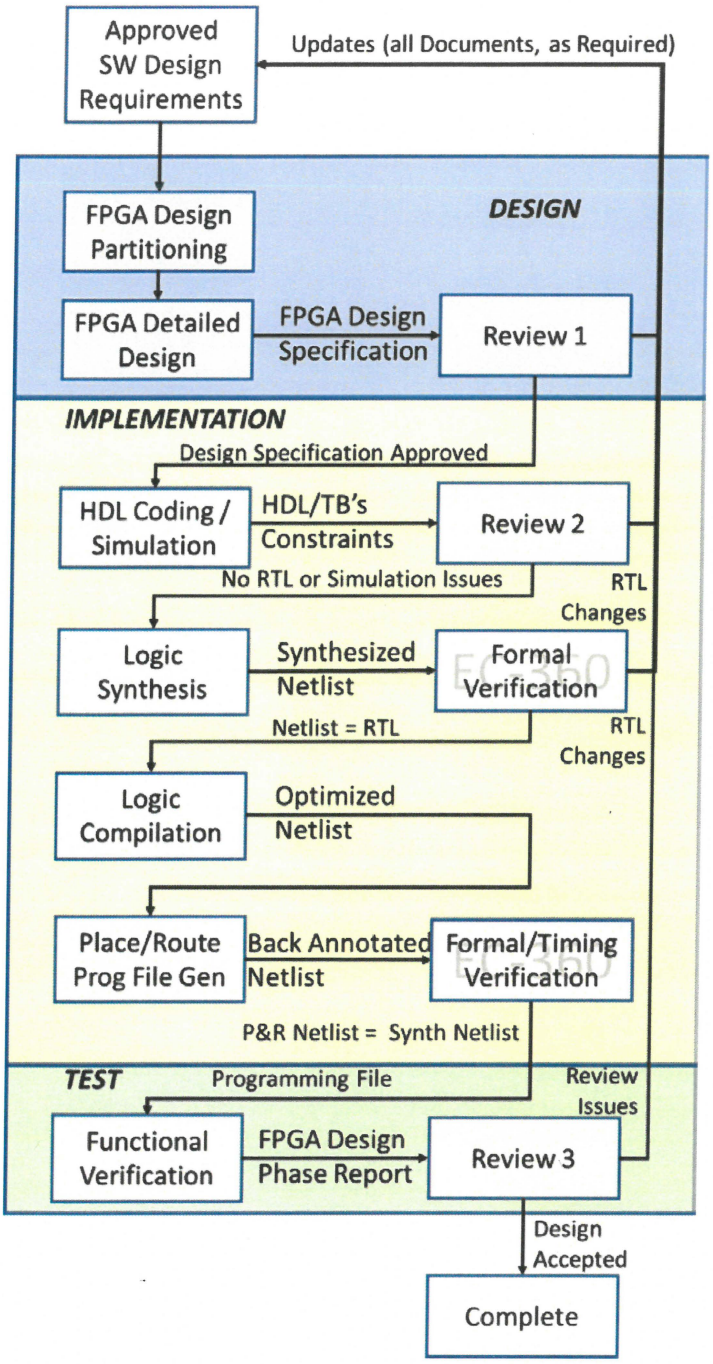

Figure 1 - FPGA Development Process Flow

5.3.1. FPGA Development Tool Requirements

FPGA development tools shall be carefully chosen. Revisions used shall be controlled by the R&D manager and not be changed unless appropriate analysis has been done and communicated to engineers involved in the development. Issues experienced with tools (problems, service packs that impact timing and functionality, etc.) should be investigated with the tool vendor and if an issue cannot be resolved, reported to the R&D Development manager.

Execution of the development tools shall be controlled by script with approved tool settings. Changes to the scripts shall be reviewed during the implementation process to get agreement on proposed changes and their understand the justification.

5.3.2. Software Design Requirements

The software design requirements document begins with a narrative description of the functional requirements for the FPGA software.

Requirements for FPGA's used in safety and safety related equipment shall emphasize reliability and deterministic behavior. This includes the requirement that all code blocks in the design shall operate in a synchronous manner.

Requirements need to specify that all error conditions encountered should be detected and reported. These requirements shall be verifiable and reviewable throughout the FPGA development process.

Additional requirements, such as critical timing, power, and board size, may be added to the requirements.

FPGA Software requirements can be found in either the Hardware Design Requirements document or the Software Design Requirements documents. In most cases, the Software (RTL) designed interacts directly with the hardware designed on the PCB and therefore including FPGA RTL requirements in the hardware design requirements specification is acceptable. Refer to WI-ENG-104 for more details.

5.3.3. Design Partitioning

Based on the documented requirements from the previous section, the internal design of the FPGA can be divided into functional blocks. These blocks can be classified into the following categories: input blocks, output blocks, processing blocks.

Input and output blocks provide an interface function with an external device. The datasheets for the external device describe the behavior of block whether input or output or input/output block. Constraints must be developed to manage the input/output timing for these external interfaces. See the user's guide for the toolset being used to create the programming file for more information.

Processing blocks use internal FPGA resources to modify data received from external devices through input/output blocks. The processing of data received shall be deterministic and in most cases controlled by a Finite-State-Machine (FSM).

If possible, functions in input, output, and processing blocks should be divided into smaller blocks in order to achieve complete verification.

Functions should be grouped according to a common clock. Where blocks using one clock interface with blocks using a different clock, this should be noted in the FPGA block diagram as a timing domain boundary.

Additional circuitry for health monitoring, built-in self-test, redundancy (SEU Mitigation), and observability needs to be considered. The objective is to have a design that has 100% testability.

FPGA CPU modules whether soft processor cores or hard I/P cores are not allowed in safety or safety related FPGA designs.

Design partitioning details shall be documented in the software design specification with an internal FPGA block diagram. The internal FPGA block diagram shall show all external interfaces to the FPGA (in high level detail) and which internal blocks interface to the external interfaces. In addition, a paragraph describing the function of each block shall be included.

Note: where safety and non-safety functions are intended to reside in the same FPGA, preferred practice is to segment the safety from the non-safety functions into separate FPGA's.

5.3.4. Detailed Design

The objective for detailed design is to define all blocks in the design partitioning, internal FPGA block diagram, in such detail that the design can be coded in HDL by persons not familiar with the architecture. The output of this step is a FPGA software design specification containing the details of each functional block (internal block diagrams, FSM state flows, algorithm details, bus widths, clock boundaries and synchronization, etc.), the details of their interfaces (bus widths, control signals, etc.), and other information necessary for HDL coding.

This detailed design information shall be captured in the Software Design Specification. Refer to WI-ENG-203.

Note: If in the process of Detailed Design, areas of the design are noted where improvements can be made to the reliability or testability, the R&D manager shall be notified for evaluation.

5.3.5. Design Review

Design reviews shall occur in multiple steps to:

- Ensure the description of the software to be implemented meets the requirements specifications. The hardware RS/DS and software RS (if required)/DS are inputs to this design review. Review 1.

- Ensure the RTL has been properly coded and simulated to proceed to the implementation phase. Also reviewed at this time are the constraint and script files needed for implementation. The updated hardware RS/DS, software RS/DS, RTL (FPGA and Simulation Test Benches), and constraint files are inputs to this design review. Review 2

- Ensure the RTL has been properly designed, implemented, and tested to close out the design phase and move into the verification and validation, and integration test phases. Review 3

At each step in the review process, if changes are made to the documents reviewed in previous steps, these changes shall be reviewed as well.

5.3.6. Design Behavioral Coding

In the next design step, the designer converts the FPGA software design specifications into behavioral descriptions that delineate

the functionality required for each module and their interactions. The HDL coding is done by using the information found in the FPGA Software Design Specification. The coding shall be done in IEEE 1364-2001 Verilog HDL. The HDL code is processed by the synthesis tools to generate synthesizable register transfer level (RTL) representation of the digital logic.

It is essential that the hierarchy in the behavioral description exactly represent the design partitioning done during the architectural design step. Any changes required, shall be brought to the attention of the author of the FPGA Software Design Specification for evaluation and approval. If approved, the FPGA Software Design Specification must be updated to reflect the new architecture.

Verilog HDL language shall be used unless a justification can be made to use VHDL or another HDL language. These must be evaluated by the R&D Development manager in charge of the FPGA development.

Also, the HDL design practices detailed in WI-ENG-103 shall be followed when creating the Verilog modules used in the next step, behavioral simulation.

5.3.7. Behavioral Simulation

The output from the Design Behavioral Coding step (HDL modules) shall be the input to this step. HDL test benches shall be created to thoroughly test each module in the design.

Test benches shall be self-checking and scriptable to allow them be added to a regression test script. Test benches shall be created to apply expected and unexpected inputs to the design under test (DUT). Correct behavior shall be verified for both the expected and unexpected inputs under automated control. Log files shall be generated to note errors that are encountered in running the test benches. These log files shall have a common format in order to provide for automated checking log files during regression testing.

[

]

Coverage statistics shall be collected. The coverage requirement is 100% statement, branch coverage, expression, and FSM. Justifications shall be created to explain deviations from this requirement and reviewed.

[

]

[

]

5.3.8. Logic Synthesis and Compilation

Logic synthesis, in preparation for physical implementation, can be done in multiple steps. The current FPGA device uses two steps, Logic Synthesis using an OEM industry standard tool and Logic Compilation, that does optimization, uses a vendor specific tool.

Regardless of the number of steps required to generate the files for physical implementation, the Formal Verification tool EC-360 shall be used following each Logic Synthesis step to verify that all RTL specified in the RTL Coding/Simulation step is present prior to in the Physical Implementation step.

It may be possible to formally verify equivalence of the netlist after the Logic Compilation step. If the netlist output from this step is demonstrated to be formally equivalent, then the Logic Synthesis formal verification step can be skipped. This is the alternate path shown in Figure 1 - FPGA Development Process .

5.3.9. Post-Synthesis/Compilation Equivalency Checking

The Formal Verification tool EC-360 from OneSpin shall be used following the Logic Synthesis and Compilation steps to verify that all RTL specified in the RTL Coding/Simulation step is present prior to Physical Implementation step. As explained in the previous section, an alternate path exists to only demonstrate equivalence after the Logic Compilation using the EC-360 tool.

5.3.10. Physical Implementation

Physical Implementation refers to the place and route and programming file generation steps in the implementation phase. This can vary from FPGA vendor to vendor, but the end result of the Physical Implementation step is the programming file for the FPGA.

If Place and Route completes successfully (this is indicated in the report file), then the vendor tool (or approved equivalent) must be used to verify that all register to register delays meet the setup and hold time requirements. When required constraint files can be modified to include false paths, multicycle paths, and paths where approved timing domain crossing is implemented. This part of the Place and Route step is called Static Timing Analysis (STA) or Verify Timing and must generate a report that lists any timing violations. STA shall be done over the required temperature range with worst-case and best-case voltage and process. The report data from this step must be preserved and saved for review after the Test Phase is completed.

The vendor supplied tool shall be used to generate the programming file used in the Test Phase for prototype verification.

5.3.11. Post-Layout Verification

Performing multiple timing back annotated simulations is an alternative method for completing the post-layout verification step.

[

]

5.3.12. Functional Verification

At the end of the FPGA Development Process, functional hardware verification shall be performed to confirm correct board/module functionality in the intended operational environment.

Functional hardware verification is performed on the entire PCB board with programmed FPGA's using a limited set of input stimuli that covers the input combinations expected during normal operation of the FPGA.

Functional hardware verification shall verify that no design issues are present when interacting with the circuitry interfacing with the FPGA device.

Items that shall be checked include:

[

]

[ ]

[

[ ]

[ ]

[ ]

[
]

An additional functional verification step to generate input stimuli that includes input combinations not expected during a normal board operation to confirm that there are no unexpected responses.

FPGA hardware design practices described in section 5.1. HFC FPGA Hardware Design should be used as guidance for the hardware verification.

Thermal testing shall be done on the device under test to verify that no functional issues are present in the intended operation range.

5.3.13. Design, Implementation, and Test Process Review

The final review shown in the process diagram shall be a comprehensive review of all design activities completed, reports generated, all design documents generated and their released numbers, action item completion status, verifying the storage locations for all files that pertain to the review and design process.

Attachment 7.1 is a checklist form for the FPGA Design Review and shall be completed in the review meeting. Attachment 7.1 is located in section 7.0 of this document. This form is complete when all fields have been entered and the Reviewer(s), FPGA Development Manager, and the V&V Project Manager have signed in the spaces allocated.

6.0 QA RECORDS

FPGA Design Process review forms are attached as Attachment 7.1.

7.0 ATTACHMENTS

7.1 FPGA Design Review Checklist

FPGA Design Review Checklist

REVIEWER(s): _____ **DATE:** _____

FPGA DEVELOPMENT MANAGER: _____ **DATE:** _____

V&V PROJECT MANAGER:_____ **DATE:** _____

FPGA DESIGN NAME: _____ **REVISION**_____

FPGA PART NUMBER: _____

[

Yes	No
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐

]

List any discrepancies found:

Suggestions to resolve discrepancies: