Use of FPGAs is on the rise, including in safety- and mission-critical applications, which previously were the exclusive realm of ASICs. As a result, FPGA designers are continually trying to improve their implementation methods to ensure safe circuit operation; work that’s spurred in part by enforcement by the regulatory authorities worldwide of safety standards such as DO-254 to ensure safety of in-flight complex hardware. These and similar standards in other countries (e.g., ED-80 in Europe) provide design assurance guidance for airborne electronic hardware; guidance that’s most commonly applied to PLD, FPGA and ASIC designs.

But aviation is just one example. Industry segments including military, automotive, space, medical, nuclear and transportation all have similar standards or concerns. The common objective of such standards is generally to ensure that the device produced will perform its intended function (as specified by requirements) under all foreseeable conditions. The development process itself becomes a big part of the proof. This means that FPGA designers and project leads are in effect frequently tasked with a twofold challenge—to not only meet a high standard of product reliability but also to prove that the proper planning, documentation, design and verification activities have been undertaken.

Selecting the right methodology and tool flow are among the most important decisions that must be made before initiating a DO-254 or other safety-critical project. No methodologies or tools are inherently certified, compliant or qualified for these types of programs. However, many companies that are concerned about design assurance are reevaluating their design methods and tools since both play an important role in overall program compliance while affecting productivity, schedule, budget and design quality.

Design Assurance—The Fourth Dimension of FPGA Synthesis

While requirements tracking and functional verification get much of the attention in design flows, the synthesis stage is of similar if not greater importance. After all, it is synthesis that takes the work of the designer and ultimately generates the corresponding hardware gates to be placed and routed in silicon. This entails not only ensuring functional equivalence of those gates to the original RTL created by the designer, but also the addition of safety features such as triple modular redundancy (TMR) when appropriate. Accordingly, before engaging in the actual design work of a safety-compliance project, it’s critical to understand how synthesis works and the broader context in which it is used.

Fortunately, when it comes to FPGA implementation, new synthesis automation technologies are helping to simplify and automate the compliance effort.

Design projects typically have stringent implementation requirements, which historically have been placed into three categories: timing performance, design area and power. Tools that address each category generally serve the needs of most design flows. However, even seemingly full-featured synthesis tools may fall short in safety-critical domains, which require such stringent design assurance that it should be considered the fourth dimension of FPGA synthesis requirements. If this aspect of synthesis is not well-understood and managed, designers can compromise their ability to ensure the final implementation matches the design intent. This in turn means noncompliance with...
the basic objective of DO-254 and similar standards, which is to ensure the design performs its intended function.

The design assurance features pertinent to synthesis can be broken down into six main categories, as described in the following sections.

Repeatable Synthesis Results
DO-254 and similar standards require that each step in the process be repeatable. Repeatability fosters confidence, since rerunning these same steps in the same environment should yield the same design results every time. Historically, synthesis is one step in the design flow that has commonly yielded different results even when the same tool version and settings are used. It can be understandably alarming when each synthesis run creates different internal data structures and non-repeatable results. The way around such variability, and to produce a consistent netlist time after time, is to use a synthesis tool that relies on deterministic object name generation and is extensively tested for repeatability. To ensure the same compute environment (a prerequisite for repeatability), the tool should automatically generate documentation describing the hardware, operating system and tool settings to ensure repeatable netlist generation.

Optimizations and Verifiable Synthesis Results
In any design process, if a synthesis tool is trusted then it is more or less assumed to produce a gate-level design that is functionally equivalent to the RTL written by designers and input to the tool. However, in a safety-critical process, there should be no “trust.” Instead, the output of every step and every tool should be reviewed or verified.

Synthesis performs various optimizations to reduce area and optimize performance in order to fit the design into the smallest and least expensive device. In most cases these optimizations are beneficial; however in high assurance design flows, they can make verification difficult. An optimization may improve area and performance but lead to simulation mismatches when comparing netlist to RTL test bench results.

Some synthesis flows now support a “design assurance mode,” which avoids tool settings and turns off optimizations that are not readily verifiable or may introduce simulation mismatches. Among the settings and optimizations important to turn off are the treatment of incomplete sensitivity lists, parallel/full case pragmas and four-state values (‘X’ or ‘U’ or ‘W’ or ‘Z’) for assignment or comparison.

Logic Equivalence Checking
While gate-level simulation is a generally accepted approach for large and complex designs, running the entire test bench with full timing can be incredibly time consuming. An alternative and much faster method of verifying synthesis results is using logical equivalency checking (LEC). LEC utilizes mathematical techniques to analyze one design model (in this case, the gate-level netlist generated by synthesis) against another (the thoroughly verified RTL code used as input to synthesis).
Using LEC on FPGA designs requires that the FPGA synthesis tool has an established working flow with a formal verification tool, mostly to share information about optimizations and methods for cross referencing these between the RTL and gate-level netlist. Such optimizations include merged, duplicated or inferred registers; re-encoded finite state machines; or inferred counters. Without this integration, setup for the LEC process can be tedious and subject to human error.

Requirements Tracing

DO-254 programs and the like use a requirements-driven engineering process. Requirements must be captured, validated, managed and traced to implementation and verification activities. Synthesis-related requirements, such as those that are performance-based, can be specified as constraints for the synthesis and place-and-route processes. For example, a set of synthesis constraints may implement a certain timing requirement, such as a dual-clock domain. Manually tracing these requirements down into the result files would be time consuming and could easily lack synchronization with current project results.

Some current tools can automate the process of tracing requirements to synthesis constraints, and through to the corresponding area and timing results from synthesis and place-and-route. An advanced synthesis tool can recognize requirement identifiers in the constraint definition files and insert references to these within the corresponding sections of the run reports, thereby generating automatically generated traceability reports.

Mitigating Soft Errors (Including SEUs) with Triple Modular Redundancy

Soft errors, such as single event upsets (SEUs), have become pervasive enough that companies complying with
DO-254 are now encouraged to examine and address the issue of circuitry prone to radiation effects. Upcoming policies – see for example the imminent standards ARP 4754A and ARP 4761A, for development and safety analysis of aircraft systems, respectively – are evolving to more explicitly address the concern, generally by applying one or more mitigation techniques.

The latest synthesis technologies can automatically mitigate soft errors through implementation of triple modular redundancy (TMR) circuitry. TMR is a widely accepted method of fault-tolerant design in which a unit is triplicated and fed into one or more majority voter circuits. If an upset occurs on any one unit, the majority voter(s)’ value will be the correct value and hence mask the fault.

Synthesis-based TMR can protect against both SEUs and single event transients (SETs are rogue voltage pulses that propagate through a combinational circuit) by triplicating some or all the design, depending on the design requirements. Synthesis is an ideal stage at which to make intelligent decisions about what to infer and when. For example, the synthesis tool can decide whether or not to infer embedded shift registers and how to treat synchronizers across different clock domains. Using a flow-based mitigation solution opens up device options to more SRAM, flash and antifuse devices from multiple device vendors, thus allowing project teams to more easily meet their capacity budget, and performance requirements.

**Reviewing Design Data and Tool Messages**

Design reviews and audits occur throughout DO-254 and safety-critical processes. It becomes more difficult to review the actual design data in netlist format (post-synthesis) and yet this is when much valuable information about the design is finally known. As a synthesis tool transforms the design from RTL to gates, it finds useful information like constraints, clock domains, multi-cycle paths and how the RTL code was interpreted. Instead of reviewing the netlist itself, mining design data from the synthesis process can provide a wealth of useful information. As just one example: since design assurance requires that all the warnings reported by design tools be documented and explained, all the messages reported by the synthesis tool need to be properly categorized as warnings, errors or information messages.

**Conclusion**

Safety compliance is a challenge that’s not going away for an increasing number of aerospace hardware developers, and those in many other industry segments. The ability to establish certifiable and highly productive design flows can be the difference between companies that are successful in this market and those that fall by the wayside. Design assurance is an essential element of these flows. The best bet, then, when getting started in design assurance? Pick a synthesis flow that goes beyond the usual design optimization goals and specifically focuses on those aspects of the design most important to safety-critical or design-assurance guidelines.

---

**Ehab Mohsen** (ehab_mohsen@mentor.com) is a technical marketing engineer in Fremont, Calif.

**Michelle Lange** (michelle_lange@mentor.com) is the DO-254/design assurance program manager in Wilsonville, Ore.