Deploying HLS in a DO-254/ED-80 Workflow

Tammy Reeve, Patmos Engineering Services, Washington, USA (*Tammy@patmos-eng.com*)

Jacob Wiltgen, Mentor, A Siemens Business, Colorado, USA (*jacob_wiltgen@mentor.com*)

Byron Brinson, Mentor, A Siemens Business, Texas, USA (*byron_brinson@mentor.com*)

David Aerne, Mentor, A Siemens Business, Oregon, USA (*david_aerne@mentor.com*)

***Abstract*—The adoption of tools into safety-critical workflows is often challenging as these new technologies must demonstrate sufficient safeness to use before being deployed in production environments. The demand for High-Level Synthesis capabilities within DO-254 projects is growing and this paper describes the requirements and considerations to successfully use High-Level Synthesis within a DO-254 workflow.**

Keywords—DO-254, Safety, Avionics, HLS, HLV, High-Level Synthesis, Audit;

I. Introduction

To remain competitive in a challenging market, avionics companies continue to innovate across all aircraft-related systems, including flight management, communication, navigation, and in-flight entertainment. New features and capabilities demanded on board aircraft have a direct impact on the complexity of semiconductor design and verification. Companies are continually challenged to deliver products on time and on budget, while simultaneously demonstrating a rigorous safety process adhering to RTCA DO-254 and EUROCAE ED-80 [1], the principal standards assuring airborne electronic hardware is safe for use. To meet these challenges, companies must adopt new design and verification methodologies making their engineers more efficient.

High-Level Synthesis continues to grow in adoption as it provides a shift-left enabling designers to quickly develop, test, and deploy new algorithms on hardware targets. The adoption of new technologies within safety-critical industries is often challenging as these technologies must be proven safe before allowed into production environments. The demonstration of safeness requires both the technical acceptance and education of the certification authorities. This paper describes the use of High-Level Synthesis (HLS) within a DO-254/ED-80 workflow and the evidence HLS provides.

II. High-Level Synthesis Overview

High-Level Synthesis is an automated process that interprets an algorithmic description of a desired behavior and creates digital hardware that implements that behavior. Similar to the evolution from schematic to hardware description language (HDL) such as Verilog or VHDL, HLS supports another shift left enabling rapid creation of complex algorithms that are then synthesized into functionally equivalent HDL. HLS methodology has evolved significantly over the past decade, allowing engineers to perform verification and test and to make basic design checks at the abstracted level.

Figure 1 and the following text provide a summary of a generic HLS workflow.

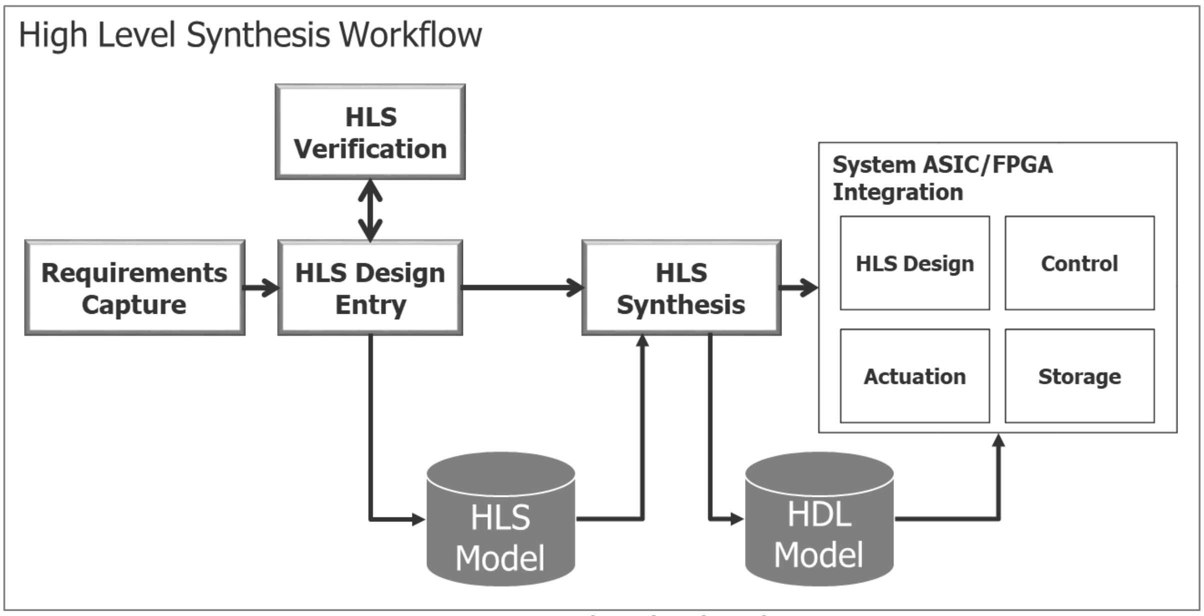

Figure 1. Generic High-Level Synthesis Flow

The first task within an HLS flow is to write SystemC/C++ describing the functional intent based on requirement specifications. After which, design checks are performed as a first round of sanity checking on the SystemC/C++ design. Similar to a traditional HDL workflow, functional verification is performed after design entry to validate functional behavior prior to HLS synthesis. Functional testing of HLS code with comprehensive coverage efficiently boosts coverage of RTL and allows designers to find holes in functional testing early to avoid surprises in RTL and find redundancies in design early to avoid excess resources. High-Level Synthesis designs are commonly leveraged to develop complex algorithmic functions within a larger system. Using a HLS model the designer can implement the hardware allocated requirements using HLS SystemC or C++ allowing the designers the ability to model the design and explore the functional operation at a higher level of abstraction.

The final step in the HLS workflow is to synthesize the SystemC/C++ into HDL. Specifying target technology and clock frequency along with additional constraints regarding loop unrolling and initiation interval (II) are used to constrain and guide synthesis. Once synthesized, traditional verification flows as applied to traditional hand-crafted HDL are performed on the generated HDL.

The resulting HDL design is then sent to backend place and route or incorporated as a module within a larger system.

It is important to note that HLS leverages libraries of previously developed functions such as math and DSP and video algorithms. The IP libraries which are not developed by the designer directly must be considered as COTS IP similarly to any libraries utilized at the VHDL or Verilog design level. CAST paper 33 [2]. as well as AMC 20-152A [4] provides the regulatory framework for how to manage and certify these previously developed library functions provided within the development environment to meet the DO-254 objectives. Developers are responsible under a DO-254 project to ensure all functions embedded in the final placed and routed device meet the objectives of DO-254 including library functions provided by HLS technologies as well as the FPGA, PLD and ASIC vendor specific libraries.

III. CHALLENGE USING HIGH-LEVEL SYNTHESIS IN A DO-254 WORKFLOW

Historically, the adoption of new technologies and methodologies face resistance in DO-254 workflows until the technologies can be properly vetted and confidence obtained. Historically, this challenge existed when moving to RTL levels of design abstraction as well as the acceptance of new functional verification methodologies. High-Level Synthesis has encountered a similar challenge. While synthesis from RTL to gates is an accepted norm today within a DO-254 workflow, the acceptance of synthesis from SystemC/C++ to RTL is often met with resistance. Additionally, the fact that designs are modeled in SystemC/C++ in an HLS environment is often interpreted as modeled base development and falling under DO-331.

The remainder of this paper will highlight the workflow, rationale, and constraints in the use of HLS in a DO-254 workflow.

IV. Deploying Mentor Catapult HLS in a DO-254/ED-80 Workflow

The Catapult High-Level Synthesis Platform empowers designers to use industry-standard ANSI C++ and SystemC to describe functional intent and move up to a more productive abstraction level. From these high-level descriptions, Catapult generates production-quality RTL and by automating the generation of bug free RTL, Catapult significantly reduces the time to verified RTL. Figure 2 below details the workflow with Catapult High-Level Synthesis.

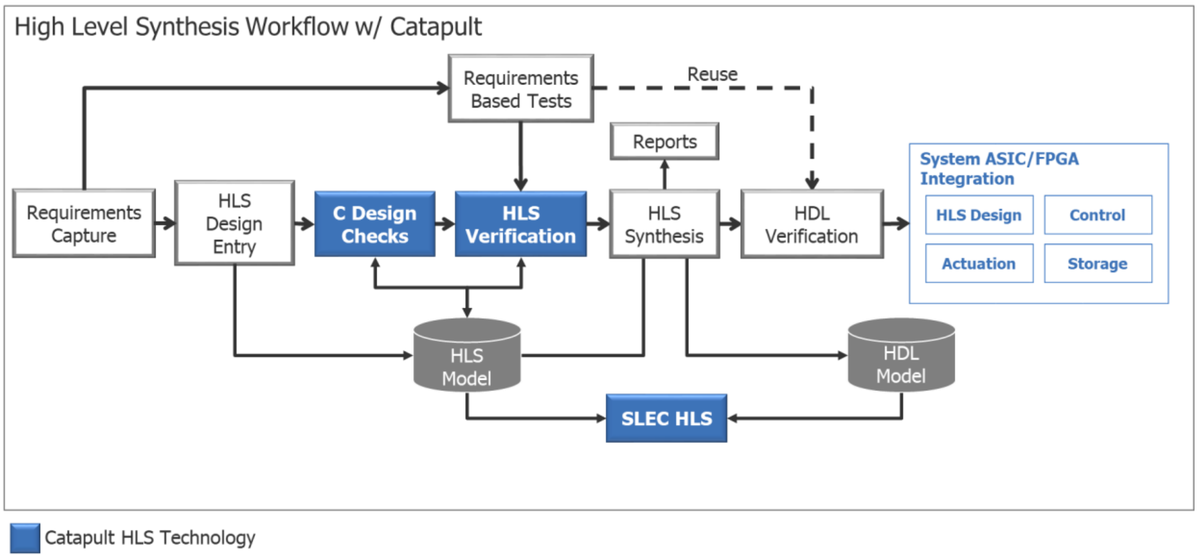

Figure 2. Mentor Catapult HLS within a DO-254 workflow

The following sections discuss how the workflow described in Figure 2 overlays with the phases of a DO-254 lifecycle.

A. *Planning*

The creation of planning information is mandatory and provides an overview of processes, procedures, and methods that will be used to develop the hardware design. The Plan for Hardware Aspects of Certification (PHAC) must describe how HLS will be used in the overall development flow and identify the version of the Mentor tool to be used and maintained. The PHAC must clearly state what objectives in DO-254 the developer plans to take credit for utilizing the HLS work flow and artifacts from these process steps defined, documented and controlled as evidence of completing these activities. Catapult HLS also provides functional verification facilities, referred to as High-Level Verification (HLV). HLV encapsulates design standard checking, achieves functional coverage goals, and formalizes the post RTL generation equivalency with the HLS model. If any of these aspects of the tool flow are utilized, tool assessment considerations must be addressed based on the design assurance level assigned and guidance in DO-254 section 11.4.

B. *Requirements Capture*

Similar to a traditional HDL flow, requirements represent the input to a flow leveraging HLS for design entry. In addition to English-language textual requirements, it is also common for models or formal methods to be provided to express requirements. In the event models or formal methods are leveraged, this approach must be clearly defined in the plans. Developers are advised to look for guidance from RTCA DO-331 and DO-333, which are for software development but provide good insight as to how to control requirements when in these forms as well as guidance on standards.

C. Conceptual Design

The objective of the conceptual design phase is to produce an artifact that provides an architecture and design description showing how the designer intends to meet the requirements. Additional, the delivery includes how the designer plans to implement the design. The conceptual design is commonly a data flow diagram or functional block diagram. HLS supports satisfying objectives in DO-254 Conceptual Design by ensuring the hardware item conceptual design is consistent with the requirements defined in DO-254 5.2.1[1]. HLS SystemC or C++ can be considered for DO-254 conceptual design if documented and commented properly.

D. Detailed Design

High-Level Synthesis provides a number of benefits above traditional RTL verification. Due to the higher level of abstraction, there are fewer lines of code, thus permitting testing to begin sooner plus the increased performance resulting from simulating at a higher level of abstraction. Finally, if done properly, the investment in tests and testbench development can be re-used for verification of the post-HLS RTL. DO-254 explicitly states HDL as one element of the detailed design data required during this phase with HDL being defined as all hardware description languages. Therefore, SystemC/C++ is considered a hardware description language and satisfies the detailed design objectives for ensuring the detailed design is developed per the hardware item requirements and conceptual design data (DO-254 5.3.1).

Similar to RTL, a code review must be performed on the HLS design source (SystemC/C++). This is done to ensure proper coding techniques and guidelines are followed. Traceability from the detailed design to the requirements is also required.

1) Enhanced Detailed Design evidence provided by Catapult HLS

Catapult Design Checker performs multiple types of design checks on the design source (SystemC/C++) to confirm proper coding techniques and guidelines are followed. These checks are a combination of lint, static, and formal-based checks and rules that identify ambiguities in the HLS design source.

E. Implementation

Per DO-254 section 5.0, implementation is the fabrication file or the loadable image file used for programming the final target device. In the HLS flow, RTL synthesis and HLS synthesis have a similar workflow and the artifacts generated from this phase are similar as well. Activities such as synthesis constraints and log file review will be performed for the implementation phase. However, the content of the artifacts will be slightly different since the synthesis is from SystemC/C++ to RTL, and then to the final program file.

1) Enhanced implementation evidence provided by Catapult HLS

A series of reports are generated during Catapult HLS synthesis detailing the commands exercised. This includes a cycle report detailing latency and throughput. Additionally, timing reports ensure timing closure on the target technology and also definition of worst case timing slack. In addition, a Bill of Materials (BoM) report is generated detailing the resources used for the target technology library, area estimation, and a list of FSMs inferred during synthesis. Lastly, for HLS enthusiasts, a Gantt chart details the scheduling decided upon during synthesis and indicates which operators are active on which clock cycle.

F. Configuration Management

The configuration management process for the all of the artifacts and data items produced in the HLS flow are identical to a typical HDL flow. It is important for DO-254 compliance that all aspects of Catapult HLS and it's libraries be documented and controlled for the life of the product in service to ensure continued airworthiness aspects related to the design and verification of avionics parts generated from this flow.

G. Verification

Functional simulation of HDL is a common and widely-accepted practice within current DO-254/ED-80 workflows. The primary objective of verification is to ensure the hardware item or function that has been designed behaves in accordance with the specified requirements. The objective of requirement-based simulation is to verify that the requirement is satisfied within the design under test given a set of input stimulus. This objective remains the same, regardless of whether the code under test is Verilog/VHDL or HLS SystemC/C++. Most HDL simulators now support mixed language simulation enabling simulation of designs created with VHDL, Verilog, SystemVerilog, and/or SystemC/C++.

Requirements based tests can be executed on the High-Level Synthesis design and provides benefits from traditional RTL verification by allowing you to validate the implementation sooner, faster, and quickly identify gaps in requirements. HLS verification utilizing requirements based test enables code coverage metrics to ascertain functional requirements have been met.

When the developer in a DO-254 development flow considers the SystemC/C++ as the design/source code, an HLS testbench can be used to collect functional and code coverage metrics, which are forms of elemental analysis defined in DO-254 appendix B and a requirement for DAL A and B designs. For DO-254 credit, attention needs to be taken that the verification aspects meet the design assurance level independence requirements of DO-254. In both a traditional RTL and SystemC/C++ flow, the testbench for design assurance level A or B must be independently developed, reviewed or executed. Coverage analysis of the HLS design source is additional evidence regarding the verification of the RTL output from the HLS flow. The verification metrics captured during HLS functional verification can also be applicable to the RTL that was generated if appropriate tool assessment measures are taken.

One measure is utilizing sequential logical equivalency (SLEC) between the HLS source code and the output RTL. Establishing equivalence allows the user to claim code coverage credit at the RTL level. The use of formal technology (SLEC) is not new to the electronic hardware certification space and has been identified as an advance verification method in Appendix B (Section 3.3.3) of RTCA DO-254/ED-80[1], detailing safety specific analysis and formal methods. SLEC, a formal technology, is used to provide additional design assurance and evidence regarding the conversion of the HLS design source to HDL. In Catapult HLS, functional equivalency is performed between the original C++ and the generated HDL. The capability demonstrates HLS synthesis did not alter the functional intent and that the generated HDL is equivalent to the HLS design source. Specifically, SLEC checks that the HLS design source and HDL are logically equivalent and also sequentially equivalent, meaning whatever data path delays or pipelining that was done in the HLS design source is also present in the HDL.

Concerns related to tool assessment that arise from utilizing SLEC may require a second means of assessment. Executing the same requirements based tests on the RTL level design that were executed on the SystemC/C++ provides that second approach. Catapult SCVerify enables testbench reuse to validate functional equivalence between the SystemC/C++ and generated RTL. This capability is extremely powerful as it leverages the HLS test environment infrastructure, executes the same test vectors on the post synthesis HDL, and provides HDL functional and code coverage metrics. This reuse of existing testbench infrastructure provides an immediate form of independent assessment between the SystemC/C++ and generated RTL.

Figure 3 below provides a graphical overlay of the two forms of assessment described previously on the already defined Catapult HLS workflow. #1 represents the reuse of the requirements based test environment and validates that the intended functionality was not altered during RTL creation. #2 represents the assessment of SLEC HLS which confirmed sequential logic equivalency between the SystemC/C++ and post-synthesis RTL is validated.

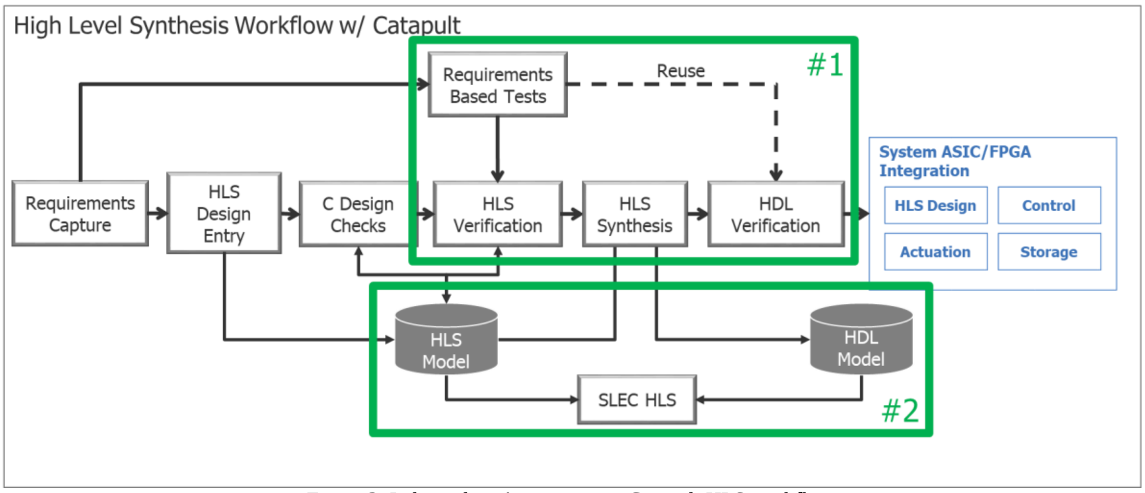

Figure 3. Independent Assessment in Catapult HLS workflow

It is important to note that the functional verification results must be retained and controlled as a verification output for DO-254 compliance. If SLEC is utilized, outputs of equivalence checking must be recorded, retained, and controlled as a verification output for DO-254 compliance.

H. Target Testing

For all designs it is important the final placed and routed output is integrated into the device vendor specific platform, ensuring the device functions according to the defined requirements.

V. CONCLUSION

The complexity of avionics systems continues to grow non-linearly and companies must continue to adopt new technologies to make programs more efficient without sacrificing the safety intent. Similar to the migration from schematic to RTL, High-Level Synthesis represents an evolution to how designs are created. HLS now include advanced verification capabilities, allowing for application of known and trusted verification techniques at the HLS level of abstraction. Thus verification of the design can begin sooner, thus ensuring product requirements have been met and also re-use of this working verification environment for performing needed verification on the post-HLS RTL. In the context of DO-254, Catapult HLS delivers a number of artifacts and supplemental evidence which provides additional confidence to regulators ensuring the design operates per the requirements.

ACRONYMS

Abbreviation	**Explanation**
HLS	High-Level Synthesis
HLV	High-Level Verification
SLEC	Sequential Logic Equivalence Checking
PHAC	Plan for Hardware Aspects of Certification
HDL	Hardware Description Language
RTL	Register Transfer Level
FPGA	Field Programmable Gate Array
PLD	Programmable Logic Device
ASIC	Application Specific Integrated Circuit

REFERENCES

[1] RTCA DO-254 / EUROCAE ED-80, Design Assurance Guidance for Airborne Electronic Hardware

[2] FAA CAST-33: Compliance to RTCA DO-254/ EUROCAE ED-80, “Design Assurance Guidance for Airborne Electronic Hardware”, for COTS Intellectual Property Used in Programmable Logic Devices and Application Specific Integrated Circuits

[3] RTCA DO-331, Model-Based Development and Verification
[4] AMC 20-152A Development Assurance for Airborne Electronic Hardware (AEH)