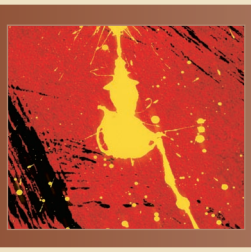
Software Tools for Safety-Critical Systems According to DO-254

Andrew Kornecki and Brian Butka
Embry-Riddle Aeronautical University
Janusz Zalewski
Florida Gulf Coast University



Software and hardware components of safety-critical systems must be developed in a unified manner.

In recent decades, safety has emerged as a major issue in many embedded applications in the aerospace, aircraft, automobile, railways, nuclear, medical, and other industries. Safety in this context means avoiding harm to individuals or society due to malfunctioning computer equipment or software. The essential requirements for these systems are so strict that they are regulated by government agencies such as the US Federal Aviation Administration (FAA) in the case of both airborne and ground aviation systems.

The general concept of safety assurance is to minimize risk that can lead to accidents. This implies that the software tools used to develop the hardware and software components in safety-critical systems must be evaluated as thoroughly as the products themselves.

MOTIVATION

Modern safety-critical systems use not only increasing numbers of microcomputers and microprocessors but also dedicated hardware to process the growing amounts of data needed to control the systems and monitor their status.

These complex programmable electronic devices are developed using conventional programming languages as well as hardware description languages to create logic designs. Developers use software tools to simulate the logic, synthesize the circuits, and place and route the electronic elements and their connections prior to final implementation. Primary components in such designs include programmable logic devices (PLDs), field-programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), and similar circuits.

Creating complex circuits is currently not considered a software activity and is left to hardware specialists. However, hardware description languages such as VHDL, Verilog, and System C are basically computer languages with their own syntax and semantics.

Hardware development relies heavily on the quality of tools that translate the software artifacts from one form to another. Both software and hardware designers use integrated programming environments with complex tools to take a project from initial concept to final product.

Thus, a critical issue for software engineering as it relates to the development of dependable safety-critical systems is the close relationship between the traditionally separate categories of software and hardware.

Software application designers who focus on developing programs to run on microprocessors often overlook PLDs and most popular FPGAs. An FPGA is a prefabricated integrated circuit that can be configured to implement a particular design by downloading a sequence of bits. In that sense, a circuit implemented on an FPGA is literally software.

In a recent *Computer* article, Frank Vahid pointed out that treating circuits as "hardware" poses problems in computing system development, especially for embedded systems ("It's Time to Stop Calling Circuits 'Hardware,'" Sept. 2007, pp. 106-108). Along those lines, we are convinced that software and hardware components in safety-critical systems must be developed in a unified manner.

CERTIFICATION CONCERNS

Developers of dependable, safety-critical systems must meet certain government regulations and engineering standards.

For example, the RTCA (www.rtca.org), formerly known as Radio Technical Commission for Aero-

nautics, provides certification guidelines for software in airborne systems installed on civilian aircraft in DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*.

On the other hand, RTCA DO-254, *Design Assurance Guidance for Airborne Electronic Hardware,* details certification procedures for hardware components used in safety-critical avionics systems from project conception to planning, design, implementation, testing, and verification. It also defines safety assurance levels for such systems ranging from A (most critical) to D (least critical).

DO-254 applies to a wide spectrum of hardware ranging from integrated technology hybrid and multichip components, to custom programmable microcoded components, to circuit board assemblies, to entire line-replaceable units. It also discusses commercial off-the-shelf components. However, FAA Advisory Circular AC-20-152 formally only requires DO-254 to be applied to ASICs, PLDs, and FPGAs.

The two RTCA documents address the qualification of tools used to create a certified airborne system in a slightly different way. It is thus conceivable that a system with a specific level of safety assurance will receive different scrutiny depending on whether it is implemented in software or hardware.

The concern is that a designer can freely implement a task using a mixture of hardware and software, but current tool qualification guidelines do not effectively deal with this type of integrated design. To ensure proper treatment of software processes in the development of complex electronic devices, there is a need to unify such procedures for software and hardware development.

In safety-critical applications with millions of gates on a chip, the role of hardware design tools and verification tools is especially critical.

Along with the process of developing complex electronic hardware for airborne applications, DO-254 describes the tool qualification process. As Figure 1 shows, Section 11.4 distinguishes between design and verification tools: "When design tools are used to generate a hardware item or the hardware design, an error in the tool could introduce an error in the hardware item; when verification tools are used to verify the hardware item, an error in the tool may cause the tool to fail to detect an error in the hardware item or hardware design."

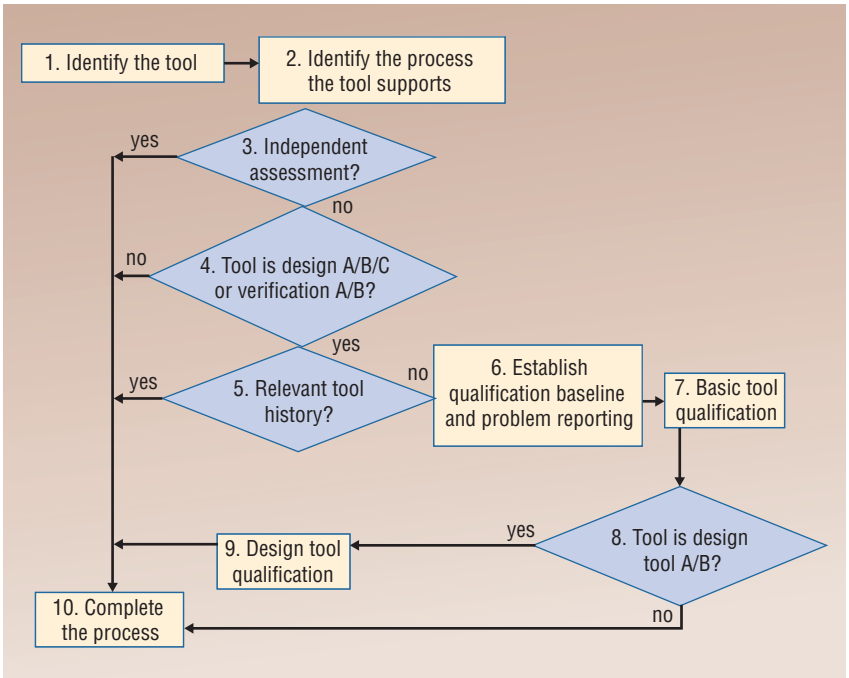


Figure 1. Tool qualification procedure according to RTCA DO-254. The standard distinguishes between design and verification tools.

INDUSTRY SURVEY

The DO-254 Users Group website (www.do-254.org) provides a good starting point for understanding industry practices in the area of tool qualification and compliance with DO-254 guidelines. Some vendors point out that the lack of research investment in certification technologies will significantly impact autonomous control approaches that can be properly flight certified and could limit the capability of future autonomous systems.

To help clarify the underlying issues in tool qualification and system certification, we surveyed industry and certification authorities on the use of programmable logic tools to design and verify complex electronic hardware according to the DO-254 guidelines.

We developed a questionnaire (www.do-254.org/?p=tools) and first distributed it to the more than 200 participants at the FAA National Software and Complex Electronic Hardware Conference in New Orleans, Louisiana, in July 2007. We also gave the questionnaire to tool experts at the November 2007 meeting of the Programmable Logic Users Group in Clearwater, Florida, and at the May 2008 Integrated Electrical Solutions Forum in Lake Buena Vista, Florida. In addition, we mailed the questionnaire to more than 150 individual aviation software and hardware developers and to a few companies that design programmable logic devices.

We received almost 40 complete responses from these efforts. Although this sample is not statistically significant, the results lead to some interesting observations.

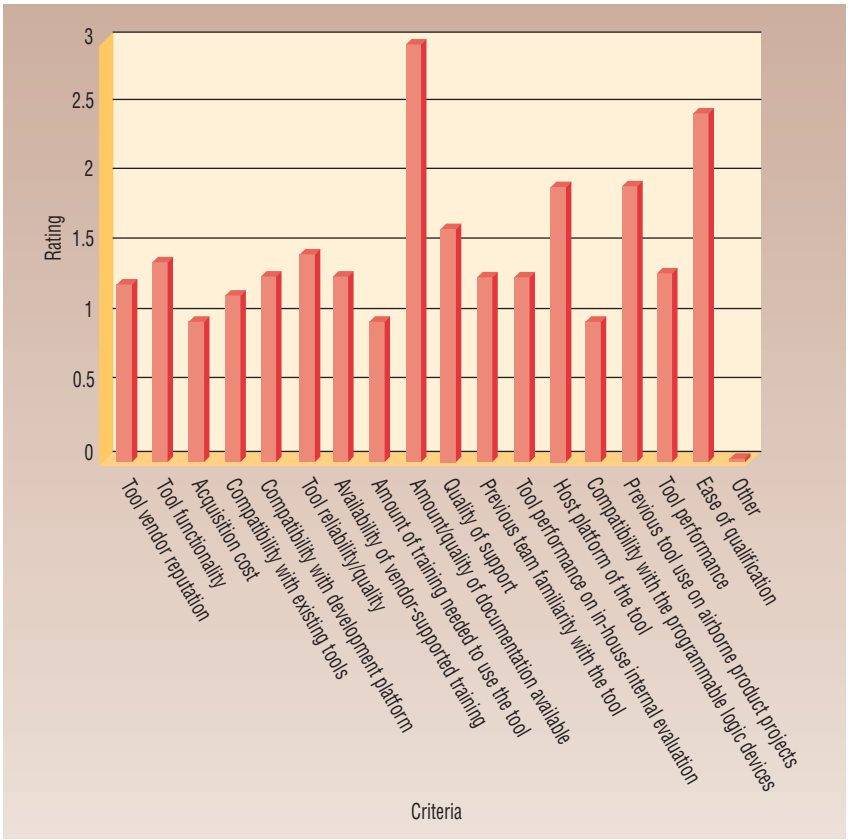


Figure 2. Tool selection criteria in DO-254 projects. The most important criteria for survey respondents were available documentation, ease of qualification, and previous tool use.

Survey population and tool use

Sixty-five percent of survey respondents were avionics or engine control developers. More than 95 percent had a technical education, with 55 percent holding a bachelor's degree and 45 percent a master's degree. Over 72 percent of respondents had professional expertise in electronics: Ninety-seven percent had more than three years of experience, and 59 percent more than 12 years.

Sixty-two percent of respondents used programmable logic tools to develop or verify systems, 26 percent were either managers or their company's designated engineering representative, 2 percent were tool developers, and 12 percent developed components. The respondents' research focus was divided among verification (32 percent), development (27 percent), hardware (22 percent), and concept/architecture (18 percent).

The respondents used a wide range of devices in their work. The most frequent were FPGAs (27 percent), followed by complex PLDs (18 percent), ASICs (15 percent), programmable array logic (11 percent), programmable logic arrays (9 percent), and electronically programmable logic devices (8 percent).

The most popular hardware device vendors in the survey were Actel (27 percent), Xilinx (24 percent), Lattice Semiconductor (13 percent), and Cypress Semiconductor (11 percent), with QuickLogic, Altera, and Atmel each below 10 percent. The most widely used tools were from Mentor Graphics (27 percent), Synplify (22 percent), Synopsys (17 percent), Aldec (11 percent), and Cadence Design Systems (8 percent). The remaining respondents used other tools.

Tool selection criteria in DO-254 projects

Figure 2 shows the respondents' criteria for selecting tools for DO-254 projects. The most important were available documentation, ease of qualification, previous tool use, and host platform, followed by quality of support, tool reliability, tool functionality, and tool vendor reputation.

The basis for selecting a tool was primarily either familiarity with the demo version (50 percent) or an extensive review and test (40 percent). The prevailing approach to reviewing and testing the tool was to train personnel and use it for a trial period on a smaller project.

Of the 14 percent of survey respondents with experience qualifying programmable logic tools for DO-254 projects, 62 percent regarded the guidelines' quality as sufficient or appropriate and 67 percent cited the ease of finding required information, while 80 percent considered the increase in workload as negligible or moderate.

While 43 percent of respondents observed that the safety improvement due to certification was marginal, 21 percent identified it as moderate, 7 percent as noticeable, and 29 percent as significant. Another cause for concern is that only 11 percent of respondents found no errors in the tools they used, while 50 percent reported few and minor errors and 17 percent significant and numerous errors.

Nevertheless, overall satisfaction level regarding programmable logic tools was high: More than 96 percent of respondents rated their satisfaction level as 4 out of 5.

Looking at the survey results, we believe that researchers must develop more objective criteria for tool certification for DO-254 projects and conduct experiments to identify vulnerable tool functions that could be a source of subsequent design faults and operational errors.

The tool assessment process must follow DO-254, but the guidelines' relative vagueness causes significant differences in interpretation by vendors. Common ground should be found between DO-254 and DO-178B guidelines.

As a step in this direction, we are developing a set of experiments that test development tools with respect to three primary error-prone features in FPGA circuits: switching noise in case a large number or all of the signals go from 0 to 1 or from 1 to 0 simultaneously, unused I/O pins that might have been left unspecified in the design files, and accuracy of meeting timing constraints. We presented preliminary results in August 2008 at the National Software and Airborne Electronic Hardware Standardization Conference in Denver.

Andrew Kornecki is a professor in the Department of Computer and Software Engineering at Embry-Riddle Aeronautical University, Daytona Beach, Florida. Contact him at kornecka@erau.edu.

Brian Butka is an associate professor in the Department of Electrical and Systems Engineering at Embry-Riddle Aeronautical University. Contact him at butkab@erau.edu.

Janusz Zalewski is a professor of computer science and engineering in the School of Engineering at Florida Gulf Coast University, Fort Myers, Florida. Contact him at zalewski@fgcu.edu.

Editor: Mike Hinchey, Lero—The Irish Software Engineering Research Centre; mike.hinchey@lero.ie