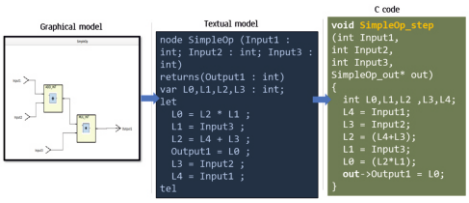
Safety-Critical C&I Systems

2

SPERTS: A Tool for Development of Safety-Critical Systems

Suraj Mukade*[1], Pratibha Sawhney[1], Prateek Saxena[1], Ashutosh Kabra[1], Amol Wakankar[1], Ajith K. J.[1], S. T. Sonnis[1], D. A. Roy[1] and Anup Bhattacharjee[2]

[1]*Reactor Control Division, Bhabha Atomic Research Centre, Mumbai 400085, INDIA*
[2]*Security Electronics & Software Systems Division, Bhabha Atomic Research Centre, Mumbai 400085, INDIA*



SPERTS Code Generation

ABSTRACT

SPERTS (Safe Programming Environment for Real-Time Systems) is an integrated environment for development of safety-critical systems. Using SPERTS, a developer can build a formal model of the system by specifying the user/system requirements using a higher level graphical language suitable for safety-critical system. The SPERTS language supports function blocks and state-machine constructs with rigorous semantics. A model built using SPERTS i) can be formally verified, ii) helps generate code, which is correct-by-construction and finally iii) produces a deployable code using target specific library. This article presents the SPERTS environment focusing on the enabling techniques behind application development, verification of requirements, validation of the safety properties and automated code generation including test cases.

KEYWORDS: *Safety-critical systems, SPERTS (Safe Programming Environment for Real-Time Systems), Nuclear power plants (NPPs), Building blocks, Model builder*

Introduction

Safe Programming Environment for Real-Time Systems (SPERTS) is a tool, which is developed to design, implement, and verify control systems for safety-critical applications. This tool has been developed at Reactor Control Division, BARC. What makes SPERTS different from conventional programming environment supporting graphical language is its formal (mathematical) model-based development approach.

The key features of the tool include the following.

I) An integrated development environment, which provides a) a graphical modelling tool, b) a code generator, and c) a simulation facility allowing developers to quickly design and verify control systems.

ii) The code generated by SPERTS is designed to be highly reliable, making it suitable for safety-critical applications. This is because of a) the formal model with mathematically defined precise semantics that is built out of the application program (user specifications) and b) the code is generated automatically from the requirements (user specifications as formal model), which is *correct-by-construction*.

iii) It provides tools for verifying the correctness of control systems along with the tools for testing, debugging, formal verification and static analysis. These tools are used to ensure that control systems meet the required safety standards and operate as intended.

iv) SPERTS enforces adherence to Programming Guidelines (PG) by a qualified code generator and built-in logical constructs that are safe to use in safety critical applications.

v) It ensures deterministic response time due to synchronous data flow constructs in the language[1].

*Author for Correspondence: Suraj Mukade
E-mail: suraj@barc.gov.in

The additional features of SPERTS that make this tool unique are automatic test-cases and report generation following standard design template.

Thus, SPERTS in not only an import substitute for highly expensive tools like SCADE [2], but it comes with enhanced features.

The Genesis

SPERTS is designed to provide a programming environment for Diverse Modular Safety Platform (DMSP), which is a computer based configurable platform consisting of a set of dedicated and qualified hardware and software components for use in safety applications of nuclear power plants (NPPs). Using SPERTS, this platform is configured and programmed with system specific application software to build different safety-critical systems.

To this end, SPERTS is built to run on open-source LINUX platform to generate code for safety applications. SPERTS has in-built support for integrating qualified hardware modules such as Input and Output boards through their board interface libraries, thus enabling singleclick hardware-software integration.

Application Development

SPERTS facilitates development for safety-critical applications by following the *V*−model of software development life-cycle (SDLC) conforming to IEC 60880 as shown in Fig.1.

The phases for detailed software design, coding, module integration and testing are integrated into the automated processes supported by SPERTS. All these features help in making the design, development, testing, integration and qualification of applications/systems (developed on DMSP) easier and faster by virtue of its trustworthiness by design.

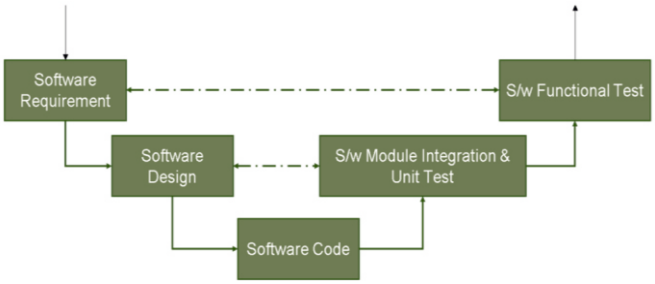


Fig.1: V-Model of SDLC.

Building Blocks of SPERTS

The essential modules that constitute SPERTS to provide an environment for building safety-critical applications using DMSP hardware are discussed briefly in this section. The structural relationship of the SPERTS modules, which help realize the goal of using formal methods of software development through high level user specification is presented in Fig.2. In addition, conformance with the standard development practices[3] is also ensured by design.

Model Builder (MB)

It is a GUI based environment that facilitate building a formal model for application logic using graphical artefacts as well as textual specifications.

It provides seamless integration of state machine and data-flow equations[4] to carry out the model development of software-based control systems and allows hierarchical model of the system i.e., development of system model by composing the models of its sub-components.

Each component of the model is denoted as a node. It is possible to reuse existing nodes of the model for specifying the other higher level nodes. SPERTS supports hierarchical state machines, which allows the designers to easily develop complicated applications in a modular fashion.

The Configuration Module (CM) in the Model Builder facilitates application specific configuration, which allows user to add the required hardware modules, specify the fail-safe state of each input/output, network parameter for communication modules, cycle time of the system etc.

Code Generator (CG)

The first step in automatic code generation involves production of textual representation of the graphical model of the application program. The textual model, generated by Model Builder, is then fed to the syntax and semantic checker module (SSC-CG) of CG, which checks the syntactic and semantic correctness of the textual model. This is followed by generation of an intermediate representation of the textual model in the form of an Abstract Syntax Tree (AST). Subsequently, it generates a target independent C program from the AST, which is semantically equivalent to the functional behaviour of the textual model[5,6]. The code generation process is shown in Fig.3. This generated code is compiled along with required dependencies (such as BSP, OS, system libraries etc.) to produce an executable for deployment on the target safety system.

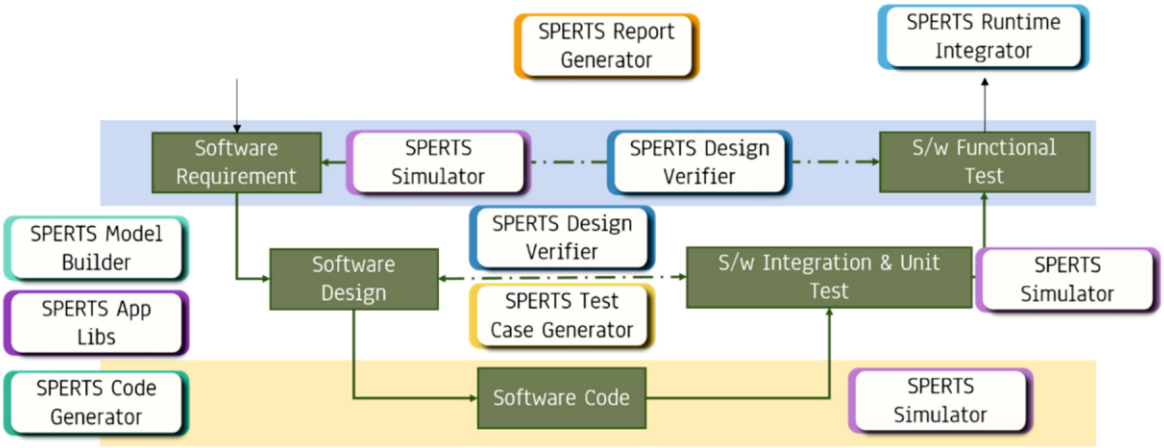


Fig.2: Use of SPERTS modules in SDLC.

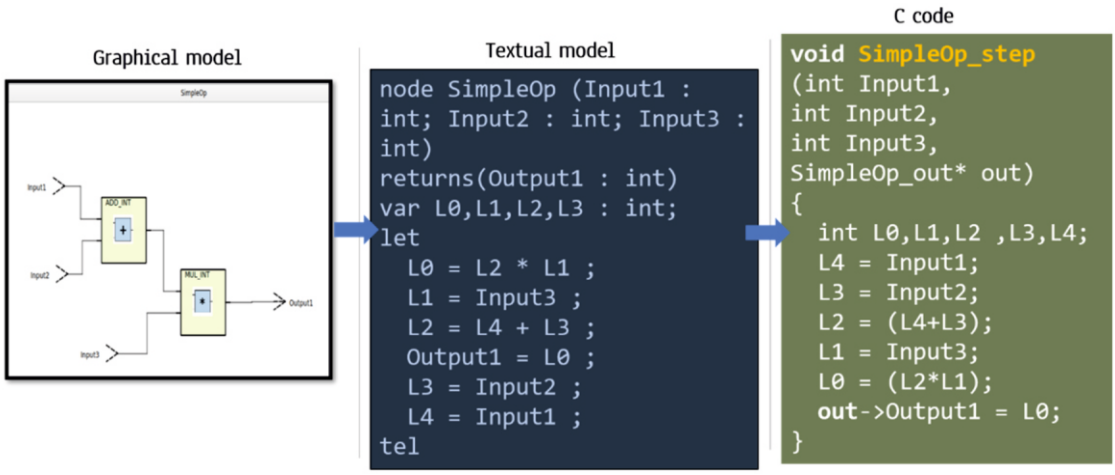


Fig.3: SPERTS Code Generation.

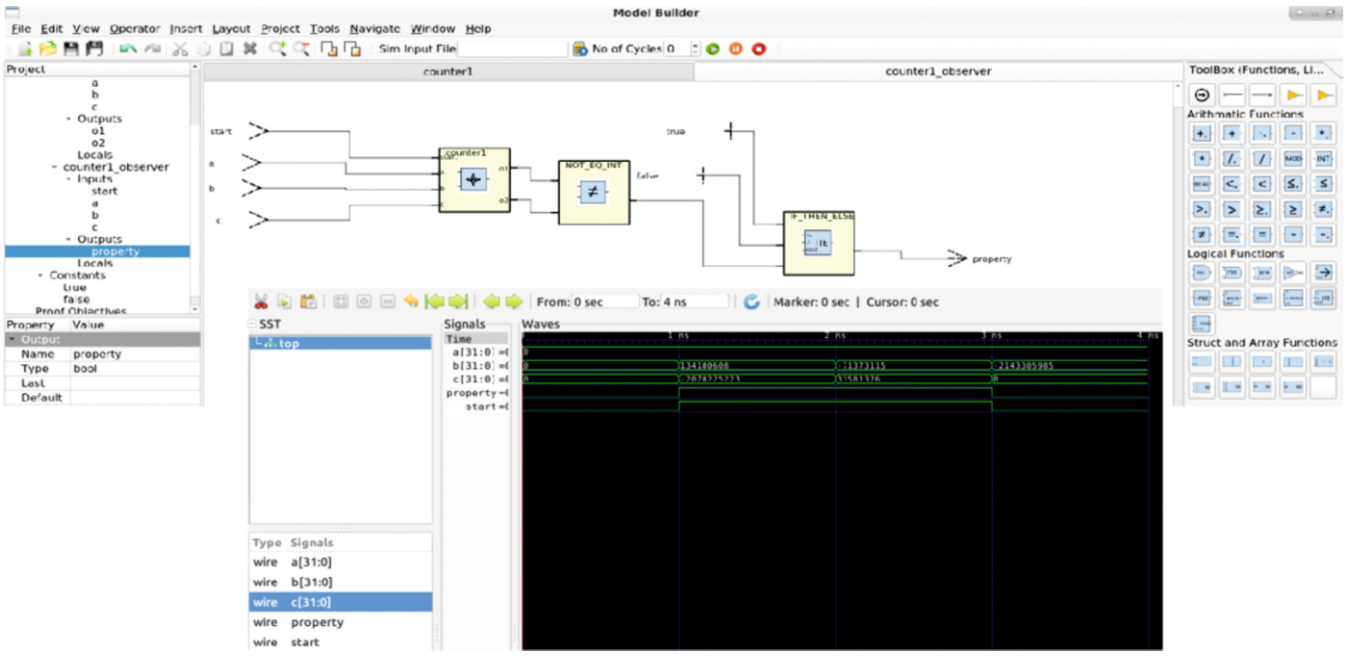
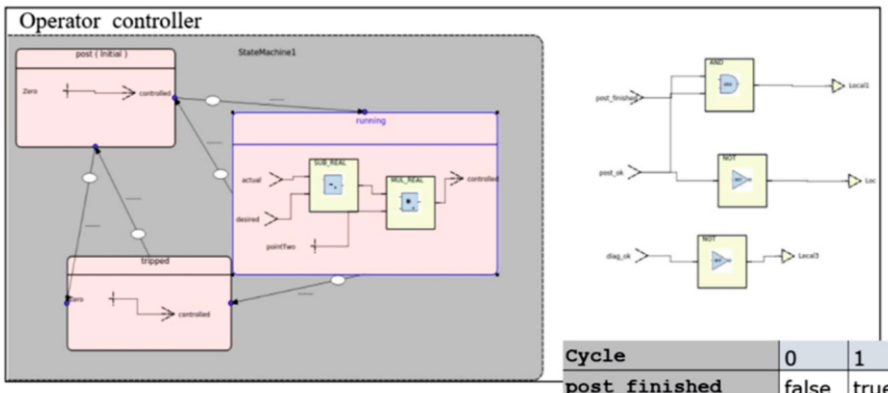


Fig.4: SPERTS Design Verifier.



Cycle	0	1	2	3	4	5	6
post_finished	false	true	true	true	true	true	false
post_ok	true	true	true	true	true	true	false
reset	false	false	false	true	false	false	true
diag_ok	true	false	true	true	false	true	false
actual	10.0	10.0	10.0	10.0	10.0	10.0	10.0
desired	5.0	5.0	5.0	5.0	5.0	5.0	5.0
Branches Covered	¬T1, ¬T2	T1	¬T3, ¬T4	T1, T3	T4	¬T5	T2,T5

Fig.5: Test case generation module of SPERTS.

Code generation for Simulation

Simulation is essential to validate the user developed model against any run-time logical error that may manifest. CG module can also be configured to facilitate generation of instrumented C code to support simulation. The instrumented C code exposes the internal state of the application at run-time which enables debugging and validation.

Model Simulator (MS)

This module supports validation of the model by simulation. Model Simulator (MS) enables debugging and simulation of models by executing the instrumented C code. CG module can be configured to generate instrumented C code for both on-host and on-target simulation. The instrumented C code makes the internal state of the application observable by outputting inputs, outputs and internal variables of each node of the application. The CG module for simulation also generates the wrapper function main for compilation and generation of executable. During the execution of the instrumented C code, the observable information is provided to the MS in a format agreed between CG and MS. MS module then graphically displays the state of the application at every instance of execution.

Design Verifier (DV)

This module of SPERTS provides support for Formal Verification using Model Checking technique. A Model Checker is a tool which takes the program and the desired property as inputs and exhaustively searches for possible violations of the property. Note that model checking is fully automatic and requires no human interaction. The main advantage of this technique is that in case the program violates the property; it is reported as the sequence of inputs which led to the violation. This is called as the counterexample, which is an execution trace leading to a state where the property is violated. By

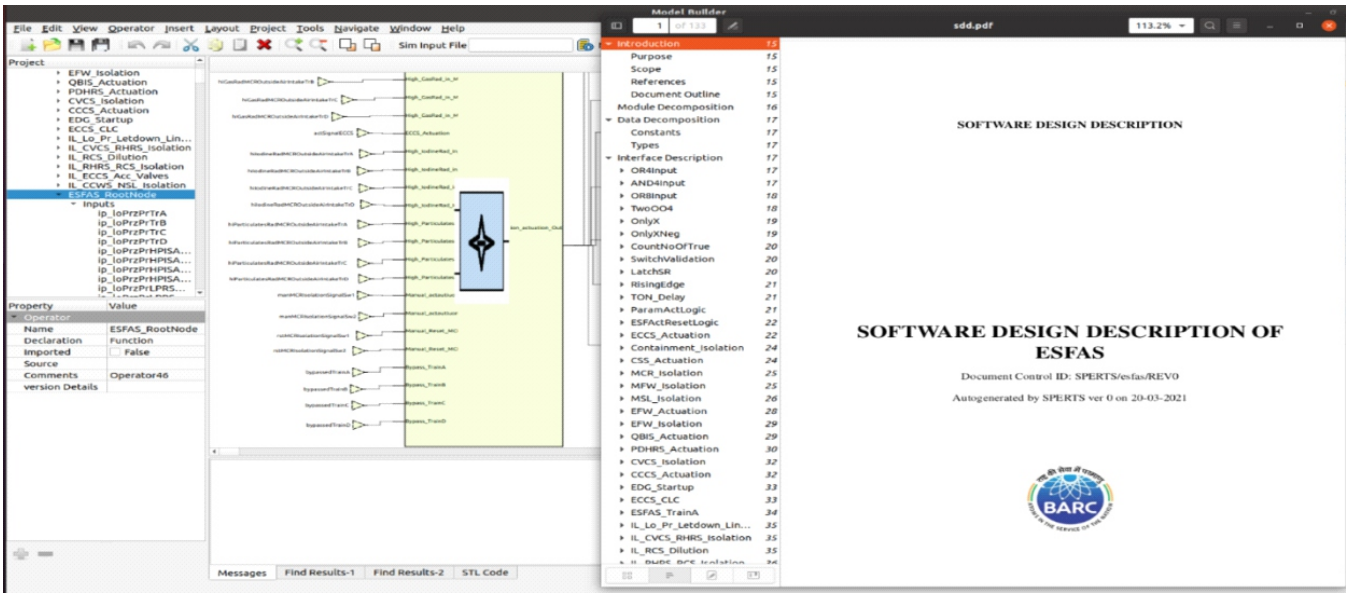


Fig.6: Report Generation Module of SPERTS.

analysing the counterexample, the source(s) of error in the model can be found. By ensuring that the model satisfies safety properties, we increase our confidence in the correctness of the model. The interface of Design Verifier with SPERT MB is presented in Fig.4.

Qualified Application Library (QAL)

This module provides the commonly used set of operators as pre-developed libraries for faster development of application. The qualified group of library operators that are supported by SPERTS QAL are i) logical operators, ii) arithmetic operators, iii) selection and look-up table, iii) trigonometric operator, iv) timer and counting operators, v) triggers, vi) bitwise operators, vii) PID operator, and viii) alarm processing operators.

Automatic Test Case Generator (ATG)

This module facilitates automatic generation of test cases for structural coverage criteria such as branch coverage and statement coverage. The generated test suite can be executed to determine and report the percentage of coverage achieved with respect to the specified coverage criteria. The generated test-cases are encoded as simulation scenarios which are accepted by the Model Simulator module. This scheme is shown in Fig.5.

Report Generator (RG)

Automatic report generation is a useful feature of SPERTS, which takes SPERTS model as input and generates a report describing the design of the application under development. The Report Generator (RG) creates the design documentation in a customized format to the comply with recommended design template as shown in Fig.6. The report is generated as PDF or HTML document.

Conclusions

SPERTS, the integrated development environment for safety-critical applications is presented in this article. The advantages of mixed mode programming using qualified graphical function blocks as well as state machine are also explained. The automated code generation, which guarantees to satisfy the specified safety properties makes this tool most suitable for Diverse Modular Safety Platform in developing safety applications. Furthermore, automated test-case generation facility makes this tool unique among the available ones.

Acknowledgements

The authors thank U. W. Vaidya, Head RCnD, S. Mukhopadhyay, Director, E&IG, BARC and P. R. Patil, former Head RCnD, for extending their support in this endeavour. The authors also thank G. Karmakar for providing technical support in writing the article.

References

[1] N. Halbwachs, P. Caspi, P. Raymond and D. Pilaud, The synchronous data flow programming language LUSTRE, Proceedings of the IEEE, Sept., 1991, 79(9), 1305-1320. doi: 10.1109/5.97300.

[2] Ansys SCADE Suite: Model-Based Development Environment for Critical Embedded Software. https: //www.ansys.com/en-in/products/ embedded-software/ansys-scade-suite.

[3] IEC 60880:2006, Nuclear Power Plants Instrumentation and Control Systems Important to Safety - Software Aspects for Computer-Based Systems Performing Category A Functions, 2006.

[4] A. Benveniste, T. Bourkey, B. Caillaud and M. Pouzet, A hybrid synchronous language with hierarchical automata: Static typing and translation to synchronous code. In Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT), Taipei, Taiwan, 2011, 137-148. doi: 10.1145/2038642.2038664.

[5] Dariusz Biernacki, Jean-Louis Colac¸o, Gr´egoire Hamon, and Marc Pouzet, Clock directed modular code generation for synchronous data-flow languages, Proceedings of the 2008 ACM SIGPLAN-SIGBED conference on Languages, compilers, and tools for embedded systems, 2008, 121–130.

[6] Auger, C., Colaçob, J. L., Hamon, G., & Pouzet, M., A formalization and proof of a modular Lustre compiler, 2012.