MODELING AND SIMULATING TIME-DEPENDENT SYSTEMS PRACTICAL EXPERIENCE

K. Benslama *, R. Bonino *
University of Geneva, Geneva, Switzerland

J. Casas-Cubillos, L. Pregernig, A. Racz
CERN, Geneva, Switzerland

ABSTRACT

Tools have become available to model and to simulate time-dependent systems at the specification level. The purpose of this paper is to describe the practical experience which has been gained from using such a tool. After a brief description of the tool, practical examples will be presented, and conclusions based on that experience will be drawn.

INTRODUCTION

Researchers at the European Laboratory for Particle Physics (CERN) use complex systems to study the structure of matter and forces in nature. In general, these systems fall into one of two broad categories: control systems and data-acquisition systems. Systems of the first category are used to control all particle-accelerator equipment, like thousands of magnets, vacuum pumps, radio-frequency equipment, etc.. In high-energy-physics experiments one will find large data-acquisition systems to collect data from hundreds of thousands detector channels, to filter the data, and to record them for further analysis.

The objective is to model and to simulate such time-dependent systems at the system level, before proceeding to detailed system design or building the systems. Ideally, a system model is implementation independent and describes the system in the problem domain. Subsequent simulation allows to study the system's dynamic behavior. When the simulation shows that the system model exhibits the expected behavior, one can take this executable system specification as a reference for implementing the actual system, mapping the problem-domain model into the application domain.

The purpose of this paper is to describe the practical experience which has been gained so far in using a system modeling and simulation tool. The following chapter will give a brief overview of the tool that has been used. Then examples of system models and their simulation will be described and, finally, some conclusions will be drawn.

TOOL

The experience described in this paper has been gained with the commercially available product Foresight [1]. Foresight is a tool-set that consists of (i) editors (graphical and text) and (ii) libraries to build a model, (iii) an analyzer to check the model and to create the simulation structure, and (iv) a simulator to exercise the model.

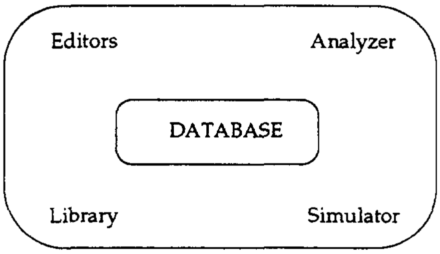

Figure 1: Foresight - Toolset

Usually, engineers will build a system model as a combination of data-flow diagrams (DFDs), state-transition diagrams (STDs), and code in the mini-spec language. One can include external functions written in C or C++. The tool has its roots in real-time structured analysis (RTSA) [2,3,4]. It uses extensions to the standard notations from the Extended Systems-modeling Language (ESML) [5] to create an executable RTSA, with the precision necessary for simulation. The product was acquired following an evaluation period of several weeks. During the evaluation period, 2 "typical" system examples have been modeled and simulated . This allowed to draw conclusions about (i) the tool's modeling and simulation capabilities, (ii) the possibilities to link it with existing computer-aided-engineering (CAE) tools, and (iii) its user-friendliness. The next chapter will describe system-modeling and simulation examples that have been carried out so far.

PRACTICAL EXPERIENCE

Until now, the tool has been used at CERN by engineers, physicists, and students. Some of the users had previous CAE or programming experience, others didn't. With one exception, all current users attended a 2-day training course, before starting those projects which we now describe in more detail.

Project 1: Model and Simulate the Readout System for a Large Particle Detector.

At CERN, a next generation particle accelerator, the Large Hadron Collider (LHC), is currently under study. The RD2 collaboration [6] is developing a concept for a large silicon detector, forming part of the Atlas experiment [7] which is planned for the LHC. The present design of the silicon detector foresees some two million detector channels. For the readout system, an application-specific integrated circuit (ASIC), serving 128 detector channels, is under development [8]. Its principle is shown in Figure 2. In each channel, one will find a pre-amplifier (Amp) and an analog memory-cell arrangement (Analog MMY), configured as a circular buffer. Further components of the ASIC are (i) the control circuitry to supervise memory read and write operations, (ii) the sparse-data-scan (SDS) block to skip empty channels during readout, (iii) an analog-to-digital converter (ADC), and (iv) an output-data buffer.

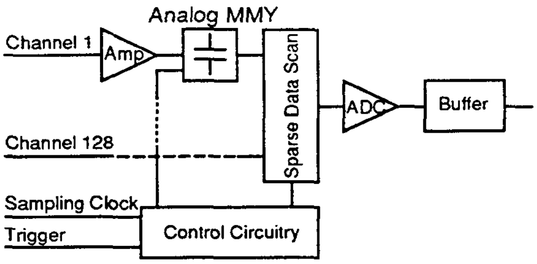

Figure 2: Silicon-detector readout scheme

The operating principle is the following. Data sampled at a given rate (every 25 ns) will be written into the analog-memory buffer, where they will occupy a certain number of locations (cells). If an interesting event occurs, logic circuitry (external to this system) will generate a trigger signal. It will arrive at the detector-readout ASIC after a fixed delay, with respect to the data that have been sampled for this event. The trigger signal is used to freeze all those analog-memory cells that hold the corresponding event data, to prevent them from being overwritten and to keep the data for further processing. Cells which have not been frozen will be overwritten during the normal operation of the circular analog-memory buffer. In general, event data will occupy a non-contiguous space in the buffer, because after repeated freeze and unfreeze operations buffer fragmentation will occur. Therefore, it is necessary to store buffer-location pointers for each recorded event in a first-in-first-out memory (FIFO) in the control circuitry. The freezing of cells also launches the readout process. The SDS block scans the 128 channels for valid data which after analog-to-digital conversion are stored in an output buffer for further processing. Current plans are to build modules which group 8 of these chips in a token-ring configuration for readout purposes.

The task was to model and to simulate such a module to monitor the processing time of each trigger at the various stages of the readout process and to study important system parameters, like optimal buffer size, pointer-memory size, acceptable readout delay values, losses as a function of these parameters, etc.. Models of the integrated circuit and the 8-chip module have been developed, validated, and simulated [9]. The models consist of a combination of DFDs (which include elements from the library), STDs and mini-specs. After validating the models, physics data were used to study the dynamic module behavior. In particular, events that meet the trigger requirements were generated. Then, the high-energy-physics simulation program GEANT [10] was used to calculate the detector response to these events and background events that were included. The resulting hit patterns, plus the effects of electronic noise, were stored in a file to provide the input data for the Foresight simulations. The following figures illustrate some of the results which have been obtained so far. They are based on a mean trigger rate of 100 kHz. Figure 3 shows the average occupancy of the output buffer and the FIFO as functions of the SDS time, i.e.; the time necessary to scan the 128 channels for valid data.

Figure 4 shows the readout-time distribution for events which occupy 3 analog-memory locations and for a SDS time of 100 ns. Figure 5 illustrates average and maximum readout times as a function of SDS time. After these encouraging results, work is under way (i) to study the system behavior under different conditions and (ii) to evaluate alternatives.

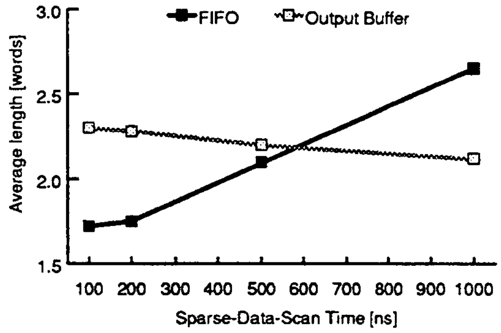

Figure 3: Average FIFO and buffer occupancy depending on SDS time

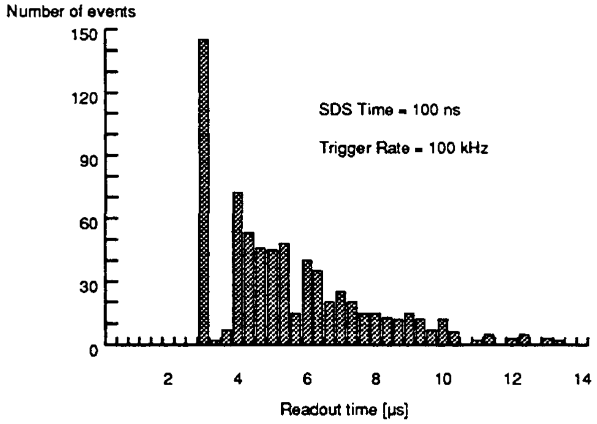

Figure 4: Readout-time distribution

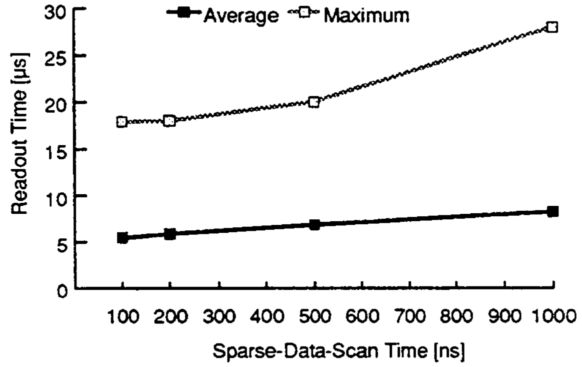

Figure 5: Readout time as function of SDS time

Project 2: Model and Analyze a Liquid-Helium Cooling System for Superconducting Magnets

The LHC accelerator will use superconducting magnets operated in pressurized helium at 1.9 °K. The cryogenic system must produce and distribute adequate refrigeration to the magnets. A special cryogenic distribution system is necessary to maintain all the magnets at the correct working temperature, regardless of their position along the circumference (27 km) of the accelerator tunnel. A full-scale (20 m) model of the cryogenic system, using heaters instead of the magnets, has been designed, built and tested [11]. The principle is shown in Figure 6. Magnets (heaters in the model) are surrounded by liquid helium and excess heat is absorbed by partly evaporating the liquid helium that circulates in a pipe. The current scheme is based on directly measuring and regulating

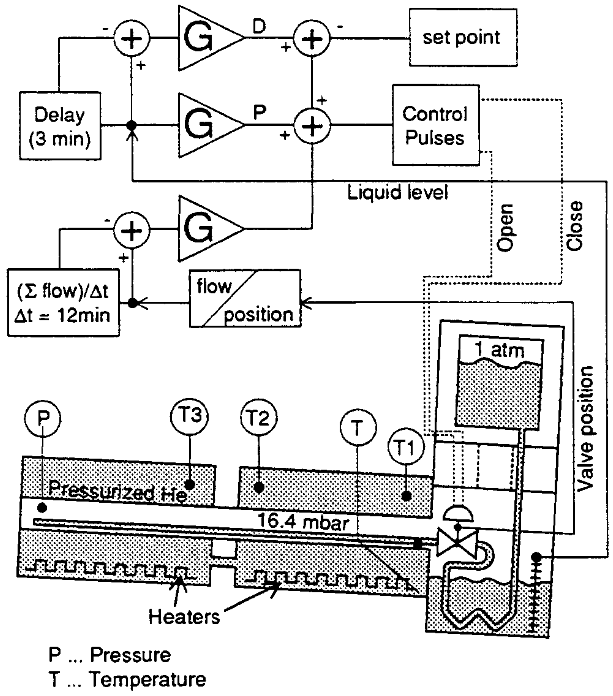

Figure 6: Cryogenic system for superconducting magnets

the temperature of the helium that surrounds the magnet. An alternative scheme is to regulate the temperature indirectly, measuring and controlling the level of the helium in the pot. The latter method has to deal with a significant delay between heat absorption and detection, but may eventually result in a lower operating temperature of the magnets. To study advantages and disadvantages of both schemes, behavioral and functional simulation models have been developed. The programmable-logic controllers of the systems and the valve were modeled as a combination of DFDs and STDs. Linked to them is the heat exchanging process, described as an external C program[12]. It has taken a cryogenics expert a few weeks to develop the models for both schemes. The models are currently in the validation phase. Simulation will then be used (i) to find optimal values for the different operating parameters of each scheme, and (ii) to decide which scheme to implement for a real-life test, to be performed on a 50 m long magnet string.

Project 3: Study the Feasibility of Building a Data-Acquisition System Based on an Existing Very-Large-Scale Integrated (VLSI) Circuit.

For LHC experiments a special VLSI chip, the MEC[13], has been developed. Its purpose is to detect and to store valid detector data for further processing. Unlike the configuration described in project 1, this one uses a data-driven scheme.

The principle is the following. Signals from certain particle detectors have a fairly short rise time, followed by a longer decay time. Such signals arrive at the inputs of the MEC chip where they are sampled at a given rate. Whenever the chip detects a sample value that exceeds a given (programmable) threshold, it assumes that the sample is from a valid detector signal. In response to this "trigger", and to acquire the "complete" signal, circuitry will transfer a pre-programmed number of samples that preceded and a pre-programmed number of samples that followed the above-threshold sample into a buffer. This set of values, assumed to represent a detector signal, is kept for further processing. Along with the samples, a unique and system-wide marker is recorded. The marker serves to retrieve all data that belong to a single event. During the readout process, the system will present a marker and all elements that hold corresponding data will respond to this request. For this project, the tasks are (i) to model the VLSI circuit and a module that uses several of these chips, and (ii) to study the system behavior as a function of different system configurations (like number of VLSI chips in subsystem loops) and parameters (like event rates, system readout time). The model of the chip has been built (as a combination of DFDs, STDs, external functions, etc.) and validated. Currently, the model of the module is in the validation phase.

CONCLUSIONS

The above described examples show that within a relatively short period of time and with a minimum amount of overhead (e.g.; training and support), it was possible to build and to simulate models of time-dependent systems.

Users find that (i) the graphical approach for building models, (ii) the libraries, and (iii) the highly interactive simulator which offers features like animation, single-step model execution, etc.; allow them to build and to validate system models in a relatively short period of time. It is felt that this approach (as opposed to writing and debugging code) is very helpful to optimize the use of creative engineering time. The price to pay for the interactivity is a reduction in simulation speed. But this issue is being addressed. A prototype that converts graphically built (and validated) system models into compilable code exists and has been demonstrated [14]. Early test results show improvements of the simulation speed by a factor of 40. If the code generator becomes available, one can build and validate system models using all the graphical and interactive features, in a first step. Then, in a second step, the compiled code will be used to collect data about the system's dynamic behavior.

A further issue is mapping the problem-domain model into the application domain, ideally at the push of a button. Work in this direction is under way, but complete solutions are not expected for the near future. It is felt that even partial solutions to that problem, like mapping STDs into VHDL (Very-high-speed-integrated-circuit Hardware Description Language) descriptions could be of great help in distributing reference models to the different parties involved in a large collaboration.

The overall conclusion of this paper is that the experience which has been gained until now in modeling and simulating real-time systems for a variety of applications is very positive.

ACKNOWLEDGMENTS

The authors wish to thank C. Eck, C. Fabjan, D. Jacobs, S. Cittolin and H. Verweij for actively supporting system-modeling and simulation. A. Thys and N. Gomez have been very helpful in getting us with this activity off the ground. Thank you also to B. Gaiser, B. Robinson, and M. Vertal from Nu Thena Systems for helping us in quickly reaching system-level-modeling proficiency and their openness and prompt reactions to our questions and suggestions.

REFERENCES

[1] Nu Thena Systems Inc. 1992. "Foresight." McLean, Virginia.

[2] DeMarco T. 1978. *Structured Analysis and System Specification*. Prentice Hall, Englewood Cliffs, N.J.

[3] Mellor, S. and Ward, P. 1985. *Structured Development for Real-Time Systems*. Vols. 1-3, Yourdon Press, New York, N.Y.

[4] Hatley, D. J. and Pirbhai, I. 1987. *Strategies for Real-Time System Specification*. Dorset House, New York, N.Y.

[5] Bruyn, W. et al. 1988. "ESML: An Extended Systems Modeling Language Based on the Data Flow Diagram." *ACM Sigsoft, Software Engineering Notes*, no. 1 (Jan.): 58-67.

[6] Clark, A. and Goessling, C. 1990. "A Proposal to Study a Tracking/Preshower Detector for the LHC." CERN DRDC/90-27. CERN, Geneva, Switzerland.

[7] Atlas Collaboration. 1992. "Letter of Intent." CERN LHCC/92-4. CERN, Geneva, Switzerland.

[8] Anghinolfi, F., et al. 1990. "Monolithic CMOS Front-end Electronics with Analog Pipelining." *In Conference Record of the 1990 IEEE Nuclear Science Symposium* (Arlington, VA, Oct. 22–27). IEEE, Piscataway, N.J., 543–552

[9] Benslama, K. 1993. "Simulation Foresight du système de lecture du détecteur SIT ATLAS/LHC." Thesis. Physics Department, University of Geneva, Geneva, Switzerland.

[10] Clark, A., Dodgson, M., Verweij, H.1992. "Estimates of SIT Occupancy Using 2-jet Triggers." CERN SIT-NO-043. CERN, Geneva, Switzerland.

[11] Bezaguet, A. et al. 1993. "The Superfluid Helium Model Cryoloop for the CERN Large Hadron Collider (LHC)." CERN AT/93-21. CERN, Geneva, Switzerland.

[12] van Weelderen, R. 1993. "Private Communication"

[13] Christiansen, J. 1993. "The MEC Demonstrator CHIP." CERN/ECP-MIC 21/4-1993. CERN, Geneva, Switzerland.

[14] Gaiser, B. 1993. "Private Communication"