Simulation and Modeling Tools in Data Acquisition System Design for Future High Energy Physics Experiments

M. J. Haney
University of Illinois, Urbana, Illinois, USA

Abstract

In light of the ever increasing complexity of high energy physics experiments, there is little choice for future experiment designers but to employ methodologies of sophistication equal or greater to those employed in developing state-of-the-art commercial electronics. This paper examines simulation and behavioral modeling, and how this approach has, and can, be applied to high energy physics experiments.

1 Introduction

Future high energy physics experiments face both an obvious, and an obscure challenge. With each new experiment, or upgrade, or publication, more and more of the "easy things" are being done; this obvious statement poses the challenge of ever increasing complexity in experiment design, to address ever more interesting problems. Fortunately, the electronics industry is growing rapidly in sophistication, and the elements available (ASICs, processors, data communications) are improving every day.

However, it is this same growth in sophistication in the electronics field which poses the obscure challenge. As increasingly complex building blocks are assembled, increasingly subtle system interactions and failures are becoming manifest. It is of little consolation to look at a triumph of human endeavor, such as the Hubble Space Telescope, and realize that it the optical system had only a "small" error.

Unless physics experiment designers are willing to embrace the same (or better) methodologies that are used in the development of the underlying electronics, then future experiments can anticipate only more subtle, and more debilitating results.

This paper briefly reviews simulation as it is used today in the electronics industry, and provides examples from both electronics, and high energy physics, as to what is done, and can be done in the future.

Due to the space limitations of this paper, many references and pointers have been omitted. Please refer to Section 5, Supplemental Notes, on details for access to this information.

2 Advanced Methods

The design of sophisticated electronics requires the use of sophisticated design automation tools. The following sections examine high level modeling, its goals and limitations, and an overview of available languages and tools.

2.1 High Level Modeling

Creating a model, and simulating it, represents only a small part of the design process. A brief overview of modeling is provided, and discrete event simulation in specific, followed by a discussion of synthesis, and verification and validation.

2.1.1 Modeling, Analysis, and Synthesis

In the following, the phrases "engineering modeling" and "engineering simulation" refer predominately to electrical engineering; "physics" modeling and simulation, as routinely used in data analysis, will be contrasted to this.

Modeling is the creation or fitting of a representation to a system or subsystem of interest. Models may be equations, executable code, scale mock-ups, etc. The extent to which the model approaches reality is often referred to (in engineering) as the level of abstraction. Highest level models, which reflect only the outermost visible manifestation of the system being modeled are called behavioral models. Typically, behavioral models are written in a high-level language, such as VHDL or Verilog (hardware description languages), or in C (or variants). Structural models reflect aspects of how the system is composed; an electrical schematic diagram, showing gates and registers is a structural representation. Physical models, the lowest level of abstraction, focus on the finest detail of the system construction. The masks used to produce an integrated circuit, identifying differing regions of silicon, are physical representations. These notions, behavioral, structural, and physical, tend to have a relative as opposed to absolute interpretation.

Analysis is the derivation of results from the model, typically through mathematic processes. Analyses may be closed form (analytic), or they may involve numerical methods, such as numerical integration and matrix manipulation. The circuit "simulation" program SPICE, in its DC analysis, is precisely that: analysis. Perhaps the most popular tool for performing engineering analysis is MatLab, although Mathematica is also used.

Simulation is the derivation of results by allowing the models to interact, either in time or space or whatever domain is appropriate to the problem. Simulations are "dynamic" as opposed to a static analysis.

Since the notion of "physics simulation" is familiar to anyone who has performed high energy event analysis, it is important to differentiate that which is done in physics from that which is done in engineering. Physics tools, such as GEANT, use Monte Carlo methods to create an input "stimulus" to their model (the detector), then use Monte Carlo methods to determine the outcome of local interactions (e.g. particle decay). Each input "event" is treated as a distinct, orthogonal aspect of the simulation.

Engineering simulation also may use Monte Carlo methods to generate an input stimulus, but it is more common for a specific (trace-driven) test case to be used as the stimulus. The outcome of local interactions is then determined from this stimulus in light of the system state. "Events" (in the physics sense) are not orthogonal, but in fact their interaction is the proximal cause of the system behavior.

The orthogonality issue is important. Since the outcome of each physical interaction is stochastic, it becomes exponentially difficult to keep track of all of the possibilities. Thus event-event interaction is largely ignored, which is justified as long as the occupancy of each part of the detector remains low. However, in engineering simulations, each "event" has a fairly well defined (deterministic) effect on the system state; this is the role of the model, to transform the system state. It is the evolution of this system state, in response to the input stimuli ("events" in the physics sense) which is the basis of the simulation itself.

The word "event" has a very different meaning in engineering simulation (as in "discrete event simulation"). Hence forth, "event" shall refer to any microscopic change in system state.

2.1.2 Discrete Event Simulation

There are several forms of simulation routinely used in engineering, but of special relevance to this discussion is discrete event simulation. In discrete event simulation, the simulation kernel maintains a time-sorted "event-queue", which contains pointers to models to be executed, and the time at which they are to execute. Since the event-queue is time-sorted, the kernel can simply draw the top-most "event" from the queue, set the system time equal to the time indicated for that event, and execute that model.

The model being executed then examines the system state, and based on its own functional role, transforms the system state and schedules additional events (i.e. other models) to be executed at future times. These events are "posted" into the event-queue, in time-sorted order.

The net effect of discrete event simulation is that very little overhead is incurred except with regard to those models that actually transform the system state. Thus, most of the computation of the simulation is spent in those models that do most of the work. There is, however, a non-negligible overhead associated with maintaining the event-queue in time-sorted order; this gives rise to discrete-task, and cycle based simulation methods, which are beyond the scope of this paper.

2.1.3 Synthesis

Also important to understand is the role of synthesis, which is the "automated" process of elaboration of a high level behavioral representation to (or toward) a low level structural or physical representation. Fully automated synthesis remains an elusive goal, but commercial solutions exist today for addressing the translation of RTL (register transfer level - a somewhat "lower" high level behavioral representation) to gates (structural).

2.1.4 Verification and Validation

Two final concepts are important: verification and validation. Verification, when the term is formally used, refers to the formal process of proving that the result of a synthesis is equivalent to the input specification. Mathematically, verification is an immensely difficult task. Validation is a looser term, in that it is the demonstration that a system performs as expected. Exhaustive validation could be used as verification, but the computational cost of this in any practical system is beyond prohibitive. Regrettably, the word "verification" is often abused to mean validation.

2.2 Goals and Limitations

There are several achievable goals to high level modeling and simulation. First and foremost is the validation of the system design; simulation makes it possible to determine if, and how, a system should function, prior to committing capital resources.

Simulation also supports performance estimation: throughput, deadtime, load imbalance, resource utilization. Each of these can be examined, and used as a metric for alternatives comparison.

The comparison of alternative solutions is a third significant goal of simulation. Comparisons of ATM vs barrel-shifting for event building, or push vs pull for coordinating data collection, can be readily performed through simulation. The basis of these studies can

then be used to guide system design, and assist in determining the best solution within the economic resources of the project.

However, in addition to the above lofty goals, there are quite realistic limitations to simulation. The model is no better than that which is put into it, or left out of it. Further, modeling and simulation are not inexpensive, either in CPU time, memory, or disk space. The UltraSPARC RISC developed by Sun Microsystems consumed 730 MBytes runtime memory for a full system model, and running SPICE as a test application takes a month, running at 6200 simulated instructions per second on 60 MHz Sparc20.

Simulation is also prone to propagation of error, both in the simulation itself, and in the users blind faith of its results.

2.2.1 Cases from Industry

Simulation is used very effectively in the industrial sector, making possible the complex electronic subsystems that are used as building blocks in high energy physics experiments. The following examples drawn from industry outline the benefits of simulation:

- Simulation as specification:

The SCI specification is largely documented in C. As a result, everyone has the same information and a common interpretation. This minimizes interoperability problems.

- To get better leverage on the design process:

Fiat Central Research reduced design cycle time by 30% by mixing schematics with VHDL; they did not want to "lose" time learning VHDL up front.

SGI started with 78 PALs, and added 3 times more features to the specification. This resulted in 18 FPGAs (13 new) by 6 engineers, which were completed in 8 months using Verilog + Synopsys (synthesis).

Motorola's Coldfire (embedded computing processor) was almost entirely developed in Verilog, then synthesized: 90% of the transistors were never "touched by human hands."

- To focus on the "problem" not the process:

TransSwitch developed a SONET chip set using VHDL at the RTL level. 80% of their time was focused on functional aspects of the problem, while only 20% was spent on implementation details. As a result, major modifications had small effects on schedules. However, it became painfully clear that large scale VHDL projects are in fact large software projects; code-review and configuration management are imperatives.

Sun Microsystems developed an UltraSPARC-I performance simulator using approximately 45,000 lines of C. Running a SPICE benchmark (15 billion instructions) takes a month, but sampling allowed a performance study to be done in 2.5 hours. In spite of a new processor, new system architecture, new silicon process, and new packaging, they were able to have multi-user UNIX running within one week of first silicon.

2.3 Languages and Tools

An every changing array of design automation tools is available; a list of commercial hardware description language vendors is available in [1].

It is relatively easy to create an event-driven simulator in an object oriented language. Defining a small number of base-classes to define an "event" with a member function of

"what_to_do", and some kernel code to support the event-queue, as well as "execute" and "post" functions provides most of the required elements. Of course, such a simple implementation places a great deal of burden on the programmer to use the pieces correctly. The real value added in creating an event driven simulator is the "ease of use" features added.

As a consequence, there are countless C/C++ based simulators available, many for free, with varying degrees of applicability. Several have been used by the high energy physics community, as well as the electronics industry. However, the electronics industry is dominated by two hardware description languages: VHDL and Verilog. Recent reviews of tools specific to these can be found in [2] and [3].

2.3.1 VHDL and Verilog

VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language) was born out of a DOD mandate for documentation. IEEE Std. 1076, VHDL is a flexible language, with object-oriented aspects (such as the entity/architecture construct), supporting delayed binding (USE) and parametric instantiation (GENERATE). For various reasons, VHDL library support has not been strong, but VITAL (VHDL Initiative Toward ASIC Libraries) is addressing this deficiency, promoting "sign-off" libraries, tools, and standards.

Verilog was developed as a proprietary language by Gateway Design Systems (now owned by Cadence), yet it became the de facto standard of Synopsys, the leading synthesis tool vendor. As a result, Verilog has a large number of model libraries for ASICs. It will soon become an IEEE standard (P1364).

Perhaps the most common complaints are that VHDL has too much syntactic overhead (i.e. it takes a great deal of "boiler plate" to get anything done), while Verilog is semantically too poor (no abstract data types, no parametric instantiation, etc.) to address extremely abstract problems.

Yet Verilog remains fast to learn and easy to use. And despite persistent rumors that VHDL is sweeping the market and that Verilog is dead, there is no reason to believe that Verilogs life time will be any shorter than that of COBOL.

2.3.2 Modsim and Others

There are also many simulation-specific languages and tools to consider. The two used most often in the high energy physics are Modsim (from CACI) and Foresight (from NuThena). Modsim is a very clean object-oriented programming environment with simulation constructs built-in. It is fast to learn, easy to use, and very productive as a simulation tool. Modsim is also an excellent starting point for learning object-oriented programming fundamentals.

Foresight is a specification-focused tool. Its underlying language, Mini-Spec, is well balanced at providing both system specification, and simulation support. A graphical front-end, combined with both VHDL and C output makes Foresight a formidable tool.

3 Cases from Physics

The use of high level tools in the high energy physics community is limited but encouraging. The following cross section should provide some insight as to what is being done. There are several additional references available via the Supplemental Notes (section 5).

3.1 VHDL

VHDL has seen only a relatively small amount of use in high energy physics community, perhaps largely due to the relatively slow appearance of cost-effective tools.

A data collection chip (DCC) was modeled [4] both as a discrete device, and as a tree of DCC devices performing event collection (distributed event building), and FASTBUS simulation tools [5] were developed to provided a virtual backplane, virtual master, and virtual slave as a basis for testing new board designs in a virtual system environment.

3.2 Verilog

Verilog has been more popular than VHDL, largely due to the simplicity of the language. Unfortunately, this same simplicity has limited Verilogs usefulness in high level modeling. The language has been used to evaluate upgrade alternatives for CDF [6], and identified design improvements in the process of focusing on the model.

Verilog has been combined with DataViews (visualization and control) and Nexpert (database interface and rules driven expert system) to provide a comprehensive approach to event builder design and diagnosis [7].

And for SCI, a hardware simulation in Verilog was compared to an architectural simulation developed in Simula (to be redone in Modsim), to study latency, memory requirements, and scalability [8].

3.3 Modsim

Modsim is well suited to large scale simulations, and has been used in a number of high energy physics studies. However, a common complaint [9] is that the limits of the machine and software version have been too easy to reach.

Modsim was used in several related SDC projects; in each case the code tended not to be reused as-is, but instead was "adjusted" to suit the needs of the moment. It is not clear whether this is evidence against object-oriented code reuse, or simply a manifestation of the not-invented-here syndrome [10, 11, 12, 13].

Modsim has also seen much use at CERN. RD13 used Modsim to model a scalable DAQ system for LHC, resulting in the DAQ Simulation Library (DSL) [14]. Code is available on-line providing a wide selection of building-block objects. DAQ modeling for Atlas at CERN "distilled" DSL into SIMDAQ. Code and documentation are available on-line. Publications to appear soon (if not already). Also at CERN, RD11-EAST (embedded architectures for second-level triggering) is also using SIMDAQ.

CLEO III is using Modsim to examine its data acquisition design [15].

3.4 C++, etc.

Substantial SCI and ATM simulation has, and is, being done using C++ or variations of it [16, 17, 18]. As systems become more complex, it is increasingly important to find commercial solutions to data transmission and collection problems. But the driving forces behind ATM, for example, are not high energy physics, but telecommunications. And the problems that they are trying to address are very different from those of the physics community. It is imperative

to determine whether commercial solutions are viable, and cost effective; simulation is one relatively inexpensive way to make this evaluation.

3.5 Other

Foresight is focused as a "specification tool." It's underlying language, Mini-Spec (derived from Ada), is especially strong in abstract data types and specification. It has been under scrutiny at CERN for some time [19]. It is currently being used to model an accelerator control system at CERN. Using the FS-C code generator, a "real-time prototype" can be executed in conjunction with real control system code.

4 Summary

We are at, or perhaps beyond, the limit where one can simply buy the pieces and casually put together interesting solutions to high energy physics experiments. The consequences of assembling an expensive system that does not live up to its promises are serious indeed, not only for the group that failed, but for the high energy community at large.

To cope with exponential complexity demands approaches which transcend linearly-scaling methods. Architectural alternatives must be evaluated without making excessive financial commitments. Hence behavioral simulation. Prototyping of small parts of a problem remains valid, but system prototyping is not tractable, as system behavior typically does not scale in a simple manner. Hence virtual prototyping and mixed-level simulation.

And the time/money/thought invested in simulation must not be wasted. If the system is to be built, then a clear path from simulation to implementation must be held in sight at all times. Hence synthesis.

Finally, the only way to be confident that the system will work in the field as it does on the screen, is through an aggressive approach to testing and fault-location. Hence validation and verification.

Unfortunately, the required system-level design tools do not exist today [20]. However, the DOD sponsored RASSP (Rapid Prototyping of ASIC Signal Processors) program is a multi-company and school research project which is firmly system-oriented, and holds quite a lot of promise. Also, the marketplace is asking for increasing sophistication in cars, TV, telephone, home heating/cooling, PDA, etc. As complexity in everyday life soars, look to (market driven) embedded processor research as a driving force behind system specification and design.

5 Supplemental Notes

This paper reflects a pair of lectures provide in August/September, 1995, at the CERN School of Computing, Arles, France. The relative freedom of the lecture environment made it possible to present a somewhat wider spectrum of information related to this topic. The lecture notes, as well as a collection of reference pointers (papers, company names and telephone addresses, URLs, etc.) are available variously via the World Wide Web from

http://eng3.hep.uiuc.edu

and from the author via email

m-haney@uiuc.edu

or conventional correspondence

Michael J. Haney

High Energy Physics

University of Illinois

1110 W. Green St.

Urbana, IL 61801 USA

References

[1] HDL tools, *ASIC & EDA Vendor Guide 1995 Supplement*, Dec/Jan., pp 12-37 (1995).
[2] L.Saunders and Y.Trivedi, 'Product Evaluation: VHDL Simulators,' *ASIC & EDA*, July, pp 12-37 (1994).
[3] L.Saunders and Y.Trivedi, 'Product Evaluation: PC-based Verilog Simulators,' *ASIC & EDA*, April, pp 12-36 (1994).
[4] E.Hughes, et al., 'Modeling and simulation of the SDC data collection chip,' *IEEE Trans. Nuc. Sci.*, vol. 39, no. 2, pp 130-137 (1992).
[5] T.D.Dean & M.J.Haney., 'FASTBUS simulation tools,' *IEEE Trans. Nuc. Sci.*, vol. 39, no. 4, pp 910-914 (1992).
[6] Schurecht, et al., 'A Verilog simulation of the CDF DAQ system,' *Conference Record of the 1991 IEEE Nuclear Science Symposium and Medical Imaging Conference*, Santa Fe, New Mexico, pp 893-897 (1991).
[7] Booth, et al., 'Software development for as switch-based data acquisition system,' *Conference Record of the 1991 IEEE Nuclear Science Symposium and Medical Imaging Conference*, Santa Fe, New Mexico, pp 913-917 (1991).
[8] A.Bogaerts, et al., 'Applications of the Scalable Coherent Interface to data acquisition at LHC,' CERN/DRDC/91-45, DRDC/P33 (1991).
[9] J.Streets, et al., 'Experience with MODSIM II,' FERMILAB-Conf-92/43, presented at the Second International Workshop on Software Engineering, Artificial Intelligence and Expert Software for High Energy and Nuclear Physics, L'Agelonde France-Telecom La Londe-les-Maures (1992).
[10] E.C.Milner, et al., 'Data acquisition studies for the Superconducting Super Collider,' *IEEE Trans. Nuc. Sci.*, vol. 39, no. 2, pp 138-142 (1992).
[11] A.W.Booth, et al., 'DAQSIM: A data acquisition system simulation tool,' *IEEE Trans. Nuc. Sci.*, vol. 40, no. 4, pp 788-793 (1993).
[12] C.C.Wang, et al., 'A simulation for the SDC on-line processing farm,' *1993 IEEE Conference Record of the Nuclear Science Symposium and Medical Imaging Conference*, San Francisco, California, pp 65-67 (1993).
[13] M.J.Haney, et al., 'The SDC DAQSim Simulation Effort,' *1994 IEEE Conference Record of the Nuclear Science Symposium and Medical Imaging Conference*, Norfolk, Virginia, pp 871-874 (1994).
[14] S.Buono, et al. (presented by R.Spiwoks), 'DAQ simulation library (DSL),' *International Data Acquisition Conference*, FermiLab, Batavia, IL, presentation #52 (1994).
[15] A.Wolf, et al., 'MODSIM based simulation of the CLEO III data acquisition system,' presented at the RT'95 Conference, East Lansing, MI (1995).
[16] E.H.Kristiansen, et al., 'Simulations with SCI as a data carrier in data acquisition systems,' *IEEE Trans. Nuc. Sci.*, vol. 41, no. 4, February 1994, pp 125-130.
[17] M.Letheren, et al., 'An asynchronous data-driven event-building scheme based on ATM switching fabrics,' *IEEE Trans. Nuc. Sci.*, vol. 41, no. 1, pp 257-266 (1994).
[18] G.Horn, et al., 'Performance simulations of networks with point-to-point links,' *International Data Acquisition Conference*, FermiLab, Batavia, IL, presentation #54 (1994).
[19] L.Pregernig, 'Executable system specifications,' presented at the Scweizer Automatik Pool Forum 21, Zurich, Switzerland (1993).

[20] NSF Workshop on CAD Needs for System Design, Boulder, CO, April 1995.