

**Fermi National Accelerator Laboratory**

**FERMILAB-Conf-92/43**

## **Experience with MODSIM II**

J. Streets, D. Berg, G. Oleynik, R. Pordes and D. Slimmer

*Fermi National Accelerator Laboratory  
P.O. Box 500, Batavia, Illinois 60510*

February 1992

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, January 13-18, 1992.

## **Disclaimer**

*This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.*

## Experience with MODSIM II\*

J.Streets, D.Berg, G.Oleynik, R.Pordes, D.Slimmer.

*Fermi National Accelerator Laboratory<sup>†</sup>,  
Batavia, IL 60510, USA.*

### ABSTRACT

We present results of computer simulations for Data Acquisition systems for large fixed target experiments in an object oriented simulation language, MODSIM. This paper summarises our experiences and presents preliminary results from the simulations already completed. We also indicate the resources required for this project.

### 1 The Problem.

On Line Support at FNAL is designing a new Data Acquisition (DA) system for the next fixed target run, scheduled in 1994. This system must serve large experiments which read data from the front end electronics at a rate of 100Mbytes/second, and require a software event filter to reduce the rate of data written to tape to 10Mbytes/second. The same system should also be scalable, down to 1Mbytes/second for test beam experiments. The data will be read out from multiple streams of CAMAC and FASTBUS. We would like to use cost effective UNIX<sup>‡</sup> computers as backend Filter Processors (FP) for online event selection.

Preliminary studies have found<sup>1</sup> several possible architectures using:

- VDAS buffers
- VSB/VME dual ported (DP) memories
- In house DP Stream Memories with IO adapter
- DP VME/ECL memories
- Vertical Busses
- Ultranet, HPPI and In House Switches

### 2 Why Simulate ?

Each architecture listed above has potential problems with performance, cost, availability and complexity. It is impossible to build some of these systems today as not all hardware has been designed. Limitations of people and money prohibit building full scale test systems for several of these architectures.

Computer simulations can help decide whether these architectures will work at the data rates required. Simulations can also help define the hardware requirements of modules which are still to be designed. The software algorithms which must run in the processors to build events can also be prototyped and tested in the computer simulation.

---

\*MODSIM II is a registered Trade Mark of CACI Products Company, 3344 North Torrey Pines Court, La Jolla, CA 92037, USA.

<sup>†</sup>Sponsored by DOE contract number DE-AC02-76CH03000.

<sup>‡</sup>UNIX is a registered trademark of UNIX System Laboratories.

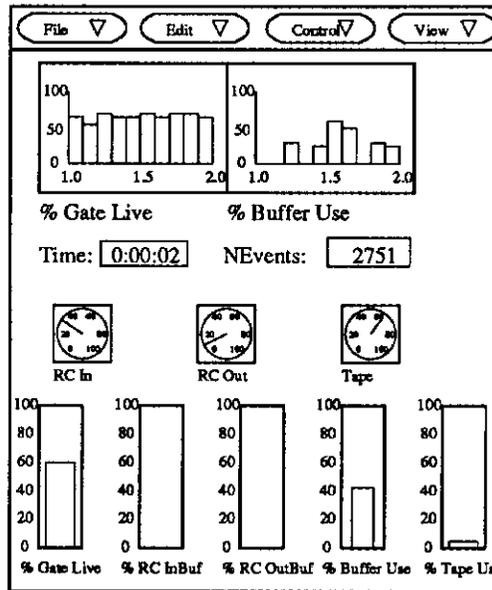


Figure 1: Example of menu and graphics widgets

A simulation of the different architectures will identify the cheapest working solution and also help tune parameters to find potential bottlenecks and maximise efficiencies of the different sub-systems. Once an architecture has been found the simulation will be used to find the best configuration parameters, such as the numbers of CAMAC and FASTBUS streams, if memory should be added to buffers or filter processors, or how variations in the word counts from detectors will affect the system. These changes could be found from the hardware, but it will be far easier to get an estimate from running the simulation.

### 3 Why MODSIM ?

The language we choose for the simulation needs to satisfy some basic requirements. The project should generate reusable software in order to minimise software development for each different architecture. The various parameters of the DA systems should be configurable at run time. The running programs should be easy to monitor so that we can find system bottlenecks. This type of monitoring is best done with graphical displays. The DA systems all contain many processes running concurrently, and the simulation language must support control of these processes and interprocess communication.

MODSIM II was chosen for its support of object oriented programming and for large process-based simulations<sup>2</sup>. This language has also been used by other laboratories<sup>3</sup>. Other simulation languages, such as Verilog and SESWorkbench were found to be aimed at finer scale simulations, or not sufficiently developed for our use. Writing a simulation in C was considered an inefficient use of resources.

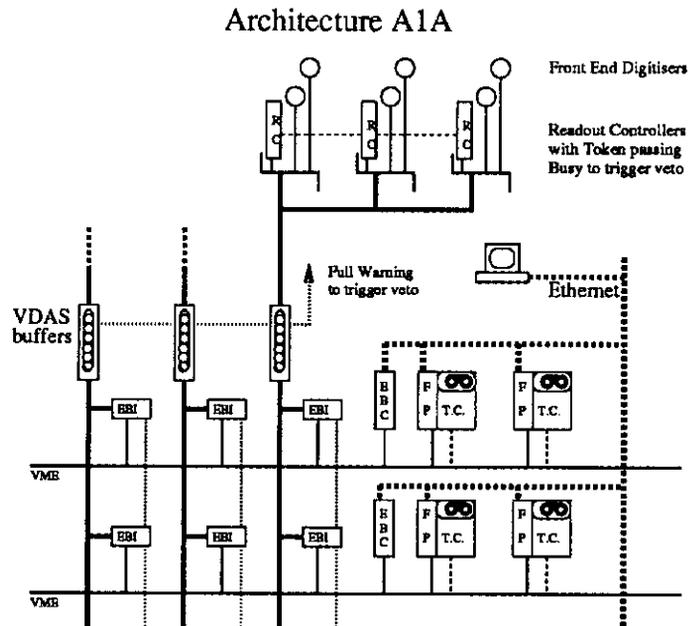


Figure 2: Diagram of the A1A Data Acquisition architecture

#### 4 Simple-DA, a first attempt.

Our first project was a simple DA system which contained one readout controller with input and output buffers, a large buffer memory and a tape drive. This project took approximately 6 person-weeks. This time include learning object oriented techniques, and the MODSIM language.

A graphical monitoring display was then added. This took about 2 person-weeks, again this time includes learning the MODSIM language. The results can be seen in figure 1. Another week of trying the menu widget produced an interface to enable the modification of initialization parameters.

The simulation was not designed to be expandable to a larger system, and so was not developed further. However it was found useful as a tool to learn about the functionality of the MODSIM class library, and some of its limitations.

#### 5 VME bus simulation.

The next project was to write a set of re-usable set of classes to simulate a VME bus. The VME object is present in all proposed architectures and has a potential for limiting event throughput. We used ROSE\* to both design and document the project.

A class of objects were written to simulate a VME bus with either a coarse or fine granularity. The coarse grained bus has the following properties.

- Only one master can use Bus.
- Masters are queued in order of request.

The fine grained bus has these additional features.

---

\*Rational Rose is a trademark of Rational, 3320 Scott Boulevard, Santa Clara, CA 95054-3197, USA.

- Obeys priorities (0-3) and slot position.
- Round Robin or priority scheduling.
- Timeouts are implemented.
- Single word and block transfers are recognised.
- Data can be written in 256 byte blocks.

These objects took approximately four weeks for one person to implement. Some of this time was spent learning the ROSE CASE tool.

## 6 A1A, a second attempt.

Our next attempt was a DA system which is similar to the fastest DA running<sup>4</sup> at FNAL, but modified for our requirements. This took 3 weeks for one person to design and implement. The system required over 200 initialization parameters, and we found that the MODSIM menu interface was unsuitable, so we wrote our own class called ConfigFile, which has a similar format to Xdefaults\* files. This took about a week.

The A1A simulation uses no inheritance and the use of MODSIM interrupts was restricted to the Exabyte<sup>†</sup> simulation routines (a feature of the architecture). Only the random number generator was used from the MODSIM class library.

Consistency checks performed to verify correct working of the model include

- The number of words read from the front end modules equals the number of words written to tape.
- The event numbers from subevents are compared during event building.
- The front end modules receive the same number of triggers.
- The measured deadtime compares well with the prediction from the front end module read out time.

## 7 Using the A1A simulation.

We used the model to simulate the readout from a proposed Kaon experiment<sup>5</sup>. By varying the number of VME crates in the system we could find the most cost effective solution to the problem of how many VME crates should be used for the fixed number of Filter Processors in the system. The number of events processed during one full spill, as a function of the number of VME crates used to distribute 16 processors, was as follows.

<i>Number of VME crates</i>	<i>Events processed</i>
16	300000
8	288000
4	274000
2	252000
1	226000

The jobs ran for 16 hours on a Sun 4/75 (SPARCstation 2), and required 170 Mbytes of virtual memory. Each job simulated 120 seconds of beam time. The decreasing number of events can be explained by the protocol used to queue the processors by the event builder controller. The simulation suggests that we can halve the number of VME crates with only a 4% loss in event throughput.

\*X Window System is a trademark of the Massachusetts Institute of Technology

<sup>†</sup>Exabyte is a trademark of the Exabyte Corporation

## 8 Next Steps.

We would like to use the VME bus object to verify results from the A1A model. We are adding graphics to A1A to aid identification of bottlenecks, and presentation to customers. We want to find a set of parameters to satisfy the requirements of the FNAL experiments which will run in 1994. We plan to create models of other architectures mentioned above (testing the reusability of the code). It will also be possible to force errors in the system, to see how the models recover from hardware and software failures, such as power supply failure, system floating point interruptions and bit errors in the front end modules. We will later simulate the run control and system initialization, and upgrade to a later version of MODSIM.

## 9 Summary and Impressions.

We have used MODSIM to simulate a data acquisition system, and found useful results. We have used the MODSIM librarian to support multi-user environment. Response from the CACI telephone support is quite good. Courses are available and we have found them instructive but not necessary. The SunOS\* graphical displays are sufficiently compatible that, although MODSIM is supported only under the SunOS UNIX operating system, we can run the programs from any X terminal.

Our simulations are near the reasonable limits of CPU and memory, and we are near the limits of version 1.6 of the MODSIM II compiler, as we have found several problems. We find that MODSIM is easier to use without inheritance, and is more suited to large scale simulations. We have found difficulties in producing object oriented code which is reusable without minor modifications of the original definitions.

Finally, we wish to stress that the results of a computer simulation are only as good as the assumptions and predictions upon which it is based.

## References

1. D.Berg, G.Oleynik, R.Pordes, J.Streets and D.Slimmer. *DART - A New High Speed Data Acquisition System, System Concept*. FNAL Internal Note DS217, October 1991.
2. CACI Products Company. *MODSIM II The Language for object oriented programming - Tutorial*. May 1991.
3. E.C.Milner, A.W.Booth, M.Botlo and J.Dorenbosch. *Data Acquisition Studies for the Superconducting Super Collider*. Proceedings of the IEEE Seventh Conference REAL TIME June 1991 p. 30.
4. See for instance  
C.Gay and S.Bracker. Transactions on Nuclear Science, NS-34, 4 1987, and  
A.Baumbaugh. *The Video Data Acquisition System (VDAS)* FNAL internal report, July 1990.
5. S.Childress, S.Cihangir, R.Coleman, M.Crisler, R.Ford, Y.B.Hsiung, D.Jensen, E.Swallow, Y.Wah. *Design Report and Impact for the KTeV Program* FNAL Report, June 1991.

---

\*SunOS is a registered trademark of Sun Microsystems