# Interactive Simulation of LEB Commissioning Procedure on a Hypercube Parallel Computer

G. Bourianoff, M. Botlo, B. Cole, S. Hunt, N. Malitsky, A. Romero

SSC Laboratory*
2550 Beckleymeade Avenue
Dallas, Texas 75237

*Abstract*

It is desirable that an interactive simulation of accelerator operation be developed in order to write and test commissioning, correction, supervisory control, closed loop control, optimization and automation code prior to machine construction. The simulator should produce realistic diagnostic information, analyze and display the information at a workstation, accept operator input, and react appropriately. Such a system has been developed by the Accelerator System Control Simulator Collaboration to model the Low Energy Booster (LEB). The system is implemented on a 64 node INTEL ISPC/860 parallel computer which operates at approximately 600 Mflops. The simulator can track 512 particles on 32 nodes at 1 turn per second using an element by element symplectic integrator based on the TEAPOT algorithm. An operator interface has been implemented on a SUN Sparc 2 workstation operating as a client to a VME based 68040 processor board running VxWorks real time operating system. Data display and operator input utilize the operator interface routines in the EPICS control system. Data access between the SPARC Card and the HYPERCUBE is accomplished currently with an interprocess socket connection. Simulation of the interactive closed orbit smoothing process will be shown.

## INTRODUCTION

Because of its raw processing power, the parallel processor is being used as the engine in an interactive simulator of the Low Energy Booster (LEB). This simulator is being developed for two purposes. The first is to develop a platform on which high level correction code can be developed and from which an operator can control the simulator with the same look and feel he or she will experience in the control room. The second purpose is to test the data handling characteristics of the EPICS control system.

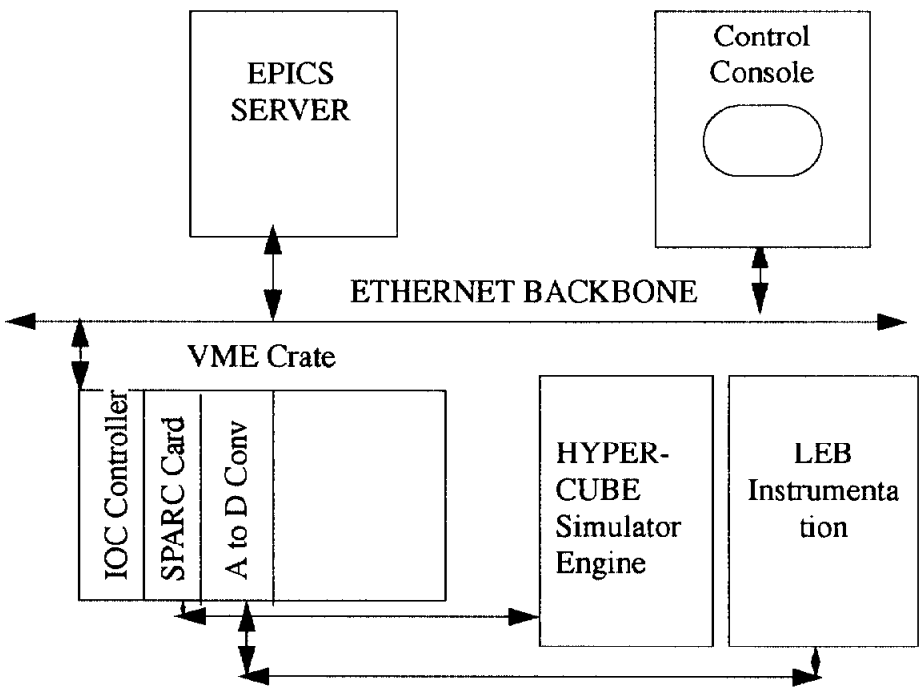


Figure1 Schematic Diagram of LEB Simulator

*Operated by the Universities Research Association Inc., for the U.S. Department of Energy under Contract DE-AC02-89ER40486.



0-7803-1203-1/93$03.00 © 1993 IEEE

## Simulator Architecture

The architecture of the simulator is shown in figure 1. The computational model running on the Hypercube is shown in the lower right hand corner adjacent to a box showing LEB instrumentation. The simulation results produced on the Hypercube are transferred over an Ethernet connection to a rack mounted SPARC card which formats the data into the same form produced by the A to D converter which actually receives the data from the LEB instrumentation. A software driver for the SPARC card then reads or writes the data into the EPICS control system where it is handled in the same manner that data produced by the actual accelerator instrumentation.

The LEB and its instrumentation do not exist at the present time. The propose of showing it in figure 1 is to emphasize the simulator engine is interchangeable with the actual instrumentation form the viewpoint of the high level control code.

The communication between the hardware components shown in Fig. 1 use a collection of hardware links and software protocols that closely resemble those that will be found in the SSC control system. The heart of the communication between the application code running in the Unix environment and the instrumentation hardware (modeled here by the Hypercube) is the EPICS database that physically resides in the Input Output Controller (IOC). The IOC itself is a Motorola 68040 based processor in a VME crate. The EPICS database has a set of pointers that connect variables that can be accessed from the UNIX world (channel access variables) to actual hardware addresses that connect to the low level instrumentation. The database also contains the information on data format, refresh frequency, error codes, etc. that are required to interpret the raw instrumentation signals.

In the simulator, data is produced asynchronously by the simulator engine. This is passed over an Ethernet connection directly to VME memory space. Event flags are posted when new data has been produced by the Hypercube. Similarly, when the high level application code has calculated some corrector settings, an event flag is posted and corrector settings are passed from VME memory to the appropriate elements in the simulation code running on the Hypercube. All data input and output operations in the application code are handled by channel access calls and therefore the code should run intact on the control room console when the accelerator is commissioned.

## SIMULATOR ENGINE

The simulator engine consists of an 64 node Intel IPSC/860 Hypercube parallel processor running a powerful simulation code based on the TEAPOT tracking algorithm described in Ref. 1. The Hypercube is a 64 node, distributed memory, MIMD computer. It utilizes the I860 RISC processor on each node. Each node executes the tracking code at approximately 8 MFLOPS in double precision. All 64 nodes therefore constitute a dedicated facility operating at approximately 0.5 GFLOPS. The nodes operate with NX, a subset of UNIX. It is a single process operating system with significantly reduced capabilities relative to the full UNIX implementation. It does however allow socket connections to individual nodes. Communications internal to the engine are done using a proprietary message passing library

The simulation code in an element by element tracking code which exactly integrates the equations of motion in a symplectic manner. The code models a real accelerator lattice with assigned errors in virtually all the lattice components. The code also simulates the operational correction process whereby lumped element correctors are set to compensate for assigned random and systematic errors. The simulation model includes various diagnostic devices, most notably BPMs.

The operational mode of the simulation code has been modified to more closely resemble an operating accelerator. Particles can be injected and tracked for a predetermined number of turns, until the are lost or tracked until an event is posted on the control console signaling additional information is to be transferred During the tracking, the system of BPM's is measuring the beam centroid position at all locations on a turn by turn basis, and sending this information to the EPICS control system. For most applications, it is sufficient to track between 16 and 64 particles in an ensemble.

## OPERATOR INTERFACE

The simulator uses the graphical user interface that is integral to EPICS. Thus, the simulator has the same look and feel as the actual operating software. The graphical user interface executes on the Control Console and communicates with the UNIX world through channel access calls.

At the present time, several high level correction modules have been written that deal with first turn injection into the LEB. Eleven modules have been written thus far as prototypes which exercise the simulator's capabilities. Figure 2 shows a screen from the operating simulator

The screen demonstrates some of the capabilities and limitations of the OPI interface. The column of buttons on the right show the individual correction modules which may be invoked from the main control screen. The general procedure is to execute the subroutines in the order they appear on the screen. The text windows to the left of the cartesian plot indicate the operation taking place and wether or not it is complete. The upper plot indicates the particle trajectory after the corrector settings shown in the bottom plot have been implemented. The correction operations require 10 to 30 seconds to execute on the simulator which is similar to the actual production code.

The text window pair associated with the top graph show maximum displacement of the horizontal and vertical trajectories in mm. The text window pair associated with the lower graph shows the maximum corrector strengths in radians.

First experience with the simulator has already yielded useful information on various features of the OPI. For

example, the scale size can be set only at compile time and the caption size cannot be set by the user at all. This is in no sense a serious problem but it does indicate the utility of the simulator in identifying inadequate or undesirable features of EPICS in general and the OPI in particular.

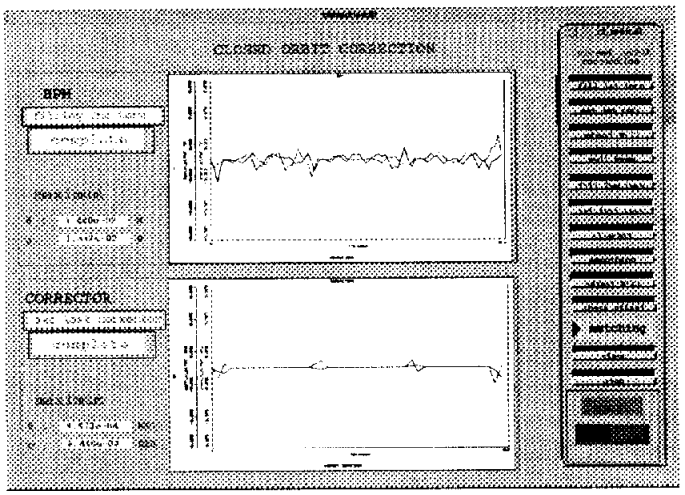


Figure 2 Typical Simulator Screen

## CORRECTION MODULES

The operation of the simulator begins with an operator sitting in front of the control console workstation shown in figure 1. He or she starts initiates the simulation code on the hypercube and initiates the simulator process on the workstation. A screen as shown in figure 2 appears at the work station and the operator pushes one of the buttons shown on the right of the screen. The simulator process reads the required information from the EPICS database, performs the required calculations and writes out modified corrector settings to the Hypercube code by way of the EPICS database.

At the present time there are two general classes of operational code that have been developed for the simulator. The first is a set of modules that deal with establishing a smooth closed orbit starting from injection into a completely uncorrected lattice with an unknown beam offset and unknown calibration of the main dipole field. The second is a tune measurement module.

The first class of code consists of eleven separate modules. The operator interface for this class is shown in Fig. 2. The individual modules are listed in table 1.

**Table 1: Correction Modules**

| Module Name | Function |
|---|---|
| fill 1st turn | inject a beam and track 1 turn |
| set 1st corr | set the first 4 correctors to zero injection offset and injection angle at entry. |
| adjust $B_o$ | adjust dipole field based on 2 BPM's |

**Table 1: Correction Modules**

| Module Name | Function |
|---|---|
| pull beam | set 4 additional correctors at beginning of super-periods |
| fill 2nd turn | inject and track 2 turns, save trajectory from 2nd turn. |
| set last corr | set last4 correctors to zero beam offset and angle at entry of 2nd turn. |
| closed orbit | Find closed orbit by averaging particle trajectories over 128 turns. |
| smoothing | smooth closed orbit |
| adjust $B_o$ | adjust Bo field so average corrector strength is zero |
| check offset | find injection offset by observing trajectory oscillations about closed orbit. |
| Matching | Correct injection offset |

## FUTURE PLANS

The simulator will also serve as a realistic environment in which to determine data handling overheads and operational bandwidth within the EPICS control system. The simulator will be extended to include a data buffer which will record data produced by the Hypercube and subsequently write it out in a "burst" mode which will equal the rate at which data is produced by the actual instrumentation.

The simulator will also accurately represent the hardware configuration that will be encountered in the LEB. Specifically, the current LEB control system plan calls for one IOC in each of the 12 sector houses and one concentrator IOC that communicates with the low level IOCs. The high level application code will access information from the concentrator and the lower level IOCs will be invisible except for the added delay in transferring data. The actual delays in this kind of configuration are not known at the present time and their determination is vital to predicting closed loop response times. It is anticipated that this work will be completed in the next few months.

## REFERENCES

1. L. Schachinger and R. Talman, Teapot: Thin-Element Accelerator Program for Optics and Tracking, *Particle Accelerators*, 1987, Vol. 22, pp. 35-56.