Creating an Interactive Simulation Environment Using a TMS320C40 Multi-DSP System

Authors: J. Schinneri, C. Steger, Dr. R. Weiss

EFRIE, France

December 1995
SPRA307

TEXAS INSTRUMENTS

TRADEMARKS

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Creating an Interactive Simulation Environment Using a TMS320C40 Multi-DSP System

Abstract

This application report describes a real-time high performance simulation environment using the Texas Instruments (TI™) TMS320C40 digital signal processor (DSP). The interactive simulation environment provides a diagnostic system for measuring, sampling, and diagnosing actual technical processes.

The real-time diagnosis of hardware errors reduces the investigative time required to identify errors and increases the opportunity for a quick response to correcting such conditions.

This document was an entry in the 1995 DSP Solutions Challenge, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Solutions Challenge, see TI's World Wide Web site at www.ti.com.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Introduction

A Hierarchical Model-Based Expert System

The aim of this project is the model-based diagnosis of faults in a production system. We use a real-time measurement system to get the state of the process and to react in a short time to avoid high repair costs. The behavior of the process is simulated using different simulators.

The main simulator is a Petri-net simulator that calculates the behavior of the entire process. [1] The Petri-net is a discrete simulator and, especially in the case of discrepancies between the state of the Petri-net simulator and the technical process, a continuous simulator is required. This simulator allows the expert system to receive more information about sub-processes, such as the pneumatic cylinder.

To get a high performance diagnostic system, the simulators as well as the diagnosis algorithm are distributed on a multi-transputer system. For the continuous simulator, a multi-DSP system comprising four TMS320C40 DSPs is used. The distributed diagnosis system local experts are implemented in process-interfaces located near the sub-processor. [2] One global expert controls the overall system.

Overview of the Simulation-Environment

The continuous local expert tries to diagnose the state of a process by comparing simulated data to data received by monitoring one or more signals of the process. In this case, simulation is continuous and requires a suitable mathematical model of the process.

The initial step is always the adaptation of the simulation model to the actual physical state of the process to obtain a normal state of the process. A deviation of the measured signal from the simulation is considered to be a fault. Furthermore, it is possible to derive parameters of the process that have changed and have caused the fault. Consequently, we can identify faulty components if the changed parameter belongs to the model description of this component.

The overall structure of the local expert comprises the following tasks, mostly implemented within independent modules:

Simulation	For each examined technical process, we maintain a simulation model [?] that represents the normal behavior of the process.

Measuring	We identify signals whose values can be measured by sensors and A/D converters and thus can be recorded over a period of time. Any faults that might occur in components of the process should be detectable by analyzing the signal data.
Sample	To compare different traces of a signal from simulation and measuring, significant characteristics called samples must be extracted from the data.
Diagnosis	After fine-tuning parameters of the simulation model, we define appropriate samples that help to analyze the behavior of the process. Different sample results for simulated and measured data indicate faulty components.

Implementation on the Hydra Board

The Hydra board is a VME-based multi-DSP card with four TMS320C40 DSPs from Texas Instruments. Each DSP chip is directly connected to all other processors via its 8-bit wide parallel, bidirectional communication ports. The remaining three out of the six ports on each DSP are carried out to the front panel of the Hydra. They can be used to connect other Hydra boards or interface cards. To monitor signals from manufacturing processes, a prototype of a multi-I/O card containing four analog-to-digital (A/D) converters (among other things) are attached to the Hydra in such a way.

The software on the Hydra is implemented under the distributed *Virtuoso* real-time operating system.[3] Special care should be taken to find a modular structure of the implementation that accommodates best to the layered operating-system kernel of Virtuoso.

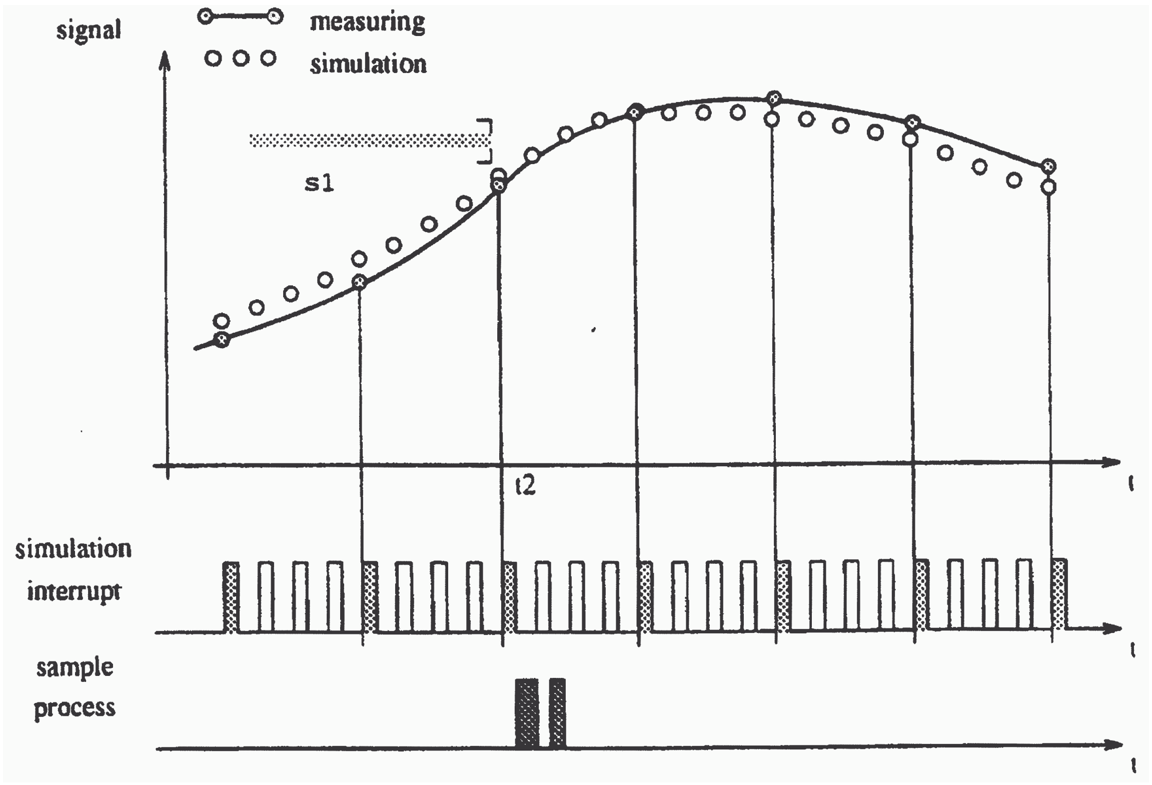
Time critical tasks are carried out within lower levels of the operating system. The tasks offer a small but powerful set of services which exploit the capabilities of the underlying hardware. The implementation of other tasks takes advantage of a comprehensive set of microkernel services for process management and comfortable distribution interprocess communications.

Interrupt Level

Interrupts are usually invoked by events that need to be handled without delay. The execution of interrupt routines always takes place with the highest priority. Since routines for measuring and simulation must meet hard real-time requirements, they are implemented as interrupt handling routines triggered by the hardware timer. To cope with large simulation models, we built in the option to distribute the simulation task to a maximum of three signal processors that exchange results via communication ports.

Figure 1 depicts the process of simulation and measuring of a signal. In most cases measuring needs to be done less frequently than computing simulation data, since the time step of the simulation has to be set to a very small value to achieve a reasonable accuracy. As soon as all data within the time interval of a sample (*s_1*) is available, a dedicated process of the nanokernel starts the computation immediately.

Figure 1. Simulation, Measuring, and Sampling



Virtuoso Nanokernel

The nanokernel is a local, hardware-dependent layer of the operating system that offers dynamically allocable processes and a reduced set of simple but efficient communication primitives.

Samples are computed at this level by nanokernel processes. Their task is to extract characteristics of the monitored signals within a time interval basis upon the data collected by simulation and measuring. Such characteristics may be maximum or rate of change of a signal or may incorporate the detection of non-linear portion of the signal. Samples can be defined and adjusted interactively by the user at run-time of the program.

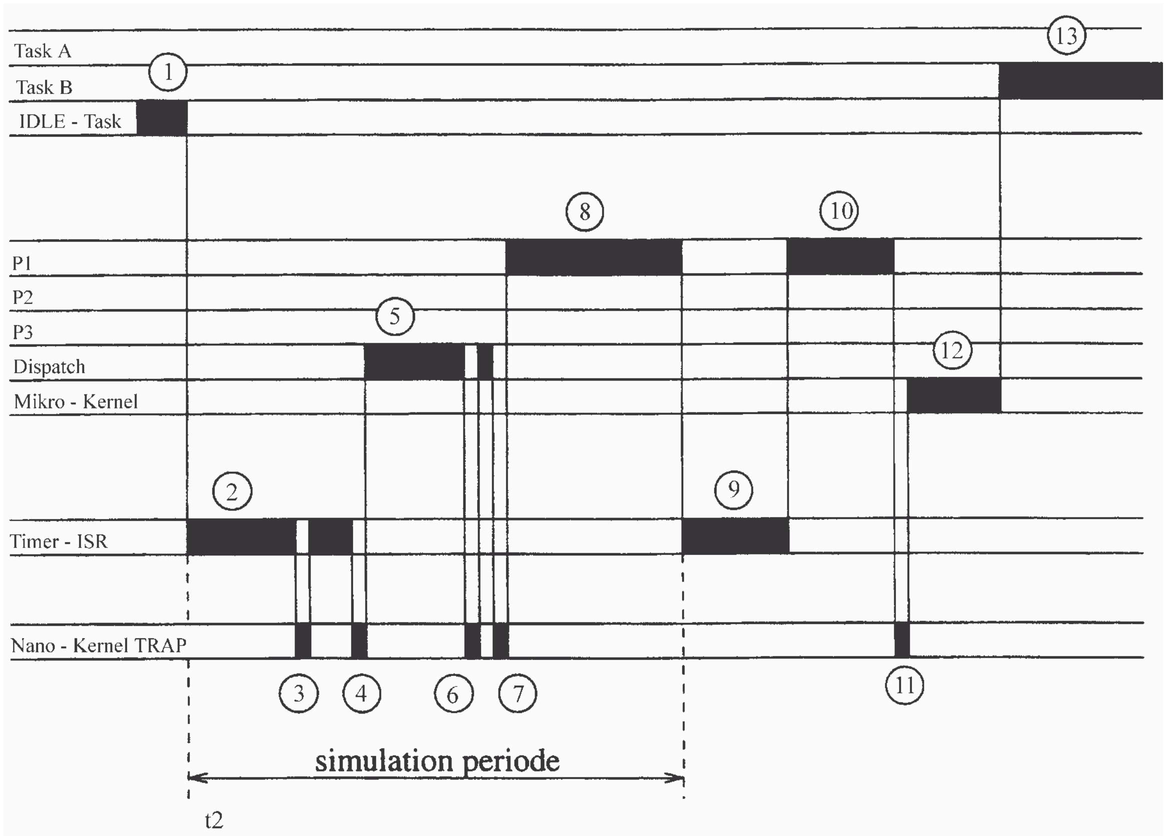
Virtuoso Microkernel

This is the standard C level offering a full set of distributed kernel services. Tasks are fully pre-emptive and priority driven but must be statically declared in a configuration file at compile-time.

An implementation of basic I/O operations allows communication with a host workstation extended to cope with Berkeley sockets. The control of activities at lower levels of Virtuoso and all client interactions was implemented through microkernel tasks. All diagnosis is done within a task based on the data prepared by lower level processes.

Figure 2 illustrates the interactions between the modules of the program executed at different levels of the operating system.

Figure 2. An Execution Trace of the Program



The highest priority is assigned to the simulation routines triggered by the hardware timer. Its interrupt preempts any activity on higher levels (1) and computation of simulation data is carried out with minimal delay (2). Sending a command word to the interface card and receiving the data via a DSP's communication port carries out simultaneous measuring of the signal data.

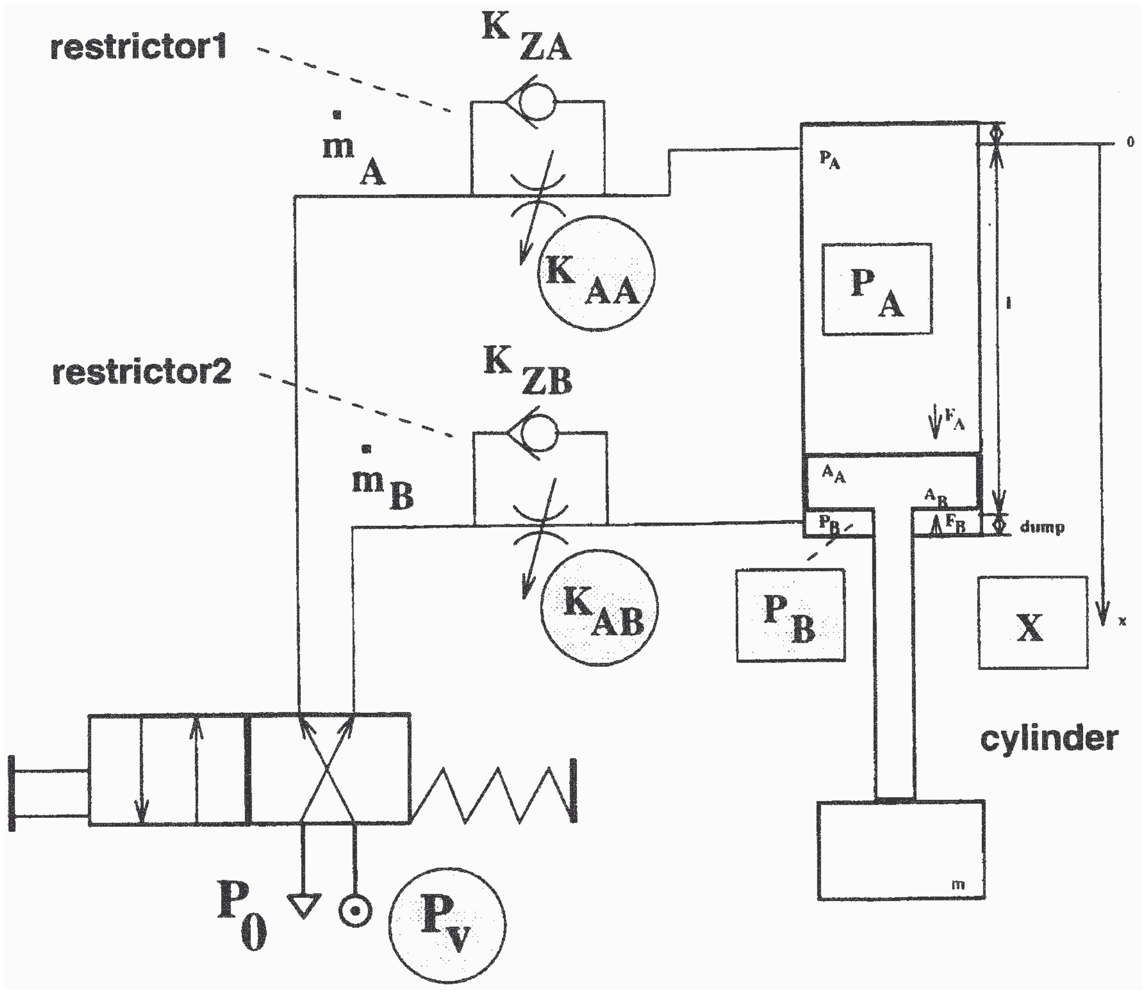
Before the control over the processor is returned to the preempted task or process (4), a nanokernel semaphore is signaled from out of the interrupted routine to notify new data to the dispatcher. The nanokernel process (5) keeps track of the progress in the simulation. It activates a sample process if enough data is gathered.

Each sample process computes some characteristics of the signal curve and can not be pre-empted by tasks or other nanokernel processes, only simulation routines (9) may interrupt at any time. The sample process stores the results of its computation into a global data structure and signals a microkernel semaphore. Any task (13) may wait for that semaphore and start to evaluate the results.

Practical Applications

The diagnosis system can be applied to components of a modular production system. Figure 3 depicts the scheme of simple pneumatic drill consisting of a pneumatic cylinder, valve, and two restrictors.

Figure 3. Pneumatic Drill of the MPS



Monitored signals are the air pressures in two chambers of the cylinder (P_a, P_B) and the vertical motion (x) of the drill. The physical parameters (K_{AA}) and (K_{AB}) of the restrictors influence the behavior of the drilling process.

The next section shows two practical applications of the program, which is a helpful tool in developing and fine-tuning of simulation models. On the other hand, its intended field of application is the online-diagnosis of active processes based on an accurate model.

Modeling

We built a server for simulation and to measure routines that interactively offer commands to control their execution. An important feature is the possibility to change the values of any parameters of the simulation model at run-time, thus adapting the behavior of the model.

Another server allows the definition of samples and the query of their results. Even the traced data of the monitored signal is made accessible to a special client running on the host workstation whose task is to plot out signals to a window.

All of these software features aid in the task of modeling, which otherwise could be a time consuming job of incorporating steps (such as editing model description files and recompiling the whole program for every new parameter value). Modeling is done by iteration with the following steps:

1) Adjust parameters of the simulation model.
2) Start real-time simulation.
3) Compare function plots of simulation to measuring and stop if the result is satisfying.

Under certain circumstances the job described above is entirely handled by a built-in function of the program. If the behavior of a signal within a time interval is mainly determined by a single parameter the program finds the appropriate value of the parameter with simple search algorithms.

Future versions of the program might extend these ideas to the handling of more complicated interdependencies in the simulation model and should implement more intelligent modeling algorithms. Figure 4 and Figure 5 present the results of such a modeling session for the simulation model of the pneumatic drill.

Figure 4. Drill Motion before Adaptation

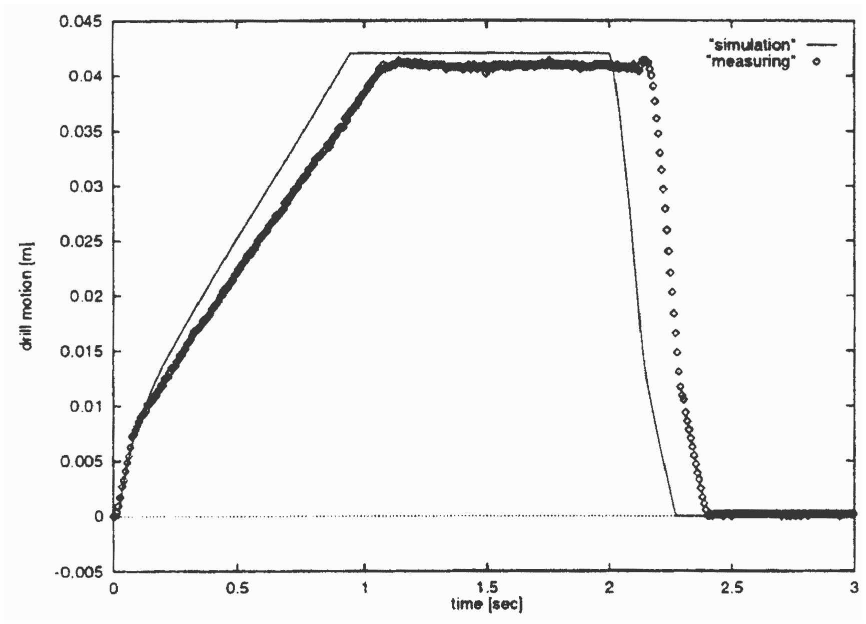
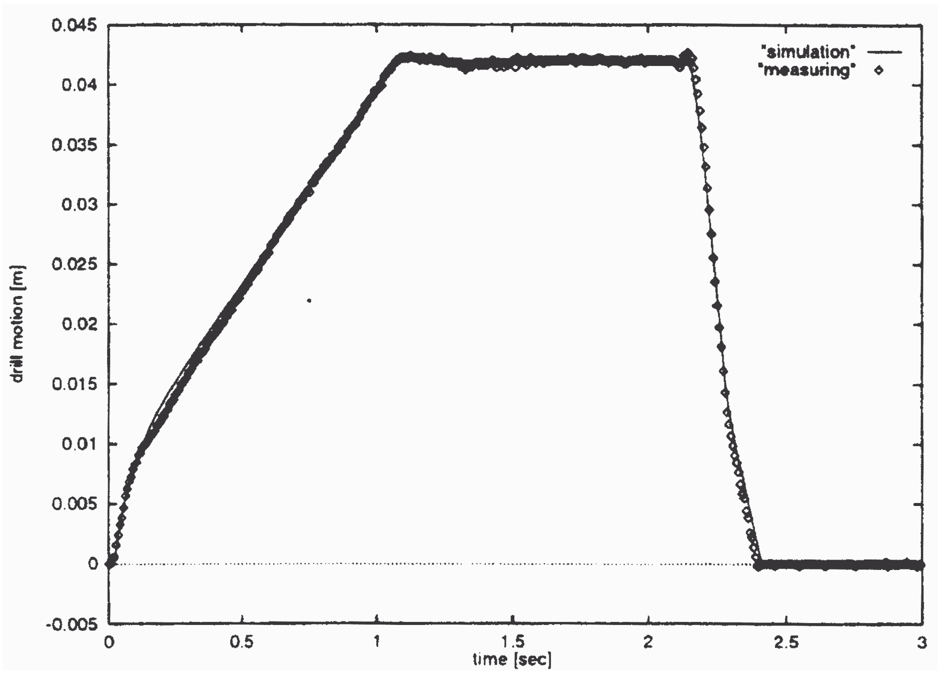


Figure 5. Drill Motion after Adaptation



Diagnosis

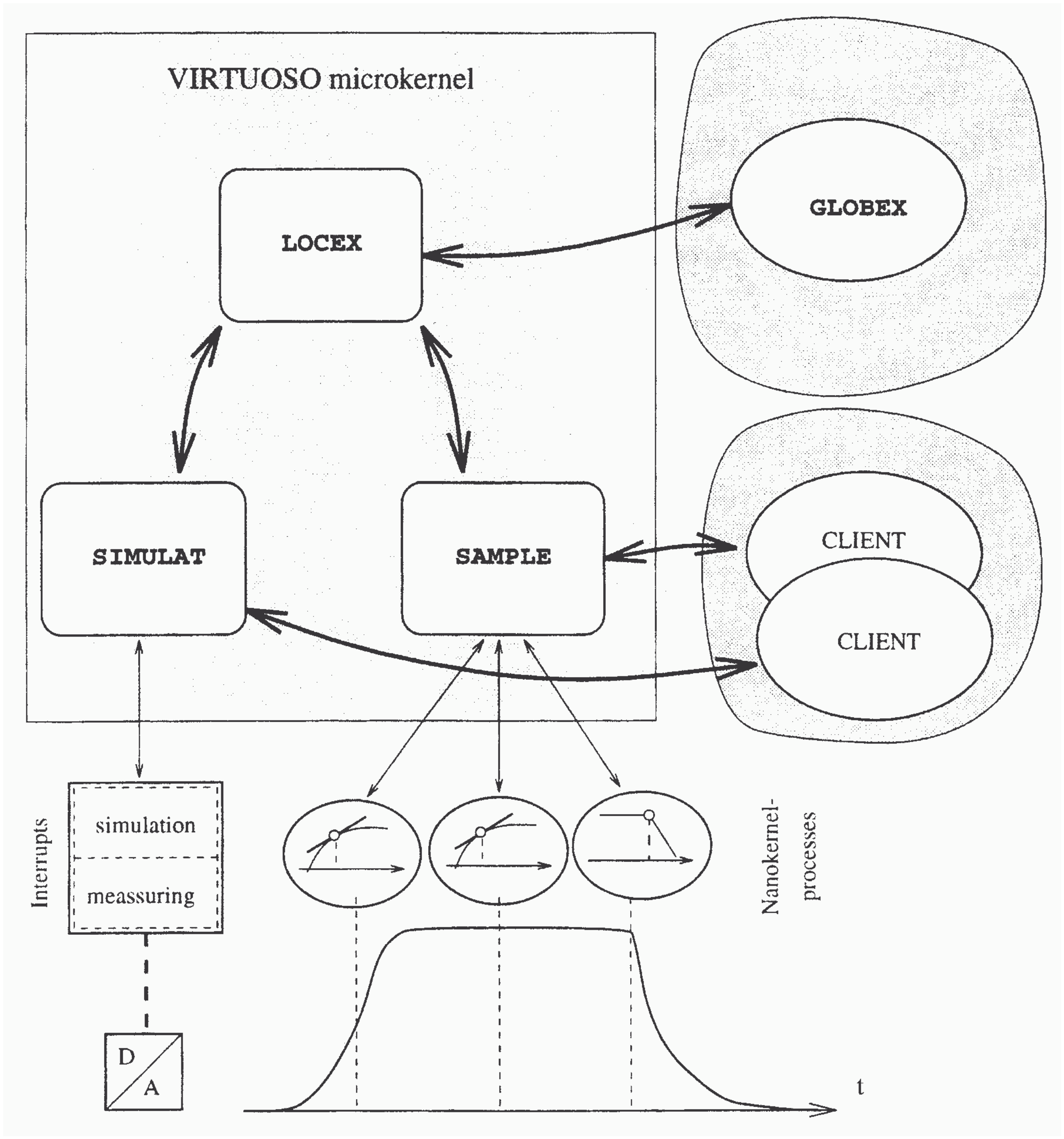
Diagnosis of a process incorporates the following tasks:

❑ Adapt the parameters of the simulation model to the actual physical state.

- ❑ Evaluate data from simulation and measuring and derive the state of the process through comparison.
- ❑ Collect information about the process and yield them to a global expert module.

Most of these functions of the expert system (module LOCEX) need to be implemented specifically for any new monitored process. The expert has access to all functions that are offered by the server tasks of simulation and sampler. Figure 6 shows the client server architecture of the overall system.[4]

Figure 6. Overall Structure of the Continuous Diagnosis System

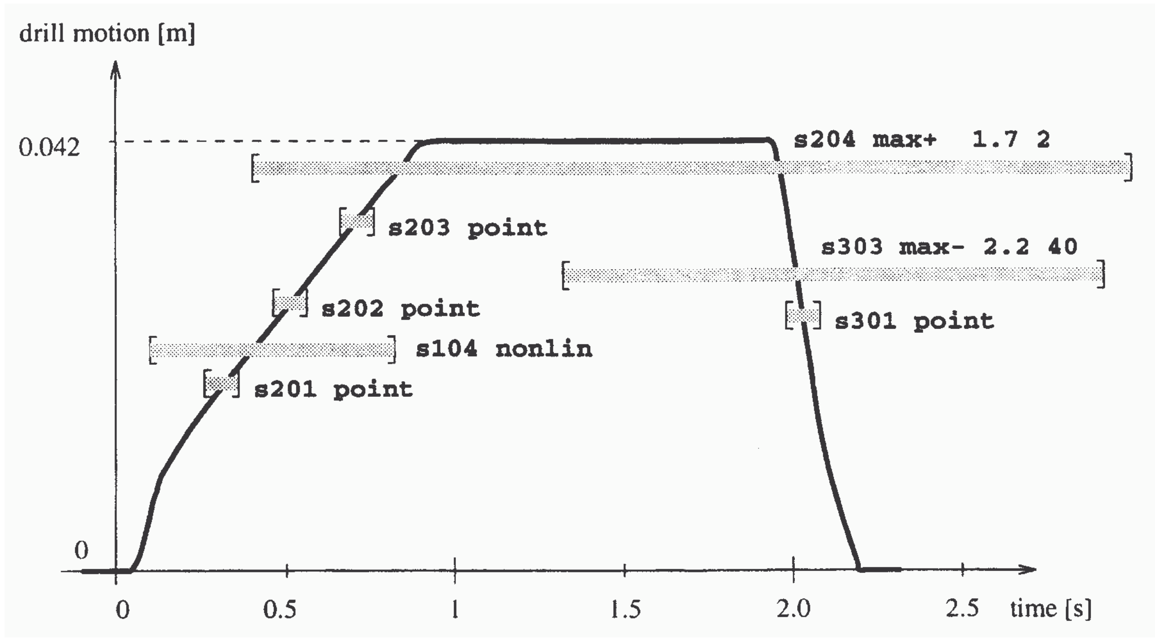


In addition to the local expert, any client running under Virtuoso or on the host workstation is able to connect to the server tasks on the Hydra board. A special client running under Helios, the global expert (GLOBEX) acts as a client of the local expert and receives the expert's diagnosis information.

The simulation model can be adapted as shown in the foregoing section, but that job has to be written as a C function, which then has to be linked to the program. Since a specific simulation model has to be developed for each monitored process, we should have comprehensive knowledge of the physical conditions and interrelations in the monitored process. This knowledge can be applied for diagnosis.

The goal is to gather enough information about the actual state of the monitored process to determine faulty components exactly and also the kind of fault. Appropriate samples analyze the behavior of the simulation model and the real process by extracting characteristic features of their signals (see Figure 7).

Figure 7. Samples for Analyzing the Signal



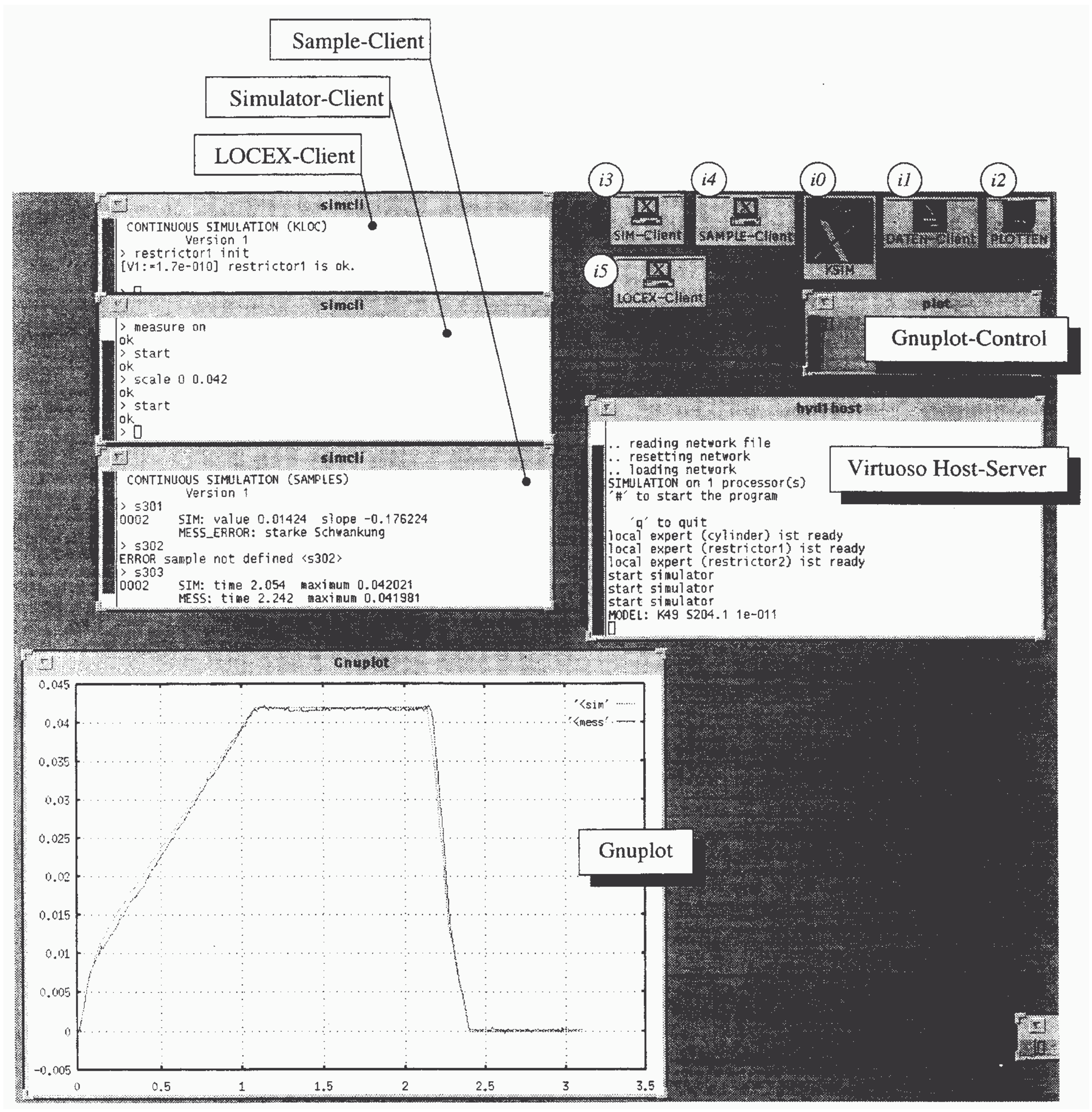
Different sample results indicate a deviation of the monitored process from the nominal state. Further examination of the samples can even yield the physical parameters that have changed, thus identifying the faulty components.

One important issue when implementing a local expert is its capability to distinguish between the normal time-dependent behavior of a process and severe faults. Not every deviation of the real process from the simulation model can be treated as a fault. It is possible that physical parameters slightly change their values but stay within allowed tolerances. The local expert must identify these cases.

User Interaction

Interactive control of the simulation environment is performed via clients on the host workstation by means of simple command-line interfaces. Figure 8 shows connections to such clients on the server modules depicted in Figure 6.

Figure 8. User Interaction with the Simulation Environment



```
CONTINUOUS SIMULATION (KLOC)
        Version 1
> restrictor1 init
[V1:=1.7e-010] restrictor1 is ok.
```

```
> measure on
ok
> start
ok
> scale 0 0.042
ok
> start
ok
>
```

```
CONTINUOUS SIMULATION (SAMPLES)
        Version 1
> s301
0002    SIM: value 0.01424  slope -0.176224
        MESS_ERROR: starke Schwankung
> s302
ERROR sample not defined <s302>
> s303
0002    SIM: time 2.054  maximum 0.042021
        MESS: time 2.242  maximum 0.041981
```

```
.. reading network file
.. resetting network
.. loading network
SIMULATION on 1 processor(s)
'#' to start the program

   'q' to quit
local expert (cylinder) ist ready
local expert (restrictor1) ist ready
local expert (restrictor2) ist ready
start simulator
start simulator
start simulator
MODEL: K49 S204.1 1e-011
```

Requests that the Simulator-Client can direct to the server SIMULAT include the starting of simulation and measuring passes and adjusting of parameters of the model. The Sample-Client connects to the server SAMPLE that allows the definition of samples and the query of their results.

When preparing a local-expert module for automatic diagnosis of a process, you should first use this interactive environment to:

- ❑ Study the influence of parameters on the behavior of the process.
- ❑ Extract and define appropriate features.
- ❑ Fine-tune model parameters.
- ❑ Find the best way to adapt the model to varying process behaviors.

When this initial step is done the knowledge about the process must be coded into the LOCEX module, which is then responsible for doing the on-line diagnosis of the process. During on-line operation it is still possible for a user to connect to all server-modules of the simulation environment. Even the local expert LOCEX is accessible (LOCEX-Client) for interactive requests of diagnosis results.

A server for raw signal data (not depicted in Figure 6) transfers simulated and measured data to the host on request of a data client, which can save the data to the local disc or pass it on to tools like GNUPLOT for visualization.

Summary

This paper presented the implementation of an interactive simulation environment. The simulator was used as a reference model in a real-time expert system for fault diagnosis in manufacturing systems. By analyzing assembly systems and manufacturing plants, it was shown that the basic structures of these systems are similar.

Consequently, we used as a model the Modular Production System (MPS) by FESTO DIDACTIC. The MPS is controlled by storage programmable controllers. The simulator was coupled to the technical process. Further researches will be directed to perform experiments with respect to the automatic diagnosis of faulty components in the technical process such as incorrectly positioned sensors.

A major goal for the next year is migrating all tasks of the overall diagnosis system, which now are distributed across Unix workstations, a Transputer network, and multi-DSPs, onto DSP-based boards running the Virtuoso operating system. We expect to get a modular and highly scalable real-time architecture for our diagnosis purpose that should be easily adaptable to any technical process.

References

[1] Ch. Steger. “Implementation of a Petri-net Simulator on TMS320C40,” *Parallel Processing in Education7*, pages 115-119, Mickoic-Tapolca, Hungary, March 1993. Impact Tempus JEP’s and Hungarian Transputer Users Groups.

[2] Ch. Steger and R. Weiss. “A Model-Based Real-Time Diagnosis for Technical Processes,” in *Proceedings of The 10th International Conference on Applications of Artificial Intelligence in Engineering*, pages 145-152, Udine, July 1995.

[3] Intelligent Systems International Inc., Linden, Belgium, *Virtuoso-User Manual*, 1993.

[4] G.F. Coulouris and J. Dollimore. *Distributed Systems, Concepts and Design*, Addison-Wesley Publishing Company.