



GEDAE™

GEDAE™ and its Benefits

GEDAE is a software development productivity tool based on a block diagram programming paradigm. The versatility of the tool makes it available for all aspects of development including signal and data processing, application control, and integration with other systems. Through the use of a block diagram you specify functionality, and GEDAE implements the application by generating code targeted to specific embedded hardware. The implementation rivals hand-coded efficiency and quality by building applications on layers of abstraction: board support packages, runtime kernel, static and dynamic schedules, data-flow graphs, and command programs. GEDAE provides all the tools needed to easily develop an application, while

improving productivity by 5x, increasing the pool of qualified developers, and reducing life cycle costs.

Figure 1 illustrates the developer productivity when using GEDAE relative to productivity when coding in C. The cost of developing an application increases dramatically when the required usage exceeds 70% of CPU or memory. With GEDAE the increases are much less dramatic. The productivity improvement grows even greater when the resources are heavily loaded. In one case, a small group of developers – not trained in multiprocessor programming – achieved >95% of memory and CPU usage on a system of 72 Sharc processors. Our customers have repeatedly verified the reduction in development cost realized by using GEDAE.

Figure 2 illustrates the increased pool of developers who are able to build efficient applications running on embedded multiprocessor systems. It is easy to map to these systems, to analyze execution on these processors, and to remove inefficiencies.

Figure 3 illustrates the increase in productivity of GEDAE users over the life cycle of a product. This increase is contrasted with the decreasing productivity of developers using C-Code. Once the developer understands and implements the data flow of the algorithms and control, the modifications are simple. This simplicity contrasts with the very difficult task of modifying a C-code implementation.

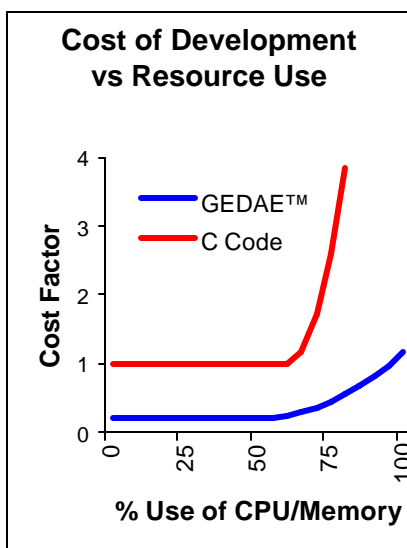


Figure 1 - GEDAE Users are more than 5 times as Productive

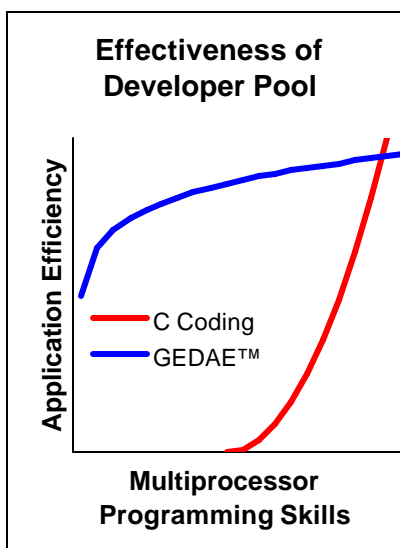


Figure 2 – More Developers are able to Effectively Program Target Hardware

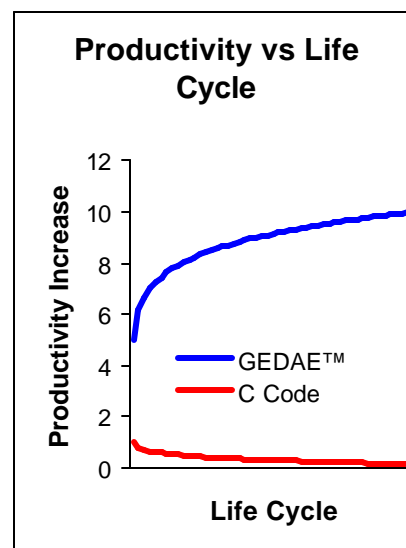


Figure 3 – The Productivity of GEDAE™ users increases during the system Life Cycle

The GEDAE Development Process

The key to GEDAE's development process (figure 4) is the ability to iterate between the major steps. The opportunity to iterate obviates the need for the waterfall development process and its associated inefficiencies. It is quite reasonable to consider changes in requirements, design, and implementation at any point during the development or the life cycle of a product.

One of the reasons that iteration is possible is the independence of developing functionality and specifying the implementation. GEDAE identifies constraints that are imposed by the data flow and limits the implementation accordingly. Another reason is that GEDAE handles all the complex

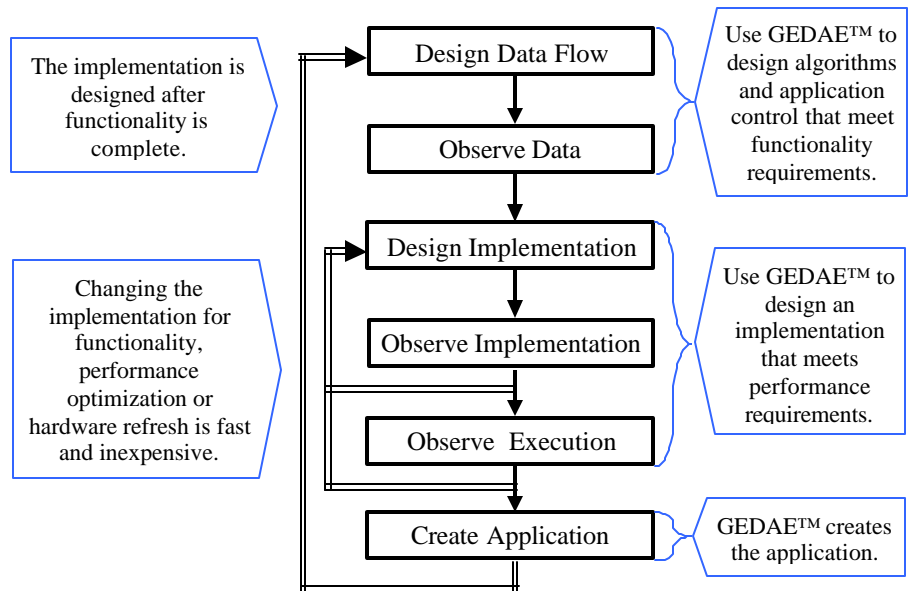


Figure 4 - The Application Development Process when using GEDAE

issues associated with streaming data, granularity changes, dynamic data flow, distribution, and data transfers. When you change the

implementation specification, then GEDAE addresses all these issues and completes the implementation within seconds.

Using GEDAE's Features

GEDAE provides intuitive interfaces for:

- Designing data flow
- Controlling implementation

- Observing functionality, implementation, and performance
- Generating a stand-alone application

GEDAE has its own language that is easy to understand. The block diagram graphical language (Figure 5) expresses your algorithm.

Various graphical displays, such as scopes and constellation diagrams, can be used to analyze and verify your algorithm.

Once you've verified your algorithm's behavior, you are ready to specify the implementation and optimize its performance. For example, mapping an application to embedded hardware takes just a few steps. From the GEDAE Group Control dialog (Figure 6 – next page), you can launch the Partition Table and the Mapping Table. The Partition Table allows you to assign boxes to partitions (p_0 and h are shown, $source$ is partitioned between src , p_1 and p_2), and the Mapping Table allows you to map the named partitions to particular processors (h is assigned to the host processor, src is assigned to processor 100 and p_0 , p_1 and p_2 to processors 101 , 102 and 103). When you rerun your graph, the results are the same, but the throughput is increased. GEDAE

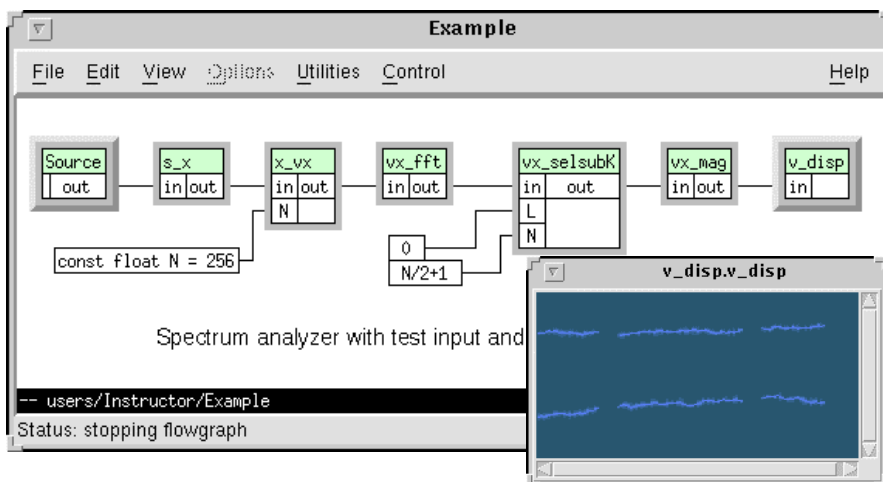


Figure 5 – A GEDAE™ data-flow graph clearly expresses the algorithm functionality.

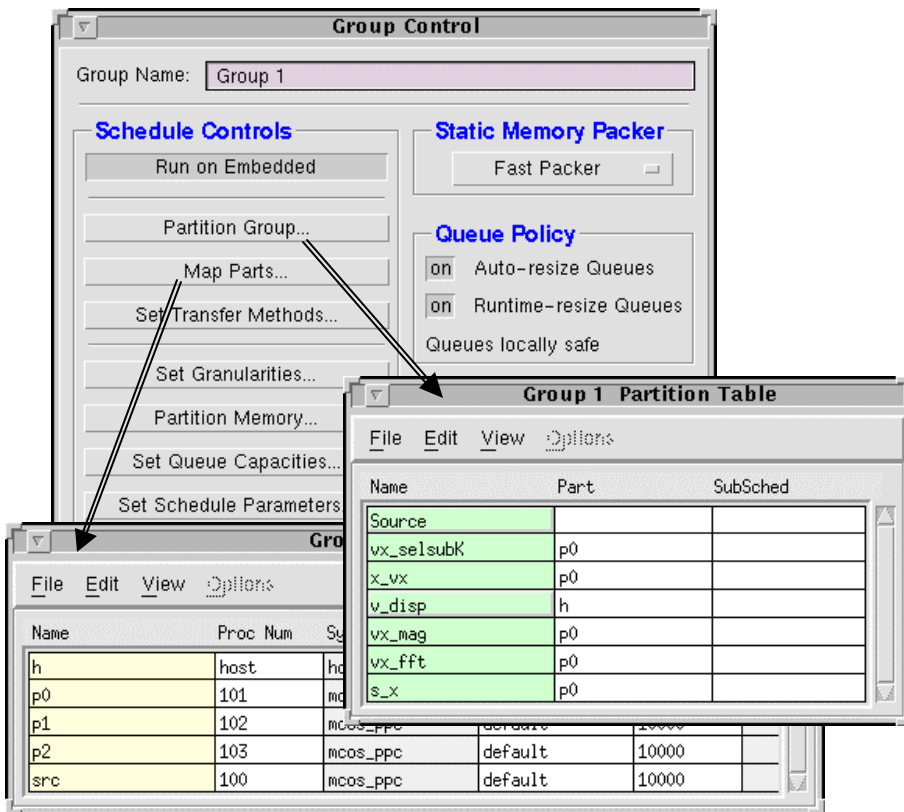


Figure 6 – Partitioning and mapping graphs increases performance without changing behavior.

adds all the necessary communications between processors to make your graph work.

You can also modify almost every aspect of an implementation using Group Control dialog tools, letting you optimize a graph's performance without changing its functionality. The optimization tools allow you to set transfer method parameters, set box firing granularities and priorities, partition and map memory, adjust dynamic queue sizes, and set dynamic scheduling policies. While GEDAE automatically sets all optimization parameters, you can modify these parameters to get the performance you need.

A variety of tools allows you to observe the implementation and to monitor performance, giving you the feedback needed to use the optimization tools. A display

showing the schedule of events is available, detailing all memory that is used and the sequence of

operations that will be performed to implement the data flow graph. During application execution, events can be collected and then displayed in a Trace Table. A Trace Table for the application in Figure 5, as implemented in Figure 6, is seen in Figure 7. Details, such as when a box begins and ends executing (black boxes) or when a box is blocked (white boxes), can be seen in the table. Processor loading and data throughput can also be observed. By using diagnostic tools like the Trace Table, your implementation modifications can be immediately evaluated.

Once the design and implementation of an application is finalized, the application can be divorced completely from the GEDAE design environment by generating a *launch package*, which can be loaded and controlled from a program you write using the GEDAE Command Program Interface.

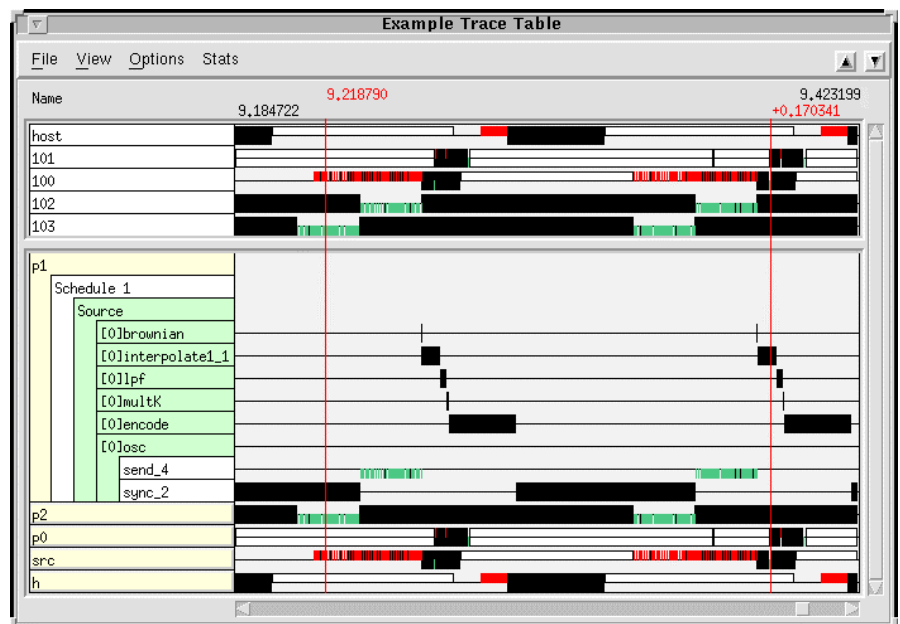


Figure 7 – The Trace Table gives you the information you need to optimize your application

Capabilities

GEDAE's intuitive design environment inherits and enhances all the power of the C language. Boxes modularize your C-code and can encapsulate any of the following behavior:

- Application Control — Dynamic behavior can be incorporated into a box, making application control possible. This ability enables everything from mode control through dynamic load balancing.
- Interface with Input/Output (I/O) — I/O devices can be interfaced through GEDAE function boxes so that graphs can do real time data processing.
- Graphical User Interfaces (GUIs) — A box library is included with GEDAE, so users can design GUIs as block diagrams.
- Dynamic Load Balancing and Fault Tolerance — Boxes with dynamic inputs and outputs allow graphs to be constructed with load balancing and fault tolerance.
- Interface with MATLAB® — A box library is included with GEDAE so that users can interface the libraries and displays of MATLAB® in their GEDAE graphs.

Blue Horizon is releasing a beta version of a performance simulator with GEDAE 3.3.1. This release primarily addresses parallelization issues. Blue Horizon will release other additions to the GEDAE product line in the future.

Supported Host and Embedded Hardware

GEDAE is currently supported for Solaris™, Windows® 2000, and Redhat Linux® workstations; VxWorks® embedded host, and CSPI, Ixthos, Mercury, and Sky multi-processor systems.

Blue Horizon offers a board support package development kit that allows you to develop a BSP for your custom hardware. The effort to implement a BSP is modest.

Application Areas

Blue Horizon customers are using GEDAE to implement a variety of applications. Some of the current applications are:

- RADAR — signal and data processing, mode control, and dynamic resource allocation.
- SONAR — signal and data processing, mode control, adaptive algorithms and application control.
- Audio — signal and data processing, interactive algorithms and application control.
- Image — compression and classification.
- Communications — software radio.
- Electro Optics — signal and data processing, mode control, adaptive algorithms and application control.

Blue Horizon Development Software, Inc.

Blue Horizon Development Software, Inc. was formed in January 2001 to complete the commercial development of GEDAE. We will continue to improve and expand our software's capabilities. Our mission is to make it easier for customers to develop digital signal processing applications and products. We provide excellent technical support and training to help you effectively use the many capabilities of GEDAE. Our success depends on the success of our customers.

While it's true we are a fairly new company, GEDAE is not a new concept to us. Our staff includes computer scientists who were responsible for the original version of our software in the late 1980's, and for the first commercially released version in 1997.

In this year alone, GEDAE has had over a 200 percent increase in users. Companies in both the United States and abroad have requested and paid for our instruction. This year our staff has trained more than one hundred computer scientists and engineers in the application of GEDAE to their company's individual needs.

To learn more about GEDAE™ contact Bill Lundgren

Phone: 856-231-4458 / Email: bill.lundgren@gedae.com

GEDAE is a trademark of Blue Horizon Development Software, Inc.
Solaris is a trademark of Sun Microsystems Incorporated
Windows 2000 is a registered trademark of Microsoft Corporation
VxWorks is a register trademark of Wind River Systems, Inc.
Linux is a register trademark of Linus Torvalds