A Compiler for HP VEE

Steven Greenbaum

Stanley Jefferson

With the addition of a compiler, HP VEE programs can now benefit from improved execution speed and still provide the advantages of an interactive interpreter.

This article presents the major algorithmic aspects of a compiler for the Hewlett-Packard Visual Engineering Environment (HP VEE). HP VEE is a powerful visual programming language that simplifies the development of engineering test-and-measurement software. In the HP VEE development environment, engineers design programs by linking visual objects (also called devices) into block diagrams. A simple example is shown in **Figure 1**. Features provided in HP VEE include:

- Support for engineering math and graphics
- Instrument control
- Concurrency
- Data management
- GUI support
- Test sequencing
- Interactive development and debugging environment.

Beginning with release 4.0, HP VEE uses a compiler to improve the execution speed of programs. The compiler translates an HP VEE program into byte-code that is executed by an efficient interpreter embedded in HP VEE. By analyzing the control structures and data type use of an HP VEE program, the compiler determines the evaluation order of devices, eliminates unnecessary run-time decisions, and uses appropriate data structures.

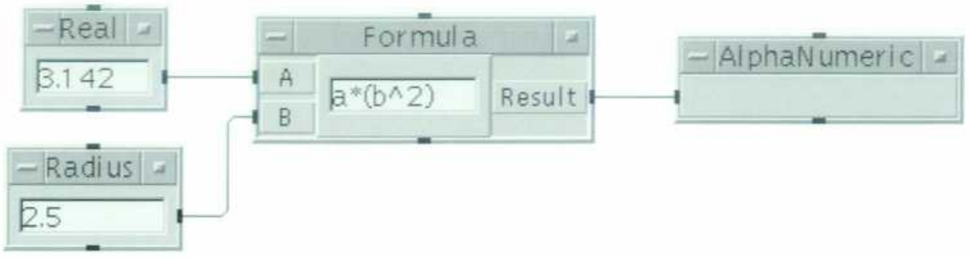
The HP VEE 4.0 compiler increases the performance of computation-intensive programs by about 40 times over previous versions of HP VEE. In applications

Steven Greenbaum
A member of the technical staff at HP Laboratories since 1989, Steve Greenbaum is currently researching "hardware-in-the-loop" systems and programming for distributed systems. He has a PhD degree in computer science (1986) from the University of Illinois at Urbana-Champaign and a BS degree in computer science (1980) from Syracuse University. Steve was born in New York City, is married and has two children. In his leisure time he enjoys playing guitar and taking field trips with his family.

Stanley Jefferson
Stanley Jefferson is a member of the technical staff at HP Laboratories, where he began his career at HP in 1990. He is currently doing research in the area of "hardware-in-the-loop" systems. He has a PhD degree in computer science (1988) from the University of Illinois at Urbana-Champaign. He received BS (1977) and MA (1979) degrees in mathematics from the University of California at Davis. Stan was born in Oakland, California, is married and has two children. He enjoys playing piano and day trips to the beach with his family.

Figure 1

A simple HP VEE program to compute the area of a circle.



where execution speed is constrained by instruments, file input and output, or display update, performance typically increases by 150 to 400 percent.

The compiler described in this article is a prototype developed by HP Laboratories to compile HP VEE 3.2 programs. The compiler in HP VEE 4.0 differs in some details.The HP VEE prototype compiler consists of five components:

- Graph Transformation. Transformations are performed on a graph representation of the HP VEE program. The transformations facilitate future compilation phases.
- Device Scheduling. An execution ordering of devices is obtained. The ordering may have hierarchical elements, such as iterators, that are recursively ordered. The ordering preserves the data flow and control flow relationships among devices in the HP VEE program. Scheduling does not, however, represent the run-time flow branching behavior of special devices such as If/Then/Else.
- Guard Assignment. The structure produced by scheduling is extended with constructs that represent run-time flow branching. Each device is annotated with boolean guards that represent conditions that must be satisfied at run time for the device to run. Adjacent devices with similar guards are grouped together to decrease redundancy of run-time guard processing. Guards can result from explicit HP VEE branching constructs such as If/Then/Else, or they can result from implicit properties of other devices, such as guards that indicate whether an iterator has run at least once.
- Type Annotation. Devices are annotated with type information that gives a conservative analysis of what types of data are input to, and output from, a device. The annotations can be used to generate type-specific code.
- Code Generation. The data structures maintained by the compiler are traversed to generate target code. The prototype compiler can generate C code and byte-code. However, code generation is relatively straightforward to implement for most target languages.

These components combine to implement the semantics explicitly and implicitly specified in an HP VEE program.

Online Information

This complete article can be found at:
http://www.hp.com/hpj/98may/ma98a13.htm

More information about HP VEE can be found at:
http://www.hp.com/go/HPVEE