1 Overview

Audience

This manual is designed for programmers who are developing parallel signal processing applications for the iWarp supercomputer. To use Assign, you should have a good knowledge of the C programming language. Knowledge of iWarp Programmed Communication Services (PCS) is not necessary unless combining array modules. For further information about PCS, refer to the *iWarp Programmer's Guide*.

Using This Manual

This manual is organized as follows:

Chapter 1	describes the manual and provides an overview of the Assign parallel program generator
Chapter 2	describes the structure of an Assign flowgraph and how to create a flowgraph generator
Chapter 3	explains how to compile and execute an Assign application
Appendix A	describes how to use Assign libraries
Appendix B	provides code for the adaptive beam interpolation (ABI) example used in chapter 2
Appendix C	describes XAssign, the Assign graphical user interface

Conventions

Documentation conventions

- *italic type* indicates items you must provide.
- `courier font` is used for program and command examples.

In a syntax description:

- **[ ]** square brackets enclosing qualifiers indicate the qualifiers are optional.
- **{ }** braces enclosing qualifiers indicate that one or more of the qualifiers is required.
- **|** a vertical rule separating qualifiers indicates that only one of the qualifiers can be selected.
- **...** a horizontal ellipsis indicates that the preceding optional item can appear one or more times in succession.

- ⋮ a vertical ellipsis indicates the continuation of statements within a program example.
- spaces are used between entities to improve readability in both syntax descriptions and programs but unless otherwise noted have no significance.

List of Related Documents

The following documents provide more information about the iWarp system:

- *Introduction to iWarp*
- *iWarp Programmer's Guide*
- *iWarp C User's Guide*
- *iWarp FORTRAN User's Guide*
- *iWarp Debugger Manual*

Assign Overview

Typically, when a programmer designs a parallel program, the programmer must build both the program and the programming model. This adds to development time and can result in a program that is difficult to change, since the programming model is specific to the program.

For this reason, parallel program generators have been designed to improve parallel programming. Assign is an application-specific parallel program generator for developing parallel signal-processing applications for the iWarp supercomputer. Assign takes a user-developed flowgraph and automatically generates a parallel program for the iWarp array, handling all of the following details:

- partitioning the flowgraph
- creating communication pathways
- transferring data from cell to cell
- handling flow control
- balancing the workload across processors and communication pathways

The Assign parallel program generator provides the following advantages for developing applications:

- hierarchical specification of flowgraphs
- ability to parameterize both flowgraphs and individual signal processing applications
- ability to model and compile large graphs with thousands of nodes
- a flexible and general-purpose mechanism for transferring data to and from the host
- a minimum of run-time scheduling overhead

The flowgraph is the underlying programming model for Assign. A flowgraph is made up of nodes and directed edges, where the nodes represent computations, and the edges represent FIFO queues. Each node in the flowgraph iterates an infinite number of times. At each iteration, the node consumes data from each of its input edges, performs a computation on the input data, and delivers data on each of its output edges. The sizes of the input and output data are indicated by integer edge labels. Figure 1-1 illustrates an example flowgraph.

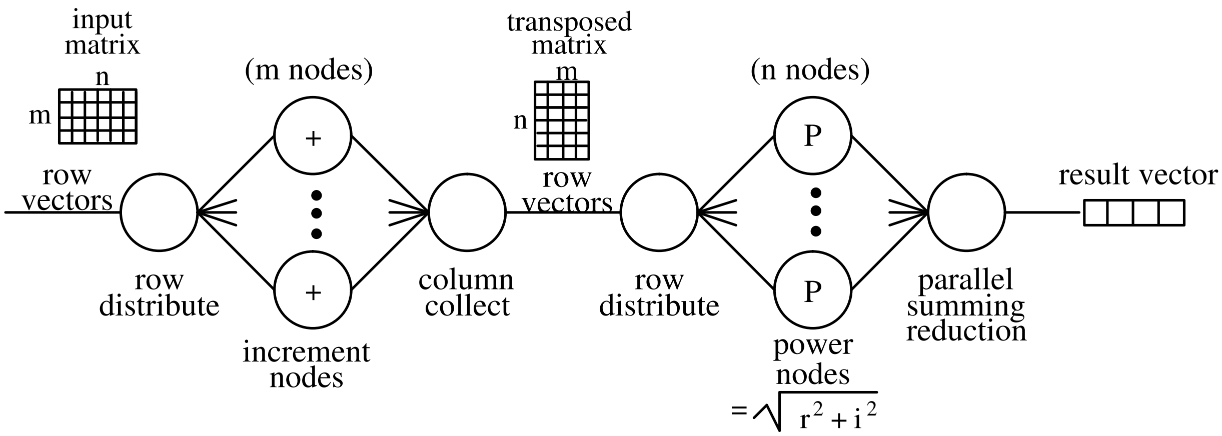

Figure 1-1: Example Flowgraph

In the first part of the example flowgraph, the rows of the input matrix are row distributed to a fan of increment nodes. The increment nodes increment each value in the row and the flowgraph then does a column collect, resulting in a transposed matrix. The transposed matrix is row distributed to a fan of power nodes. Finally, a parallel summing reduction is done to yield a single result vector. The code for this example flowgraph is provided in Appendix B. Portions of the code are used in examples in Chapter 2.

The hierarchical nature of Assign flowgraphs allow you to build applications in a sequence of incremental steps. The following list outlines the steps to build an Assign application.

1. Write node functions — The node functions are sequential C functions, instances of which correspond to nodes in the flowgraph. You can compile and test each function independently on your Sun workstation.
2. Generate and test subgraphs — Since Assign flowgraphs are hierarchical, graphs can be composed of smaller subgraphs. You can generate and test these subgraphs on your Sun workstation using the node functions developed in step 1. To compile for your Sun workstation, use the -host option on the Assign command line. Chapter 3 describes the command options. Flowgraphs are generated by writing and running a C program called a graph generator. Chapter 2 describes how to write a graph generator.
3. Combine and test subgraphs — Once each subgraph has been tested and is working properly, you can combine the subgraphs to form the final flowgraph. You can compile and test the final graph on your Sun workstation, using the -host option on the Assign command line.
4. Run the final graph — When you have tested the final subgraph, you can recompile and run the flowgraph on your iWarp system.

Chapter 2 provides a detailed description of Assign flowgraphs and describes how to write a graph generator.

XAssign: The Assign Graphical User Interface

The conventional way to generate an Assign flowgraph is to write a C program that calls interface routines to generate the flowgraph. Although this allows for maximum flexibility in generating a flowgraph, it requires you to write and compile a new program to generate each new flowgraph.

XAssign, the Graphical User Interface for the Assign signal processing compiler, provides an alternative to Assign's C programming interface. It is designed to provide a more intuitive interface that frees you from the details of programming and command line options and lets you focus on the flowgraph programming model itself.

XAssign's key features are the following:

- intuitive visual interface
- creation and modification of flowgraphs
- support for compilation and execution of flowgraphs
- extensive help facility
- reusable library of standard node definitions

For a complete description of how to use XAssign, refer to Appendix C.