3 *Compiling and Executing*

The Assign parallel program generator is an individual compilation tool included within the Assign driver. This driver includes the following passes, which are invoked automatically when you use Assign:

- build (b): This pass compiles the flowgraph generator using the Sun C compiler. The resulting executable (*name-gen**) is then executed to make the intermediate representation (*name.graph*).
- assign0 (a): The assign0 pass reads the intermediate representation and creates the input to PCS. This is a cell program (*name-cell.c*) and an array descriptor (*name-array.c*). If you only want to execute on the Sun system (for development purposes), only a C source program is generated (also called *name-cell.c*). No array descriptor file is generated for host execution. There may also be a *name.sum* file generated if internal debugging is used. This is not used in a future pass, but it may contain some useful summary information.
- host (h): If you choose to execute on the Sun, you invoke the Sun C compiler (cc) to compile the cell program and link it together with primitive object files into an executable file (*name**). You can execute this file to simulate the flowgraph on the Sun. The following two passes are not done if you choose this pass. The h pass is not done if you choose the c and l passes. The c and l passes are the default.
- combine (c): This pass uses the iWarp C compiler (iwcc) to compile the cell program into a .o file. The array combiner (iwcb) is then used to compile the array descriptor program, and to create an array module (*name-array*) with the cell program and primitive .o files included.
- load (l): This is the final pass. This pass takes the array module, links it with appropriate libraries, and generates an executable script (*go**). You then need only type ‘‘go’’ to execute the flowgraph on the hardware array. You can also use the -X command option to automatically execute the flowgraph on the array. The command options are described later in this chapter. Other intermediate files left by iwload are: name-array-init.{c,o}, name-array-map.{c,o}, name-array-make, and name-array_x.x.x* (the executable image). The iwload manual page describes these intermediate files and their contents.

The intermediate files created during the passes are (by default) placed into the directory specified by the TMPDIR environment variable and erased when they are no longer needed. You can use command options to retain these intermediate files and stop after certain passes. The command options are described later in this chapter.

Figure 3-1 provides an illustration of the Assign passes and lists the files produced and used in each pass. The following information may be helpful in understanding the figure:

- name-gen.c is the flowgraph descriptor
- cc is the Sun C compiler
- iwcc is the iWarp C compiler
- iwcb is the iWarp array combiner
- iwamld is the PCS loader

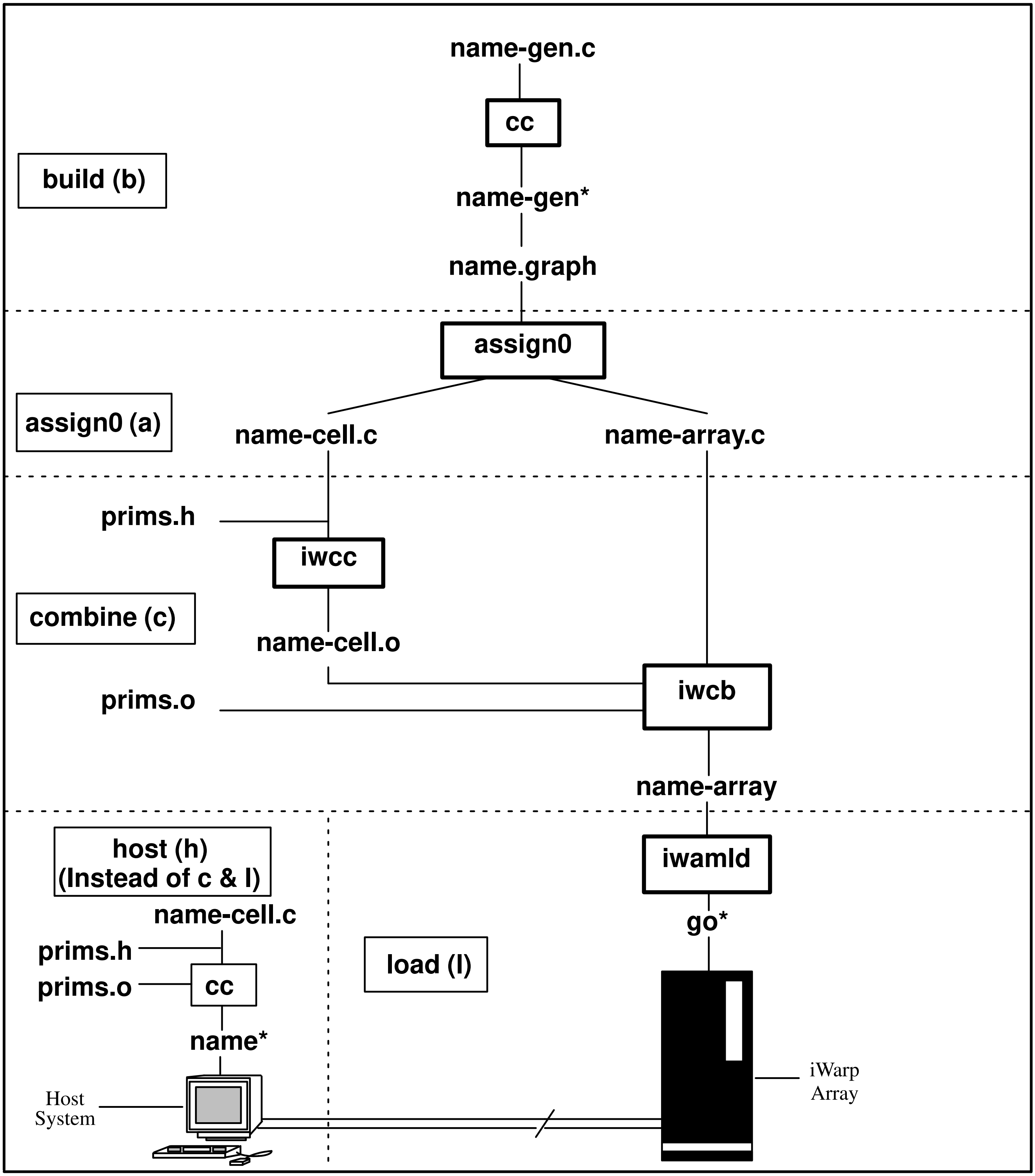

Figure 3-1: Assign Driver Compilation and Execution Passes

Invoking Assign

To invoke Assign, enter the assign command as follows:

```
assign [option ...] filename
```

filename is the prefix in the *name-gen.c* file.

option can be any of the options described in the following section. If no options are specified, the default output is an executable file and a script to load and run the executable on the specified array. No other files will be left in the local or TMPDIR directories.

Assign Command Options

-a	Stop after Assign creates the input files to PCS. This is the same as -Ga.
-c	Stop after iwcb creates the array module. This is the same as -Gc.
-col *num*	Set the number of columns in the array module generated by iwcb. This option (with -row) also informs the assign pass how many cells are available for compilation.
-host	Generate code for execution on the host system. The default is code generation for execution on the iWarp system.
-hetero	Generate heterogeneous code for execution on individual iWarp cells. The default is homogeneous code (same program executed on all cells).
-i *iters*	Invoke the source node(s) of the flowgraph *iters* times. The default is one iteration.
-o *filename*	Name the executable *filename* instead of ‘‘filename_array_0.0.0’’. This also updates the file name in the script file. This is the same as -Wl,-o *filename*.
-p	Turn on profiling.
-row *num*	Set the number of rows in the array module generated by iwcb. This option (with -col) also informs the assign pass how many cells are available for compilation.
-spool	Use spooling communication instead of systolic. By default, Assign uses systolic communication when generating the array descriptor file.
-D*name*[=*def*]	Define *name* for the build pass (of the filename-gen.c file’s compilation) as if by a #define directive. If no =*def* is given, *name* is defined as 1. The -D option has lower precedence than the -U option. That is, if the same name is used in both the -D and -U options, the name is undefined regardless of the order of the options. This is the same as using -Wb,-D*name*[=*def*].
-E *arg*[,*arg* ...]	Pass one or more arguments specified by *arg* to the build pass for execution of the filename-gen executable.

-G*pass* Stop the Assign driver after pass *pass*. This causes any intermediate files created in this pass to be left in the local directory (all other intermediate files are put into the temp directory and erased unless -T is used). *pass* can be any of the following:

b	build
a	assign0
h	host
c	combiner
l	load

-H *host_name* Specify the iWarp array attached to *host_name* as the target array. By default, the target iWarp array is assumed to be the value of the environment variable IWARP_HOSTNAME. If this variable is not set, the target array is assumed to be on the current machine.

-O Optimize compilation of the cell program.

-P *paths* Specify *paths* as the maximum number of inbound and outbound pathways for each cell. The default is 2.

-S *sib_num* Specify SIB number *sib_num* on the target iWarp array. By default, the SIB is the value of the environment variable IWARP_SIBNUM. If this variable is not set, the SIB is assumed to be number zero.

-T Do not delete intermediate files. Leave them in the local directory. By default, intermediate files are given a unique name, placed in the directory specified by the TMPDIR environment variable, and erased when no longer needed. When you do not use -T, iWarp executables are left in the temporary directory. Currently, this directory must be on the host machine.

-U*name* Remove any initial definition of *name*, where *name* is a reserved symbol that is predefined by the preprocessor.

-V[V] Display the version of the Assign compiler and all invoked passes. With -V, compilation proceeds as specified. With -VV, no compilation is performed.

-W*pass,arg*[,*arg* ...] Hand off the command arguments specified by *arg* to pass *pass*, where *pass* is one of the following Assign passes:

b	build
a	assign0
h	host
c	combiner
l	load

-X Execute the flowgraph on the target iWarp array after generating the executable image and the execution script. The executable image and execution script are treated as intermediate files. This is the same as -Wl,-e.

-Y*pass,dir* Get pass *pass* from directory *dir* instead of from the standard area.

-Z*pass*	Start the driver in pass *pass*. You must use the -T option with -Z, because the necessary inputs to pass *pass* are assumed to be in the current working directory.
-#[#[#[#]]]	Print debugging information at the specified level.

-#	Print lines per minute information upon driver completion.
-##	Print the invocation of each pass and its options. Inherits -#.
-###	Print execution time of each individual pass. Inherits -##.
-####	Print the invocation of each pass and its options, but does not do any execution..

Host vs. iWarp Execution

You have the option of executing the flowgraph on the Sun host. When building a large flowgraph, you will want to build it hierarchically. You can compile and execute each of these hierarchical sub-graphs on the Sun for verification. You can then combine each of these sub-graphs into one graph generator with little alteration to the subgraph files. This main graph can also be tested on the Sun host.

Once the graph is functioning properly, you can remove the -host option and recompile the entire graph. The graph can then be executed with real data on the iWarp array, where it will run much faster than on the Sun because the true parallelism of the iWarp system can be used.

Array Modules

The purpose behind using the -c (or -Gc) command option is to create an array module that can be combined with other array modules. This means a flowgraph can be combined with the output of an Adapt program or any user-written array program. This also provides another way to combine two or more flowgraphs, whether they are the same or different.

In order to combine array modules, you should be familiar with PCS. You must write a new array descriptor which instantiates and connects the array modules via external ports. If any cells are instantiated, they will require cell programs that control the flow of information. For more information about using PCS, refer to the *iWarp Programmer's Guide*.

Compilation and Execution Example

The following example completes the abi example described in Chapter 2 by showing the steps needed to successfully execute the flowgraph on the iWarp array.

Primitive Compilation

When the target machine is the iWarp array, you must compile the primitives into object code using the iWarp C compiler. In this example, the -O option tells the compiler to optimize the code. The -c option tells the compiler to create an object file with the same name as the source file, except that it ends in .o instead of .c. Refer to the *iWarp Programmer's Guide* for descriptions of other iwcc options that may be useful for your compilation.

NOTE

The calls to add_obj_file() in the graph descriptor should specify the path to where .o files can be found.

Use the iwcc command to compile the files as follows:

```
% iwcc –O –c cxrd.c cxincr.c cxcc.c power.c prred.c –I/iwarp/include
```

Invoking the Driver and Executing

Use the assign command to invoke the Assign driver and create an execution script.

```
% assign abi –H <host_name> –S <sib number> –T –##
```

The driver will look for the graph descriptor file called abi-gen.c in the current directory. The -## option shows all intermediate steps and the options passed to those steps. In this example, the -T option is used to retain intermediate files. If you would not like the intermediate files left in the current directory, remove the -T option and copy the primitive .o files to your $TMPDIR directory.

The final result is an execution script called *go*. To execute the flowgraph simply type go. The output will be left in a stdout file, unless you use other primitive output routines.

```
% go
```

Host Compilation.

The following example uses the -host option to create an executable file you can run on the host system.

```
% cc –O –c cxrd.c cxincr.c cxcc.c power.c prred.c –I/host/include
```

```
% assign abi –host –T –##
```

This command has the same effect as the previous example, except the final executable will be called abi, and should be executed on the host system. The output will go to stdout, unless you use other primitive output routines.