2.1 Starting PeaCE

PeaCE has a server-client architecture. The server runs the PeaCE daemon process and the PeaCE kernel (Ptcl). To get started, you should first run a PeaCE daemon on the service. Next, run a PeaCE client in any client machine. Since the PeaCE daemon invokes the PeaCE kernel, there is no need to run the kernel explicitly. The current version of the server uses a file for the block library.

2.1.1 Running PeaCE daemon

Environment variable DISPLAY setting

The environment variable DISPLAY is needed to execute PeaCE kernel. In your home directory of the server machine, type:

```
setenv DISPLAY          or          setenv DISPLAY localhost:0
```

Running PeaCE daemon

PeaCE daemon uses a designated port to communicate with PeaCE client. To run PeaCE daemon in your server machine with port number 5555, type:

```
peacedaemon 5555&
```

2.1.2 Running PeaCE client (HAE)

HAE is the graphical user interface of PeaCE. The client machine on which you want to run a PeaCE client must have a java interpreter. To pop-up a “Start PeaCE” dialog in your client machine, change current directory to where HAE is installed, and type:

```
gohae
```

Then the following dialog box will appear (Figure 2-1):
To start PeaCE:

1. Ensure that the **Main** tab is selected. Type the server machine IP in the Server IP field. If the server machine is same as the client machine, type `localhost`. Type the user ID in the User ID field. Type the password in the Password field.

2. Click the **Option** tab. The Option panel is displayed (Figure 2-2). Type PeaCE daemon port number, e.g., `5555`, in the Port Number field. If you do not execute an external X server such as eXceed or Xmanager, check the XServer checkbox.
3. Click the **Main** tab again. Click the **Start PeaCE**. Then, the PeaCE window will be displayed as shown in Figure 2-3.

![PeaCE window](image)

**Figure 2-3 PeaCE window**

If you cannot see this window, please make sure that your peaceDaemon is running on the server machine with the same port number as specified in the “Port Number” field in the Option panel.

### 2.2 Running a demo application

This section describes how to run an existing demo application. Please ignore the computation model of the block diagram for now.

#### 2.2.1 Opening a demo application

To open a demo application:

1. Select **Open** from the **File** menu. An Open dialog will be displayed.
2. Navigate to the CGC\Basic directory in the `{install_directory}\schematic\Peace\Demo` directory and click **butterfly** to select it (Figure 2-4).
3. Click Open. PeaCE displays the demo schematic in the main window (Figure 2-5).

![Figure 2-5 Butterfly demo schematic]

2.2.2 Running the demo application

To run the demo application:

1. Select Run It! from the Run menu. A Run butterfly dialog is displayed (Figure 2-6).

![Figure 2-6 Run butterfly dialog]
2. Click **Go**. A generated source code (butterfly.c) will be displayed (Figure 2-7). The code will be compiled and run in the host machine. An xgraph is displayed as a result (Figure 2-8). This demo illustrates how functional simulation is performed in PeaCE.

![Figure 2-7 Generated C code (butterfly.c)](image1)

![Figure 2-8 Xgraph output of butterfly demo](image2)
2.3 Creating a simple application

This section describes how to create a simple application in PeaCE. Let’s make the following function:

\[ y = \cos(0.2 \times x) + 0.4 \]

2.3.1 Creating a new schematic

First, you have to create a new schematic to draw a block diagram:

1. Select New from the File menu. A New dialog will be displayed.
2. Navigate to the directory where you want to save the schematic and enter a schematic name, for example cosTest (Figure 2-9).
3. Click Open. PeaCE will create a new schematic and open it in the main window (Figure 2-10).
2.3.2 Configuring domain and target

After an empty schematic window is open, select the domain and the target of the design. For a computation task, you will draw a dataflow graph and generate a C code from the specification. The domain would be “CGC (Code Generation in C). And the target is the default target that indicates functional simulation in the host machine.

To select the domain and the target, click the Design tab. Then you will get a browser window as shown in Figure 2-11. It contains a domain drop-down list and a target drop-down list and target states. See the next chapter for more information on domain and target and target states. Make sure that CGC is selected in domain drop-down list and default-CGC is selected in target drop-down list. Leave all the target states as they are.

Figure 2-11 Design browser window of demo

2.3.3 Using blocks in SPDF library

The dataflow model used in PeaCE is called SPDF (Synchronous Piggybacked DataFlow). Chapter 4 is devoted to the SPDF domain. Since the “cosTest” example will be composed of SPDF blocks, first search for the library blocks to be reused. If you cannot find a library block, you have to create one. But in this example you are able to locate all the library blocks you will use. Now let’s figure out what blocks you need for the equation:

\[ y = \cos (0.2 \times x) + 0.4. \]

First you need a block to generate the argument of \( \cos \) function. The Ramp block is used to generate a sequence of argument values. Constant value 0.4 is provided by the Const block. And you need two operator blocks: Add and Cos.

To instantiate a Ramp block, do the following steps:

1. Click the Lib tab. The library browser will be displayed (Figure 2-12).
2. Navigate to the *Src* directory in the *SPDF* directory and click *Ramp.ilink* to select it.

3. Place the *Ramp* block in the schematic window (Figure 2-13) by moving your mouse to the desired location.

Place following blocks similarly (Figure 2-14).

- *SPDF\Src\Const.ilink*
- *SPDF\Math\Cos.ilink*
- *SPDF\Math\Add.ilink*
- *SPDF\Sink\Xgraph.ilink*
2.3.4 Editing block states

Now you should edit the block states if needed. In this demo, you have to edit the states of the Ramp block of which the step state is 0.2 and the value state is 0. And you edit the “level” state of the Const block to 0.4.

To edit the block states, do the following:

1. Ensure that the Design tab is selected.
2. Click the Ramp block to select it. The block states will be displayed in the Design browser window.
3. Type 0.2 in the value field of the step state (Figure 2-15).

![Figure 2-15 Ramp block states](image)

4. Click the Const block to select it and type 0.4 in the value field of the level state (Figure 2-16).

![Figure 2-16 Const block states](image)

2.3.5 Connecting blocks
Now you connect the blocks. To connect two blocks, Ramp and Const, do the following:
1. Move the mouse pointer to the output port of the Ramp block. You can see a green circle and the port name.
2. Click the circle to start drawing a line from the port.
3. Move the mouse pointer to draw a line as you wish. If you want to make a turn, click at the turning point.
4. Click the circle of the input port of the Cos block to complete the line.

Complete the other lines similarly (Figure 2-17).
- From the output port of the Cos block to the input port of the Add block.
- From the output port of the Const block to the input port of the Add block.
- From the output port of the Add block to the input port of the Xgraph block.

![Figure 2-17 cosTest application](image)

2.3.6 Running the application

To run the application:
1. Select **Run It!** from the **Run** menu. A Run cosTest dialog is displayed.
2. Type 100 in the When to stop field (Figure 2-18).

![Figure 2-18 Run cosTest dialog](image)
3. Click Go. A generated source code (cosTest.c) will be displayed (Figure 2-19), compiled and run in the host machine. An xgraph is displayed as a result (Figure 2-20).

PeaCE has many built-in demos in schematic\Peace\demo directory. If you want to get an idea how to use PeaCE, we recommend you to run some demos. Some demos give you exciting outputs. Drawing a block diagram is basically what you have to do in PeaCE if all the blocks you need are already existent in the block library. Since PeaCE is not a commercial tool, not many library blocks are provided. So you should know how to create your own blocks.