

Cadence® Virtual Component Co-Design

Tutorial 2: Hierarchical Modeling

Product Version 2.2
December 2002

© 2001-2002 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

Introduction 7

Start the VCC Software 8

2

Behavior Model 9

Create New Data Types 11

Create the Server Behavior Diagram 14

Create a New Behavior Diagram 15

Verify Configuration Rules 16

Create the server_std and server_db Blocks 19

Add Ports to the Blocks 22

Add an Input Port 27

Connect the Ports 29

Add Memory References 30

Generate the Symbol 32

Create the Controller Behavior Diagram 33

Create a New Behavior Diagram 33

Create the Timer and sinComb Blocks 35

Add Ports to the Blocks 38

Add an Output Port 44

Add a Behavior Timer 45

Connect the Ports 46

Add Parameters and Memory References 47

Generate the Symbol 49

Create a Top-Level Behavior Diagram 50

Create a New Behavior Diagram 50

Instantiate the Blocks 52

Specify Block Parameters 55

Wire the Blocks 61

Specify Top-Level Parameters 62

Tutorial 2: VCC Hierarchical Modeling

<u>Bind the Parameters</u>	63
<u>Associate the Memory References with the Behavior Memory</u>	64
<u>Create Functional Behavior Models</u>	66
<u>Create a Functional Model for the rtc_testbench Block</u>	66
<u>Create a Functional Model for the server_db Block</u>	76
<u>Create a Functional Model for the server_std Block</u>	83
<u>Create a Functional Model for the sinComb Block</u>	94
<u>Create a Functional Model for the Timer Block</u>	105
<u>Run a Functional Simulation</u>	111
<u>Create an Analysis Session</u>	112
<u>Run the Simulation</u>	114
<u>What if Your Model Won't Simulate?</u>	116
<u>How is the Behavior Memory Connected?</u>	117

3

<u>Architecture Model</u>	119
<u>Create a New Architecture Diagram</u>	120
<u>Place the Architecture Blocks</u>	122
<u>Specify Block Properties</u>	123
<u>Add Buses to the Diagram</u>	132
<u>Add a Data Bus</u>	132
<u>Add an Interrupt Bus</u>	135
<u>Wire the Architecture Blocks</u>	140
<u>Organize the Blocks</u>	141
<u>Wire the Blocks</u>	141
<u>Save the Diagram</u>	145
<u>Generate the Netlist</u>	145
<u>What if Your Netlist Won't Generate?</u>	147

4

<u>Mapping Diagram</u>	149
<u>Specify Performance Views</u>	150
<u>Server Performance Views</u>	150
<u>Controller Performance Views</u>	154

Tutorial 2: VCC Hierarchical Modeling

<u>Create the Mapping Diagram</u>	158
<u>Instantiate Behavior and Architecture</u>	160
<u>Map the Behavior to the Architecture</u>	160
<u>Set Communication Patterns</u>	171
<u>Bind Memory Segments to RAM</u>	175
<u>Run a Simulation</u>	177
<u>What if Your Model Won't Simulate?</u>	181

5

<u>Analysis</u>	183
<u>Simulate using Probes</u>	184
<u>Begin an Analysis Session</u>	184
<u>Add a Generic Probe to the controller Block</u>	185
<u>Run a Simulation</u>	187
<u>Analyze the Simulation</u>	189
<u>Simulate using Sweep Parameters</u>	195
<u>Begin an Analysis Session</u>	195
<u>Create the samplingTime Parameter</u>	196
<u>Bind the samplingTime Parameter</u>	197
<u>Set the Sweep Parameters</u>	197
<u>Run a Simulation</u>	200
<u>Simulate using a Single-CPU Architecture</u>	202
<u>Create a Single-CPU Mapping Diagram</u>	202
<u>Specify Task Priorities</u>	205
<u>Bind the New Memory Segments to RAM</u>	208
<u>Begin an Analysis Session</u>	210
<u>Run a Simulation</u>	211

Tutorial 2: VCC Hierarchical Modeling

Introduction

This tutorial is the second in a series. Before you begin this tutorial, you should have successfully completed *VCC Tutorial1: Architecture Services*.

This tutorial demonstrates the hierarchical nature of VCC models. In this tutorial, you will create hierarchical relationships between blocks, by instantiating lower-level blocks in higher-level diagrams, and between parameters, by referencing lower-level parameters to higher-level parameters.

This tutorial walks you through the design of behavior and architecture models for a real-time controller.

- The behavior model for this real-time controller contains two main subsystems: the server and the controller.
 - The server is composed of a finite state machine that receives external commands and changes state among *OFF*, *INIT*, and *ONLINE*. Another component passes control parameters to the controller through a shared memory.
 - The controller implements the periodic execution of a control algorithm that is a simple sine wave convolution. The output of the controller could represent, for example, the trajectory of a linear electromechanical actuator.
- The architecture model for this real-time controller contains two CPUs; one CPU hosts the server application and the second CPU hosts the controller application. Shared memory communication is realized by a simple memory over a data bus.



To provide optimum screen space for the Cadence® Virtual Component Co-Design (VCC) tool, it is best to use the tutorial in printed form, so print the PDF version of this tutorial before you begin any of the exercises.

Note: The library you will create in this tutorial is named *Tutorial2*; however, some screen graphics might show the generic name *Tutorial*. Please use only the name *Tutorial2* as specified in the instructions.

Tutorial 2: VCC Hierarchical Modeling

Introduction

Start the VCC Software

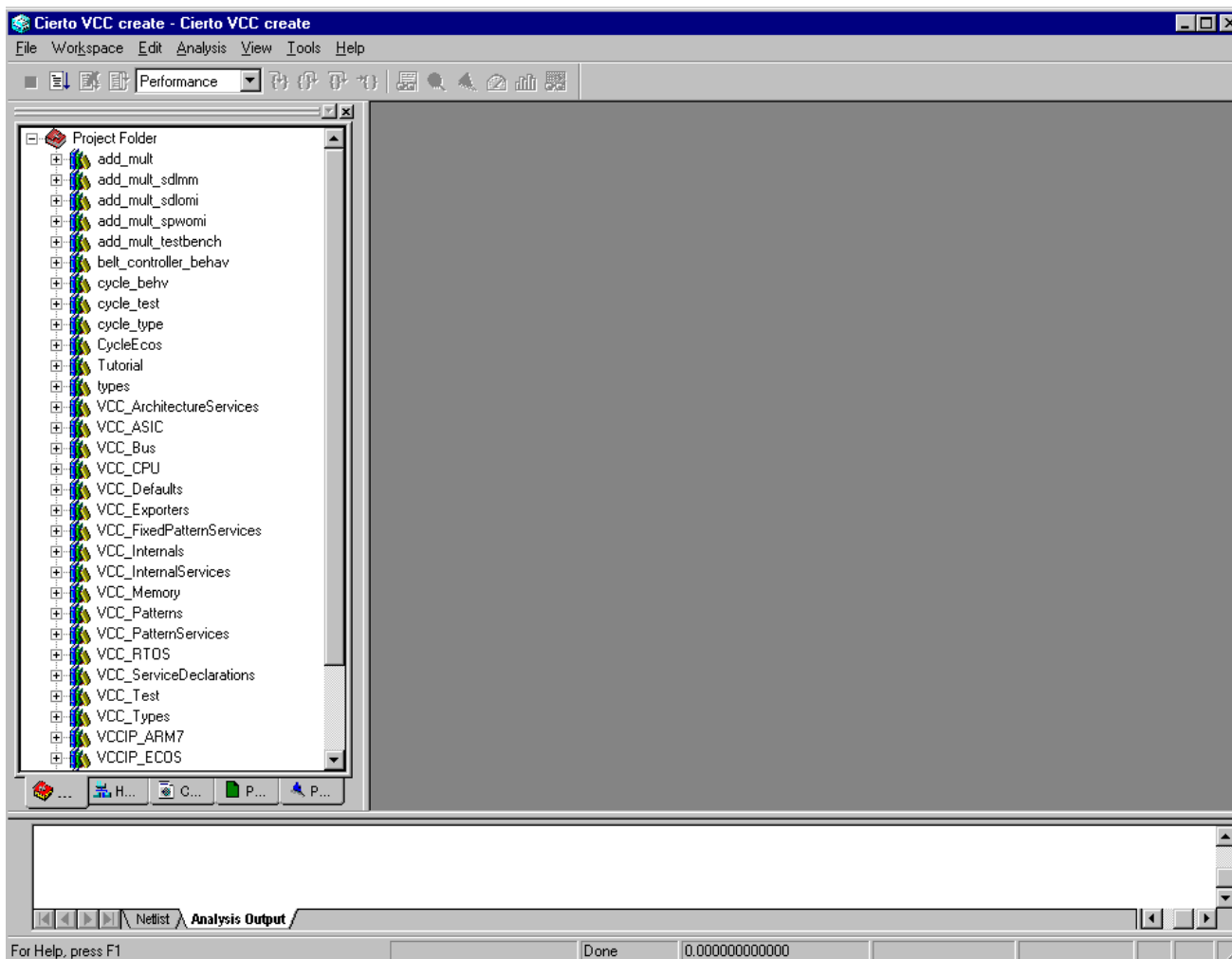
1. To start the VCC Create software:

- ☐ Windows: Choose *Start > Programs > Cadence VCC > Create*.
- ☐ Solaris or HP-UX: Type *create&*.

2. If VCC Create opens with all the toolbars displayed, use the *View > Toolbars* menu command to close them:

- ☐ Choose *View > Toolbars > <Toolbar name>*.
- ☐ One at a time, choose each of the toolbars with a check in front of it.

You won't need the toolbars for this tutorial, and closing the toolbars gives you more screen space for your designs.



Behavior Model

The behavior model for this real-time controller design contains two main subsystems: the server and the controller.

- The server is composed of a finite state machine that receives external commands and changes state among *OFF*, *INIT*, and *ONLINE*. Another component passes control parameters to the controller through a shared memory.
- The controller implements the periodic execution of a control algorithm that is a simple sine wave convolution. The output of the controller could represent, for example, the trajectory of a linear electromechanical actuator.

This model implements the following behavior:

- While the server is in the *OFF* state, the controller output is zero.
- While the server is in the *INIT* state, the controller output is equal to the offset parameter value.
- While the server is in the *ONLINE* state, the controller output is the offset plus a sine waves convolution.

To create this real-time controller behavior model, you will

- [Create New Data Types](#) on page 11
- [Create the Server Behavior Diagram](#) on page 14
- [Create the Controller Behavior Diagram](#) on page 33
- [Create a Top-Level Behavior Diagram](#) on page 50
- [Create Functional Behavior Models](#) on page 66
- [Run a Functional Simulation](#) on page 111

If you have problems with your behavior model that you can't resolve, instructions for using a golden copy from the database are in [What if Your Model Won't Simulate?](#) on page 116.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

This tutorial uses a behavior memory and memory references. If the concept of memory references is new to you, you might want to read [How is the Behavior Memory Connected?](#) on page 117.

In the behavior models section of the tutorial, you will create custom behavior blocks for the following functions:

- *rtc_testbench*

A C++ model that simulates the commands coming from external word and trigger co-design finite state machine (CFSM) state transitions.

- *server_std*

A state transition diagram (STD) that implements the server finite state machine.

- *server_db*

A Whitebox C model that passes parameters.

- *sinComb*

A Whitebox C model that implements the sine waves convolution algorithm.

- *Timer*

A C++ model whose outputs trigger the *sinComb* execution.

You will then instantiate these blocks into higher-level blocks to create the real-time controller behavior model.

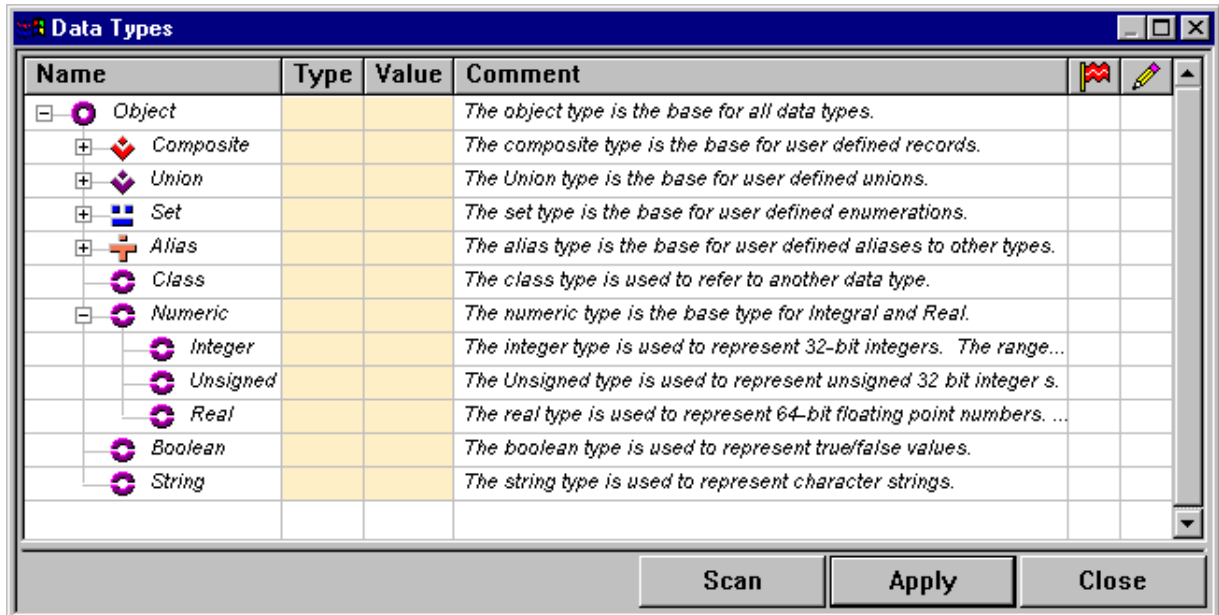
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Create New Data Types

This behavior model requires the data types: *OFF*, *INIT*, and *ONLINE*. These data types do not exist, so you must create them.

1. To open the Data Types browser, choose *Edit > Data Types* from the main menu.

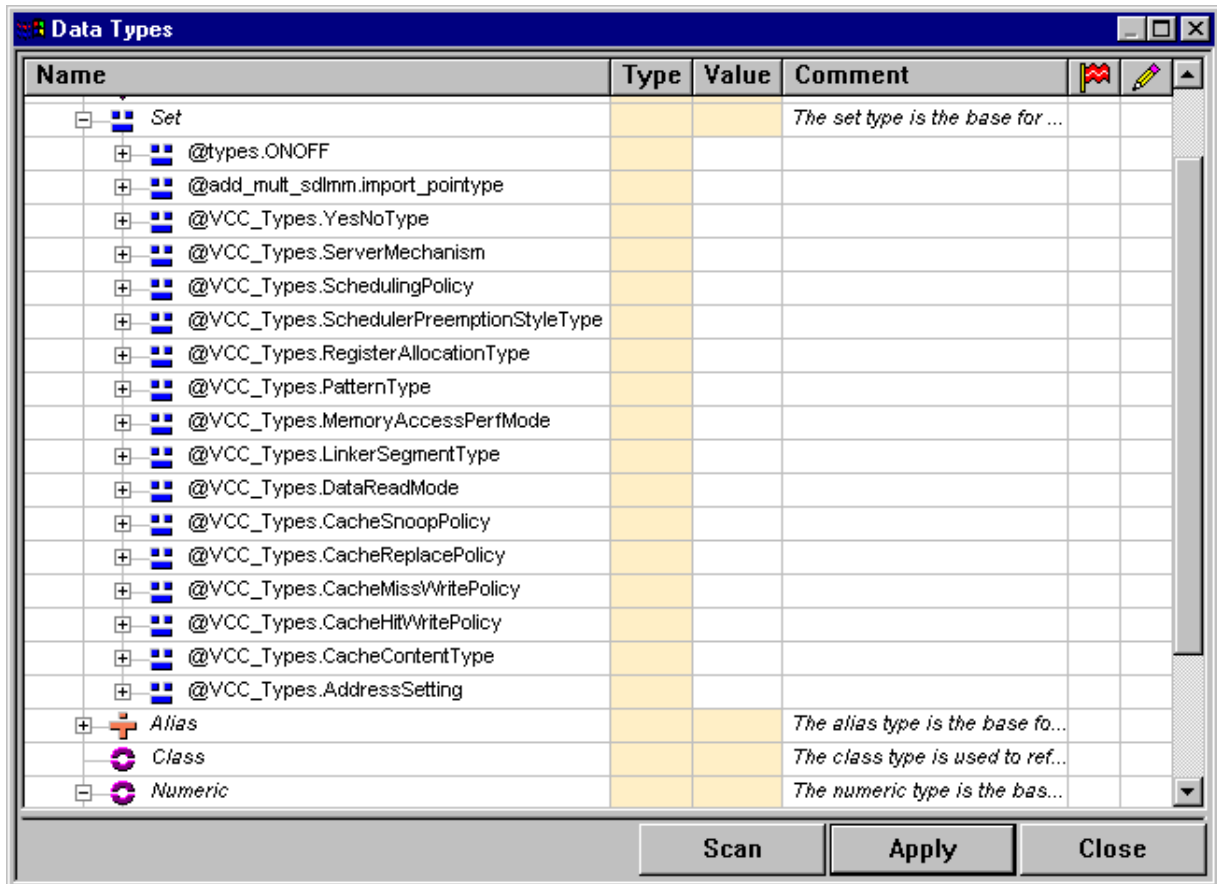


The new data type you are going to create is a *Set* type.

2. To expand the *Set* data types, click the plus sign to the left of the *Set* heading.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

Make sure you can see both the heading for the *Set* data types and the last *Set* data type in the list.



3. Right click on the *Set* heading and choose *New Set* from the pop-up menu.

VCC adds a new data type to the bottom of the list. This new data type name defaults to *@voicemail_behav.Set0*.

4. Change the new data type name to:

`@Tutorial2.CommandType`

Note: Do not press *Enter* after you change the names of the data type or the members. Simply move the mouse as the instructions indicate. If you press *Enter* at any time while in the Data Types editor, you trigger the *Apply* function and you will get the Save dialog. If this happens, click *Save* and continue working in the Data Types editor.

Tutorial 2: VCC Hierarchical Modeling

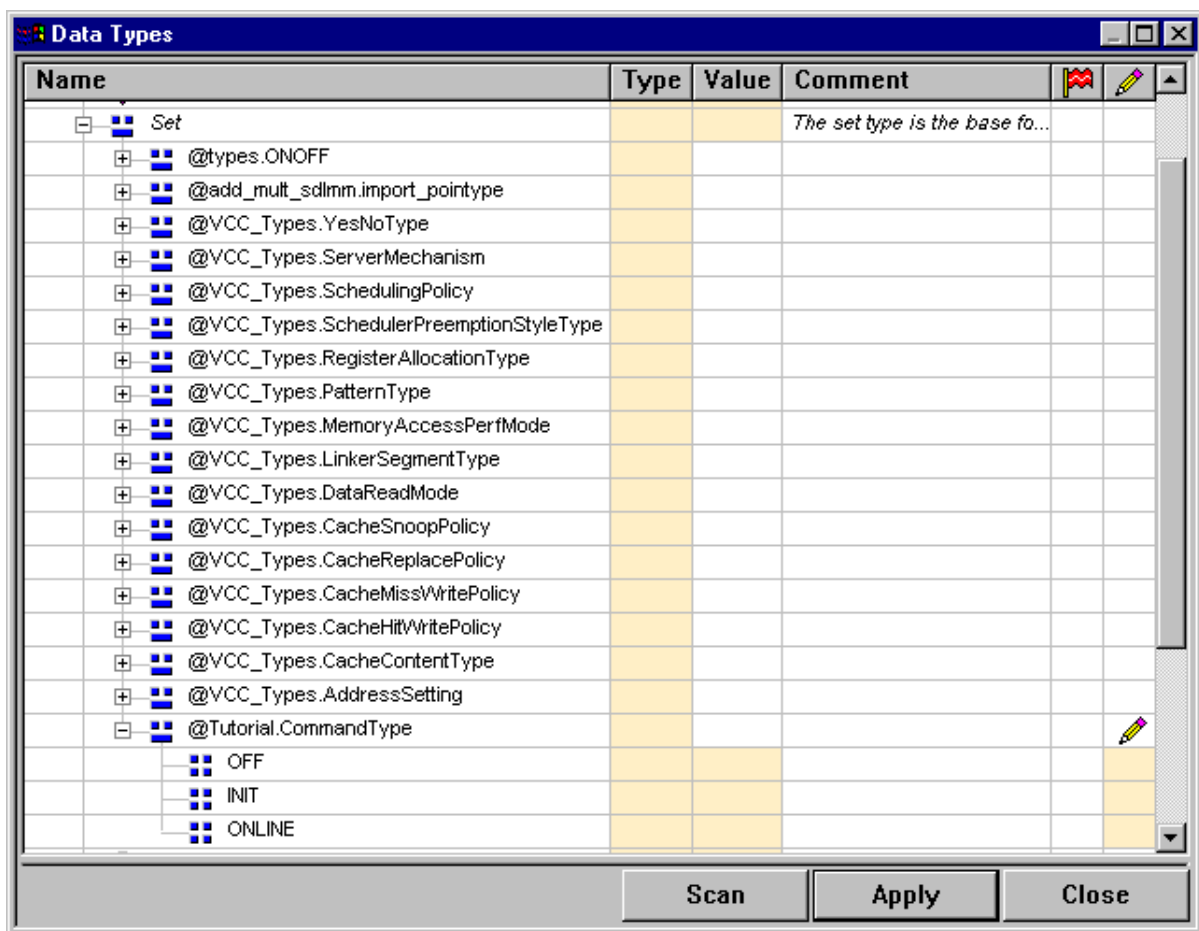
Behavior Model

5. To add a new *Set* member, right click on the *@Tutorial2.CommandType* name and choose *New Set Member* from the pop-up menu.

VCC adds a new member to the bottom of the *Set* data types list. This new member name defaults to *member0*.

6. Change the new member name to *OFF*.
7. Repeat steps 5 and 6, naming the second member *INIT* and the third member *ONLINE*.

The list of *Set* data types now looks like this:

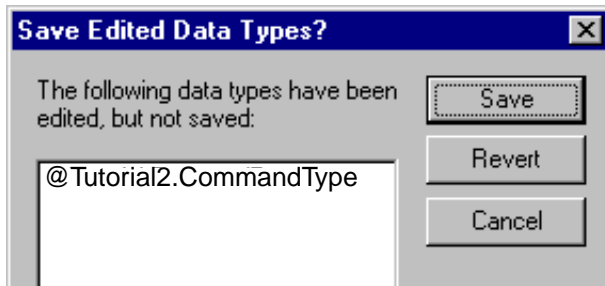


8. When you have finished entering the new data type and its three members, click *Close* in the Data Types dialog.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The Save Edited Data Types? dialog asks if you want to save the new data type.



9. Click **Save** in the Save Edited Data Types? dialog.

Note: If you press *Enter* at any time while in the Data Types editor, you trigger the *Apply* function and you will get this dialog.

Create the Server Behavior Diagram

The real-time controller requires a custom server. In this section, you will:

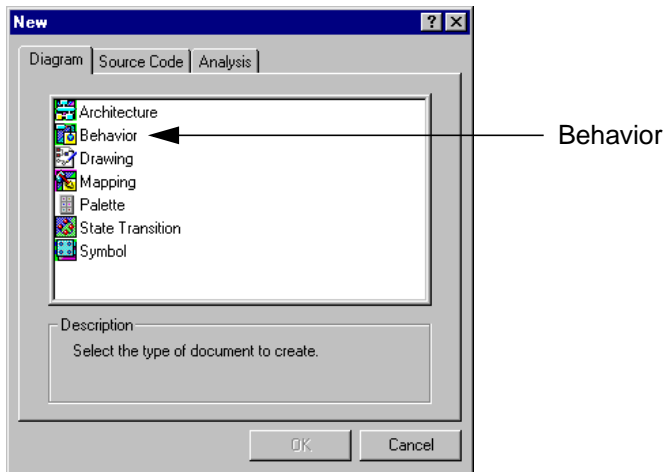
- [Create a New Behavior Diagram](#) on page 15
- [Verify Configuration Rules](#) on page 16
- [Create the server_std and server_db Blocks](#) on page 19
- [Add Ports to the Blocks](#) on page 22
- [Add an Input Port](#) on page 27
- [Connect the Ports](#) on page 29
- [Add Memory References](#) on page 30
- [Generate the Symbol](#) on page 32

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

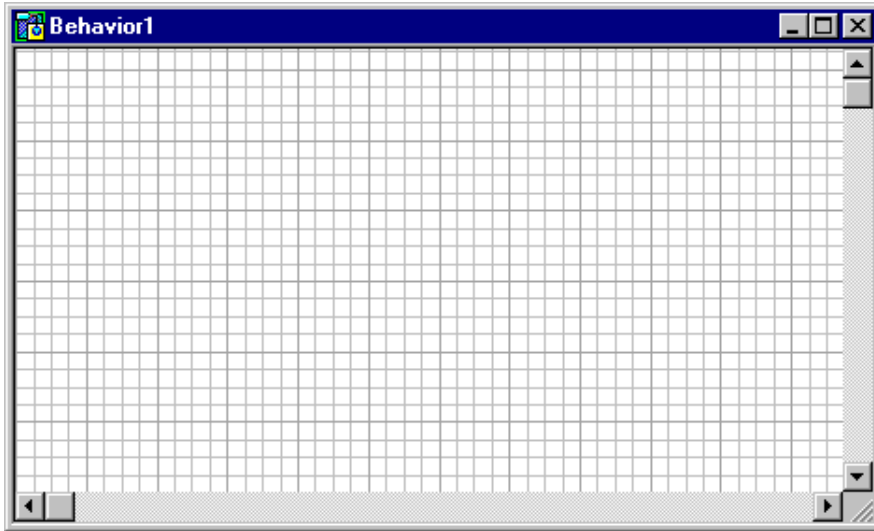
Create a New Behavior Diagram

1. To open a new behavior diagram, choose *File > New* from the main menu.



2. Double-click the *Behavior* icon in the New dialog.

A new window opens. The default name is *Behavior1*.

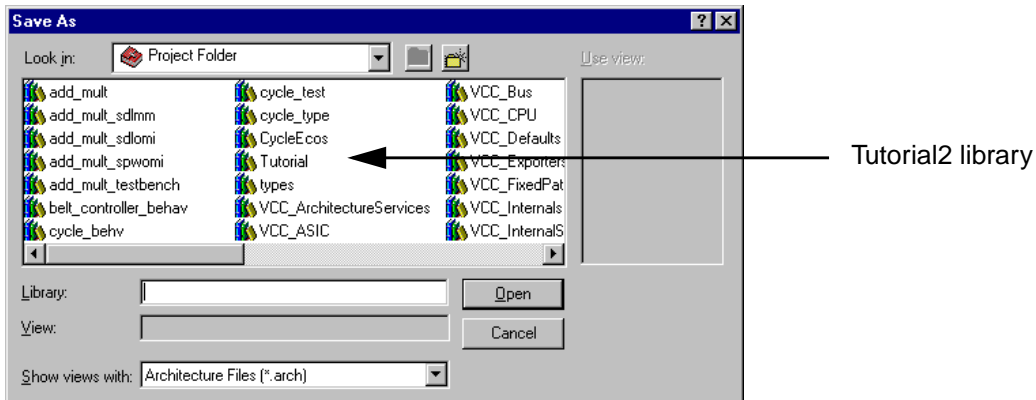


3. To save the behavior diagram, choose *File > Save* from the main menu.

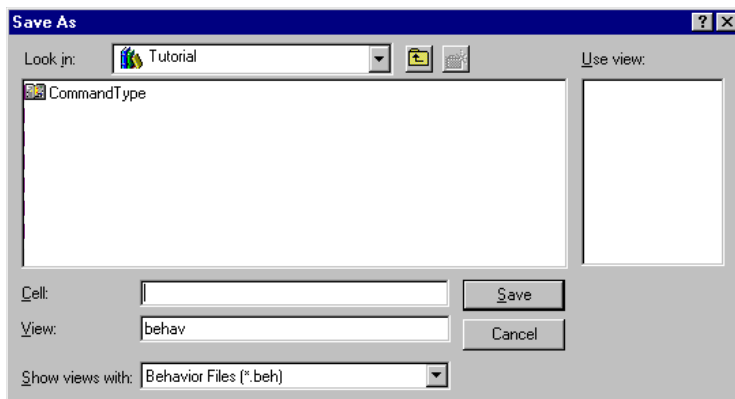
VCC prompts you to choose a library.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

4. Double-click the *Tutorial2* library icon.



VCC prompts you to specify a cell name. The View name defaults to *behav*.



5. Enter the Cell name *server* and click *Save* in the *Save As* dialog.

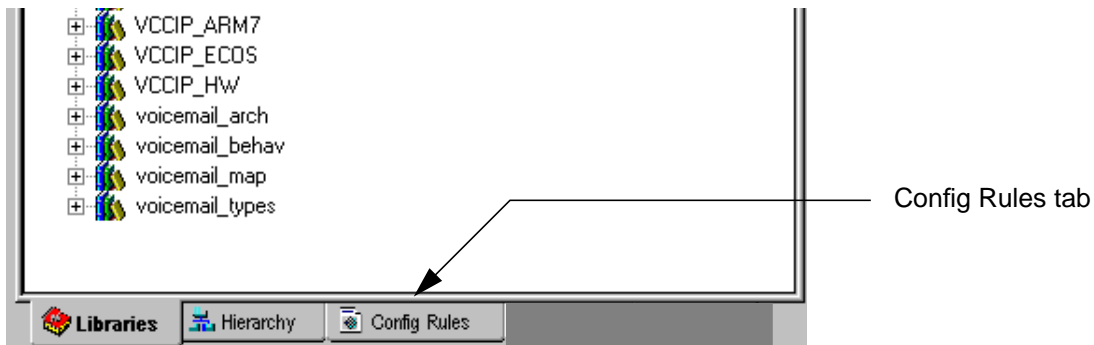
Note: You can save your cellviews as often as you like. If you are required to save your changes, a dialog asks if you want to “Save session changes?” Click *Yes* in the dialog. Any time the system asks if you want to save a session or a configuration, click *Yes*.

Verify Configuration Rules

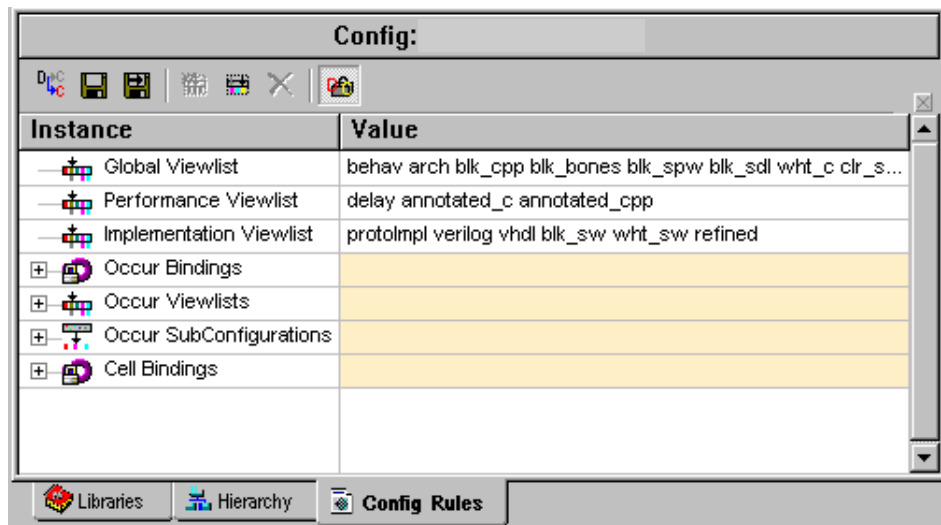
The Configuration Rules for a diagram tell the simulator which view to use. To verify that all possible views are specified in your configuration rules, take a look at the current values for the Global Viewlist.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

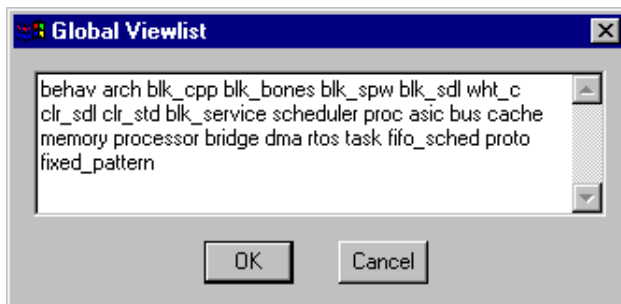
1. Click the *Config Rules* tab at the bottom of the workspace.



The Config Rules tab displays the views specified for your design.



2. Double-click the *Global Viewlist* heading.



3. Verify that your Global Viewlist contains the following views:

Tutorial 2: VCC Hierarchical Modeling Behavior Model

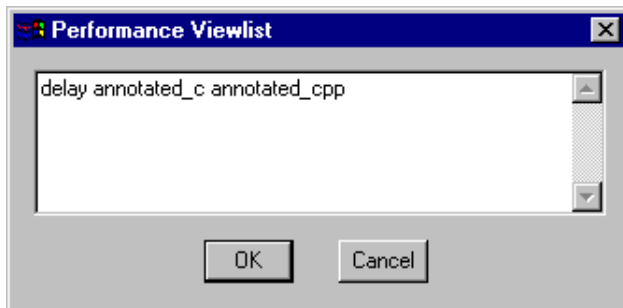
```
behav arch blk_cpp blk_bones blk_spw blk_sdl wht_c clr_sdl clr_std blk_service  
scheduler proc asic bus cache memory processor bridge  
dma rtos task fifo_sched proto fixed_pattern
```

This specifies the order in which to bind a model to a symbol; therefore, which model to simulate. If a symbol doesn't have a *behav* view, the system looks for an *arch* view, then *blk_cpp*, and so forth. Following this order, the system uses the first view it encounters for a simulation.

4. If any of these views are missing from the list, add the view name to the list.
5. When you are finished, click *OK* in the Global Viewlist dialog.

If a dialog prompts you to edit the default or edit a copy, click *Edit Default*.

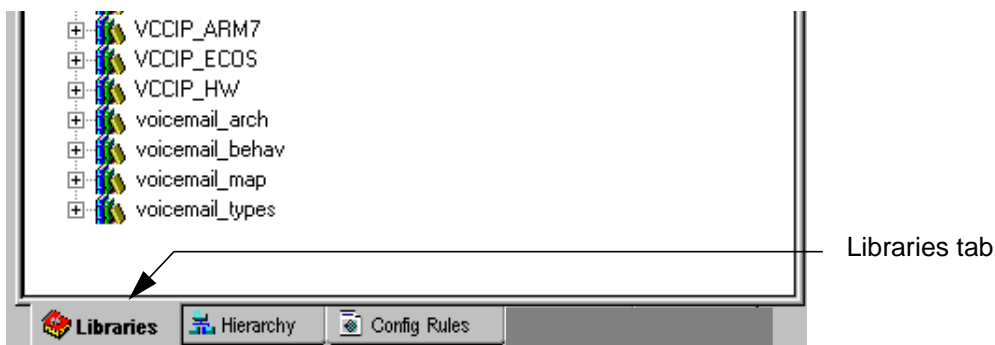
6. Double-click the *Performance Viewlist* heading.



7. Verify that your Performance Viewlist contains the following views:

```
delay annotated_c annotated_cpp
```

8. If any of these views are missing from the list, add the view name to the list.
9. When you are finished, click *OK* in the Performance Viewlist dialog.
10. To return to the Project Folder list, click the *Libraries* tab at the bottom of the workspace.



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Create the *server_std* and *server_db* Blocks

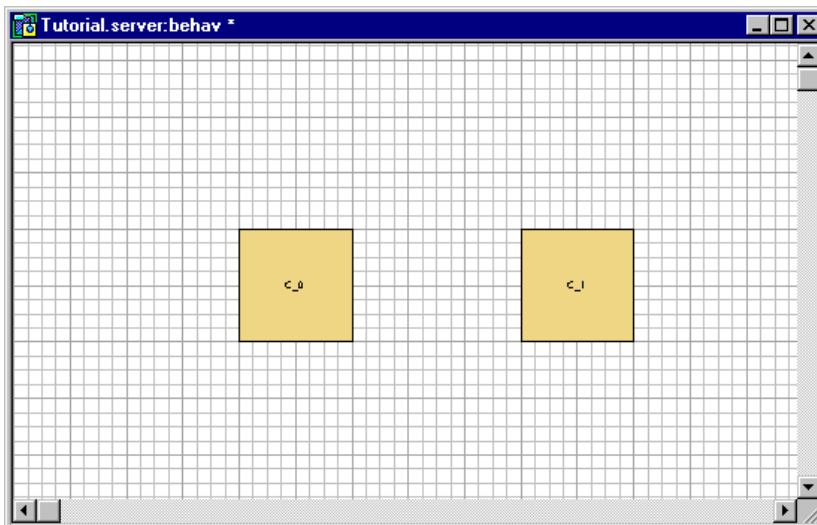
The *server_std* and *server_db* blocks do not exist, so you will create them.

- The *server_std* block is a state transition diagram that implements the finite state machine.
- The *server_db* is a Whitebox C model that passes parameters.

Draw the Blocks

1. To draw a new block, choose *Behavior > Block* from the main menu.
2. In the behavior diagram, click to place the upper-left corner of the block, then click to place the lower-right corner of the block.
3. Repeat steps 1 and 2 to draw a second block.

Your blocks should look something like this:



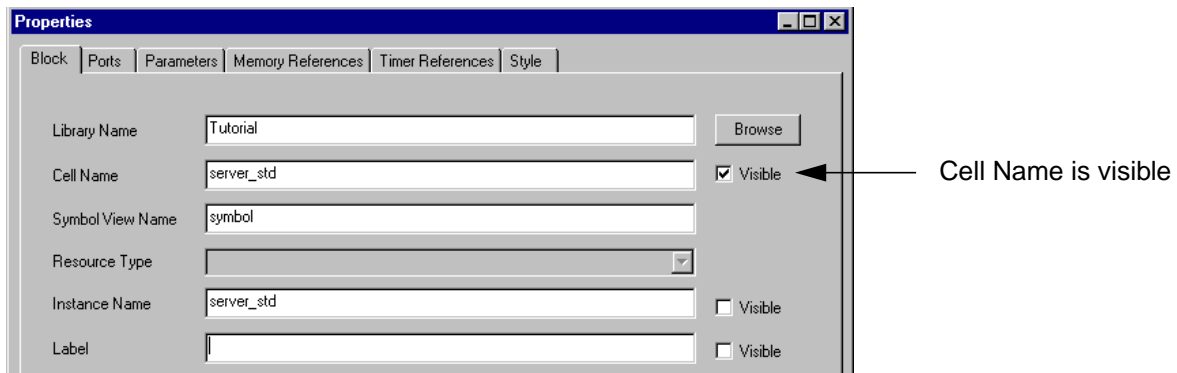
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Specify Block Properties

1. To specify the properties for the first block, right click on the leftmost block and choose *Properties* from the pop-up menu.

The Properties dialog opens.



For this block, you will enter the following values:

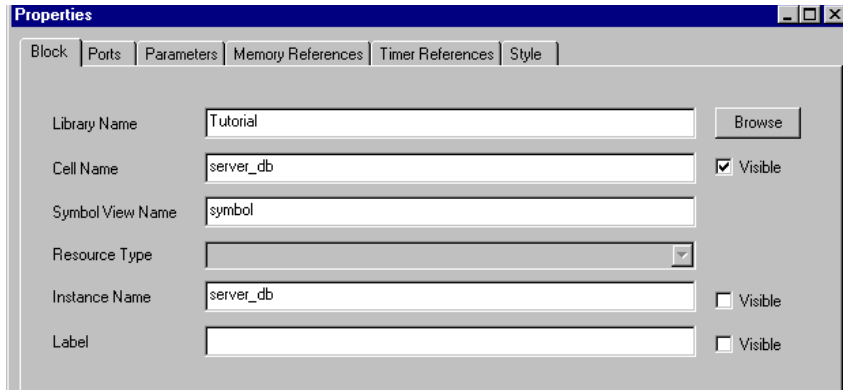
Library Name	Tutorial2
Cell Name	server_std
Symbol View Name	symbol
Instance Name	server_std

2. To enter each of these values, type the value in the appropriate field.
3. Make the Cell Name visible.
4. When you have entered all these values, click *OK* in the Properties dialog.
5. To specify the properties for the second block, right click on the rightmost block and choose *Properties* from the pop-up menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The Properties dialog opens.



For this block, you will enter the following values:

Library Name	Tutorial2
Cell Name	server_db
Symbol View Name	symbol
Instance Name	server_db

6. To enter each of these values, type the value in the appropriate field.
7. Make the Cell Name visible.
8. When you have entered all these values, click *OK* in the Properties dialog.
9. To save the changes to the *server:behav* diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Add Ports to the Blocks

You will now add an input port to the *server_db* block and an input and an output port to the *server_std* block.

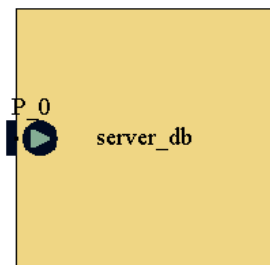
1. Click the *server_db* block to select it.

Make sure you see the green markers indicating that the block is selected.

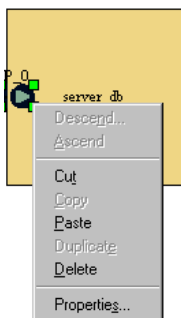
2. Choose *Behavior > Add Port* from the main menu.

When your cursor returns to the diagram, you'll see a port attached to it.

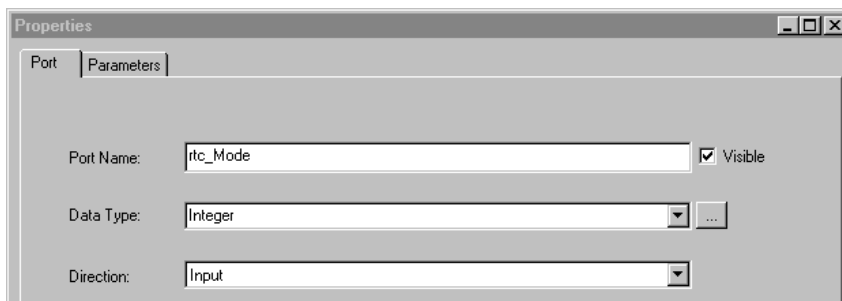
3. To place the port, click on the left edge of the block.



4. To specify properties for the port, right click on the port and choose *Properties* from the pop-up menu.



The Properties dialog opens.



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

For this port, you will enter the following values:

Port Name	<code>rtc_Mode</code>
Data Type	<code>Integer</code>
Direction	<code>Input</code>

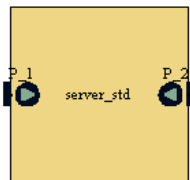
5. Type `rtc_Mode` in the Port Name field.
6. Make the Port Name visible.
7. To enter the Data Type, you can type `Integer` or choose from the pulldown menu.
8. To enter the Direction, you can type `Input` or choose from the pulldown menu.
9. When you have entered all these values, click *OK* in the Properties dialog.
10. Click the `server_std` block to select it.

Make sure you see the green markers indicating that the block is selected.

11. Choose *Behavior > Add Port* from the main menu.

When your cursor returns to the diagram, you'll see a port attached to it.

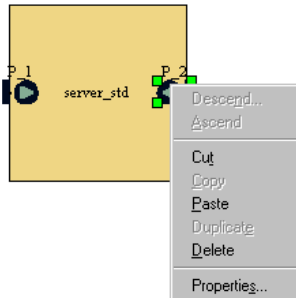
12. To place the first port, click on the left edge of the block.
13. Select the `server_std` block and choose *Behavior > Add Port* again.
14. Place the second port on the right edge of the block.



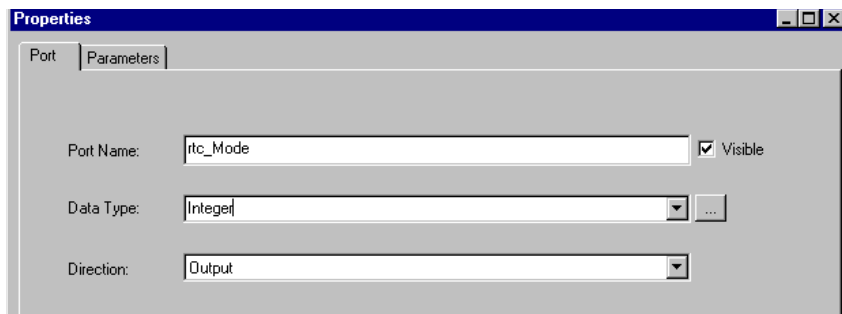
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

15. To specify properties for the right port, right click on the port and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following values:

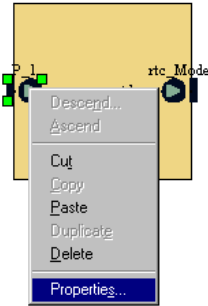
Port Name	rtc_Mode
Data Type	Integer
Direction	Output

16. Type *rtc_Mode* in the Port Name field.
17. Make the Port Name visible.
18. To enter the Data Type, you can type *Integer* or choose from the pulldown menu.
19. To enter the Direction, you can type *Output* or choose from the pulldown menu.
20. When you have entered all these values, click *OK* in the Properties dialog.

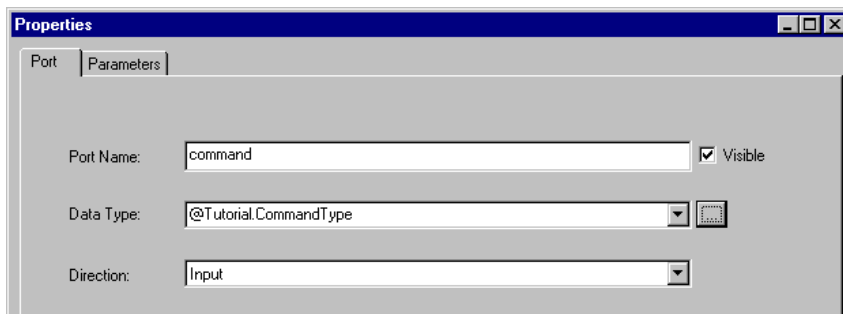
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

21. To specify properties for the left port, right click on the port and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this port, you will enter the following values:

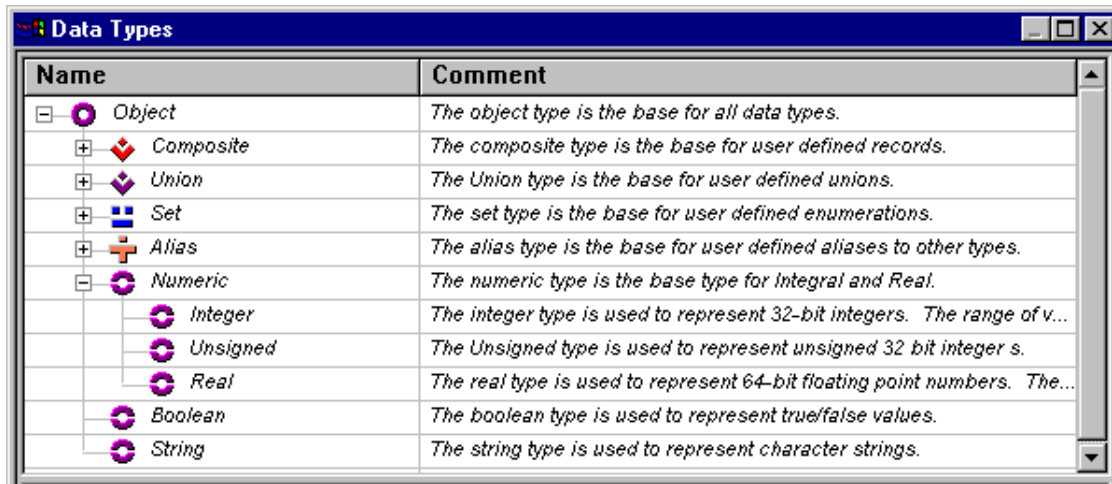
Port Name	command
Data Type	@Tutorial2.CommandType
Direction	Input

22. Type *command* in the Port Name field.
23. To enter the Data Type, click the browser (...) button.

Tutorial 2: VCC Hierarchical Modeling

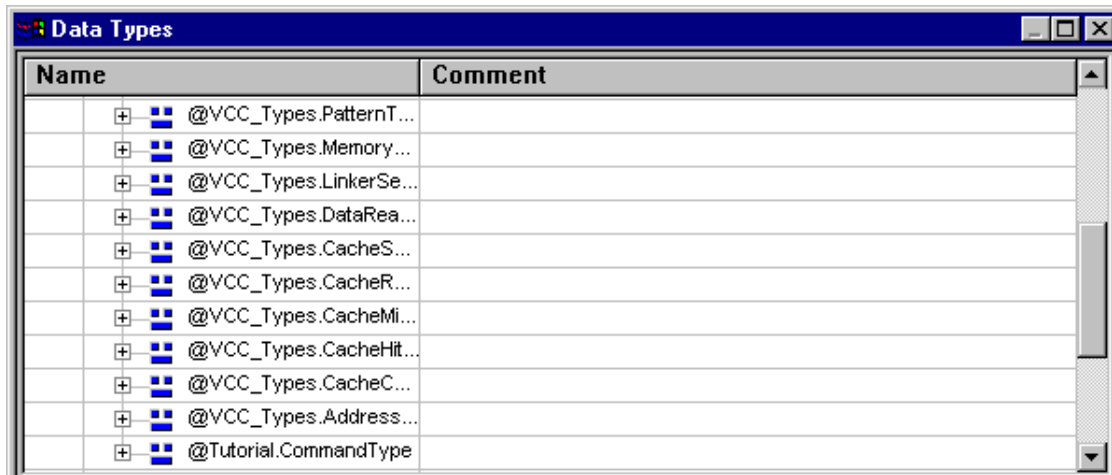
Behavior Model

The Data Types browser opens.



The new data type you created, *@Tutorial2.CommandType*, is a Set type.

24. Click the plus sign to expand the Set data types.
25. Scroll to the bottom of the Set types list and double-click *@Tutorial2.CommandType*.



The Data Types browser closes and *@Tutorial2.CommandType* is entered in the Properties form.

26. To enter the value for Direction, you can type the value *Input* or click the down arrow and choose from the pulldown menu.
27. When you have entered all these values, click *OK* in the Properties dialog.
28. To save the changes to the *server:behav* diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Add an Input Port

When you place the server block in the next level of hierarchy, you must have access to the blocks. For this, you will use a dedicated Input Port.

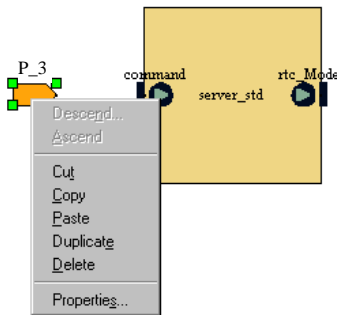
1. To add an input port to the *server* diagram, choose *Behavior > Input Port* from the main menu.

When you move your cursor back to the behavior diagram, you'll see an input port symbol attached.

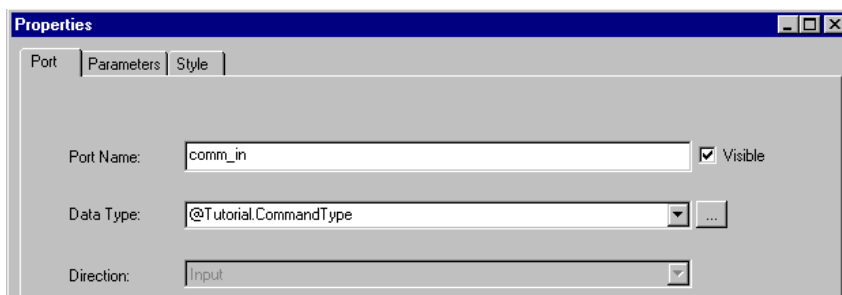
The input port symbol looks like this:



2. To place the input port, click to the left of the *server_std* block.
3. Right click on the input port symbol and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following values:

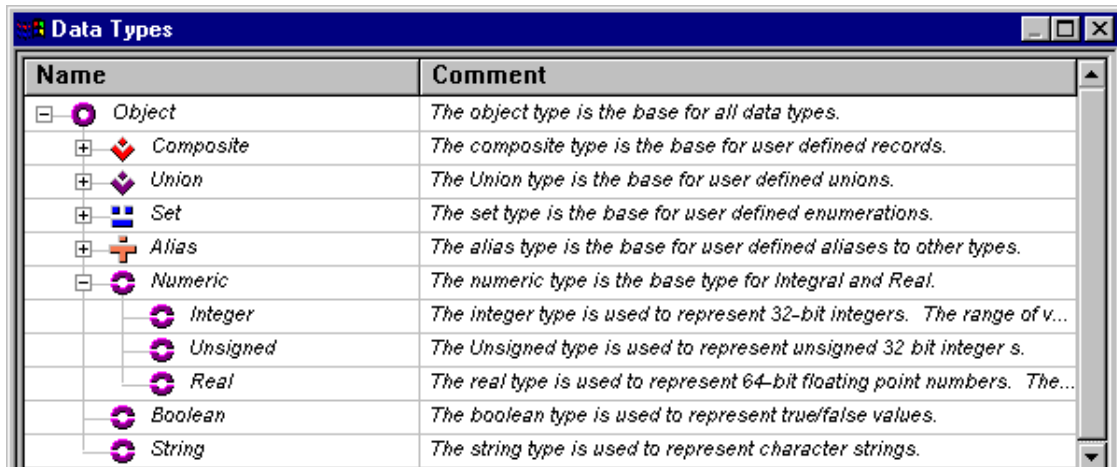
Port Name	comm_in
Data Type	@Tutorial2.CommandType

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

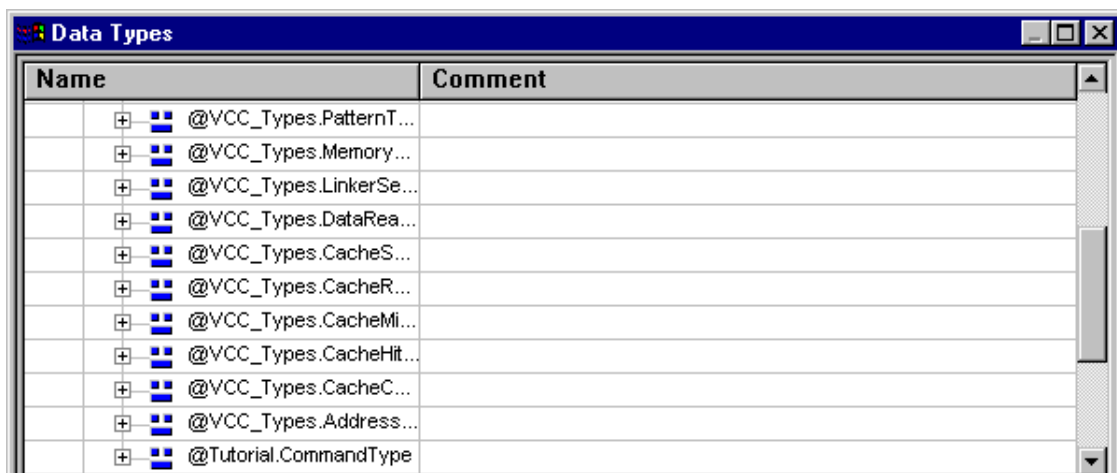
4. Type *comm_in* in the Port Name field.
5. Make the Port Name visible.
6. To enter the Data Type, click the browser (...) button.

The Data Types browser opens.



The new data type you created, *@Tutorial2.CommandType*, is a *Set* type.

7. Click the plus sign to expand the *Set* data types.
8. Scroll to the bottom of the *Set* types list and double-click *@Tutorial2.CommandType*.



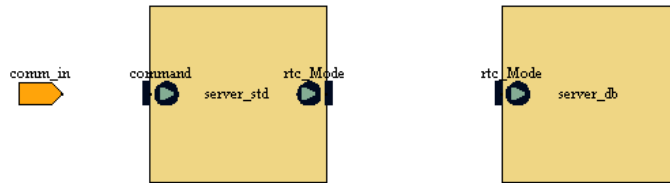
The Data Types browser closes and *@Tutorial2.CommandType* is entered in the Properties form.

9. When you have entered all these values, click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Your diagram now looks like this:

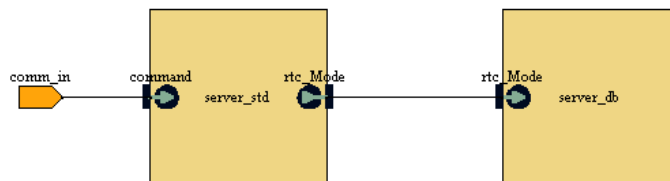


Connect the Ports

You now have ports on each of your two blocks and an external port (*comm_in*) at the top level of the *server* block. In this section, you will connect all the ports within the *server* block. Later, you will connect the *comm_in* port to the next level of hierarchy.

1. Choose *Behavior > Wire* from the main menu.
2. Click the *comm_in* input port, then click the *command* input port on the *server_std* block.
3. Choose *Behavior > Wire* again.
4. Click the *rtc_Mode* output port on the *server_std* block, then click the *rtc_Mode* input port on the *server_db* block.

Your diagram now looks like this:



5. To save the changes to the *server* diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

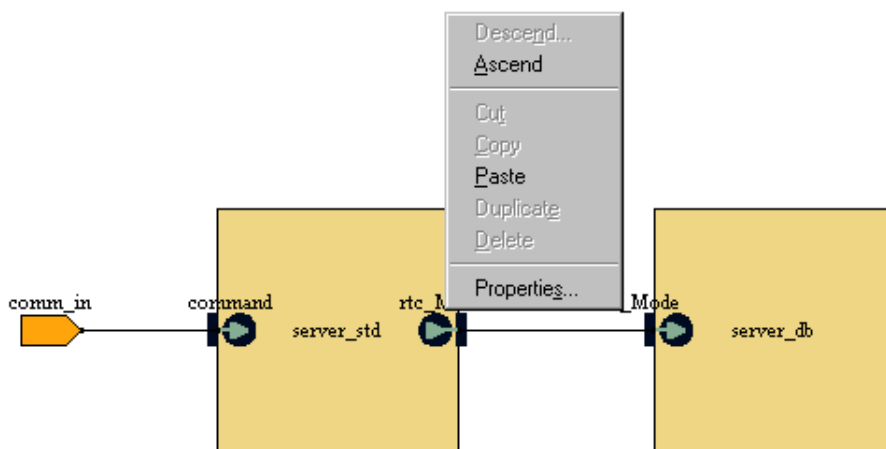
Behavior Model

Add Memory References

As you can see, there is no obvious output for this block. The output of the *server* block will be a virtual connection (a memory reference) to a shared memory in the next level of hierarchy. To specify this connection, you will add a memory reference to the top level of this diagram.

1. Right click in the background of the diagram.

Do not click on any of the objects in the diagram. The parameter you are going to specify is at the top level, not on one of the blocks.



2. Choose *Properties* from the pop-up menu.

The Properties form opens.



3. Click the *Memory References* tab.

There are no Memory References specified, so all the fields are blank.

4. Right click in any of the blank fields, and choose *New Property* from the pop-up menu.

A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

You will enter this new property.

Name	Type	Instance	Mode
sh_mem_s	Real[6]		WriteOnly

5. In the Name field, enter *sh_mem_s*.
6. Tab to the Type field and enter *Real[6]*.



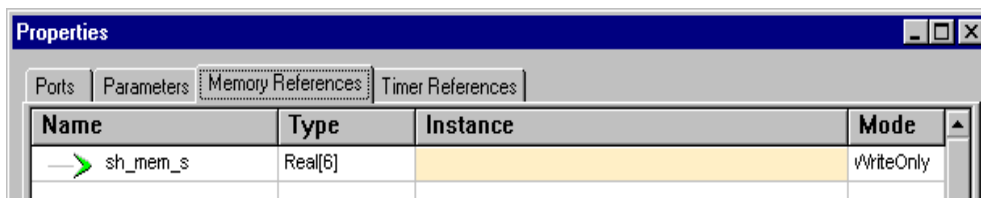
Tip

You can simply type the *Real[6]* value, or you can use the pulldown menu. If you choose *Real* from the pulldown menu, you must type the *[6]* to complete the value.

7. Tab to the Mode field and choose *WriteOnly* from the pulldown menu.

For now, you cannot enter a value for Instance. At the next level in the hierarchy, you will specify a shared memory instance to which this memory reference is connected.

Your new property looks like this:



8. When you have entered all these values, click *OK* in the Properties dialog.
9. To save the changes to the server diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Generate the Symbol

All blocks are saved as cells. All cells have an *interface* view and a *symbol* view.

- The *interface* view describes the ports and parameters for the cell.
- The *symbol* view is the graphical representation of the cell and can be instantiated into designs.

When you “generate a symbol” for the *server*, VCC creates the *interface* and *symbol* views, bundles them with the *behav* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

Note: The parameters of the cellview are no longer at the top level of the behavior diagram in which the block was created. If you need to change any of these parameters later, you must edit the *symbol* view.

1. To generate a symbol for this behavior diagram, choose *Behavior > Generate Symbol* from the main menu.

The Behavior Notation dialog tells you this will create an *interface* file and a *symbol* file.

2. Click *OK* in the Behavior Notation dialog.

You can see in the Project Folder that the new *server* cell is added to the *Tutorial2* library.



The system creates the *server:interface* and the *server:symbol* views.

3. Make sure the *server:behav* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

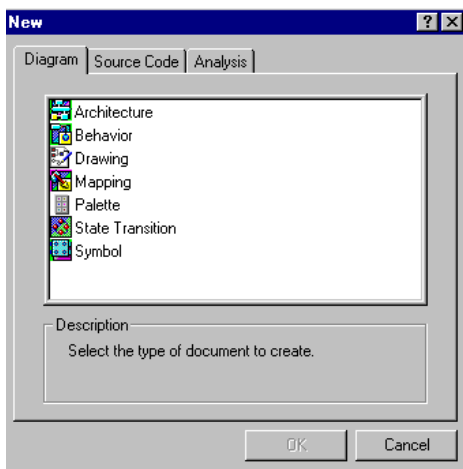
Create the Controller Behavior Diagram

The real-time controller requires a custom controller. In this section, you will:

- [Create a New Behavior Diagram](#) on page 33
- [Create the Timer and sinComb Blocks](#) on page 35
- [Add Ports to the Blocks](#) on page 38
- [Add an Output Port](#) on page 44
- [Add a Behavior Timer](#) on page 45
- [Connect the Ports](#) on page 46
- [Add Parameters and Memory References](#) on page 47
- [Generate the Symbol](#) on page 49

Create a New Behavior Diagram

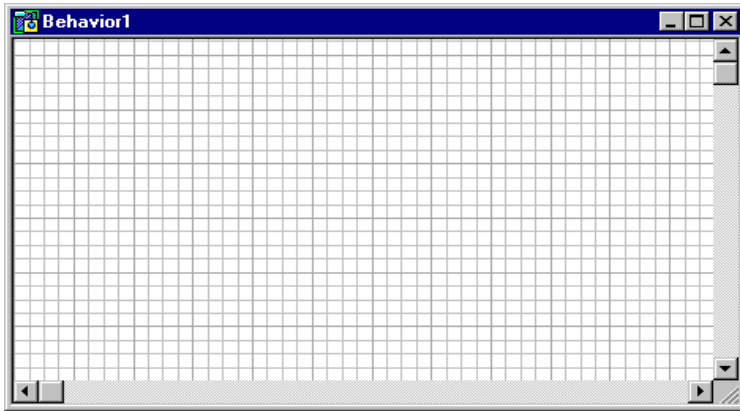
1. To open a new behavior diagram, choose *File > New* from the main menu.



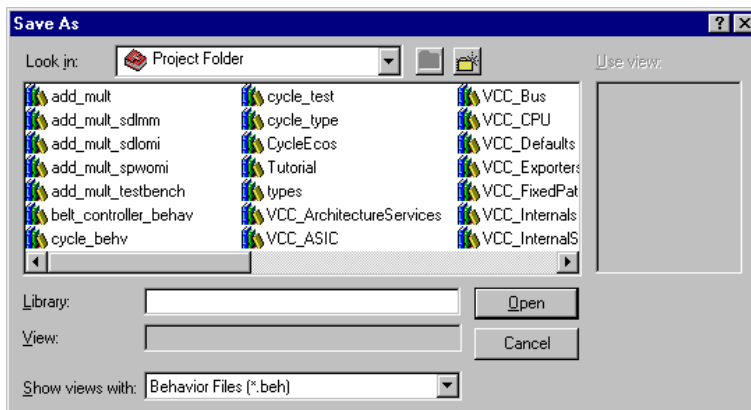
Tutorial 2: VCC Hierarchical Modeling Behavior Model

2. Double-click the *Behavior* icon in the New dialog.

A new window opens.



3. To save the behavior diagram, choose *File > Save* from the main menu.
4. The Save As browser opens and VCC prompts you to choose a library.

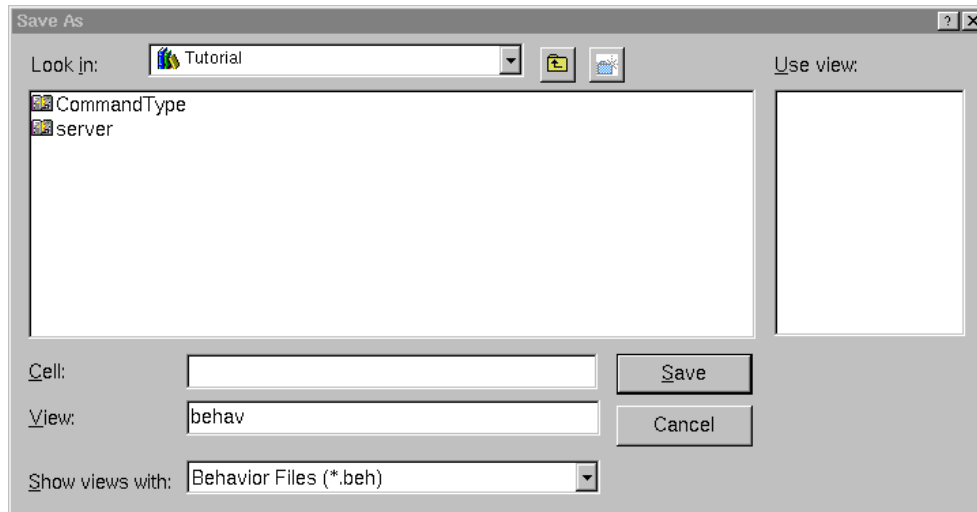


5. To choose the *Tutorial2* library from the Project Folder, double-click the *Tutorial2* icon in the library list.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

VCC opens the *Tutorial2* library and prompts you to specify a Cell name. The View name defaults to *behav*.



6. Enter the Cell name *controller* and click *Save*.

Create the *Timer* and *sinComb* Blocks

The *Timer* and *sinComb* blocks do not exist, so you must create them.

- The *Timer* block is a C++ model whose output triggers the *sinComb* execution.
- The *sinComb* block is a Whitebox C model that implements the sine waves convolution algorithm.

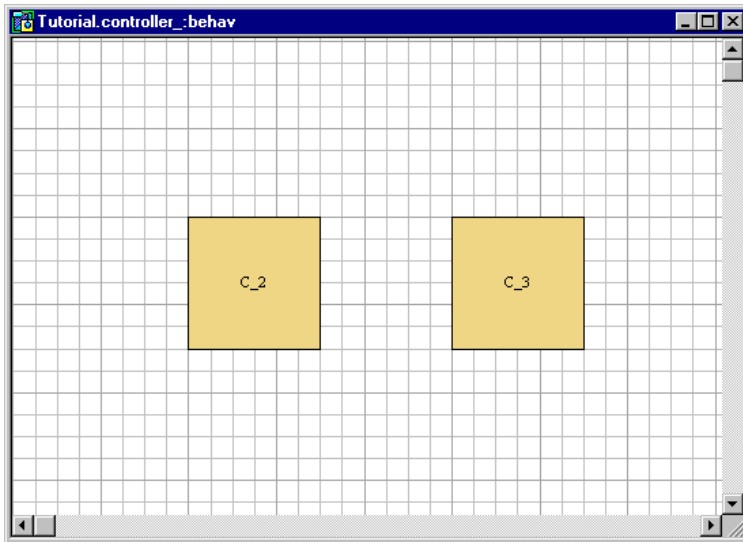
Draw the Blocks

1. To draw a new block, choose *Behavior > Block* from the main menu.
2. In the behavior diagram, click to place the upper-left corner of the block, then click to place the lower-right corner of the block.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

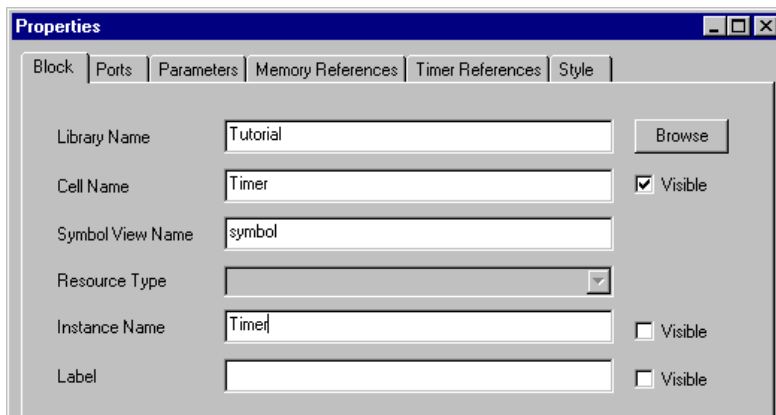
3. Repeat steps 1 and 2 to draw a second block.



Specify Block Properties

1. To specify the properties for the first block, right click on the leftmost block and choose *Properties* from the pop-up menu.

The Properties dialog opens.



For this block, you will enter the following values:

Library Name	Tutorial2
Cell Name	Timer
Symbol View Name	symbol

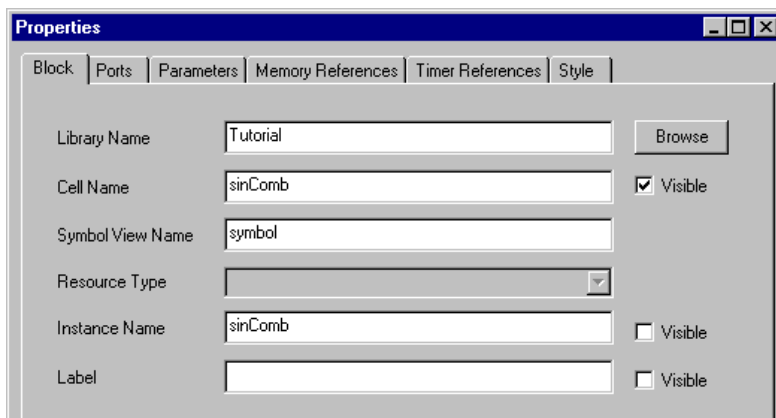
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Instance Name Timer

2. Enter *Tutorial2* for the Library Name.
3. Enter *Timer* for the Cell Name and make it visible.
4. Enter *symbol* for the Symbol View Name.
5. Enter *Timer* for the Instance Name.
6. When you have entered all these values, click *OK* in the Properties dialog.
7. To specify the properties for the second block, right click on the rightmost block and choose *Properties* from the pop-up menu.

The Properties dialog opens.



For this block, you will enter the following values:

Library Name Tutorial2
Cell Name sinComb
Symbol View Name symbol
Instance Name sinComb

8. Enter *Tutorial2* for the Library Name.
9. Enter *sinComb* for the Cell Name and make it visible.
10. Enter *symbol* for the Symbol View Name.
11. Enter *sinComb* for the Instance Name.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

12. When you have entered all these values, click *OK* in the Properties dialog.
13. To save the changes to the *controller* diagram, choose *File > Save* from the main menu.

Add Ports to the Blocks

You will add an output port to the *Timer* block and input and output ports to the *sinComb* block.

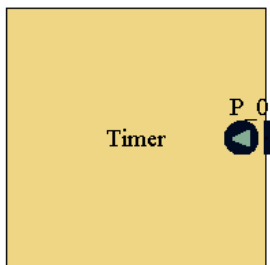
1. Click the *Timer* block to select it.

Make sure you see the green markers indicating that the block is selected.

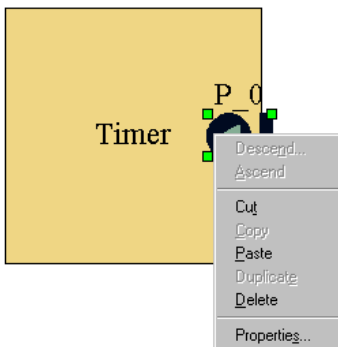
2. Choose *Behavior > Add Port* from the main menu.

When your cursor returns to the diagram, you'll see a port attached to it.

3. To place the port, click on the right edge of the block.



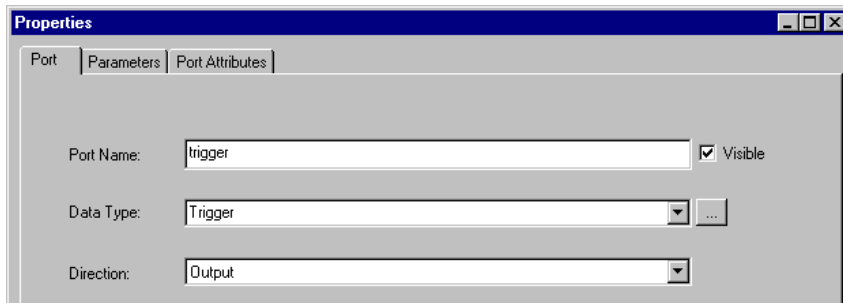
4. To specify properties for the port, right click on the port and choose *Properties* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The Properties dialog opens.

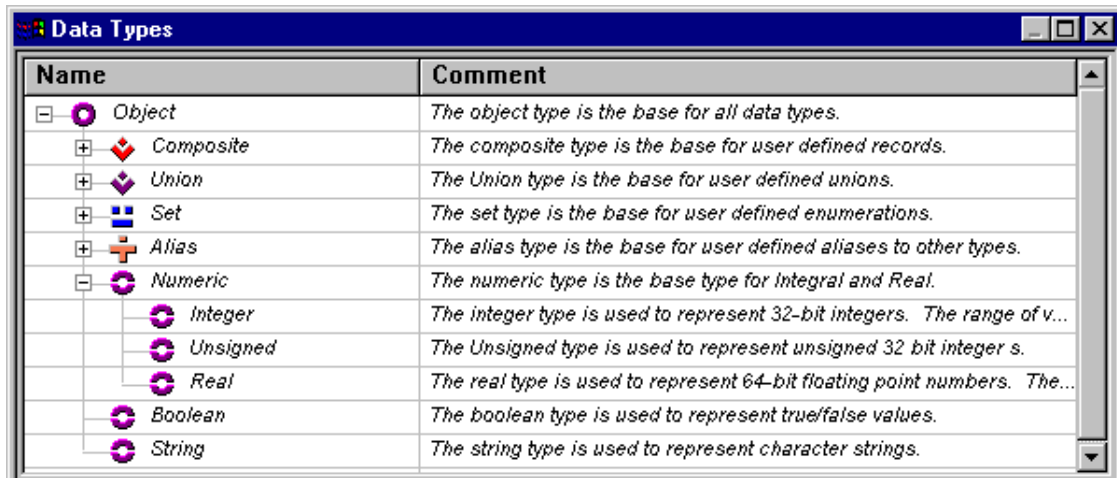


For this block, you will enter the following values:

Port Name	trigger
Data Type	Trigger
Direction	Output

5. Enter *trigger* for the Port Name.
6. To enter the Data Type, click the browser (...) button.

The Data Types browser opens.

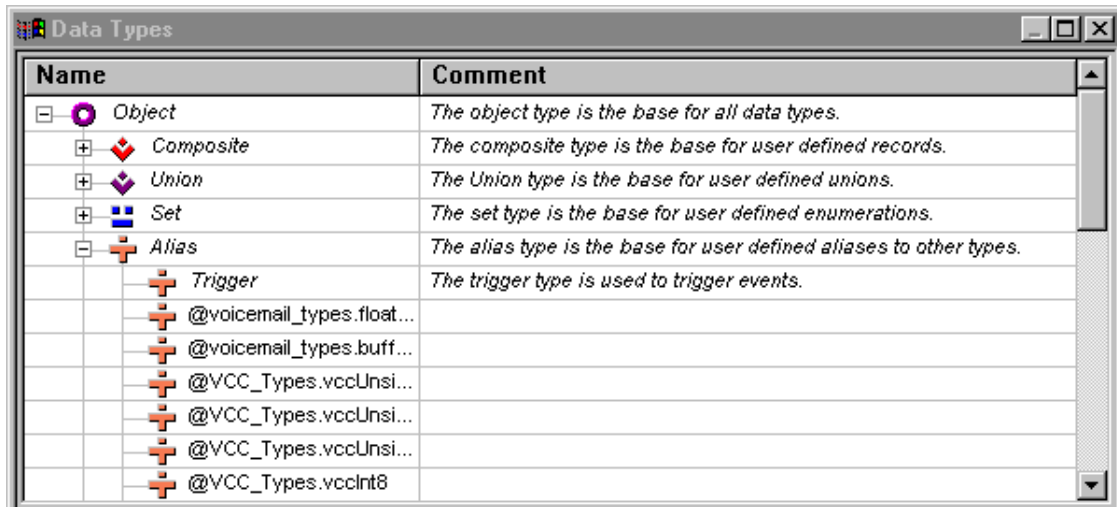


Trigger is an *Alias* type.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

- Click the plus sign to expand the *Alias* data types.



- Double-click *Trigger*.

The Data Types browser closes and *Trigger* is entered in the Properties form.

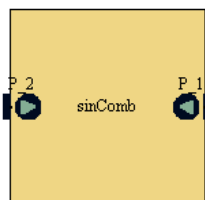
- To enter the value for Direction, you can type *Output* or choose from the pulldown menu.
- When you have entered all these values, click *OK* in the Properties dialog.
- Click the *sinComb* block to select it.

Make sure you see the green markers indicating that the block is selected.

- Choose *Behavior > Add Port* from the main menu.

When your cursor returns to the diagram, you'll see a port attached to it.

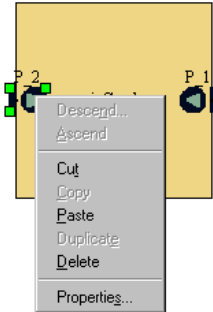
- To place the first port, click on the left edge of the block.
- Select the *sinComb* block and choose *Behavior > Add Port* again.
- Place the second port on the right edge of the block.



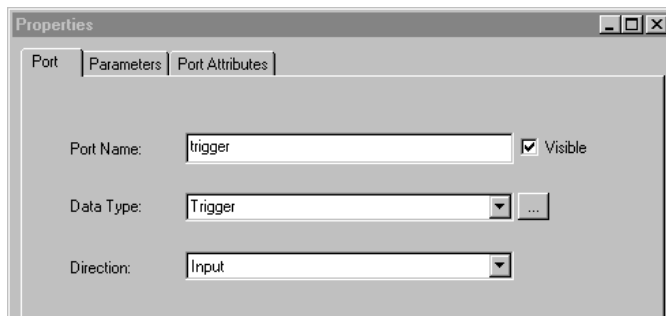
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

16. To specify properties for the input port on the *sinComb* block, right click on the leftmost port and choose *Properties* from the pop-up menu.



The Properties dialog opens.



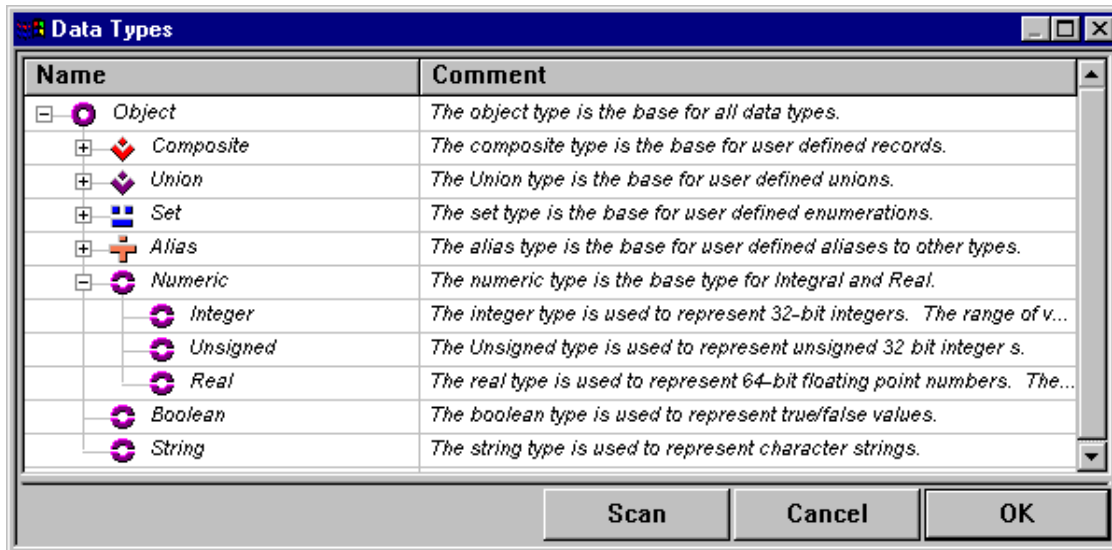
For this block, you will enter the following values:

Port Name	trigger
Data Type	Trigger
Direction	Input

17. Enter *trigger* for the Port Name.
18. To enter the Data Type, click the browser (...) button.

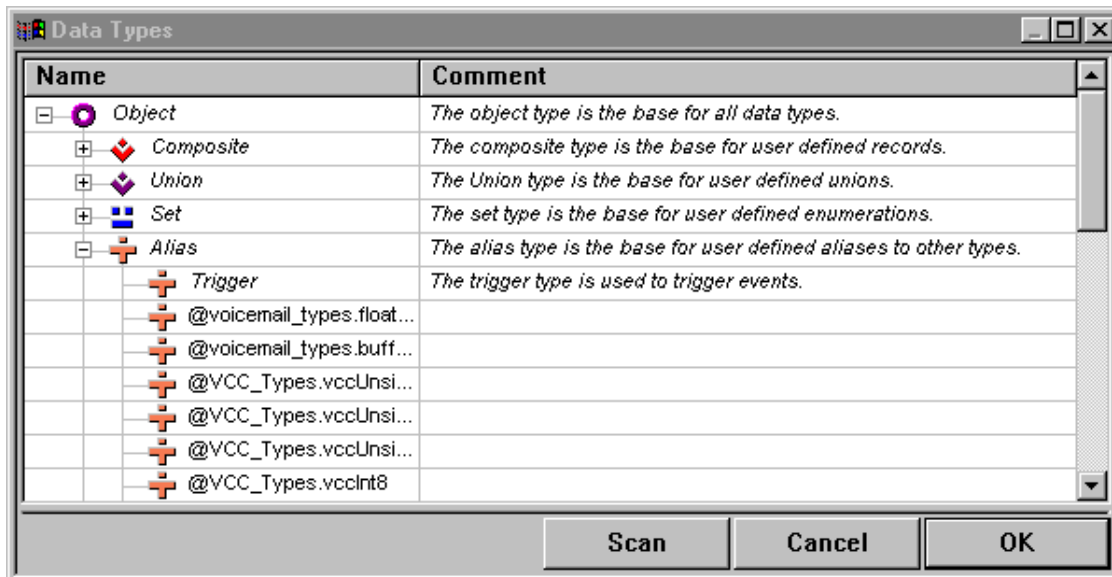
Tutorial 2: VCC Hierarchical Modeling Behavior Model

The Data Types browser opens.



Trigger is an *Alias* type.

19. Click the plus sign to expand the *Alias* data types.



20. Double-click *Trigger*.

The Data Types browser closes and *Trigger* is entered in the Properties dialog.

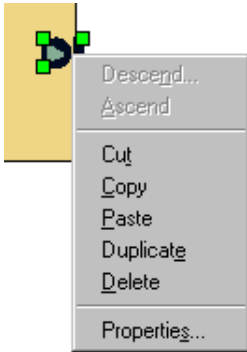
21. To enter the value for Direction, you can type *Input* or choose from the pulldown menu.

22. When you have entered these values, click *OK* in the Properties dialog.

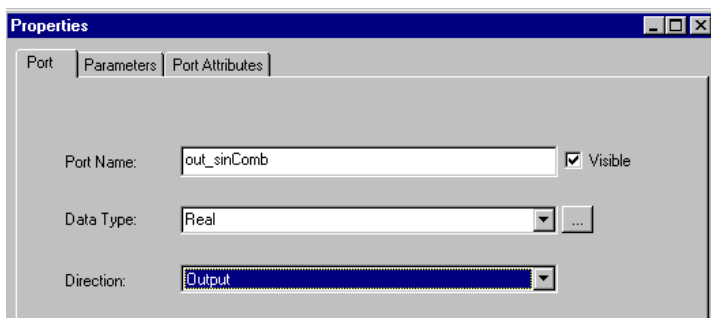
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

23. To specify properties for the output port on the *sinComb* block, right click on the rightmost port and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following values:

Port Name	out_sinComb
Data Type	Real
Direction	Output

24. Enter *out_sinComb* for the Port Name.
25. To enter the Data Type, you can type *Real* or choose from the pulldown menu.
26. To enter the Direction, you can type *Output* or choose from the pulldown menu.
27. When you have entered all these values, click *OK* in the Properties dialog.
28. To save the changes to the *controller* diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

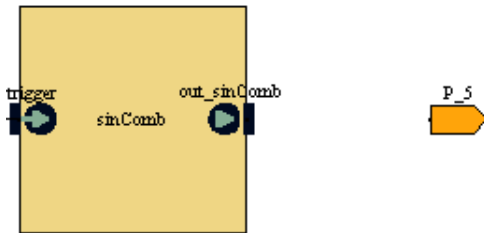
Add an Output Port

When you place the *controller* block in the next level of hierarchy, you must have access to the output of the controller. For this, you will use a dedicated Output Port.

1. To add an output port to the *controller* diagram, choose *Behavior > Output Port* from the main menu.

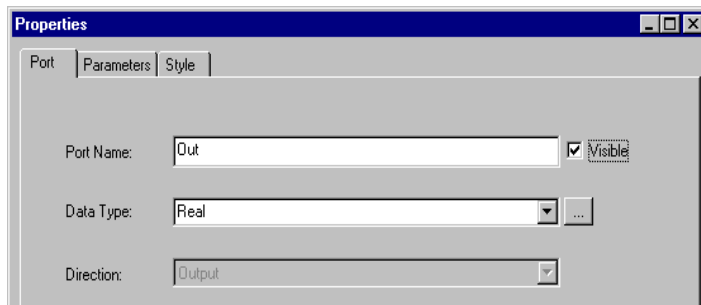
When you move your cursor back to the behavior diagram, you'll see an output port symbol attached.

2. To place the output port, click to the right of the *sinComb* block.



3. Right click on the output port symbol and choose *Properties* from the pop-up menu.

The Properties dialog opens.



For this port, you will enter the following values:

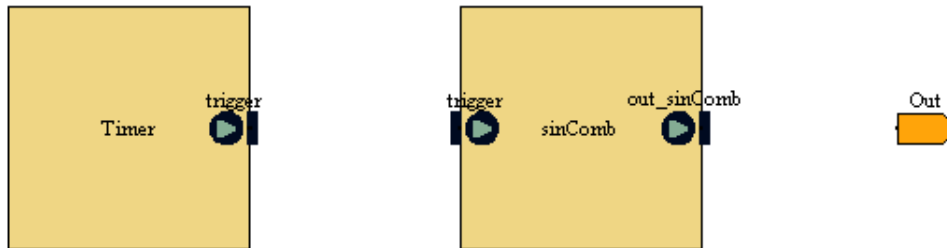
Port Name	Out
Data Type	Real

4. Enter *Out* for the Port Name and make it visible.
5. To enter the Data Type, you can type *Real* or choose from the pulldown menu.
6. When you have entered all these values, click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Your diagram now looks like this:



Add a Behavior Timer

Behavior timers model periodic or pre-scheduled activation of a behavior model. In this diagram, the behavior timer will activate the *Timer*.

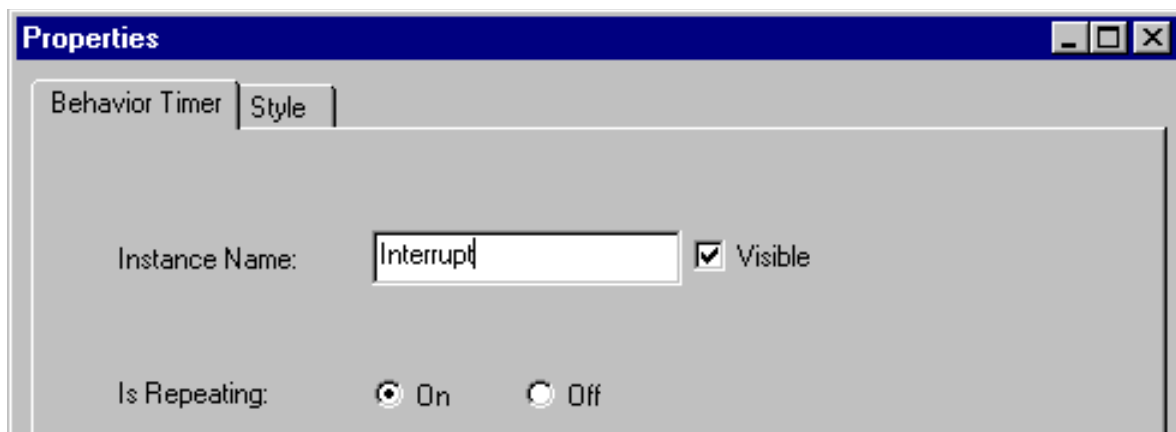
1. Choose *Behavior > Behavior Timer* from the main menu.

When you move your cursor back to the behavior diagram, you'll see a behavior timer block attached.

2. To place the behavior timer, click above the *Timer* and *sinComb* blocks.
3. Right click on the behavior timer and choose *Properties* from the pop-up menu.



The Properties dialog opens.



Tutorial 2: VCC Hierarchical Modeling

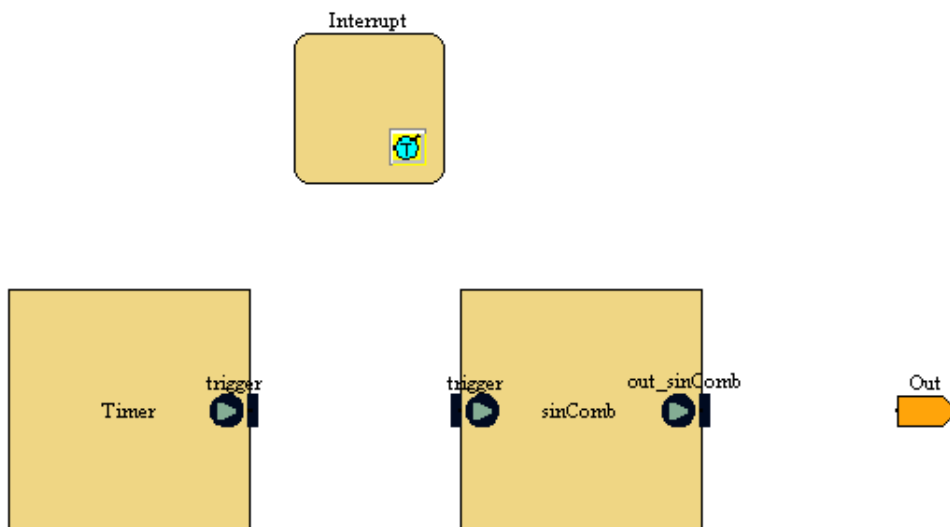
Behavior Model

For this block, you will enter the following values:

Instance Name `Interrupt`
Is Repeating `On`

4. Enter *Interrupt* for the Instance Name and make it visible.
5. Set *Is Repeating* to *On*.
6. When you have entered all these values, click *OK* in the Properties dialog.

Your diagram now looks like this:



7. To save the changes to the *controller* diagram, choose *File > Save* from the main menu.

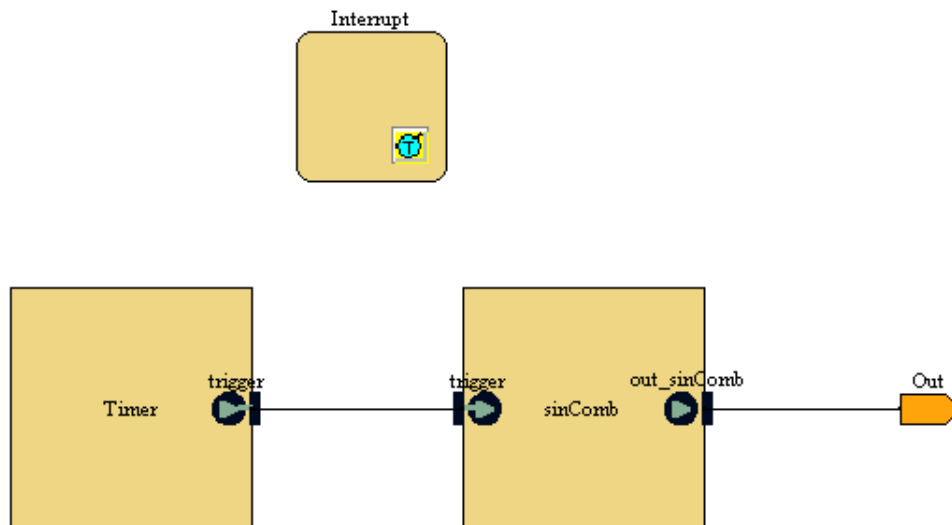
Connect the Ports

1. Choose *Behavior > Wire* from the main menu.
2. Click the *trigger* output port on the *Timer* block, then click the *trigger* input port on the *sinComb* block.
3. Choose *Behavior > Wire* again.
4. Click the *out_sinComb* output port on the *sinComb* block, then click the *Out* output port.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Your diagram now looks like this:



5. To save the changes in the *controller* diagram, choose *File > Save* from the main menu.

Add Parameters and Memory References

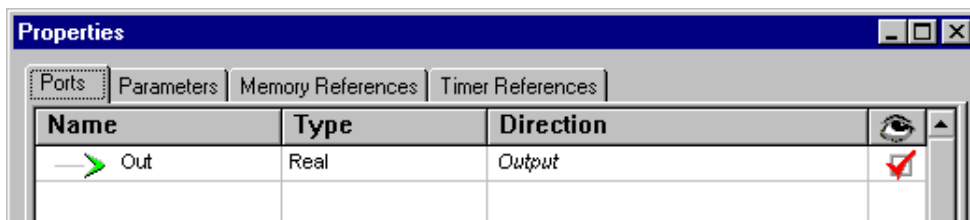
As you can see, there is no obvious input for this *controller* block. The input of the controller will be a virtual connection (a memory reference) to a shared memory in the next level of hierarchy. To specify this connection, you will add a memory reference to the top level of this diagram.

1. Right click in the background of the diagram.

Do not click on any of the objects in the diagram. The parameter you are going to specify is at the top level, not on one of the blocks.

2. Choose *Properties* from the pop-up menu.

The Properties form opens. The default tab is the *Ports* tab. This tab shows the parameter for the top-level output port, *Out*.



3. Click the *Parameters* tab.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

There are no Parameters specified, so all the fields are blank.

4. Right click in any of the blank fields, and choose *New Property* from the pop-up menu.

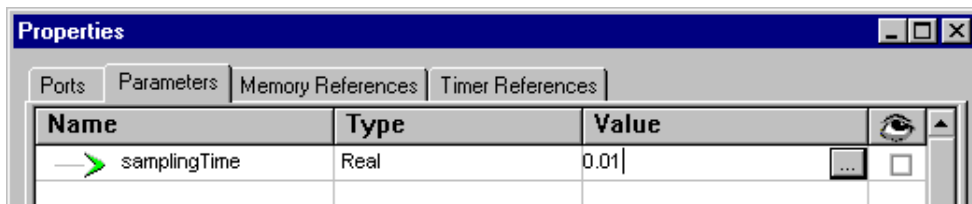
A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property.

Name	Type	Value
samplingTime	Real	0.01

5. Enter *samplingTime* for the Name.
6. Tab to the Type field and type *Real* or choose from the pulldown menu.
7. Double-click in the Value field and enter *0.01*.

Your new property looks like this:



8. Click *Apply*.
9. Click the *Memory References* tab.

There are no Memory References specified, so all the fields are blank.

10. Right click in any of the blank fields, and choose *New Property* from the pop-up menu.

A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property.

Name	Type	Instance	Mode
sh_mem_c	Real[6]		ReadOnly

11. Enter *sh_mem_c* for the Name.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

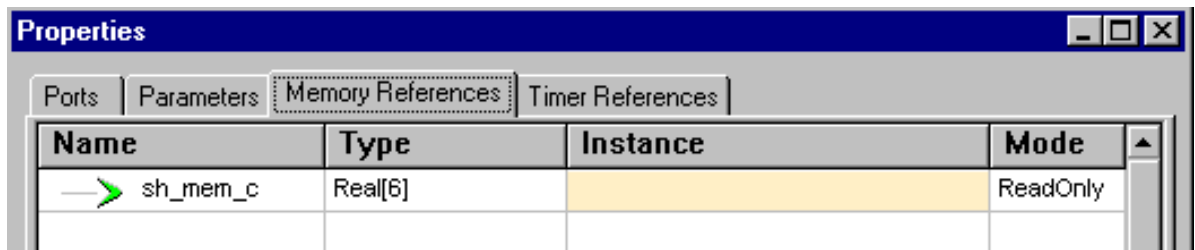
12. Tab to the Type field and enter *Real[6]*.

If you choose *Real* from the pulldown menu, you must type the *[6]* to complete the value.

13. Double-click in the Mode field and choose *ReadOnly* from the pulldown menu.

For now, you cannot enter a value for Instance. At the next level in the hierarchy, you will specify a shared memory instance to which this memory reference is connected.

Your new property looks like this:



14. When you have entered all these values, click *OK* in the Properties dialog.
15. To save the changes to the *controller* diagram, choose *File > Save* from the main menu.

Generate the Symbol

When you “generate a symbol” for the *controller* cellview, VCC creates the *interface* and *symbol* views, bundles them with the *behav* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

1. To generate a symbol for this behavior diagram, choose *Behavior > Generate Symbol* from the main menu.
2. Click *OK* in the Behavior Notation dialog.

You can see in the Project Folder that the new *controller* cell is added to the *Tutorial2* library.



The system creates the *controller:interface* and the *controller:symbol* views.

3. Make sure the *controller:behav* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

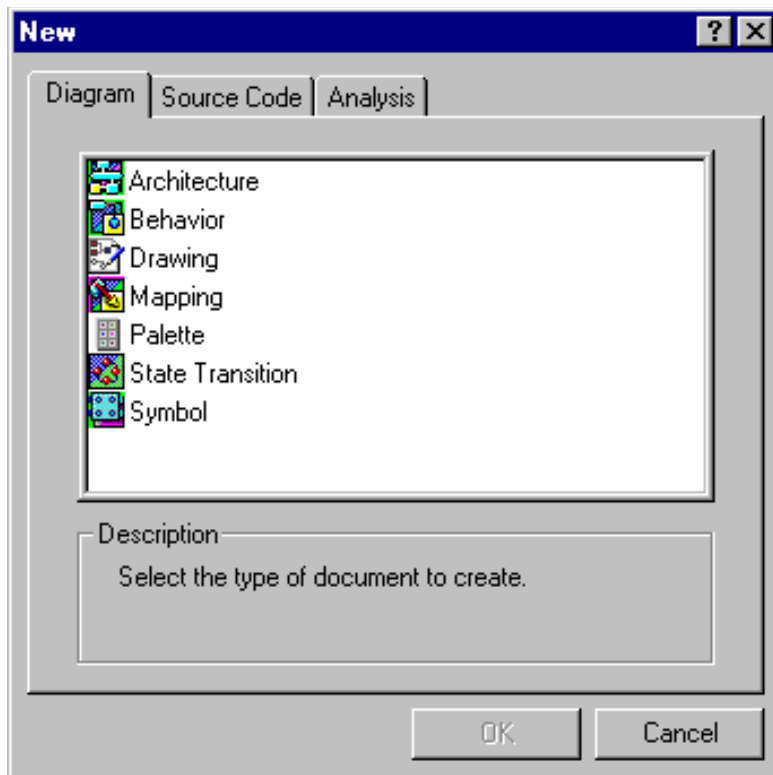
Create a Top-Level Behavior Diagram

To create the top-level behavior model for this real-time controller, you will instantiate the two main subsystems: the *server* and the *controller*. In this section, you will:

- [Create a New Behavior Diagram](#) on page 50
- [Instantiate the Blocks](#) on page 52
- [Specify Block Parameters](#) on page 55
- [Wire the Blocks](#) on page 61
- [Specify Top-Level Parameters](#) on page 62
- [Bind the Parameters](#) on page 63
- [Associate the Memory References with the Behavior Memory](#) on page 64

Create a New Behavior Diagram

1. To open a new behavior diagram, choose *File > New* from the main menu.

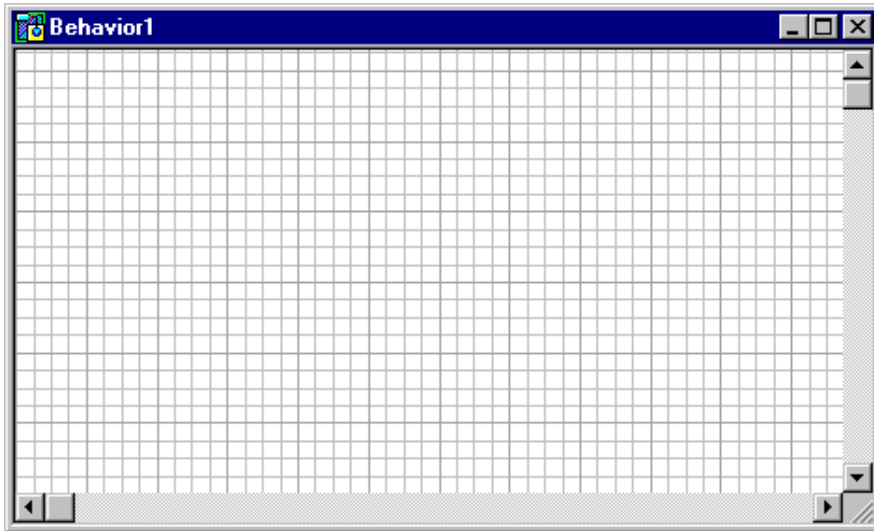


Tutorial 2: VCC Hierarchical Modeling

Behavior Model

2. Double-click the *Behavior* icon in the New dialog.

A new behavior window opens.

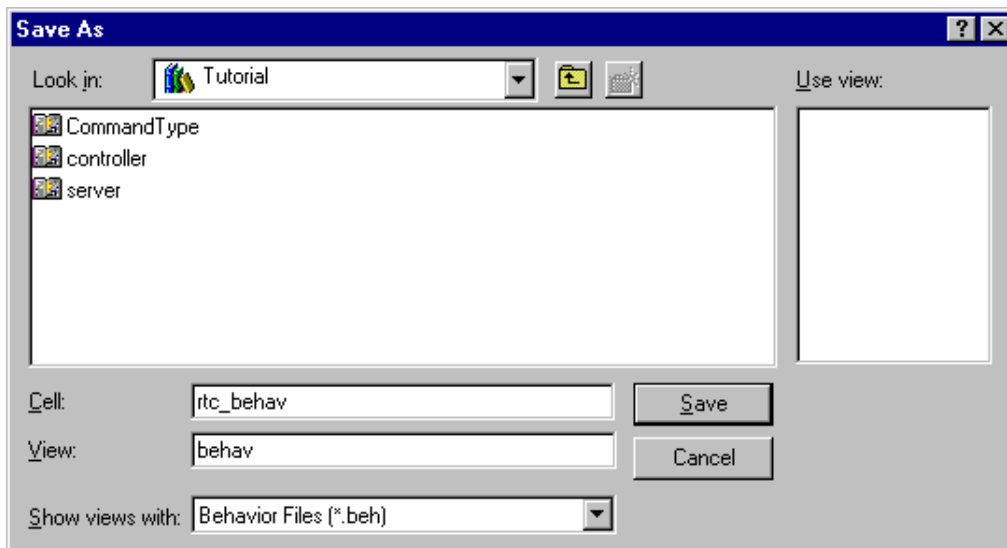


3. To save the behavior diagram, choose *File > Save* from the main menu.

VCC prompts you to choose a library.

4. Double-click the *Tutorial2* icon in the browser.

VCC opens the *Tutorial2* library and prompts you for a Cell name. The View name defaults to *behav*.



5. Enter the Cell name *rtc_behav* and click *Save*.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Instantiate the Blocks

1. On the Project Folder, expand the *Tutorial2* library, then expand the *controller* cell.
2. Instantiate the *controller* symbol view in your diagram.

To instantiate a block, click the symbol view, drag the symbol to your diagram, and drop the symbol.

3. In the *Tutorial2* library, expand the *server* cell.
4. Instantiate the *server* symbol view in your diagram.
5. When you finish instantiating the *server* block, click the minus sign to collapse the expanded *Tutorial2* library.

Your diagram now looks like this:

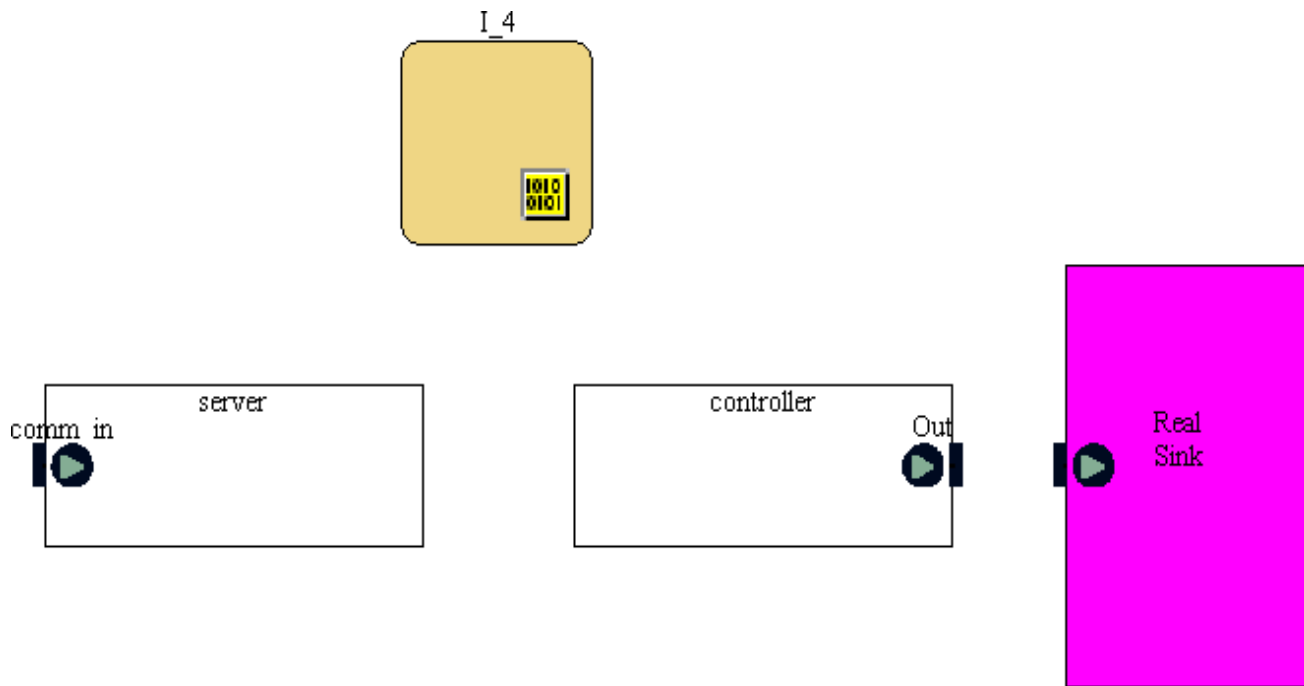


You will now add three blocks to this diagram: a *real_sink*, a behavior memory, and a testbench.

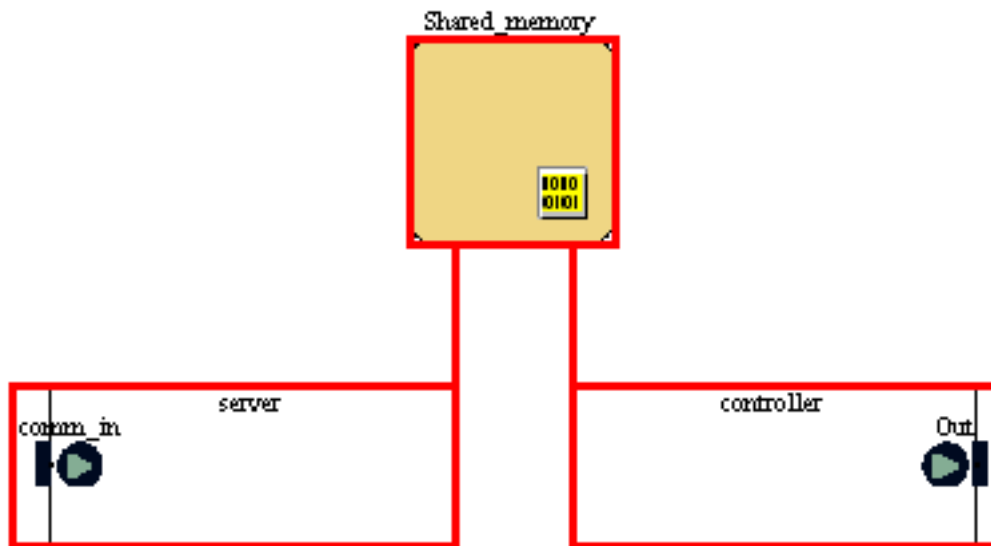
6. On the Project Folder, expand the *add_mult_testbench* library, then expand the *real_sink* cell.
7. Instantiate the *real_sink* symbol view in your diagram to the right of the *controller* block.
8. To instantiate a Behavior Memory in your diagram, choose *Behavior > Behavior Memory* from the main menu.
9. Instantiate the *Behavior Memory* in your diagram above the *controller* and *server* blocks.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

Your diagram now looks like this:



Because of the memory references in the *server* and *controller* blocks, the behavior memory is virtually connected to these blocks. You can see these connections only when the cursor is over the behavior memory.



Tutorial 2: VCC Hierarchical Modeling

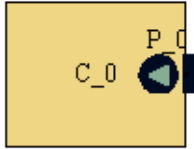
Behavior Model

10. To create a new block that you will use for the *rtc_testbench*, choose *Behavior > Block* from the main menu.
11. In the behavior diagram, draw a block to the left of the *server* block.
12. To add an output port to this new block, choose *Behavior > Add Port* from the main menu.

When you move your cursor back into the diagram, a port is attached to it.

13. Place the port on the right edge of the block.

The new block looks like this:

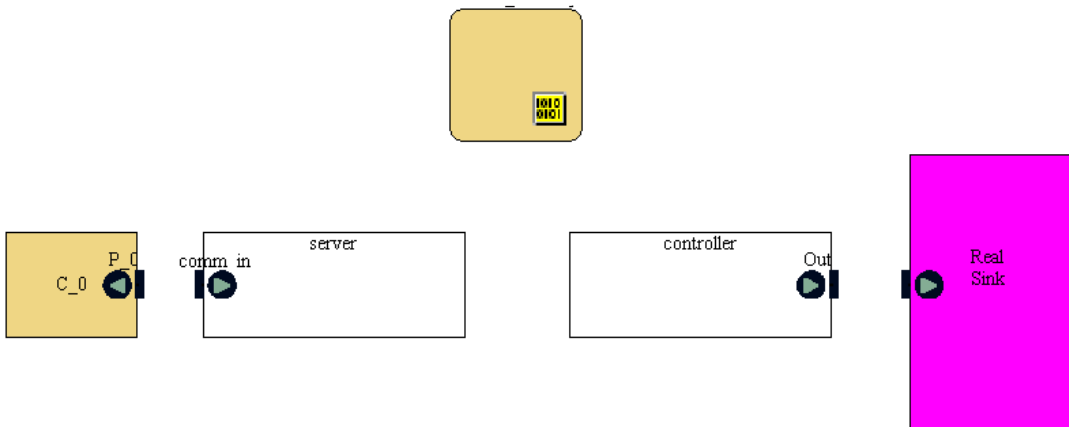


14. To save the behavior diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

Specify Block Parameters

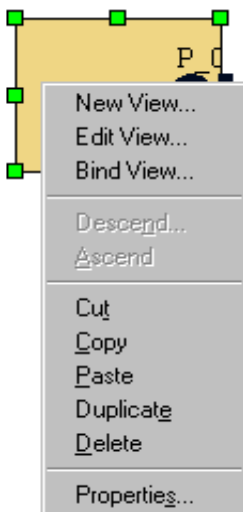
Your diagram looks like this:



Moving from left to right, you will specify parameter values for each of the blocks.

Specify *rtc_testbench* Parameters

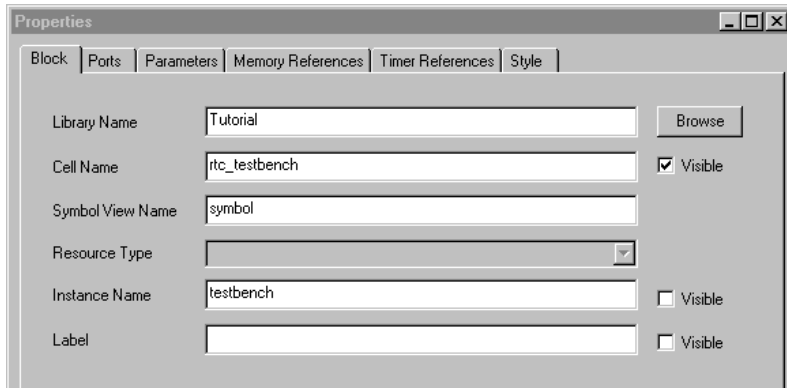
1. Right click on the new block to the left of the *server* block and choose *Properties* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

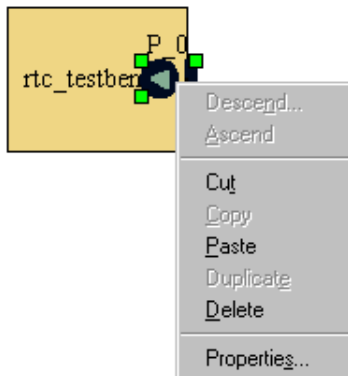
The Properties dialog opens.



For this block, you will enter the following values:

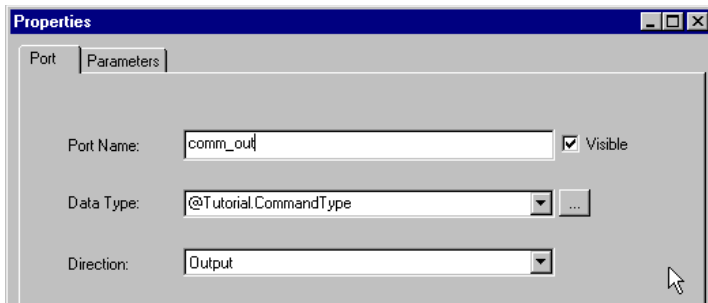
Library Name	Tutorial2
Cell Name	rtc_testbench
Symbol View Name	symbol
Instance Name	testbench

2. To enter each of these values, type the value in the appropriate field.
3. Make the Cell Name visible.
4. When you have entered all these values, click *OK* in the Properties dialog.
5. Right click on the *rtc_testbench* output port and choose *Properties* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling Behavior Model

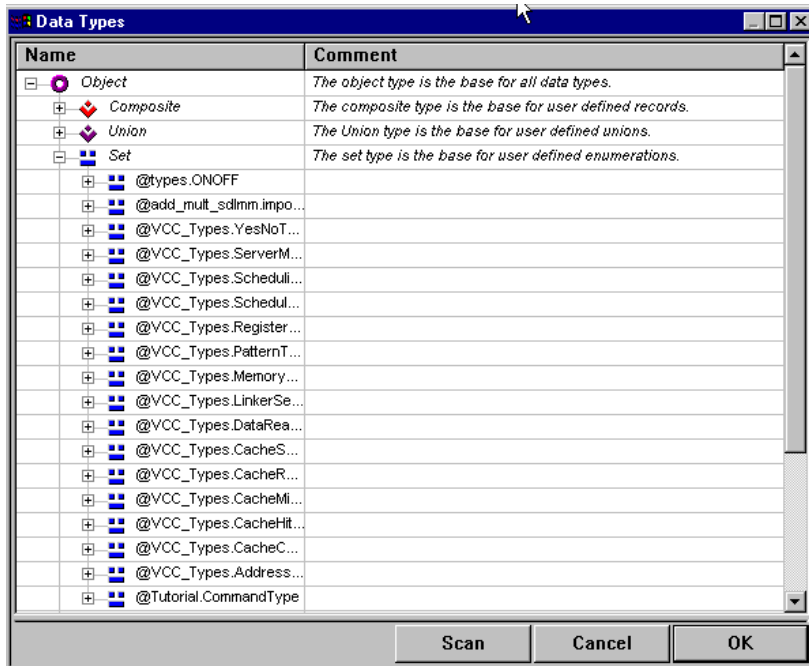
The Properties dialog opens.



For this block, you will enter the following values:

Port Name	comm_out
Data Type	@Tutorial2.CommandType
Direction	Output

6. Enter *comm_out* for the Port Name and make it visible.
7. To enter the Data Type, click the browse (...) button.
8. Expand the *Set* types.



9. Scroll to the bottom of the Set types list.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

10. Double-click *@Tutorial2.CommandType*.

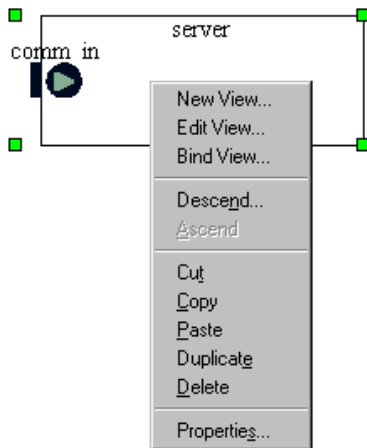
The Data Types browser closes and the *@Tutorial2.CommandType* data type is entered in the Data Type field.

11. To enter the Direction, you can type *Output* or choose from the pulldown menu.

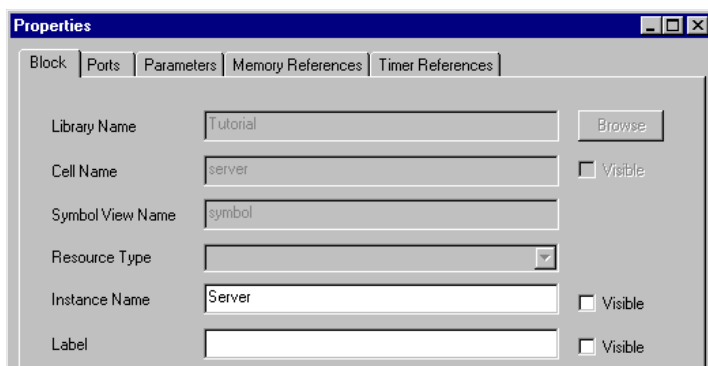
12. When you have entered all these values, click *OK* in the Properties dialog.

Specify server Parameters

1. Right click on the *server* block and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following value:

Instance Name Server

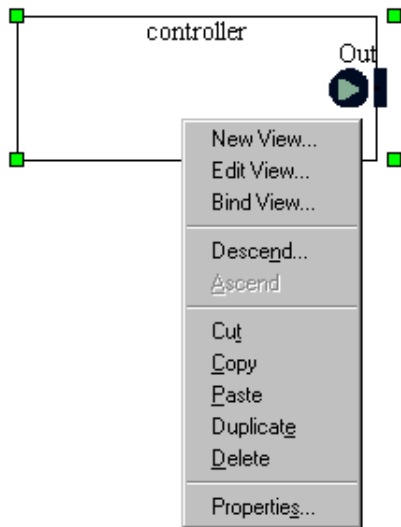
2. Enter *Server* for the Instance Name, then click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling

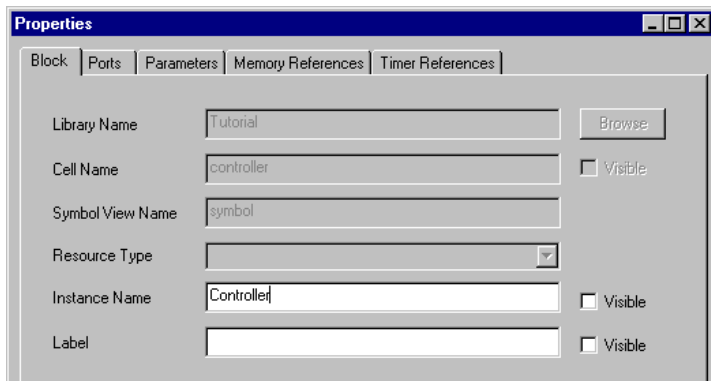
Behavior Model

Specify *controller* Parameters

1. Right click on the *controller* symbol and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following value:

Instance Name Controller

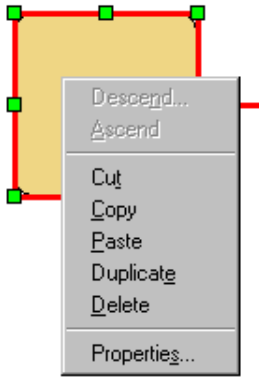
2. Type *Controller* in the Instance Name field.
3. When you have entered this value, click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling

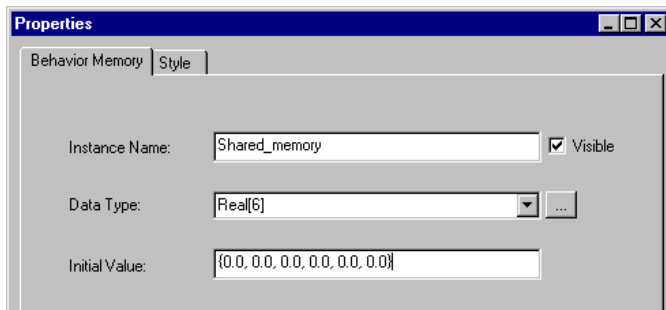
Behavior Model

Specify *Behavior Memory* Parameters

1. Right click on the *Behavior Memory* and choose *Properties* from the pop-up menu.



The Properties dialog opens.



For this block, you will enter the following values:

Instance Name	Shared_memory
Data Type	Real[6]
Initial Value	{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}

2. Enter *Shared_memory* for the Instance Name and make it visible.
3. To enter the Data Type, you can type *Real[6]* or choose from the pulldown menu.
4. In the Initial Value field, enter:
`{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}`
5. Click *OK* in the Properties dialog.
6. To save the behavior diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

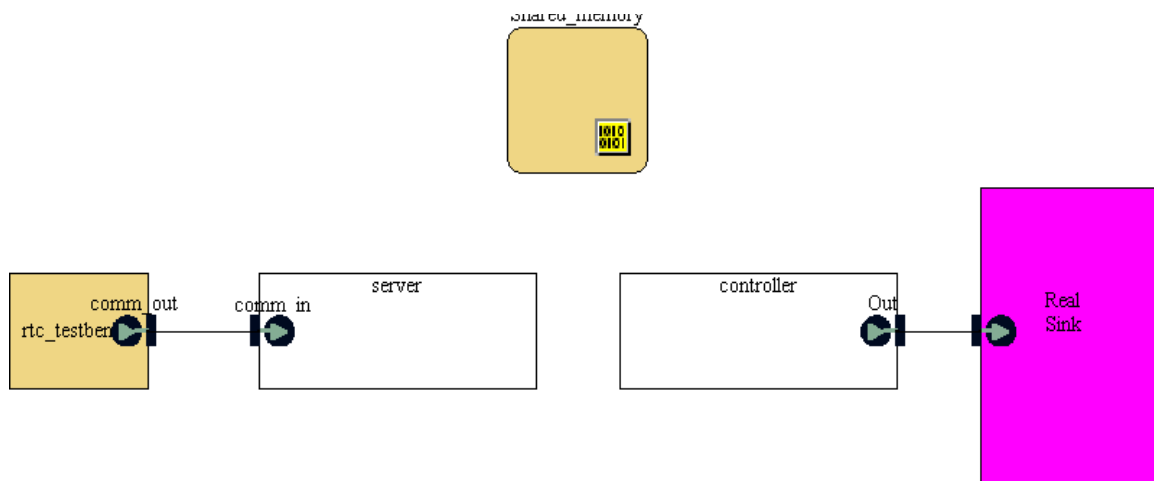
Behavior Model

Wire the Blocks

You will wire the blocks at the top level. Remember the Shared Memory is already connected to the *server* and *controller* using memory references.

1. To create a connection wire, choose *Behavior > Wire* from the main menu.
2. Click the *comm_out* output port on the *rtc_testbench*, then click the *comm_in* input port on the *server*.
3. Choose *Behavior > Wire* again.
4. Click the *Out* output port on the *controller*, then click the (unnamed) input port on the *Real Sink*.

Your diagram looks like this:



5. To save the behavior diagram, choose *File > Save* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Specify Top-Level Parameters

You can now add properties to the top level of the *rtc_behav* model. These are not block properties like you've seen before; these are properties for the top-level diagram.

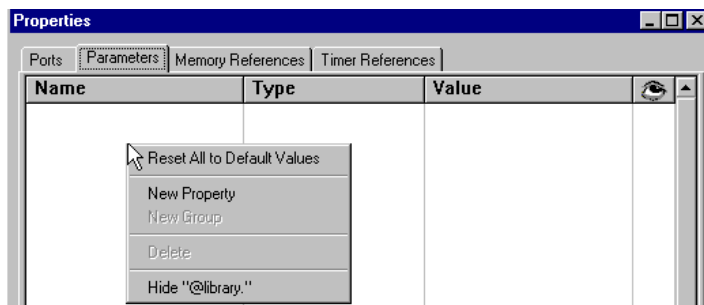
1. Right click in the white space in the behavior diagram and choose *Properties* from the pop-up menu.

The Properties dialog opens.

2. Click the *Parameters* tab in the Properties dialog.

There are no properties displayed for the *Parameters* tab.

3. Right click in the white space on the dialog and choose *New Property* from the pop-up menu.

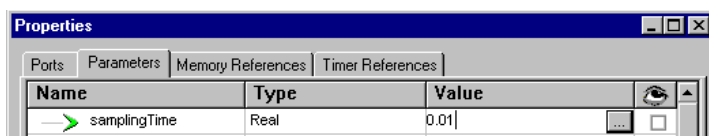


A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property:

Name	Type	Value
samplingTime	Real	0.01

4. Enter *samplingTime* for the Name.
5. Tab to the Type field and type *Real* or choose from the pulldown menu.
6. Double-click in the Value column and type *0.01*.



7. When you have entered all these values, click *OK* in the Properties dialog.

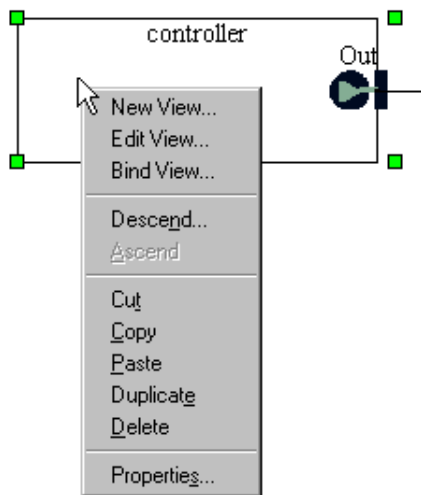
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Bind the Parameters

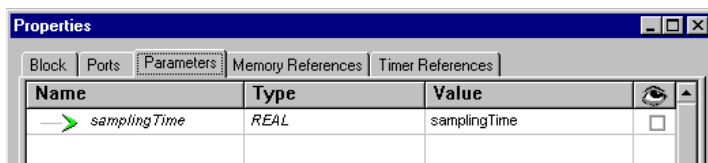
Bind the top-level *samplingTime* parameter with the controller's *samplingTime* parameter.

1. Right click the *controller* block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

2. Click the *Parameters* tab.



3. Change the Value to *samplingTime*, then click *OK* in the Properties dialog.
4. To save the behavior diagram, choose *File > Save* from the main menu.

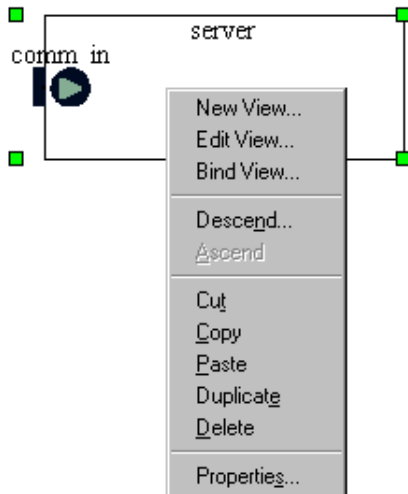
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Associate the Memory References with the Behavior Memory

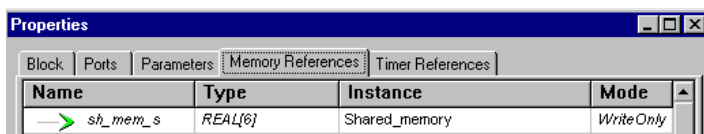
You will establish the relationship between the memory references in the *server* and *controller* and the *Shared_memory* instance at the top level of the diagram.

1. Right click the *server* block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

2. Click the *Memory References* tab.



You will enter these values for the memory reference:

Name	Type	Instance	Mode
sh_mem_s	Real[6]	Shared_memory	WriteOnly

Note: The Name, Type, and Mode values are already set. These are the values you specified at the block level for the *server* block. At the top level of this behavior diagram, you are now connecting the memory reference in the *server* block to the *Shared_memory* block.

3. Double-click in the Instance field of the *sh_mem_s* memory reference.
4. Click the down arrow and choose *Shared_memory*.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Shared_memory is the instance name you assigned to the Behavior Memory.

5. Repeat steps 1-4 for the *controller* block, entering these values for the memory reference:

Name	Type	Instance	Mode
sh_mem_c	Real[6]	Shared_memory	ReadOnly

Note: The Name, Type, and Mode values should already be correct. These are the values you specified at the block level for the *controller* block. At the top-level of this behavior diagram, you are now connecting the memory reference in the *controller* block to the *Shared_memory* block.

6. When you have entered these memory references for both the *server* and the *controller*, click *OK* in the Properties dialog.
7. To save the behavior diagram, choose *File > Save* from the main menu.

Create Functional Behavior Models

For each of the blocks that make up the *rtc_behav* diagram and the lower-level blocks that make up the *server* and *controller*, you will specify a functional model.

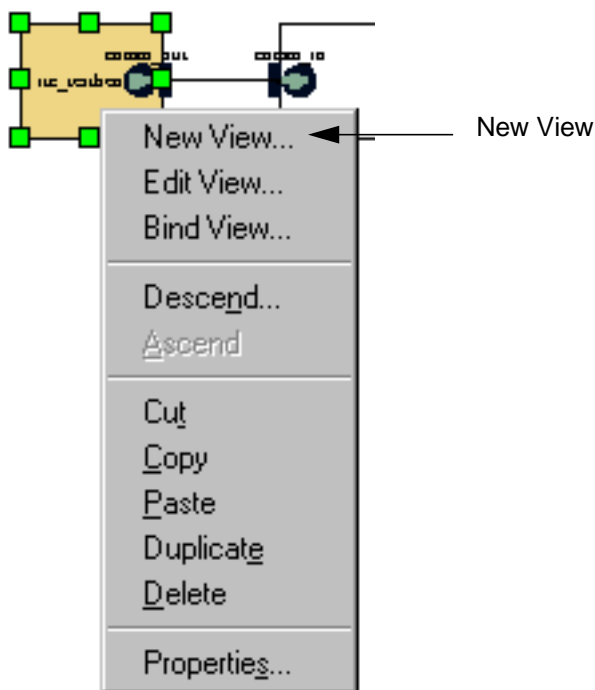
A functional model can be in one of many forms: Blackbox C++, Whitebox C, Clearbox STD, and so forth.

In this section, you will:

- [Create a Functional Model for the *rtc_testbench* Block](#) on page 66
- [Create a Functional Model for the *server_db* Block](#) on page 76
- [Create a Functional Model for the *server_std* Block](#) on page 83
- [Create a Functional Model for the *sinComb* Block](#) on page 94
- [Create a Functional Model for the *Timer* Block](#) on page 105

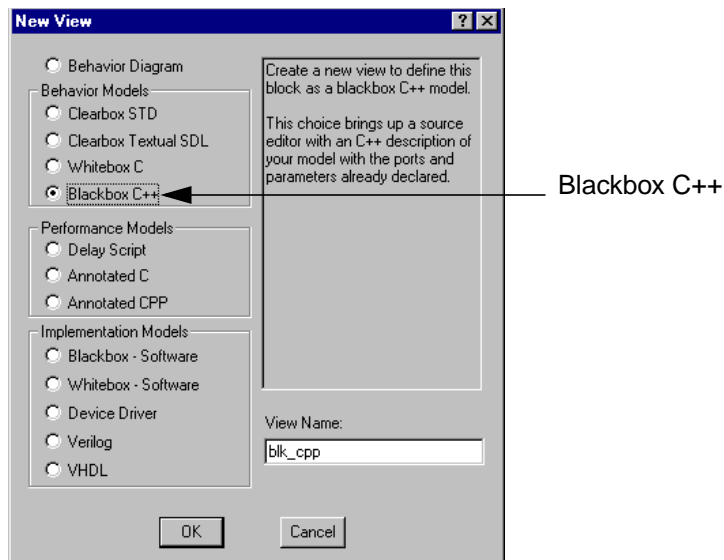
Create a Functional Model for the *rtc_testbench* Block

1. To create a new functional view for the *rtc_testbench*, right click on the *rtc_testbench* block and choose *New View* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling Behavior Model

The New View dialog opens.



2. In the Behavior Models section of the New View dialog, choose *Blackbox C++*, then click *OK*.

The View Name defaults to *blk_cpp*.

The *rtc_testbench:blk_cpp black.cpp* template file opens.

```
Tutorial.rtc_testbench:blk_cpp black.cpp
#include "black.h"

CPP_MODEL_IMPLEMENTATION::CPP_MODEL_IMPLEMENTATION(const ModuleProto &proto, InstanceIr
: CPP_MODEL_INTERFACE(proto, inst)
{
}

void CPP_MODEL_IMPLEMENTATION::Init()
{
}

// Should you chose to use Await(), uncomment the following member
// function definition and uncomment the declaration for it within the .h
// file. It will be called after the beginning of the simulation
// as soon as the block gets scheduled.

// N.B., this function should never return.
// N.B., be sure to set the functional package parameter for this
// block named "UsesAwait" to "@VCC_Types.YesNOType::Yes".
//
//void CPP_MODEL_IMPLEMENTATION::Begin()
//{
//}

// Should you chose to use Await (see above), this member function will
// never be called unless your Begin() member function calls it explicitly.
// We suggest that you comment it out, but this is not required.

// Should you chose not to use Await(), you must provide a definition
// for this member function. It will be called each time the block reacts.
//
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

3. Right click in the template file and choose *Properties* from the pop-up menu.

```
#include "black.h"

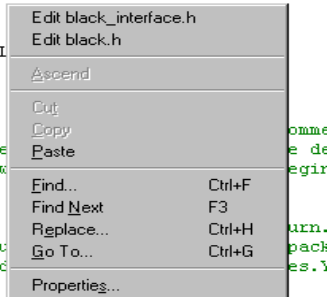
CPP_MODEL_IMPLEMENTATION::CPP_MODEL_IMPLEMENTATION()
: CPP_MODEL_INTERFACE(proto, inst)
{
}

void CPP_MODEL_IMPLEMENTATION::Begin()
{
}

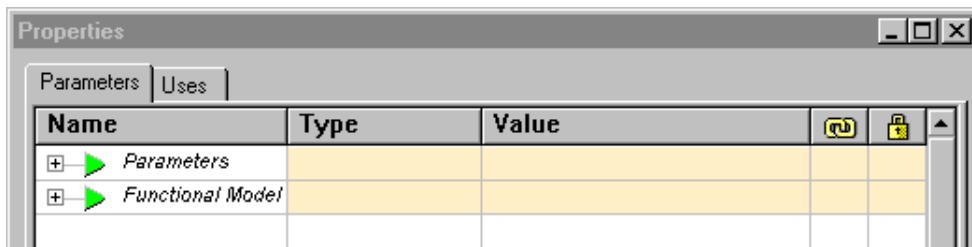
// Should you
// function de
// file. It w
// as soon as

// N.B., this
// N.B., be su
// block name

//void CPP_MODEL_IMPLEMENTATION::Begin()
//{
//}
```



The Properties dialog opens.

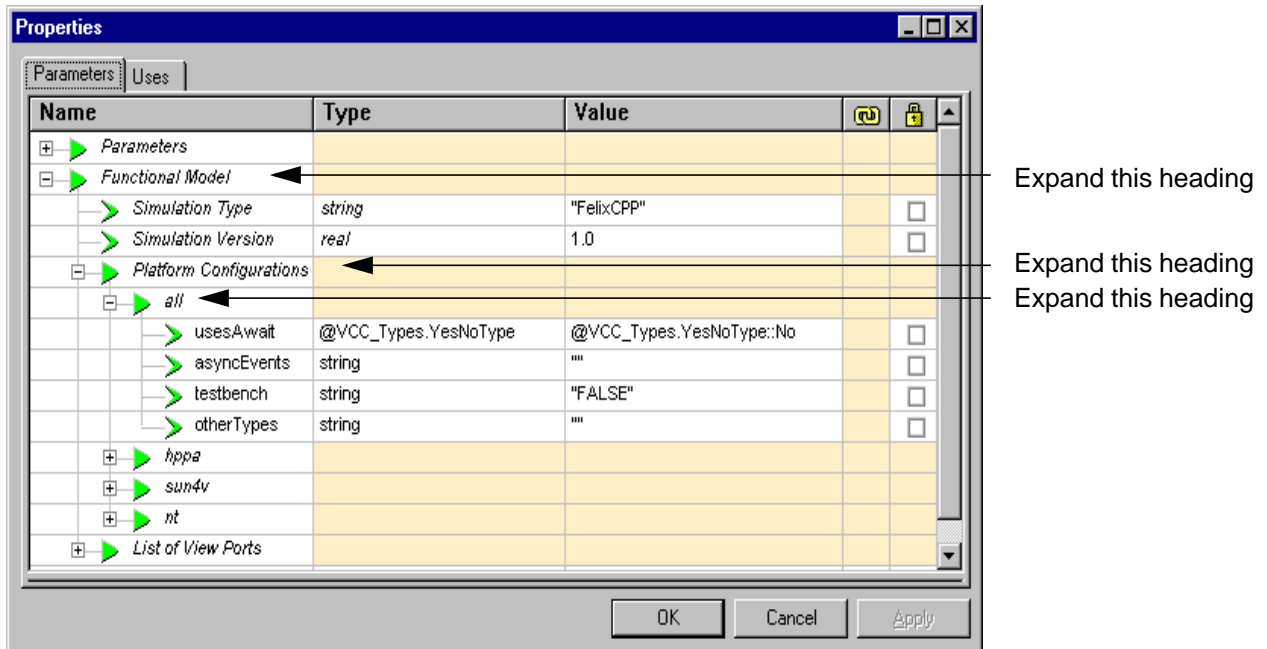


4. Click the plus sign to expand the *Functional Model* heading.
5. Click the plus sign to expand the *Platform Configurations* heading.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

- Click the plus sign to expand the *all* heading.



You will set these parameter values:

Name	Type	Value
asyncEvents	string	"AsyncEvent:integer"

- Double-click in the Value field for the *asyncEvents* parameter.
- Enter the value:

"AsyncEvent:integer"

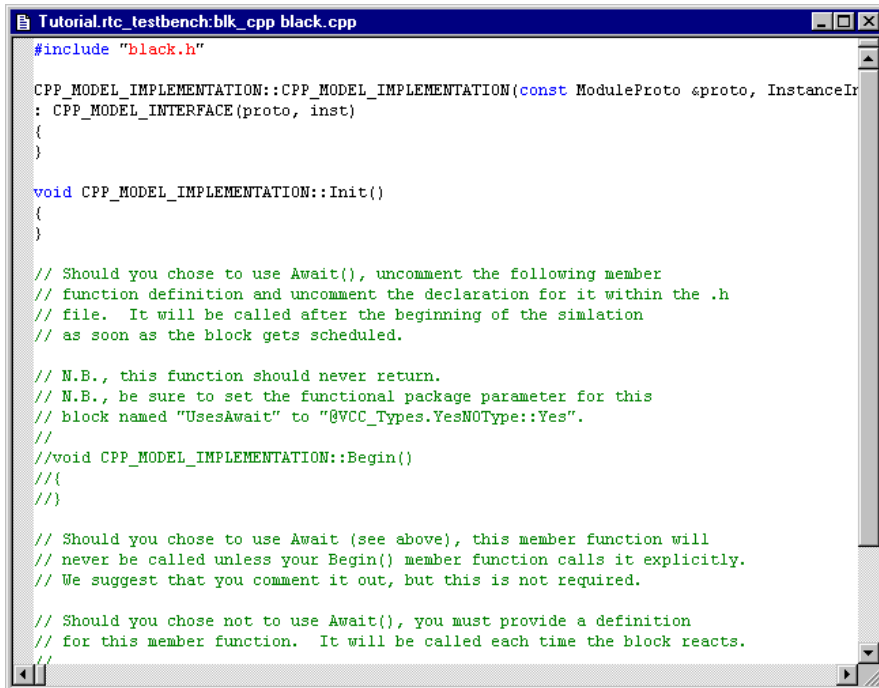
The value must include the beginning and ending quotations.

- When you have completed these changes, click *OK* in the Properties dialog.

The Properties dialog closes and you return to the *rtc_testbench:blk_cpp black.cpp* template file.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

Edit the Header File



```
#include "black.h"

CPP_MODEL_IMPLEMENTATION::CPP_MODEL_IMPLEMENTATION(const ModuleProto &proto, InstanceId inst)
: CPP_MODEL_INTERFACE(proto, inst)
{
}

void CPP_MODEL_IMPLEMENTATION::Init()
{
}

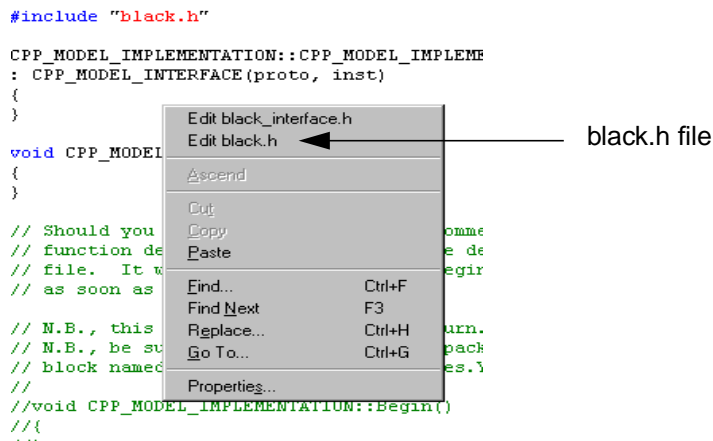
// Should you chose to use Await(), uncomment the following member
// function definition and uncomment the declaration for it within the .h
// file. It will be called after the beginning of the simlation
// as soon as the block gets scheduled.

// N.B., this function should never return.
// N.B., be sure to set the functional package parameter for this
// block named "UsesAwait" to "@VCC_Types.YesNoType::Yes".
//
//void CPP_MODEL_IMPLEMENTATION::Begin()
//{
//}

// Should you chose to use Await (see above), this member function will
// never be called unless your Begin() member function calls it explicitly.
// We suggest that you comment it out, but this is not required.

// Should you chose not to use Await(), you must provide a definition
// for this member function. It will be called each time the block reacts.
//
//
```

1. Right click in the template file and choose *Edit black.h* from the pop-up menu.



```
#include "black.h"

CPP_MODEL_IMPLEMENTATION::CPP_MODEL_IMPLEMENTATION(const ModuleProto &proto, InstanceId inst)
: CPP_MODEL_INTERFACE(proto, inst)
{
}

void CPP_MODEL_IMPLEMENTATION::Init()
{
}

// Should you chose to use Await(), uncomment the following member
// function definition and uncomment the declaration for it within the .h
// file. It will be called after the beginning of the simlation
// as soon as the block gets scheduled.

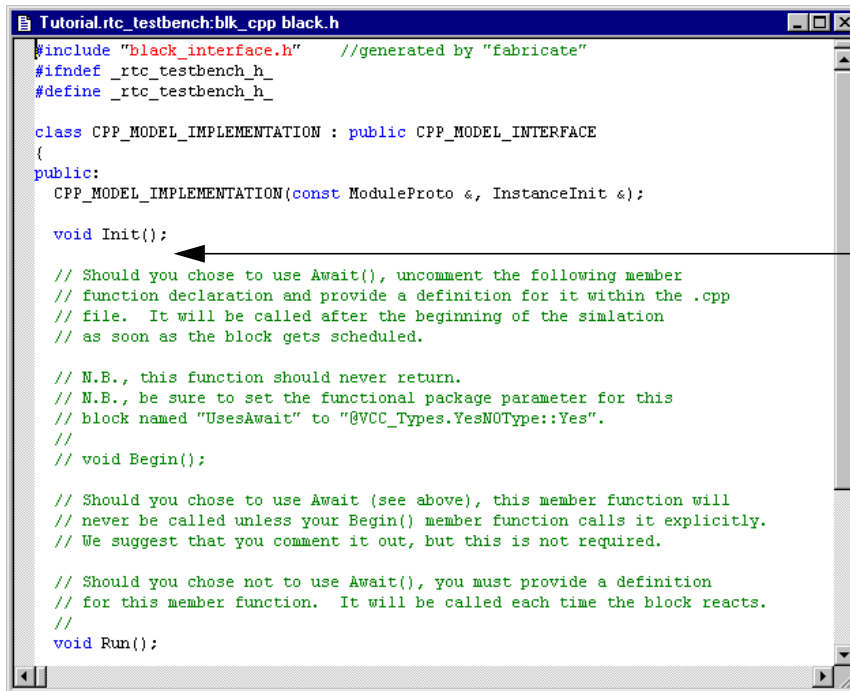
// N.B., this function should never return.
// N.B., be sure to set the functional package parameter for this
// block named "UsesAwait" to "@VCC_Types.YesNoType::Yes".
//
//void CPP_MODEL_IMPLEMENTATION::Begin()
//{
//}

// Should you chose to use Await (see above), this member function will
// never be called unless your Begin() member function calls it explicitly.
// We suggest that you comment it out, but this is not required.

// Should you chose not to use Await(), you must provide a definition
// for this member function. It will be called each time the block reacts.
//
//
```

Tutorial 2: VCC Hierarchical Modeling Behavior Model

The *rtc_testbench:blk_cpp black.h* template file opens.



2. Enter the following code after the *void Init();* statement:

```
bool started;
bool initialized;
```

The *Init* block of the *black.h* file now looks like this:

```
public:
    CPP_MODEL_IMPLEMENTATION(const ModuleProto &, InstanceInit &);

    void Init();
    bool started;
    bool initialized;
```

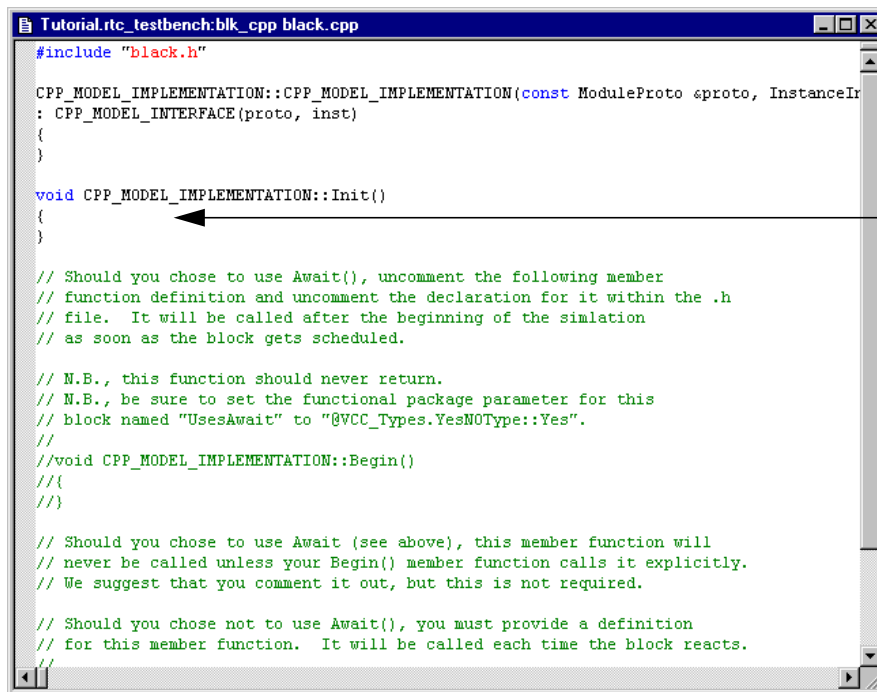
3. Make sure the *rtc_testbench:blk_cpp black.h* text file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

The *black.h* text file closes and you return to the *black.cpp* template file.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

4. Enter the following code in the `void CPP_MODEL_IMPLEMENTATION::Init()` block between the opening and closing braces.

```
// Schedule itself in one second
float deltaT = 1.0;
typeObjectInteger one = 1;
AsyncEvent.Schedule(TNow() + (float) deltaT, one);
initialized = false;
started = false;
```



Enter new code here

The *Init* block of the *black.cpp* file now looks like this:

```
void CPP_MODEL_IMPLEMENTATION::Init()
{
// Schedule itself in one second
float deltaT = 1.0;
typeObjectInteger one = 1;
AsyncEvent.Schedule(TNow() + (float) deltaT, one);
initialized = false;
started = false;
}
```


Tutorial 2: VCC Hierarchical Modeling

Behavior Model

5. Enter the following code in the *void CPP_MODEL_IMPLEMENTATION::Run()* block between the opening and closing braces.

```
if (AsyncEvent.Enabled()) {
    if (!started) {
        // Initialize the state of server to be OFF
        typeObjectSet
        OFF(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "OFF");

        comm_out.Post(OFF);
        started = true;

        // Schedule itself to set INIT state on in one more second
        float deltaT = 1.0;
        typeObjectInteger one = 1;
        AsyncEvent.Schedule(TNow() + (float) deltaT, one);
        return;
    }

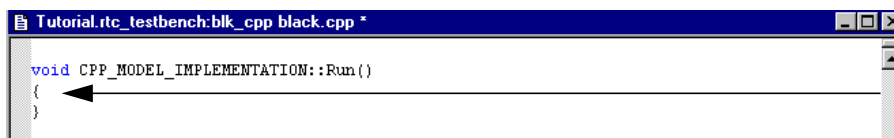
    if (!initialized) {

        // INIT
        typeObjectSet
        INIT(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "INIT");
        comm_out.Post(INIT);
        initialized = true;

        // Schedule itself to set ONLINE state on in 2 seconds
        float deltaT = 2.0;
        typeObjectInteger one = 1;
        AsyncEvent.Schedule(TNow() + (float) deltaT, one);

    } else {

        // ONLINE
        typeObjectSet
        ONLINE(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "ONLINE");
        comm_out.Post(ONLINE);
    }
}
```



Enter new code here

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The *Run* block of the *black.cpp* file now looks like this:

```
void CPP_MODEL_IMPLEMENTATION::Run()
{
    if (AsyncEvent.Enabled()) {
        if (!started) {
            // Initialize the state of server to be OFF
            typeObjectSet
            OFF(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "OFF");
            comm_out.Post(OFF);
            started = true;
            // Schedule itself to set INIT state on in one more second
            float deltaT = 1.0;
            typeObjectInteger one = 1;
            AsyncEvent.Schedule(TNow() + (float) deltaT, one);
            return;
        }
        if (!initialized) {
            // INIT
            typeObjectSet
            INIT(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "INIT");
            comm_out.Post(INIT);
            initialized = true;
            // Schedule itself to set ONLINE state on in 2 seconds
            float deltaT = 2.0;
            typeObjectInteger one = 1;
            AsyncEvent.Schedule(TNow() + (float) deltaT, one);
        } else {
            // ONLINE
            typeObjectSet
            ONLINE(typeDefinition::loadStandardType("@Tutorial2.CommandType"), "ONLINE");
            comm_out.Post(ONLINE);
        }
    }
}
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

6. Make sure the *rtc_testbench:blk_cpp black.cpp* is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

The *rtc_testbench:blk_cpp black.cpp* text file closes and you return to the *rtc_behav:behav* diagram.

Generate the Symbol

When you “generate a symbol” for the *rtc_testbench* cell, VCC creates the *interface* and *symbol* views, bundles them with the *behav* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

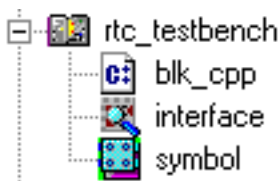
1. Click the *rtc_testbench* block to select it.
2. Choose *Behavior > Generate Symbol* from the main menu.

At the bottom of the Create window, VCC reports:

Info: Interface file was generated successfully

3. Choose *Workspace > Refresh Workspace* from the main menu.

You can see in the Project Folder that the new *rtc_testbench* cell is added to the *Tutorial2* library.



The system creates the *rtc_testbench:interface* and the *rtc_testbench:symbol* views.

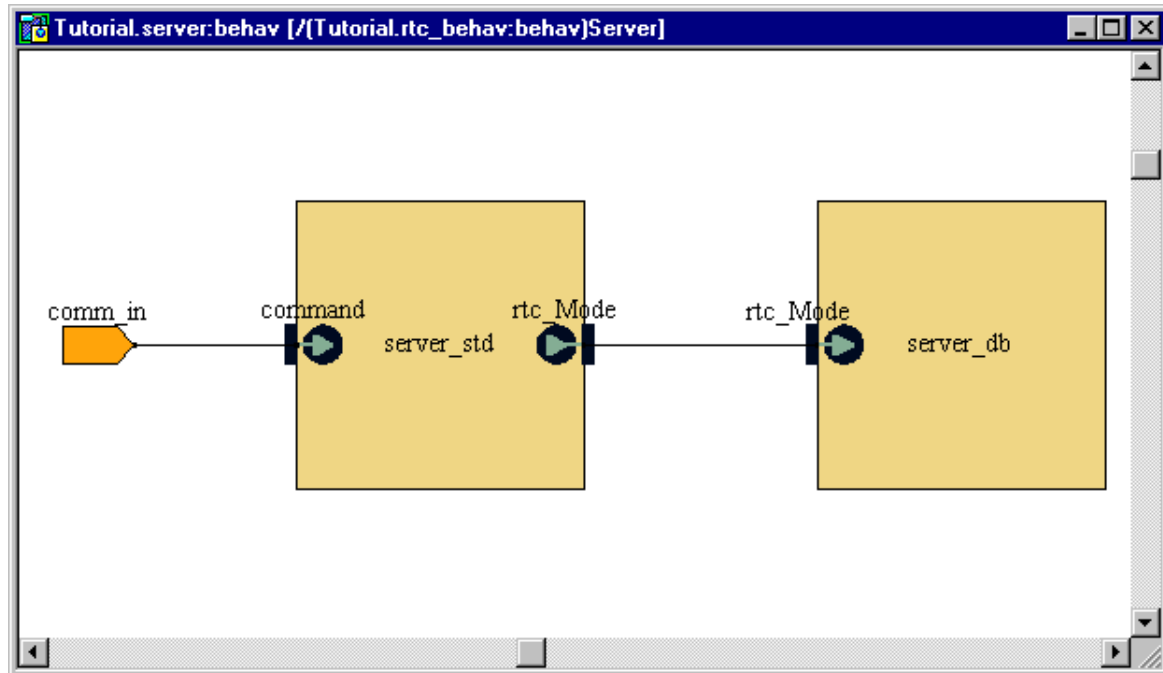
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Create a Functional Model for the *server_db* Block

1. To open the *server* behavior model, double-click on the *server* block in the *rtc_behav:behav* diagram.

The *server:behav* diagram opens.



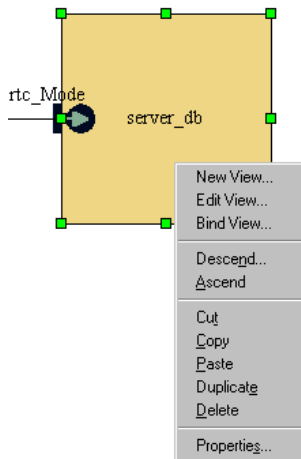
Note: The title of the window indicates the hierarchy of this model as it is instantiated in the *rtc_behav* diagram. If you had opened the *server:behav* model by double-clicking the cellview in the Project Folder, the title would not display the hierarchy within the *rtc_behav* diagram. You can use either method to open and edit low-level blocks in a diagram.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Add Parameters and a Memory Reference to the `server_db` Block

1. Right click on the `server_db` block and choose *Properties* from the pop-up menu.

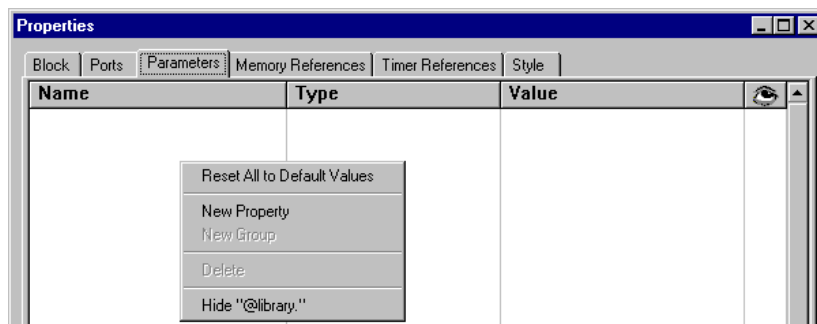


The Properties dialog opens.

2. Click the *Parameters* tab.

There are no Parameters specified, so all the fields are blank.

3. Right click in the blank area of the dialog and choose *New Property* from the pop-up menu.



A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

You will create these five new properties:

Name	Type	Value
Offset	Real	15.0
Steps	Integer	1
DeltaFr	Real	0.0
Gain	Real	5.0
StartFr	Real	1.0

4. Enter the first new property name in the Name field.
5. Tab to the Type field, click the down arrow, and choose the property type from the pulldown menu.

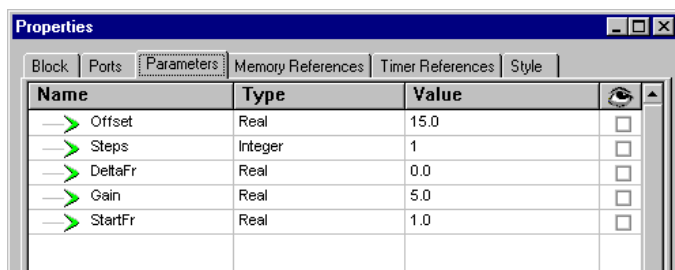


Tip

If you prefer, you can type the property types in the field; you don't have to use the pulldown menus.

6. Double-click in the Value field and enter the value for the new property.
7. Click *Apply* in the Properties dialog.
8. Repeat steps 3 through 7 for each of the new properties.

Your new properties look like this:



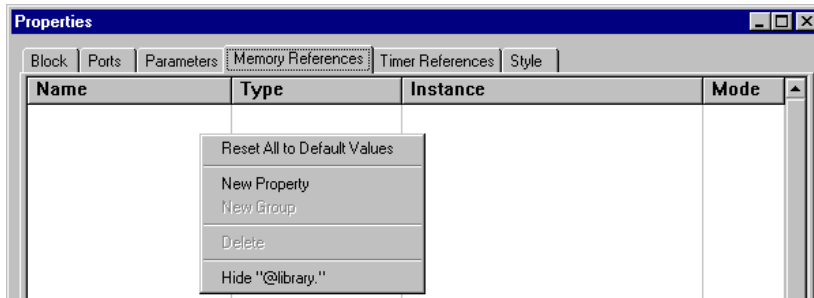
9. When you have entered the last of the new properties, click the *Memory References* tab in the Properties dialog.

There are no Memory References specified, so all the fields are blank.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

- Right click in the blank area of the dialog and choose *New Property* from the pop-up menu.



A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property:

Name	Type	Instance	Mode
sh_mem_db	Real[6]	sh_mem_s	WriteOnly

- Type *sh_mem_db* in the Name field.
- Tab to the Type field and enter *Real[6]*.

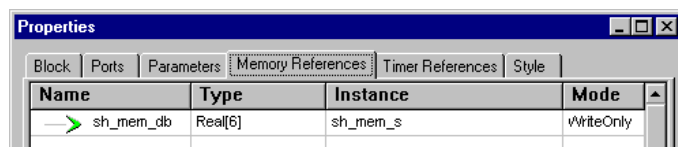


Tip

If you choose *Real* from the pulldown menu, you must type the *[6]* to complete the value.

- Tab to the Instance field and type *sh_mem_s*.
- Tab to the Mode field, click the down arrow, and choose *WriteOnly*.

Your new property looks like this:



- When you have entered all these values, click *OK* in the Properties dialog.

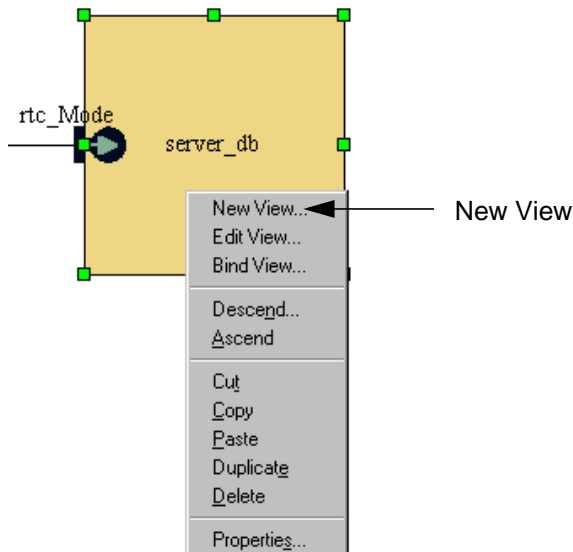
The *sh_mem_db* memory reference in the *server_db* block is now referenced to the *sh_mem_s* memory reference at the top level of the *server* block.

Tutorial 2: VCC Hierarchical Modeling

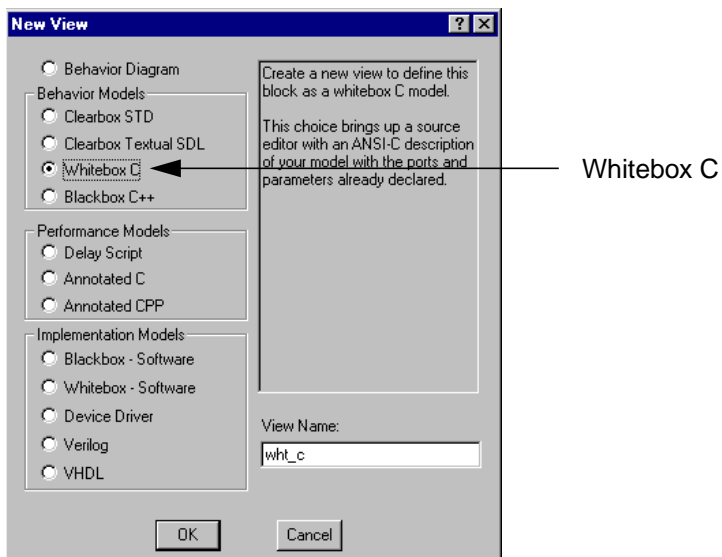
Behavior Model

Specify a Whitebox C Behavior Model for the *server_db* Block

1. Right click on the *server_db* block and choose *New View* from the pop-up menu.



The New View dialog opens.



2. In the Behavior Models section, select *Whitebox C*.

The View Name defaults to *wht_c*.

3. Click *OK* in the New View dialog.

The *server_db:wht_c white.c* template file opens.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

4. Enter the following code in the *void poin_entry_Run()* block:

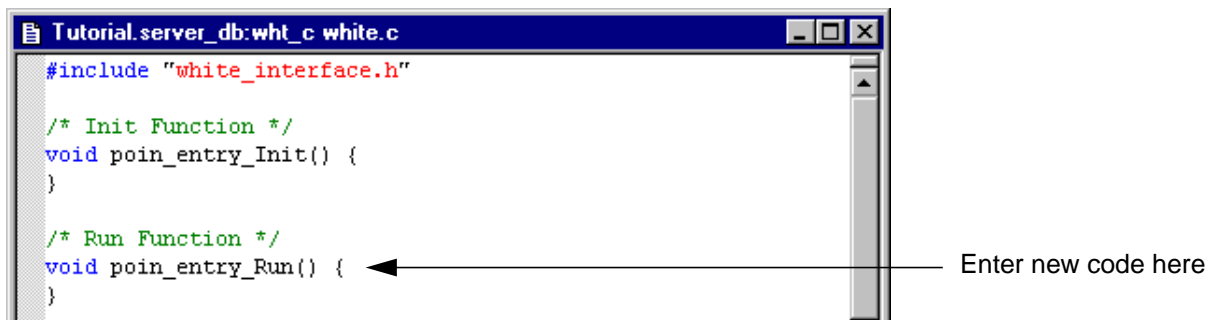
```
double rtcPar[6];

if (rtc_Mode_Enabled()) {

    rtcPar[0]= (double) rtc_Mode_Value();
    rtcPar[1]= Offset_Value();
    rtcPar[2]= (double) Steps_Value();
    rtcPar[3]= DeltaFr_Value();
    rtcPar[4]= Gain_Value();
    rtcPar[5]= StartFr_Value();

    sh_mem_db_Write(&rtcPar);

}
```



The *poin_entry_Run* block of the *white.c* file now looks like this:

```
/* Run Function */
void poin_entry_Run() {

    double rtcPar[6];

    if (rtc_Mode_Enabled()) {

        rtcPar[0]= (double) rtc_Mode_Value();
        rtcPar[1]= Offset_Value();
        rtcPar[2]= (double) Steps_Value();
        rtcPar[3]= DeltaFr_Value();
        rtcPar[4]= Gain_Value();
        rtcPar[5]= StartFr_Value();

        sh_mem_db_Write(&rtcPar);

    }

}
```

5. Make sure the *server_db:wht_c white.c* is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Generate the Symbol

When you “generate a symbol” for the *server_db*, VCC creates the *interface* and *symbol* views, bundles them with the *wht_c* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

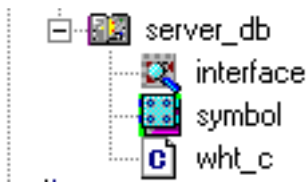
1. Click the *server_db* block to select it.
2. Choose *Behavior > Generate Symbol* from the main menu.

At the bottom of the Create window, VCC reports:

Info: Interface file was generated successfully.

3. Choose *Workspace > Refresh Workspace* from the main menu.

You can see in the Project Folder that the new *server_db* cell is added to the *Tutorial2* library.



The system creates the *server_db:interface* view and a *server_db:symbol* view.

Create a Functional Model for the *server_std* Block

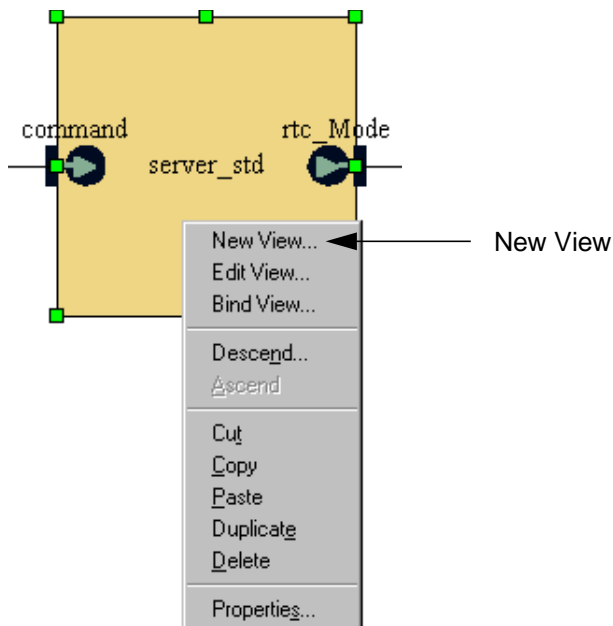
You will now create a Clearbox state transition diagram (STD) model for the *server_std* block. A state transition diagram consists of a set of states, a set of input events, a set of output events, and a state transition function.

This model implements the following behavior:

- While the server is in the *OFF* state, the controller output is zero.
- While the server is in the *INIT* state, the controller output is equal to the offset parameter value.
- While the server is in the *ONLINE* state, the controller output is the offset plus a sine waves convolution.

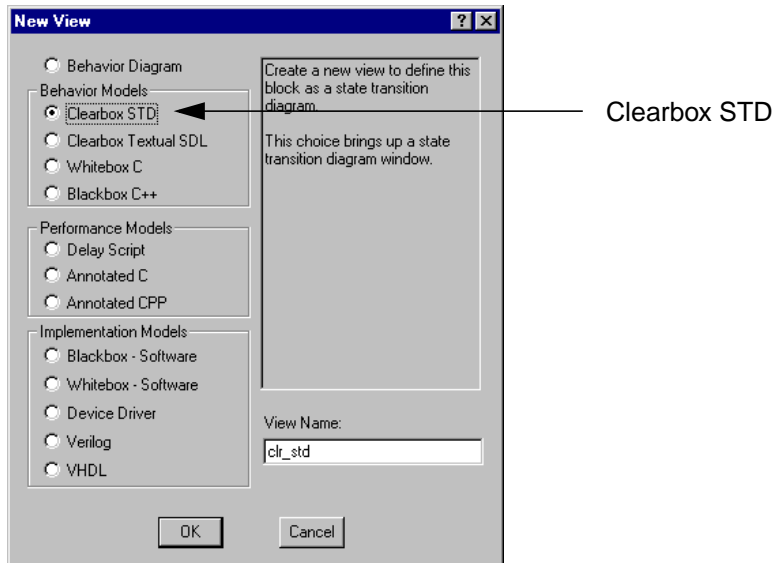
Create a New STD Diagram

1. Right click on the *server_std* block and choose *New View* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling Behavior Model

The New View dialog opens.

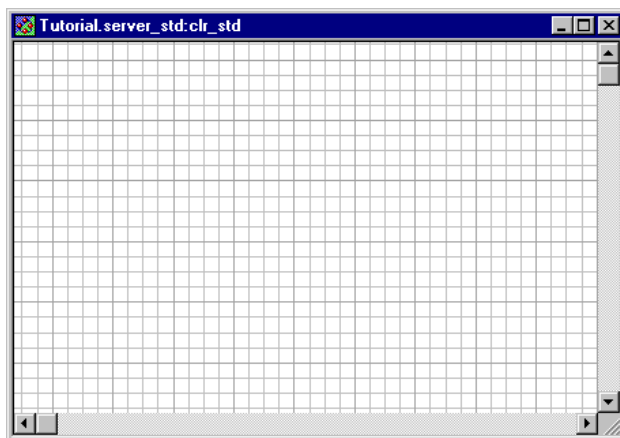


2. In the Behavior Models section, select *Clearbox STD*.

The View Name defaults to *clr_std*.

3. Click *OK* in the New View dialog.

A new empty window for the *server_std:clr_std* view opens.



Tutorial 2: VCC Hierarchical Modeling Behavior Model

Place States in the Diagram

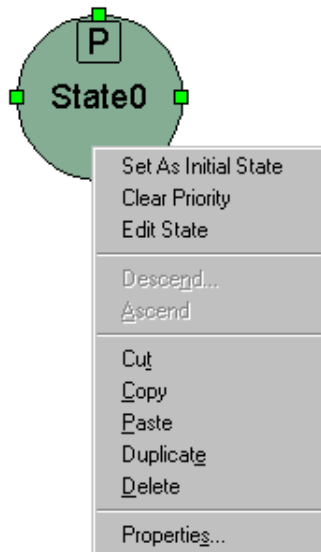
1. Choose *State Transition > Single State* from the main menu.

When you move your cursor back into the window, you see a *state* symbol attached to it.

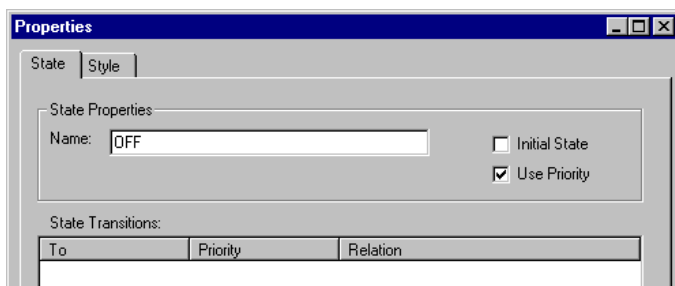
2. Click to place this state in your diagram.
3. Repeats steps 1 and 2 until you have three states in your diagram:



4. Right click on *State0* and choose *Properties* from the pop-up menu.



The Properties dialog opens.

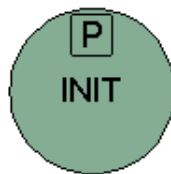
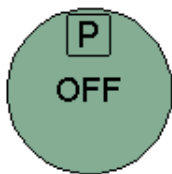


Tutorial 2: VCC Hierarchical Modeling

Behavior Model

5. Change the Name to *OFF* and click *OK*.
6. Right click on *State1* and choose *Properties* from the pop-up menu.
7. Change the Name to *INIT* and click *OK*.
8. Right click on *State2* and choose *Properties* from the pop-up menu.
9. Change the Name to *ONLINE* and click *OK*.

The states now look like this:



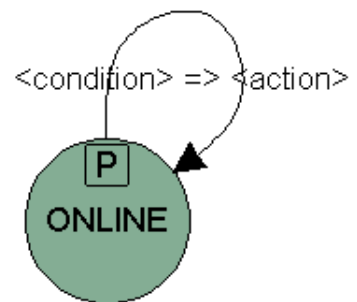
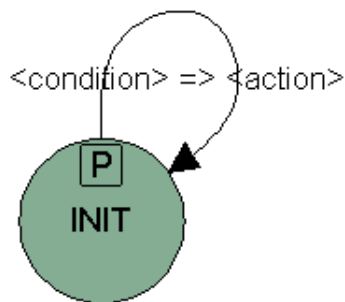
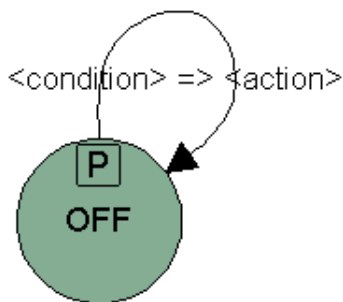
Connect the States

You will now create the connections between each of the states.

- Each of the states connects to itself.
 - The OFF state connects to the INIT state.
 - The INIT states connects to the ONLINE and the OFF states.
 - The ONLINE states connects to the INIT and the OFF states.
1. To connect each state to itself, choose *State Transition* > *Transition* from the main menu, then double-click on the state.

Repeat this step for each of the three states.

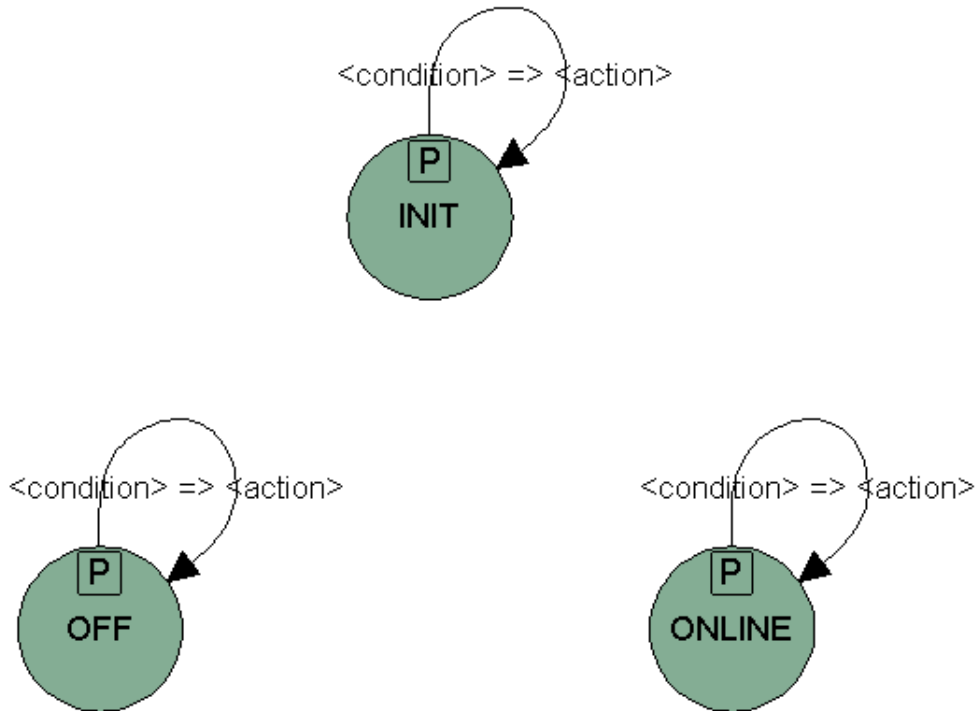
The states now look like this:



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

2. To make it easier to represent the connections between the states, rearrange them like this:



3. To create the following connections, choose *State Transition > Transition* from the main menu, click on the “from” state, then click on the “to” state.
- ☐ From the OFF state to the INIT state
 - ☐ From the INIT state to the OFF state
 - ☐ From the INIT state to the ONLINE state
 - ☐ From the ONLINE state to the OFF state
 - ☐ From the ONLINE state to the INIT state



Tip

To create bends in the transition lines (as you see on the following page), click once at the beginning of the transition, click to place ‘elbows’ in the line, then click at the terminus. You can later move the bend to get the arc you want. Creating these bends prevents the transitions from being straight lines directly on top of each other.

Tutorial 2: VCC Hierarchical Modeling

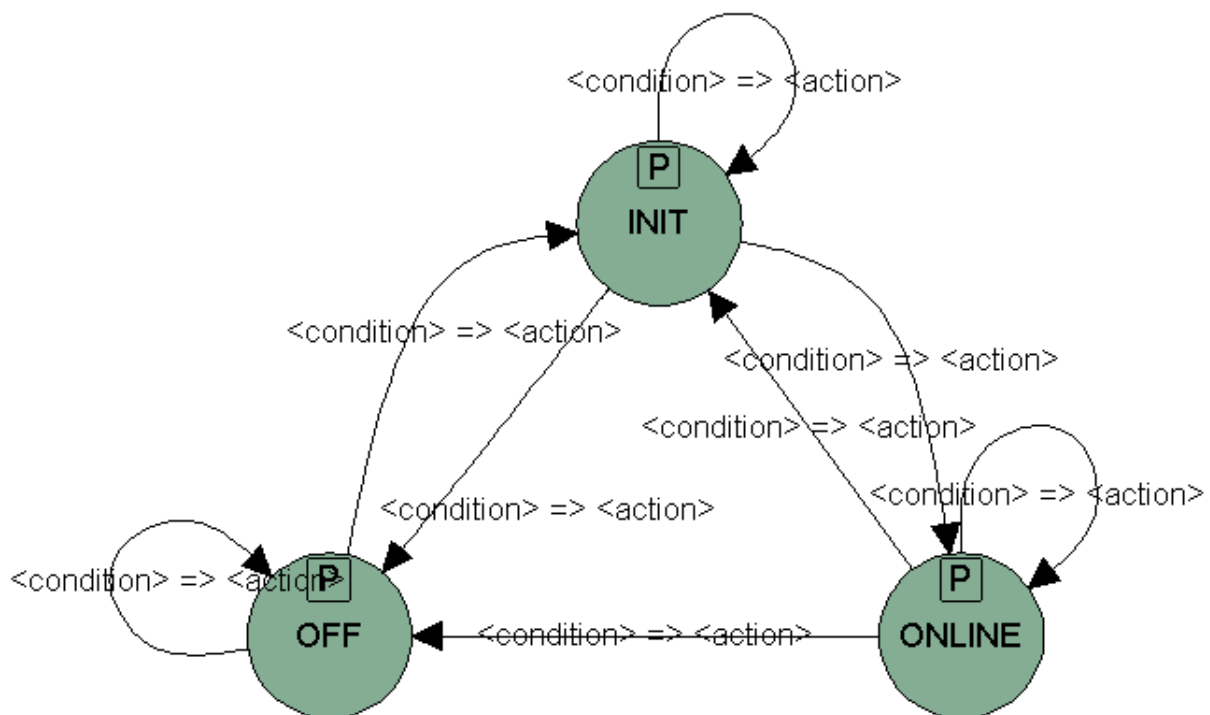
Behavior Model



Tip

To confirm that your transitions are connected to the states as you place them, look for red markers at the ends of the transition lines. A red marker indicates that the transition line is connected. After all the transitions are placed, you can drag a state to confirm that all the transitions are connected. If a transition does not move with the state, it is not connected.

You can arrange the state transition connections however you like. Your diagram will look something like this:



You will define values for all these state transition conditions and actions later.

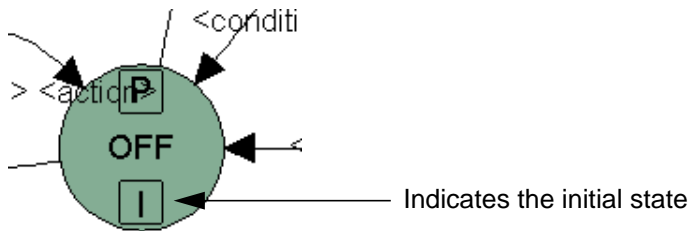
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

Set the Initial State

1. Click the OFF state to select it.
2. Choose *State Transition > Initial State* from the main menu.

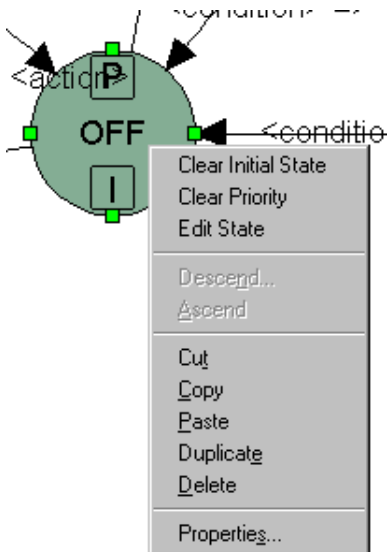
The OFF state is now set as the initial state. The state symbol changes to indicate this.



Define the State Transition Conditions and Actions

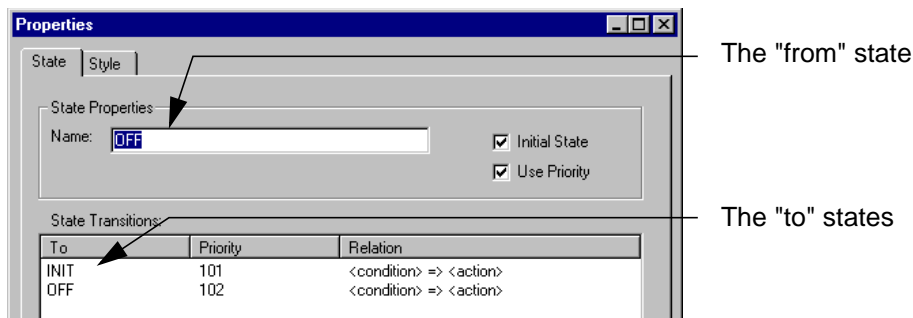
For each connection you created in the “Connect the States” section, you will now define the condition and action.

1. Right click on the OFF state and choose *Properties* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling Behavior Model

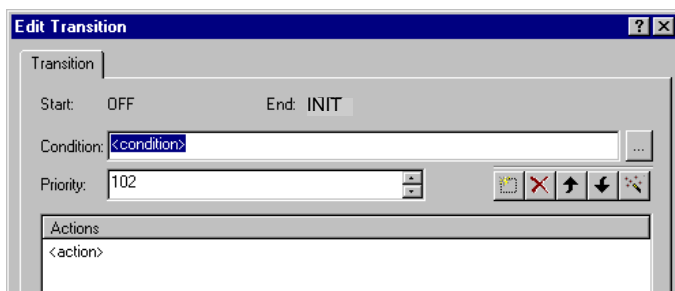
The Properties dialog opens.



The State Transitions list defines the connections “from” the OFF state “to” other states. For the OFF state, you created a connection from the OFF state to itself and a connection from the OFF state to the INIT state.

2. In the State Transitions list, double-click the INIT entry.

The Edit Transition dialog opens.



This defines the connection from the OFF state to the INIT state.

3. Change the Condition definition to

```
Enabled(command) && (command != @Tutorial2.CommandType::INIT)
```

↑
The "to" state



Tip

You can copy and paste the Condition definition text and reuse it in the following Condition definitions. Only the End or “to” state (for this condition, INIT) in the Condition definition changes each time.

4. Double-click the Action definition and change it to

```
rtc_Mode = 1
```

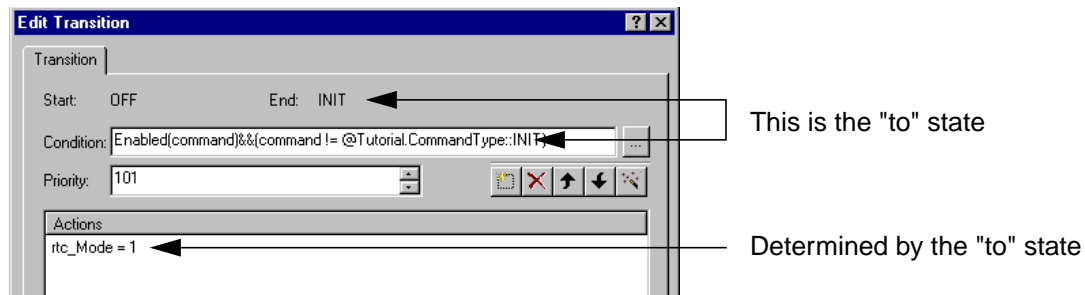
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

In each entry, the value for *rtc_Mode* is determined by the “to” state:

- ❑ When the “to” state is OFF, the *rtc_Mode* is 0.
- ❑ When the “to” state is INIT, the *rtc_Mode* is 1.
- ❑ When the “to” state is ONLINE, the *rtc_Mode* is 2.

The Condition and Action definitions look like this:



5. Click *OK* in the Edit Transition dialog.

The Edit Transition dialog closes and you return to the Properties dialog.

6. In the State Transitions list, double-click the OFF entry.

The Edit Transition dialog opens.

7. Change the Condition to

```
Enabled(command) && (command != @Tutorial2.CommandType::OFF)
```

8. Change the Action to

```
rtc_Mode = 0
```

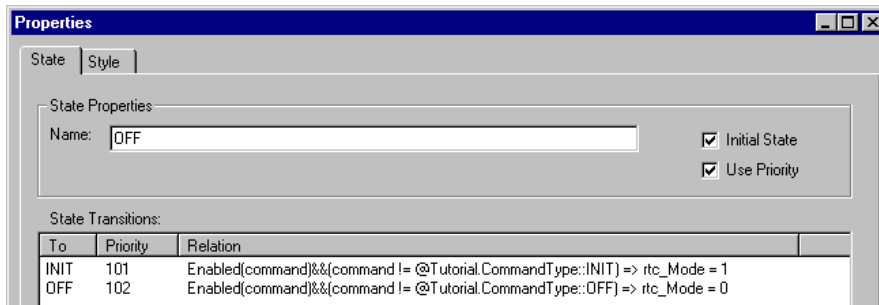
This defines the connection from the OFF state to itself.

9. Click *OK* in the Edit Transition dialog.

The Edit Transition dialog closes and you return to the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

The properties for the OFF state look like this:



The Properties dialog for the OFF state shows the following configuration:

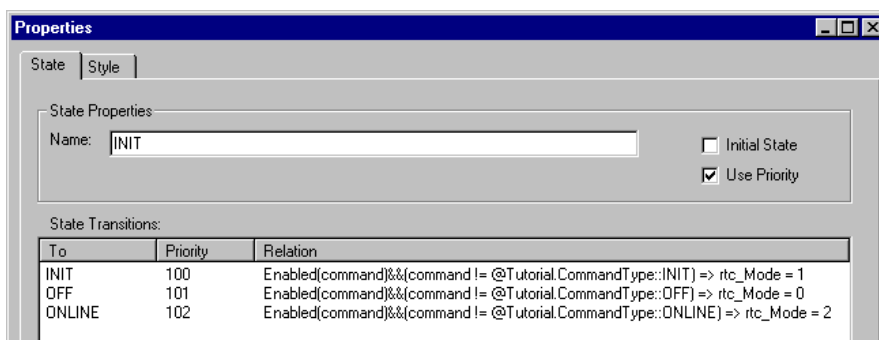
- State Properties:**
 - Name: OFF
 - ☒ Initial State
 - ☒ Use Priority
- State Transitions:**

To	Priority	Relation
INIT	101	Enabled(command)&&(command != @Tutorial.CommandType::INIT) => rtc_Mode = 1
OFF	102	Enabled(command)&&(command != @Tutorial.CommandType::OFF) => rtc_Mode = 0

10. When your Conditions and Actions are defined, click **OK** in the Properties dialog.

11. Repeat steps 1 through 10 for the INIT and ONLINE states.

□ The properties for the INIT state look like this:

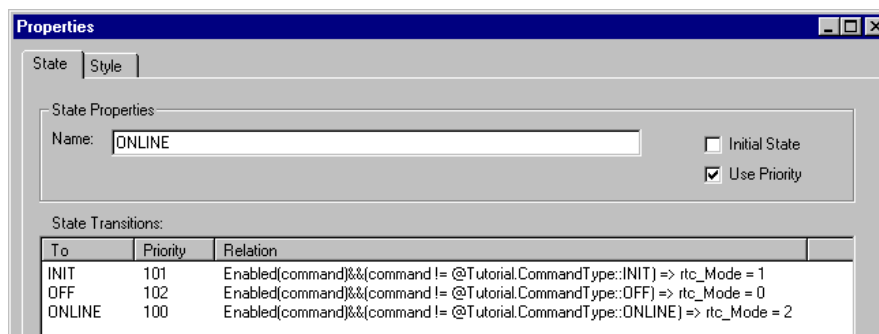


The Properties dialog for the INIT state shows the following configuration:

- State Properties:**
 - Name: INIT
 - ☐ Initial State
 - ☒ Use Priority
- State Transitions:**

To	Priority	Relation
INIT	100	Enabled(command)&&(command != @Tutorial.CommandType::INIT) => rtc_Mode = 1
OFF	101	Enabled(command)&&(command != @Tutorial.CommandType::OFF) => rtc_Mode = 0
ONLINE	102	Enabled(command)&&(command != @Tutorial.CommandType::ONLINE) => rtc_Mode = 2

□ The properties for the ONLINE state look like this:



The Properties dialog for the ONLINE state shows the following configuration:

- State Properties:**
 - Name: ONLINE
 - ☐ Initial State
 - ☒ Use Priority
- State Transitions:**

To	Priority	Relation
INIT	101	Enabled(command)&&(command != @Tutorial.CommandType::INIT) => rtc_Mode = 1
OFF	102	Enabled(command)&&(command != @Tutorial.CommandType::OFF) => rtc_Mode = 0
ONLINE	100	Enabled(command)&&(command != @Tutorial.CommandType::ONLINE) => rtc_Mode = 2



Tip

When you have defined all your connections, your state transition diagram is probably impossible to read. If you want to read the text, click and drag any of the states to separate them.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

12. When you have defined all the connections for the state transition diagram, make sure *server_std:clr_std* is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

The state transition diagram closes and you return to the *server:behav* diagram.

Generate the Symbol

When you “generate a symbol” for the *server_std*, VCC creates the *interface* and *symbol* views, bundles them with the *clr_std* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

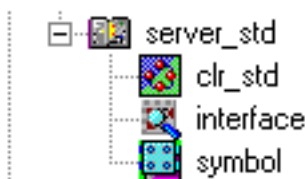
1. Click the *server_std* block to select it.
2. Choose *Behavior > Generate Symbol* from the main menu.

At the bottom of the Create window, VCC reports:

Info: Interface file was generated successfully

3. Choose *Workspace > Refresh Workspace* from the main menu.

You can see in the Project Folder that the new *server_std* cell is added to the *Tutorial2* library.



The system creates the *server_std:interface* view and the *server_std:symbol* view.

4. Make sure the *server:behav* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

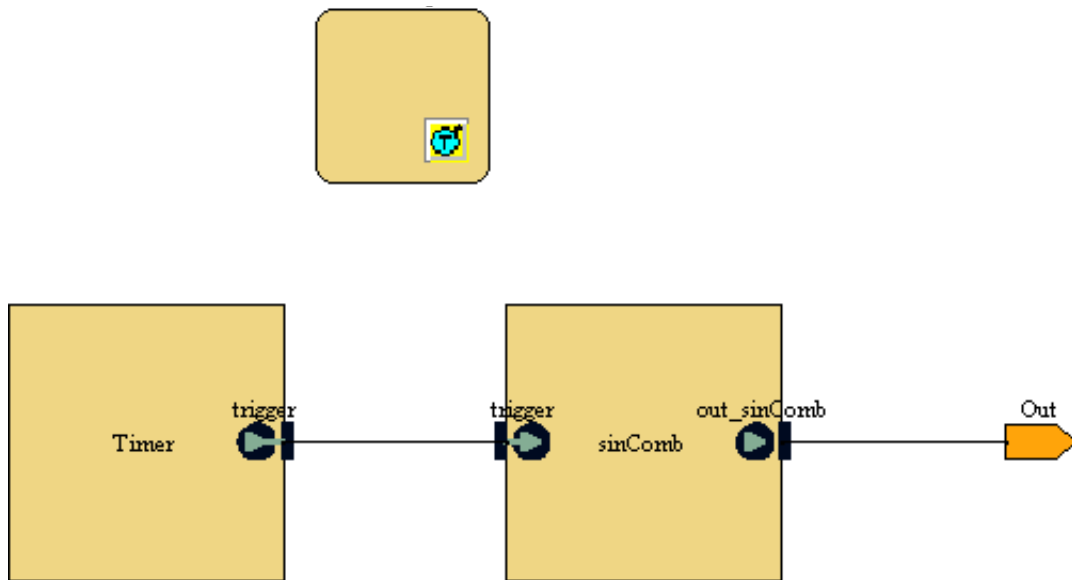
Tutorial 2: VCC Hierarchical Modeling Behavior Model

Create a Functional Model for the *sinComb* Block

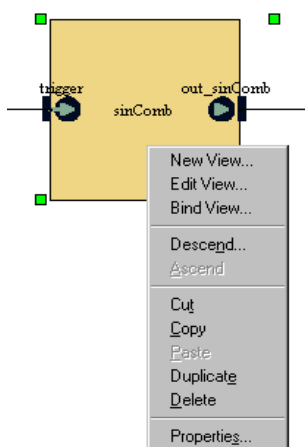
The *sinComb* block is one of the components of the *controller* block.

1. To open the *controller* block, double-click on the *controller* block in the *rtc_behav:behav* diagram.

The *controller:behav* cellview opens.



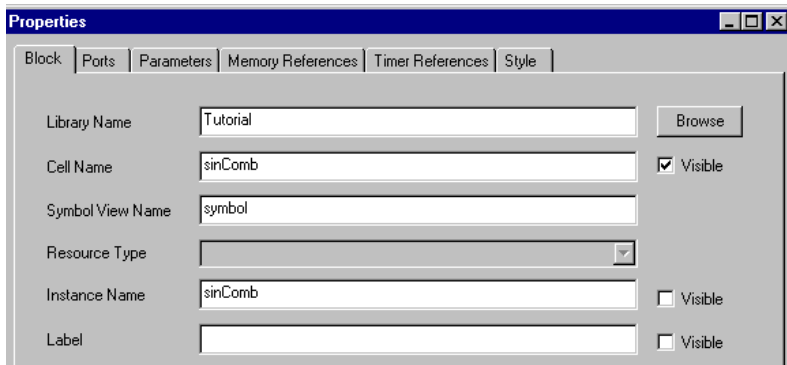
2. Right click the *sinComb* block and choose *Properties* from the pop-up menu.



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

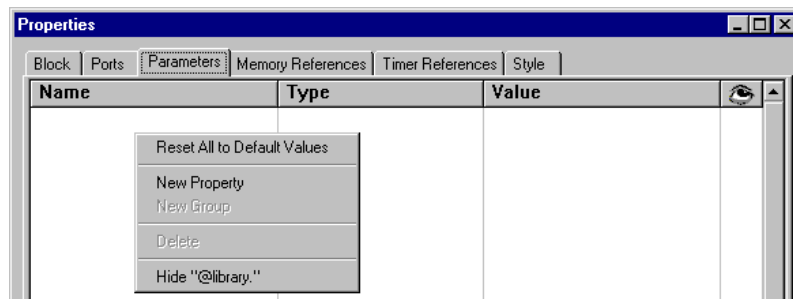
The Properties dialog opens.



3. Click the *Parameters* tab.

There are no Parameters specified, so all the fields are blank.

4. Right click in the blank area of the dialog and choose *New Property* from the pop-up menu.



A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property:

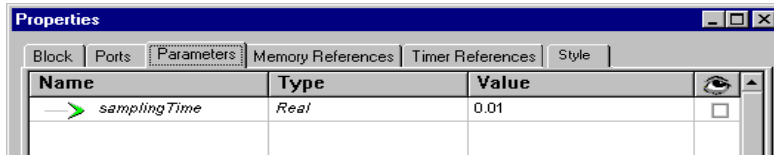
Name	Type	Value
samplingTime	Real	0.01

5. In the Name field, enter *samplingTime*.
6. Tab to the Type field and enter *Real*.
7. Tab to the Value field and enter *0.01*.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The new property looks like this:



8. Click *Apply*.

9. Click the *Memory References* tab.

There are no Memory References specified, so all the fields are blank.

10. Right click the blank area of the dialog and choose *New Property* from the pop-up menu.

You will enter this new property:

Name	Type	Instance	Mode
sh_mem_sc	Real[6]	sh_mem_c	ReadOnly

11. In the Name field, enter *sh_mem_sc*.

12. Tab to the Type field and enter *Real[6]*.



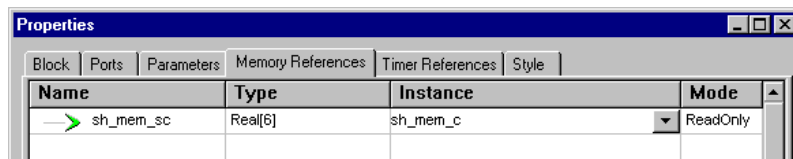
Tip

If you choose *Real* from the pulldown menu, you must type the *[6]* to complete the value.

13. Tab to the Instance field and enter *sh_mem_c*.

14. Tab to the Mode field, click the down arrow, and choose *ReadOnly* from the pulldown menu.

The *sh_mem_sc* property looks like this:



Tutorial 2: VCC Hierarchical Modeling

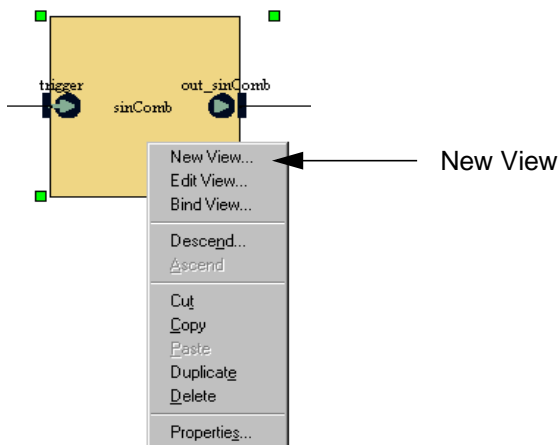
Behavior Model

15. When you have defined these new properties, click *OK* in the Properties dialog.

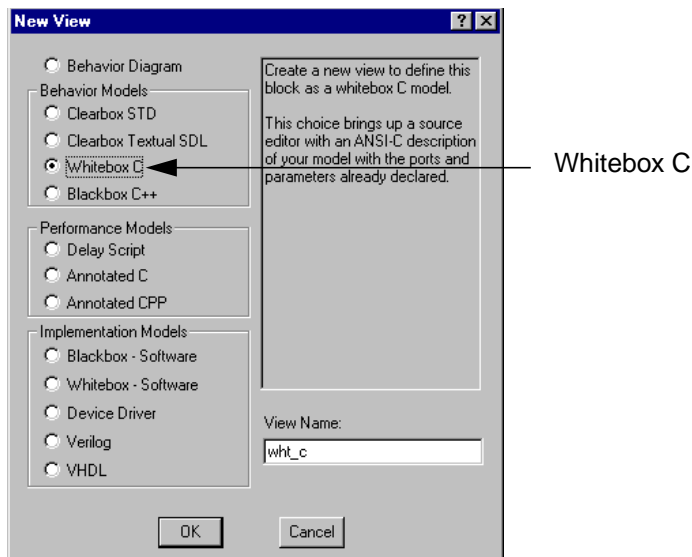
The *sh_mem_sc* memory reference in the *sinComb* block is now referenced to the *sh_mem_c* memory reference at the top level of the *controller* block.

Specify an Implementation Model for the *sinComb* Block

1. Right click on the *sinComb* block and choose *New View* from the pop-up menu.



The New View dialog opens.



2. In the Behavior Models section, select *Whitebox C*.

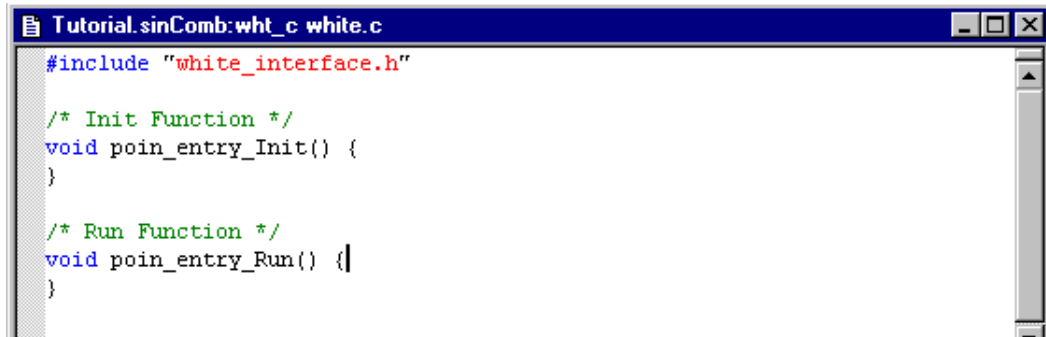
The View Name defaults to *wht_c*.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

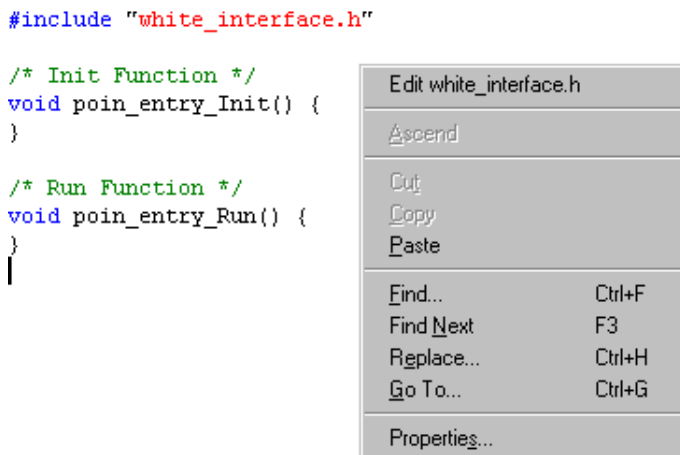
3. Click *OK* in the New View dialog.

The *sinComb:wht_c white.c* template file opens.

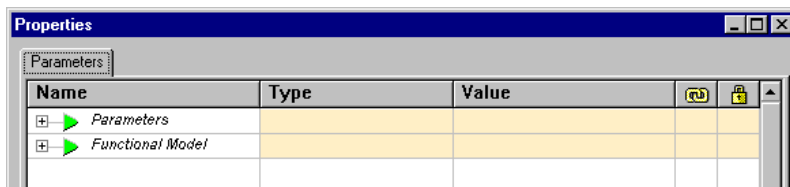


Because you are using a Whitebox C model that calls Standard C library methods, you must declare your standard C library in the Properties dialog.

4. Right click anywhere in the text window and choose Properties from the pop-up menu.



The Properties dialog opens.



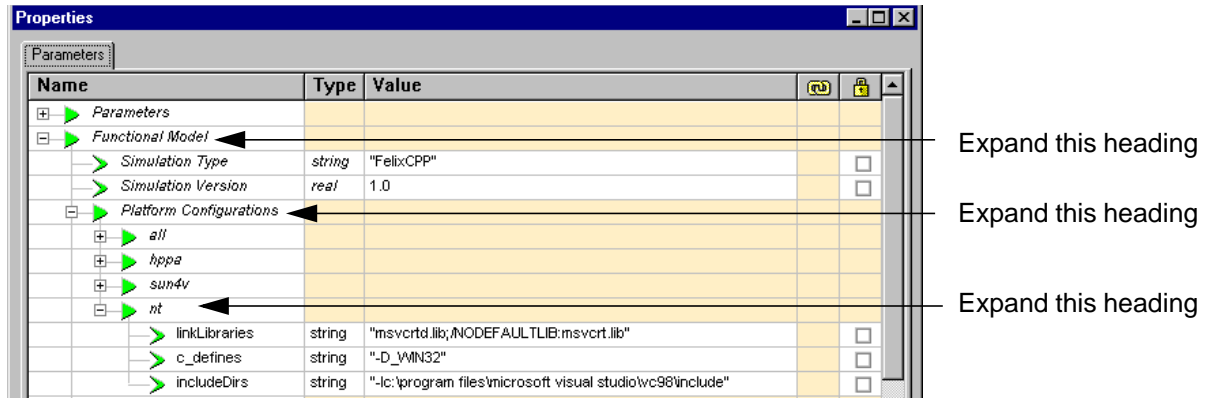
5. Click the plus sign to expand the *Functional Model* heading.
6. Click the plus sign to expand the *Platform Configurations* heading.

If you are using an HP-UX or Solaris platform, skip to the “For HP-UX or Solaris Users Only” section on page 108.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

- Click the plus sign to expand the *nt* heading.



- Double-click in the Value field for the *linkLibraries* parameter.
The following values must include the beginning and ending quotations.

- Enter the value

"msvcrt.lib; /NODEFAULTLIB:msvcrt.lib"

- Double-click in the Value field for the *c_defines* parameter.

- Enter the value

"-D_WIN32"

- Double-click in the Value field for the *includeDirs* parameter.

- Enter the path to *devstudio* in your Microsoft Visual C++ installation, for example:

"-Ic:\program files\microsoft visual studio\vc98\include"

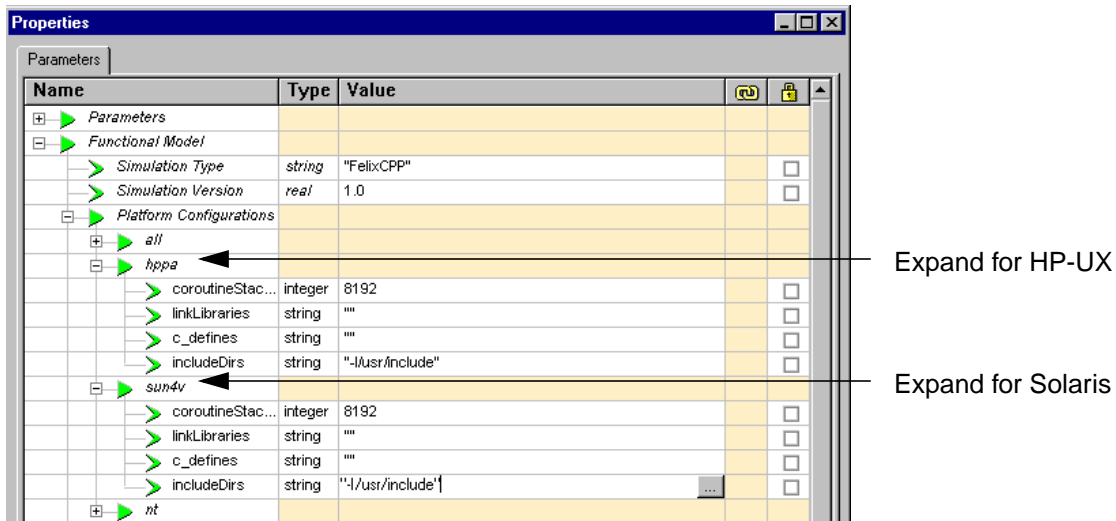
- When you have entered all these values, click *OK* to close the Properties dialog.

The Properties dialog closes and you return to the *sinComb:wht_c white.c* template file.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

For HP-UX or Solaris Users Only

1. Click the plus sign to expand the *hppa* or *sun4v* heading.



2. Double-click in the Value field for the *includeDirs* parameter.

3. Enter the value

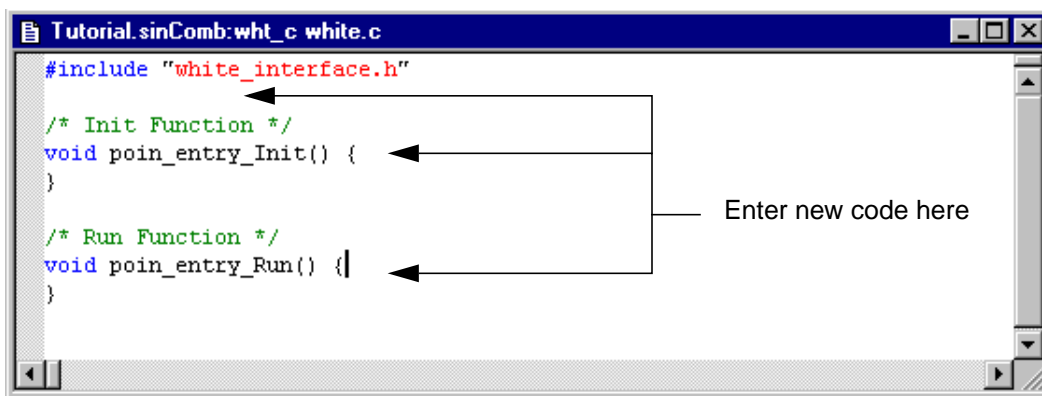
"-I/usr/include"

This value must include the beginning and ending quotations.

4. Click OK to close the Properties dialog.

The Properties dialog closes and you return to the *sinComb:wht_c white.c* template file.

Edit the Whitebox C Template File



Tutorial 2: VCC Hierarchical Modeling

Behavior Model

1. Enter the following lines of code after the `#include "white_interface.h"` statement:

```
#include <math.h>
#include <stdio.h>
FILE *f;
```

The `#include` section in the code in the `white.c` file now looks like this:

```
#include "white_interface.h"

#include <math.h>
#include <stdio.h>

FILE *f;
```

2. Enter the following lines of code in the `poin_entry_Init()` block between the opening and closing braces.

```
f = fopen("debug_out.txt", "w");
fprintf(f, "Debugging data \n\n\n\n");
```

The `poin_entry_Init()` code in the `white.c` file now looks like this:

```
/* Init Function */
void poin_entry_Init() {
    f = fopen("debug_out.txt", "w");
    fprintf(f, "Debugging data \n\n\n\n");
}
```

3. Enter the following lines of code in the `poin_entry_Run()` block between the opening and closing braces.

```
double (*rtcPar)[6];
double Offset, DeltaFr, StartFr, Gain, pOut, output, samplingTime;
int i, rtc_Mode, Steps;
static double nCount;

if (trigger_Enabled()) {

    rtcPar = sh_mem_sc_Read();
    rtc_Mode = (int) (*rtcPar)[0];
    Offset = (*rtcPar)[1];
    Steps = (int) (*rtcPar)[2];
    DeltaFr = (*rtcPar)[3];
    Gain = (*rtcPar)[4];
    StartFr = (*rtcPar)[5];
    samplingTime = samplingTime_Value();
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

```
/* Debugging info into output file */
fprintf (f, "rtc_Mode %d ", rtc_Mode);
fprintf (f, "Offset %f ", Offset);
fprintf (f, "Gain %f ", Gain);
fprintf (f, "samplingTime %f \n", samplingTime);

switch (rtc_Mode)
{
    case 0:
        out_sinComb_Post(0.0);
        nCount=1;
        break;

    case 1:
        out_sinComb_Post(Offset);
        nCount=1;
        break;

    case 2:
        pOut=0;
        for(i=0;i<Steps;i++)
            pOut+=(double)sin(6.283*(StartFr+i*DeltaFr)\
                *samplingTime*nCount);
        output = Offset + Gain * pOut;
        nCount++;
        out_sinComb_Post(output);
        break;

    default:
        /* nothing to do */
        ;
}
}
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The *poin_entry_Run()* code in the *white.c* file now looks like this:

```
/* Run Function */
void poin_entry_Run() {
double (*rtcPar)[5];
double Offset, DeltaFr, StartFr, Gain, pOut, output, samplingTime;
int i, rtc_Mode, Steps;
static double nCount;

if (trigger_Enabled()) {

    rtcPar = sh_mem_sc_Read();
    rtc_Mode = (int) (*rtcPar)[0];
    Offset = (*rtcPar)[1];
    Steps = (int) (*rtcPar)[2];
    DeltaFr = (*rtcPar)[3];
    Gain = (*rtcPar)[4];
    StartFr = (*rtcPar)[5];
    samplingTime = samplingTime_Value();

    /* Debugging info into output file */
    fprintf(f, "rtc_Mode%d ", rtc_Mode);
    fprintf(f, "Offset %f ", Offset);
    fprintf(f, "Gain %f ", Gain);
    fprintf(f, "samplingTime %f \n", samplingTime);

    switch (rtc_Mode)
    {
        case 0:
            out_sinComb_Post(0.0);
            nCount=1;
            break;

        case 1:
            out_sinComb_Post(Offset);
            nCount=1;
            break;

        case 2:
            pOut=0;
            for(i=0; i<Steps; i++)
                pOut+=(double)sin(5.283*(StartFr+i*DeltaFr)*samplingTime*nCount);
            output = Offset + Gain * pOut;
            nCount++;
            out_sinComb_Post(output);
            break;

        default:
            /* nothing to do */
            ;
    }
}
}
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

4. Make sure the *sinComb:wht_c white.c* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

The *sinComb:wht_c white.c* text file closes and you return to the *controller:behav* diagram.

Generate the Symbol

When you “generate a symbol” for the *sinComb*, VCC creates the *interface* and *symbol* views, bundles them with the *wht_c* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

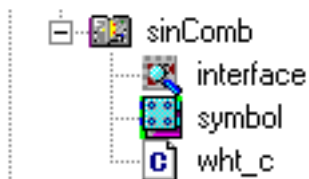
1. Click the *sinComb* block to select it.
2. Choose *Behavior > Generate Symbol* from the main menu.

At the bottom of the Create window, VCC reports:

Info: Interface file was generated successfully.

3. Choose *Workspace > Refresh Workspace* from the main menu.

You can see in the Project Folder that the new *sinComb* cell is added to the *Tutorial2* library.



The system creates the *sinComb:interface* view and the *sinComb:symbol* view.

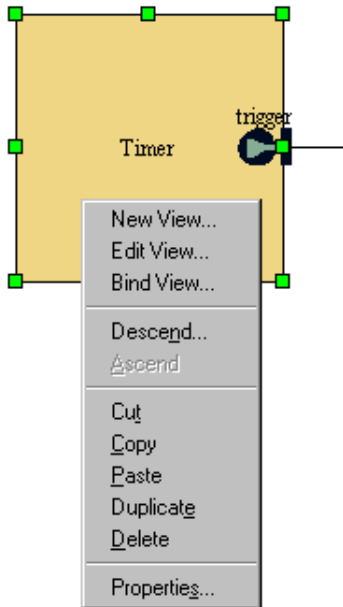
4. Make sure the *controller:behav* is the active window and choose *File > Save* from the main menu.
5. Leave the *controller:behav* diagram open and continue to the next section.

Tutorial 2: VCC Hierarchical Modeling

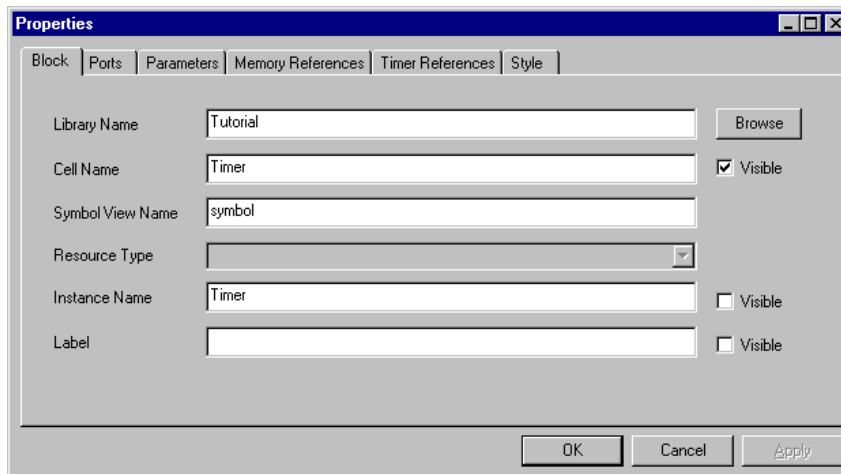
Behavior Model

Create a Functional Model for the Timer Block

1. Right click the *Timer* block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

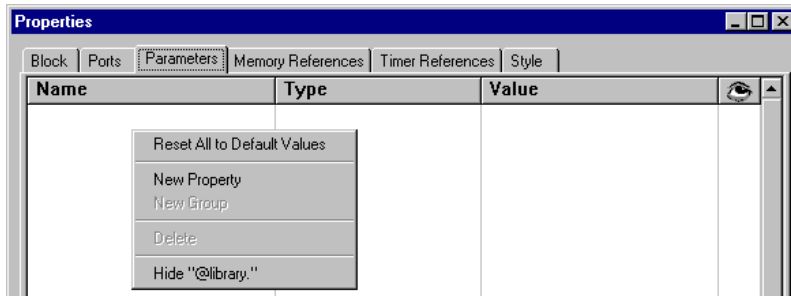


2. Click the *Parameters* tab.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

There are no Parameters specified, so all the fields are blank.



3. Right click in the blank area of the dialog and choose *New Property* from the pop-up menu.

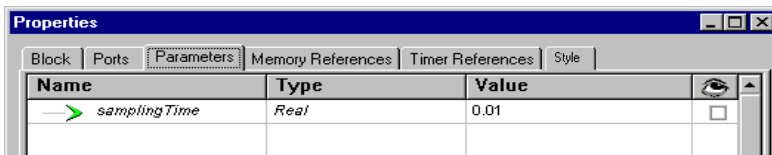
A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property:

Name	Type	Value
samplingTime	Real	0.01

4. In the Name field, enter *samplingTime*.
5. Tab to the Type field and enter *Real*.
6. Tab to the Value field and enter *0.01*.

Your new property looks like this:

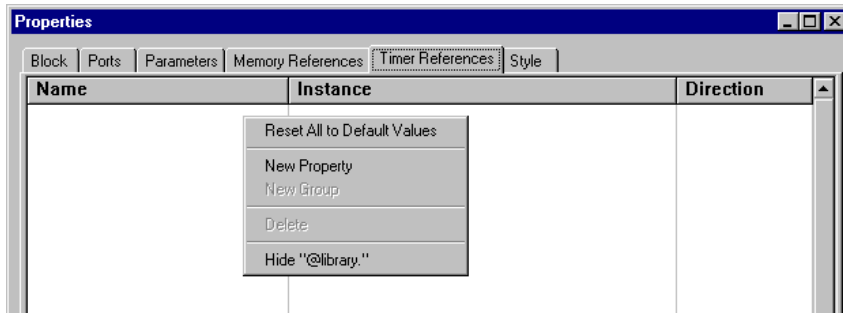


7. Click *Apply*.
8. Click the *Timer References* tab.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

There are no Timer References specified, so all the fields are blank.



9. Right click the blank area of the dialog and choose *New Property* from the pop-up menu.

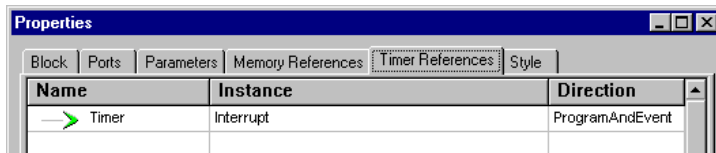
A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property:

Name	Instance	Direction
Timer	Interrupt	ProgramAndEvent

10. In the Name field, enter *Timer*.
11. Tab to the Instance field and enter *Interrupt*.
12. Tab to the Direction field, click the down arrow, and choose *ProgramAndEvent* from the pulldown menu.

The new property looks like this:



13. When you have defined these new properties, click *OK* in the Properties dialog.

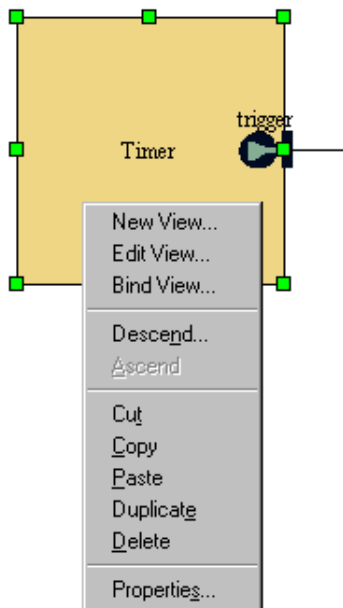
Tutorial 2: VCC Hierarchical Modeling

Behavior Model

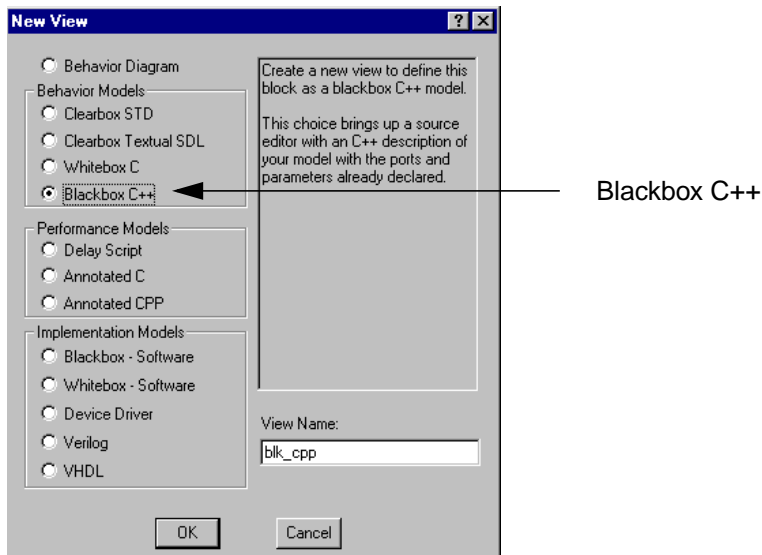
Specify a Blackbox C++ Model for the Timer Block

You will create a Blackbox C++ model for the *Timer* block.

1. Right click on the *Timer* block and choose *New View* from the pop-up menu.



The New View dialog opens.



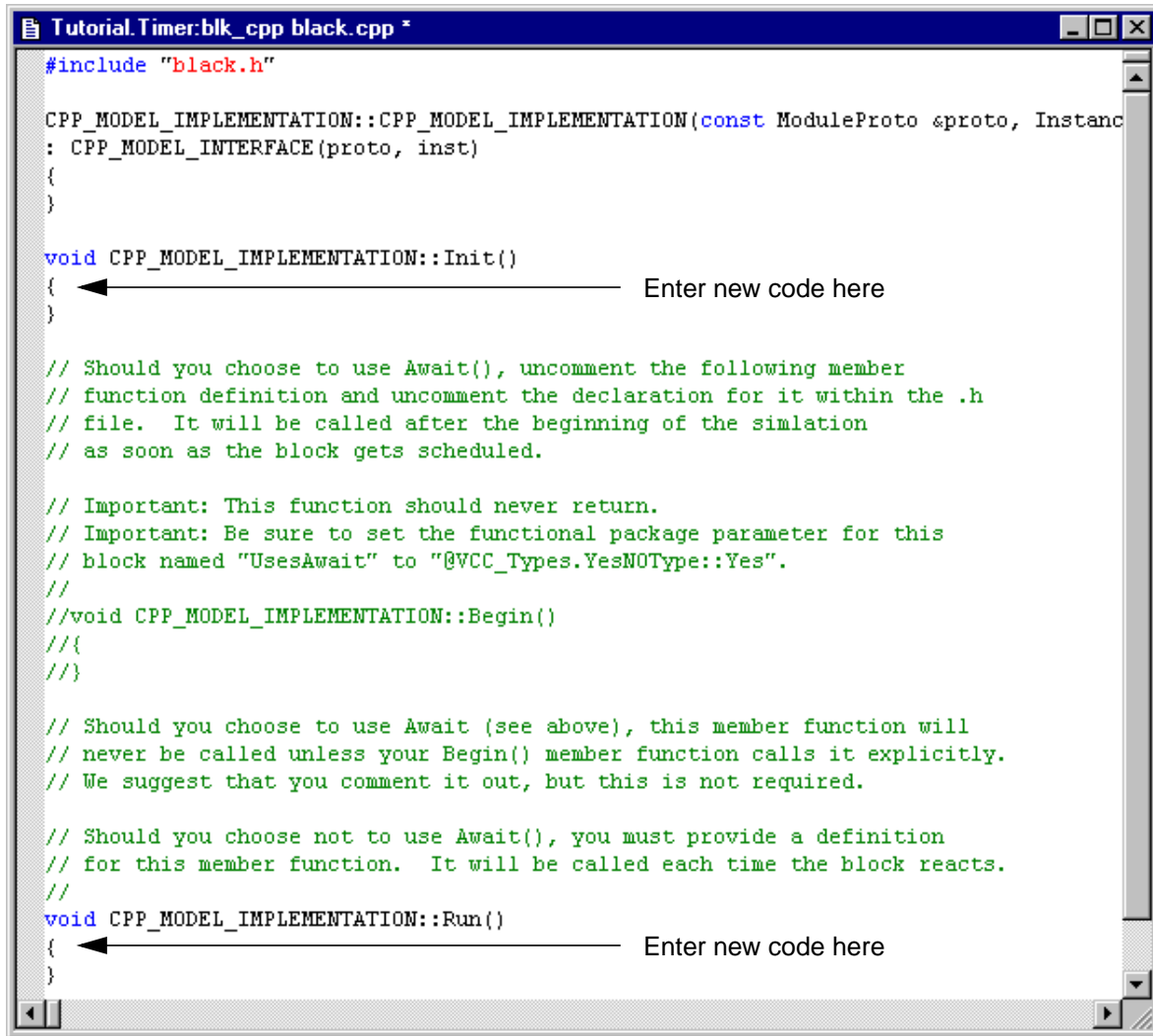
2. In the Behavior Models section, select *Blackbox C++*.

The View Name defaults to *blk_cpp*.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

3. Click **OK** in the New View dialog.

The *Timer:blk_cpp black.cpp* template file opens.



```
#include "black.h"

CPP_MODEL_IMPLEMENTATION::CPP_MODEL_IMPLEMENTATION(const ModuleProto &proto, Instance
: CPP_MODEL_INTERFACE(proto, inst)
{
}

void CPP_MODEL_IMPLEMENTATION::Init()
{
    ← Enter new code here
}

// Should you choose to use Await(), uncomment the following member
// function definition and uncomment the declaration for it within the .h
// file. It will be called after the beginning of the simulation
// as soon as the block gets scheduled.

// Important: This function should never return.
// Important: Be sure to set the functional package parameter for this
// block named "UsesAwait" to "@VCC_Types.YesNOType::Yes".
//
//void CPP_MODEL_IMPLEMENTATION::Begin()
//{
//}

// Should you choose to use Await (see above), this member function will
// never be called unless your Begin() member function calls it explicitly.
// We suggest that you comment it out, but this is not required.

// Should you choose not to use Await(), you must provide a definition
// for this member function. It will be called each time the block reacts.
//
void CPP_MODEL_IMPLEMENTATION::Run()
{
    ← Enter new code here
}
```

4. Enter the following lines of code in the `void CPP_MODEL_IMPLEMENTATION::Init()` block.

Be sure to enter the new code between the opening and closing braces in the template.

```
Timer.SetTimer(samplingTime.Value());
Timer.StartTimer();
```

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

The code in the *black.cpp* file now looks like this:

```
void CPP_MODEL_IMPLEMENTATION::Init()
{
    Timer.SetTimer(samplingTime.Value());
    Timer.StartTimer();
}
```

5. Enter the following lines of code in the `void CPP_MODEL_IMPLEMENTATION::Run()` block.

Be sure to enter the new code between the opening and closing braces in the template.

```
typeObjectTrigger (x);
trigger.Post(x);
```

The code in the *black.cpp* file now looks like this:

```
void CPP_MODEL_IMPLEMENTATION::Run()
{
    typeObjectTrigger (x);
    trigger.Post(x);
}
```

6. Make sure the *Timer:blk_cpp black.cpp* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

The *Timer:blk_cpp black.cpp* text file closes and you return to the *controller:behav* diagram.

Generate the Symbol

When you “generate a symbol” for the *Timer*, VCC creates the *interface* and *symbol* views, bundles them with the *blk_cpp* view you created, and adds the new cellview to the Project Folder. This new cellview is now reusable like any other cellview in a library.

1. Click the *Timer* block to select it.
2. Choose *Behavior > Generate Symbol* from the main menu.

At the bottom of the Create window, VCC reports:

```
Info: Interface file was generated successfully.
```

3. Choose *Workspace > Refresh Workspace* from the main menu.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

You can see in the Project Folder that the new *Timer* cell is added to the *Tutorial2* library.



The system creates the *Timer:interface* view and the *Timer:symbol* view.

4. Make sure the *controller:behav* file is the active window, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

Run a Functional Simulation

A functional simulation verifies that your behavior design works correctly.

In this section, you will:

- [Create an Analysis Session](#) on page 112
- [Run the Simulation](#) on page 114

Tutorial 2: VCC Hierarchical Modeling

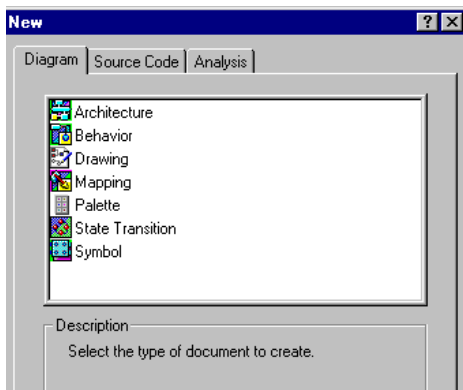
Behavior Model

Create an Analysis Session

Each time you run an analysis, you can create a new analysis session. You might want to create multiple configurations, run analysis on each configuration, then compare the results of the analysis sessions.

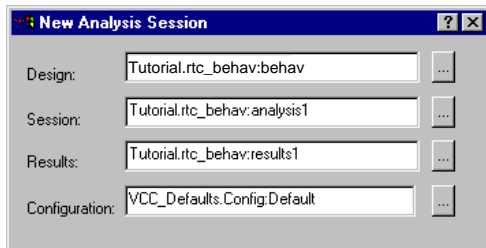
1. Choose *File > New* from the main menu.

The New dialog opens.



2. Click the *Analysis* tab, then double-click the *Analysis* icon.

The New Analysis Session dialog opens.



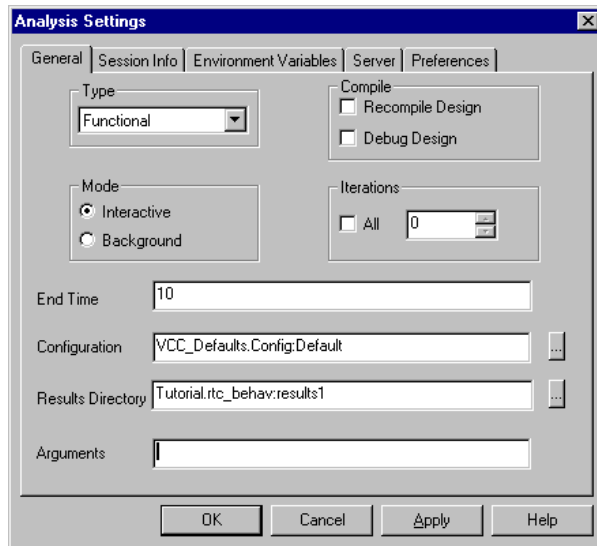
3. Accept the default values and click *OK*.

Tutorial 2: VCC Hierarchical Modeling Behavior Model

Specify Settings

1. Choose *Analysis > Settings* from the main menu.

The Analysis Settings dialog opens.



2. Set the Type to *Functional*.
3. Set the Mode to *Background*.
4. Set the End Time to 10.
5. Click *OK* in the Analysis Settings dialog.

Run the Simulation

Note: When the session runs in *Background* mode, you don't have to explicitly run the *Analysis > Initialize* command.

1. To begin the analysis, choose *Analysis > Go/Continue* from the main menu.

A dialog asks if you want to "Save session changes?"

2. Click **Yes** in the Analysis Cockpit dialog.

VCC reports the progress in the Analysis Output window at the bottom of the screen.

Successful analysis message:

```
Info: Run(0): Linking models into simulator ...  
Info: Run(0): Done.
```

Unsuccessful analysis message (caused by a missing parameter value):

```
Error: The parameter "<parameter name>" has no value.  
{in /(Tutorial2.rtc_behav:behav)rtc_testbench}.
```

This error message is only a sample. You might see other types of messages, but the information in the message should help you correct your design.

3. Correct any errors and run the analysis until the simulation is successful.

There are several ways to display and locate errors in your simulation. The following tips describe some of these methods.

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

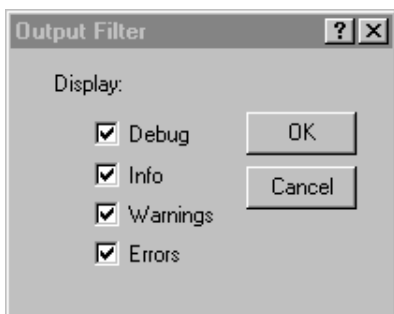
How to Locate Errors



Tip

To control the output of messages to the Analysis Output window, right click in the window and choose *Output Filters*.

The Output Filter dialog lets you choose which types of messages you want displayed in the Analysis Output window.



- *Debug* reports detailed information about what the simulator is doing.
- *Info* reports the block-by-block progress of the simulation.
- *Warnings* do not cause the netlist generation to fail.
- *Errors* cause the netlist generation to fail.



Tip

To open the object where an error occurred, double-click on the error in the Analysis Output window.



Tip

To view the complete output of the analysis, open the following directory:

Windows NT: C:\MyWorkspace\Tutorial2\rtc_behav\results

Solaris and HP-UX:

<install directory>/myworkspace/Tutorial2/rtc_behav/results

This directory contains several error, debugging, and information files.

For the *rtc_behav* model, the most important output information is in the *debug_out.txt* file for the *sinComb* block.

What if Your Model Won't Simulate?

If you are unable to correct all the errors in your behavior model, you can use the example behavior model for your mapping diagram later in this tutorial.

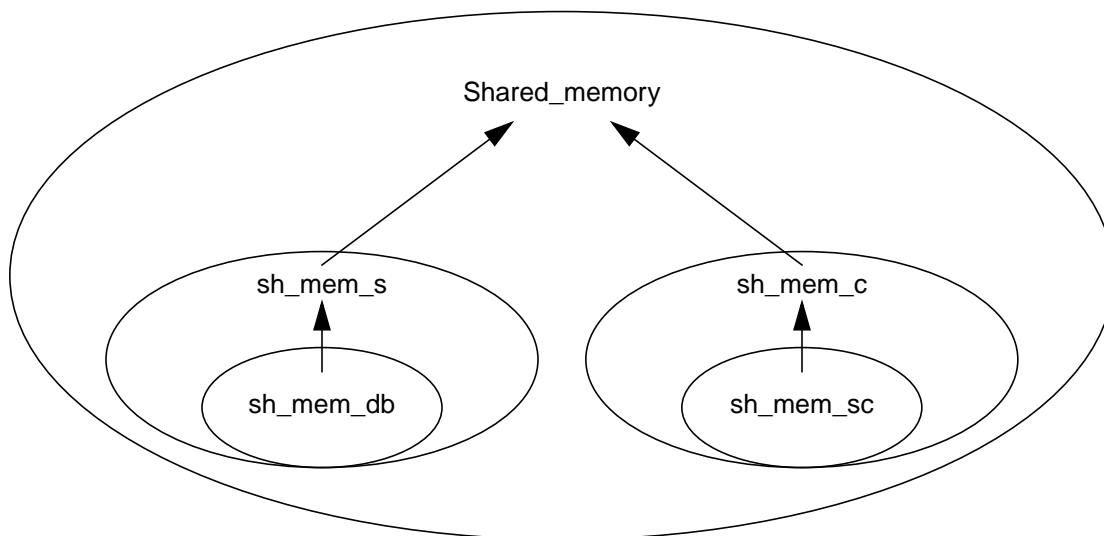
The example behavior cellview is named *x_rtc_behav*. When the instructions for the mapping diagram tell you to instantiate *rtc_behav*, you can use *x_rtc_behav*.

How is the Behavior Memory Connected?

As you can see in your diagram, you didn't add any wires between the *server*, *Shared_memory*, and *controller* blocks. So how are these blocks connected? If you place your cursor over the *Shared_memory* block, you can see red lines that indicate connections to both the *server* and *controller* blocks. Where did these connections come from?

Beginning with the low-level blocks in the *server* and *controller*, you specified memory reference parameters. Using hierarchical memory reference parameters, you connected these low-level blocks to higher-level blocks, eventually connecting to the *Shared_memory* at the top level.

Memory Reference Hierarchy



Memory reference parameters are specified on the Memory References tab of the Properties dialog for each block in the hierarchy.

- The *sh_mem_db* memory reference in the *server_db* block is assigned to the *sh_mem_s* memory reference in the *server* block.

Block	Ports	Parameters	Memory References	Timer References	Style
Name	Type	Instance	Mode		
sh_mem_db	Real[6]	sh_mem_s	WriteOnly		

Tutorial 2: VCC Hierarchical Modeling

Behavior Model

- The *sh_mem_s* memory reference in the *server* block is assigned to the *Shared_memory* behavior memory at the top level.

Block Ports Parameters Memory References Timer References			
Name	Type	Instance	Mode
sh_mem_s	REAL[6]	Shared_memory	WriteOnly

- The *sh_mem_sc* memory reference in the *sinComb* block is assigned to the *sh_mem_c* memory reference in the *controller* block.

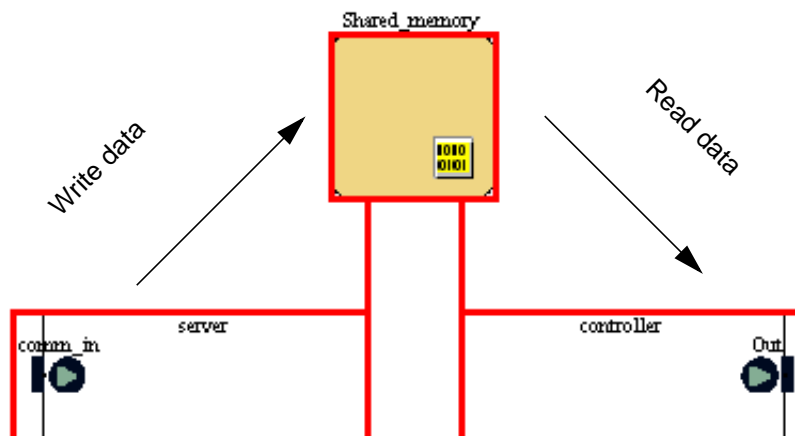
Block Ports Parameters Memory References Timer References			
Name	Type	Instance	Mode
sh_mem_sc	Real[6]	sh_mem_c	ReadOnly

- The *sh_mem_c* memory reference in the *controller* block is assigned to the *Shared_memory* behavior memory at the top level.

Block Ports Parameters Memory References Timer References			
Name	Type	Instance	Mode
sh_mem_c	REAL[6]	Shared_memory	ReadOnly

The *Instance* specified for each memory reference parameter defines the connection to the next level. The *Mode* specifies the direction of the data flow. The memory reference parameters in the *server* block are *WriteOnly*; the memory reference parameters in the *controller* block are *ReadOnly*. Through these memory reference connections, data is written from the *server* block to the behavior memory, then read from the behavior memory to the *controller* block.

Memory Reference Connections



Architecture Model

The architecture model for this real-time controller design contains two CPUs; one CPU hosts the *server* application and the second CPU hosts the *controller* application. Shared memory communication is realized by a simple memory over a data bus.

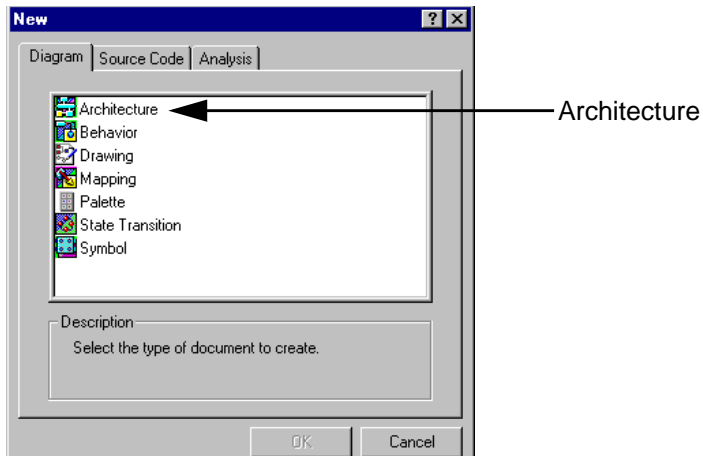
To create this architecture model, you will

- [Create a New Architecture Diagram](#) on page 120
- [Place the Architecture Blocks](#) on page 122
- [Specify Block Properties](#) on page 123
- [Add Buses to the Diagram](#) on page 132
- [Wire the Architecture Blocks](#) on page 140
- [Generate the Netlist](#) on page 145

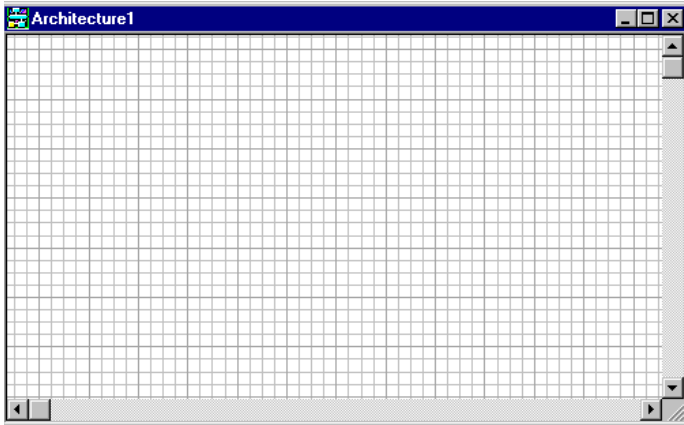
If you have problems with your architecture model that you can't resolve, instructions for using a golden copy from the database are in [What if Your Netlist Won't Generate?](#) on page 147.

Create a New Architecture Diagram

1. To open a new architecture diagram, choose *File > New* from the main menu.



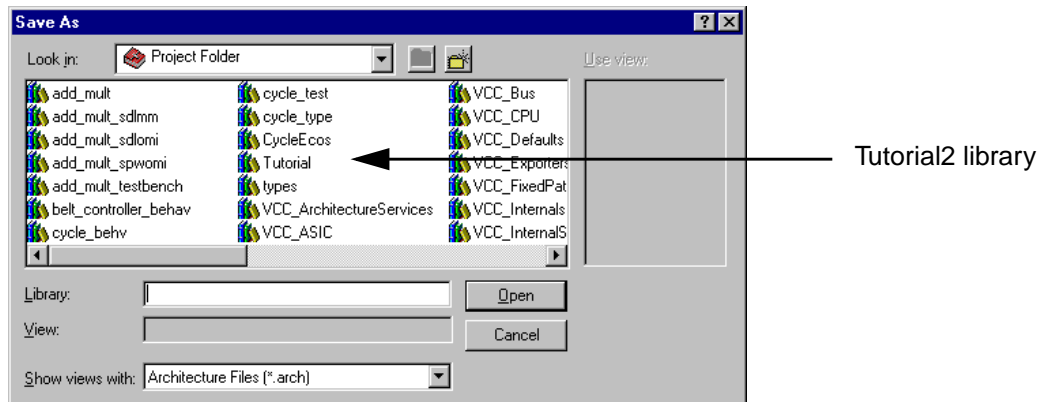
2. Double-click the *Architecture* icon in the New dialog.
A new window opens. The default name is *Architecture1*.



3. To save the architecture diagram, choose *File > Save* from the main menu.

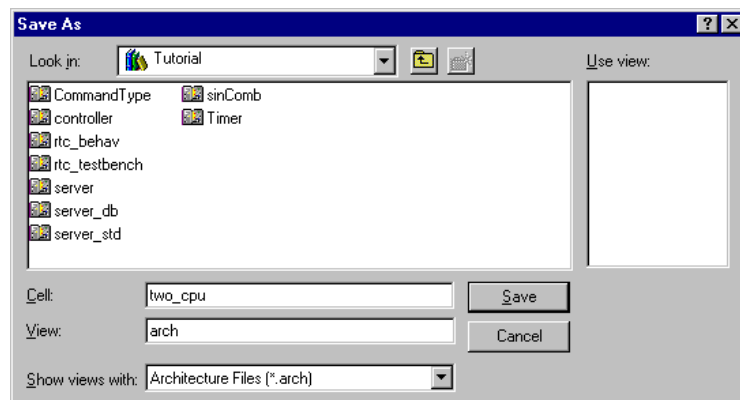
Tutorial 2: VCC Hierarchical Modeling Architecture Model

VCC prompts you to choose a library.



4. Double-click the *Tutorial2* library icon.

VCC prompts you to specify a cell name. The View defaults to *arch*.



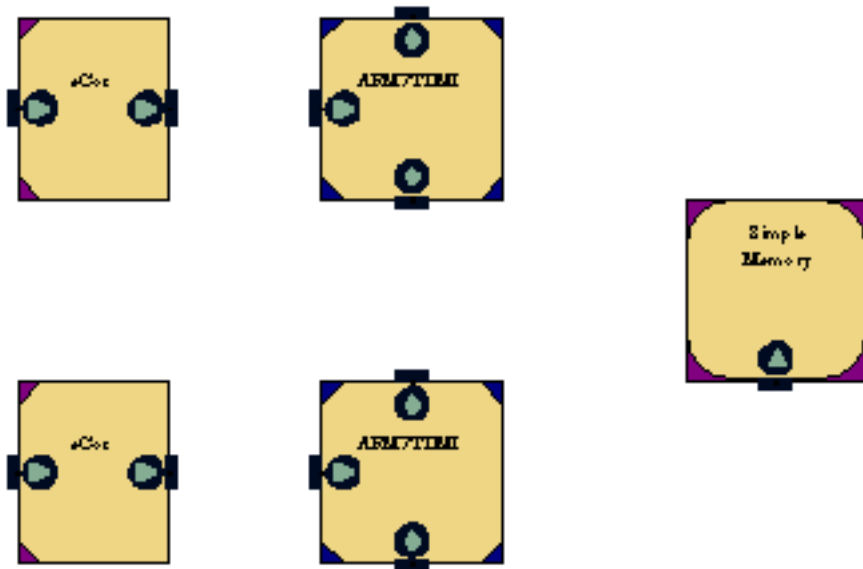
5. In the Cell field, enter *two_cpu* and click *Save*.

Place the Architecture Blocks

This design uses architecture blocks from the *VCCIP_ECOS*, *VCCIP_ARM7*, and *VCC_Memory* libraries.

1. On the Project Folder, expand the *VCCIP_ECOS* library, then expand the *eCos* cell.
2. Instantiate two copies of the *eCos* symbol view in your diagram.
3. On the Project Folder, expand the *VCCIP_ARM7* library, then expand the *ARM7TDMI* cell.
4. Instantiate two copies of the *ARM7TDMI* symbol view in your diagram.
5. On the Project Folder, expand the *VCC_Memory* library, then expand the *SimpleMemory* cell.
6. Instantiate the *SimpleMemory* symbol view in your diagram.

Your diagram now looks like this:

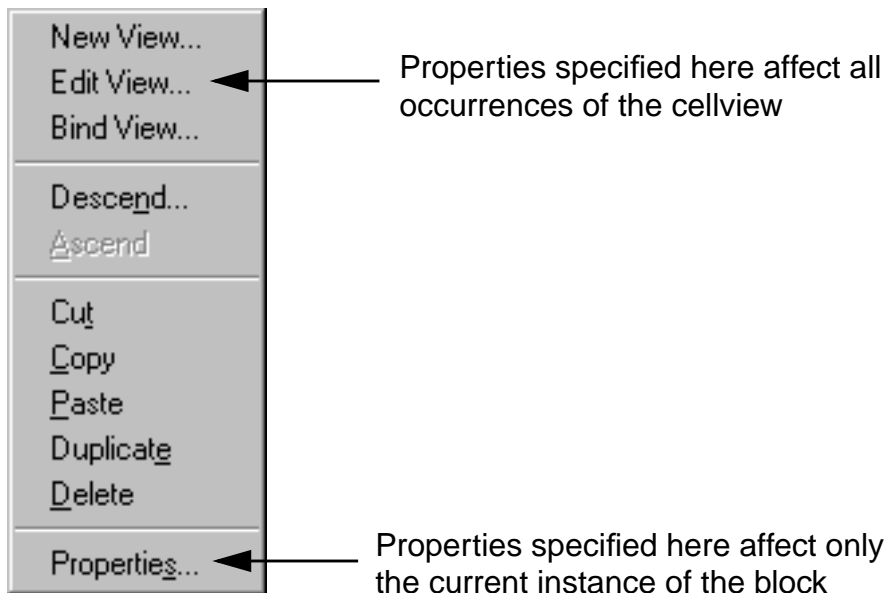


7. When you finish instantiating these blocks, click the minus signs to collapse the expanded *VCCIP_ECOS*, *VCCIP_ARM7*, and *VCC_Memory* libraries.

Specify Block Properties

Each of the blocks you instantiated from the standard libraries requires parameters that are specific to this design.

- Some properties are specified at the instance level. These properties affect only one instance of a block.
- Other properties are specified at the view level. These properties affect all occurrences of a cellview.

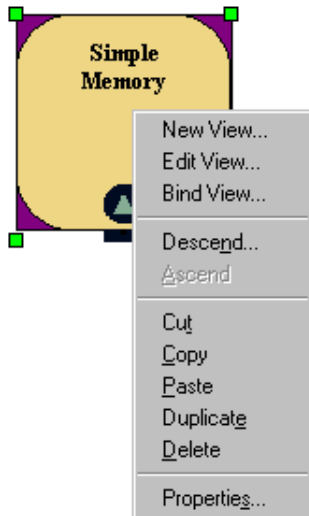


Tutorial 2: VCC Hierarchical Modeling

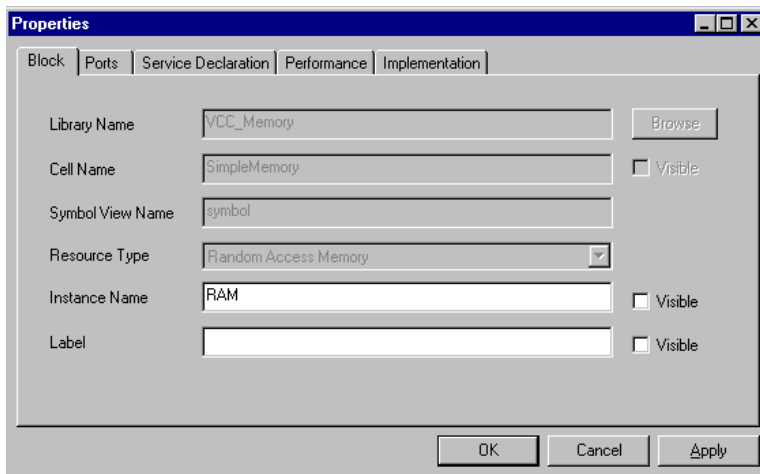
Architecture Model

Simple Memory

1. To specify an instance name for the *Simple Memory* block, right click on the block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

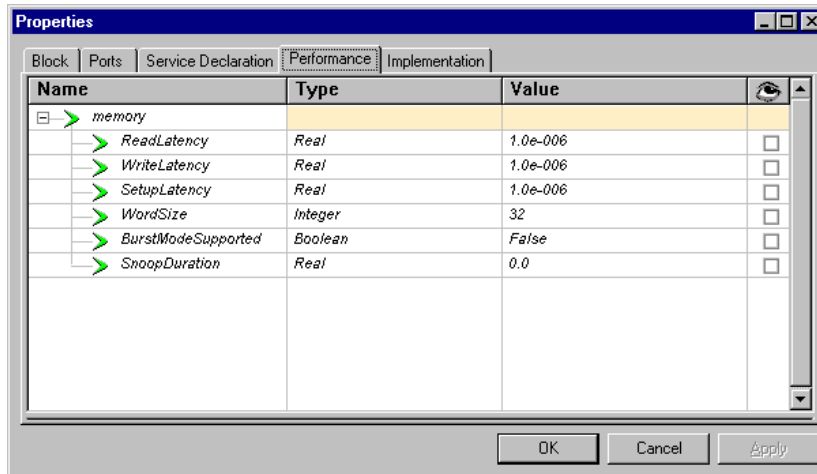


2. Enter *RAM* for the Instance Name and click *Apply*.
3. Click the *Performance* tab.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

4. Click the plus sign to expand the *Memory* header.



5. Double-click the Value field and enter the value $1.0e-6$ for each of the following parameters:

ReadLatency

WriteLatency

SetupLatency

- ☐ On Windows NT, when you enter $1.0e-6$, the system automatically changes it to $1.0e-006$.
- ☐ On Solaris and HP-UX, when you enter $1.0e-6$, the system automatically changes it to $1.0e-06$.

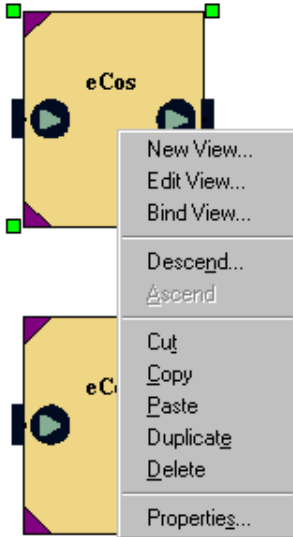
6. When you have entered these three parameter values, click *OK* in the Properties form.

Tutorial 2: VCC Hierarchical Modeling

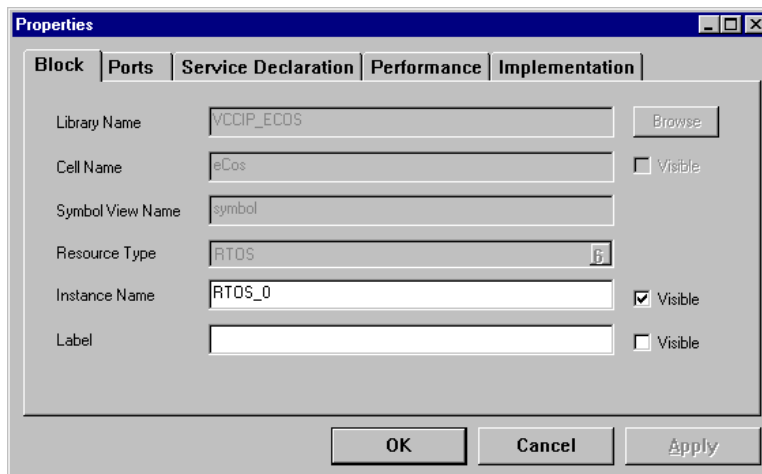
Architecture Model

eCos

1. To specify an instance name for the first eCos block, right click on the block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

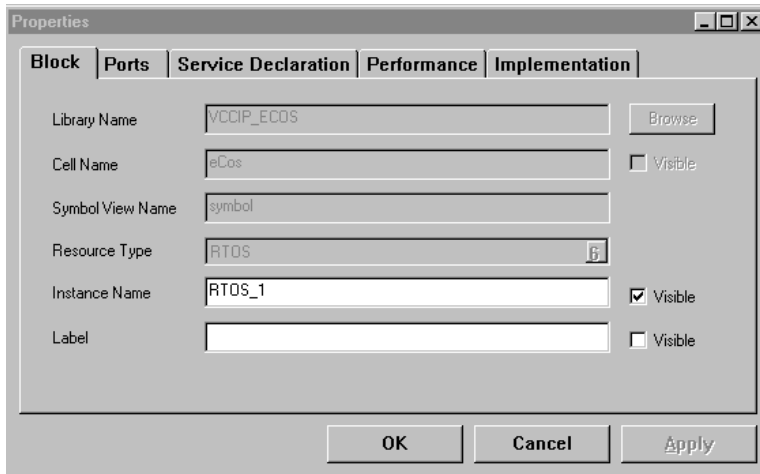


2. Enter *RTOS_0* for the Instance Name and make it visible.
3. Click *OK* in the Properties dialog.

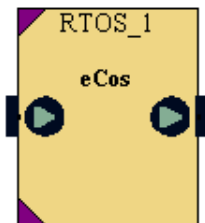
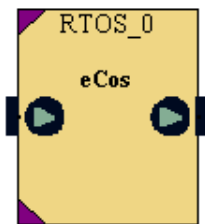
Tutorial 2: VCC Hierarchical Modeling

Architecture Model

- Repeat steps 1-3 for the second eCos block, setting the Instance Name to *RTOS_1*.



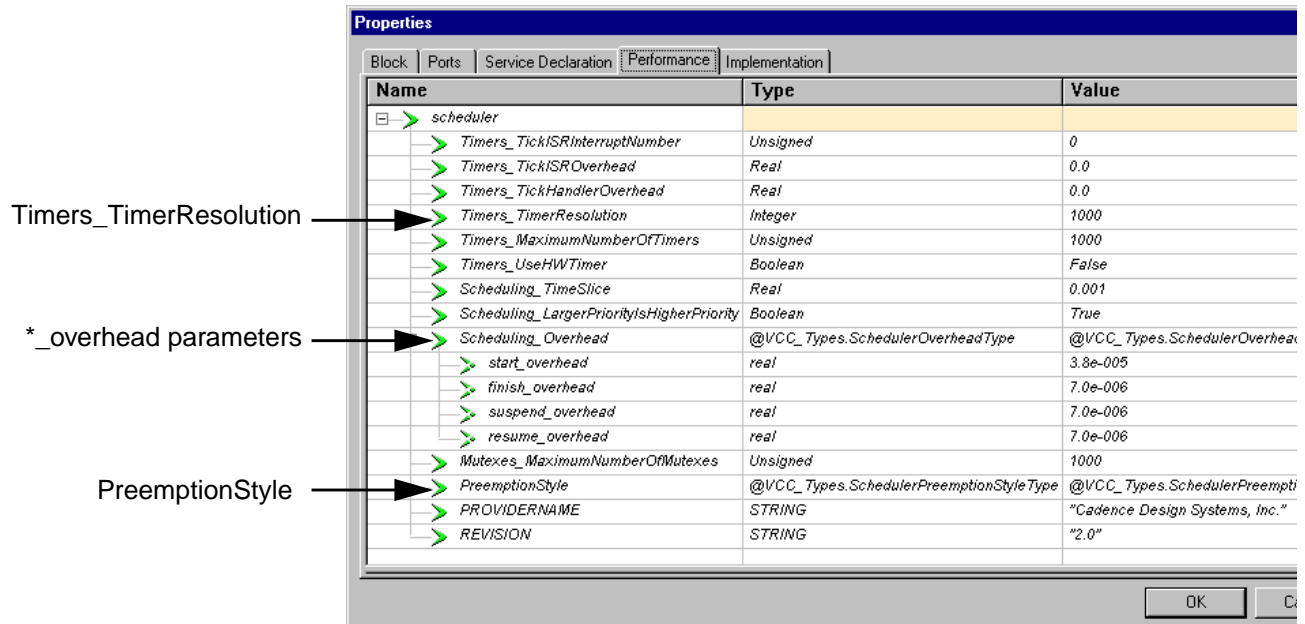
Your eCos blocks now look like this:



- To specify the scheduler parameters, right click on the *RTOS_0* block and choose *Properties* from the pop-up menu.
- Click the *Performance* tab.

Tutorial 2: VCC Hierarchical Modeling Architecture Model

7. Click the plus sign to expand the *scheduler* header.



8. Enter the following values for these parameters:

Timers_TimerResolution	1000
start_overhead	3.8e-5
finish_overhead	7.0e-6
suspend_overhead	7.0e-6
resume_overhead	7.0e-6
PreemptionStyle	Suspend

9. When you have entered all these parameter values, click *OK* in the Properties dialog.

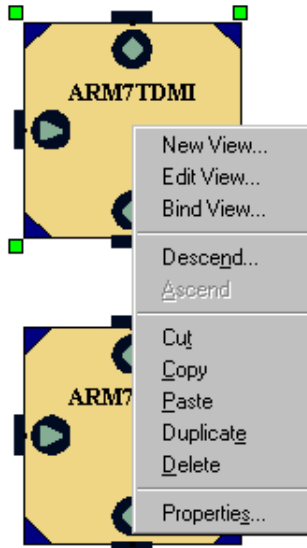
10. Repeat steps 5-8 for the *RTOS_1* block.

Tutorial 2: VCC Hierarchical Modeling

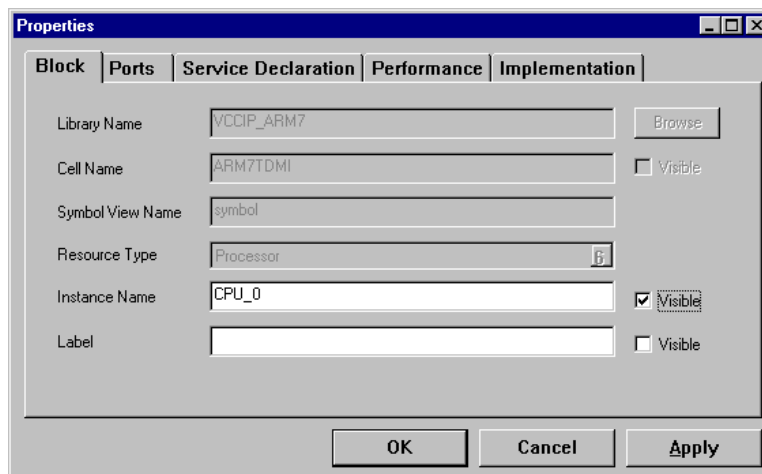
Architecture Model

ARM7TDMI

1. To specify an instance name for the first *ARM7TDMI* block, right click on the block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

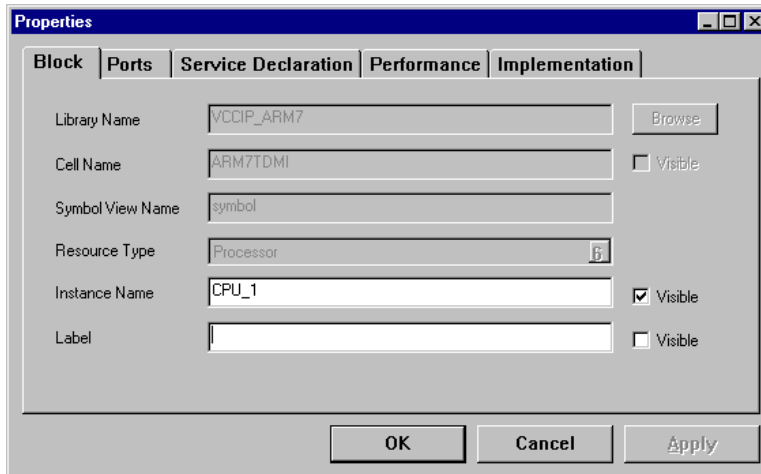


2. Enter *CPU_0* for the Instance Name and make it visible.
3. Click *OK* in the Properties dialog.

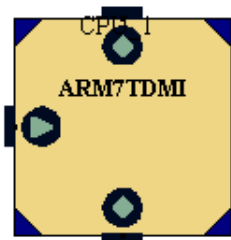
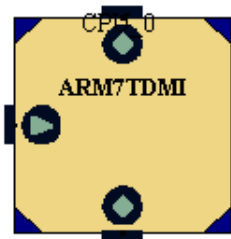
Tutorial 2: VCC Hierarchical Modeling

Architecture Model

- Repeat steps 1-3 for the second *ARM7TDMI* block, setting the Instance Name to *CPU_1*.

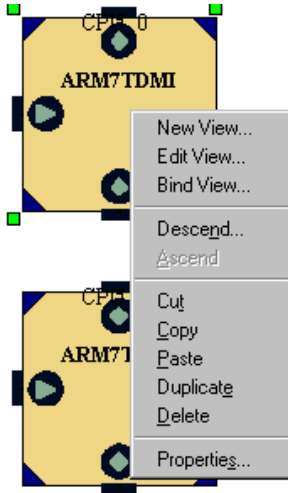


Your *ARM7TDMI* blocks now look like this:



Tutorial 2: VCC Hierarchical Modeling Architecture Model

5. To specify the processor parameters, right click on the *CPU_0* block and choose *Properties* from the pop-up menu.



The Properties dialog opens.

6. Click the *Performance* tab.
7. Click the plus sign to expand the *proc* header.

Name	Type	Value
proc		
WordSize	Unsigned	32
IC_ControllerOn	Boolean	True
IC_NumberOfIRQs	Unsigned	20
IC_PrioritiesOn	Boolean	True
Compiler_StackSize	Unsigned	1024
Compiler_BitsPerByte	Unsigned	8
Compiler_TypeSpecification	SizeofSpecification	@VCC_Types.SizeofSpecification(4,8,1,4,4,...
MemoryAccess_MaxTimeSpanFor...	Real	0.0
MemoryAccess_MaxBytesOfMerge...	Unsigned	0
MemoryAccess_MergeTransaction...	Integer	4
MemoryAccess_MergeTransactions	Boolean	False
MemoryAccess_InstructionMode	MemoryAccessPerfMode	@VCC_Types.MemoryAccessPerfMode::None
MemoryAccess_DataWriteMode	MemoryAccessPerfMode	@VCC_Types.MemoryAccessPerfMode::None
MemoryAccess_DataReadMode	MemoryAccessPerfMode	@VCC_Types.MemoryAccessPerfMode::None
MemoryAccess_CommunicationMo...	MemoryAccessPerfMode	@VCC_Types.MemoryAccessPerfMode::Bus...
CPS	Real	33000000.0

8. For the *CPS* parameter, double-click the Value field and enter 33000000.0 (thirty three million).

Note: You can enter this value as 33e6 and the system will automatically change it to 33000000.0.

9. Click *OK* in the Properties dialog.
10. Repeat steps 5-9 for the *CPU_1* block.

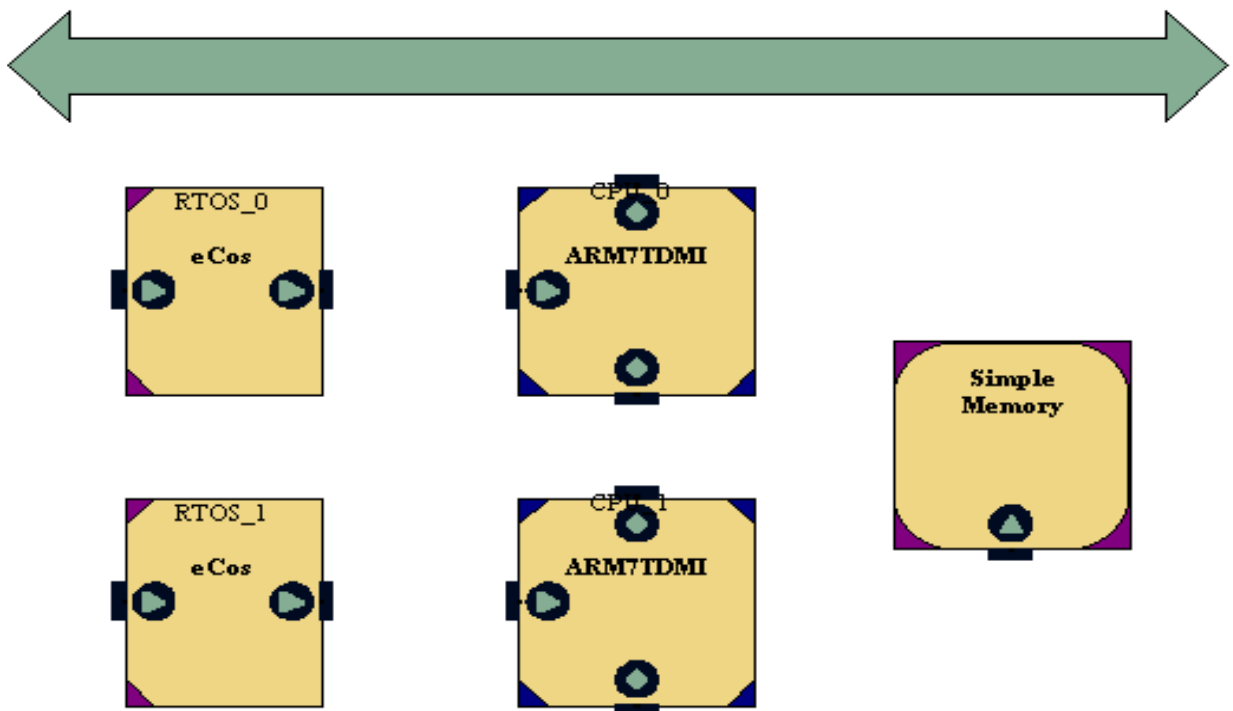
Add Buses to the Diagram

This architecture diagram uses two buses: a *data* bus and an *interrupt* bus. In many ways, a bus is just like any other architecture block with instance information and assigned parameters.

Add a Data Bus

1. To add a bus to your diagram, choose *Architecture > Bus* from the main menu.
2. At the top of the diagram, click to place the starting point of the bus, drag your cursor across the diagram, and double-click to terminate the bus.

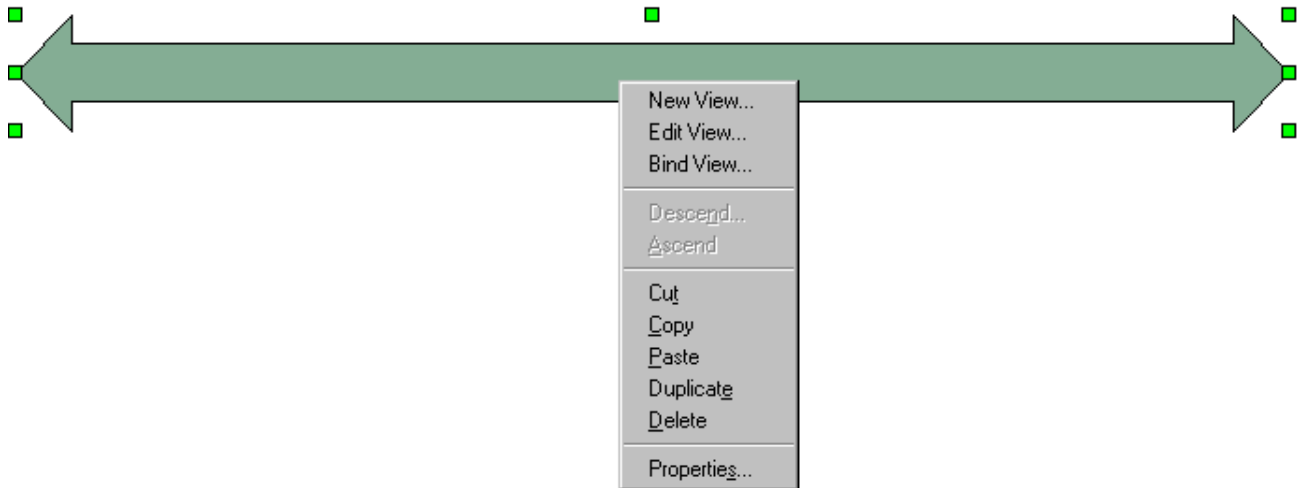
Your diagram should look like this:



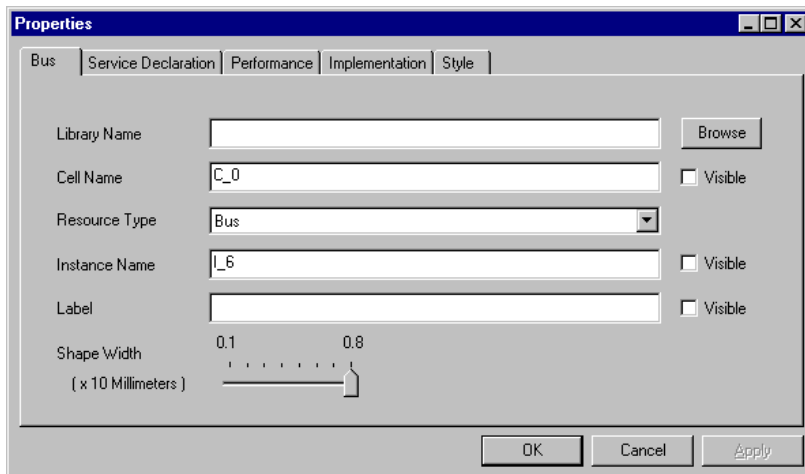
Tutorial 2: VCC Hierarchical Modeling

Architecture Model

3. Right click on the bus and choose *Properties* from the pop-up menu.



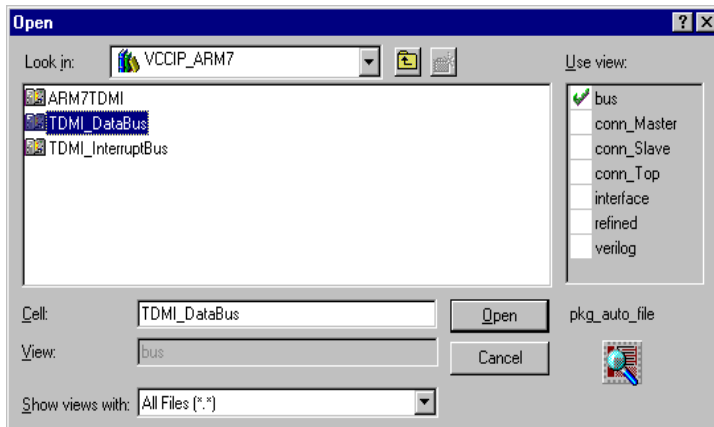
The Properties dialog opens.



4. To open a browser where you can specify the Library Name and Cell Name, click *Browse*.
5. Scroll to the *VCCIP_ARM7* library and double-click the library icon.

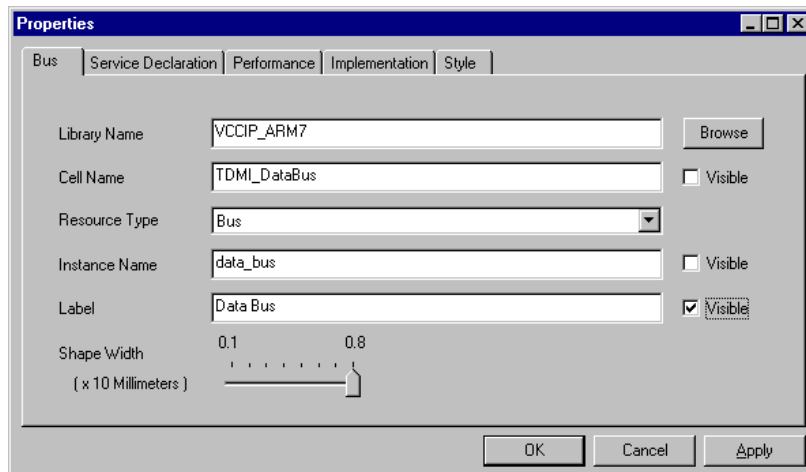
Tutorial 2: VCC Hierarchical Modeling Architecture Model

6. Select the *TDMI_DataBus* cell and click *Open*.



The View defaults to *bus*.

7. In the Properties dialog, set the Instance Name to *data_bus*.
8. Set the Label to *Data Bus* and make it visible.



9. To save these values, click *Apply*.

Do not close the Properties dialog. You will now set the Performance properties.

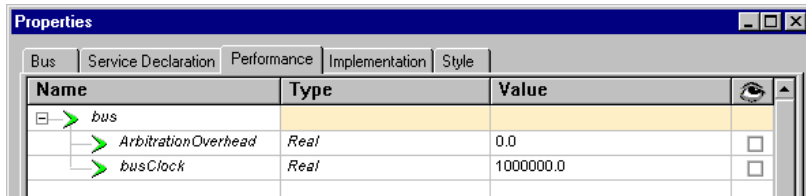
10. Click the *Performance* tab.

The Performance tab displays the *bus* parameter header.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

11. Click the plus sign to expand the *bus* header.



12. Double-click the Value field for *busClock* and enter the value *1000000.0* (one million).

Note: You can enter this value as *1.0e6* and the system will automatically change it to *1000000.0*.

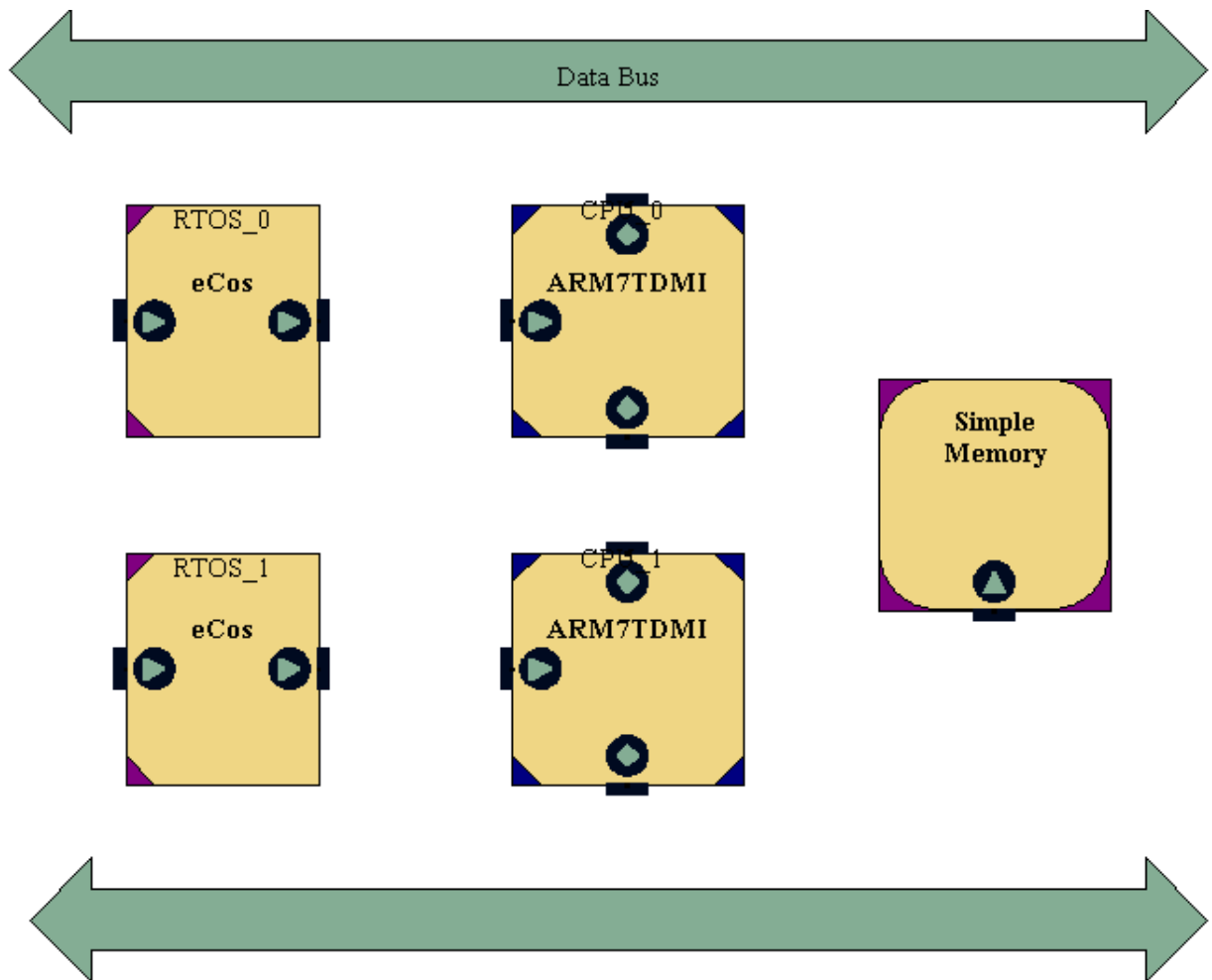
13. Click *OK* in the Properties dialog.

Add an Interrupt Bus

1. To add a second bus to your diagram, choose *Architecture > Bus* from the main menu.
2. At the bottom of the diagram, click to place the starting point of the bus, drag your cursor across the diagram, and double-click to terminate the bus.

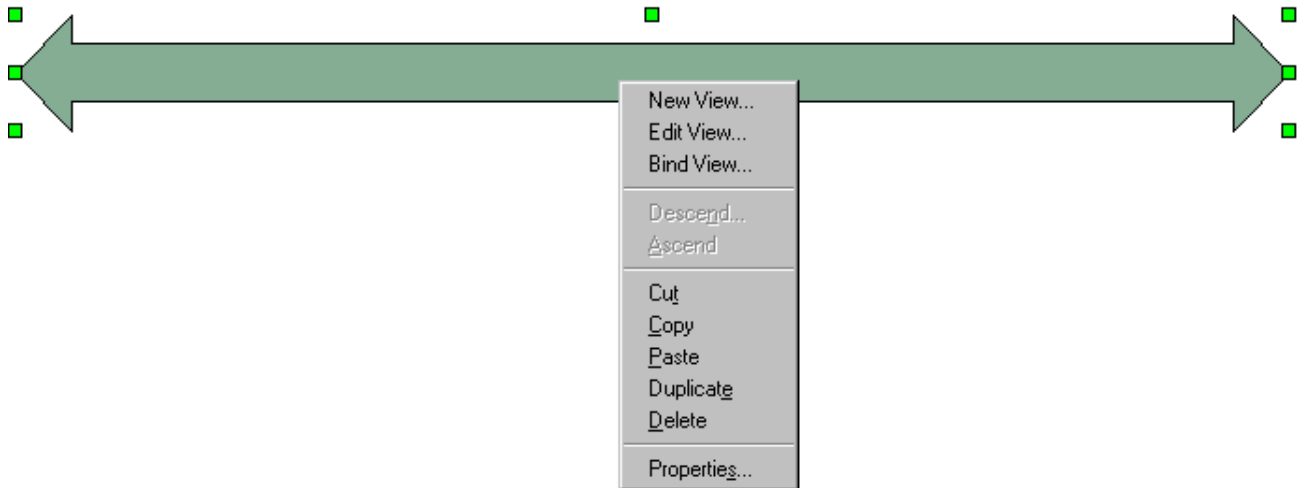
Tutorial 2: VCC Hierarchical Modeling Architecture Model

Your diagram should look like this:

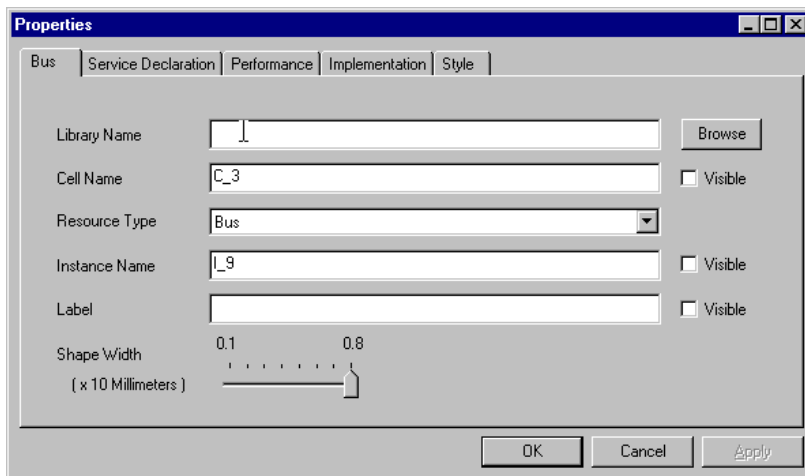


Tutorial 2: VCC Hierarchical Modeling Architecture Model

3. Right click on the new bus and choose *Properties* from the pop-up menu.



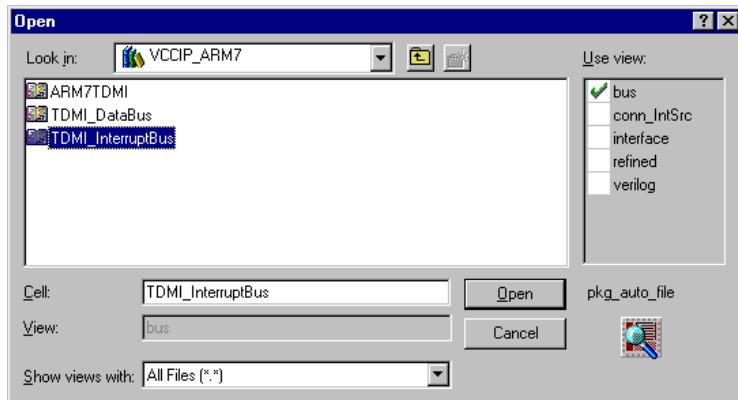
The Properties dialog opens.



4. To open a browser where you can specify the Library Name and Cell Name, click *Browse*.
5. Scroll to the *VCCIP_ARM7* library and double-click the library icon.

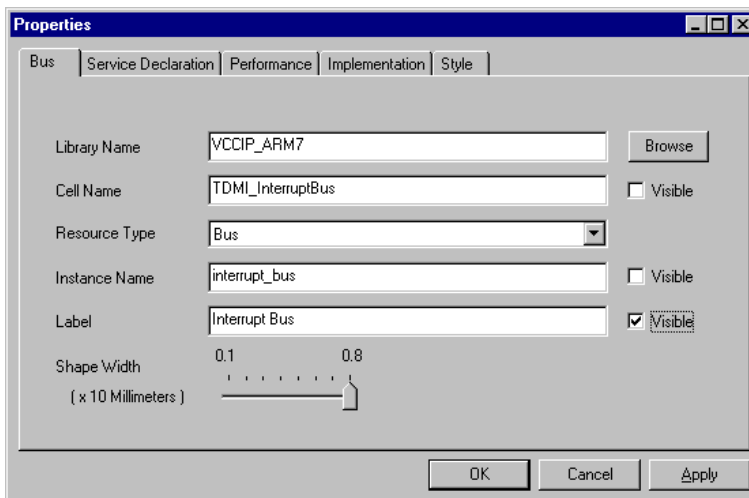
Tutorial 2: VCC Hierarchical Modeling Architecture Model

6. Select the *TDMI_InterruptBus* cell and click *Open*.



The View defaults to *bus*.

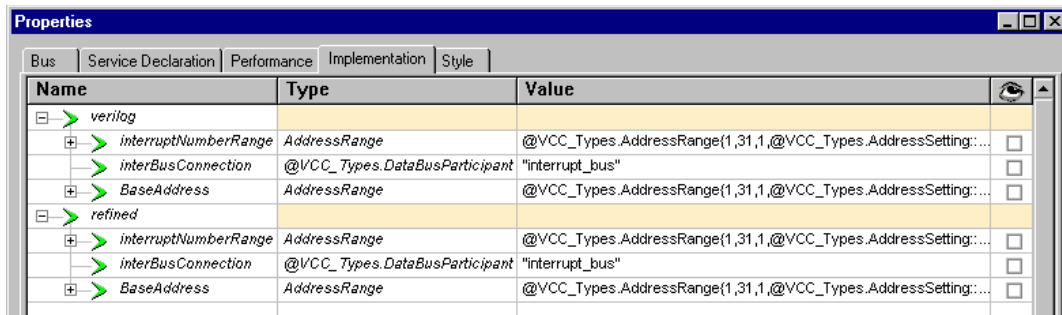
7. In the Properties dialog, set the Instance Name to *interrupt_bus*.
8. Set the Label to *Interrupt Bus* and make it visible.



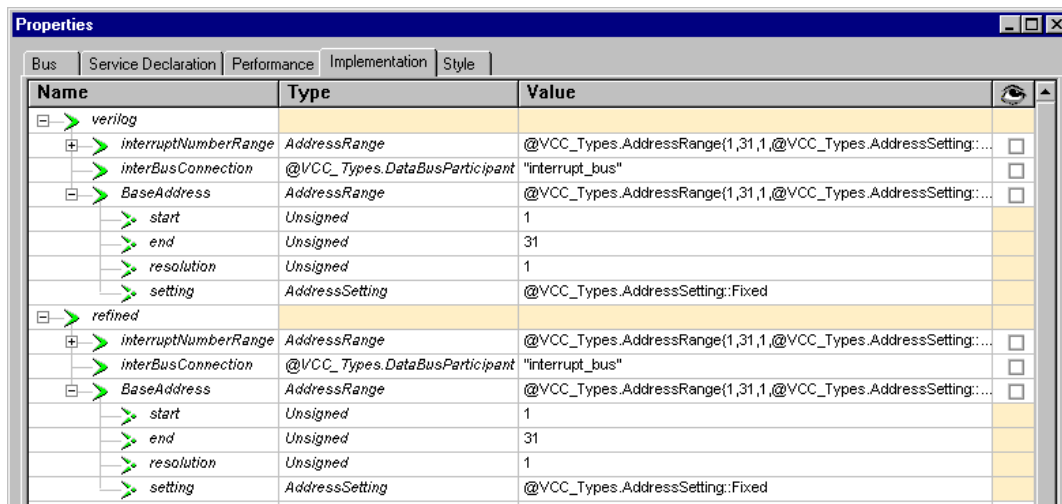
9. Click *Apply*, then click the *Implementation* tab.
10. Expand the *verilog* and *refined* headers.
11. Double-click in the Value field for the *verilog* > *interBusConnection* parameter.
12. Click the down arrow and choose "*interrupt_bus*".
13. Double-click in the Value field for the *refined* > *interBusConnection* parameter.

Tutorial 2: VCC Hierarchical Modeling Architecture Model

14. Click the down arrow and choose “*interrupt_bus*”.



15. Expand the *verilog > BaseAddress* and *refined > BaseAddress* headers.



16. Enter the following values for the *verilog > BaseAddress* and *refined > BaseAddress* parameters:

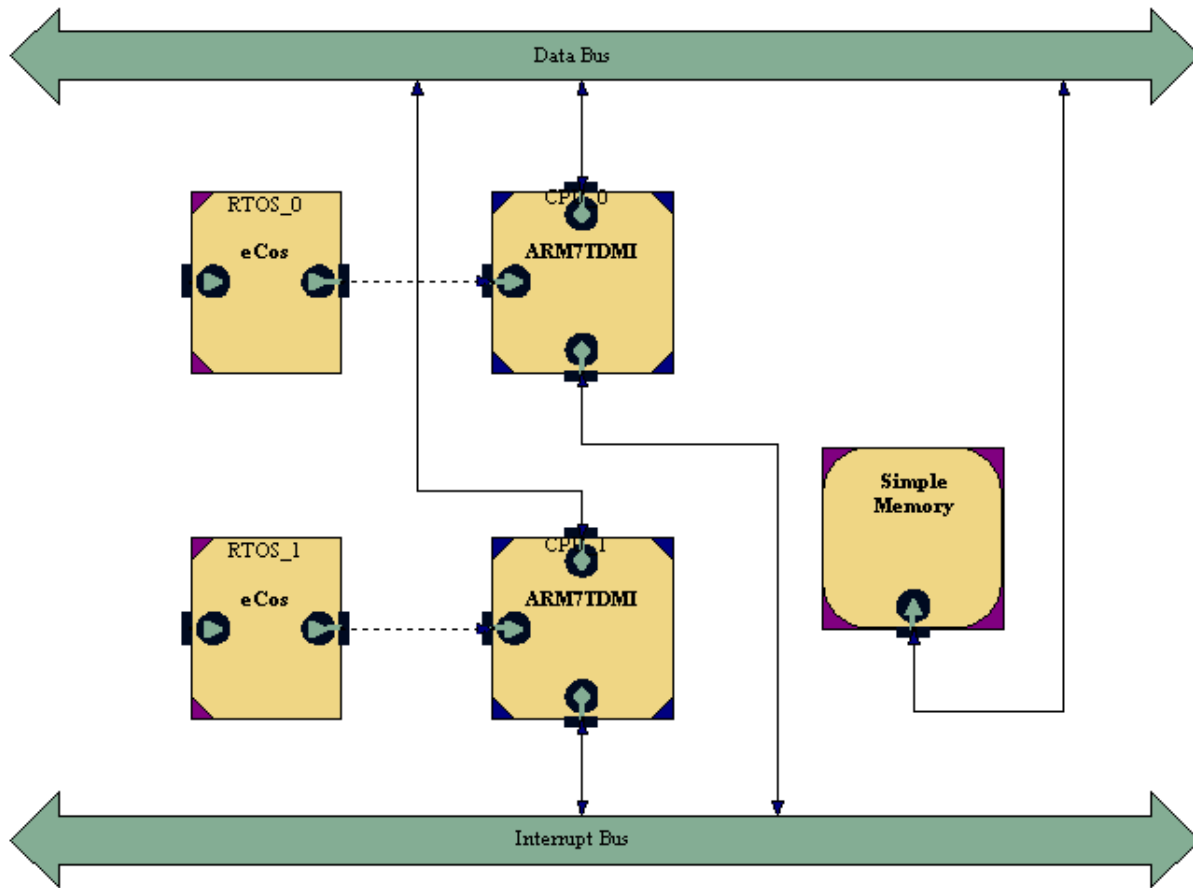
start	1
end	31
resolution	1
setting	Fixed

17. Click OK in the Properties dialog.

Wire the Architecture Blocks

Architecture communication wires can connect blocks to blocks or blocks to buses. Scheduler assignments can connect schedulers to other blocks.

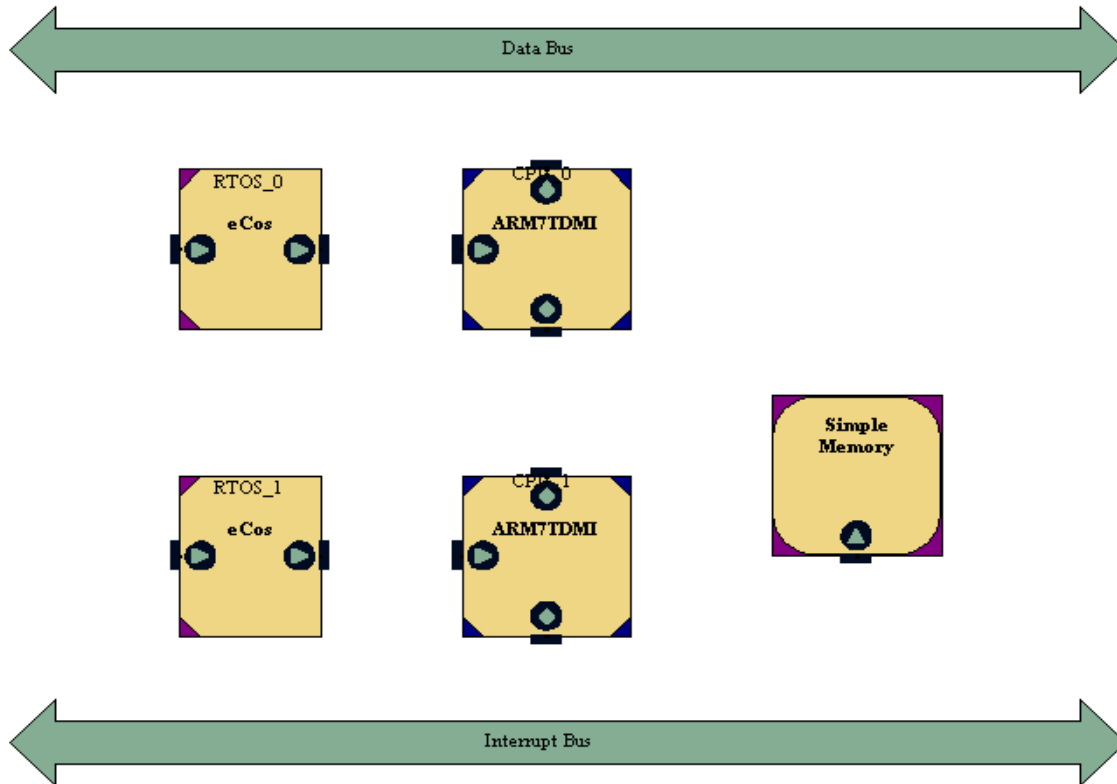
When you complete the wiring steps in this section, your diagram should look like this:



Tutorial 2: VCC Hierarchical Modeling Architecture Model

Organize the Blocks

1. Make sure your blocks are arranged similar to this diagram:



Wire the Blocks

To connect the blocks in this diagram, you will connect some of the ports to the data or interrupt bus. You will connect some of the ports using a scheduler assignment and others using a communication wire.

RTOS to CPU

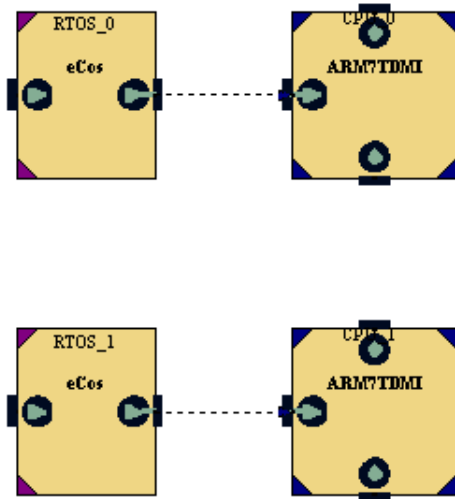
You will first connect *RTOS_0* to *CPU_0* and *RTOS_1* to *CPU_1*.

1. Choose *Architecture > Scheduler Assignment*.
2. Click the output port of *RTOS_0*, then click the input port of *CPU_0*.
3. Choose *Architecture > Scheduler Assignment*.
4. Click the output port of *RTOS_1*, then click the input port of *CPU_1*.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

The connections look like this:



This connection is not an ordinary communication wire; it is a scheduler assignment. A scheduler represents a scheduling policy for an architecture resource that governs how the behavior models assigned to the resource gain access to and share the resource.

Tutorial 2: VCC Hierarchical Modeling

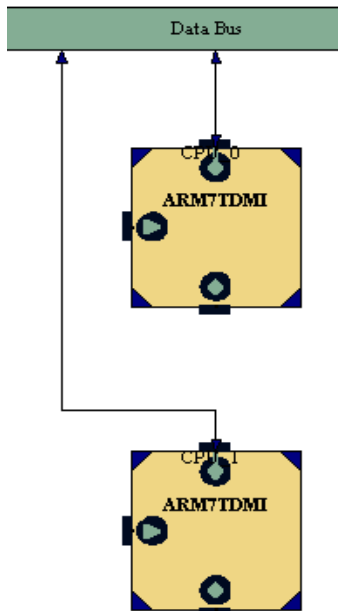
Architecture Model

CPU to Data Bus

You will connect both *CPU_0* and *CPU_1* to the Data Bus.

1. Choose *Architecture > Communication Wire*.
2. Click the upper bidirectional port of *CPU_0*, then click the edge of the Data Bus.
3. Choose *Architecture > Communication Wire*.
4. Click the upper bidirectional port of *CPU_1*, then click the edge of the Data Bus.

The connections look like this:



CPU to Interrupt Bus

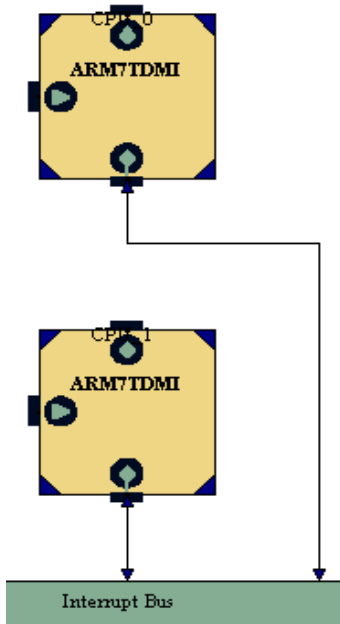
You will connect both *CPU_0* and *CPU_1* to the Interrupt Bus.

1. Choose *Architecture > Communication Wire*.
2. Click the lower bidirectional port of *CPU_0*, then click the edge of the Interrupt Bus.
3. Choose *Architecture > Communication Wire*.
4. Click the lower bidirectional port of *CPU_1*, then click the edge of the Interrupt Bus.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

The connections look like this:

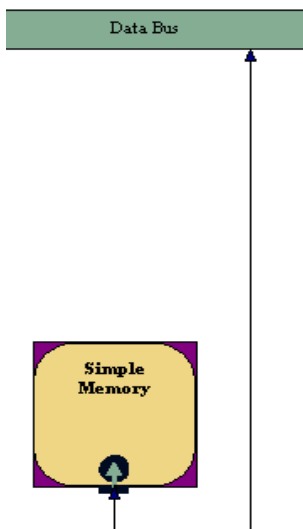


Simple Memory to Data Bus

You will connect the *Simple Memory* block to the Data Bus.

1. Choose *Architecture > Communication Wire*.
2. Click the input port of the Simple Memory, then click the edge of the Data Bus.

The connection looks like this:



Save the Diagram

- To save the architecture diagram, choose *File > Save* from the main menu.

Generate the Netlist

1. To generate a netlist (and discover any errors) for the architecture diagram, choose *Tools > Verify* from the main menu.
2. Ignore the following messages that the Child ports on both RTOS blocks are unconnected:

```
Warning: The port 'Child' is not connected.
{in /(Tutorial2.two_cpu:arch)RTOS_0.Child}:
Warning: The port 'Child' is not connected.
{in /(Tutorial2.two_cpu:arch)RTOS_1.Child}:
Warning: Some ports in the instance 'eCos[RTOS_0]' are not connected.
{in /(Tutorial2.two_cpu:arch)RTOS_0}:
Warning: Some ports in the instance 'eCos[RTOS_1]' are not connected.
{in /(Tutorial2.two_cpu:arch)RTOS_1}:
```

Successful netlist generation message:

```
Info: Code generation for Tutorial2.two_cpu:arch was successful.
```

Unsuccessful netlist generation message (caused by a missing parameter value):

```
Error: The parameter "<parameter name>" has no value.
{in /(Tutorial2.two_cpu:arch)RTOS_1}.
```

This error message is only a sample. You might see other types of messages, but the information in the message should help you correct your design.

3. Correct any errors and run *Tools > Verify* until the code generation is successful.

There are several ways to display and locate errors in your netlist generation. The following tips describe some of these methods.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

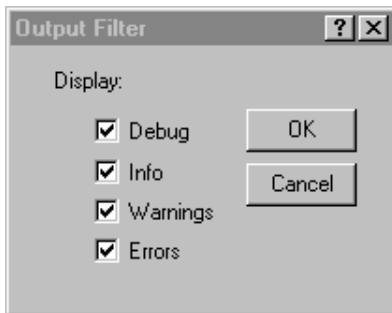
How to Locate Errors



Tip

To control the output of messages to the Netlist window, right click in the window and choose *Output Filters*.

The Output Filter dialog lets you choose which types of messages you want displayed in the Netlist window.



- *Debug* reports detailed information about what the simulator is doing.
- *Info* reports the block-by-block progress of the simulation.
- *Warnings* do not cause the netlist generation to fail.
- *Errors* cause the netlist generation to fail.



Tip

To open the object where an error occurred, double-click on the error in the Netlist window.

What if Your Netlist Won't Generate?

If you are unable to correct all the errors in your architecture model, you can use the example architecture model for your mapping diagram later in this tutorial.

The example architecture cellview is named *x_two_cpu*. When the instructions for the mapping diagram tell you to instantiate *two_cpu*, you can use *x_two_cpu*.

Tutorial 2: VCC Hierarchical Modeling

Architecture Model

Mapping Diagram

The mapping diagram maps the real-time controller behavior to the real-time controller architecture.

In this chapter of the tutorial, you will map the behavior you created in Chapter 2 to the architecture you created in Chapter 3.

To create this mapping diagram, you will do the following:

- [Specify Performance Views](#) on page 150
- [Create the Mapping Diagram](#) on page 158
- [Set Communication Patterns](#) on page 171
- [Bind Memory Segments to RAM](#) on page 175
- [Run a Simulation](#) on page 177
- [What if Your Model Won't Simulate?](#) on page 181

If you have problems with your mapping diagram that you can't resolve, instructions for using a golden copy from the database are in [What if Your Model Won't Simulate?](#) on page 181.

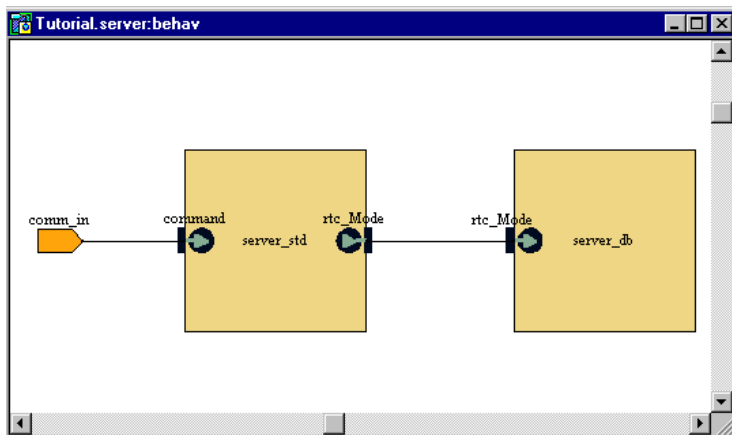
Specify Performance Views

Each of the behavior blocks must have at least one performance view.

Server Performance Views

You will open the *server* behavior diagram and specify performance views for each of the behavior blocks.

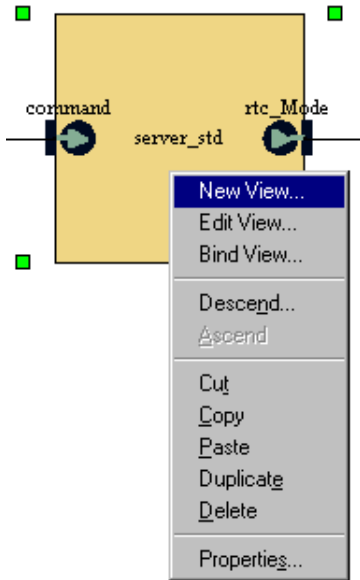
1. On the Project Folder, expand the *Tutorial2* library, then expand the *server* cell.
2. To open the *server* behavior diagram, double-click the *behav* icon.



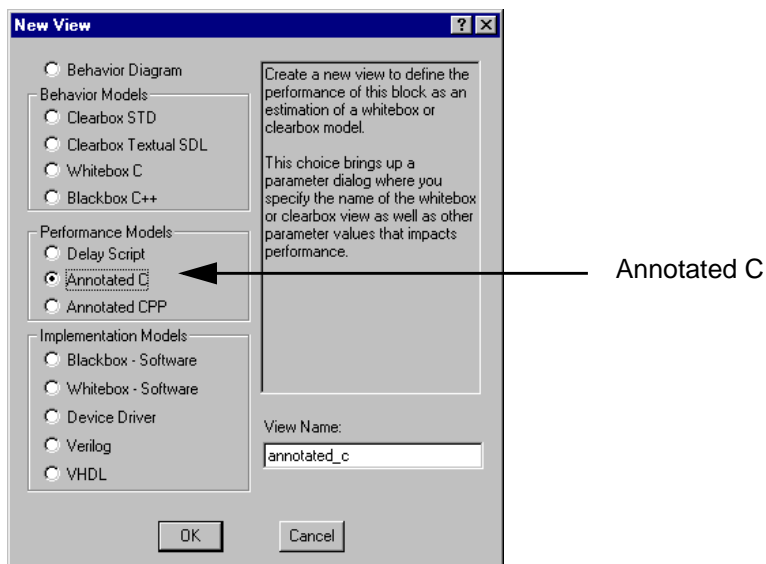
Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

server_std

1. Right click the *server_std* block and choose *New View* from the pop-up menu.



The New View dialog opens.



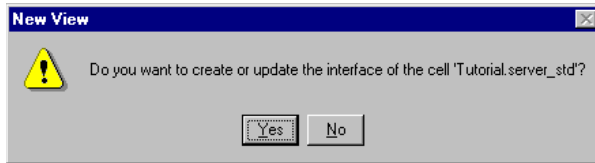
2. In the Performance Models section, select *Annotated C* and click *OK*.

The View Name defaults to *annotated_c*.

Tutorial 2: VCC Hierarchical Modeling

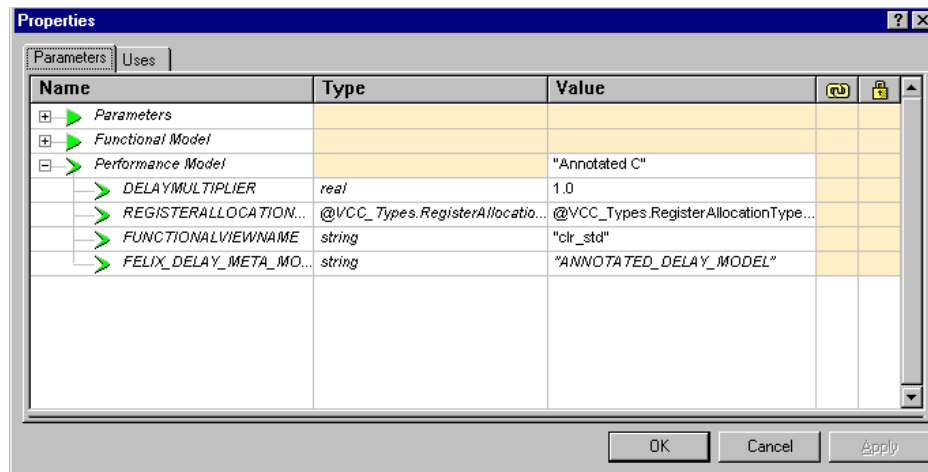
Mapping Diagram

The dialog asks if you want to update the *server_std* interface.



3. Click Yes in the dialog.

The Properties dialog opens.



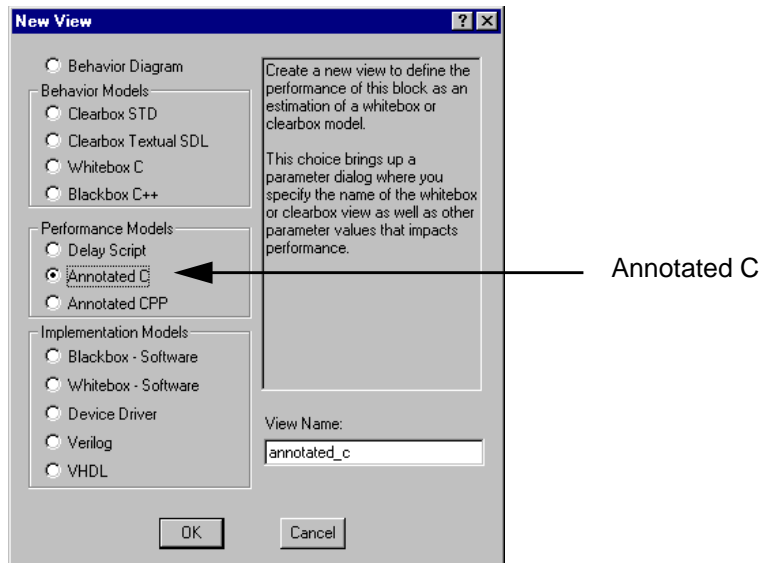
4. Click the plus sign to expand the *Performance Model* header.
5. Double-click the Value field for FUNCTIONALVIEWNAME and enter the value:
"clr_std"
You must enclose the string in quotes.
6. Click OK in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

server_db

1. Right click the *server_db* block and choose *New View* from the pop-up menu.

The New View dialog opens.



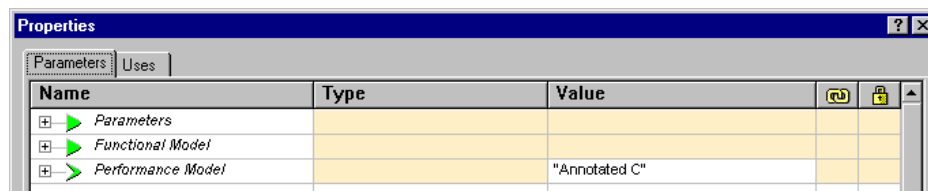
2. In the Performance Models section, select *Annotated C* and click *OK*.

The View Name defaults to *annotated_c*.

The dialog asks if you want to update the *server_db* interface.

3. Click *Yes* in the dialog.

The Properties dialog opens.



4. Accept the current values and click *OK* in the Properties dialog.

Save and Close

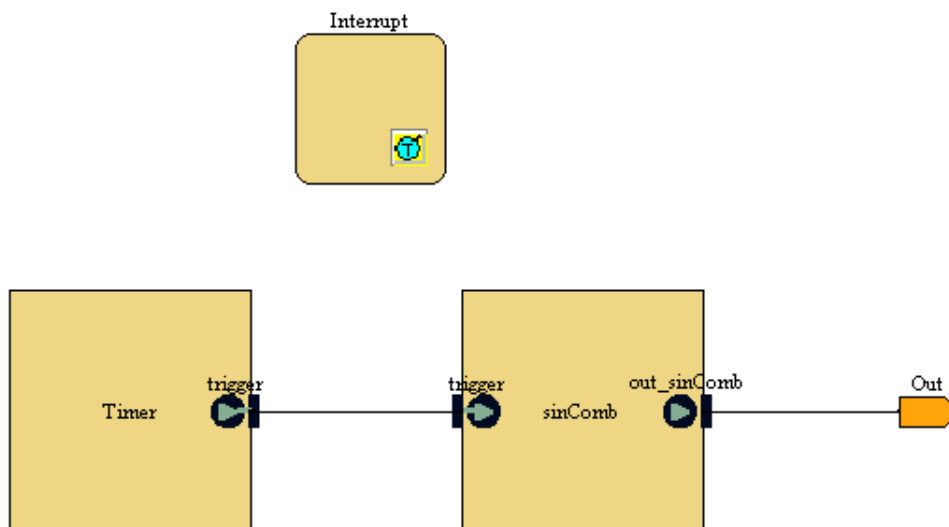
- To save and close the *server* diagram, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

Controller Performance Views

You will open the *controller* behavior diagram and specify performance views for each of the behavior blocks.

1. On the Project Folder, expand the *Tutorial2* library, then expand the *controller* cell.
2. To open the *controller* behavior diagram, double-click the *behav* icon.



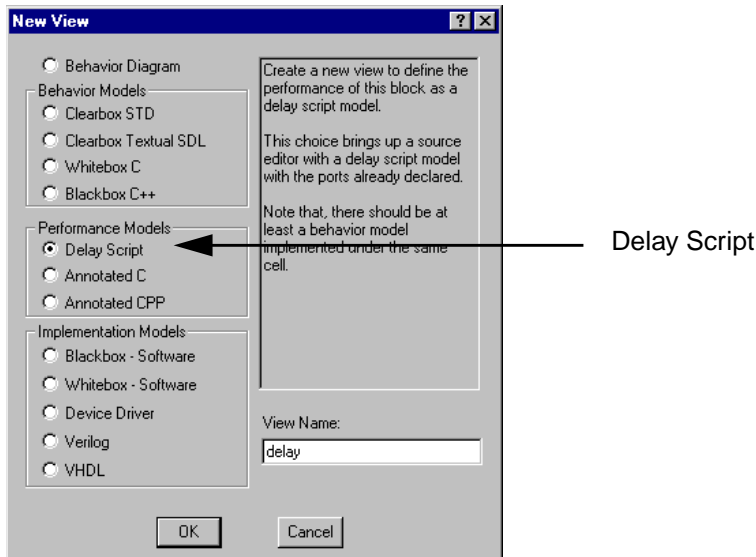
sinComb

The *sinComb* block has a delay script performance model.

1. Right click the *sinComb* block and choose *New View* from the pop-up menu.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

The New View dialog opens.

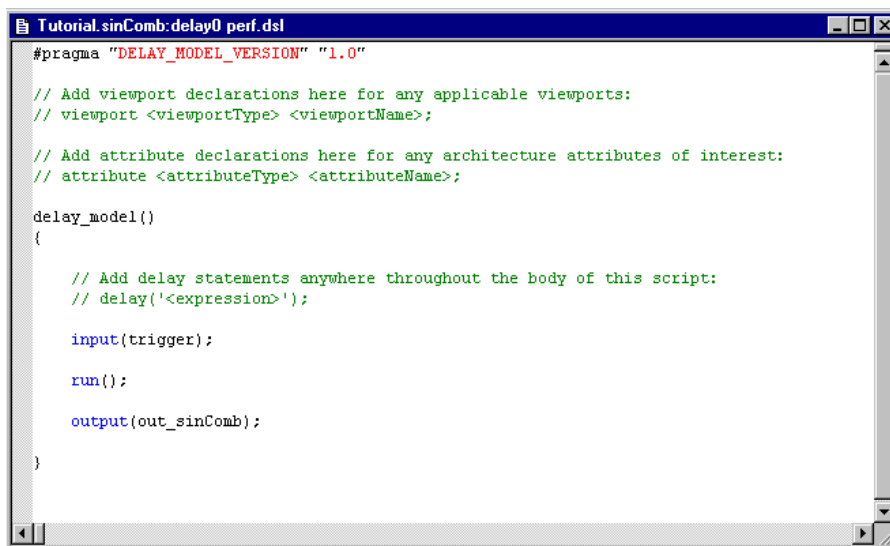


2. In the Performance Models section, select *Delay Script* and click *OK*.

The dialog asks if you want to update the *sinComb* interface.

3. Click *Yes* in the dialog.

The delay script (*perf.dsl*) file opens.



4. After the *Run* expression, enter the following delay expression:

```
delay('120e-6');
```

Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

Your delay script now looks like this:

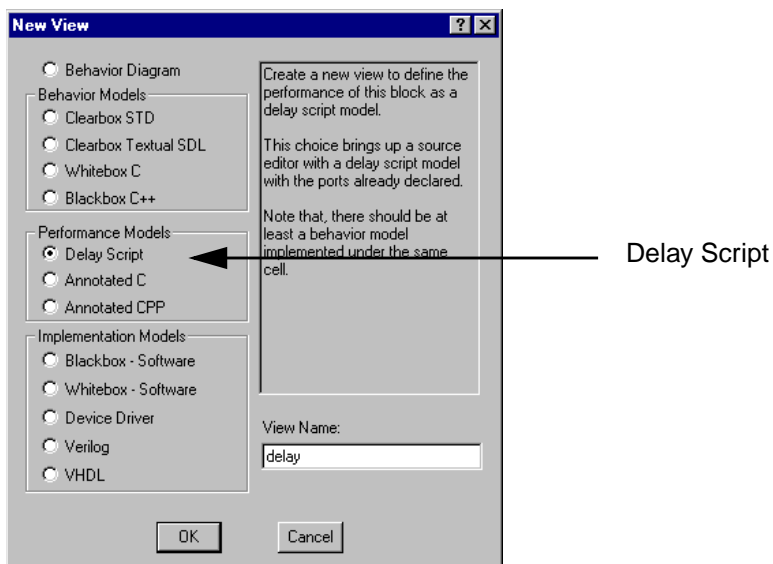
```
delay_model()  
{  
  
    // Add delay statements anywhere throughout the body of this script:  
    // delay('<expression>');  
  
    input(trigger);  
  
    run();  
    delay('120e-6');  
    output(out_sinComb);  
}
```

5. To save the *perf.dsl* file, choose *File > Save* from the main menu.
6. To close the *perf.dsl* file, choose *File > Close* from the main menu.

Timer

1. Right click the *Timer* block and choose *New View* from the pop-up menu.

The New View dialog opens.



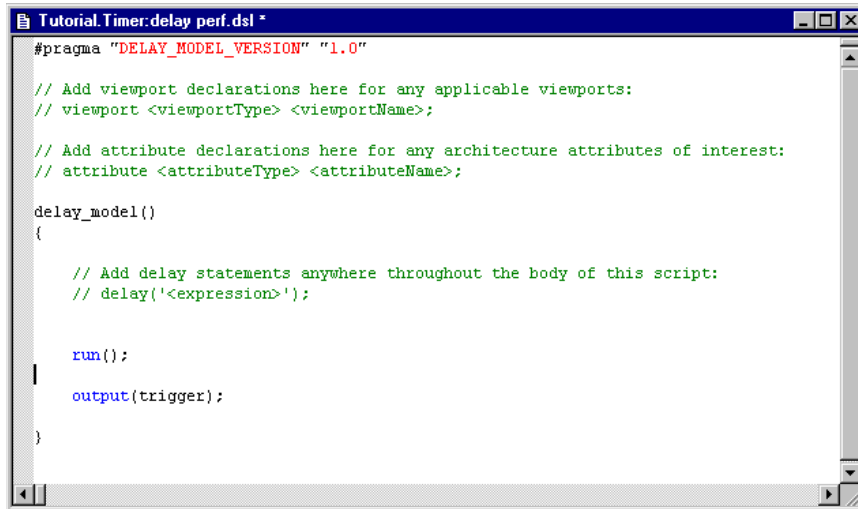
2. In the Performance Models section, select *Delay Script* and click *OK*.

The dialog asks if you want to update the interface of the *Tutorial2.Timer*.

3. Click *Yes* in the dialog.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

The delay script (*perf.dsl*) file opens.



4. After the *Run* expression, enter the following delay expression:

```
delay('15e-6');
```

Your delay script now looks like this:

```
delay_model()
{
    // Add delay statements anywhere throughout the body of this script:
    // delay('<expression>');

    run();
    delay('15e-6');
    output(trigger);
}
```

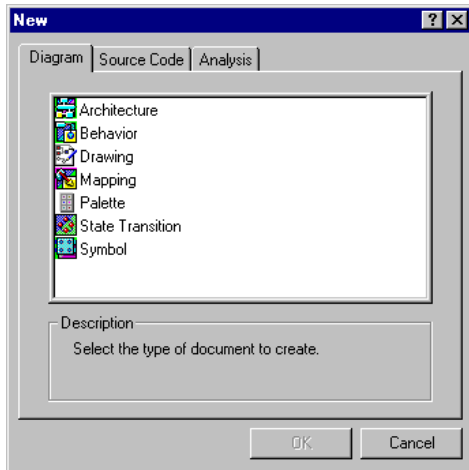
5. To save and close the *perf.dsl* file, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

Save and Close

- To save and close the *controller* diagram, choose *File > Save* from the main menu, then choose *File > Close* from the main menu.

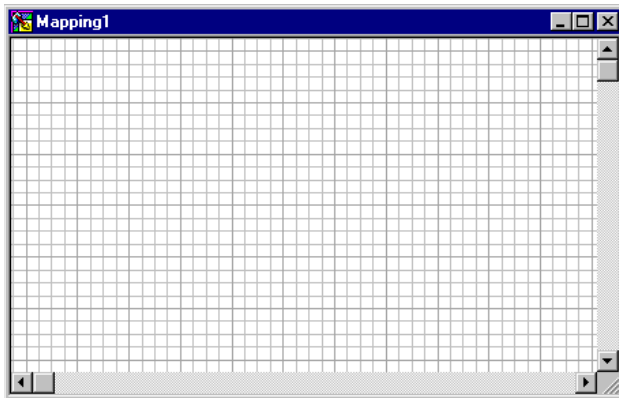
Create the Mapping Diagram

1. To open a new mapping diagram, choose *File > New* from the main menu.



2. Double-click the *Mapping* icon in the New dialog.

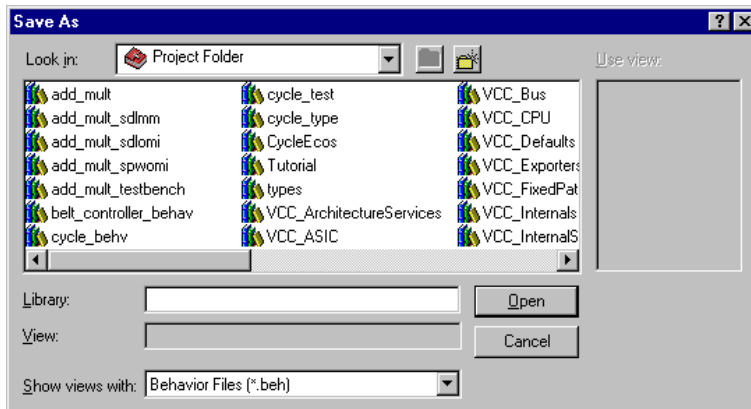
A new window opens. The default name is *Mapping1*.



3. To save the mapping diagram, choose *File > Save* from the main menu.

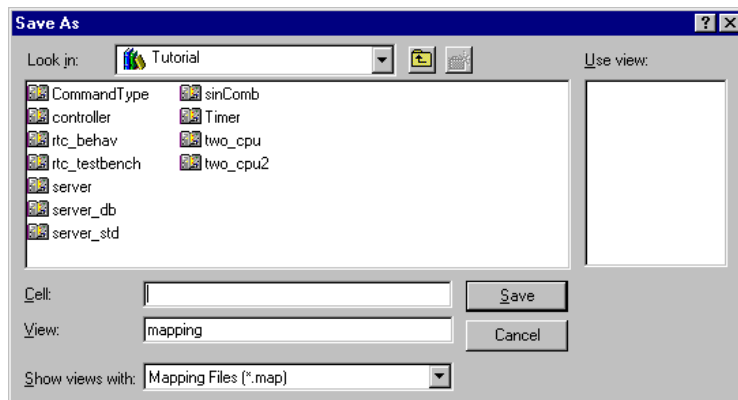
Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

VCC prompts you to choose a library.



4. To choose the *Tutorial2* library from the Project Folder, double-click the *Tutorial2* icon in the library list.

VCC prompts you to choose a cell name. The View name defaults to *mapping*.



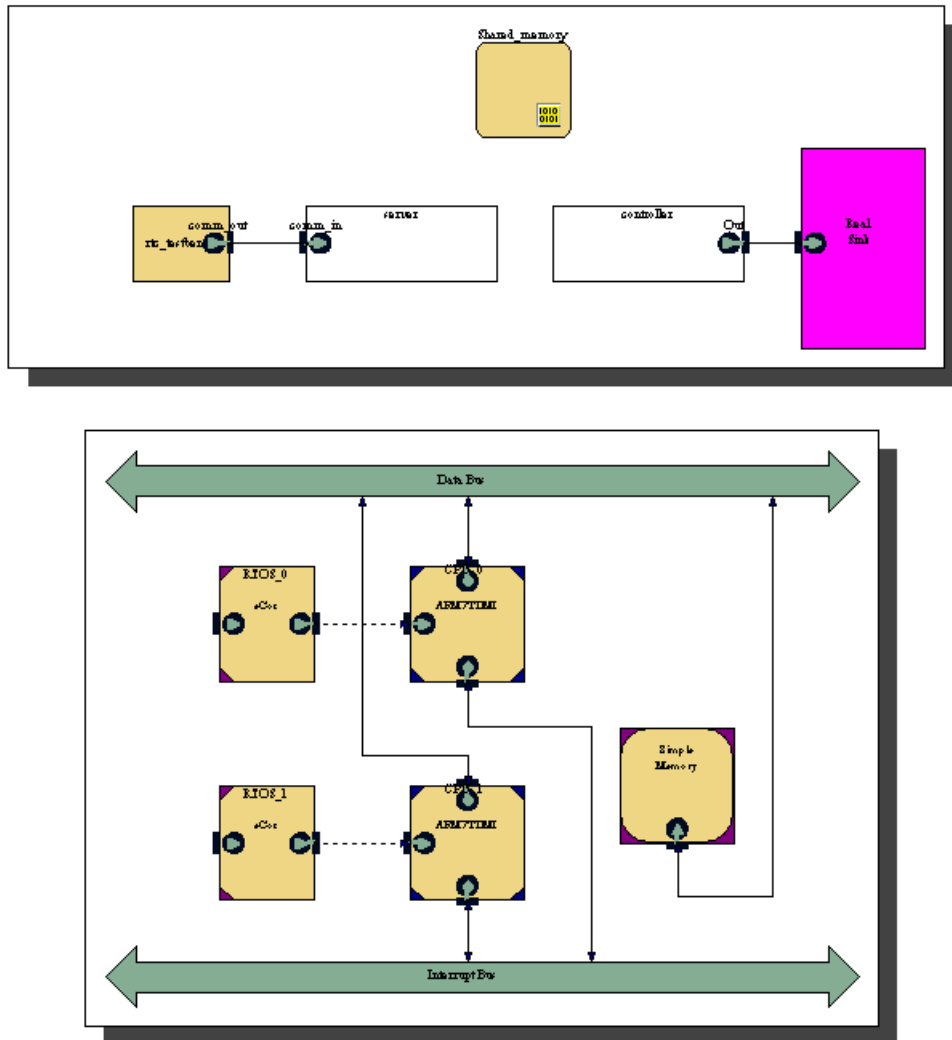
5. Enter the Cell name *rtc_map* and click *Save*.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

Instantiate Behavior and Architecture

1. Instantiate *Tutorial2.rtc_behav:behav* in your diagram.
2. Instantiate *Tutorial2.two_cpu:arch* in your diagram.

Your mapping diagram now looks like this:



Map the Behavior to the Architecture

Because you used lower-level blocks to make up the *server* and *controller*, the behavior is a hierarchical diagram. To map the lower-level behavior blocks to the architecture, you must expand the *server* and *controller* blocks.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

Expand the Hierarchy

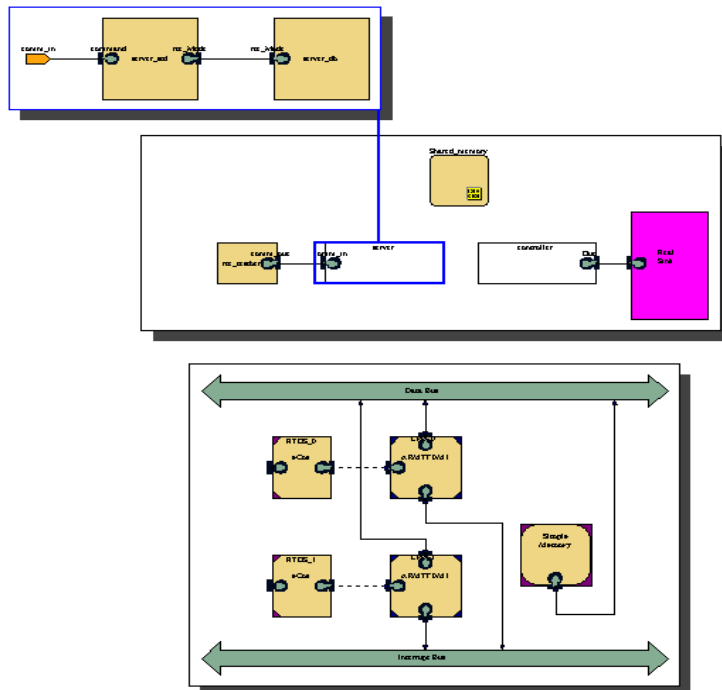
1. To expand the hierarchy of the *server*, select the *server* block in the behavior diagram, then choose *Mapping > Hierarchical Mapping* from the main menu.

When you return your cursor to the mapping diagram, you will see the expanded hierarchy of the server. You will also see a blue virtual connection between the server and its expanded diagram.

2. To place the expanded hierarchy, click in the upper left corner of the mapping diagram.

You can place this anywhere that doesn't interfere with the behavior or architecture diagrams at the top level.

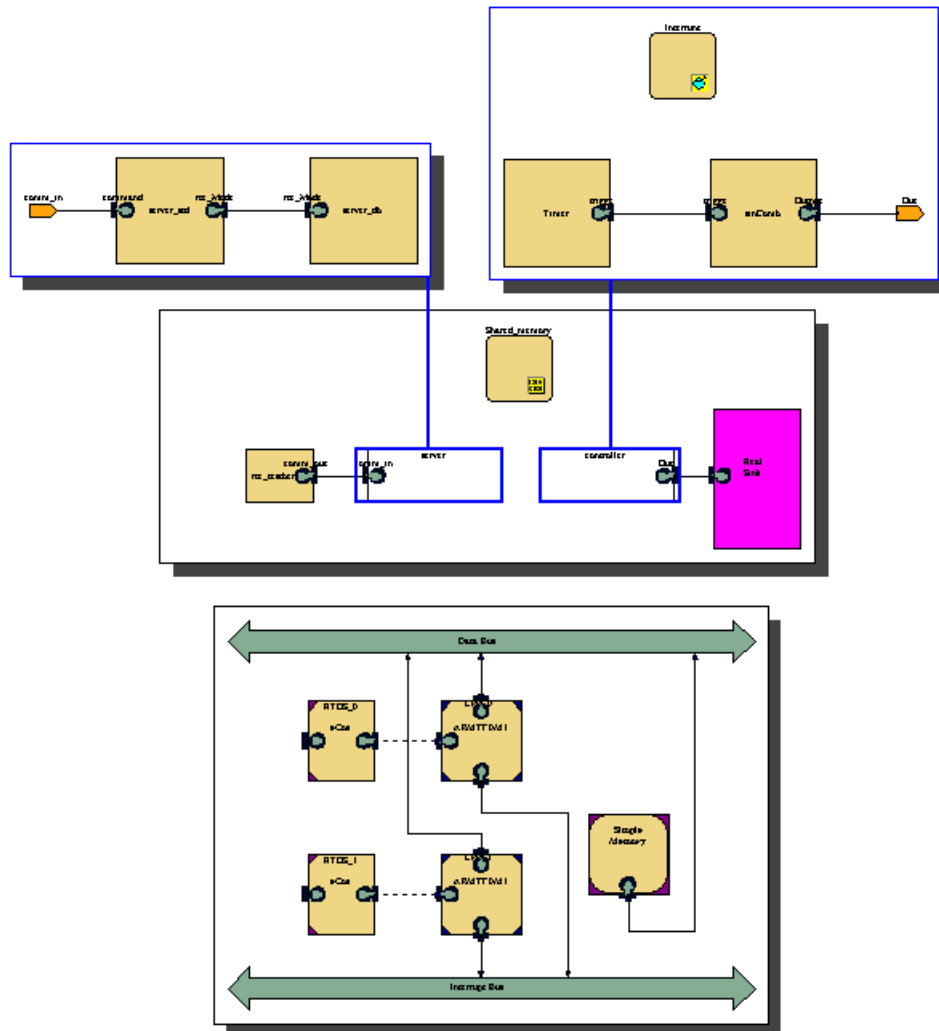
Your diagram now looks like this:



3. To expand the hierarchy of the *controller*, select the *controller* block in the behavior, then choose *Mapping > Hierarchical Mapping* from the main menu.
4. To place the expanded hierarchy, click in the upper right corner of the mapping diagram.

Tutorial 2: VCC Hierarchical Modeling

Your diagram now looks like this:



Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

Connect the Blocks

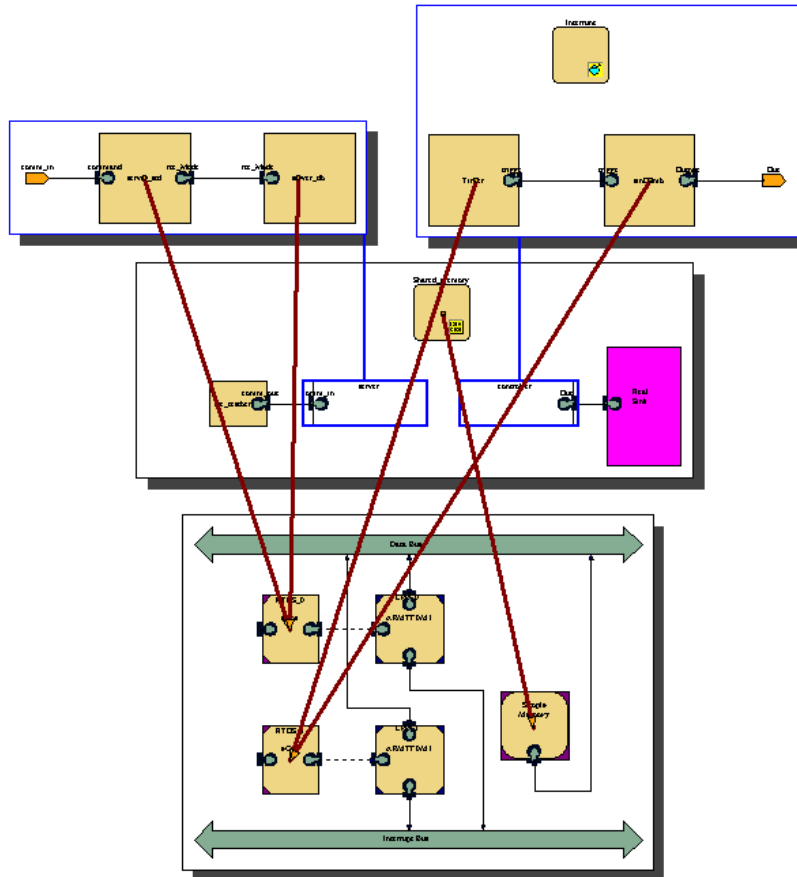
You will now connect the following behaviors to the specified architectures:

Behavior	Architecture
server_std	RTOS_0
server_db	RTOS_0
Shared Memory	RAM (Simple Memory)
Timer	RTOS_1
sinComb	RTOS_1

1. Choose *Mapping > Mapping Connection* from the main menu.
2. Click the specified block in the behavior diagram, then click the specified block in the architecture diagram.
3. Repeat these steps for all five connections.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

Your diagram now looks like this:



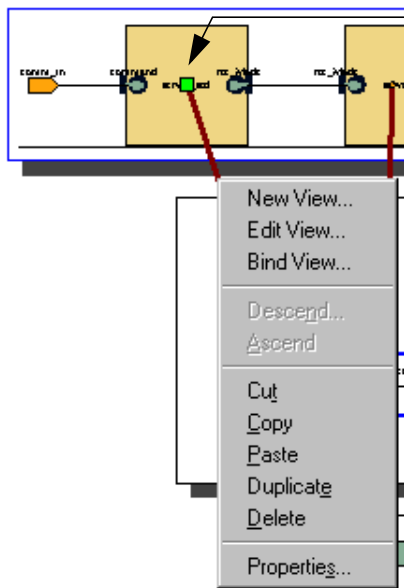
Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

Specify Performance Parameters

Each of the mapping connections you created in the previous section must have its performance parameters defined.

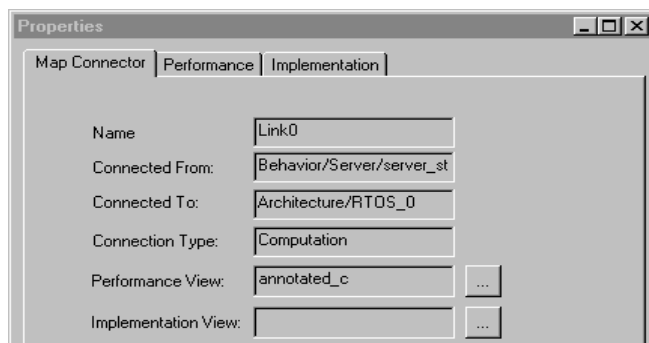
`server_std`

1. To specify the performance parameters for the mapping connection between the `server_std` and the `RTOS_0`, right click the connection and choose *Properties* from the pop-up menu.



The square markers at the end of the connection confirm that the connection wire is selected.

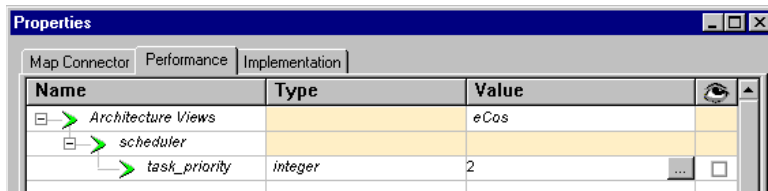
The Properties dialog opens.



2. Click the *Performance* tab.
3. Click the plus sign to expand the *Architecture Views* header.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 2.

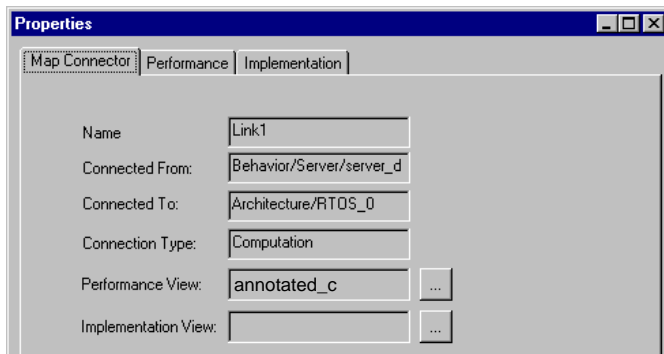


6. Click OK in the Properties dialog.

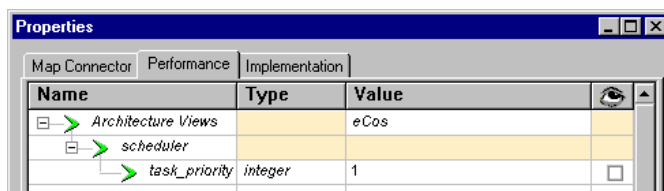
server_db

1. To specify the performance parameters for the mapping connection between the *server_db* and the *RTOS_0*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 1.



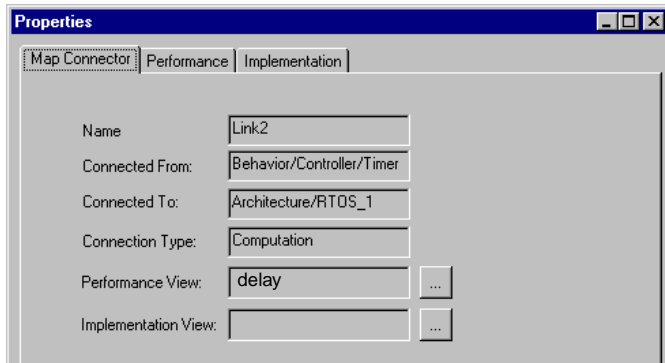
6. Click OK in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

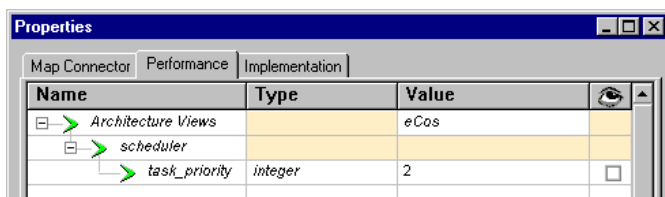
Timer

1. To specify the performance parameters for the mapping connection between the *Timer* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 2.



6. Click *OK* in the Properties dialog.

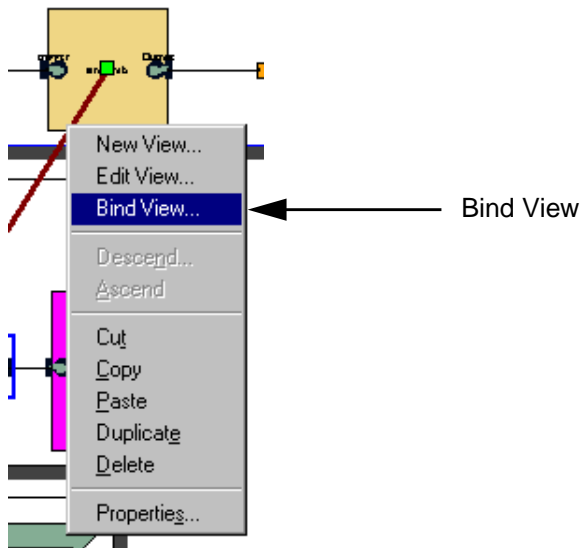
Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

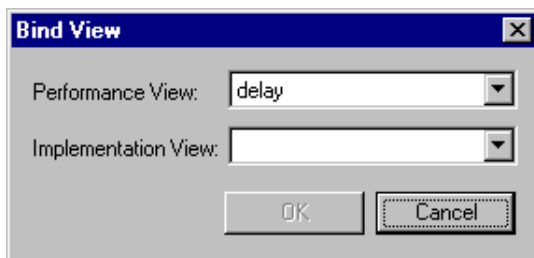
sinComb

The *sinComb* block has more than one performance view, so you will tell the system to use the *delay* view for this mapping connection.

1. To bind the mapping connection to the *delay* performance view, right click on the connection between *sinComb* and *RTOS_1* and choose *Bind View* from the pop-up menu.



The Bind View dialog opens.



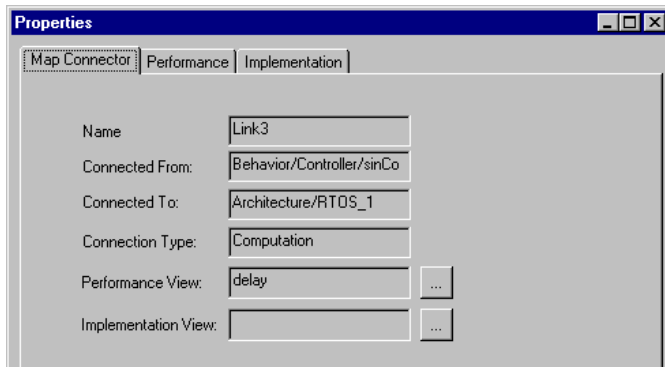
2. To bind the *delay* view, click the down arrow for Performance View, choose *delay*, then click *OK*.

Note: If the *delay* view is already displayed, you can click *Cancel* in the Bind View dialog.

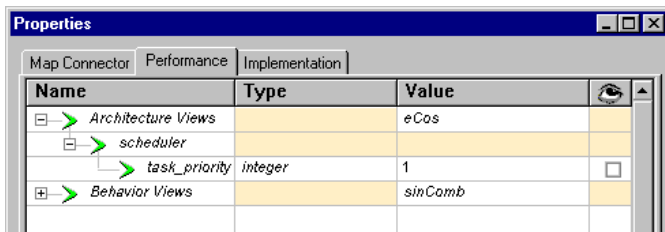
3. To specify the performance parameters for the mapping connection between the *sinComb* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

The Properties dialog opens.



4. Click the *Performance* tab.
5. Click the plus sign to expand the *Architecture Views* header.
6. Click the plus sign to expand the *scheduler* header.
7. Double-click the Value field for *task_priority* and enter 1.



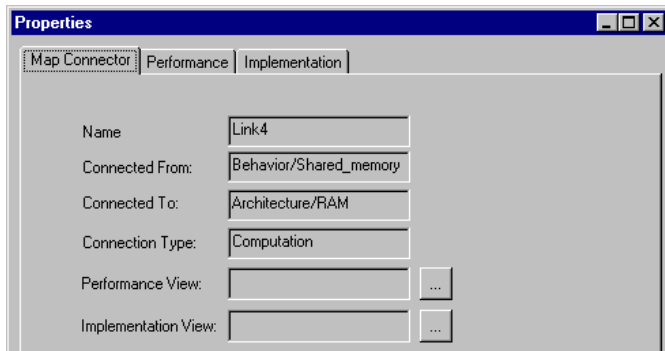
8. Click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

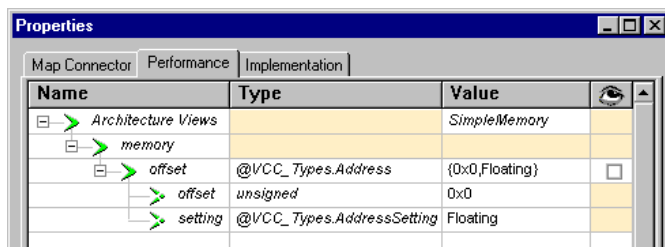
Shared_Memory

1. To specify the performance parameters for the mapping connection between the *Shared_Memory* and the *Simple Memory*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *memory* header.
5. Click the plus sign to expand the *offset* header.
6. Double-click the Value field for the *offset* parameter and enter
0x0
7. Double-click the Value field for the *setting* parameter, click the down arrow, and choose *Floating*.



The values you specified for the two *offset* parameters are automatically entered for the *offset* header.

This parameter lets VCC select a valid address.

8. Click *OK* in the Properties dialog.

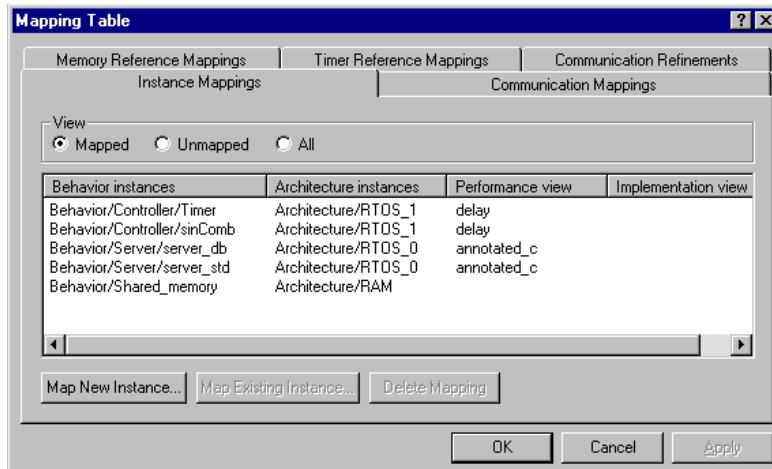
Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

Set Communication Patterns

To correctly model the delays on the communication links in the model, you must use communication patterns. You can automatically map these patterns from the Mapping Table.

1. To open the Mapping Table, choose *Mapping > Mapping Table* from the main menu.

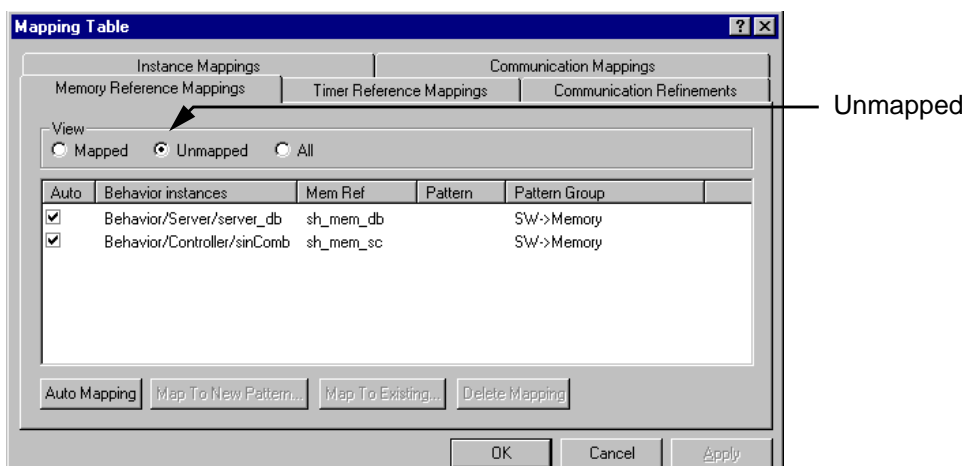


The Instance Mappings tab shows the five mapping connections you created at the top level of the mapping diagram.

2. Click the *Memory Reference Mappings* tab.

This tab defaults to mapped references, but you have no mapped references, so the field is blank.

3. On the Memory Reference Mappings tab, select *Unmapped*.

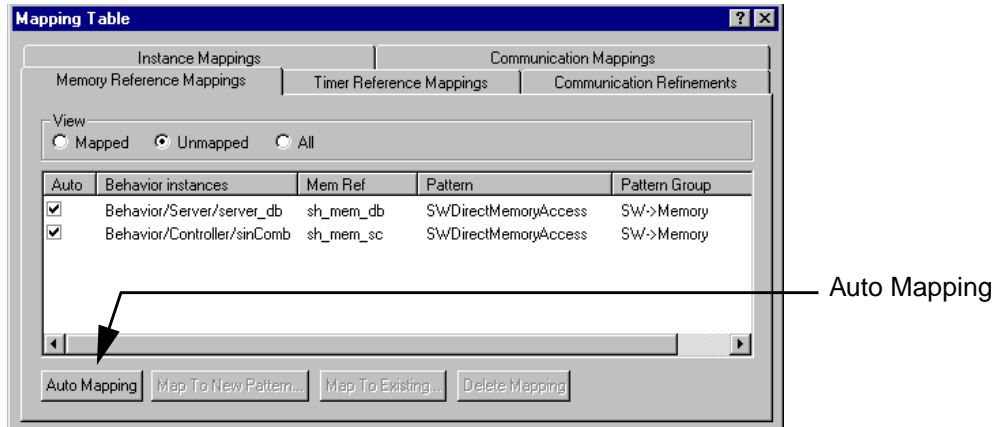


You can now see the unmapped references in your model.

Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

4. To automatically map the *server_db* and *sinComb* shared memory communication links, click *Auto Mapping*.

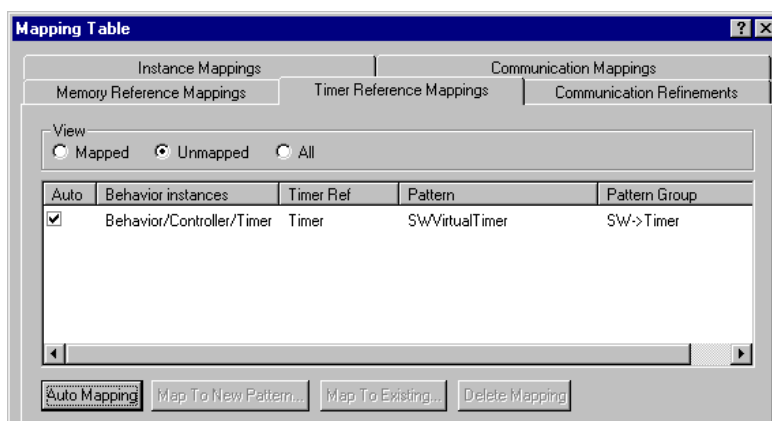


The system links these references to the *SWDirectMemoryAccess* communication pattern. (This will be an orange star on your mapping diagram.)

5. Now that these references are mapped, select *Mapped* so you can see them listed as mapped references.

Do not close the Mapping Table dialog.

6. Click the *Timer Reference Mappings* tab.
7. Select *Unmapped*.
8. To automatically map the *Timer* reference, click *Auto Mapping*.



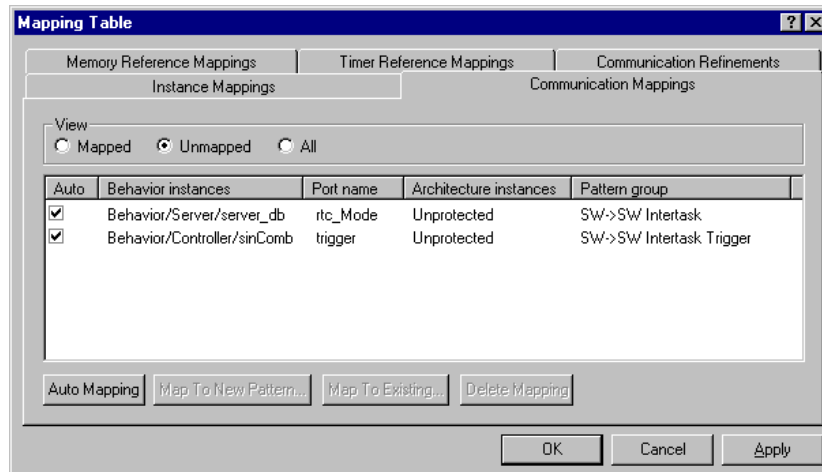
The system links this timer reference to the *SWVirtualTimer* communication pattern. (This will be an orange star on your mapping diagram.)

Do not close the Mapping Table dialog.

Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

9. Click the *Communication Mappings* tab.
10. Select *Unmapped*.
11. To automatically map the *rtc_Mode* and *trigger* ports, click *Auto Mapping*.



The system links these ports to the *Unprotected* architecture instance. (This will be a blue star on your mapping diagram.)

12. Now that these references are mapped, select *Mapped* so you can see them listed as mapped references.
13. To close the Mapping Table, click *OK*.

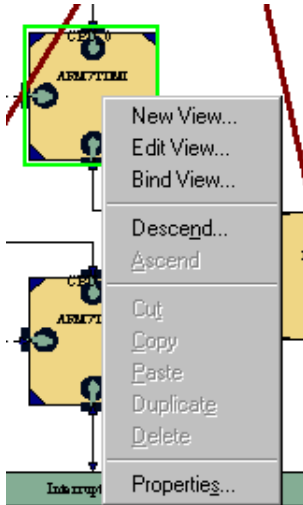
These mappings have added structures to the mapping diagram, but they might be placed outside your current window.

14. To view these new structures, choose *View > Fit Page* from the main menu.

Bind Memory Segments to RAM

You will bind the memory segments for the CPUs to the Simple Memory whose instance name is *RAM*.

1. Right click on *CPU_0* and choose *Properties* from the pop-up menu.



The Properties dialog opens.

2. Click the plus sign to expand the *stack* segment.
3. Click the plus sign to expand the *code* segment.
4. Click the plus sign to expand the *bss* segment.

Each of these segments contains a parameter named *mem*.

5. For each of the *mem* parameters, enter the value

"RAM"

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

The value must include the beginning and ending quotations.

Name	Type	Value
stack	LinkerSegmentInfo	@VCC_Types.LinkerSegmentInfo{@VCC_Types.LinkerSegmentType::STACK, ...}
type	LinkerSegmentType	@VCC_Types.LinkerSegmentType::STACK
mem	MemoryParticipant	"RAM"
offset	Address	@VCC_Types.Address(0x0,@VCC_Types.AddressSetting::Floating)
size	SegmentSizeEstimate	@VCC_Types.SegmentSizeEstimate(1024,True)
cacheable	Boolean	True
code	LinkerSegmentInfo	@VCC_Types.LinkerSegmentInfo{@VCC_Types.LinkerSegmentType::CODE, ...}
type	LinkerSegmentType	@VCC_Types.LinkerSegmentType::CODE
mem	MemoryParticipant	"RAM"
offset	Address	@VCC_Types.Address(0x0,@VCC_Types.AddressSetting::Floating)
size	SegmentSizeEstimate	@VCC_Types.SegmentSizeEstimate(336,True)
cacheable	Boolean	True
bss	LinkerSegmentInfo	@VCC_Types.LinkerSegmentInfo{@VCC_Types.LinkerSegmentType::DATA, ...}
type	LinkerSegmentType	@VCC_Types.LinkerSegmentType::DATA
mem	MemoryParticipant	"RAM"
offset	Address	@VCC_Types.Address(0x0,@VCC_Types.AddressSetting::Floating)
size	SegmentSizeEstimate	@VCC_Types.SegmentSizeEstimate(112,True)

stack segment
mem parameter

code segment
mem parameter

bss segment
mem parameter

6. To close the Properties dialog, click **OK**.

7. Right click on *CPU_1* and choose *Properties* from the pop-up menu.

The Properties dialog opens.

8. Click the plus sign to expand the *stack* segment.

9. Click the plus sign to expand the *bss* segment.

Each of these segments contains a parameter named *mem*.

10. For each of the *mem* parameters, set the value to

"RAM"

The value must include the beginning and ending quotations.

Name	Type	Value
stack	LinkerSegmentInfo	@VCC_Types.LinkerSegmentInfo{@VCC_Types.LinkerSegment...}
type	LinkerSegmentType	@VCC_Types.LinkerSegmentType::STACK
mem	MemoryParticipant	"RAM"
offset	Address	@VCC_Types.Address(0x0,@VCC_Types.AddressSetting::Floa...
size	SegmentSizeEstimate	@VCC_Types.SegmentSizeEstimate(1024,True)
cacheable	Boolean	True
bss	LinkerSegmentInfo	@VCC_Types.LinkerSegmentInfo{@VCC_Types.LinkerSegment...}
type	LinkerSegmentType	@VCC_Types.LinkerSegmentType::DATA
mem	MemoryParticipant	"RAM"
offset	Address	@VCC_Types.Address(0x0,@VCC_Types.AddressSetting::Floa...
size	SegmentSizeEstimate	@VCC_Types.SegmentSizeEstimate(100,True)

stack segment
mem parameter

bss segment
mem parameter

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

11. To close the Properties dialog, click *OK*.

Run a Simulation

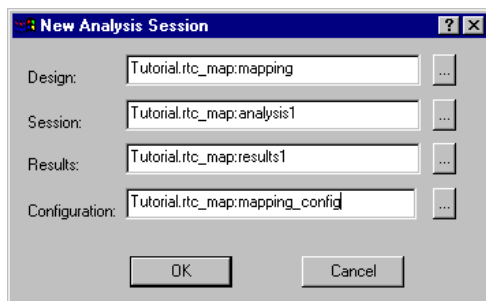
1. To create a new analysis session, choose *File > New* from the main menu.



2. Click the *Analysis* tab.

3. Double-click the *Analysis* icon in the New dialog.

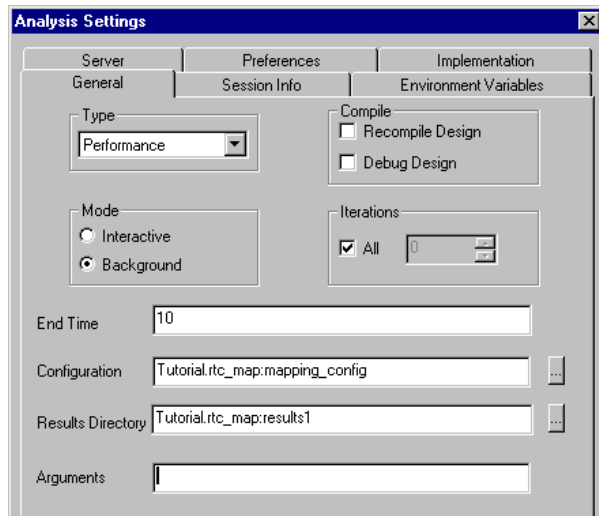
The New Analysis Session dialog opens.



4. Accept the default values and click *OK*.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

5. To specify the analysis settings, choose *Analysis > Settings* from the main menu.

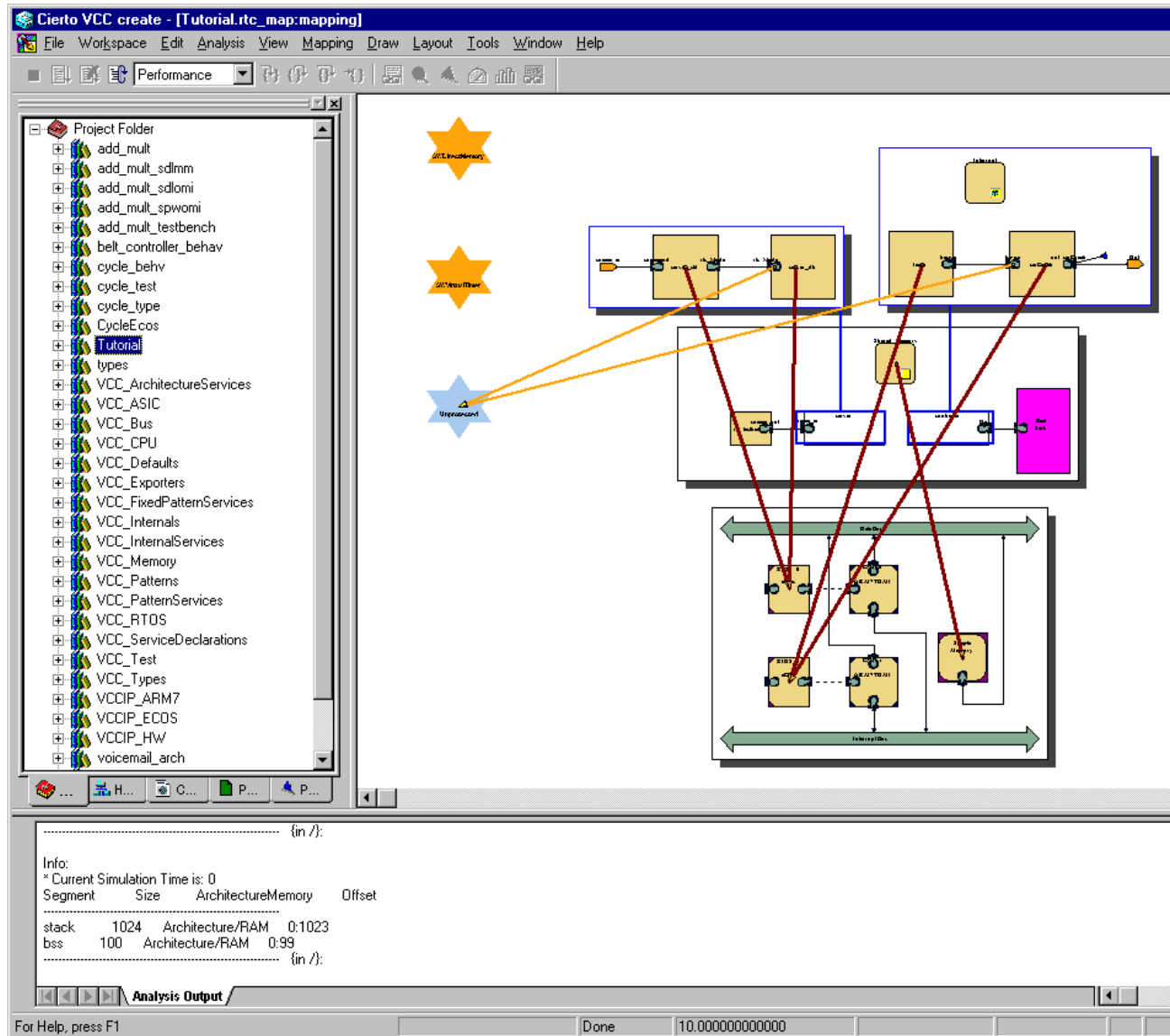


6. Set Type to *Performance*.
7. Set Mode to *Background*.
8. Set End Time to *10*.
9. Click *OK* in the Analysis Settings dialog.

A dialog asks if you want to “Save session changes?”
10. Click *Yes* in the Analysis Cockpit dialog.
11. Choose *Analysis > Go/Continue* from the main menu.

Tutorial 2: VCC Hierarchical Modeling Mapping Diagram

VCC reports the progress in the Analysis Output window at the bottom of the screen.



There are several ways to display and locate errors in your analysis output. The following tips describe some of these methods.

Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

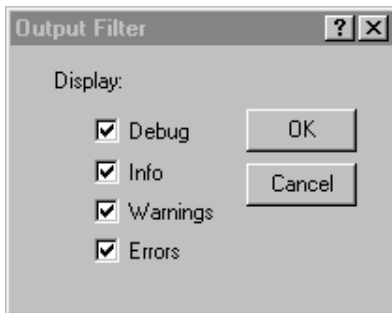
How to Locate Errors



Tip

To control the output of messages to the Analysis Output window, right click in the window and choose *Output Filters*.

The Output Filter dialog lets you choose which types of messages you want displayed in the Analysis Output window.



- *Debug* reports detailed information about what the simulator is doing
- *Info* reports the block-by-block progress of the simulation
- *Warnings* do not cause the simulation to fail
- *Errors* cause the simulation to fail



Tip

To open the object where an error occurred, double-click on the error in the Analysis Output window.



Tip

To view the output of the analysis, open the following directory:

Windows NT: C:\MyWorkspace\Tutorial2\rtc_map\results

Solaris or HP-UX: <install directory>/myworkspace/Tutorial2/rtc_map/results

This directory contains several error, debugging, and information files.

What if Your Model Won't Simulate?

If you are unable to correct all the errors in your mapping diagram, you can use the example mapping diagram named *x_rtc_map:mapping*.

Tutorial 2: VCC Hierarchical Modeling

Mapping Diagram

Analysis

To perform analysis on your design, you can do any of the following:

- [Simulate using Probes](#) on page 184
- [Simulate using Sweep Parameters](#) on page 195
- [Simulate using a Single-CPU Architecture](#) on page 202

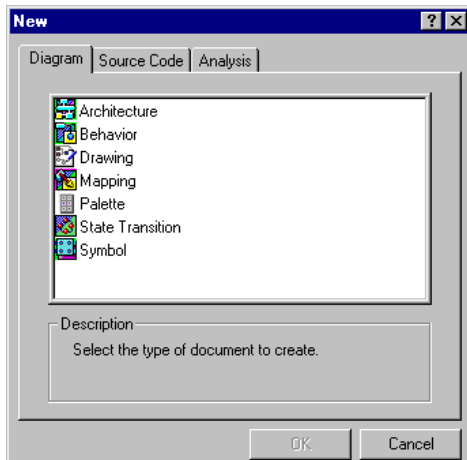
Each of these analysis sessions can be independently created. You can also combine these exercises. For example, you can run a single simulation that uses sweep parameters on a single-CPU architecture and sample the output data with a probe.

Simulate using Probes

To help you check for correct behavior in the model, you can add probes. Because probes are part of an analysis session, not components of the mapping diagram, you must begin an analysis session before you add a probe. After you add probes, you continue the analysis session. In this exercise, you will add a probe to the output of the behavior diagram.

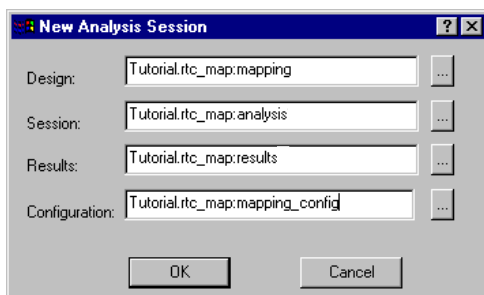
Begin an Analysis Session

1. In the Project Folder, open the *rtc_map* mapping diagram.
2. To create a new analysis session, choose *File > New* from the main menu.



3. Click the *Analysis* tab.
4. Double-click the *Analysis* icon in the New dialog.

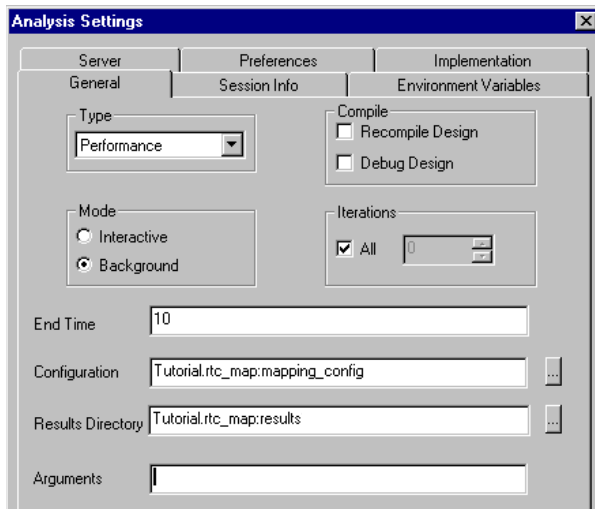
The New Analysis Session dialog opens.



5. Accept the default values and click *OK*.

Tutorial 2: VCC Hierarchical Modeling Analysis

6. To specify the analysis settings, choose *Analysis > Settings* from the main menu.

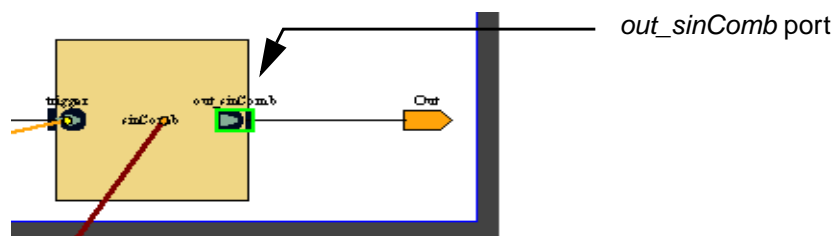


7. Set Type to *Performance*.
8. Set Mode to *Background*.
9. Set End Time to *10*.
10. Click *OK* in the Analysis Settings dialog.

Add a Generic Probe to the *controller* Block

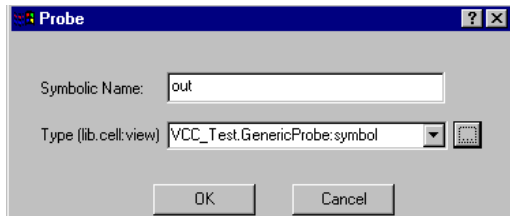
You will add a probe to the *out_sinComb* output port of the *sinComb* block in the behavior diagram.

1. Select the output port on the *sinComb* block, then choose *Analysis > Insert Probe* from the main menu.



Tutorial 2: VCC Hierarchical Modeling Analysis

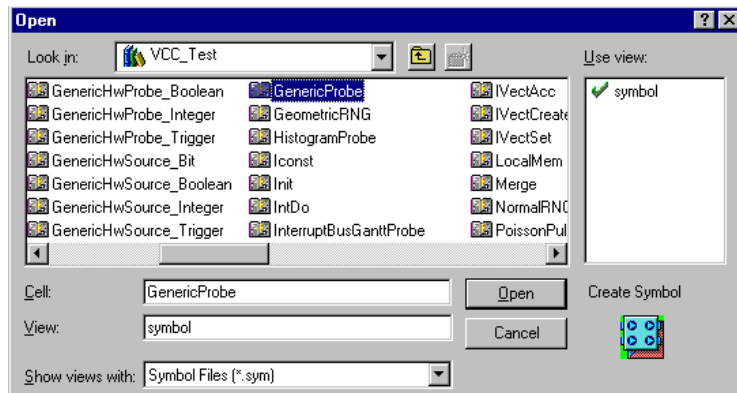
The Probe dialog opens.



2. Enter the Symbolic Name *out*.
3. To enter the Type (lib.cell:view), click the browser (...) button.

The browser opens to your Project Folder.

4. Scroll to *VCC_Test* and double-click the icon.
5. Select the *GenericProbe* cell and click *Open*.



The View name defaults to *symbol*.

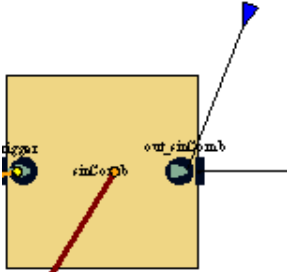
6. Click *OK* in the Probe dialog.

The graphical representation of the probe is quite small and hard to see because it sits on top of the port. Immediately after you create this probe, you can see the green markers that indicate the probe is selected.

7. Grab the selected probe and drag it away from the port.

Tutorial 2: VCC Hierarchical Modeling Analysis

You can now see the blue arrow that represents the probe.



Tip

If you don't see this probe, refresh your view using any of the View menu commands.

Run a Simulation

Note: When the session runs in *Background* mode, you don't have to explicitly run the *Analysis > Initialize* command.

1. Choose *Analysis > Go/Continue* from the main menu.

A dialog asks if you want to save the session changes.

2. Click *Yes* in the dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

Analyze the Simulation

When the simulation completes, the Analysis Output window reports:

Info: Done.

Info:

* Current Simulation Time is: 0

Segment	Size	ArchitectureMemory	Offset
stack	1024	Architecture/RAM	0:1023
bss	112	Architecture/RAM	0:111
code	336	Architecture/RAM	0:335

Info:

* Current Simulation Time is: 0

Segment	Size	ArchitectureMemory	Offset
stack	1024	Architecture/RAM	0:1023
bss	100	Architecture/RAM	0:99

You can now view the output data collected at the probe.

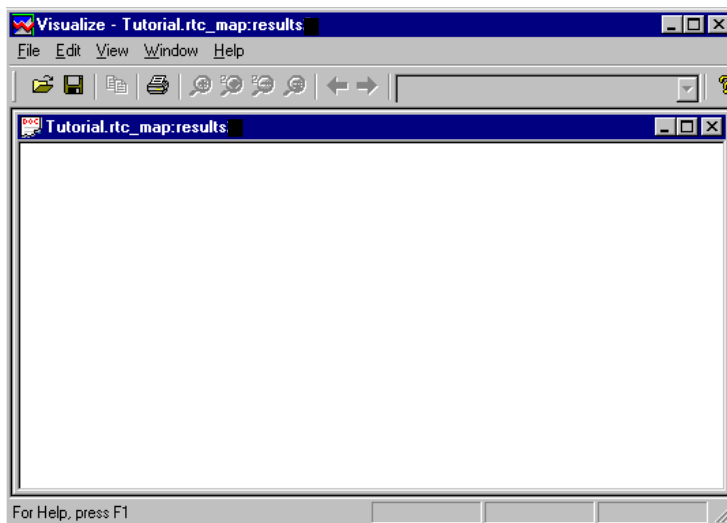
1. Choose *Analysis > Chart Results* from the main menu.

The dialog informs you that

charts file has zero charts - run File -> New Chart to get started.

2. Click *OK* in the dialog.

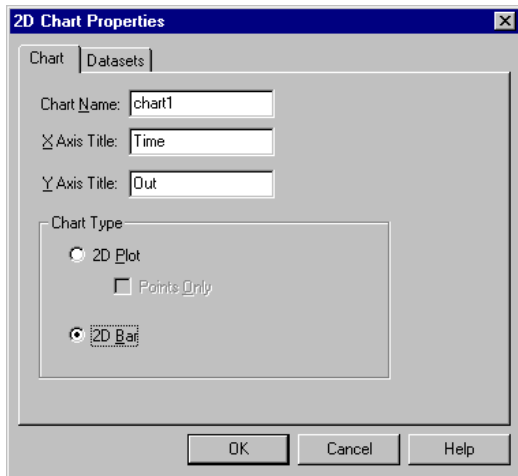
The Visualize tool opens with an empty results window.



3. From the Visualize main menu, choose *File > New Chart > 2D*.

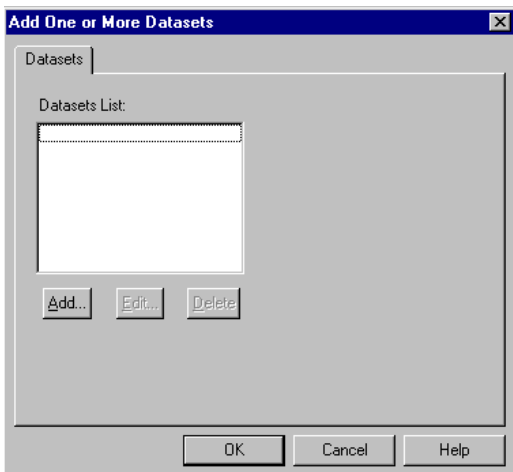
Tutorial 2: VCC Hierarchical Modeling Analysis

The New 2D Chart dialog opens.



4. Enter the Chart Name *chart1*.
5. Enter the X Axis Title *Time*.
6. Enter the Y Axis Title *Out*.
7. Set the Chart Type to 2D Bar.
8. Click *OK* in the New 2D Chart dialog.

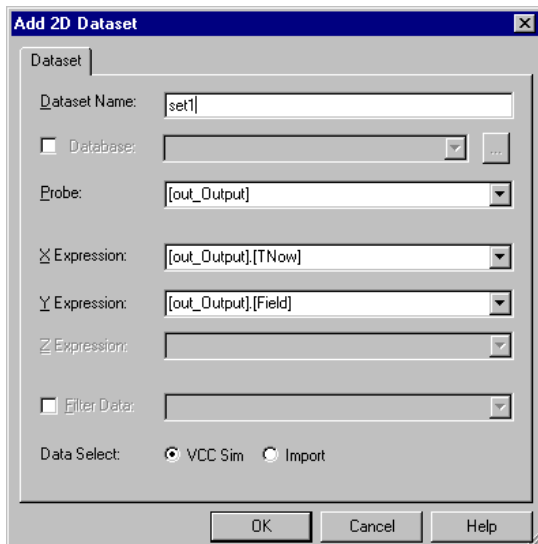
The Add One or More Datasets dialog opens.



9. Click *Add*.

Tutorial 2: VCC Hierarchical Modeling Analysis

The Add 2D Dataset dialog opens.



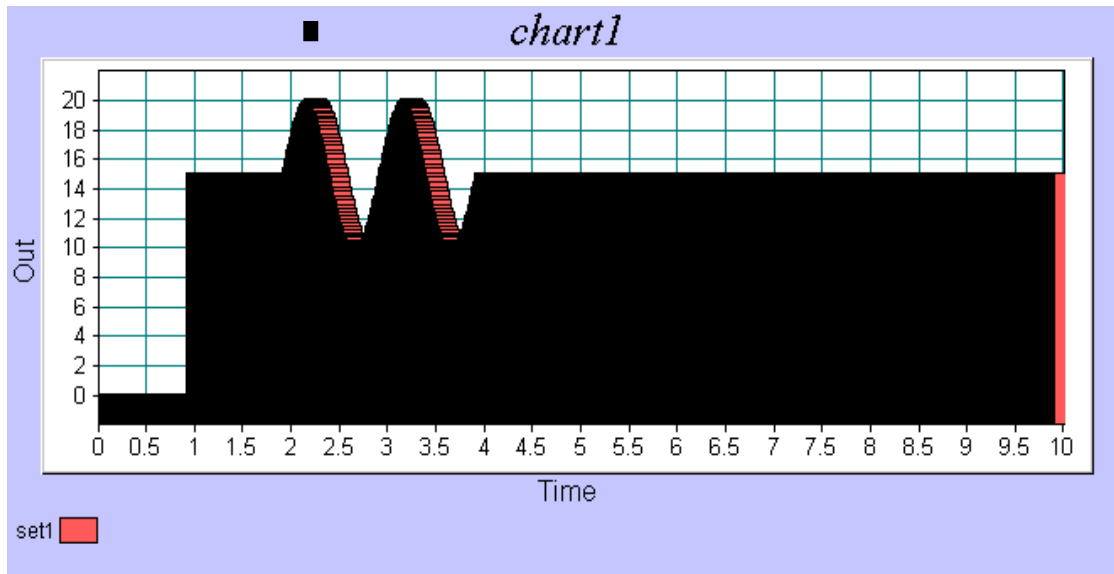
10. Leave the Dataset Name as *set1*.
11. To set the Probe value, click the down arrow and choose *[out_Output]*.
12. To set the X Expression value, click the down arrow and choose *[out_Output].[TNow]*.
13. To set the Y Expression value, click the down arrow and choose *[out_Output].[Field]*.
14. When you have entered all these values, click *OK* in the Add 2D Dataset dialog.

VCC returns you to the Add One or More Datasets dialog.

15. Click *OK* to close the Add One or More Datasets dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

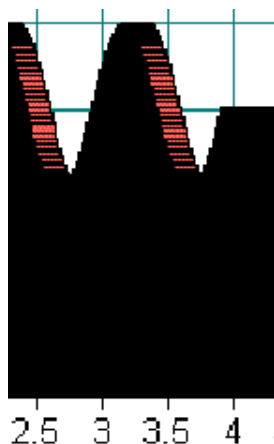
The output data collected at the probe is displayed in the chart.



The time axis on this chart displays the entire 10 seconds of the simulation. The data is so compressed that it looks like one horizontal bar.

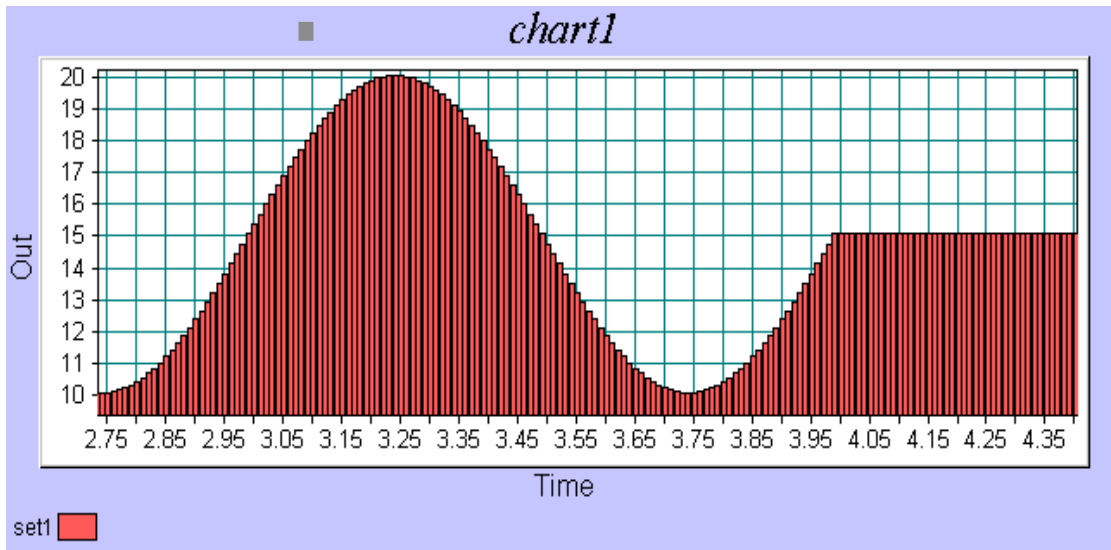
16. To view more detail in a smaller segment of time, draw a box around a small section of the data.

Make sure the bottom of your box does not extend past the horizontal axis. The entire box must be inside the X and Y axes.



Tutorial 2: VCC Hierarchical Modeling Analysis

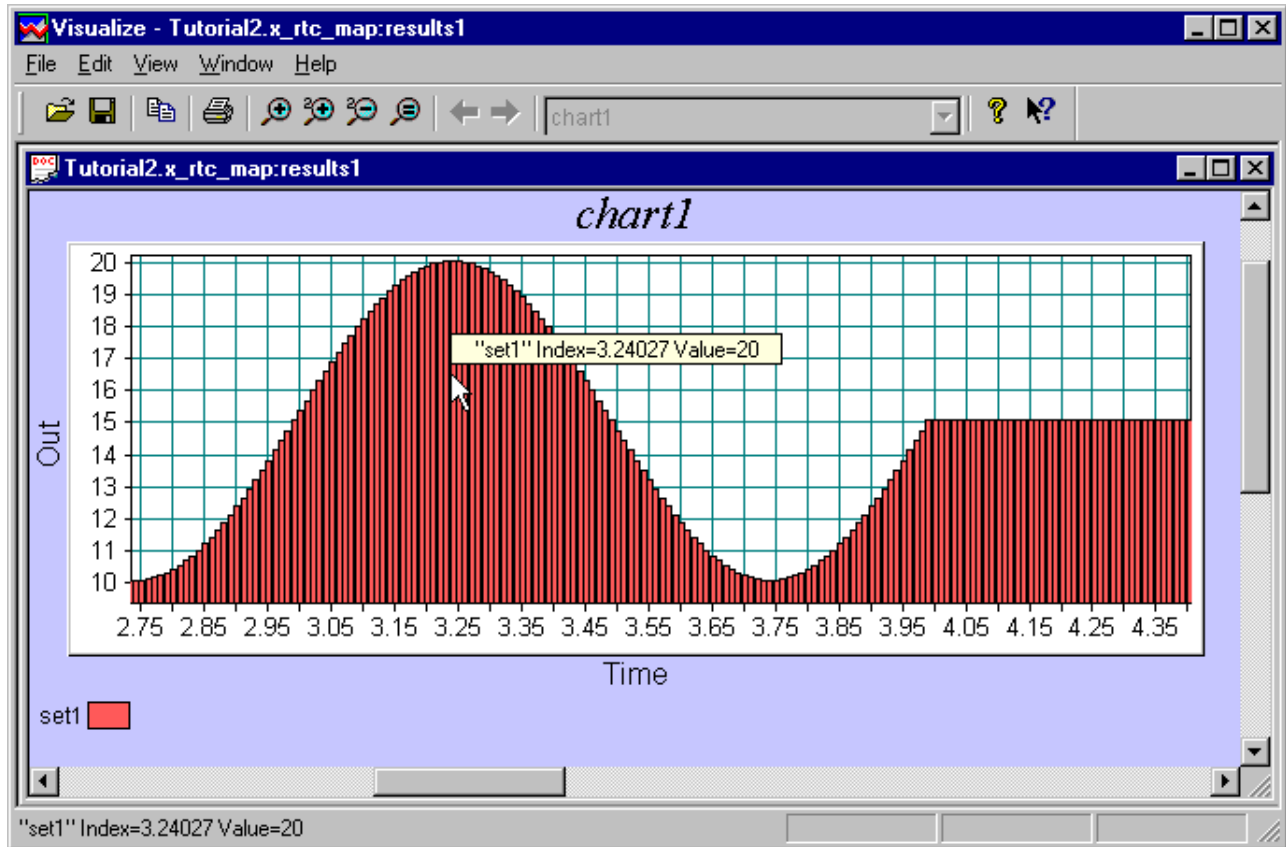
The display zooms in on this section of the bar graph.



You can see in this example that the output occurs every 1/100th of a second. This agrees with the 0.01 sampling time you specified for the controller.

Tutorial 2: VCC Hierarchical Modeling Analysis

17. To view the Index and Value for any individual data point, place your cursor over the bar.



The Index and Value for the data point are displayed in the lower left corner of the Visualize window and in a pop-up on the chart.

Note: If you want to use this same probe for any of the other analysis exercises, keep this analysis session open and continue to use it for the other exercises.

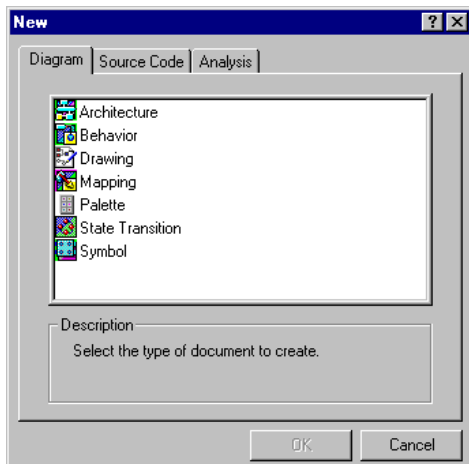
Simulate using Sweep Parameters

For this simulation, you will define the *samplingTime* parameter to sweep through multiple values during the session.

Note: If you are using the same analysis session you created in either of the other exercises, skip to [“Create the samplingTime Parameter”](#) on page 196.

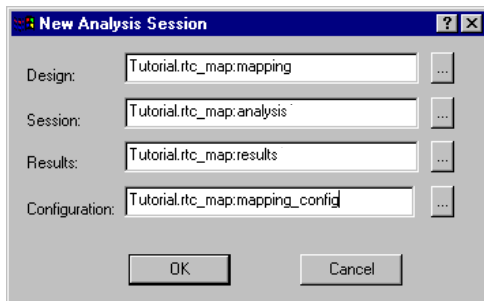
Begin an Analysis Session

1. In the Project Folder, open the *rtc_map* mapping diagram.
2. To create a new analysis session, choose *File > New* from the main menu.



3. Click the *Analysis* tab.
4. Double-click the *Analysis* icon in the New dialog.

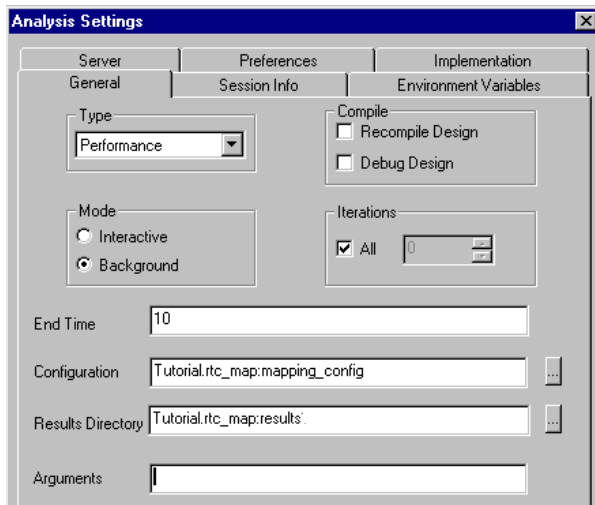
The New Analysis Session dialog opens.



5. Accept the default values and click *OK*.

Tutorial 2: VCC Hierarchical Modeling Analysis

6. To specify the analysis settings, choose *Analysis > Settings* from the main menu.



7. Set Type to *Performance*.
8. Set Mode to *Background*.
9. Set End Time to *10*.
10. Click *OK* in the Analysis Settings dialog.

Create the *samplingTime* Parameter

1. Right click in the blank area of the mapping diagram and choose *Properties* from the pop-up menu.

There are no Parameters specified, so all the fields in the Properties dialog are blank.

2. Right click in any of the blank fields, and choose *New Property* from the pop-up menu.

A green arrow and a flashing cursor in the Name field indicate that you can enter a new property name.

You will enter this new property.

Name	Type	Value
samplingTime	Real	0.01

3. Enter *samplingTime* for the Name.
4. Tab to the Type field and type *Real* or choose from the pulldown menu.

Tutorial 2: VCC Hierarchical Modeling

Analysis

5. Double-click in the Value field and enter *0.01*.

This is the value you will set to sweep, but the sweep values will be associated with the analysis session, not with the mapping diagram itself.

6. Click *OK* in the Properties dialog.

Bind the *samplingTime* Parameter

You will bind the *samplingTime* parameter you just created to the *samplingTime* parameter in the behavior diagram.

1. Right click on the behavior diagram and choose *Properties* from the pop-up menu.
2. Click the *Parameters* tab.
3. Change the Value to *samplingTime*.

This tells the simulation to use the value of the top-level *samplingTime* parameter.

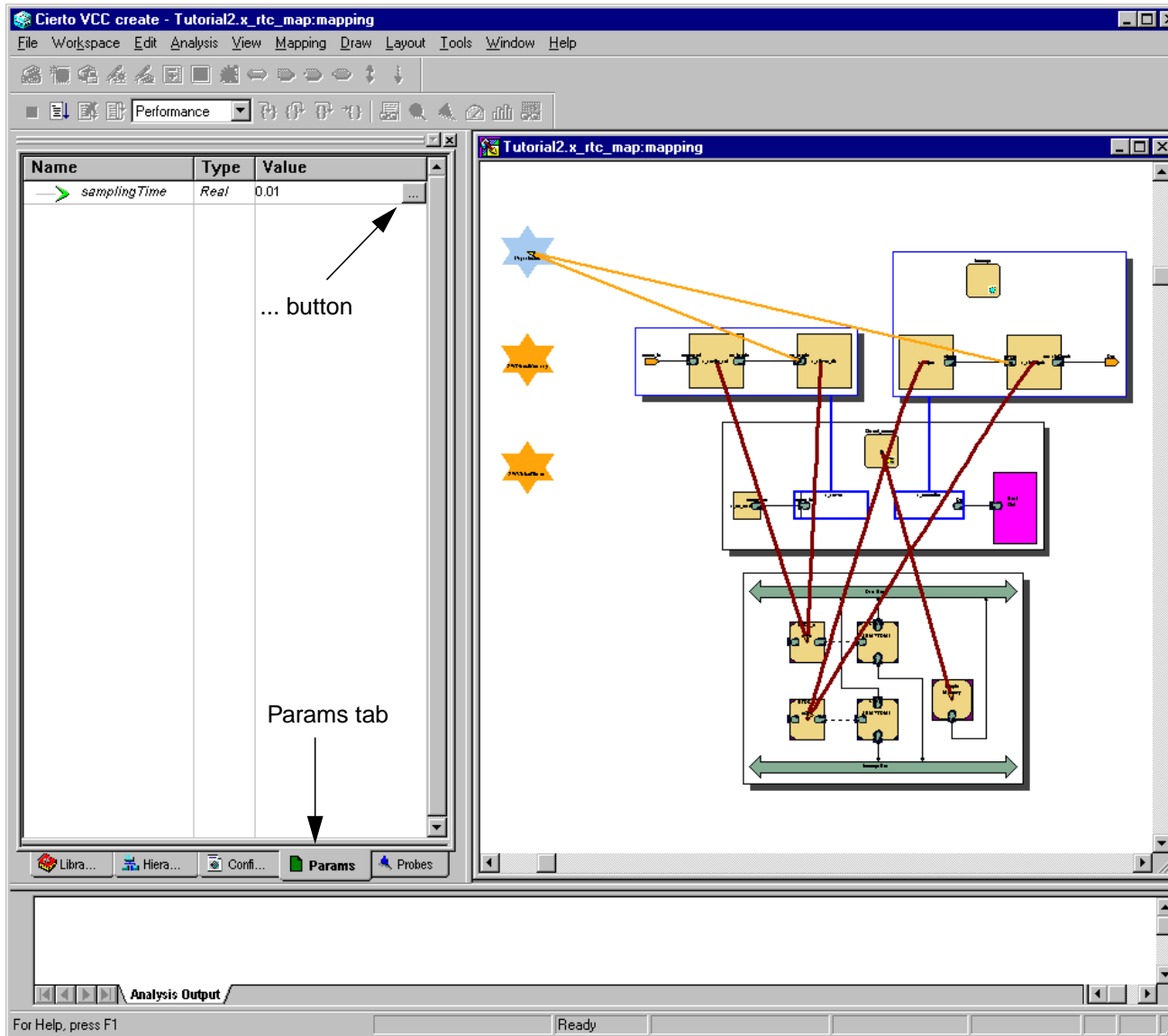
4. Click *OK* in the Properties dialog.

Set the Sweep Parameters

Instead of using a static parameter value for *samplingTime* in this simulation, you will define sweep parameters for this value.

Tutorial 2: VCC Hierarchical Modeling Analysis

1. To display the top-level parameters in the workspace, click the *Params* tab.

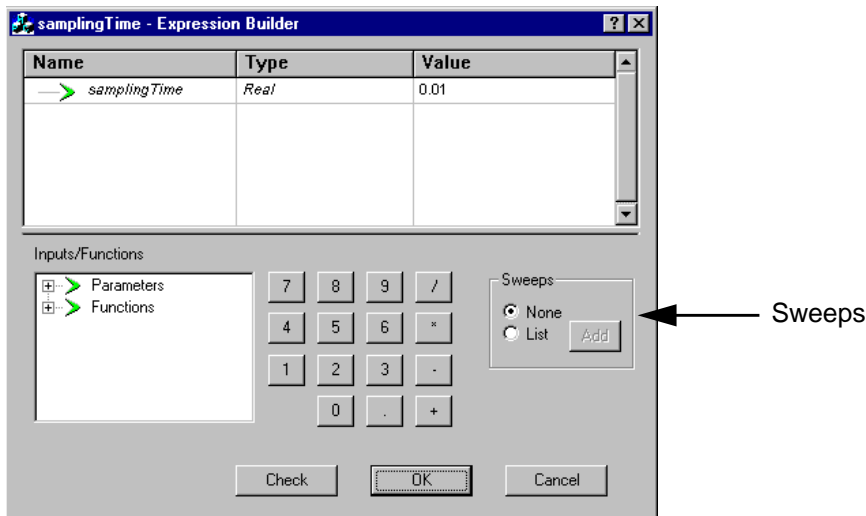


The top-level *samplingTime* parameter is the only parameter displayed.

2. To open the Expression Builder, double-click in the Value field then click the (...) button.

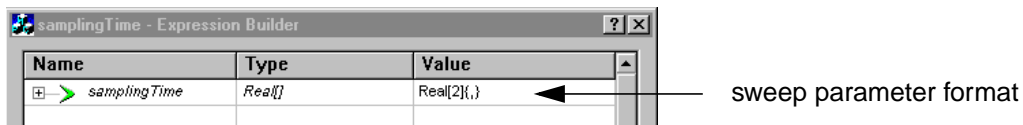
Tutorial 2: VCC Hierarchical Modeling Analysis

The Expression Builder displays the current values for the *samplingTime* parameter.



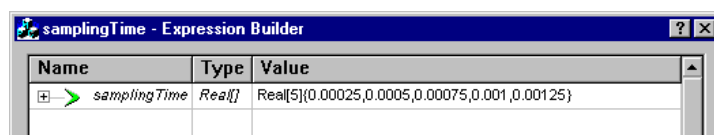
3. In the Sweeps section, choose List, then click *Add*.

The Value for *samplingTime* now displays the format for entering the sweep parameters.



4. Enter the following expression for the Value:
`Real[5]{250e-6, 500e-6, 750e-6, 1000e-6, 1250e-6}`
5. To confirm your new expression, click *Check*.
A dialog reports that your expression is OK.
6. Click *OK* in the Expression Check dialog.

Your expression now looks like this:



Tutorial 2: VCC Hierarchical Modeling

Analysis

7. Click *OK* in the Expression Builder.
8. To save these changes, choose *File > Save* from the main menu.

The sweep parameters are saved as part of the analysis session, not as part of the mapping diagram. If you look at the *samplingTime* parameter for the top level of the mapping diagram, you will see its value is still 0.01.

Run a Simulation

Note: When the session runs in *Background* mode, you don't have to explicitly run the *Analysis > Initialize* command.

1. Choose *Analysis > Go/Continue* from the main menu.
A dialog asks if you want to save the session changes.
2. Click *Yes* in the dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

VCC reports the progress in the Analysis Output window at the bottom of the screen.

The screenshot displays the VCC software interface. The main window shows a hierarchical circuit diagram with three levels of abstraction, indicated by yellow stars. The top level is a high-level block diagram. The middle level shows a more detailed circuit with components like 'RAM' and 'CPU'. The bottom level shows a very detailed circuit with many components. Red lines connect the components across the levels, showing the mapping. The left pane shows a table with columns 'Name', 'T...', and 'Value'. The bottom pane shows the 'Analysis Output' window with the following text:

Segment	Size	ArchitectureMemory	Offset
stack	1024	Architecture/RAM	0:1023
bss	112	Architecture/RAM	0:111
code	336	Architecture/RAM	0:335
..... (in /):			

Info: Run(4):
* Current Simulation Time is: 0

Segment	Size	ArchitectureMemory	Offset
stack	1024	Architecture/RAM	0:1023
bss	100	Architecture/RAM	0:99
..... (in /):			

The bottom pane also has tabs for 'Netlist', 'Analysis Output', and 'Archive'.

The output reports Run(0) through Run(4), one run for each of the sweep parameters.

Simulate using a Single-CPU Architecture

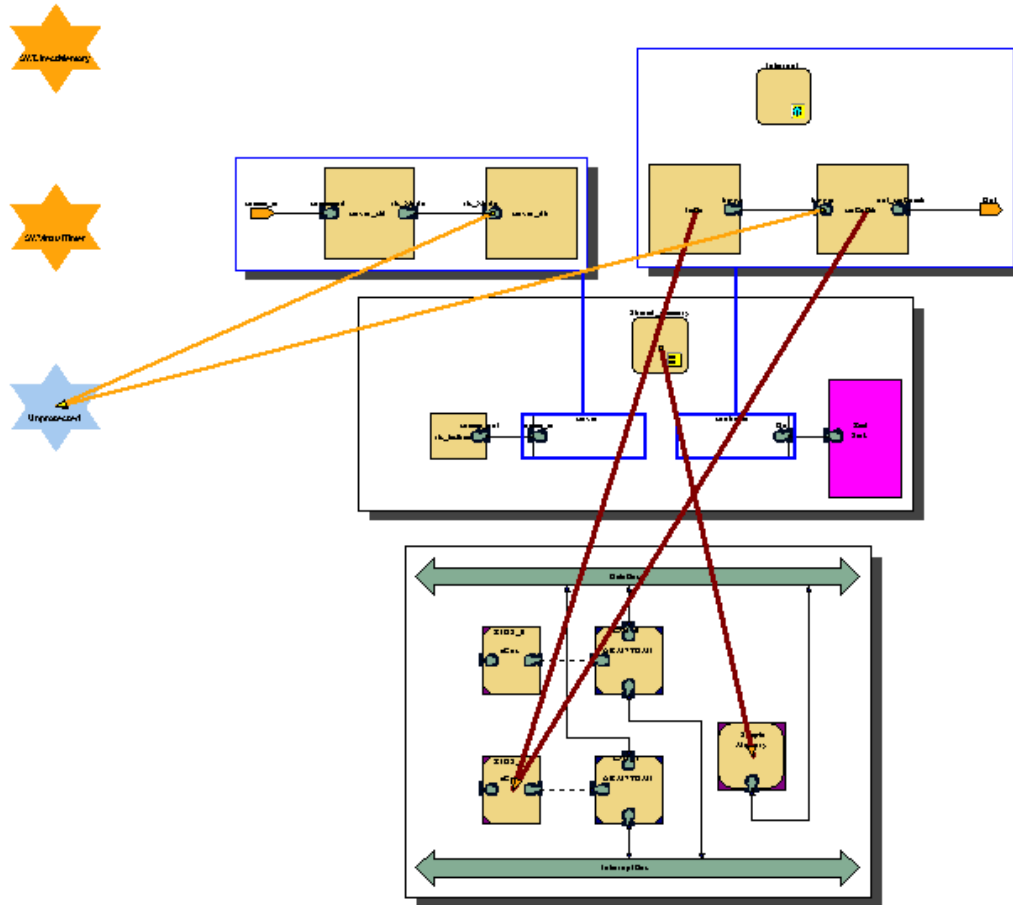
In this section, you will investigate whether the real-time controller can fit onto a single-CPU architecture.

Create a Single-CPU Mapping Diagram

1. With the *rtc_map* mapping diagram open, choose *File > Save As* from the main menu.
2. Save the diagram as *Tutorial.one_cpu:mapping*.
3. Select and delete the mapping connection between the *server_std* behavior block and the *RTOS_0* architecture block.
4. Select and delete the mapping connection between the *server_db* behavior block and the *RTOS_0* architecture block.

Tutorial 2: VCC Hierarchical Modeling Analysis

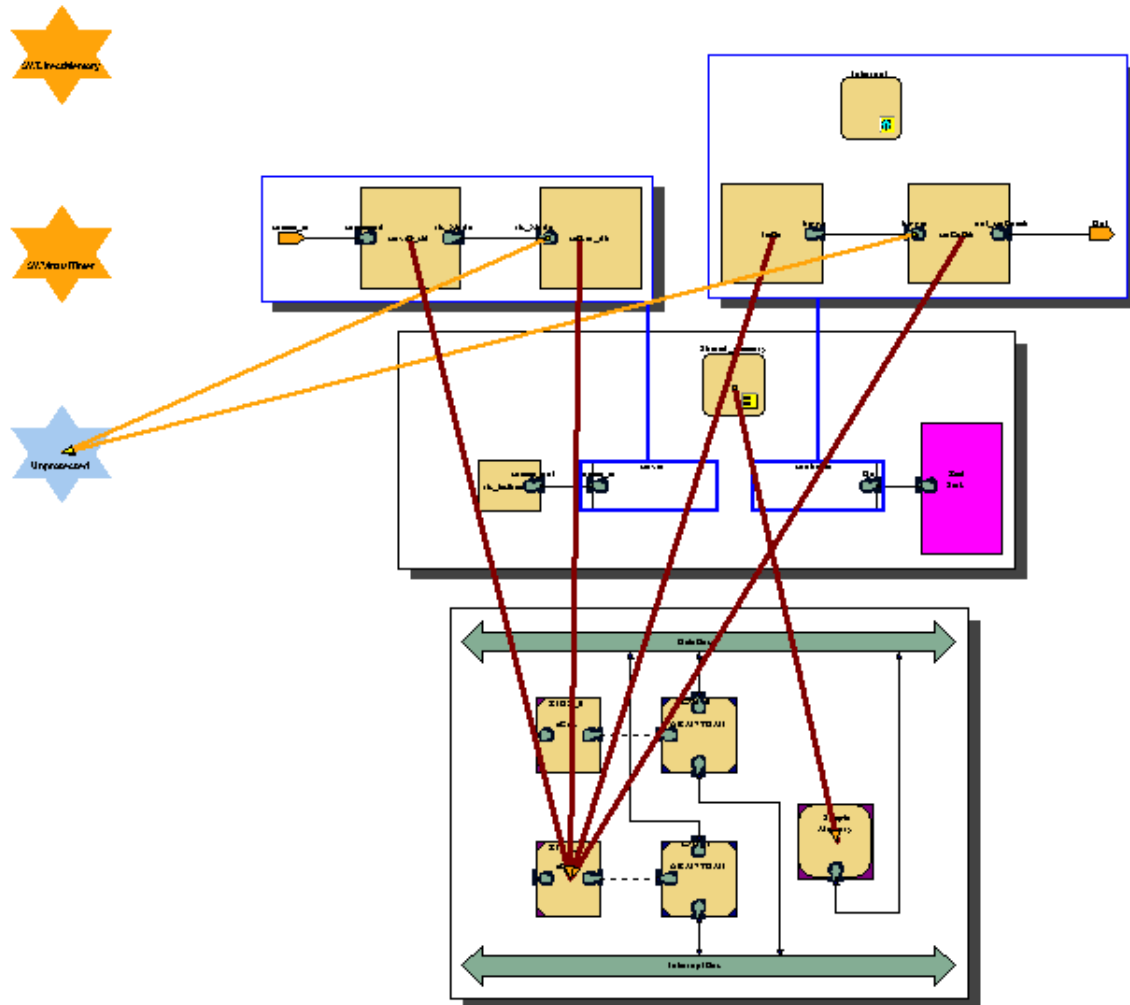
Your diagram now looks like this:



5. Choose *Mapping > Mapping Connection* from the main menu.
6. Click the *server_std* behavior block, then click the *RTOS_1* architecture block.
7. Choose *Mapping > Mapping Connection* from the main menu.
8. Click the *server_db* behavior block, then click the *RTOS_1* architecture block.

Tutorial 2: VCC Hierarchical Modeling Analysis

Your diagram now looks like this:



Tutorial 2: VCC Hierarchical Modeling Analysis

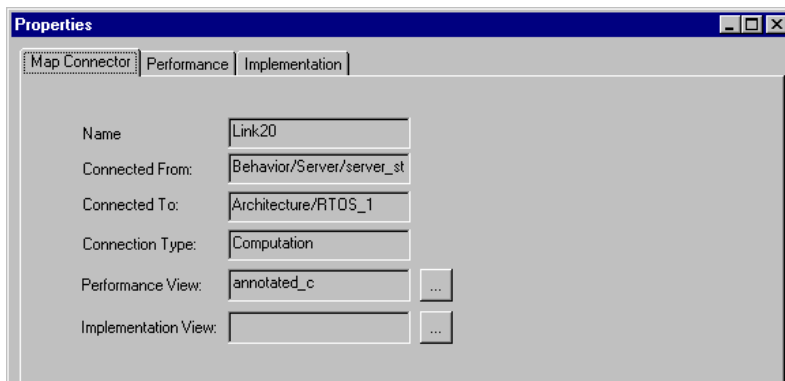
Specify Task Priorities

Now that the *server_std* and *server_db* blocks are mapped to the *RTOS_1*, each of the mapping connections to the *RTOS_1* block must have its task priority set to a new value.

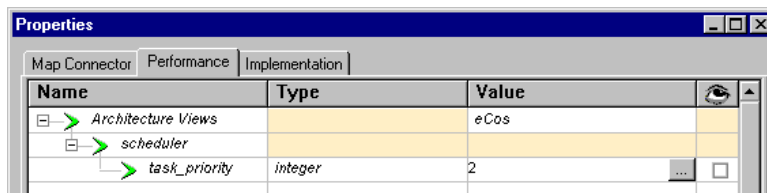
server_std

1. To specify the task priority for the mapping connection between the *server_std* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.



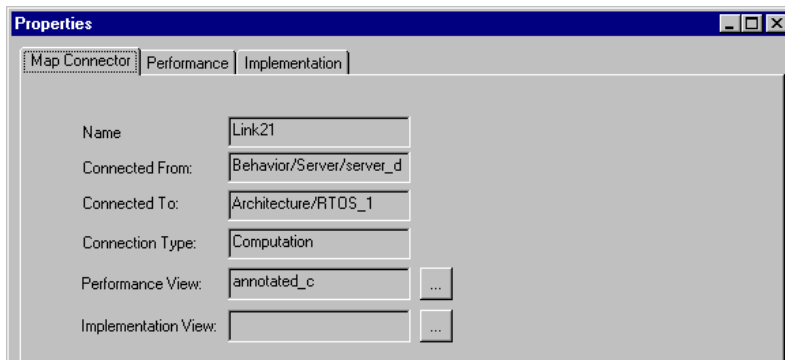
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 2.
6. Click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

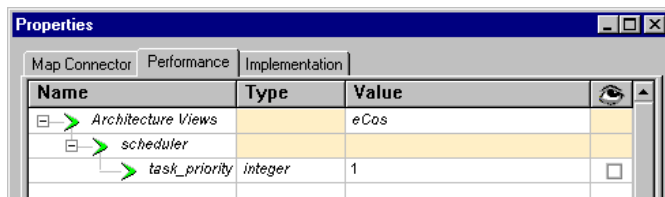
server_db

1. To specify the task priority for the mapping connection between the *server_db* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.



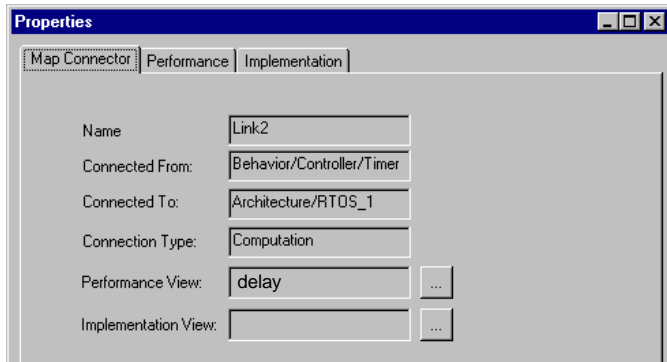
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 1.
6. Click OK in the Properties dialog.

Timer

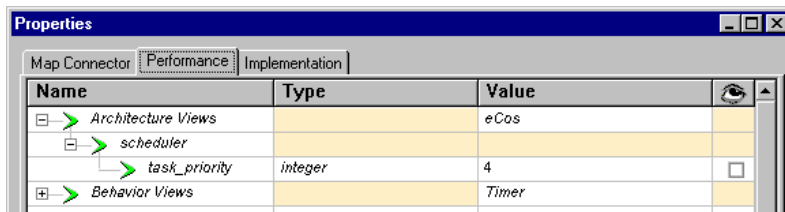
1. To specify the task priority for the mapping connection between the *Timer* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

Tutorial 2: VCC Hierarchical Modeling Analysis

The Properties dialog opens.



2. Click the *Performance* tab.



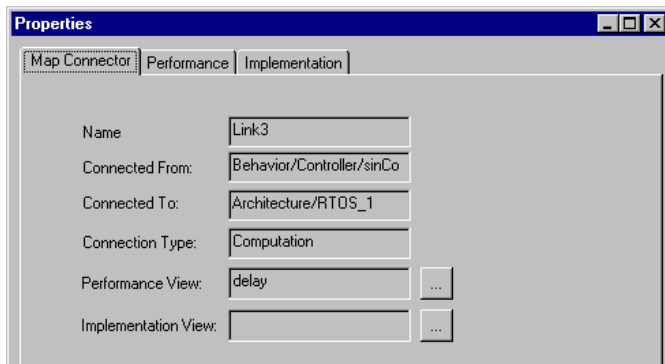
3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 4.
6. Click *OK* in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

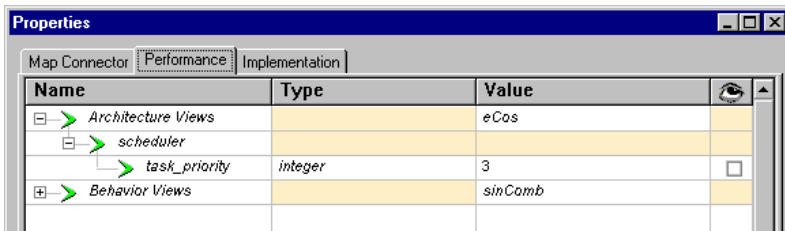
sinComb

1. To specify the task priority for the mapping connection between the *sinComb* and the *RTOS_1*, right click the connection and choose *Properties* from the pop-up menu.

The Properties dialog opens.



2. Click the *Performance* tab.



3. Click the plus sign to expand the *Architecture Views* header.
4. Click the plus sign to expand the *scheduler* header.
5. Double-click the Value field for *task_priority* and enter 3.
6. Click *OK* in the Properties dialog.

Bind the New Memory Segments to RAM

You will bind the new memory segments for the *CPU_1* block to the Simple Memory whose instance name is *RAM*.

1. Right click on *CPU_1* and choose *Properties* from the pop-up menu.

The Properties dialog opens. The memory segments for the *stack* and *bss* parameters are already set.

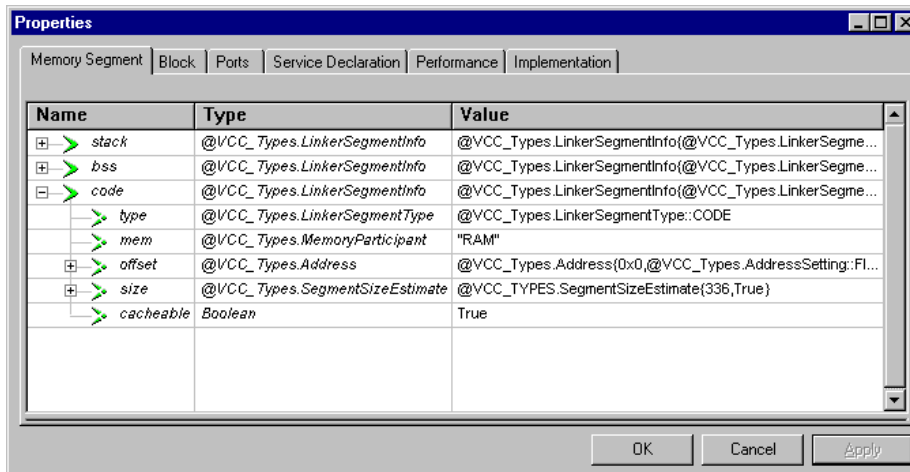
2. Click the plus sign to expand the *code* header.

Tutorial 2: VCC Hierarchical Modeling Analysis

3. For the *mem* parameter, enter the value

"RAM"

The value must include the beginning and ending quotations.



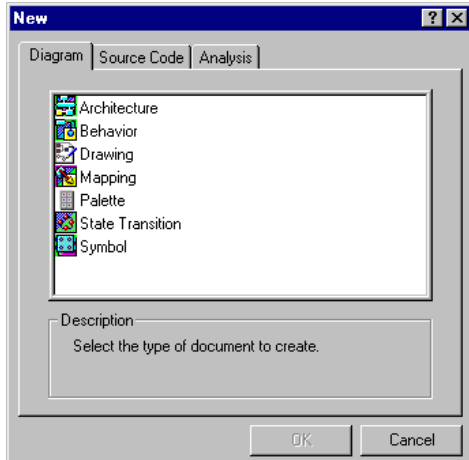
4. Click **OK** in the Properties dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

Begin an Analysis Session

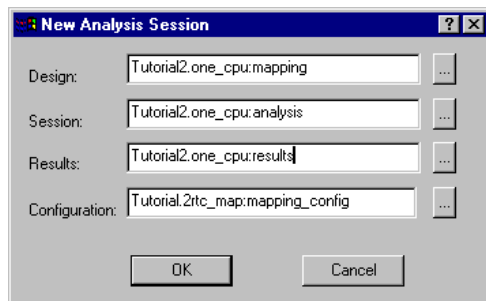
Note: If you are using the same analysis session you created in either of the other exercises, skip to “[Run a Simulation](#)” on page 211.

1. To create a new analysis session, choose *File > New* from the main menu.



2. Click the *Analysis* tab.
3. Double-click the *Analysis* icon in the New dialog.

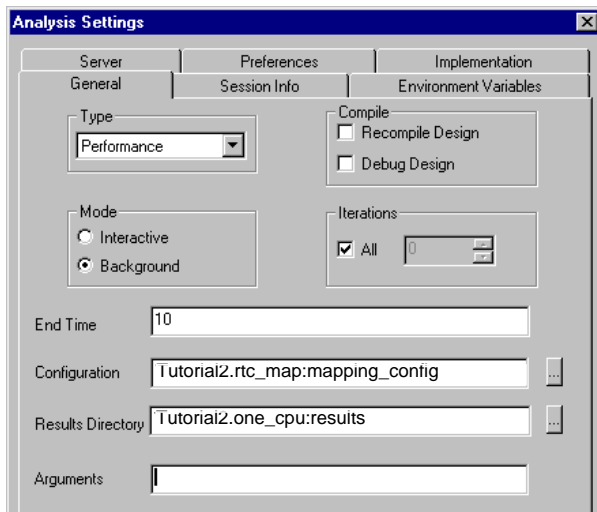
The New Analysis Session dialog opens.



4. Accept the default values and click *OK*.

Tutorial 2: VCC Hierarchical Modeling Analysis

5. To specify the analysis settings, choose *Analysis > Settings* from the main menu.



6. Set Type to *Performance*.
7. Set Mode to *Background*.
8. Set End Time to *10*.
9. Click *OK* in the Analysis Settings dialog.

Run a Simulation

Note: When the session runs in *Background* mode, you don't have to explicitly run the *Analysis > Initialize* command.

1. Choose *Analysis > Go/Continue* from the main menu.
A dialog asks if you want to save the session changes.
2. Click *Yes* in the dialog.

Tutorial 2: VCC Hierarchical Modeling Analysis

VCC reports the progress in the Analysis Output window at the bottom of the screen.

The screenshot displays the VCC (Verilog Compiler Compiler) software interface. The main window is titled "Tutorial.one_cpu.mapping". On the left, a project tree lists various components and modules, including "add_mult_spwomi", "add_mult_testbench", "belt_controller_behav", "cycle_behv", "cycle_test", "cycle_type", "CycleEcos", "test", "Tutorial", "CommandType", "controller", "one_cpu", "rtc_behav", "rtc_map", "rtc_testbench", "server", "server_db", "server_std", "sinComb", "Timer", "two_cpu", "Tutorial2_Examples", "types", "VCC_ArchitectureServices", "VCC_ASIC", "VCC_Bus", "VCC_CPU", "VCC_Defaults", "VCC_Exporters", "VCC_FixedPatternServices", "VCC_Internals", "VCC_InternalServices", "VCC_Memory", "VCC_Patterns", and "VCC_PatternServices". The central area shows a hierarchical circuit diagram with multiple blocks and interconnections. A blue star icon labeled "Unplaced" is visible on the left side of the diagram. The bottom panel displays the "Analysis Output" window, showing the following information:

```
Info: Run[0]:
* Current Simulation Time is: 0
Segment    Size    ArchitectureMemory    Offset
-----
stack      1024    Architecture/RAM    0:1023
code       336    Architecture/RAM    0:335
bss        112    Architecture/RAM    0:111
{in /}:
```

The status bar at the bottom indicates "For Help, press F1" and "Done 0.000000000000".