

Design Exploration Tutorial for HDL Detective

Software Version 2001.3

7 June 2001



**Copyright © Mentor Graphics Corporation 2001.
All rights reserved.**

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND 03/97

U.S. Government Restricted Rights. The SOFTWARE and documentation have been developed entirely at private expense and are commercial computer software provided with restricted rights. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement provided with the software pursuant to DFARS 227.7202-3(a) or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

Web site: <http://www.hldesigner.com>
Email: hldesigner_support@mentor.com

This is an unpublished work of Mentor Graphics Corporation.

Trademark Information

The following names which appear in this documentation set are trademarks, registered trademarks or service marks of Mentor Graphics Corporation:

HDL Designer Series™, HDL Designer™, HDL Pilot™, HDL Detective™, HDL Author™, HDL2Graphics™, FPGA Advantage™, Interconnect Table™, Interface-Based Design™, IBD™, Inventra™, LeonardoInsight™, LeonardoSpectrum™, Mentor™, Mentor Graphics®, ModelSim®, ModuleWare™, Renoir™, Seamless® and Seamless CVE™.

The following names which appear in this documentation set are trademarks, registered trademarks or service marks of other companies:

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Exchange, FrameMaker and PostScript are registered trademarks of Adobe Systems Incorporated.

Altera, MegaWizard and MAX+PLUS are registered trademarks and Quartus a trademark of Altera Corporation.

ClearCase Attache is a trademark and ClearCase is a registered trademark of Rational Software Corporation.

DesignSync is a registered trademark of Synchronicity Incorporated.

FLEXIm is a trademark of Globetrotter Software, Incorporated.

Hewlett-Packard (HP), HP-UX and PA-RISC are registered trademarks of Hewlett-Packard Company.

Leapfrog, NC-Verilog, Verilog and Verilog-XL are trademarks and registered trademarks of Cadence Design Systems Incorporated.

Netscape is a trademark of Netscape Communications Corporation.

Quartus and APEX are trademarks of Altera Corporation.

SPARC is a registered trademark and SPARCstation is a trademark of SPARC International Incorporated.

SpyGlass is a trademark of Interra Inc.

Sun Microsystems and Sun Workstation are registered trademarks of Sun Microsystems Incorporated. Sun and SunOS are trademarks of Sun Microsystems Incorporated.

Synopsys, Design Analyzer, Design Compiler, FPGA Express, VCS, VCSi and VSS are trademarks of Synopsys Incorporated.

Synplify is a registered trademark of Synplicity Incorporated.

The Graphics Connection is a trademark of Square One.

Visual SourceSafe and Windows are trademarks of Microsoft Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Incorporated.

Xilinx is a registered trademark and Core Generator a trademark of Xilinx, Incorporated.

Other brand or product names that appear in the documentation are trademarks or registered trademarks of their respective holders.

TABLE OF CONTENTS

| | |
|--|------|
| About This Manual | vii |
| Introduction..... | vii |
| Copying Text From the Acrobat Viewer | viii |
| Typographic Conventions..... | viii |
| Design Exploration Tutorial | 1-1 |
| Welcome to HDL Detective | 1-1 |
| The Design Browser | 1-1 |
| Import the Fibonacci Design | 1-3 |
| Browse the Fibonacci Design | 1-7 |
| Convert HDL text to Graphics..... | 1-8 |
| Add a Panel to the Block Diagram | 1-10 |
| Display the IBD View | 1-11 |
| Set the HDL Import Options..... | 1-13 |
| Recover a State Diagram View..... | 1-14 |
| Relevel the State Diagram | 1-15 |
| Recover a Flow Chart View | 1-16 |
| Appendix A | |
| Fibonacci Sequencer Design | A-1 |
| Source Code..... | A-1 |
| Importing the Design | A-2 |
| The Recovered Design..... | A-2 |
| The Accumulator Design Unit | A-4 |
| The Fibgen Block Diagram | A-8 |
| The Control State Machine | A-9 |
| The Fibgen_tb Test Bench | A-10 |
| The Fibgen Tester Design Unit..... | A-11 |
| Simulation Results | A-14 |
| Graphical Designs..... | A-15 |

TABLE OF CONTENTS [continued]

About This Manual

Introduction


This manual provides a self-paced tutorial with step-by step procedures for importing a simple [HDL text](#) design and using the [HDL Detective](#) tool to explore and visualize the design.

A number of other tutorials for users of other [HDL Designer Series](#) tools can be accessed from links in the [HDL Designer Series Bookcase](#) which is available from the **Help** menu in all application windows.



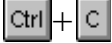
HDL Pilot users should perform the [Design Management Tutorial](#). Users of HDL Author or HDL Designer can learn about the additional facilities available in these tools by performing the [Interface-Based Design Tutorial](#) or [Graphical Design Tutorial](#).


All user commands in the tutorial procedures are referenced using their menu path (shown in bold text) or toolbar button. However, many commands can also be accessed using keyboard shortcuts. Refer to the shortcut tables in the help pages for full lists of the available shortcuts. These lists can be accessed by choosing **Shortcuts** from the HDL Designer **Help** menu.

When new terminology is introduced, the keywords (shown in blue when this document is viewed online) are hyperlinked to a definition in the Glossary. (For example, the keywords [HDL text](#) and [HDL](#) in the first sentence above.)

User and reference information including an online glossary of terminology can be accessed at any time from the **Help Topics** index. The **Help Topics** provide a contents list, keyword index and full text search facility. In addition many help topics can be accessed directly by  buttons from many of the application dialog boxes.

Copying Text From the Acrobat Viewer

You can copy text from this document by choosing the  (**Text Select**) tool button or  shortcut key in the Acrobat viewer and choosing **Copy** from the Acrobat **Edit** menu (or using the  shortcut).

The text can be pasted into a text editor (or application dialog box) using the  shortcut or the **Paste** menu option if one is provided in the destination window. In the graphic editors, you can use the **Paste Special** option to explicitly paste text from the system clipboard.



If you copy HDL text from a tutorial help page, check that punctuation characters are copied correctly. In particular, line feed characters may not be translated on UNIX systems and may need to be re-entered.

Typographic Conventions

The following conventions have been used in this manual:

| Style | Usage |
|-----------------------|--|
| Bold | Menu options and command line switches. |
| <i>Italic</i> | Pathnames (and object names derived from file names). Also used for manual titles (for example, <i>ModuleWare Reference Guide</i>). |
| Courier font | Monospaced Courier font is used for code examples. |
| Links | Cross references within the text are shown in blue. |



When pathnames (or window titles derived from pathnames) are shown in this tutorial, the PC convention (\) is used.

Design Exploration Tutorial

Welcome to HDL Detective

This tutorial introduces [HDL Detective](#), the [HDL Designer Series](#) tool for design exploration and visualization.




This tutorial shows how **HDL Detective** can be used to import a design described by VHDL or Verilog [HDL text](#) and visualize the design structure as a hierarchy of graphical [design unit views](#).

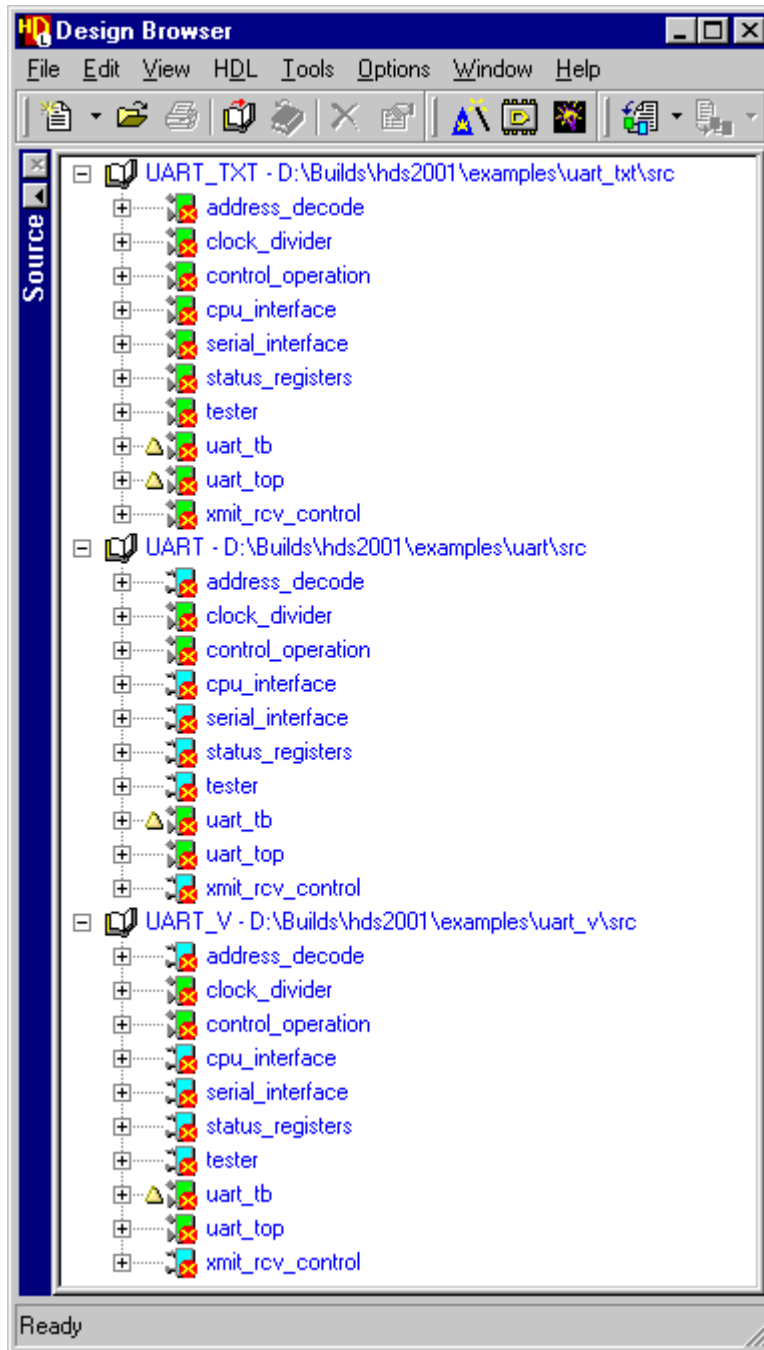
Alternative tutorials for users of the [HDL Pilot](#), [HDL Author](#) or [HDL Designer](#) tools can be accessed from the [HDL Designer Series Tutorials Bookcase](#).


The Design Browser

The [design browser](#) opened when [HDL Detective](#) is invoked displays a [source browser](#) containing the libraries that were open the last time you invoked a HDL Designer Series tool.




When you invoke for the first time, three example libraries are displayed: a mixed language HDL text design named *UART_TXT*, a graphical VHDL design named *UART* and the corresponding graphical Verilog design named *UART_V*. These example designs are described in the [Design Management Tutorial](#).

If the HDL Designer Series has been installed on a readonly file system (typical for UNIX installations) all objects are shown in blue with an  overlay on the icon.

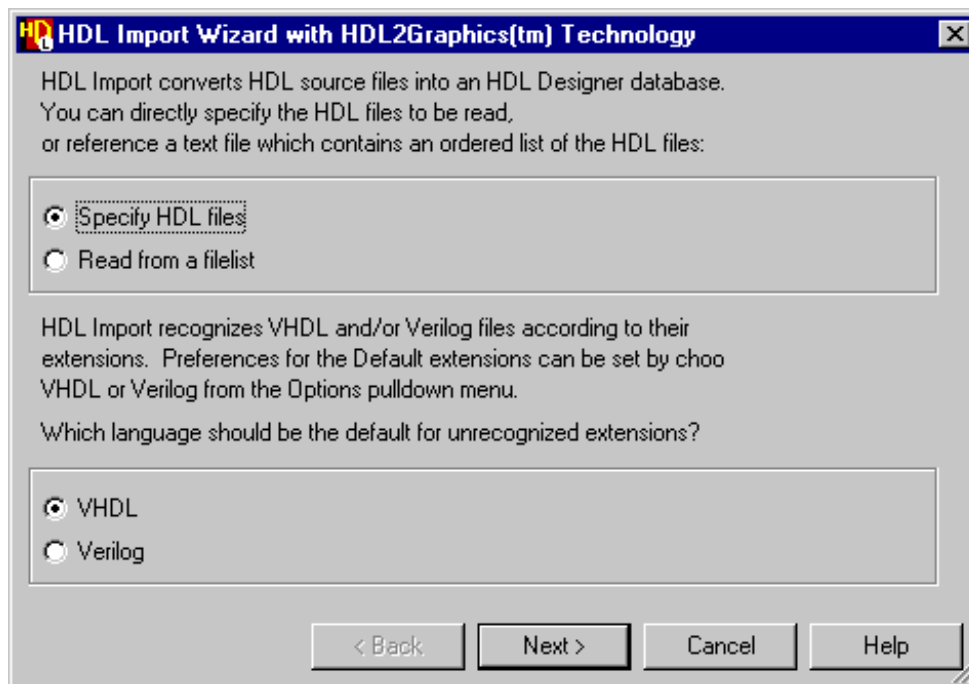


The UART examples are not used in this tutorial and can be closed by selecting each [library](#) and using the  button.

Import the Fibonacci Design

Use the pulldown  on the  button in the design browser and select the  option from the dropdown palette (or choose **Text HDL Import** from the **HDL Import** cascade of the **HDL** menu) to display the HDL Import wizard.

Select **Specify HDL files** in the first page of the wizard:



You can optionally choose to **Read from a filelist** that contains a list of source file pathnames. This can be useful if a suitable list already exists or when you have saved a file list to repeatedly import new versions of the same source files.

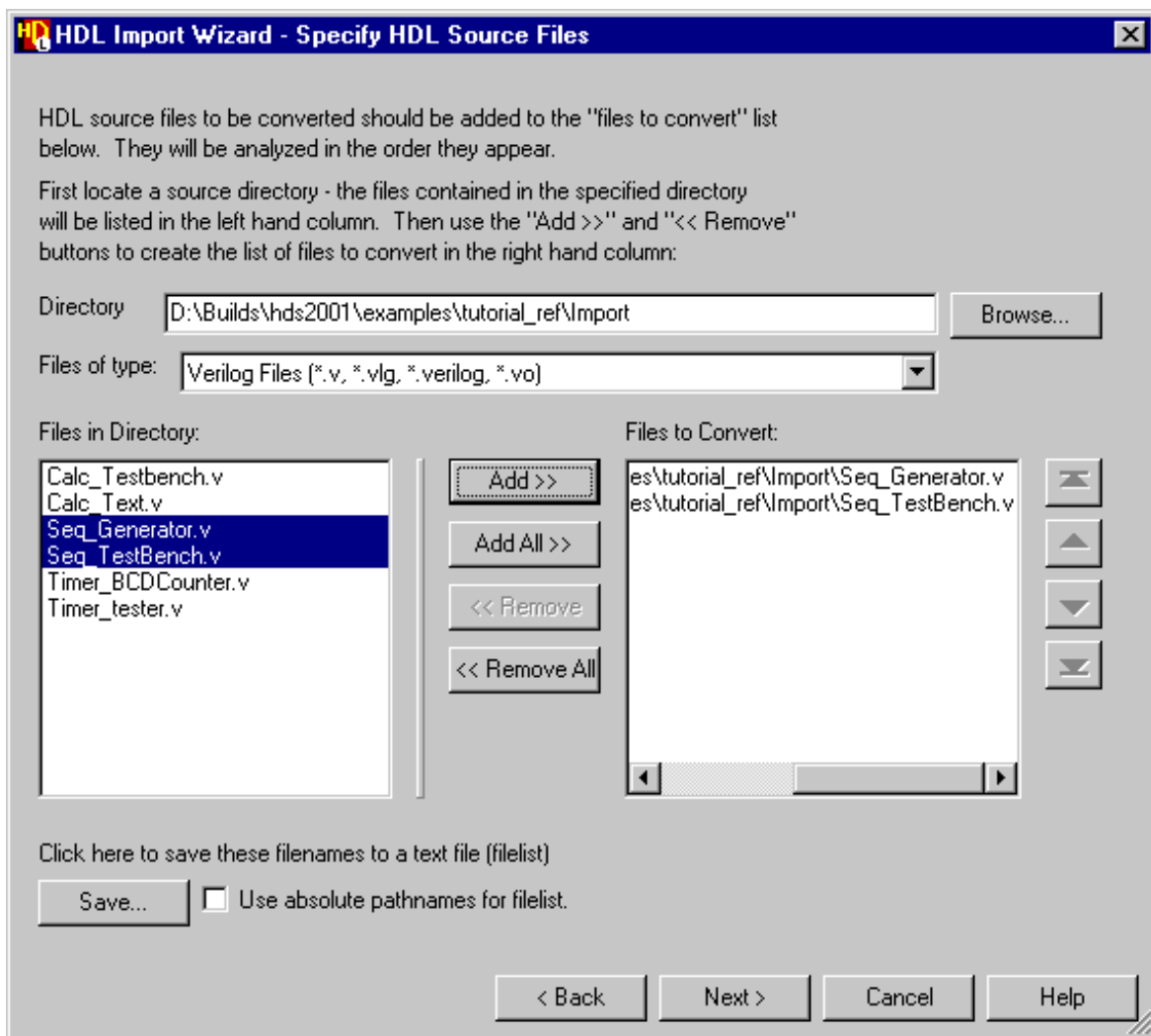
The HDL Import Wizard recognizes the hardware description language used by files from the file extensions (such as *.vhd* or *.v*) saved in your preferences. The language option is only required in the wizard if your HDL is in an unrecognized file type.

Click  to display the Specify HDL Source Files page of the wizard.

Use the **Browse...** button to locate the Fibonacci sequencer design source code in the examples sub-directory of your HDL Designer Series installation. For example, if HDS is installed in the directory *D:\Builds\hds2001*, the path is:

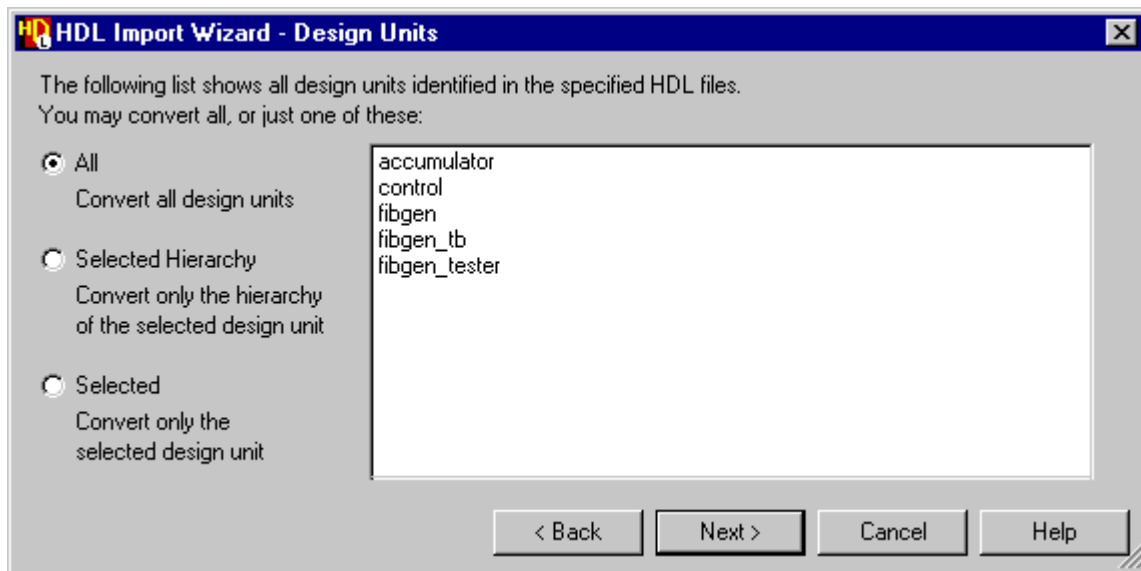
D:\Builds\hds2001\examples\tutorial_ref\Import

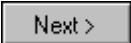
Choose Verilog Files or VHDL files from the pulldown filter for **Files of type** and select the *Seq_Generator.v* and *Seq_TestBench.v* files (if you are using Verilog) or the *Seq_Generator.vhd* and *Seq_TestBench.vhd* files (if you are using VHDL) and use the **Add >>** button to move these files into the list of Files to Convert:

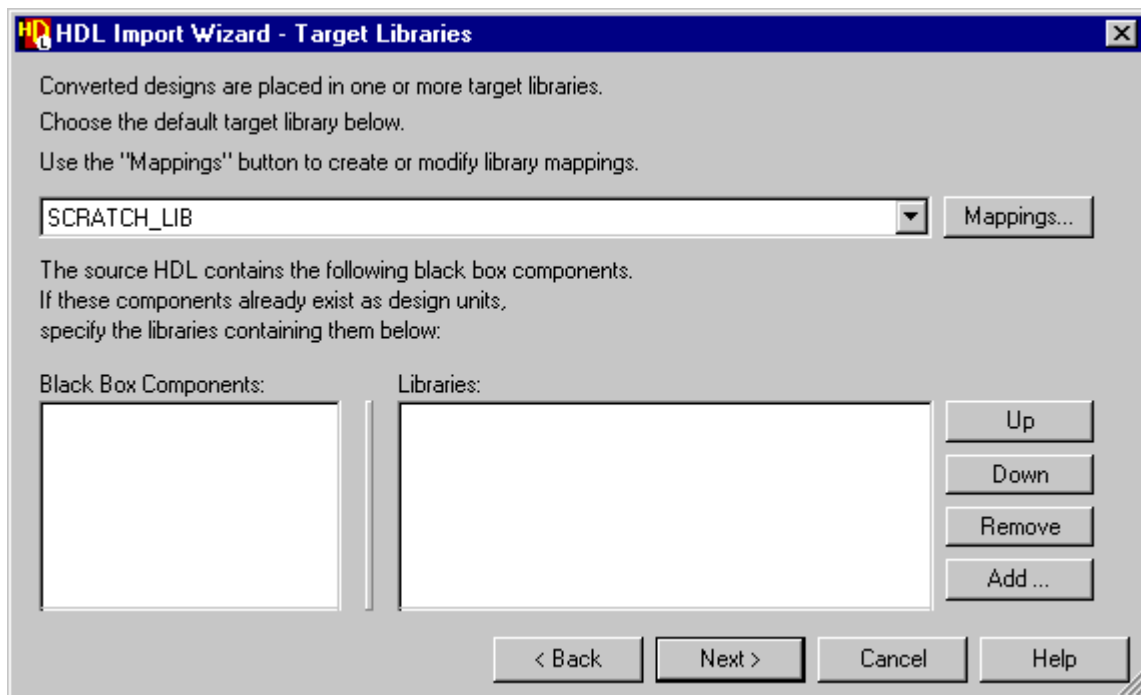


Click the **Next >** button to display the Design Units page of the wizard.

The Design Units page allows you to choose from a list of [design units](#) that have been recognized in the source HDL code.



Check that the **All** option (for Convert all design units) is selected. Click the  button to display the Target Libraries page.

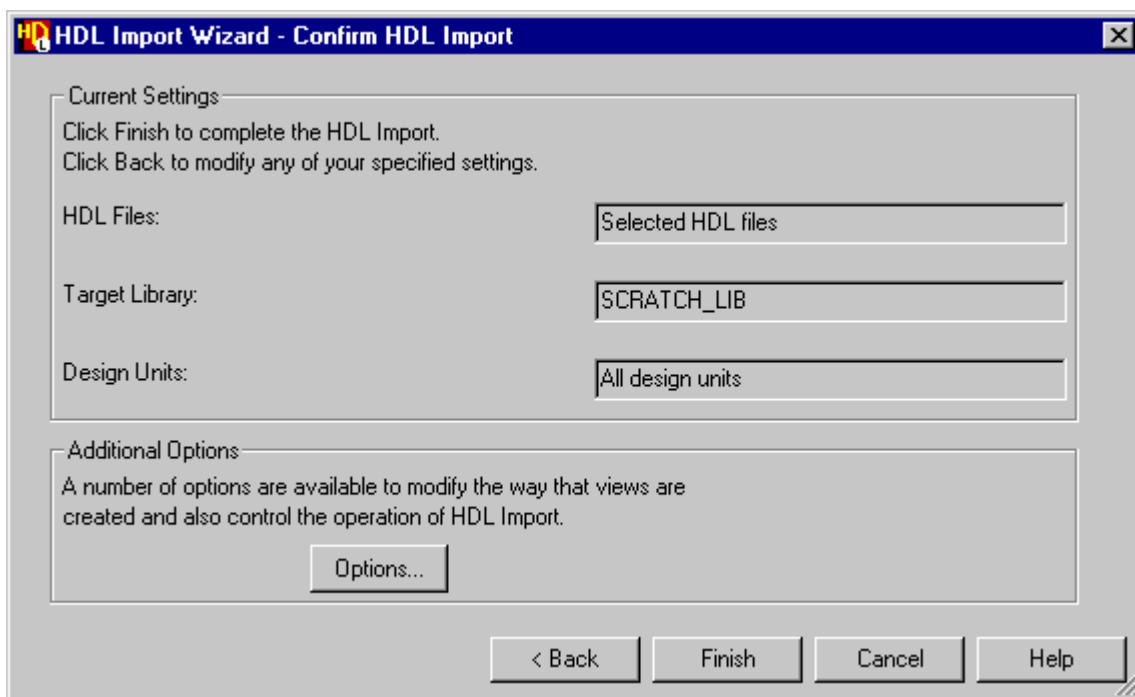


The Target Libraries page allows you to specify the library used for the imported design. For this tutorial, choose the *SCRATCH_LIB* library which is automatically available in your default library mapping. Alternatively, you can use the button to setup an alternative library mapping.



The dialog box also allows you to add target libraries if there are any instantiations of black box components with no corresponding design unit in the source HDL. These should not be required for this design since you chose to convert all design units.

Click the button to display the Confirm HDL Import page. Click the button on this page to complete the HDL import.






The HDL Log Window shows the progress of the import operation and indicates that five HDL text [design unit views](#) have been created in the *SCRATCH_LIB* library.

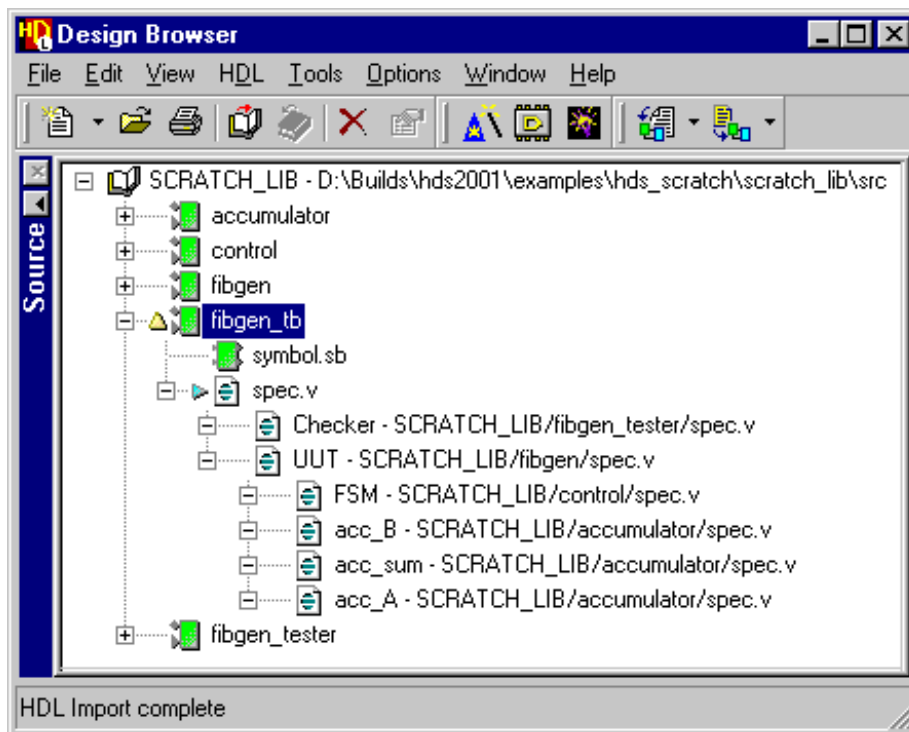
The following summary report is displayed on completion:

```
5 HDS design units saved
5 components
  5 HDL views
```



Browse the Fibonacci Design

Use the  icon to expand the *SCRATCH_LIB* library in the source browser. Notice that a  icon is displayed adjacent to the *fibgen_tb* design unit. This **test bench** component is marked as the top level design unit and each **Verilog module** or **VHDL entity** in the source HDL code is partitioned into a separate design unit and instantiated as a HDL text view in the hierarchy.


Select the *fibgen_tb* design unit and view its hierarchy by choosing **Expand All** from the design browser **View** menu or by using the  icons to expand each design unit.




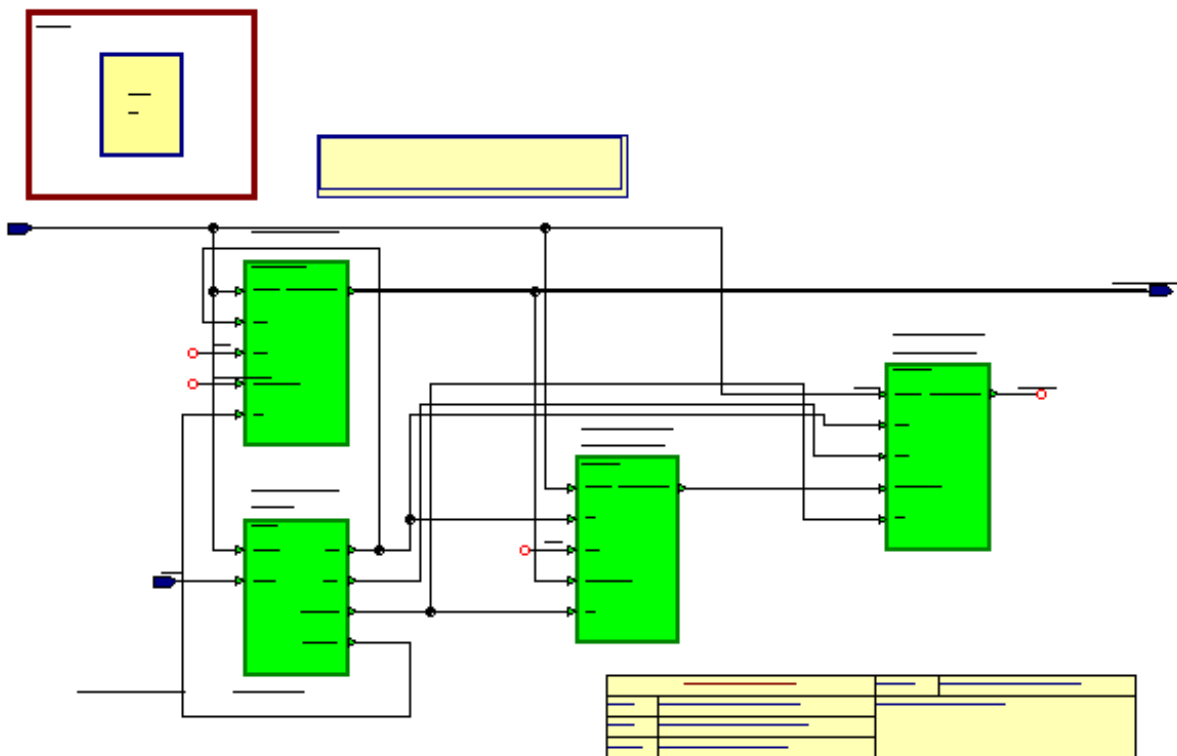
The hierarchy shows that the test bench (*fibgen_tb*) contains the *fibgen* design unit instantiated as the unit under test (*UUT*) and the *fibgen_tester* instantiated as the *Checker*. The *UUT* hierarchy is described by the control design unit (instantiated as *FSM* and three instantiations (*acc_A*, *acc_B* and *acc_sum*) of the accumulator.

You can open the text editor on any of the HDL text views (shown using the  icon for Verilog or  icon for VHDL) describing these design units by double-clicking on its entry in the source browser.

Convert HDL text to Graphics

Select the *UUT* instance and use the **Right** mouse button to choose **Single Level** from the **Convert to Graphics** cascade in the popup menu (or use the  button).

Notice that the icon used for the *UUT* instance changes to  indicating that it has been converted to a **block diagram** view. Double-click to open the block diagram which should look similar to the following picture:



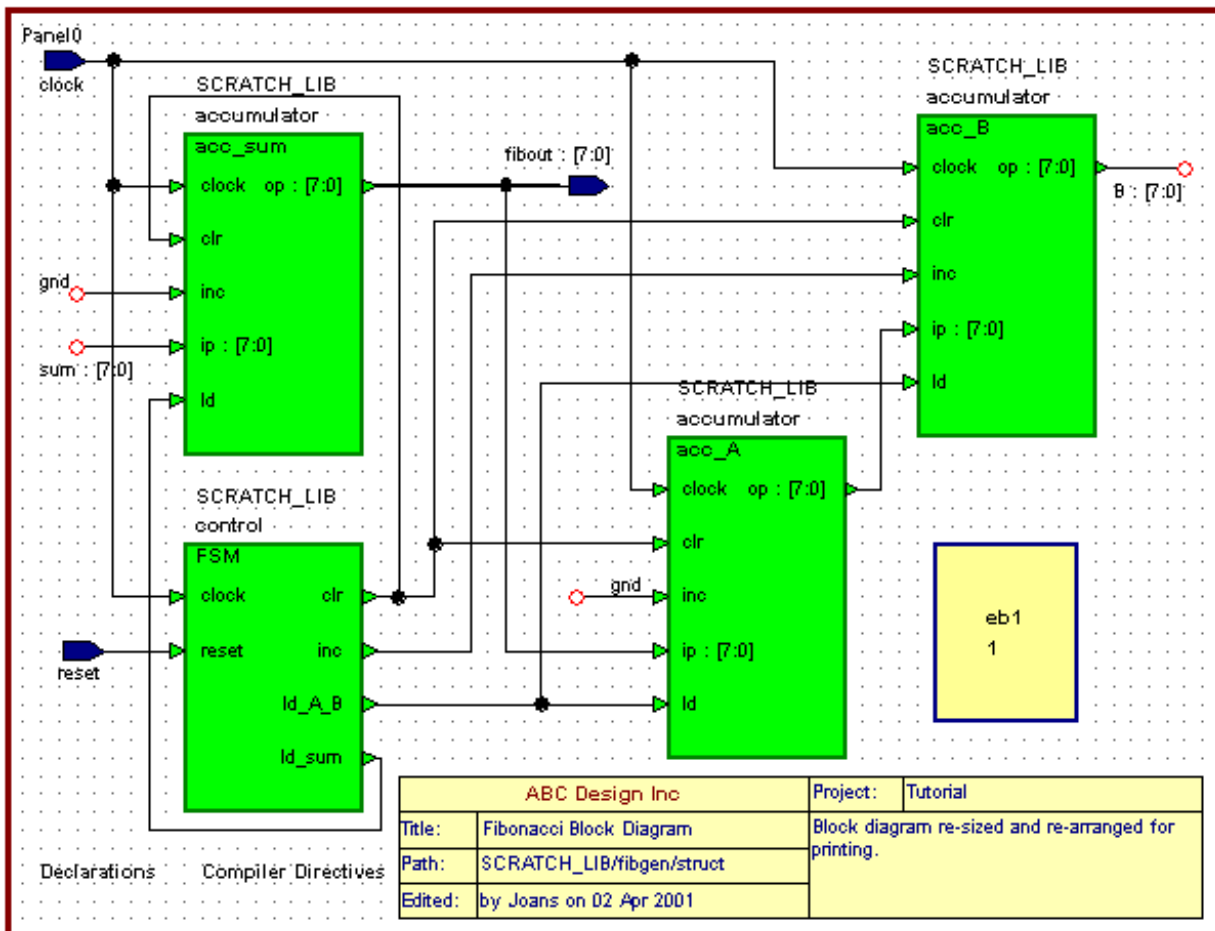
The block diagram shows the four component (*acc_A*, *acc_B*, *acc_sum* and *FSM*) with the **signal** connections between them and an unconnected **embedded block** (eb1) containing concurrent assignment statements. You can open down into the HDL describing these objects by double-clicking over them on the diagram or by choosing **Open** from the popup menu.

The block diagram is completed by a default title block and **comment text** giving information about when the diagram was created.

You can zoom in or view an area to make text objects readable or you can view a [graphic tip](#) giving information about the object under the cursor.

HDL Detective does not allow you to make logical edits which would change the HDL description for a [graphic editor](#) view. However, you can make non-logical edits to prepare a diagram for printing or export to a documentation tool.

For example, you can move or resize objects, add comment text or [comment graphics](#) and edit the title block as shown in the following picture:




You can print a diagram directly or use the **Document Export** option from the **File** menu to export the complete diagram, current window contents or a named panel as a PostScript file which can optionally be converted to several standard graphics formats.

Add a Panel to the Block Diagram

Select the Declarations text string on the block diagram. This is created near the bottom left corner of the diagram by default.

 You can use the  button or choose **View Area** from the **View** menu to zoom in to the corner of the diagram.

Drag the Declarations text string to an empty part of the diagram and choose **Show Text** from the popup menu to display the signal declarations list for the diagram. (This is hidden by default when a block diagram is created by converting a HDL text view.)

Use the  button (or choose **Panel** from the **Add** menu) and drag a new panel (Panel1) around the signal declarations.

```

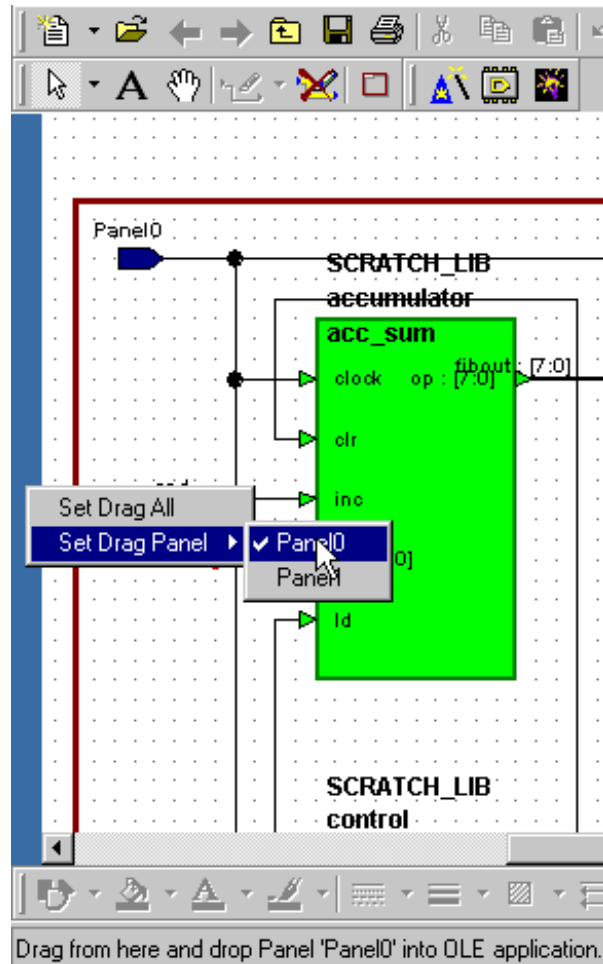
Panel1
. . .
Declarations
. . .
Ports:
. . .
wire      clock;
. . .
wire      reset;
. . .
wire [7:0] fibout;
. . .
Pre User:
. . .
Diagram Signals:
. . .
// // hds interface_end
//
. . .
// // Internal signal declarations
wire [7:0] A;
. . .
wire [7:0] B;
. . .
wire      clr;
. . .
wire      gnd;
. . .
wire      inc;
. . .
wire      ld_A_B;
. . .
wire      ld_sum;
. . .
wire [7:0] sum;
. . .
Post User:
. . .

```

The signal declarations are now displayed in a separate panel form the graphical diagram layout and can be printed or exported separately by specifying the panel name. For example, the Print dialog box contains an option to select a named panel for printing.

On a Windows workstation, you can use Object Linking and Embedding (OLE) to drag a complete or partial graphic editor view directly unto a documentation tool such as Microsoft Word or Adobe FrameMaker.

This can be done by pressing the **Right** mouse button over the blue border on the left of a diagram to display a popup menu and choosing **Set Drag All** or **Set Drag Panel <panel name>**.



The diagram can then be dragged directly into the documentation tool using the **Left** mouse button. A diagram included in this way can be opened from within the documentation tool by simply double-clicking on the picture to invoke the HDL Designer Series tool.

Display the IBD View

Choose **Show IBD** from the **Diagram** menu in the block diagram to open an alternative editor view which displays the interfaces and connections as a tabular **Interface-Based Design** view.

Notice that each of the four components and the embedded block are shown as separate columns in the IBD view matrix. A separate column (E in the example below) shows the external interface for the design unit.

The screenshot shows a window titled 'SCRATCH_LIB\fibgen\struct (IBD)'. The window contains a menu bar (File, Edit, View, Table, Tools, Simulation, Add, Options, Window, Help) and a toolbar with various icons. Below the toolbar is a table with 10 columns labeled A through J and 15 rows. The table is divided into sections: a header section (rows 1-3), a signal declaration section (rows 4-15), and a status bar at the bottom.

| | A | B | C | D | E | F | G | H | I | J |
|----|-------|---------------|------------|------|--------|--------|---------|-------------|-------------|-------------|
| 1 | Order | Name | Constraint | Type | fibgen | eb1 | control | accumulator | accumulator | accumulator |
| 2 | | Instance Ref: | | | | 1 | FSM | acc_A | acc_B | acc_sum |
| 3 | | Port Map: | | | | | | | | |
| 4 | 1 | A | [7:0] | wire | | | | op | ip | |
| 5 | 2 | B | [7:0] | wire | | | | | op | |
| 6 | 3 | clr | | wire | | clr | clr | clr | clr | |
| 7 | 4 | gnd | | wire | | | inc | | | inc |
| 8 | 5 | inc | | wire | | inc | | | inc | |
| 9 | 6 | ld_A_B | | wire | | ld_A_B | ld | ld | | |
| 10 | 7 | ld_sum | | wire | | ld_sum | | | | ld |
| 11 | 8 | sum | [7:0] | wire | | | | | | ip |
| 12 | 9 | clock | | wire | I | clock | clock | clock | clock | |
| 13 | 10 | reset | | wire | I | reset | | | | |
| 14 | 11 | fibout | [7:0] | wire | O | | | ip | | op |
| 15 | | | | | | | | | | |

At the bottom of the window, there is a status bar with the text 'All' and 'Ready'.

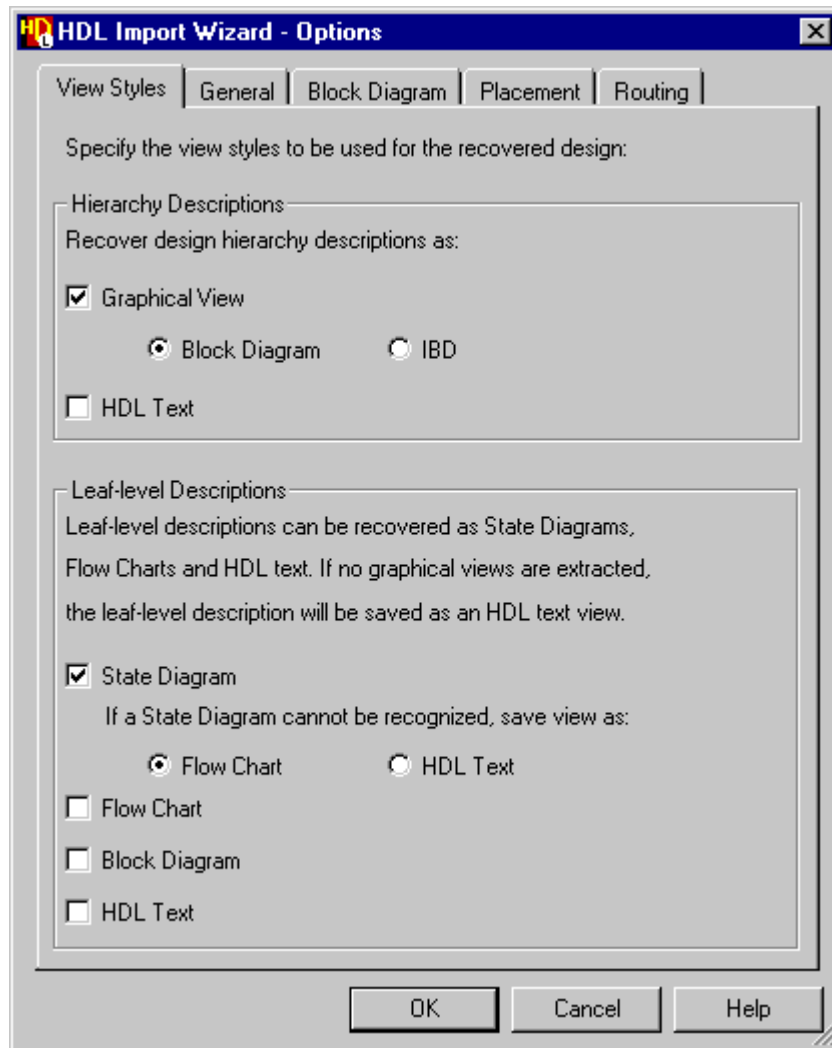
The rows in the matrix contain the signal declarations and interconnections are shown by port names in the component columns for each connected signal or by the letter I (Input) or O (Output) for the external interface.



Tabular views do not currently support OLE. However, you can choose **Copy Picture** from the **Table** menu to copy a table into the Windows clipboard and then paste from the clipboard into your documentation tool.

Set the HDL Import Options

Choose **HDL Import** from the **Options** menu in the design browser to display the HDL Import Wizard Options dialog box:




Select the **State Diagram** option for leaf-level descriptions and confirm the dialog box.

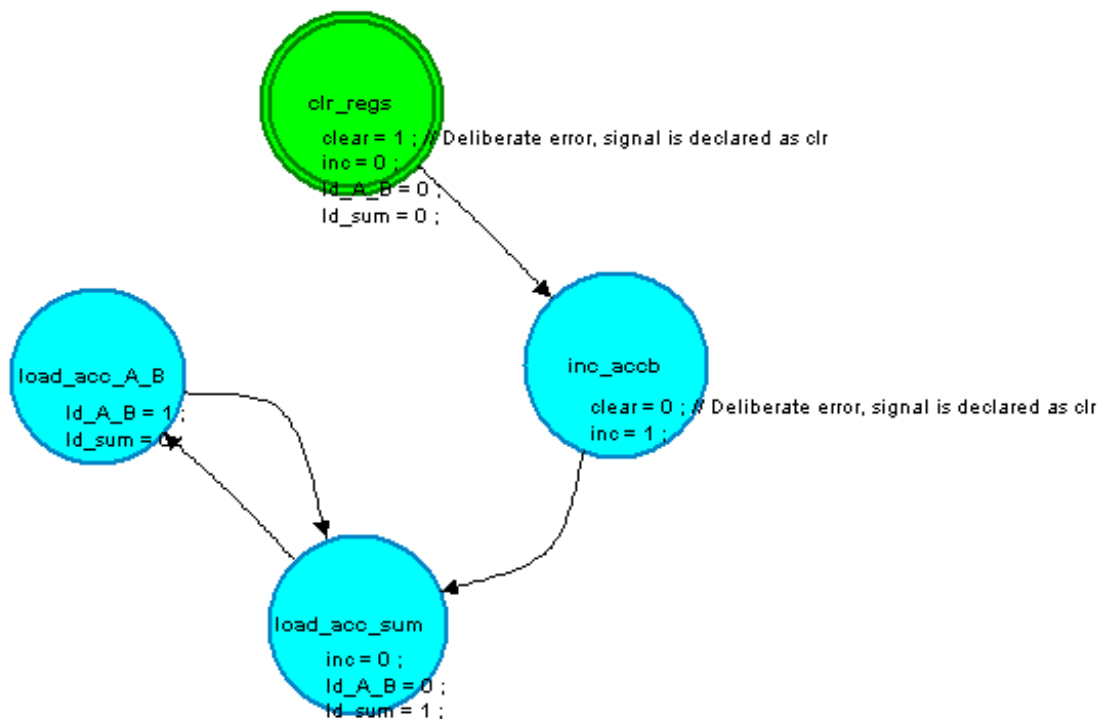


If the **Flow Chart** radio button below the state diagram option is set, any source view not recognized as a state machine is imported as a flow chart.

Recover a State Diagram View

Select the *FSM* instance in the source browser and use the  button.

Notice that the icon used for the *FSM* instance changes to  indicating that it has been converted to a [state diagram](#) view. Double-click to open the state diagram which should look similar to the following picture:



This simple state machine comprises a [start state](#) (*clr_regs*) and three [simple states](#) (*inc_accb*, *load_acc_sum* and *load_acc_A_B*) connected by [transitions](#).

You can move or resize objects, add panels and edit the title block or make any other non-logical edits which would not change the logical definition of the state machine.

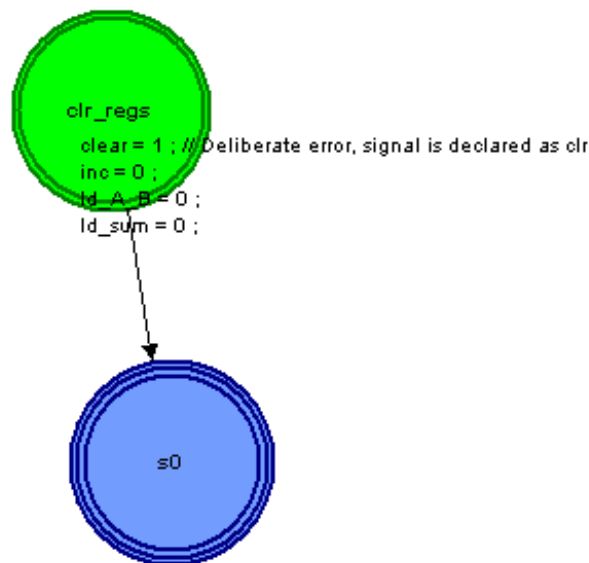
Notice that there is a deliberate signal assignment error in the [actions](#) specified for the *clr_regs* and *inc_accb* states. HDL Detective does not allow you to correct these errors (since this would change the logical definition). However, you could edit the original source code outside of HDL Detective and re-import the code to view the corrected state diagram.

Relevel the State Diagram

The example used in this tutorial comprises only four states but [state machines](#) recovered from real designs may have many states. In these cases, it can be useful to break the initial flat state diagram into one or more hierarchical diagrams.

Select the *inc_accb*, *load_acc_sum* and *load_acc_A_B* states and choose **Add Hierarchy** from the **Re-level** cascade in the **Diagram** or popup menu.

Notice that the selected states are replaced by a [hierarchical state](#) (*s0*).

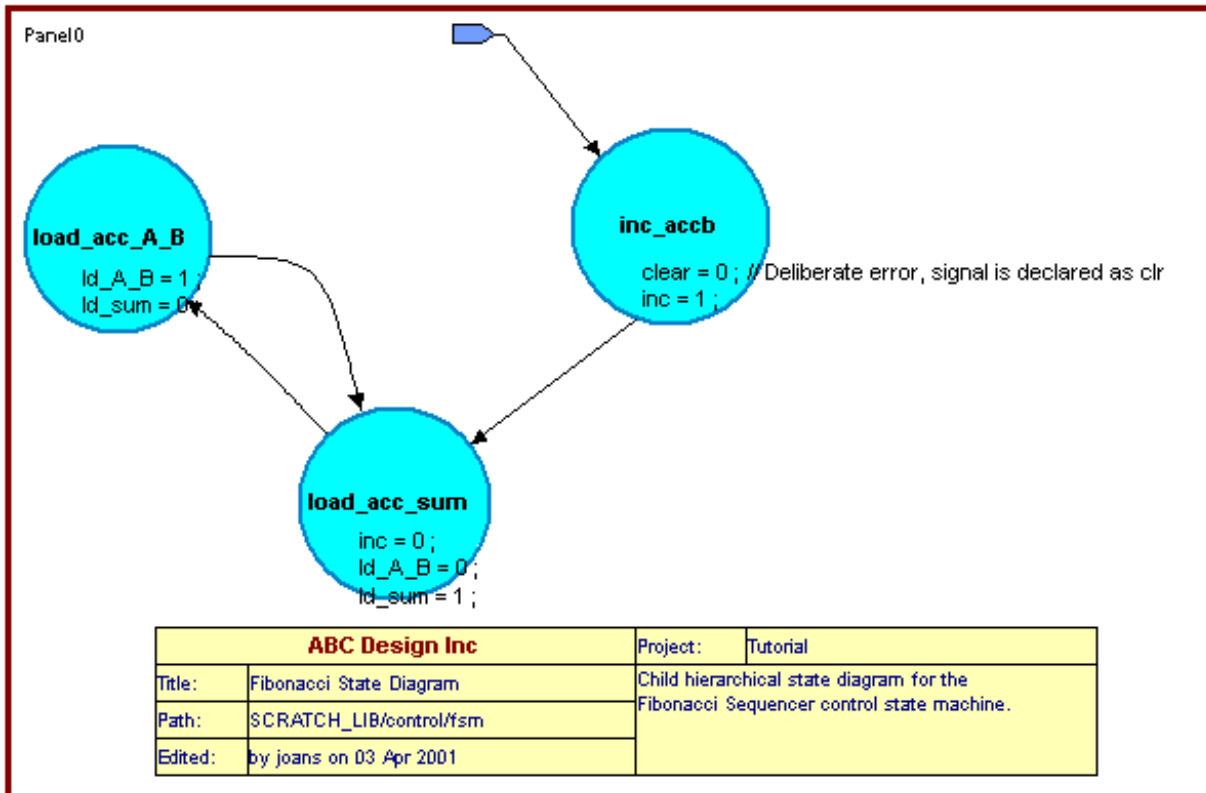


The selected states have been moved into a new [child](#) hierarchical state diagram which can be opened by double-clicking on the hierarchical state. The child diagram is saved as part of the same design unit view and is logically identical to the original flat diagram.

You can remove state machine hierarchy by selecting the hierarchical state and choosing **Remove Hierarchy** from the **Re-level** cascade in the **Diagram** or popup menu. This operation brings all states in the child diagram up a level into the parent diagram.



Any hierarchical state diagram can be printed, exported or inserted into a design document using OLE in the same way as was previously described for block diagrams.

You can also add a title block and comment text, comment graphics or panels to any diagram in the hierarchy. For example, a title block and a panel have been added to the following child state diagram view:

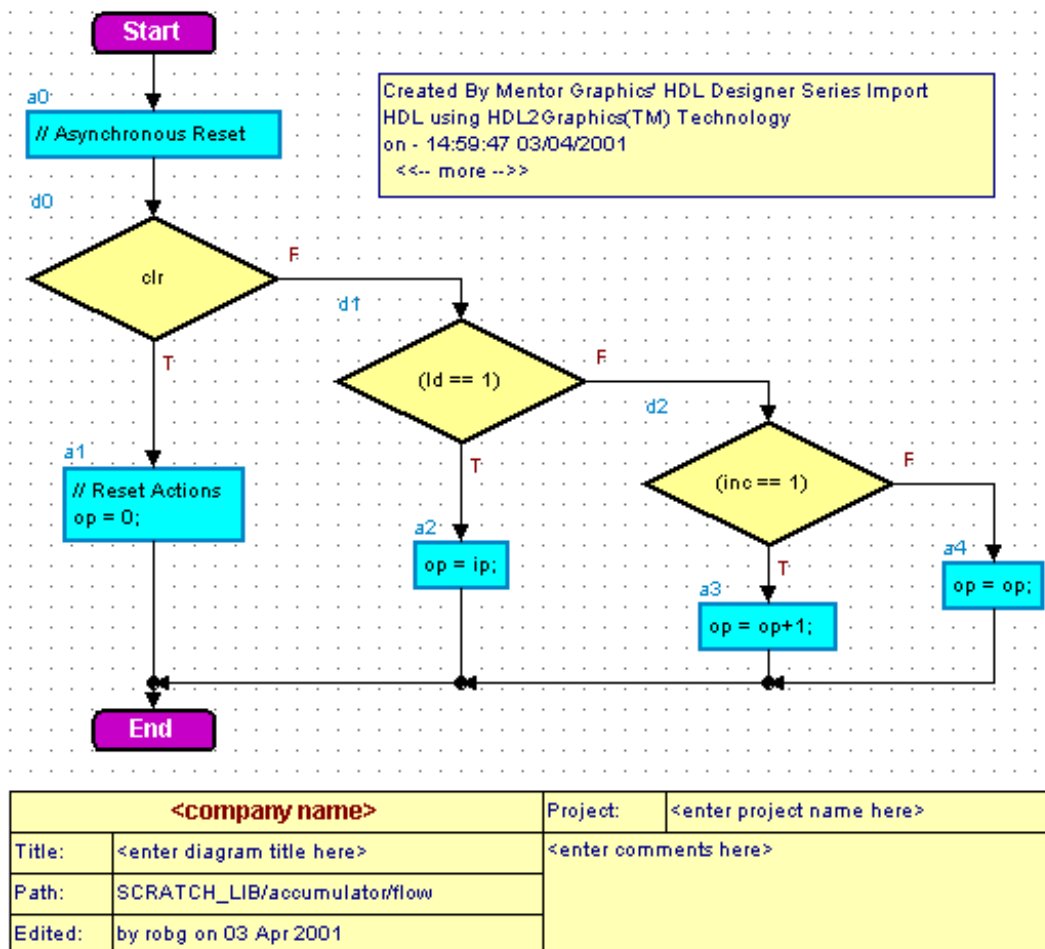


Recover a Flow Chart View

Any HDL text view can be optionally recovered as a [flow chart](#). The alternative flow chart option is set by default when you set the state diagram HDL import option and any HDL text view which is not recognized as a state machine is automatically converted to a flow chart.

Select the *acc_sum* instance of the *accumulator* design unit view in the source browser and use the  button. Since the *accumulator* is not recognized as a state machine, it is converted to a flow chart. Although there are three instances of the *accumulator* in the Fibonacci design they are described by the same design unit view and the icon used for all instances in the source browser changes to .

Double-click to open the flow chart which should look similar to the following picture:



If you are using VHDL, the flow chart has an additional decision box for the clock event as shown in the picture on [page A-6](#).

Notice how the HDL code is represented by a number of separate **action boxes** with conditional statements represented by **decision boxes**. A more complex design may also include **case boxes**, **loops** and **wait boxes**.

A flow chart can be printed, exported or inserted into a design document using OLE in the same way as was previously described for block diagrams and state diagrams. You can also add a title block and comment text, comment graphics or panels.

You have now completed the Design Exploration Tutorial. For more information about how a HDL design can be visualized using graphical diagrams, refer to the online help topic.

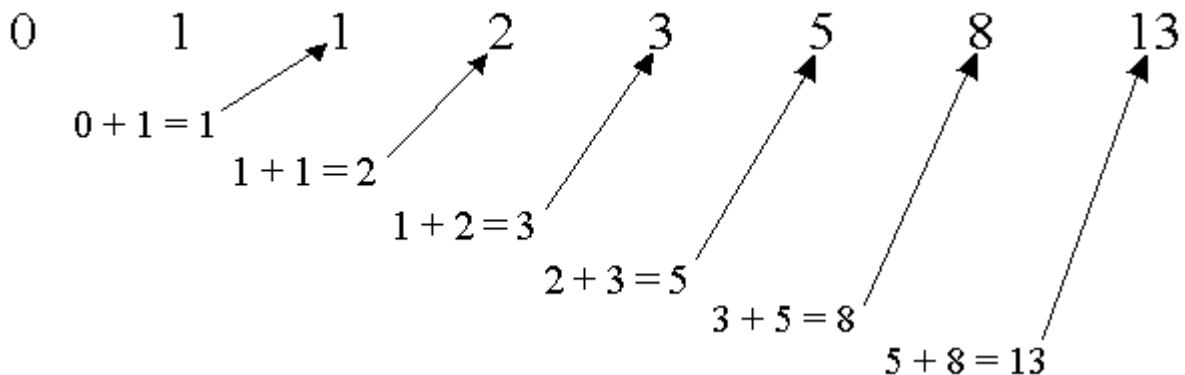
For more information about the Fibonacci Sequencer example used in this tutorial, refer to [Appendix A](#).

Appendix A

Fibonacci Sequencer Design

The Fibonacci Sequencer design is provided as source VHDL and Verilog code which can be imported as a graphical or text design. Examples of the imported design containing alternative graphical views are also provided.

The design implements a simple Fibonacci count sequence which repeatedly sums the previous two numbers.



Source Code

The source code is located in the *examples* sub-directory of your HDL Designer Series installation. For example if HDS is installed in *D:\Builds\hds2001*, the path is:


```
D:\Builds\hds2001\examples\tutorial_ref\Import
```

You can use the filelists in these directories to recover the complete designs or you can recover the test bench and sequence generator individually.

Importing the Design

You can import the design by choosing **Full HDL Import** or **Text HDL Import** from the **HDL Import** cascade in the design browser **HDL** menu to display the HDL Import wizard.

You can choose to specify the source files individually or to read them from the supplied filelist. If you use the filelist, a placement file is automatically referenced which modifies the default layout and adds a panel to the block diagram. Note however that the recovered layout may also be influenced by the current HDL import preferences.

The design can be imported into any existing library or you can use the  button to set a new library mapping. You can choose to recover all design units in the design or any selected design unit.

You can use [HDL2Graphics](#) to recover graphical views by selecting the **Block Diagram** or **IBD** option for hierarchy descriptions and also setting the **State Diagram** and **Flow Chart** options for leaf-level descriptions.

You can import the entire design as HDL text design units by selecting the **HDL Text** options for both hierarchy descriptions and leaf-level descriptions.

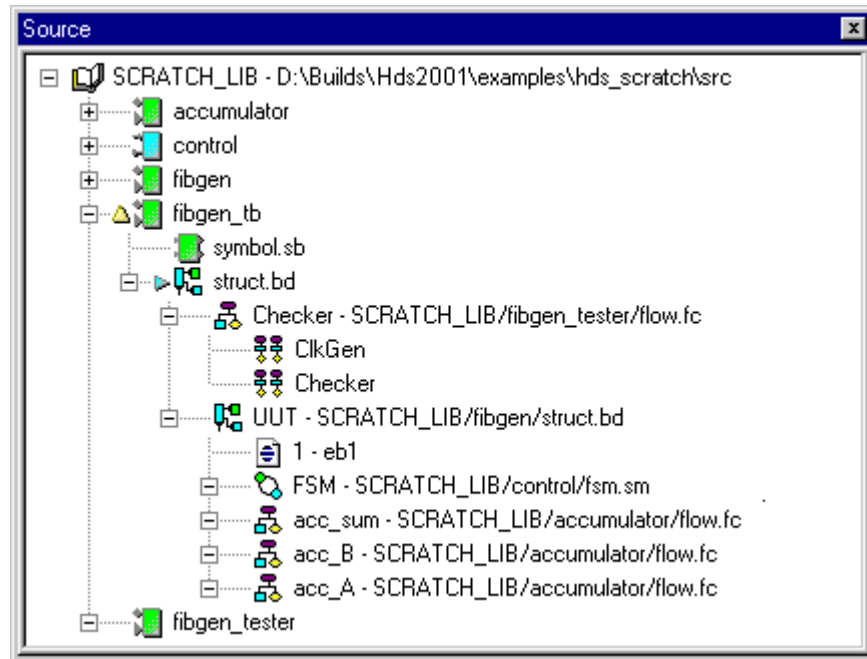
The Recovered Design

The complete recovered design comprises five design units: *accumulator*, *control*, *fibgen*, *fibgen_tb* and *tester*.

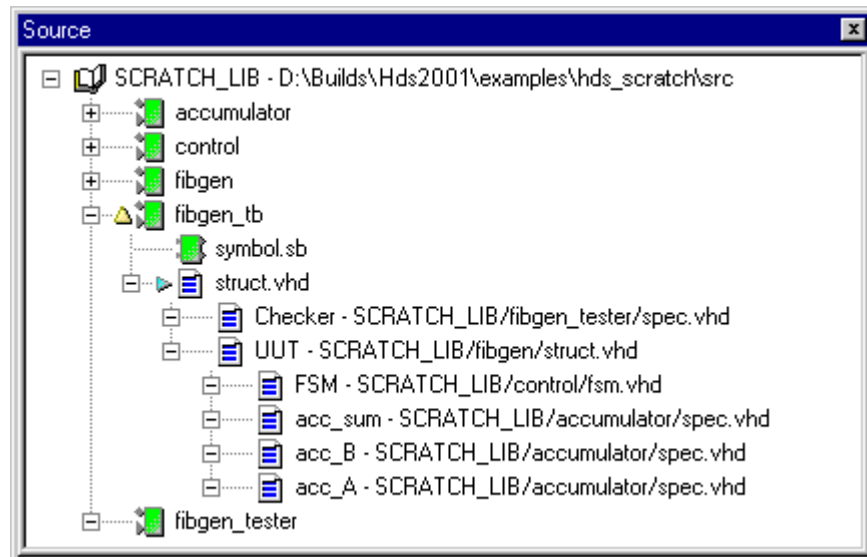
The *fibgen* and *fibgen_tb* design units are recovered as block diagram or HDL text views representing the structure of the design. The *accumulator* and *control* design units are recovered as instantiated views on the *fibgen* view. The *tester* design unit is similarly recovered as an instantiated view on the *fibgen_tb* view.

You can optionally recover the *control* design unit as a graphical state machine or flow chart. The *accumulator* and *tester* design units can be recovered as flow charts or HDL text views.

The following picture shows the expanded design units in the source browser when the VHDL design is recovered using graphical views:



The following picture shows the Verilog design recovered as HDL text views:



The Accumulator Design Unit

The Fibonacci sequencer is based on three instantiations of an *accumulator* component. If you are using VHDL, the *accumulator* component is described by the following HDL text view:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY accumulator IS
  PORT(
    clock : IN      std_logic  ;
    clr   : IN      std_logic  ;
    inc   : IN      std_logic  ;
    ip    : IN      std_logic_vector (7 DOWNTO 0) ;
    ld    : IN      std_logic  ;
    op    : BUFFER  std_logic_vector (7 DOWNTO 0)
  );
END accumulator ;
ARCHITECTURE spec OF accumulator IS
BEGIN
  truth_process: PROCESS(clock)
  BEGIN
    IF (clock'EVENT AND clock = '1') THEN
      IF (clr = '1') THEN
        -- Reset Actions
        op <= "00000000";
      ELSE
        -- Block 1
        IF (ld = '1') THEN
          op <= ip;
        ELSIF (inc = '1') THEN
          op <= unsigned(op)+1;
        ELSE
          op <= op;
        END IF;
      END IF;
    END IF;
  END PROCESS truth_process;
END spec;
```

If you are using Verilog, the *accumulator* component is described by the following HDL text view:

```
// Module accumulator.spec

// External Declarations
module accumulator(clock, clr, inc, ip, ld, op);

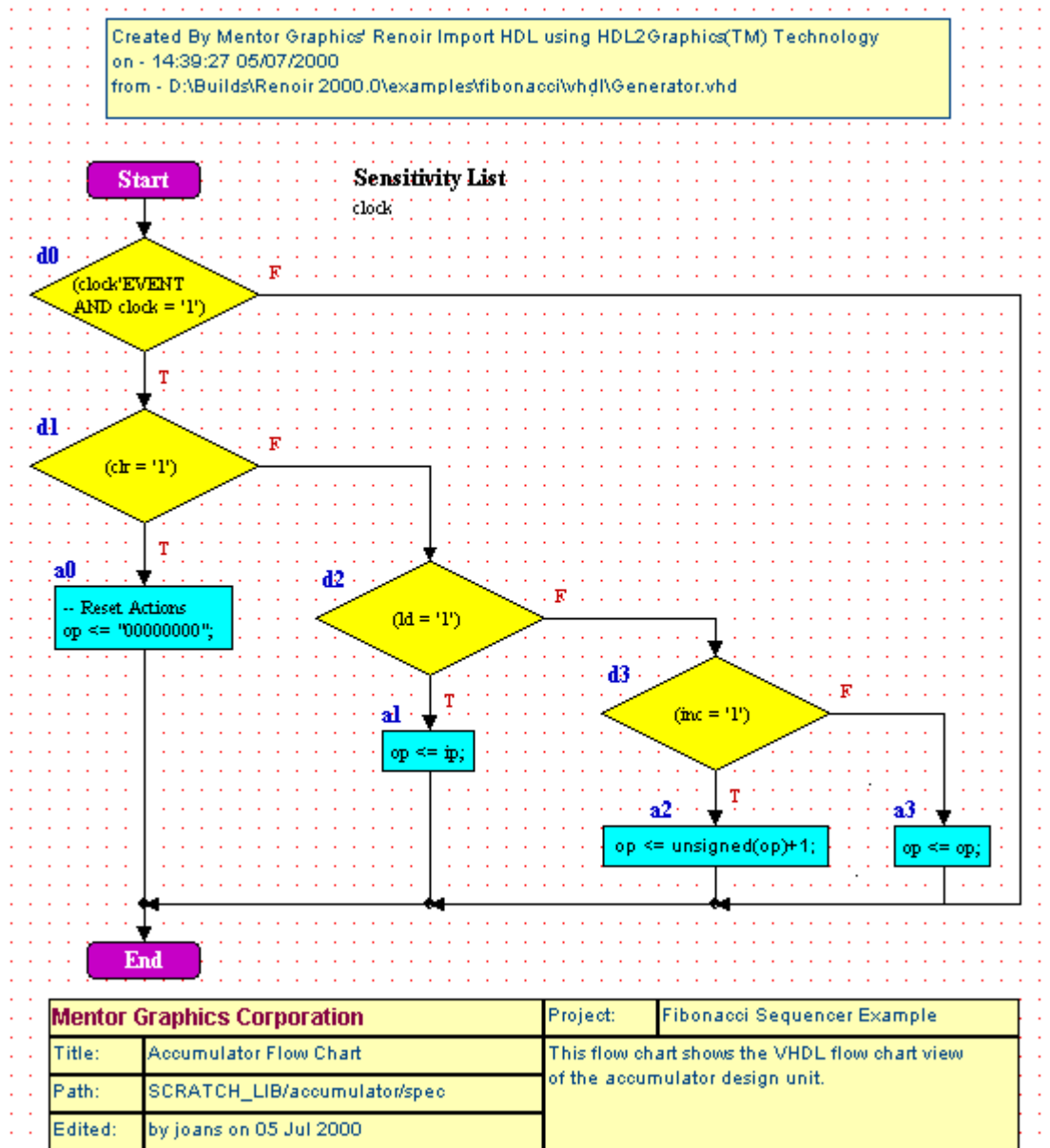
// Internal Declarations
input      clock;
input      clr;
input      inc;
input [7:0] ip;
input      ld;
output [7:0] op;
wire [7:0] ip;
wire ld;
wire clr;
wire inc;
wire clock;
reg [7:0] op;

always @ (posedge clock) begin
    if(clr) begin
        // Reset Actions
        op = 0;
        end
    else begin

        if ((ld == 1))
            op = ip;
        else if ((inc == 1))
            op = op+1;
        else
            op = op;
        end
    end
end

endmodule // accumulator
```

If you have recovered the *accumulator* design unit as a graphical flow chart, it will look similar to the following picture:



The VHDL flow chart is shown. The HDL syntax is slightly different for Verilog as shown on [page 17](#)

If you created your own *accumulator* design unit, it could also be drawn as a truth table. For example if you are using VHDL:

| | A | B | C |
|---|-----|-----|----------------|
| 1 | ld | inc | op |
| 2 | '1' | | ip |
| 3 | | '1' | unsigned(op)+1 |
| 4 | | | op |
| 5 | | | |



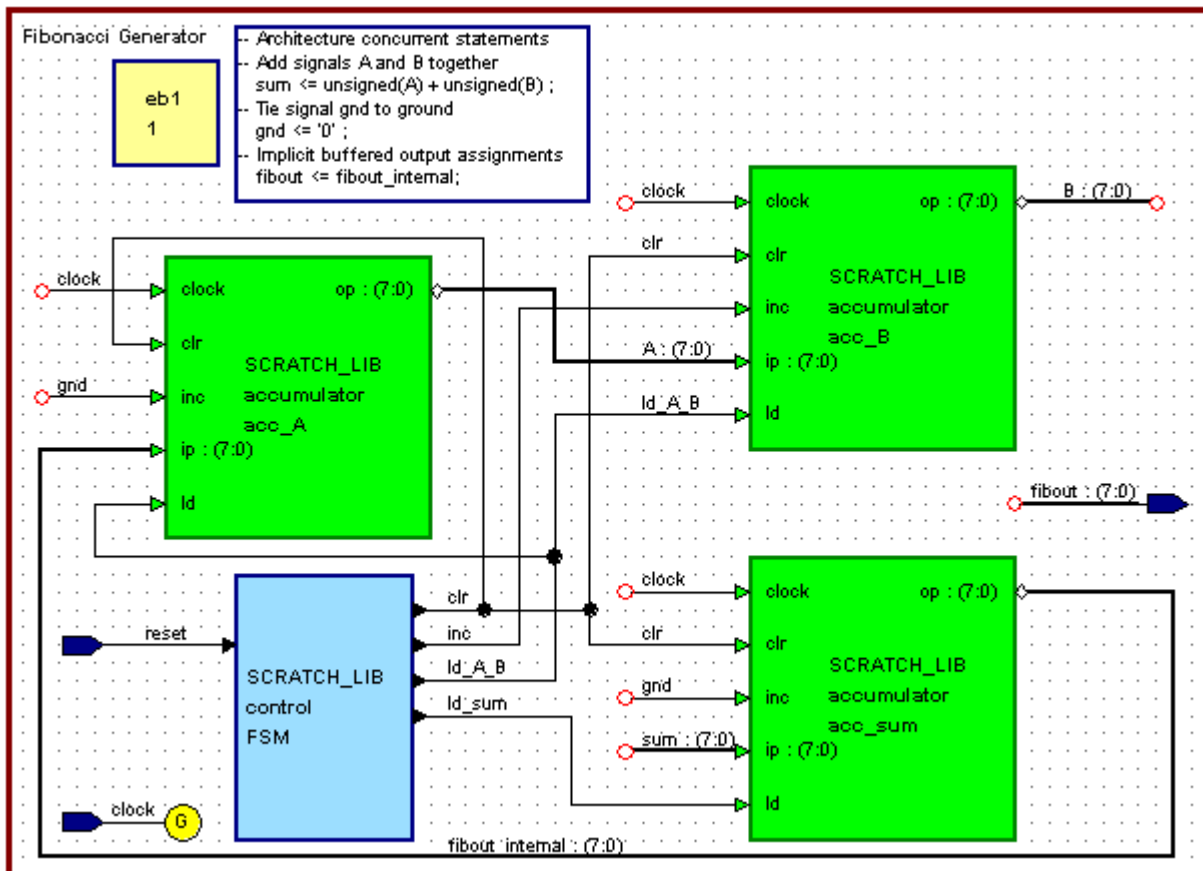
Truth table views can only be created using **HDL Author - Graphics**, **HDL Author - Pro**, **HDL Designer Graphics** or **HDL Designer - Pro**. However, existing truth tables can be opened readonly by any HDS tool.

When the accumulator is described graphically, it is necessary to specify the clock and reset conditions in the generation properties for the diagram as follows:

| | |
|-------------------|---|
| Table: | Sequential |
| HDL Style: | If |
| Clock: | Rising <i>clock</i> |
| Sensitivity list: | Auto |
| Reset actions: | <i>op</i> <= "00000000"; (VHDL) or <i>op</i> = 0; (Verilog) |
| Reset: | High <i>clr</i> |
| Mode: | Synchronous |


The *Fibgen* Block Diagram

The recovered *Fibgen* block diagram should look similar to the example below:

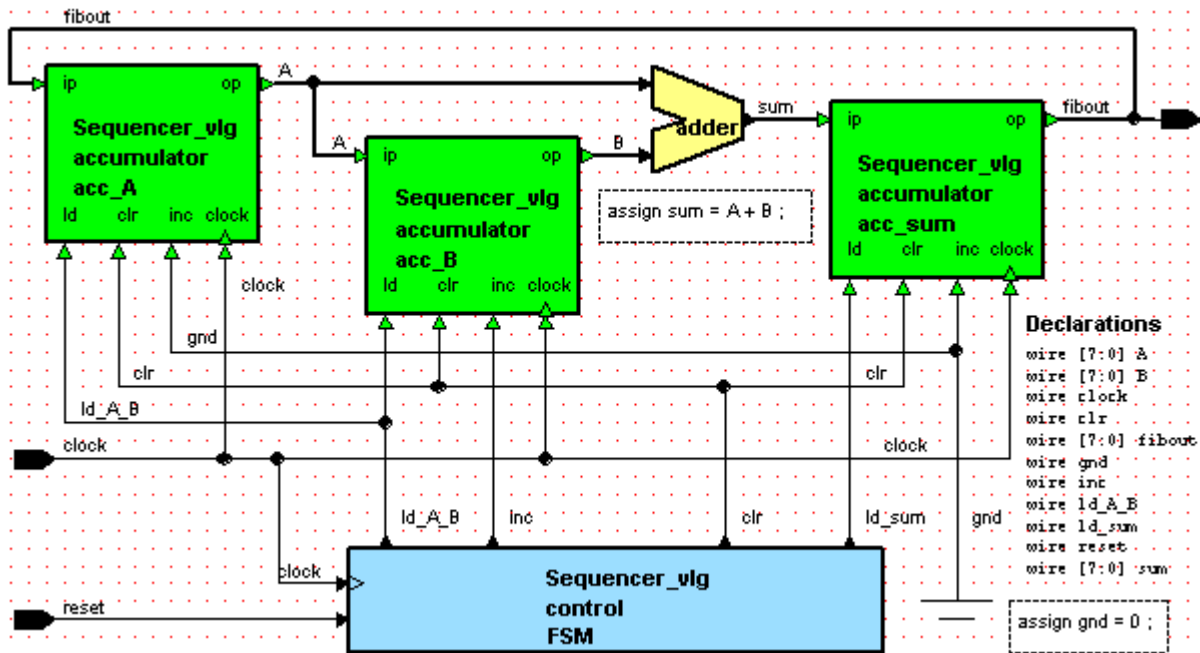




The block diagram comprises three component instantiations of the *accumulator* design unit connected to a *control* state machine block.

Notice that an embedded HDL text view (of embedded block *eb1*) is used to sum the outputs from *acc_A* and *acc_B* and provide the *ip* signal to the *acc_sum* accumulator. The HDL text also ties the *inc* input on *acc_B* and *acc_sum* to the *gnd* signal (logic 0) and assigns the *fibout* output to the *fibout_internal* output from *acc_sum*. These signals are connected by name and do not need to be physically connected on the diagram.

Notice also that the outputs (*op*) from the accumulators are recovered as buffer ports in the VHDL design (shown as )

The following picture shows an alternative layout for the Verilog *Fibgen* block diagram which is functionally identical to the recovered design:



In this version, one embedded block (named *adder* and drawn using an arithmetic logic unit shape) is connected to sum the outputs from *acc_A* and *acc_B*. A separate embedded block is used for the *gnd* connection and has been given a logical ground shape . Notice also how a clock indicator  has been used on the *clk* input to each block and component.

The Control State Machine

The recovered *control* state machine includes two deliberate mistakes which give rise to an error report when you generate HDL for the design.

VHDL:

"control_fsm.vhd", line 122: Error, identifier 'clear' is not declared

Verilog:

"control.v", line 75: Error, illegal use of wire for 'clear'

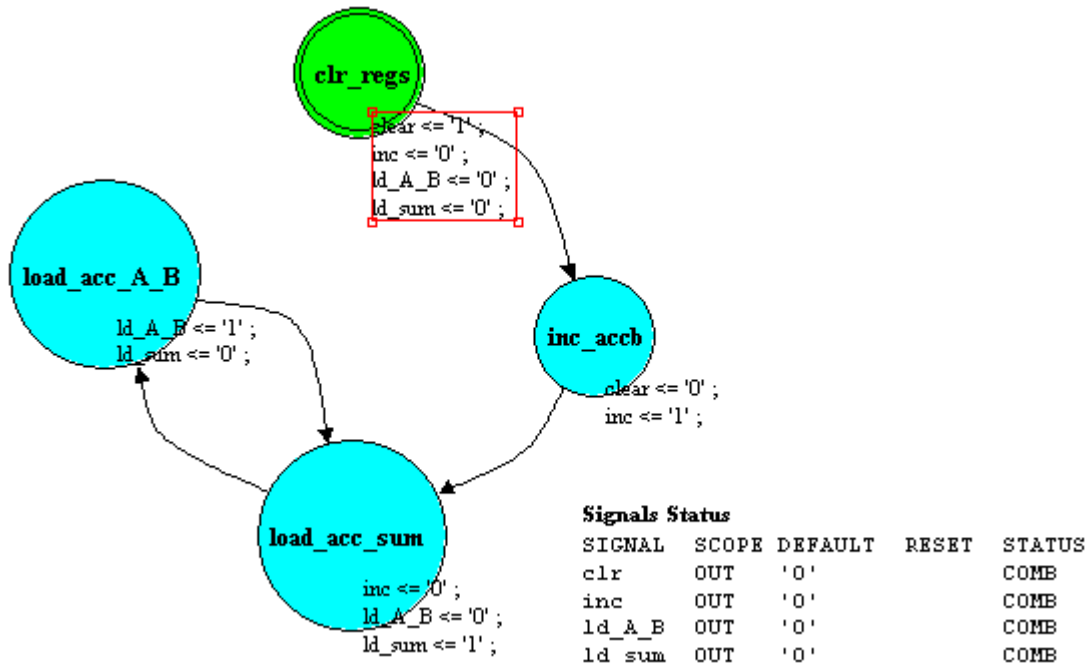
"control.v", line 86: Error, illegal use of wire for 'clear'



One error message is issued if you are using VHDL or two messages if you are using Verilog.

These errors have been included to illustrate the error cross-referencing feature which allows you to jump directly to the location of an error on the graphical diagram by double-clicking on the error message in the log window.

For example, the following picture shows the VHDL state diagram:

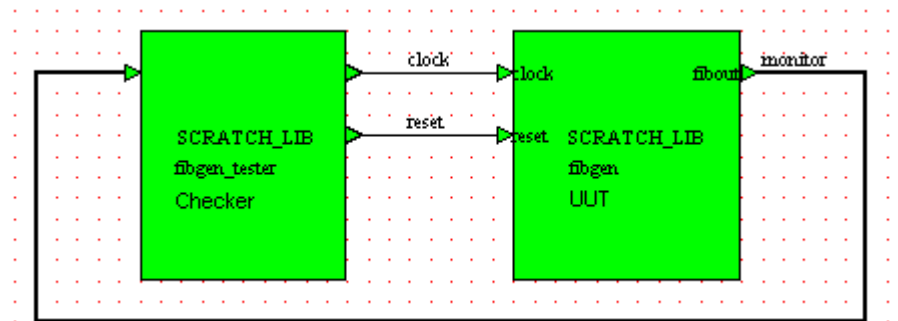


The error messages are issued because a signal *clear* has been used in the state actions for the *clr_regs* and *inc_accb* states although the actual signal declared in the interface by the parent block diagram is *clr*.

Once you have corrected the error by correcting the name of the signal used in the state actions, you should be able to regenerate HDL for the design without any further errors.

The Fibgen_tb Test Bench

The *fibgen_tb* test bench is a block diagram containing an instantiation of the *fibgen* unit under test and a corresponding *fibgen_tester* checker component which provides the *clock* and *reset* signals to the design. The *fibgen_tester* component also has a *monitor* input which examines the *fibgen* output signal.



The Fibgen Tester Design Unit

The *fibgen_tester* design unit implements a clock with a 100ns period and 50% duty cycle using the following concurrent HDL code:

VHDL

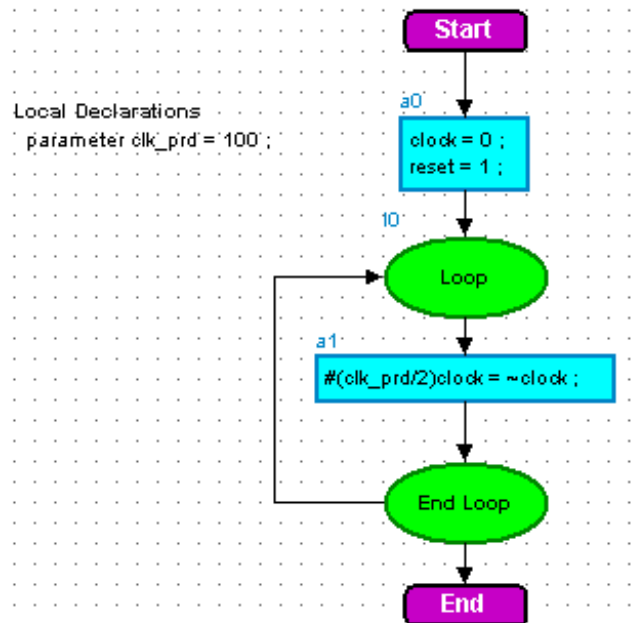
```
-- Architecture declarations
CONSTANT clk_prd : time := 100 ns ;
SIGNAL int_clock : std_logic := '0';

-- Architecture concurrent statements
-- Clock Generator
int_clock <= not int_clock AFTER clk_prd / 2;
clock <= int_clock;
```

Verilog

```
// Concurrent statements
// Clock Generator
initial
begin : ClkGen
parameter clk_prd = 100 ;
clock = 0 ;
reset = 1 ;
forever #(clk_prd/2)clock = ~clock ;
end
```

If you have enabled flow charts in the HDL2Graphics options, the Verilog clock generator is recovered as a concurrent flow chart with the *clk_prd* parameter declared as a local declaration.



Note that the reset signal must be initialized to 1 in the Verilog design.

There is a separate process to monitor the *fibgen* output.

VHDL

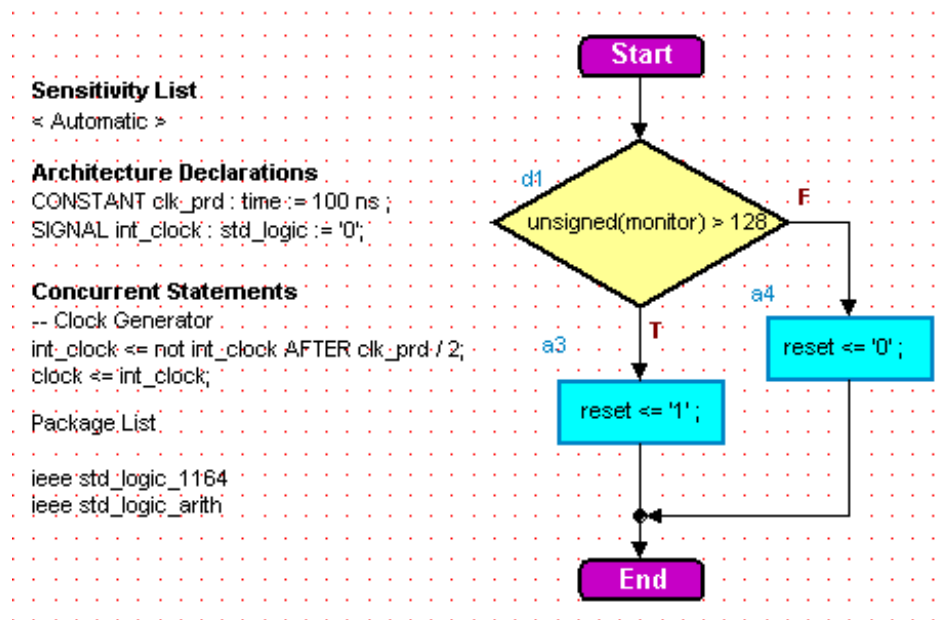
The *fibgen* output is monitored by the following VHDL code which prevents the design from overflowing by resetting when the *fibgen* output is greater than 128:

```

-----
process1 : PROCESS (monitor)
-----
BEGIN
  IF unsigned(monitor) > 128 THEN
    reset <= '1' ;
  ELSE
    reset <= '0' ;
  END IF;
END PROCESS process1;

```

Alternatively, if you enabled flow charts in the HDL2Graphics options, the following chart is recovered:



Verilog

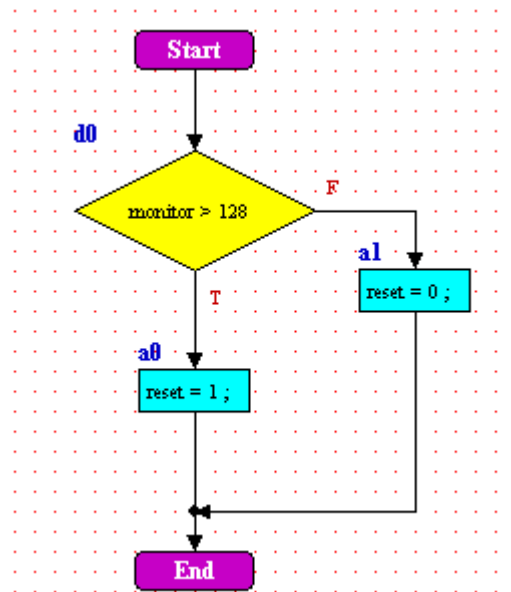
The *fibgen* output is monitored by the following Verilog code::

```

// Flowchart process1
always @ (monitor)
begin : process1
    if (monitor > 128) begin
        reset = 1 ;
    end
    else begin
        reset = 0 ;
    end
end
end

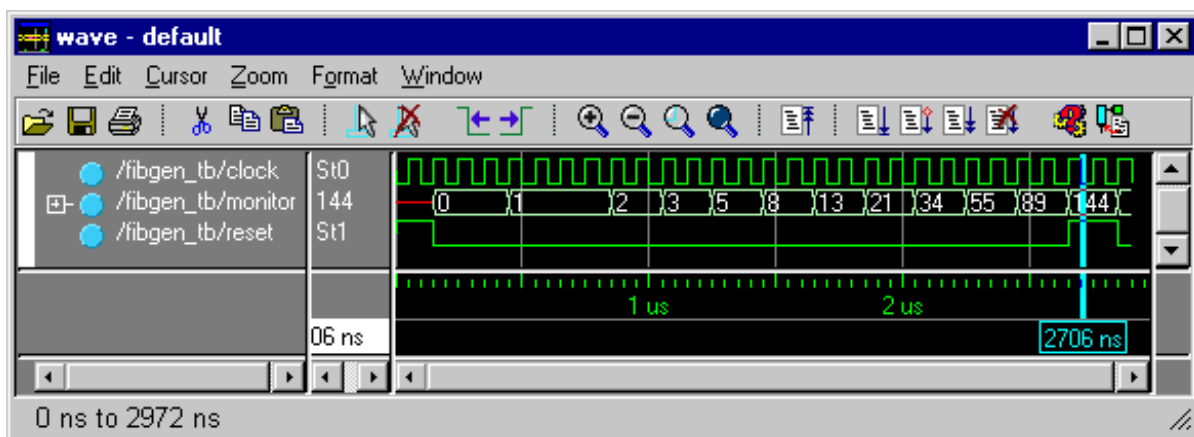
```

Alternatively, if you have enabled flow charts in the HDL2Graphics options, the concurrent flow chart shown overleaf is recovered.



Simulation Results

The *fibgen_tb* test bench can be used to simulate the Fibonacci sequencer design. For example, if you display a ModelSim wave window for the clock, reset and monitor signals, and run the simulator for 3000 ns, the following waveforms should be produced:



Notice that the count sequence automatically resets to zero after counting up to 144. The simulation can be run for any period, automatically resetting the count each time the output exceeds 128.

Graphical Designs

The *examples* sub-directory also includes completed graphical VHDL and Verilog versions of the Fibonacci sequencer design. These can be accessed from the *Sequencer_vhd* or *Sequencer_vlg* library in the design browser.

The example designs include alternative truth table flow chart and HDL text views for the *accumulator* design unit and alternative flow chart and HDL text views for the *fibgen_tester* design unit.

The HDL text views are set as the default views for these design units so that the HDL can be regenerated using **HDL Pilot**. However, the alternative views can be made the default view by using the **Set Default View** command from the design browser **Edit** menu.

