Chapter 9

WATCH Design - Hardware Verification Tutorial

This chapter demonstrates how to use the Hardware Debugger to download, verify, and debug a single design using a Xilinx demonstration board as your target device. This chapter contains the following sections.

- "Preparing for the Tutorial"
- "Testing the Design Using a Demonstration Board"
- "Downloading and Verifying the Bitstream"
- "Synchronous Debugging"
- "Asynchronous Debugging"

Preparing for the Tutorial

This tutorial uses an XC4003E design and is targeted at the FPGA demonstration board. The WATCH tutorial can be downloaded from http://www.xilinx.com/support/techsup/tutorials

In addition to the demonstration board, you also need the XChecker cable. Although other cables are available from Xilinx, the XChecker cable is currently the only cable that supports readback capabilities.

Note: This tutorial assumes that you are familiar with your schematic entry tool and the Xilinx Design Manager. It also assumes that you have implemented your design. If you are not familiar with these tools or have not implemented your design, complete the tutorials for those tools prior to proceeding or see the *Quick Start Guide*.

Testing the Design Using a Demonstration Board

The FPGA demonstration board includes both an XC3000 family socket and an XC4000 family socket. This tutorial only targets the XC4000 family.

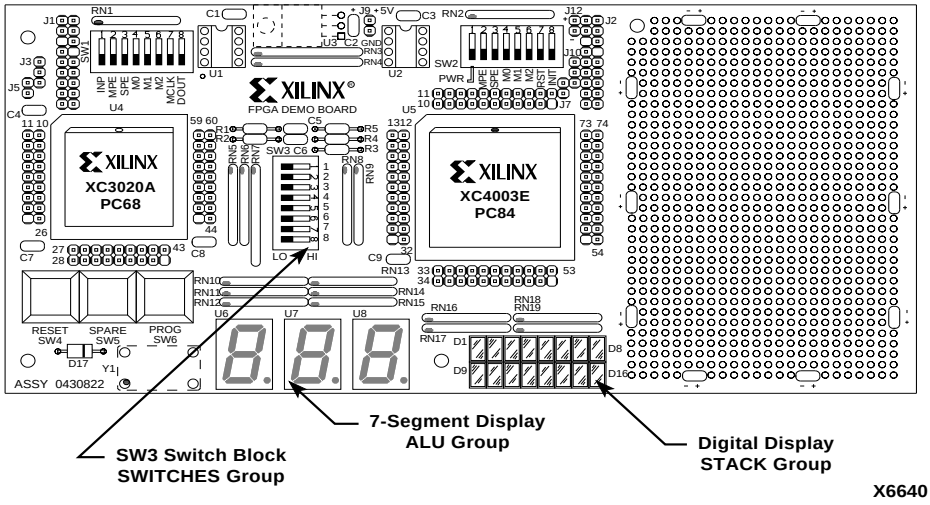

Figure 9-1 FPGA Demonstration Board Components

Preparing the Design for Readback

Locate the WATCH tutorial for the XC4003E device and edit it as explained in this section to enable readback. This step is not necessary if you intend only to download your design.

1. Include the DEBUG_CKT macro in your WATCH design.
2. Connect the DEBUG_CLK_8M pin of the DEBUG_CKT macro to the F8M pin of the OSC4 symbol.
3. Connect the SYS_CLK_15HZ pin DEBUG_CKT macro to the F15 pin of the OSC4 symbol.

4. Connect the CLK pin of the DEBUG_CKT macro to the input of the BUFG symbol.

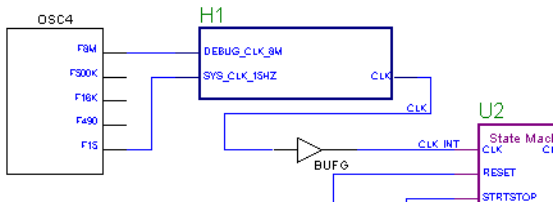

Figure 9-2 DEBUG_CKT Symbol Connections

The DEBUG_CKT macro provides three functions used in debugging the WATCH design. These are not all necessary, but are offered as examples for accomodating in-circuit testing.

1. The READBACK Symbol.

   The “Readback Symbol Connections” figure shows a detailed view of the READBACK symbol and its connections.

Figure 9-3 Readback Symbol Connections

   The READBACK symbol is the only necessary component for enabling readback verifying and capturing features. While the TRIG and DATA could be routed to any user I/Os, the MD0 and MD1 are used, respectively, because these are connected to the the J2 header pins on the demonstration board to simplify the XChecker cable connections to the RT and RD flying leads, respectively..

2. The STARTUP symbol.

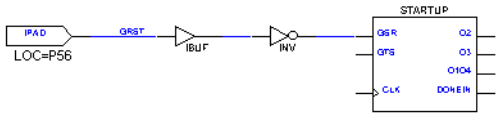

Figure 9-4 STARTUP Symbol Connections

The STARTUP symbol provides access to the GSR (GlobalSetReset) net which when asserted re-inititializes all the flip-flops in the FPGA. For debugging purposes this will be connected to the RST flying lead so that the Hardware Debugger can assert a global reset. In this design the connection to the GSR is inverted because the GSR pin is active HIGH while the RST pin of the XChecker cable is active LOW. The GRST input signal is constrained to P56 so that the RESET button of the demonstration board may also be used to assert the GSR.

3. Clock MUXing network.

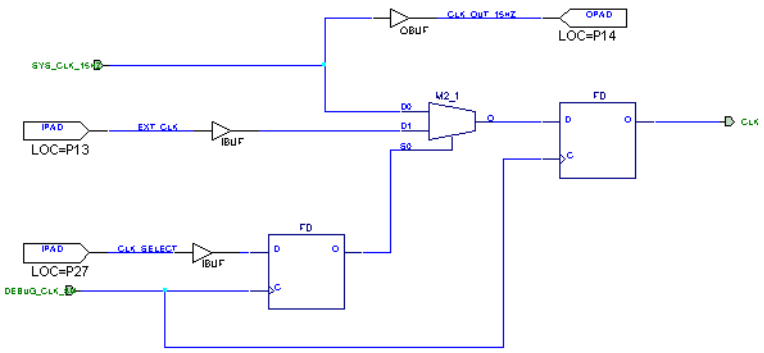

Figure 9-5 Clock Muxing Circuitry

The WATCH design utilizes the internal FPGA oscillator as the system clock. Synchronous debugging requires interupting this connection so that the clock input may be driven by the cable and

thus controlled by the Hardware Debugger. This provides an opportunity to demonstrate good design practice when multiplexing clock signals in an FPGA.

Though the WATCH design is a low speed application that would not be critically affected by clock "glitching", it is generally considered poor design practice to "gate" clocks. Therefore, a flip-flop registers the output of the MUX2 to remove glitching and restore the clock phase. The select (S0) of the MUX2 is registered for switch debouncing. There are many other clock multiplexing methods that may be better for other applications.

The CLK_SELECT input is constrained to Pin 27 of the XC4003E. This site can be controlled by the SW3-7 switch on the demonstration board. Placing this switch in the open position will select the internal oscillator for the system clock. Closing this switch will select an external clock located at P13. This external clock input can then be driven by the CLKO flying lead of the XChecker cable and thus controlled by the Hardware Debugger.

Generating a Bitstream

The next step generates a bitstream for the WATCH design from the Design Manager. To generate a bitstream, you must open the implemented WATCH design from the Design Manager.

Note: If you do not have a project for the WATCH design, use the Design Manager to create a project for that design. If a project already exists for the WATCH design but the project has not been updated, run the **`Design`** → **`New Version`** command to read in the changes made in the schematic and to create a new version reflecting the updated schematic. Then, implement the design as explained in the "Implementing a Design" section of the *Design Manager/Flow Engine Reference/User Guide*.

Set the Configuration Bitstream options as explained in this section to generate a configuration file that you can use for programming, verifying, and debugging XC4000E designs. You must enable a pull-up resistor for the DONE pin, which is used for device configuration. You must also enable readback for the device.

1. In the Design Manager project view, select the implementation revision.
2. Select **`Design`** → **`Implement`**.
3. In the Implement dialog box, click the **`Options`** button.

   The design implementation Options dialog box appears.
4. Select **`Produce Configuration Data`** in the Optional Targets group box.
5. Click the **`Edit Template`** button corresponding to the configuration template.

   The Configuration Template dialog box is displayed as shown in the "Design Manager Configuration Template Dialog Box" figure.

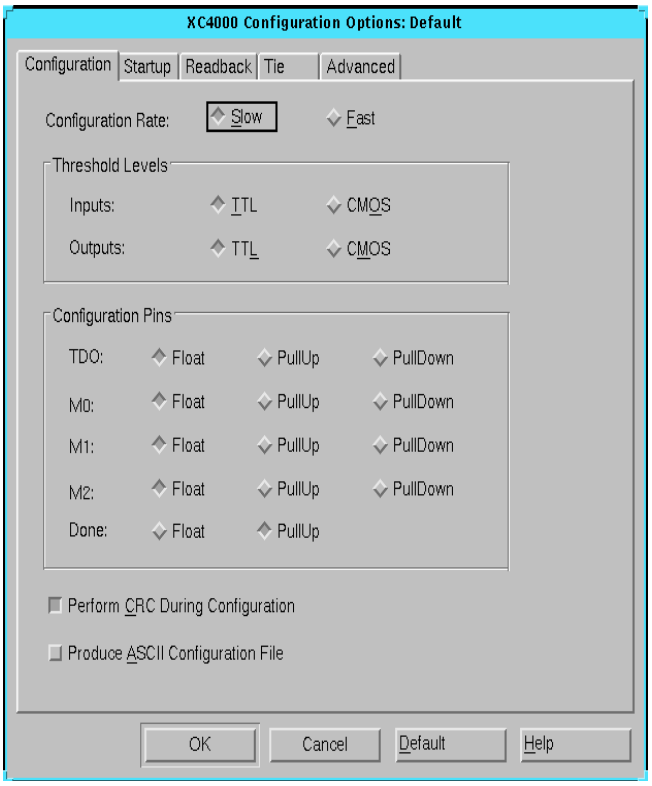

Figure 9-6 Design Manager Configuration Template Dialog Box

6. Select **PullUp** next to the DONE pin in the Configuration Pins box to enable a pull-up resistor for the DONE pin.
7. Select **Perform CRC During Configuration** to perform a CRC check of your bitstream during configuration.
8. Select the **Readback** tab.
9. Select the **CCLK** readback clock.
10. Select **Enable Bitstream Verification and In-Circuit Hardware Debugging**. This option generates the logic allocation file.

Note: For more information about configuration options, read the "Implementation Options" chapter in the *Design Manager/Flow Engine Reference/User Guide*.

11. Click **OK** to return to the Options dialog box.
12. In the Options dialog box, click **OK**.
13. In the Implement dialog box, click **Run** to compile the design and produce the configuration data.

Connecting the Cable

To load the configuration bitstream to the demonstration board, you need one of the three available hardware cables: an XChecker cable, a parallel cable, or a serial cable. All three cables work with any of the Xilinx demonstration boards; however, the XChecker cable is the only cable that supports readback verification and debugging.

Before initiating the physical downloading of the design into the FPGA on a Xilinx demonstration board, you must hook up the board correctly to your computer.

You must also connect several control and power pins between the board and the cable. The bundles of leads supplied with the cables are labeled to help you connect the board to the cable.

Finally, you must connect a pair of power and ground pins to a regulated 5 volt power supply to provide power to the board and cable.

1. Plug one end of the cable into the back of your computer.

If you are using a parallel cable, attach the cable to a parallel port. If you are using a serial cable or the XChecker cable, connect the cable to a serial port.

2. Connect the other end of the cable to your demonstration board.If using the XChecker cable you should have two different sets of jumpers available to use.

   "Flying Leads" are bound and keyed at one end, and seperate and labeled at the other. Flying Leads are shipped with each cable type.

   "XChecker Jumpers" are bound and keyed at both ends. They are shorter than the flying leads and are not labeled. XChecker Jumpers are only shipped with the demonstration board.

   For use with the WATCH Tutorial we recommend using the Flying Leads since using the XChecker Jumpers will require some additional jumpers not supplied. However, in general for downloading to the Demonstration board the XChecker Jumpers are a fast and easy method for attaching the XChecker Cable.

3. Connections from the cable to the demonstration board for downloading are shown in the "Cable Connections (Downloading)" table.

Table 9-1 Cable Connections (Downloading)

Cable Label	FPGA Board (XC4000E)
VCC	J2-1
GND	J2-3
No Connection	J2-5
CCLK	J2-7
D / P	J2-9
DIN	J2-11
XChecker and Serial Download Cable	
PROG	J2-13

Table 9-1 Cable Connections (Downloading)

Cable Label	FPGA Board (XC4000E)
XChecker Cable Only	
INIT	J2-15
RST	J2-17 (Pin 56)

Note: The RST connection is not necessary for downloading XC4000 designs. This connection is used by the Hardware Debugger for resetting the FPGA design after configuration. If you're using the Flying Lead connectors then connect the RST lead to Pin 56 of the XC4003E. If you're using the XChecker Jumpers then to make this connection you must close the J7 jumpers on the demonstration board.

The "FPGA Design Demonstration Board" chapter of the *Hardware User Guide* discusses in detail the demonstration board and how to hook it up.

4. Connect the RT and RD pins, which are used for triggering and capturing readback data. Refer to the "Cable Connections (Verification and Debugging)" table for pin location information.

Table 9-2 Cable Connections (Verification and Debugging)

XChecker Cable Label	FPGA Board (XC4000E)
CCLK	J2-7
RT	J2-2
RD	J2-4
TRIG	J2-6
CLKI	J2-16
CLKO	J2-18 (Pin 13)

5. Connect the CLKO lead to Pin 13 of the XC4003E. The CLKI and TRIG lead can be left unconnected.

Note: For synchronous debugging, if you're using the XChecker Jumpers then another jumper connection must be made from J10 to Pin 13.

6. Ensure that the power supply is connected to the demonstration board at J9 *and is turned on*.

   The power connections for the demonstration board are shown in the "Demonstration Board Power Connections" table.

Table 9-3 Demonstration Board Power Connections

FPGA Board	
J9-1	+5 volts
J9-2	Gnd

FPGA (XC4000E) Demonstration Board

Make sure the FPGA demonstration board is set up for slave mode configuration. The configuration mode for the XC4000E family part is controlled by the SW2 bank of switches. Set the switches as shown in the "SW2 Switch Settings for XC4000E Configuration" table.

Table 9-4 SW2 Switch Settings for XC4000E Configuration

Switch	Label	Setting
SW2-1	PWR	Don't Care
SW2-2	MPE	Open
SW2-3	SPE	Open
SW2-4	M0	Closed
SW2-5	M1	Closed
SW2-6	M2	Closed
SW2-7	RST	Closed
SW2-8	INIT	Open

Note: The RST switch SW2-7 must be Open in order to configure the XC4000E device without disturbing the XC3000 if it has already been configured. However, this tutorial utilizes only the XC4000E and the WATCH design requires this connection to be Closed so that the RESET button is connected to the GSR input at Pin 56.

Downloading and Verifying the Bitstream

After the cable is connected to your computer, you can download the bitstream. If you are using an XChecker cable, you can also verify the design.

1. From the Design Manager, select **`Tools`** → **`Hardware Debugger`** or click the Hardware Debugger icon.

   The Communications dialog box appears.

2. Click a cable type in the Cable Type field. Select the auto detect option or the correct port from the Port drop-down list box and the appropriate baud rate from the Baud Rate drop-down list box.

   After you have used a certain kind of cable and set the correct port, the information is saved in a file called *design_name*.xck in your design directory, so you do not have to specify it each time.

Note: If you invoked the Hardware Debugger from within the Design Manager after selecting the desired implementation revision in the project view, the configuration file is already loaded in the Hardware Debugger. If you did not select an implementation revision, complete steps 4 and 5, following, to load a bitstream file.

3. Select **`File`** → **`New`** → **`Project`**.

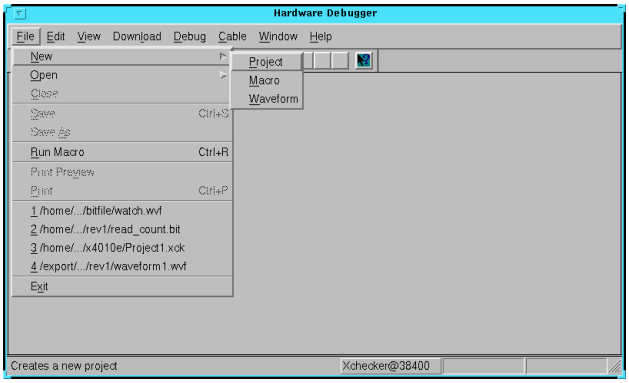

Figure 9-7 Hardware Debugger GUI

4. Select the input file name: **WATCH.BIT** and click **OK**.

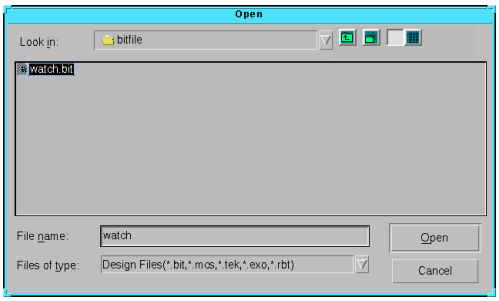

Figure 9-8 New Project Dialogue Box

If the Hardware Debugger informs you that the LL file is not found with the following pop-up box then go back through the "Generating a Bitstream" section of the tutorial to generate an LL file and set the apprpriate readback options.

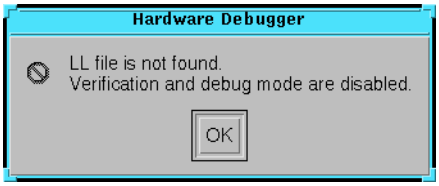

Figure 9-9 Pop-up Message Box

5. If you are using the serial download cable to program an XC4000 family part, press the PROG button on the demonstration board. This step is not necessary if you are using the XChecker cable or the parallel cable.
6. Select **Cable** → **Logic Level of Pins** to check that the status of the Done pin is Low.
7. Select **Download** → **Download Design** or click the following toolbar button.

If you are using an XChecker cable, select **Download** → **Download and Verify** or click the following toolbar button if you want to verify the design. You must also connect the RT and RD pins for readback to be available.

If the FPGA is successfully configured, the following message appears.

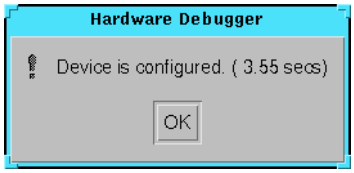

Figure 9-10 Pop-up Message Box

If the DONE signal does not go High, check the connections between the cable and the demonstration board, power the board off and on, and try downloading again. Also, ensure that the bitstream is targeted for an XC4003E device.

If the Hardware Debugger informs you in a message that the current design does not include the READBACK block connected, check your schematic to ensure that the DEBUG_CKT symbol is connected as shown in the "DEBUG_CKT Symbol Connections" figure.

Note: The serial download cable has limited functionality when used with XC4000 family parts and may report that DONE went High even if you do not press the PROG button as in step 6. If this occurs, the part is not re-configured. Download the bitstream again, this time pressing the PROG button prior to configuration. Cycling the power off and on before starting the download has the same effect.

If you chose the Download and Verify command, the software initiates a design verification after downloading. The output of the design verification is displayed in a message box.

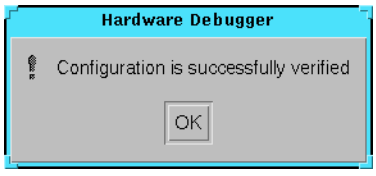

Figure 9-11 Pop-up Message Box

Testing the Design

After configuring the XC4003E with the WATCH design the functionality can be manually tested and observed on the LED Displays. The WATCH design implements the functionality of a Stopwatch.

The stopwatch counts up to 1 minute and starts over displaying tens of seconds, seconds, and tenths of a second. The stopwatch waits for a start command. Counts until a Stop command. Holds the value that it was stopped at, and can either start again from there or be reset.

The FPGA demonstration board has a row of eight rocker switches (SW3) and two buttons (RESET and SPARE) that provide input to the design.

Table 9-5 Schematic Labels vs. Demo Board Switches

WATCH Design Signal	Demo Board Switch/Button
CLK_SELECT	SW3-8
RESET	SW3-7
GRST	RESET
STRTSTOP	SPARE

In the WATCH design, the SW3-7 selects the clock source, SW3-8 is a synchronous state machine reset, the RESET button is a global asynchronous reset, and the SPARE button starts and stops the stopwatch.

To operate the stopwatch function:

1. Open switches SW3-7 and SW3-8.

   This selects the internal oscillator as the system clock and deasserts the state machine synchronous reset.

2. Press the RESET button to reset the stopwatch to 00.

3. Press the SPARE button to start the count.

   The tenths of a second are displayed on the lower row of 8 LEDs starting from D16 to D9. Where D16 represents 0.2 and D9 represents 0.9 seconds.

The seconds are displayed on the right hand side seven segment LED display U8.

The tens of seconds are displayed on the middle seven segment LED display U7.

4. Press the SPARE button again to stop the count.

   The LED displays hold the value at which the count was stopped at.

5. Press either the SPARE button again to resume count or the RESET button to reset the count back to 00.

Synchronous Debugging

Debugging offers a means for capturing the internal CLB output states and displaying them in waveforms like a simulator. The Hardware Debugger offers two types of debugging modes: Synchronous and Asynchronous.

This section describes the synchronous debugging mode. In the synchronous debugging mode, the Hardware Debugger gives the user control over the system clock allowing for a specified number of clocks between snapshots of the internal FPGA states. For a description on triggering snapshots asynchronously see the “Asynchronous Debugging” section.

Before you can begin debugging your design you must make sure that the cable is properly connected for synchronous debugging. This was not needed for the downloading and verifying section so it was skipped.

1. Connect the CLKO to P13 of the XC4003E.
2. Close switch SW3-7. This selects the external clock connection at P13.

To debug your design, you must setup the debugging mode, set the Trigger type, set the clock type, and include signals and signal groups in your display list.

Setting up the Synchronous Debugging Mode

To set the debugging mode, follow these steps.

1. Select the **Debug** → **Synchronous Mode** or click the following toolbar button.

2. Select **View**→ **Control Panel** to activate the Debug Control Panel.

   The appropriate options are enabled in the Debug Control Panel as shown in the "Debug Control Panel" figure.

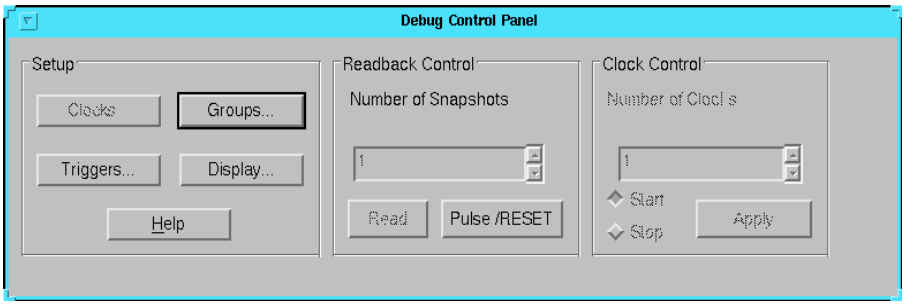

Figure 9-12 Debug Control Panel

3. In the Debug Control Panel, click the Clocks button to display the CLKO Clock Settings dialogue box.

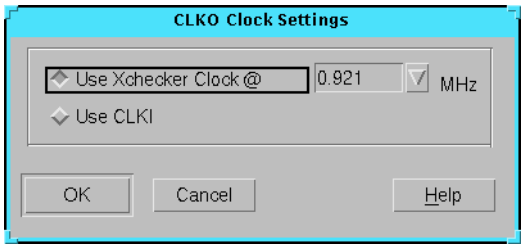

Figure 9-13 Clock Settings Dialogue Box

The Use Xchecker Clock option should already be set. If not then select it. The default clock speed setting of 0.921 MHz will work just fine for this example.

4. In the Debug Control Panel, click the **Triggers** button to display the Synchronous Trigger Settings dialog box.

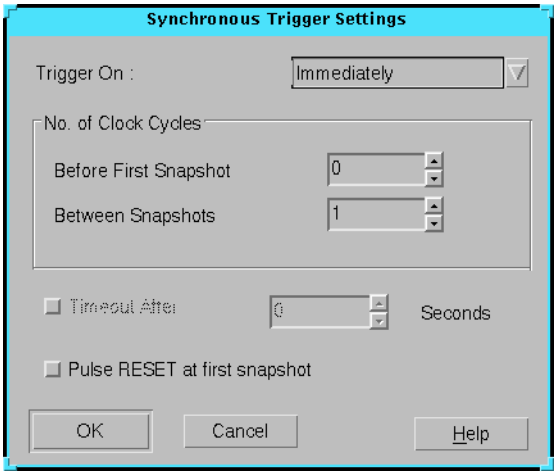

Figure 9-14 Trigger Settings Dialogue Box

5. Select **Immediately** from the Trigger On pull-down menu.
6. Click **OK**.

Specifying Signal Groups

You now need to specify which signals to view. Follow these steps to add signals to the list of signals to display.

The WATCH design has two internal buses that represent the value of the seconds counter prior the HEX2LED conversion: ONES[3:0] and TENS[3:0]. These signals are broken up into there individual bits during the implementation phase.

In the Hardware Debugger they can be recombined into a bus format for a more convenient display.

1. In the Debug Control Panel, click the **Groups** button to display the Signal Groups dialog box from which you can group signals into a bus for easy viewing.

   The Signal Groups dialog box appears as shown in the “Signal Groups Dialog Box” figure.

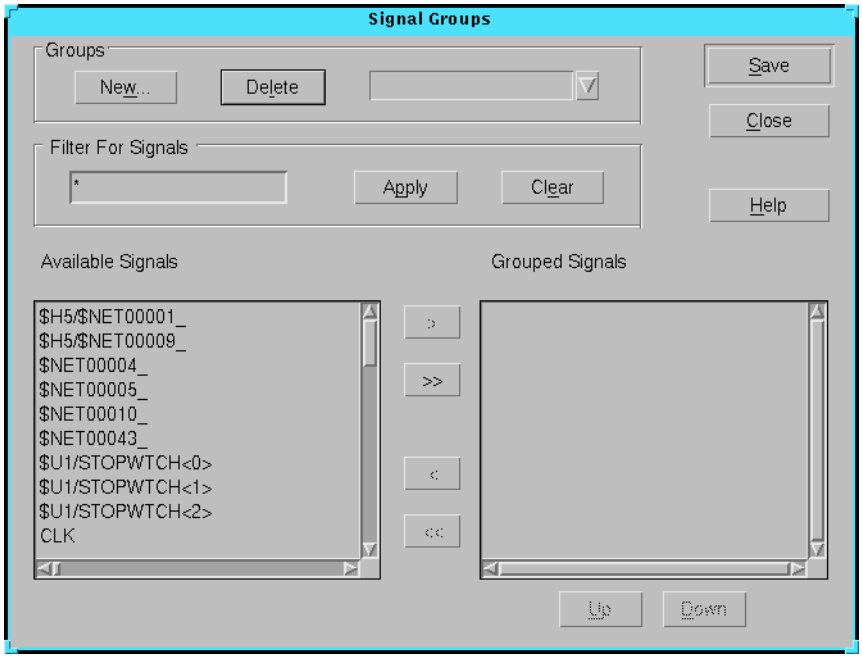

Figure 9-15 Signal Groups Dialog Box

2. To create a new group, click **New** in the Groups group box. The Group Name box appears, as shown in the “Group Name Dialog Box” figure.

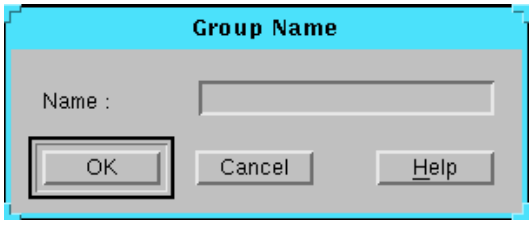

Figure 9-16 Group Name Dialog Box

3. Type the name **ones** in the Group Name dialog box and click on OK. The new group name appears in the Groups field of the Signal Groups dialog box.

4. In the Available Signals field scroll until you see the signal names ONES<0> thru ONES<3>. Select these signals and move them into the Grouped Signals field with the “>” button.

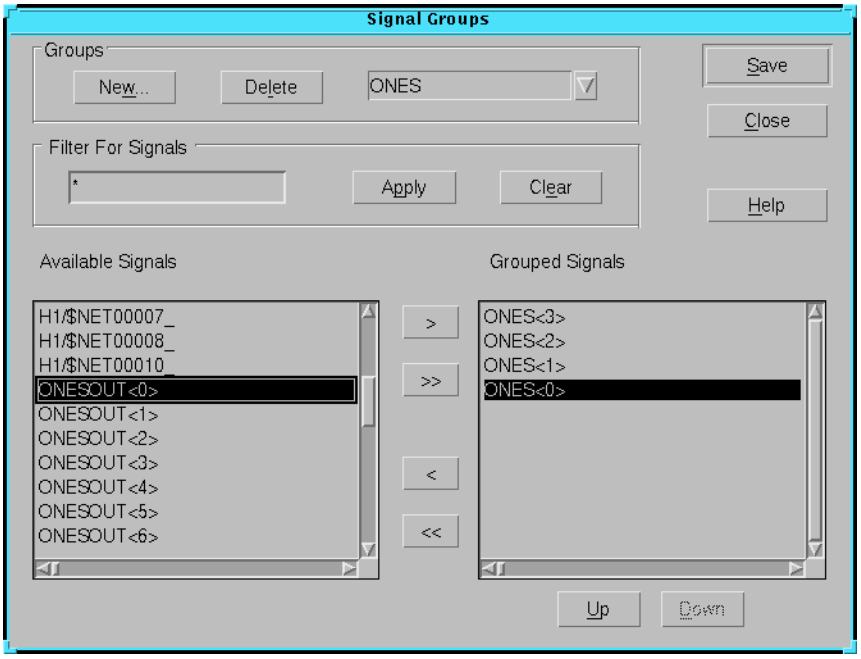

Figure 9-17 Signal Groups Dialogue Box.

Note: You can globally define the signals to display in the selection list-box by typing the first characters of the signals followed by a wildcard character (*) in the Filter For Signals box and clicking **Apply**

The MSB (ONES<3>) needs to be at the top of the list with the rest decending sequentially. Select the signals as needed and use the Up and Down buttons to adjust their order in the list.

5. When you are done specifying the group click on Save.
6. Make another group for the TENS.
7. Click **Close** when you are done adding groups.

Adding Signal Groups to Your Display List

In this section, you use the Display Signals dialog box to select the signals to view and debug. To add signals and the groups you just defined to your display list, follow these steps.

1. In the Debug Control Panel, click **`Display`** to invoke the Display Signals dialog box.
2. Use the Display Signals dialog box, shown in the "Display Signals Dialog Box" figure, to include the Signals TENTHSOUT<0> thru TENTHSOUT<9> in the Displayed Signals field.

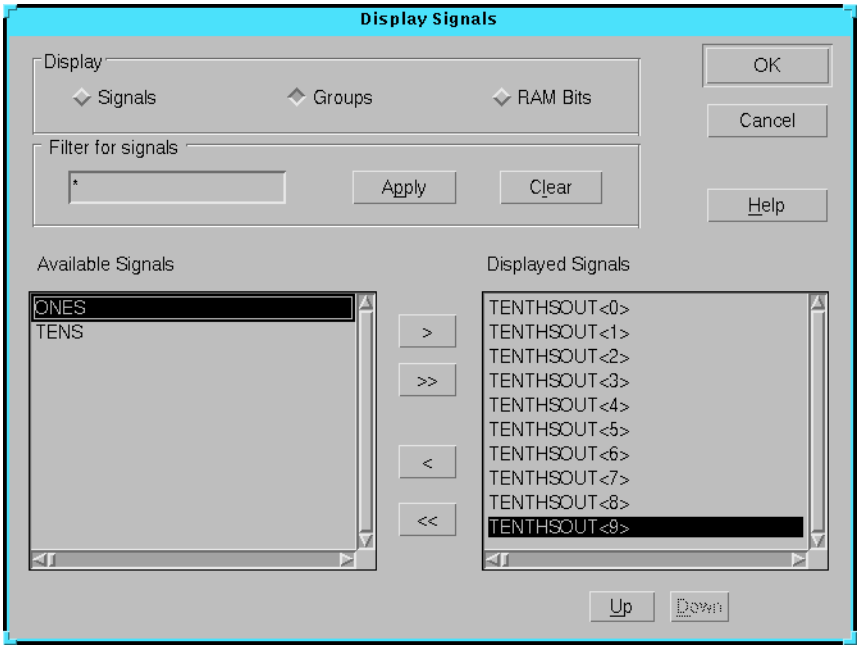

Figure 9-18 Display Signals dialogue box.

3. Click the **`Groups`** radio button in the Display group box to show the available signal groups that you just defined.
4. Click the >> button to move the Available Signals to the Displayed Signals list.
5. Click on OK.

A new Waveform window appears with the selected signals for display.

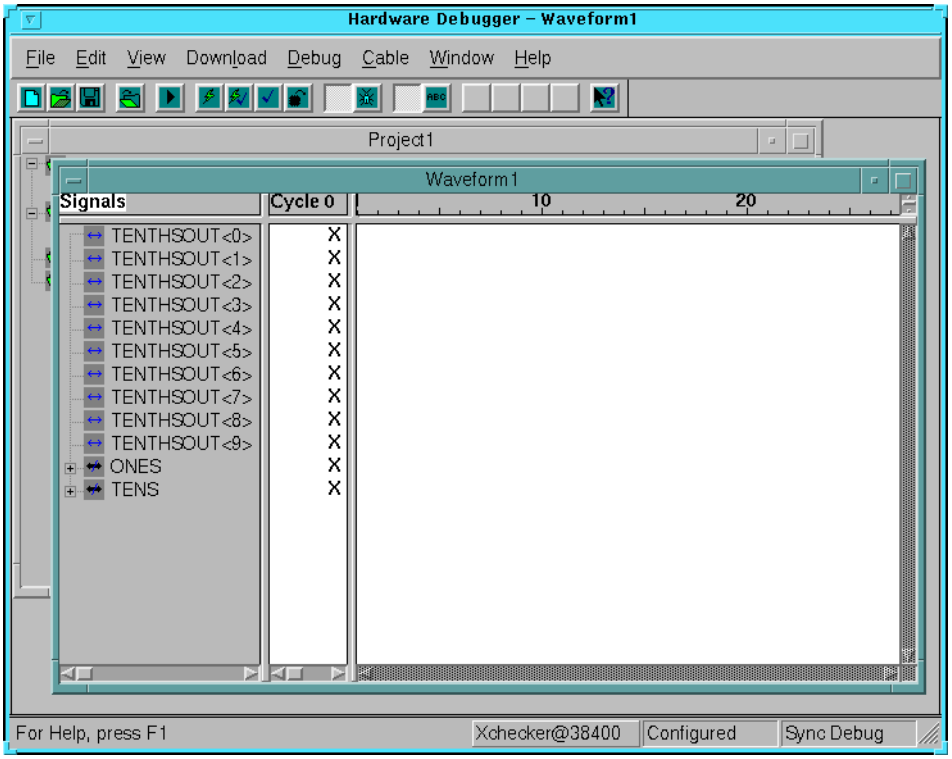

Figure 9-19 Waveform Window

Reading the Device States

Now we can begin reading snapshots of the internal device states. As we go we will adjust the number of clock cycles before each snapshot so that we can see the three buses independently transition appropriately.

1. First we must initialize the counters so that they can begin counting. In the Debug Control Panel under Readback Control click on the Pulse /RESET button.

2. In the Debug Control Panel set the Number of Clocks to 1. Hold down the SPARE button on the demonstration board and click on the Apply button once.
3. In the Readback Control of the Debug Control Panel set the Number of Snapshots to 10.
4. Click on the Read button.

   The Hardware Debugger will now take ten snapshots of the selected signal states incrementing the clock once between each snapshot.

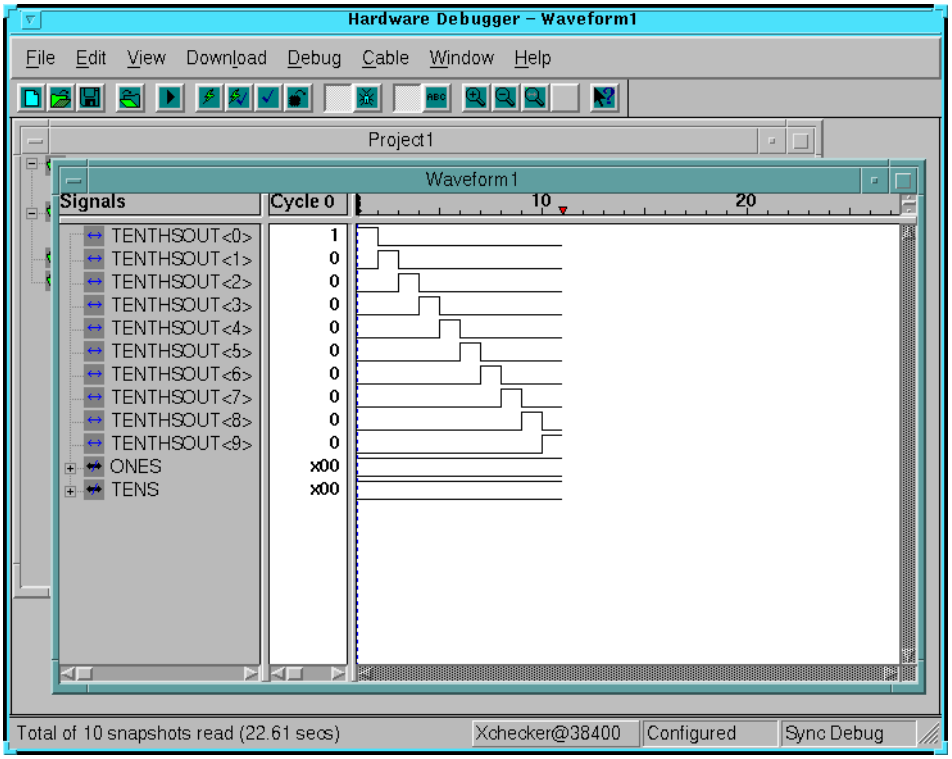

Figure 9-20 Waveform Window

5. In the Synchronous Trigger Settings increase the No. of Clock Cycles Before First Snapshot and Between Snapshots both to 10.

6. In the Readback Control of the Debug Control Panel reduce the Number of snapshots to 9.
7. Click on the Read button.

   We now see the ONES bus cycling though it's range of values in the waveform window.

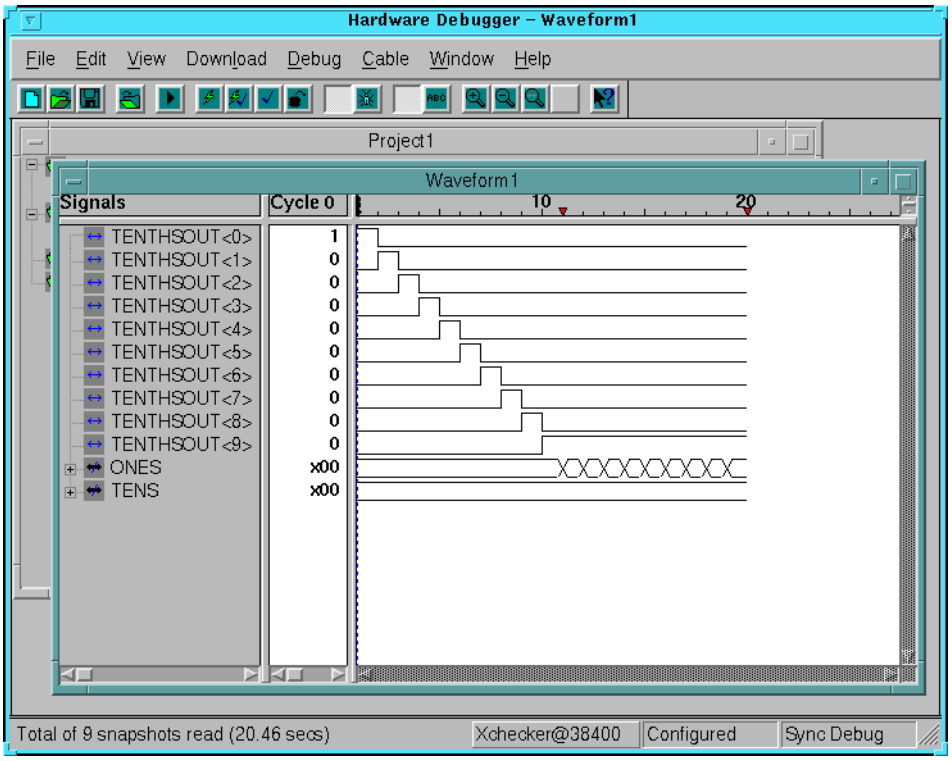

Figure 9-21 Waveform Window

8. In the Synchronous Trigger Settings decrease the No. of Clock Cycles Before First Snapshot to 0, and between Snapshots to 1.
9. Decrease the Number of Snapshots in the Readback Control to 1.
10. Increase the Number of Clocks in the Clock Control to 100. You can explicitly type in the desired number instead of scrolling for it.

11. Click on Apply.
12. Click on Read.

    The TENS bus now transitions. Repeat this several times to observe the full values range of the TENS bus.

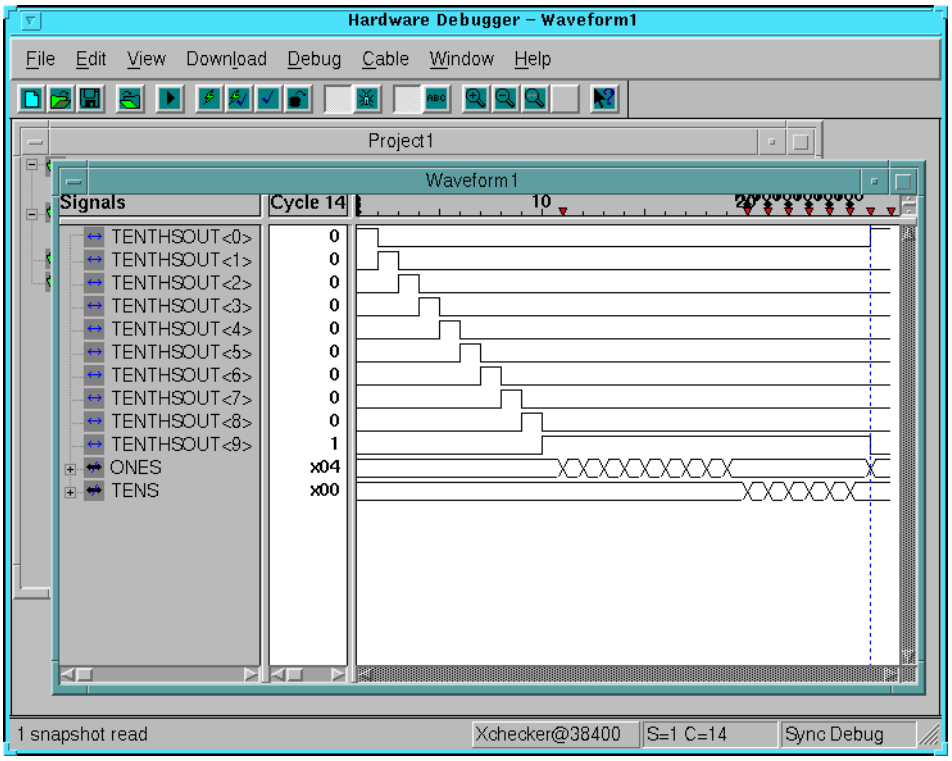

Figure 9-22 Waveform Window

The logical value of the displayed signals are shown in numerical form in the Cycle column. Click on the portion of the waveform that you would like to see the numerical listing for and the Cycle column automatically updates itself.

Note: To start fresh with the same waveform click the Right Mouse button once. A pop-up menu appears. Select Clear All Waveforms. All unsaved data read will be lost.

Changing the Signals Groups Radix

You may choose which radix you prefer you signals and groups be displayed in. The groups should have defaulted to HEX. To change to binary follow these steps.

1. In the Waveform window, click on the TENS groups.
2. Select **`View`** → **`Group Radix`** → **`Binary`**.

   You may need to expand the Cycle column in order to view the value..

Saving and Closing the Waveform Window

When you are down with a debugging session before exiting the waveform window you can save it for future reference.

1. Select **`File`**→ **`Close`**
2. Click Yes on the follwing pop-up box.

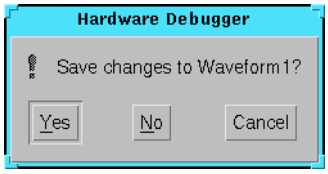

Figure 9-23 Pop-up Dialogue Box

3. Select a name for the file and click on Save.

Asynchronous Debugging

In the previous section we used synchronous debugging to verify the operation of the counters in the WATCH design. When the System clock connection for an application cannot be broken or otherwise interupted, as we did in the WATCH design with the Clock multi-plexing, Asynchronous Debugging may be used in absence of controlling the clock with the XChecker cable.

This section describes how to perform Asynchronous Debugging by setting up an external trigger to control snapshot timing.

Objective for this Section

Our objective is to determine the encoding method used for the internal stopwatch state machine in the WATCH design. Since this tutorial may be used with multiple design entry tutorials, including both schematic capture and HDl compilers, encoding schemes may very.

The stopwatch state machine has six states shown in the “Stopwatch State Table”.

Table 9-6 Stopwatch State Table

STATE	Description	Binary Code
CLEAR	Power-on initialization.	
ZERO	Reset counters to 0.	
START	Begin counting.	
COUNT	Keep counting.	
STOP	Stop counting.	
STOPPED	Hold count value.	

The Binary Code has been left blank for you to fill in.

Setting up the Demonstration Board

Before getting started we must return the WATCH design to using the internal oscillator by, and change some cable connections.

1. Open SW3-7.
2. Connect the TRIG flying lead to P14.

   Pin 14 of the FPGA conveniently provides an output of the internal clock. We can use this as a trigger. Any signal may be used for triggering as long as it represents a transition that you are interested in capturing. In this excercise we want to capture the next state transition after changing an input logic level from the buttons and switches.

Setting up the Asynchronous Debugging Mode

To set the debugging mode, follow these steps.

1. Select the **Debug** → **Asynchronous Mode** or click the following toolbar button.

Note: The Clock Control section of the Debug Control Panel is now disabled.

2. Click on the Trigger button in the Debug Control Panel.

   Now the Asynchronous Trigger Settings dialogue box appears.

3. Select to Trigger On External. Click OK.

4. Make a new Signal Group for the STOPWTCH statemachine outputs. For a detailed description on making signal groups return to the “Specifying Signal Groups” section.

   The name of the signals should be $$/STOPWTCH<0> and so on for bits <1> and <2>. The $$/ represents some arbitrary hierarchical name. Since this statemachine exists below a macro in the design, randomly generated instance names for the macro may be placed by an HDL compiler into the total signal name.

   Do not be concerned about this. The STOPPWTCH name should be unique; therefore select the closest match from the Available Signal list.

5. Add the stopwtch signal group that you just made along with the RST_INT signal.

6. Set the Number of Snapshots in the Readback Control to 2. Capturing two readbacks per state transition will tell us if we’ve captured an erroneous value caused by a timing glitch.

Capturing the State Machine

Follow these steps to capture and display the six states of the stopwatch state machine.

1. Press the RESET button.
2. Click on Read in the Readback control.
3. Click on the waveform itself inside the waveform window and note the value. This state is the CLEAR state.
4. Press and HOLD the SPARE button. While still holding the SPARE button down, click on Read.
5. Click on the new segment of the waveform and note the value. This is the START state.
6. Release the SPARE button and click on Read again. Note the value of the new section of the waveform. This is the COUNT state.
7. Again press and hold the SPARE button. Click Read while holding down the SPARE button. This is the STOP state.
8. Release the SPARE button and Read again. This is the STOPPED state.
9. Close switch SW3-8 and Read again. This is the ZERO state.

Now you have the encoding scheme for the internal state machine. Your waveform should look something like that shown in the “Asynchronous Debugging Waveform” figure.

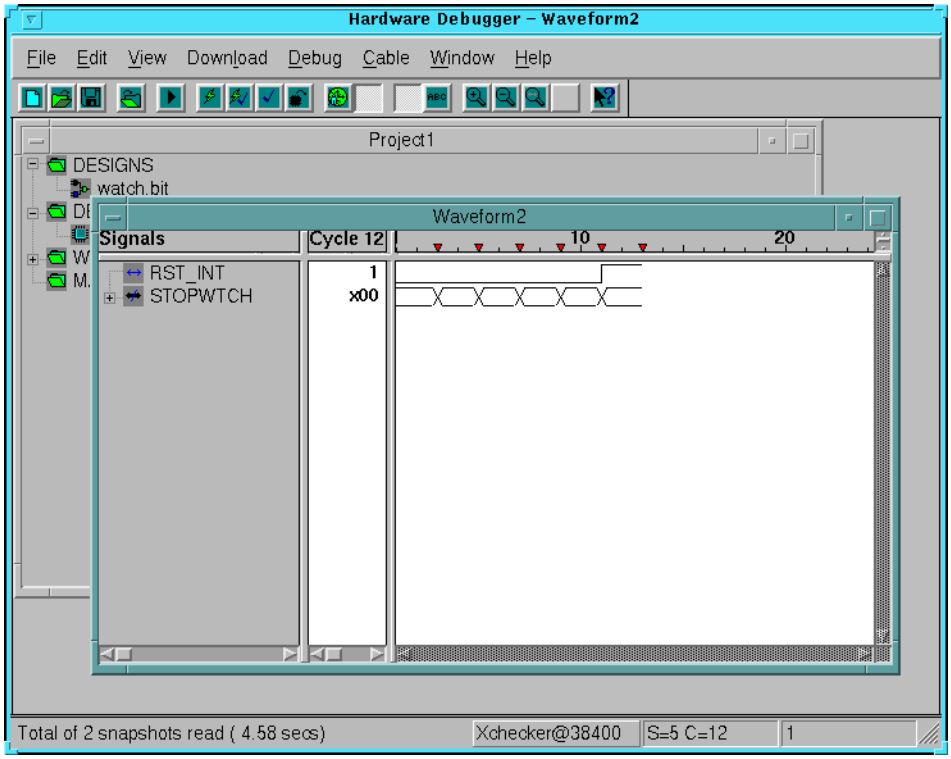

Figure 9-24 Asynchronous Debugging Waveform

For more information on using the Hardware Debugger refer to the Hardware Debugger Reference / User Guide Index.