T320C2700B0 Customizable Digital Signal Processor (cDSP™) Modeling User's Guide

Literature Number: SPRU260A
February 1999

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Preface

Read This First

About This Manual

This manual introduces the T320C2700B0 ('C2700B0) customizable DSP (cDSP™) design kit components and describes how to use the 'C2700B0 logic simulation models on various electronic design automation (EDA) platforms.

This guide assumes you have gone through the Mentor Graphics™, Verilog™, and ModelSim™ training and have a working knowledge of the Mentor Graphics QuickSIM-II™, QuickHDL™, and ModelSim Verilog Model (Windows NT™) logic simulators, the Cadence™ Verilog-XL™ simulator, the TI design support software (TIDSS™) environment, and related flows. For details of the TIDSS flow and environment, see the *Submicron ASIC Products Design Software Manual* and the *Submicron ASIC Products TI Design Support Software Release 4.2 for Mentor 8.5* release notes listed under *Related Documentation From Texas Instruments*.

This design manual is part of a complete set of documentation for the cDSP products. It can be used as a stand-alone document or in conjunction with related TI documents (see *Related Documentation From Texas Instruments*).

Notational Conventions

This document uses the following conventions:

- Program listings, program examples, and interactive displays are shown in a `special typeface` that is similar to that of a typewriter. Examples use a **`bold version`** of the special typeface for emphasis. Interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  The following is a sample program listing that places emphasis on .even:

```
0011  0005  0001          .field    1, 2
0012  0005  0003          .field    3, 4
0013  0005  0006          .field    6, 3
0014  0006                .even
```

The following is an example of a system prompt and a command you might enter:

```
C:  csr –a /user/ti/simuboard/utilities
```

- Any string within angle brackets is considered to be a variable. In syntax descriptions, the variable is written in a typeface similar to that of the text. The following is an example of a variable syntax:

  <file name> Path name of a UNIX™ file
  <signal> Name of a signal

- In syntax descriptions, square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you must specify the information within the brackets; you must not enter the brackets themselves. The following is an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. The following is an example of a list:

  ```
  { * | *+ | *– }
  ```

  This provides three choices: *, *+, or *–.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

Information About Cautions

This book contains cautions.

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

The information in a caution is provided for your protection. Please read each caution carefully.

Related Documentation From Texas Instruments

The following books describe the T320C2700B0 device and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

Submicron ASIC Products ASIC Compiler Environment (ACE™) User's Guide (literature number SRGU003) describes the features and capabilities of the ASIC Compiler Environment (ACE) and provides view descriptions of the compiler functions available in ACE. It also contains installation information for network administrators and execution information for ACE users.

Submicron ASIC Products Design Software Manual (literature number SRGA003) provides a design flow overview of the tools and processes of the 0.8-μm gate array series and describes in depth each particular software tool.

Submicron ASIC Products TI Design Support Software Release 4.2 for Mentor™ 8.5 (literature number SRUA018) describes the release notes for HP 9000™/HP 700™ series workstations with HP-UX™ 9.0x and 10.x, Sun™ SPARCstation™ 10/20 platforms, with SunOS™ 4.1.3 and Solaris™ 2.4 and 2.5.a, and Sun ULTRAsparc™ 1 platform with Solaris 2.5.1.

TMX320C2700-E3 Digital Signal Processor In-Circuit Emulation Device (literature number SPRS068) data sheet contains the pinout, block diagram, component descriptions, timing information, electrical specifications, and mechanical package for the device.

TMS320C2700B0 Customizable Digital Signal Processor (cDSP) Core (TSC6000 ASIC Libraries) (literature number SPRS069) data sheet contains the signal descriptions, functional block diagram, an example of system using the core, component descriptions, information on memory mapping, and clocking data for the core.

T320C2700B0 Customizable DSP (cDSP) System Configuration Design Guide (literature number SPRU272) describes the system configurations, signal connections, and electrical requirements needed to design with the T320C2700B0 core. It also contains information on test requirements and considerations to enable testing of the T320C2700B0 cDSP device.

TMS320C27xx DSP CPU and Instruction Set Reference Guide (literature number SPRU220) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C27xx 16-bit fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320C27xx Linker User's Guide (literature number SPRU225) describes the linker for the TMS320C27xx device. The linker configures system memory and combines partially linked files, output modules, object files, linker command files, archive libraries, and runtime-support libraries into a single output module.

Trademarks

If You Need Assistance . . .

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	http://www.ti.com/dsps
320 Hotline On-line™	http://www.ti.com/sc/docs/dsps/support.htm

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(281) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
DSP Hotline	(281) 274-2320	Fax: (281) 274-2324	Email: dsph@ti.com
DSP Modem BBS	(281) 274-2323		

DSP Internet BBS via anonymous ftp to ftp://ftp.ti.com/pub/tms320bbs

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:

Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32	Email: epic@ti.com
Deutsch	+49 8161 80 33 11 or +33 1 30 70 11 68		
English	+33 1 30 70 11 65		
Francais	+33 1 30 70 11 64		
Italiano	+33 1 30 70 11 67		
EPIC Modem BBS	+33 1 30 70 11 99		
European Factory Repair	+33 4 93 22 25 40		
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10	

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200
Hong Kong DSP Hotline	+852 2 956 7268	Fax: +852 2 956 1002
Korea DSP Hotline	+82 2 551 2804	Fax: +82 2 551 2828
Korea DSP Modem BBS	+82 2 551 2914	
Singapore DSP Hotline		Fax: +65 390 7179
Taiwan DSP Hotline	+886 2 377 1450	Fax: +886 2 377 2718
Taiwan DSP Modem BBS	+886 2 376 2592	

Taiwan DSP Internet BBS via anonymous ftp to ftp://dsp.ee.tit.edu.tw/pub/TI/

❑ Japan

Product Information Center	+0120-81-0026 (in Japan)	Fax: +0120-81-0036 (in Japan)
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259
DSP Hotline	+03-3769-8735 or (INTL) 813-3769-8735	Fax: +03-3457-7071 or (INTL) 813-3457-7071
DSP BBS via Nifty-Serve	Type “Go TIASP”	

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail: Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: comments@books.sc.ti.com

Note: When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Contents

Chapter 1

The Design Kit

The ’C2700B0 cDSP design kit allows you access to cDSP simulation models so you can start the design process. This chapter briefly describes the key features of the design kit.

Topic **Page**

1.1 Introduction

The T320C2700B0 cDSP design kit enables you to build your system by providing you with various components such as simulation models, memory wrappers, detector rules, the T320C2700B0 cell generic object-oriented database (GOOD), the conversion utility COFF2SIM, and the DSPnetGEN golden netlist files (GNFs). These components assist in carrying your design through the Texas Instruments design support software (TIDSS) cDSP flows seamlessly for a given application-specific integrated circuit (ASIC) technology. This manual discusses the various components of the design kit and elaborates on the T320C2700B0 core simulation models, their features, and their usage.

For more information about these components, see the *T320C2700B0 Customizable DSP (cDSP™) System Configuration Design Guide*. For more information about the TIDSS cDSP flows, see the *Submicron ASIC Products Design Software Manual*.

1.2 T320C2700B0 Simulation Models

The design kit contains the simulation models for the ’C2700B0 core that run on the following electronic design automation (EDA) and hardware operating system (OS) platforms:

- Mentor QuickSim-II (Mentor B4 series) on Solaris and HP10X
- Cadence Verilog-XL (Cadence 9702) on Solaris and HP10X
- Mentor QuickHDL (Mentor B4 series) on Solaris and HP10X
- ModelSim Verilog Model (ModelSim PE 4.7i) on WindowsNT

Each of these simulation models have some features that are common to one another and some that are platform specific. All these models are based on the C programming language and are wrapped with specific programming language interface (PLI) calls. All models have complete core functionality including test (JTAG) and emulation features. All models also support functional and timing simulations. For details of model features, see the rest of the manual.

1.3 T320C2700B0 Cell GOOD

The cell generic object-oriented database (GOOD) contains all the characterized paths and constraints of the ’C2700B0 core along with the input slews, output loads, and clock periods. It also contains the interface pins specification. The paths for the cell GOOD are characterized for the TSC6000 technology. All the timing information for the simulation models is captured from the cell GOOD.

1.4 Memory Wrappers and External Interface

The memory wrappers enable the ’C2700B0 core to be connected to various ASIC compiler environment (ACE) memories. The different kinds of memory wrappers include the SARAM MK, MR, and MV wrappers that enable the ACE MK, MR, and MV memories to be connected to the core. Also included are the ROM wrappers that connect the ACE ROMs to the core. In addition to memory wrappers, the design kit contains the ’C2700B0 external interface (XINTF) module that allows for external memory and peripheral connections.

The XINTF and all of the memory wrappers are available as very high speed integrated circuit (VHSIC) hardware description language (VHDL) register language (RTL). They are also available as EDA-specific netlists such as Cadence Verilog and the Mentor Graphics electronic design data model (EDDM). For more information about memory wrappers and XINTF specifications, see the *T320C2700B0 Customizable DSP (cDSP™) System Configuration Design Guide*.

1.5 COFF2SIM

The COFF2SIM is a program that converts the ’C2700B0 COFF to the EDA-specific programmation file formats such as QuickSIM-II, Verilog, and QuickHDL. These programmation files are loaded into the ACE memory compilers.

To use COFF2SIM, enter the following command at the system prompt:

```
% coff2sim –file <config file> –format
 <Verilog|VHDL|IKOS|Quicksim> –processor hex27
```

The required parameters for the command line are as follows:

–file <...> Specify the name of the configuration file
–format <...> Specify the format of the output file

The optional parameters for the command line are as follows:

–log [file] Log message to file (default is screen)
–nohex Generate hex500 command file but do not run it
–help Print the command line option and quit

For more information about the COFF2SIM inputs, see the T320C2700B0 README file and the design kit installation directory.

1.6 Synthesis Models

Synthesis models for the ’C2700B0 core are the technology-specific black-box models containing path delays, constraints, and load and slew information of the pins. Synthesis models, however, do not have any functionality. Their purpose is to enable design netlist translations, as well as netlist optimizations around the ’C2700B0 core. Currently, only the Synopsys synthesis models are supported.

1.7 Detector Rules

In addition to the technology-specific rules supported by the ASIC design kits, the T320C2700B0 design kit contains the ’C2700B0 core detector rules. Detector rules ensure that your netlist is in compliance with the rules that are needed for a proper design. These are as follows:

- Default tie-offs for interrupts
- Clockout connectivity checks through the clock-tree synthesis (CTS) buffers
- Unused input tie off checks
- JTAG connectivity checks
- Memory interface connectivity checks

For details on supported core detector rules, see the detector rule specifications in the design kit installation directory.

1.8 DSPnetGEN GNFs

The design kit contains golden netlist files (GNFs) for the core, memory wrappers, XINTF, and some ACE memories. The GNF files assist in generating the system design netlist through use of a tool called DSPnetGEN. Some of the features of these GNFs include configurability and error checking. For the details of DSPnetGEN GNFs and the procedure for generating a high-level VHDL netlist, see the *T320C2700B0 Customizable DSP (cDSP™) System Configuration Design Guide*.

1.9 ATPG Models

The core automatic test pattern generation (ATPG) models are black-box models. The ATPG models enable the generation of the test vector for the design. Currently, they are available for the Mentor Graphics Flextest and Fastscan tools.

1.10 T320C2700B0 Documentation

In addition to this manual, the design kit contains other documentation, which includes the *T320C2700B0 Customizable Digital Signal Processor (cDSP™) Core (TSC6000 ASIC Libraries)* datasheet and the *T320C2700B0 Customizable DSP (cDSP™) System Configuration Design Guide*. The datasheet contains the electrical and timing specifications, signal descriptions, and information on memory mapping and clocking data for the 'C2700B0 core. The design guide describes the system configurations, signal connections, and test requirements to enable you to design with the 'C2700B0 core.

Chapter 2

T320C2700B0 cDSP Simulation Models

This chapter describes the T320C2700B0 cDSP simulation models and their behavior.

For detailed signal descriptions and behavior, see the *T320C2700B0 Customizable Digital Signal Processor (cDSP™) Core (TSC6000 ASIC Libraries)* datasheet.

Topic **Page**

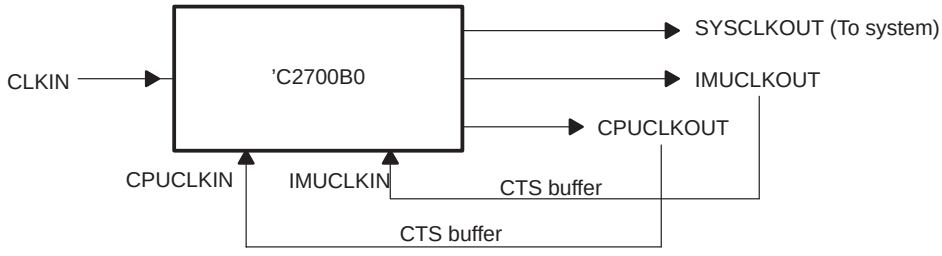
2.1 Model Behavior

The following sections describe the functions of the design that determine whether the ’C2700B0 cDSP models behave properly.

2.1.1 Clock Connections

An understanding of the clocking scheme is essential when creating a design using the ’C2700B0 cDSP model. Figure 2–1 shows the required clock connections. The ’C2700B0 model has a primary input clock, CLKIN, and three output clocks, CPUCLKOUT, IMUCLKOUT, and SYSCLKOUT. CPUCLKOUT must be connected to CPUCLKIN through a clock-tree synthesis (CTS) buffer. IMUCLKOUT must be connected to IMUCLKIN through a CTS buffer. These connections are essential for the model to operate properly. SYSCLKOUT is the system clock and connects to other parts of the system. For a description of each clock signal, see the *T320C2700B0 Customizable Digital Signal Processor (cDSP™) Core (TSC6000 ASIC Libraries)* datasheet.

Figure 2–1. T320C2700B0 Clock Connections



2.1.2 Reset Sequence/Model Initializations

The input signal RSN is an active low pin. RSN must be active for at least 1 CLKIN period for the ’C2700B0 core to have a proper reset. The output signal SYSRSN indicates whether the core is still in the reset state or has completed reset. The core is in the reset state when SYSRSN is low. The core has completed reset when SYSRSN is high.

For SYSRSN timings, see the *T320C2700B0 Customizable Digital Signal Processor (cDSP™) Core (TSC6000 ASIC Libraries)* datasheet.

Table 2–1 shows the logic values needed on the input signals for proper reset and core initializations.

Table 2–1. Input Signals and their Logic Values Needed for Proper Reset and Core Initializations

Input Signals	Logic Values
TMS	0
ET0I	1
ET1I	1
SLAVEIN	0
TRST	0
TDI	0
TCK	0
BREAKHI	0
BREAKLO	0
AEVTQUAL	0
ANASTOP	0
DEVTQUAL	0
MONPRIV	0
CTOOLSACK	0
INTN[13:0]	3FFF
NMIN	1
RTOSINTN	1
TRACEHI	0
TRACELO	0
PERISCOUT	0
PERISCPATH	0
EXTCNT0	0
EXTCNT1	0
EXTTRGR	0
ENPROT	0

The preceding setup ensures that the following occurs:

- The output signal SYSRSN goes high for the core to complete reset.
- The output clocks, SYSCLKOUT, IMUCLKOUT, and CPUCLKOUT, are generated properly.

All ready signals must be pulled high so that a proper reset occurs and the core is ready for program fetching and execution. Ready signals that must be pulled high are as follows:

- POREADY[5:0]
- DROREADY[5:0]
- DWOREADY[5:0]
- B0POREADY
- B1POREADY
- B0DROREADY
- B1DROREADY
- B0DWOREADY
- B1DWOREADY

2.1.3 State of the Signals During and After Reset

All memory-interface output signals are in an inactive state during reset. Also, during model initializations, some of the output address busses such as the PAB[31:0] are uninitialized. Therefore, you must not rely upon model output values during reset.

2.1.4 Violation of Constraints

The model's behavior is unpredictable after a constraint violation has occurred. You will know that a constraint violation has occurred if the model displays an error or a warning message in your system. The following is an example of a setup/hold violation:

```
ERROR: Stability time violated on PRDY to CPUCLKIN
The PRDY changed at time=300.01 ns and CPUCLKIN changed at
time=300 ns
The expected setup time: 0.02 and hold time: 0.03 ns
```

In the event of a constraint violation, discontinue the present simulation and resimulate the design after adjusting or correcting your design.

The constraints are checked for setup/hold and pulse duration violations. For details of constraint types and their values, see the *T320C2700B0 Customizable Digital Signal Processor (cDSP™) Core (TSC6000 ASIC Libraries)* datasheet.

All of the simulation models support skew checks—that is, maximum separation time—between CPUCLKIN and IMUCLKIN. Hence, warning messages are generated whenever CPUCLKIN and IMUCLKIN are separated by more than 50 ps, the maximum skew limit for this technology.

The following is an example of a skew violation:

```
ERROR >> The skew between CPUCLKIN and IMUCLKIN is
         violated
      >> Max Expected 0.05 ns: Observed 0.12 ns
      >> The simulation results may not be correct;
         Correct the clock skews and rerun the
         simulations
```

In such cases, you must restart the simulations after adjusting the clock CTS buffers and make sure that the skew violations do not occur again.

2.2 Unknown Inputs

The input stimuli can be unknown during initializations due to propagation delays of the buffers or other logic driving these pins. During this time, the values of the core outputs may be unknown. In addition, if any of the inputs such as input data busses (DRDB and PRDB), ready signals (DRDY and PRDY), and input clocks (CLKIN) go unknown after processor reset has occurred, the behavior of the model may be unpredictable.

2.3 Register Display

You can obtain the register log information during simulation by setting the environment variable ANKOOR_REG_LOG as follows:

```
% setenv ANKOOR_REG_LOG reg.log
```

At the end of the simulation, the reg.log file displays the register values for all the clock cycles.

Table 2–2 shows registers that are traced in hex format in the log file.

Table 2–2. Registers Traced in the Log File in Hex Format

Register	Description
ACC	Accumulator
AR0	Auxiliary register 0
AR1	Auxiliary register 1
AR2	Auxiliary register 2
AR3	Auxiliary register 3
AR4	Auxiliary register 4
AR5	Auxiliary register 5
DP	Data page pointer
IC	Instruction counter
IER	Interrupt enable register
IFR	Interrupt flag register
IREGH	Instruction register (high)
IREGL	Instruction register (low)
PC	Program counter
PREG	Program register
RPTC	Repeat counter
SP	Stack pointer
ST0	Status register 0
ST1	Status register 1
TREG	Temporary register
XR6	Extended register 6
XR7	Extended register 7

During reset, the values of these registers are unknown. Therefore, these values are displayed as an x in the log file until the reset is complete.

2.4 Timing Simulations

In any TI cDSP ASIC flow, the timings for different parts of the design must meet the design performance requirements. Timing simulations are essential because they assist in verifying the timing requirements of a design.

All simulation models have placeholders for path delays, constraints, and pulse widths. Since the values for these are obtained through back-annotated data, you must do timing simulations with back-annotated data. This ensures that the models are technology dependent and therefore, timing accurate.

All simulation models support the three timing options—minimum, nominal, and maximum.

2.4.1 Timing Simulations in Mentor QuickSIM-II Simulation Environment

In the QuickSIM-II logic simulator, the type of simulation you choose to do is determined by the timing option –tim. The timing simulations in QuickSIM-II are invoked by using the –tim option. In this mode, the outputs appear after the appearance of the path delays specified in the Mentor Graphics standard delay format (MSDF) delay file. This mode also checks for and reports any input setup/hold constraint violations and pulse duration violations of the clock. The use of the –tim option in invoking timing simulations is demonstrated as follows:

```
% quicksim test –tim min –ts 0.01
```

This example shows that for simulation purposes, QuickSIM-II picks up minimum timing numbers that have a resolution of 0.01 ns. You can pick up maximum timing numbers by substituting the word min, in this example, with the word max. Similarly, you can pick up nominal timing numbers by substituting the word min, in this example, with the word typ.

For more information about the –tim option, see the Mentor Graphics QuickSIM-II Online Help.

Note:

The MSDF delay file is generated through the TIDSS flow. For details of the TIDSS flow, see the *Submicron ASIC Products Design Software Manual*.

2.4.2 Timing Simulations in Cadence Verilog-XL Simulation Environment

In this environment, timing simulations can be invoked by using back-annotated data that is in a standard delay format (SDF). The following example demonstrates simulating a design by picking up maximum delays from SDF:

```
% verilog test.v +define+maxdelays
```

In this example, if you want to pick up minimum delays from SDF, substitute the word maxdelays with mindelays. If you want to pick up nominal delays, substitute the word maxdelays with nomdelays.

2.4.3 Timing Simulations in Mentor QuickHDL Simulation Environment

To select the timing models in the QuickHDL simulation environment, you must select the configuration design unit conf_t320c2700b0_vital by entering the following command at the system prompt:

```
% qhsim conf_t320c2700b0_vital
```

2.4.4 Timing Simulations in ModelSim Verilog Model (Windows NT) Simulation Environment

In this environment, timing simulations can be invoked by using back-annotated data that is in a standard delay format (SDF). The following example demonstrates simulating a design by picking up maximum delays from SDF:

```
vsim test.v +define+maxdelays
```

In this example, if you want to pick up minimum delays from SDF, substitute the word maxdelays with mindelays. If you want to pick up nominal delays, substitute the word maxdelays with nomdelays.

2.5 Function-Only Simulations

Function-only simulations are used for verifying the functionality of your design. Typically, the function-only mode of simulation is used in the early phases of design when you need to confirm that correct connections have been made for proper functioning of the model. Function-only simulations do not account for the time it takes for data to transmit from one point to another in different parts of the design. Therefore, no timing checks are performed and no output path delays are used. Consequently, you cannot back-annotate timing data during the simulation session.

The function-only mode of simulation provides an improved performance in terms of speed because it is not dependent on a timing sequence that slows down data transmission. This is especially prevalent in the Mentor Graphics QuickSIM-II simulator because of its capacity to switch off the timing queue altogether.

A faster simulation serves a better purpose than a timing accurate one in cases where you want quick results rather than timing accurate ones—for instance, when you are verifying a design's functionality.

2.5.1 Function-Only Simulation in Mentor QuickSIM-II Simulation Environment

QuickSIM-II supports function-only simulation where it switches off the timing collar. This is demonstrated as follows:

```
% quicksim test
```

In this environment, all the outputs of the model have zero delays (or take 2 or 3 QuickSIM time steps).

The absence of the timing collar enables a fast, functionally correct simulation.

CAUTION

Do not use the function-only simulation mode if you want to back-annotate timing data during the simulation. Since there are no placeholders for timing paths and timing constraints, back-annotation is not possible.

2.5.2 Function-Only Simulation in Cadence Verilog-XL Simulation Environment

The ’C2700B0 Verilog-XL model has a directive, TI_functiononly, to turn off the Verilog timing section. By enabling this switch at the system prompt, the simulations are run with a resolution of 1 ns. To invoke the Verilog directive for function-only simulation, enter the following command at the system prompt:

```
% verilog T320C2700B0.v +define+TI_functiononly
```

2.5.3 Function-Only Simulation in Mentor QuickHDL Simulation Environment

The QuickHDL model has a configuration design unit conf_t320c2700b0_func that you must select to do function-only simulations in the Mentor QuickHDL environment. To enable function-only simulation, enter the following command at the system prompt:

```
% qhsim conf_t320c2700b0_func
```

2.5.4 Function-Only Simulation in ModelSim Verilog Model (Windows NT) Simulation Environment

The ’C2700B0 ModelSim Verilog Model has a directive, TI_functiononly, to turn off the Verilog timing section. By enabling this switch at the system prompt, the simulations are run with a resolution of 1 ns. To invoke the ModelSim Verilog Model (Windows NT) directive for function-only simulation, enter the following command at the system prompt:

```
verilog T320C2700B0.v +define+TI_functiononly
```

Appendix A

Mentor Graphics Model Installations

This appendix describes the procedure for installing the Mentor Graphics QuickSIM-II and the QuickHDL models in the design.

For QuickSIM-II commands and related information, see the Mentor Graphics QuickSIM-II Online Help. For QuickHDL commands and related information, see the Mentor Graphics QuickHDL Online Help.

Topic **Page**

A.1 Installation Directory Structure

This section describes how to uncompress the 'C2700B0 release and display the subdirectories that are necessary for running the simulations. Before you begin the installation process, you must download the installation file from the design kit website or the FTP site specified in the design kit release memo.

To install the Mentor Graphics QuickSIM-II and QuickHDL models on your system, follow these steps:

1) Change the directory to the location where the design kit needs to be installed by entering the following command at the system prompt:

   ```
   % cd <installationDir>
   ```

2) Uncompress the compressed release by entering the following command at the system prompt:

   ```
   % unzip T320C2700B0_mentor_designkit_<version>.zip
   ```

   This command creates a directory called T320C2700B0_mentor_designkit_<version>.

3) Go to this directory by entering the following command at the system prompt:

   ```
   % cd T320C2700B0_mentor_designkit_<version>
   ```

4) List the contents of this directory:

   ```
   % ls
   ```

   This command displays the README file and the integration subdirectory. The README file contains detailed components of the design kit.

5) Go to the integration subdirectory:

   ```
   % cd integration
   ```

6) List the contents of the integration subdirectory to display the custom_data subdirectory.

7) Go to the custom_data subdirectory:

```
% cd custom_data
```

8) List the contents of the custom_data subdirectory to display the following subdirectories:

```
custom_data/
           COFF2SIM/          : Contains utility to convert COFF to
                                programmation file
          T320C2700B0_2.01/    : Contains various components of the core
                               (see directory structure on the following
                                  page)
          cDSP_CROMW_1.0/     : ROM wrapper. contains the VHDL RTL,
                                VHDL netlist, and the EDDM netlists
          cDSP_MVSRAMW_1.0/    : SARAM wrapper for MV ACE memory contains
                                the VHDL RTL, VHDL netlist, and the
                                 EDDM netlists
          cDSP_P2XINTF_1.0/    : XINTF module contains the VHDL RTL, VHDL
                                netlist, and the EDDM netlists
          cDSP_SRAMW_2.0/     : SARAM wrapper for MK and MR ACE memories
                               contains the VHDL RTL, VHDL netlist, and
                                the EDDM netlists
```

The wrappers and the XINTF directories contain the VHDL RTL, the VHDL netlist, and the EDDM databases, which capture the functionality of the wrappers for simulation purposes.

The following is the directory structure for the T320C2700B0 core:

```
T320C2700B0_2.01/
           1.8v/
                 GOOD/          : Contains GOOD database
                 rules          : Contains Detector Rules
           hp700/
             atpgm8            : Contains ATPG black-box models
            synopsys           : Contains Synopsys black-box models
             quicksim           : QuickSim-II models
                lib/
                 1.8v/
                  sim/
                  cell_lib      : Contains top-level model
                  prim_lib      : Contains primitive parts
             quickhdl
                obj/
                 1.8v/
                               : Contains shared objects
                src/
                 1.8v/
                               : Contains the model VHDL
           sun5/
                                 : Same as hp700
         documentation          : Contains the datasheet & modeling guide
         example_qhdl            : Contains QuickHDL example
         example_qsim            : Contains QuickSim-II example
```

You are now ready to start the instantiation process.

A.2 QuickSIM-II Model Instantiations

Before you begin instantiating your model, do the following:

- ❑ Make sure that the MGC_HOME environment variable is set to point to the Mentor b4 installation path that is specific to your network.
- ❑ Set theMGC_LOCATION_MAP environment variable to a location map file path. To do so, find a sample file called *location_map* in the *custom_data/documentation/examples* directory (see Section A.1). The *location_map* file must contain the following variables:
  - $C27_LIBRARY: This variable points to the directory containing the top-level schematic and symbol for the T320C2700B0 model. The path for this directory is:

    <installationDIR>/custom_data/T320C2700B0_2.01/1.8v/hp700/quicksim/lib/1.8v/sim/cell_lib
  - $C27_PARTS: This variable points to the directory containing the constituent parts of the model. The path for this directory is:

    <installationDIR>/custom_data/T320C2700B0_2.01/1.8v/hp700/quicksim/lib/1.8v/sim/prim_lib

You must have these variables to get to the ’C2700B0 model and do proper simulations.

Note:

You must specify the correct installation directory (installationDIR) in the location map file in order to pick up the preceding directories.

Once you have completed the preliminaries, you can begin instantiating your model.

Instantiation is the process by which you put a model in your design for purposes of simulation. Instantiating the ’C2700B0 cDSP model allows you to place the model in your design so that you can start the simulation. The QuickSIM-II ’C2700B0 model is instantiated when you pick up the component $C27_LIBRARY/T320C2700B0 in the design architecture (DA). For details of how to pick up this component, see the Mentor Graphics Design Architecture Online Help.

A.3 Viewpoint Setting

Viewpoint setting enables you to have functional and timing simulations using the QuickSIM-II 'C2700B0 model. The 'C2700B0 model has been registered with generic timing numbers that are independent of any specific ASIC technology. Follow these steps to set up the viewpoint for the design:

1) Invoke the design viewpoint editor (DVE) by entering the following command at the system prompt:

```
% $MGC_HOME/bin/dve
```

2) Using the Mentor Graphics Open Design Viewpoint, open a viewpoint test called viewpt.
3) In the Open Design Viewpoint window, fill in the Component Name and the View Point Name fields as follows:

   Component Name: Enter the name of the design.
   View Point Name: Enter viewpt.

4) In the design configuration window, add the following items:

```
Parameter          Name:              technology
                   Value:             ti_cell
                   Property Type:     string

Primitive          Name:              model
                   Value:             technology
                   Property Type:     expression
```

   In this example, the generic model is registered with the label ti_cell. This label is used for picking up the correct timing values during simulation. The model does not pick up the correct timing values if you use another label.

5) Save the design viewpoint.

Note:

The 'C2700B0 cDSP model does not use built-in primitives.

A.4 QuickSIM-II Example Testbench

The QuickSIM-II example testbench allows you to see the instantiation process for the 'C2700 DSP core, the SARAM wrapper, the CROM wrapper, the MV SARAM wrapper, and the ACE memories. The memory mapping for this testbench is as follows:

Program Space	Data Space
B0 SARAM 1K × 16 @0x0	B0 SARAM 1K × 16 @0x400
——	B1 SARAM 1K × 16 @0x0
RAM_1 2K × 16 @1000 (MK)	RAM_1 2K × 16 @1000 (MK)
RAM_2 512K × 32 @2000h (MV)	RAM_2 512K × 32 @2000h (MV)
ROM_1 2K × 16 @3FF800	ROM_1 1K × 32 @3FF800

The program is preloaded into the ROM. The code is then downloaded into B0 and data constants moved to B1. The code is finally run from B0. The program has accesses to other random-access memories (RAMs).

To run the QuickSIM-II example testbench using the 'C2700B0 model, follow these steps:

1) Go to the example directory by entering the following command at the system prompt:

```
% cd <installationDIR>/T320C2700B0_mentor_
designkit_2.01/integration/custom_data/T320C2700B0_
2.01/documentation/example_qsim
```

2) Edit the location map file for the proper directory paths as defined in section A.2 on page A-5.
3) Invoke QuickSIM-II by entering the following command at the system prompt:

```
% MGC_HOME/quicksim \$C3FUNC/c3func/viewpt –tim min –ts
0.01
```

4) Trace the required outputs by entering the following command on the QuickSIM-II command line:

```
% dof trace
```

5) Apply the stimulus by entering the following command on the QuickSIM-II command line:

```
% dof stim
```

6) Run for the required number of cycles by entering the following command on the QuickSIM-II command line:

```
% run <total time>
```

A.5 QuickHDL Model Instantiations

Before you begin instantiating your model, do the following:

- ❑ Make sure that the MGC_HOME environment variable is set to point to the mentor b4 installation path that is specific to your network.
- ❑ Set the MGC_LOCATION_MAP environment variable to a location map file path. To do so, find a sample file called *location_map* in the *custom_data/documentation/examples* directory (see Section A.1). The *location_map* file contains the variable C27QHDL_BIN, which points to the valid QuickHDL binary file of the T320C2700B0 model. The path for this directory is:

  <installationDIR>/custom_data/T320C2700B0_2.01/sun5/quickhdl/lib/obj/3.3v.

Note:

You must specify the correct installation directory (installationDIR) in the location map file in order to pick up the preceding directories.

Once you have completed the preliminaries, you can begin instantiating your model.

The T320C2700B0 model is instantiated when you pick up the VHDL component T320C2700B0 from the installation directory. For details of how to pick up this component, see the Mentor Graphics QuickHDL Online Help.

The model is registered with generic timing numbers that are independent of any specific technology.

A.6 QuickHDL Example Testbench

The QuickHDL example testbench allows you to see the instantiation process for the 'C2700 DSP core, the SARAM wrapper, the CROM wrapper, the MV SARAM wrapper, and the ACE memories.The memory mapping for the QuickHDL example testbench is as follows:

Program Space	Data Space
B0 SARAM 1K × 16 @0x0	B0 SARAM 1K × 16 @400h
——	B1 SARAM 1K × 16 @0x0
RAM_1 2K SARAM @1000h	RAM_1 2K SARAM @1000h
RAM_2 1K × 16 @2000h (MV)	RAM_2 1K × 16 @2000h (MV)
CROM 2K × 16 @3FF800h	CROM 2K × 16 @3FF800h
XINTF	XINTF
Zone 0 @8000h (mapped by entering into the configuration registers)	Zone 0 @8000h (mapped by entering into the configuration registers)

To run the QuickHDL example testbench using the 'C2700B0 model, follow these steps:

1) Go to the example directory by entering the following command at the system prompt:

```
%cd <installationDIR>/T320C2700B0_mentor_designkit_2.01
/integration/custom_data/T320C2700B0_2.01/documentation
/example_qhdl
```

2) Edit the example.build script for the proper directory path as specified in section A.5 on page A-9.
3) Run the example.build script by entering the following command at the system prompt:

```
% example.build
```

This script compiles all the relevant source files for the T320C2700B0 core, the wrapper netlists, and the ACE memory VHDL files used in the example. The compiled source files are placed in the quickhdl_work library.

4) Edit the location_map to give the correct path for the variable C27QHDL_BIN by entering the following command at the system prompt:

```
% setenv MGC_LOCATION_MAP ./location_map
```

5) Set the environment variable C27QHDL_BIN to the path pointing to the binary files.

   For Solaris, enter the following command at the system prompt:

```
% setenv C27QHDL_BIN <installation_DIR>/integration/
  custom_data/T320C2700B0_2.01/sun5/quickhdl/obj/1.8v
```

   For HP, enter the following command at the system prompt:

```
% setenv C27QHDL_BIN <installation_DIR>/integration/
  custom_data/T320C2700B0_2.01/hp700/quickhdl/obj/1.8v
```

6) Invoke the QuickHDL simulator qhsim by entering the following command at the system prompt:

```
% qhsim –t ps c3test &
```

7) Give an initial stimulus for a proper reset and clocks by entering the following command on the qhsim command line:

```
% do stimulus.do
```

8) Run for the required number of cycles by entering the following command on the qhsim command line:

```
% run <total time>
```

Note:

Through VHDL configurations, you can select either the function-only model (see section 2.5.3 on page 2-12) or the VHDL initiative towards ASIC libraries (VITAL)-based timing model (see section 2.4.3 on page 2-10) for the T320C2700B0 core. By default, the timing model is selected. The default timing model has no configuration bindings. If you choose to do function-only simulations, you must have the configuration design unit conf_t320c2700b0_func selected in your design.

Appendix B

Cadence Verilog-XL Model Installation

This appendix describes the procedure for installing the Cadence Verilog-XL model in the design.

For Verilog-XL commands and related information, see the Cadence Online Help.

Topic **Page**

B.1 Installation Directory Structure

This section describes how to uncompress the 'C2700B0 release and display the subdirectories that are necessary for running the simulations. Before you begin the installation process, you must download the installation file from the design kit website or the FTP site specified in the design kit release memo.

To install the Cadence Verilog-XL model on your system, follow these steps:

1) Change the directory to the location where the design kit needs to be installed by entering the following command at the system prompt:

   ```
   % cd <installationDir>
   ```

2) Uncompress the compressed release by entering the following command at the system prompt:

   ```
   % unzip T320C2700B0_cadence_designkit_<version>.zip
   ```

   This command creates a directory called T320C2700B0_cadence_designkit_<version>.

3) Go to this directory by entering the following command at the system prompt:

   ```
   % cd T320C2700B0_cadence_designkit_<version>
   ```

4) List the contents of this directory:

   ```
   % ls
   ```

   This command displays the README file and the integration subdirectory. The README file contains detailed components of the design kit.

5) Go to the integration subdirectory:

   ```
   % cd integration
   ```

6) List the contents of the integration subdirectory to display the custom_data subdirectory.

7) Go to the custom_data subdirectory:

```
% cd custom_data
```

8) List the contents of the custom_data subdirectory to display the following subdirectories:

```
custom_data/
           COFF2SIM/          : Contains utility to convert COFF to
                                programmation file
          T320C2700B0_2.01/   : Contains various components of the core
                                (see the following directory structure)
          cDSP_CROMW_1.0/     : ROM wrapper. contains the VHDL RTL,
                                VHDL netlist, and the Verilog netlists
          cDSP_MVSRAMW_1.0/    : SARAM wrapper for MV ACE memory contains
                                the VHDL RTL, VHDL netlist, and the
                                 Verilog netlists
          cDSP_P2XINTF_1.0/    : XINTF module contains the VHDL RTL, VHDL
                                netlist, and the Verilog netlists
          cDSP_SRAMW_2.0/     : SARAM wrapper for MK and MR ACE memories
                                contains the VHDL RTL, VHDL netlist, and
                                the Verilog netlists
```

The wrappers and the XINTF directories contain the VHDL RTL, the VHDL netlist, and the Verilog netlists, which capture the functionality of the wrappers for simulation purposes.

The following is the directory structure for the T320C2700B0 core:

```
T320C2700B0_2.01/
           1.8v/
                 GOOD          : Contains GOOD database
                 rules         : Contains Detector Rules
           hp700/
            synopsys           : Contains Synopsys black-box models
             veritool
                lib/
                 1.8v/         : Contains Verilog model

             obj/              : Contains the Verilog archive libraries

for the core
          sun5                  : Same as hp700
         documentation/        : Contains the datasheet & modeling guide
            example/           : Contains the Verilog example testbench
```

You are now ready to start the instantiation process.

B.2 Verilog-XL Model Instantiations

Before you begin instantiating your model, make sure that the Cadence environment variable CDS_INST_DIR is set to point to the cds9504 Cadence installation directory.

To instantiate the Verilog-XL 'C2700B0 model, you must edit the design netlist and select the T320C2700B0 model from the installation directory.

For details of the instantiation process and Verilog-XL commands, see the Cadence Online Help.

The instantiation process for an example Verilog-XL testbench is shown in section B.3, *Verilog-XL Example Testbench*.

B.3 Verilog-XL Example Testbench

The Verilog-XL example testbench allows you to see the instantiation process for the ’C2700 DSP core, the SARAM wrapper, the CROM wrapper, and the ACE memories. The memory mapping for this testbench is as follows:

Program Space	Data Space
B0 SARAM 1K × 16 @0x0	B0 SARAM 1K × 16 @0x400
——	B1 SARAM 1K × 16 @0x0
RAM_1 2K × 16 @1000	RAM_1 2K × 16 @1000
RAM_2 1K × 16 @2000h	——
ROM_1 2K × 16 @3FF800	ROM_1 1K × 32 @3FF800

The program is preloaded into the ROM. The code is then downloaded into B0, routines moved to RAM @1000h, and data constants moved to B1. The code is finally run from B0.

To run the Verilog-XL example testbench using the 'C2700B0 model, follow these steps:

1) Go to the example directory by entering the following command at the system prompt:

```
%  cd  <installationDIR>/T320C2700B0_cadence_design-
kit_2.00/integration/custom_data/T320C2700B0_2.00/docu-
mentation/example
```

2) In the directory example, set the environment variable to the Cadence installation directory by entering the following command at the system prompt:

```
% setenv CDS_INST_DIR <cadence installationDIR>
```

3) Create the Verilog executable by entering the following command at the system prompt:

```
% cr_vlog
```

4) Invoke Verilog-XL by entering the following command at the system prompt:

```
% verilog –f cmd_file –s
```
(Here, Verilog is the executable)

5) Run Verilog-XL by entering the following commands consecutively at the Verilog-XL command prompt:

```
> #36450 $stop;
> .
> $finish;
```

Appendix C

ModelSim Verilog Model (Windows NT) Installation

This appendix describes the procedure for installing the ModelSim Verilog Model (Windows NT) in the design.

For the ModelSim Verilog Model (Windows NT) commands and related information, see the ModelSim Online Help.

Topic **Page**

C.1 Installation Directory Structure

This section describes how to uncompress the 'C2700B0 release and display the subdirectories that are necessary for running the simulations. Before you begin the installation process, you must download the installation file from the design kit website or the FTP site specified in the design kit release memo.

To install the ModelSim Verilog Model (Windows NT) on your system, follow these steps:

1) Change the directory to the location where the design kit needs to be installed by entering the following command at the system prompt:

   ```
   cd <installationDir>
   ```

2) Uncompress the compressed release by entering the following command at the system prompt:

   ```
   unzip T320C2700B0_cadence_designkit_<version>.zip
   ```

   This command creates a directory called T320C2700B0_cadence_designkit_<version>.

3) Go to this directory by entering the following command at the system prompt:

   ```
   cd T320C2700B0_cadence_designkit_<version>
   ```

4) List the contents of this directory:

   ```
   ls
   ```

   This command displays the README file and the integration subdirectory. The README file contains detailed components of the design kit.

5) Go to the integration subdirectory:

   ```
   cd integration
   ```

6) List the contents of the integration subdirectory to display the custom_data subdirectory.

7) Go to the custom_data subdirectory:

```
cd custom_data
```

8) List the contents of the custom_data subdirectory to display the following subdirectories:

```
custom_data/
           COFF2SIM/          : Contains utility to convert COFF to
                                programmation file
          T320C2700B0_2.01/   : Contains various components of the core
                                (see the following directory structure)
          cDSP_CROMW_1.0/     : ROM wrapper. contains the VHDL RTL,
                                VHDL netlist, and the Verilog netlists
          cDSP_MVSRAMW_1.0/    : SARAM wrapper for MV ACE memory contains
                                the VHDL RTL, VHDL netlist, and the
                                 Verilog netlists
          cDSP_P2XINTF_1.0/    : XINTF module contains the VHDL RTL, VHDL
                                netlist, and the Verilog netlists
          cDSP_SRAMW_2.0/     : SARAM wrapper for MK and MR ACE memories
                                contains the VHDL RTL, VHDL netlist, and
                                the Verilog netlists
```

The wrappers and the XINTF directories contain the VHDL RTL, the VHDL netlist, and the Verilog netlists, which capture the functionality of the wrappers for simulation purposes.

The following is the directory structure for the T320C2700B0 core:

```
T320C2700B0_1.00/
           1.8v/
                 GOOD          : Contains GOOD database
                 rules         : Contains Detector Rules
           winNT/
             veritool/
               lib/
                 1.8v          : Contains the Verilog model
               obj/
                               : Contains the model dynamic linkable
                                  library (DLL)

         documentation/        : Contains the datasheet & modeling guide
          example/             : Contains the Model example testbench
```

You are now ready to start the instantiation process.

C.2 ModelSim Verilog Model (Windows NT) Instantiations

To instantiate the ModelSim Verilog Model, you must edit the design netlist and select the T320C2700B0 model from the installation directory.

For details of the instantiation process and the ModelSim Verilog Model commands, see the ModelSim Online Help.

The instantiation process for an example ModelSim Verilog Model testbench is shown in section C.3, *ModelSim Verilog Model (Windows NT) Example Testbench*.

C.3 ModelSim Verilog Model (Windows NT) Example Testbench

The ModelSim Verilog Model example testbench allows you to see the instantiation process for the 'C2700 DSP core, the SARAM wrapper, the CROM wrapper, and the ACE memories. The memory mapping for this testbench is as follows:

Program Space	Data Space
B0 SARAM 1K × 16 @0x0	B0 SARAM 1K × 16 @0x400
——	B1 SARAM 1K × 16 @0x0
RAM_1 2K × 16 @1000	RAM_1 2K × 16 @1000
RAM_2 1K × 16 @2000h	——
ROM_1 2K × 16 @3FF800	ROM_1 1K × 32 @3FF800

The program is preloaded into the ROM. The code is then downloaded into B0, routines moved to RAM @1000h, and data constants moved to B1. The code is finally run from B0.

To run the ModelSim Verilog Model example testbench using the 'C2700B0 model, follow these steps:

1) Go to the example directory by entering the following command at the system prompt:

```
cd          <installationDIR>/T320C2700B0_winNT_design-
kit_1.00/integration/custom_data/T320C2700B0_1.00/
documentation/example
```

2) Edit the run.bat script for the correct ModelSim installation directory path.
3) Run the run.bat script by entering the following command at the system prompt:

```
run.bat
```

4) In the ModelSim graphical user interface (GUI), select Project and set it to modelsim.ini (see the ModelSim Online Help).

   This loads up the T320C2700B0.dll into the simulation.

5) Make sure that the T320C2700B0.dll is loaded by looking out for the system generated message prompt in the ModelSIM GUI.
6) Run the simulations by entering the following command on the ModelSim Verilog command line:

```
vsim –t ps –lib work +mindelays toptb
```

   In this step, toptb is the module name.
   You will now be in the Transcript window.

7) On the Transcript window command line, enter the following command to get to the Waveform window:

```
do wave.do
```

8) Run for 36450 ns by entering the following command at the system prompt:

```
run 36450
```

Index

E

F

G

I

L

M

O

Q

U

V

W

X