

---

# ***CoreMP7 Development Kit***

*User's Guide*



---

## **Actel Corporation, Mountain View, CA 94043**

© 2006 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 50200075-0

Release: August 2006

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.

Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.

This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

### **Trademarks**

Actel and the Actel logo are registered trademarks of Actel Corporation.

Adobe and Acrobat Reader are registered trademarks of Adobe Systems, Inc.

All other products or brand names mentioned are trademarks or registered trademarks of their respective holders.

---

# Table of Contents

	Introduction . . . . .	5
	Document Contents . . . . .	5
	Document Assumptions . . . . .	5
1	Contents and System Requirements . . . . .	7
	Development Kit Contents . . . . .	7
	System Requirements . . . . .	7
2	Hardware Components . . . . .	9
	CoreMP7 Evaluation Board . . . . .	9
	Detailed Board Description and Usage . . . . .	9
	PLL Parts/Usage on M7A3P/E . . . . .	11
	Programming the Development Kit with a FlashPro3 Programmer . . . . .	14
3	Setup and Self Test . . . . .	25
	Software Installation . . . . .	25
	Hardware Installation . . . . .	25
	Programming the Test File . . . . .	25
4	Actel CoreMP7 Design Flow . . . . .	27
	CoreMP7 System Creation . . . . .	27
	FPGA Design Creation and Verification . . . . .	29
	FPGA Design Implementation . . . . .	30
	FPGA Programming Software . . . . .	31
	Microprocessor Design Creation and Programming . . . . .	31
5	Quickstart Tutorial . . . . .	33
	Actel CoreConsole 1.1 . . . . .	34
	Actel Libero IDE v7.1 . . . . .	49
	ARM RealView Developer Kit – Actel Edition . . . . .	86
	Running the Reversi Game via the On-Chip Debugger . . . . .	104
A	M7A3PE600 and M7A3P1000 FG484 Package Connections . .	105
	484-Pin FGBGA Package . . . . .	106

<b>B</b>	<b>Board Schematics . . . . .</b>	<b>125</b>
	Top-Level View . . . . .	125
	CoreMP7 Schematics . . . . .	125
<b>C</b>	<b>Signal Layers . . . . .</b>	<b>139</b>
<b>D</b>	<b>Product Support . . . . .</b>	<b>147</b>
	Customer Service . . . . .	147
	Actel Customer Technical Support Center . . . . .	147
	Actel Technical Support . . . . .	147
	Website . . . . .	147
	Contacting the Customer Technical Support Center . . . . .	148
	<b>Index . . . . .</b>	<b>149</b>

---

# Introduction

Thank you for purchasing the Actel CoreMP7 Development Kit.

This guide provides the information required to easily evaluate the CoreMP7 intellectual property (IP) core and M7A3P/E devices.

The CoreMP7 Development Kit software includes a base set of common IP for use in your embedded system. The CoreMP7 Evaluation Board also includes additional hardware to facilitate your system development; however, additional purchases may be required to use certain hardware found on the development board, such as the 10/100 Ethernet, USB 1.1, or CAN 2.0A/B interfaces.

## Document Contents

[Chapter 1 – Contents and System Requirements](#) describes the contents of the CoreMP7 Development Kit.

[Chapter 2 – Hardware Components](#) describes the components of the CoreMP7 Evaluation Board.

[Chapter 3 – Setup and Self Test](#) describes how to set up the CoreMP7 Evaluation Board and how to perform a self test.

[Chapter 4 – Actel CoreMP7 Design Flow](#) introduces the design flow for CoreMP7 using Actel CoreConsole®, Actel Libero® Integrated Development Environment (IDE), and ARM® RealView Developer Kit.

[Chapter 5 – Quickstart Tutorial](#) illustrates a sample Verilog design for the CoreMP7 Evaluation Board.

[Appendix A – M7A3PE600 and M7A3P1000 FG484 Package Connections](#) provides a table listing the board connections.

[Appendix B – Board Schematics](#) provides illustrations of the CoreMP7 Evaluation Board.

[Appendix C – Signal Layers](#) provides illustrations of the six signal layers of the CoreMP7 Evaluation Board.

[Appendix D – Product Support](#) describes Actel support services.

## Document Assumptions

This user's guide assumes the following:

- You intend to use Actel Libero IDE and ARM RealView Developer Kit.
- You have installed and are familiar with Actel Libero IDE v7.0 and ARM RealView Developer Kit v2.2, or later versions of either suite.
- You are familiar with Verilog.
- You are familiar with PCs and the Windows operating system.



---

# Contents and System Requirements

This chapter details the contents of the CoreMP7 Development Kit and lists the power supply and software system requirements.

## Development Kit Contents

The CoreMP7 Development Kit includes the following:

- CoreMP7 Evaluation Board
- Actel Libero IDE Gold
- Actel CoreConsole IP Deployment Platform
- Actel SoftConsole GNU-based C compiler with basic debugger and FlashPro3 JTAG support
- CoreMP7 User's Guide and Tutorial
- CD-ROM with design examples
- Universal 9 V DC power supply providing output up to 2 A  
CUI, Inc. Part Number: DTS090220U-P5P-SZ
- Actel FlashPro3 Programmer (optional, depending on kit ordered)

For the CD-ROM contents, review the *ReadMe.doc* file at the top level of the CD-ROM.

## System Requirements

The system requirements for Actel Libero IDE and ARM RealView Developer Kit are as follows:

- 1.0 GHz Pentium-class processor
- 750 MB hard disk space
- 256 MB RAM
- CD-ROM drive
- USB 1.1 (USB 2.0 recommended)
- Windows 2000 SP4 or Windows XP SP2





---

# Hardware Components

This chapter describes the hardware components of the CoreMP7 Evaluation Board.

## CoreMP7 Evaluation Board

Figure 2-1 on page 10 shows a top-level view of the CoreMP7 Evaluation Board. The board consists of the following:

- Wall-mount power supply connector with switch and LED indicator
- Switches to select from 1.5 V, 2.5 V, and 3.3 V  $V_{CCI}$  (I/O Bank) voltages on banks 4–7 (for the M7A3PE600) or banks 3–4 (for the M7A3P1000)
- 10-pin, 0.1"-pitch programming connector compatible with Altera connections
- 48 MHz oscillator and 32 kHz oscillator for real-time clock (RTC) calculations
- Eight LEDs driven by outputs from the device
- Jumpers allowing disconnection of all external circuitry from the FPGA
- One monostable pulse generator switch
- Eight switches providing input to the device
- Two RS-232 serial interfaces
- Two 10/100 Ethernet interfaces (only populated on boards with M7A3P1000)
- One Controller Area Network (CAN) 2.0B serial interface
- One USB 1.1 serial interface

For further information, refer to “M7A3PE600 and M7A3P1000 FG484 Package Connections” on page 105 and “Board Schematics” on page 125.

## Detailed Board Description and Usage

The CoreMP7 Evaluation Board has various advanced features that are covered in later sections of this chapter. The Development Kit version can be identified as the one that has the FPGA soldered directly to the board.

A block diagram of the CoreMP7 Evaluation Board is shown in Figure 2-1 on page 10 and will facilitate understanding of the more detailed schematics shown in “Board Schematics” on page 125.

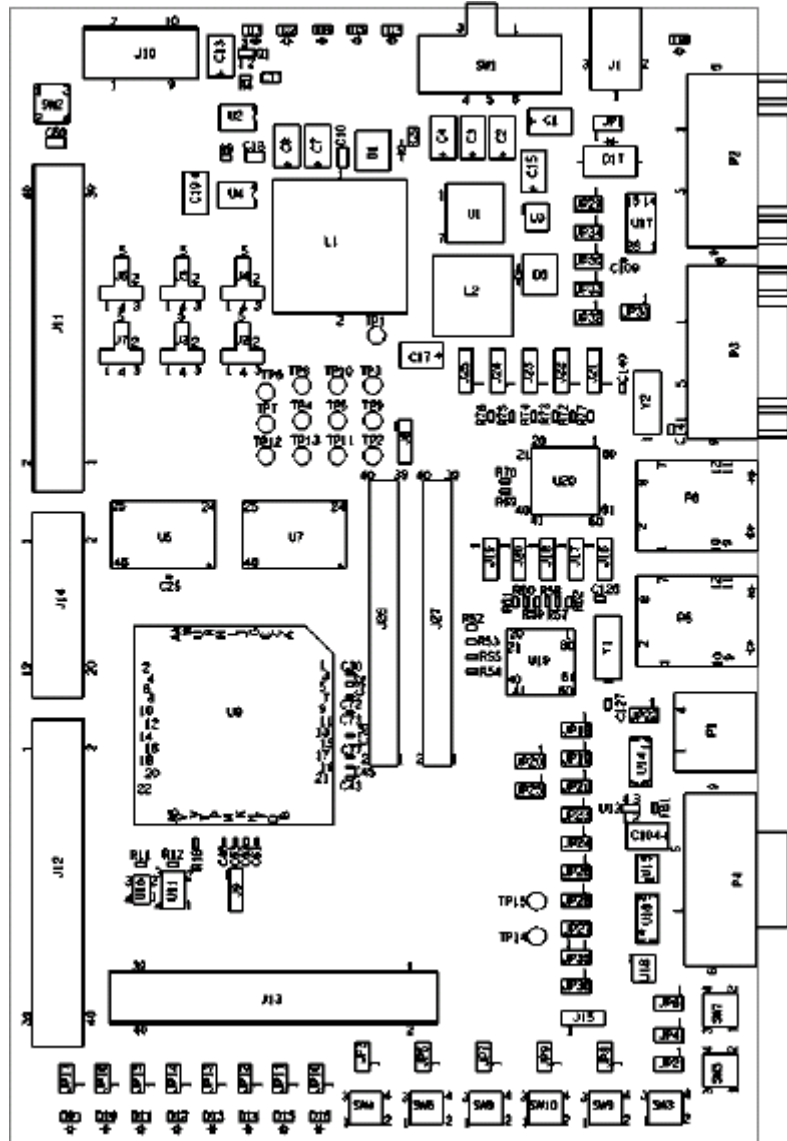


Figure 2-1. CoreMP7 Evaluation Board Top-Level View

Full schematics are available on the Development Kit tutorial CD-ROM supplied with the kit. The schematics are also available for download from the [Actel website](#). The dedicated electronic version of the schematics can be enlarged to a far greater degree than can be shown in the printed version of this manual; hence, the interested reader is referred to the dedicated schematics for the appropriate level of detail.

## PLL Parts/Usage on M7A3P/E

### Instructions for PLL Activation on the CoreMP7 Evaluation Board

To use the PLLs on the CoreMP7 Evaluation Board, power must be applied to their respective analog supply rails. For the west side middle PLL, known as PLF, the VCCPLF line must be connected to VCC, which is held at 1.5 V. The same is true of VCCPLC for the PLL on the east side, known as PLC. These voltages are not connected by default on the board for three reasons:

- The PLC analog voltage rails are not available on M7A3P devices, only in the M7A3PE family; only the west side PLL, namely PLF, is available on M7A3P devices. On M7A3P devices, the remaining pins are used as general purpose I/Os. The same board is used for M7A3PE and M7A3P devices.
- The aim is to demonstrate the lowest possible power consumption for the part. Perpetually powering the PLL lines would not achieve the lowest power.
- It is easy to connect the appropriate pins together when desired. This is why the pins are available on the jumper-based headers.

A variety of valid connections is possible. Three examples are as follows:

- For PLF, connect pin M6 (VCCPLF) to VCC via jumper JP42.
- For PLC, connect pin M18 (VCCPLC) to VCC via jumper JP44.
- For PLA, connect pin F7 (VCCPLA) to VCC via jumper JP40

**Note:** PLA, PLB, PLD, and PLE are only available on M7A3PE devices.

To facilitate use, Actel supplies jumpers with selected production versions of the kit to allow users to quickly connect and disconnect these voltage supply rails. If a user has lost the jumpers or has a kit without jumpers, it is a simple matter of soldering short, insulated connecting wire to the appropriate header pins on the corresponding PLL jumper block.

## Power Supplies

A 9 V power supply is provided with the Development Kit (Figure 2-2). There are many power supply components on the Evaluation Board to illustrate the many ways that differing voltage banks may be used with M7A3P and M7A3PE technology. These voltage banks are not all required for general use of the M7A3P silicon. They are provided for illustrative purposes only.

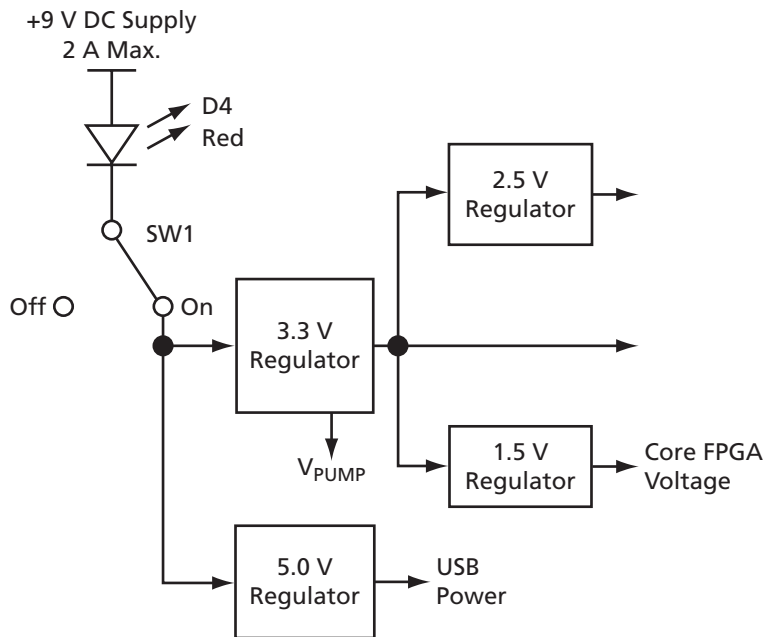


Figure 2-2. Power Supply Block Diagram

To use the CoreMP7 Evaluation Board with a wall-mount power supply, use the switching brick power supply provided with the kit.

The external +9 V center-positive power supply provided to the board via connector J1 goes to a voltage regulator chip, U1. As soon as the external voltage is connected to the board, the red “power applied” LED, D4, illuminates to indicate that an external supply has been connected. As soon as switch SW1 is moved to the ON position, the disabling ground signal is removed from pin 7 of U1, and the regulator begins to provide power at its output.

The switching voltage regulator (U1) provides a dedicated 3.3 V supply at its output. The board’s 3.3 V supply is used to feed separate regulators that deliver 1.5 V (via U2) and 2.5 V (via U4). The 1.5 V supply is required for the core voltage of the M7A3P/E family, and the 2.5 V supply is required for demonstrating LVDS extended I/O bank capability.

The presence of these voltages is indicated by the illumination of three green LEDs (D2, D3, and D7) at the top middle of the board. Each LED is labeled with the voltage it represents and its component identifier. All three voltages are selectable on I/O banks 4–7 on the M7A3PE device.

**Note:** Only M7 ProASIC3E devices have eight I/O banks. M7 ProASIC3 devices have four I/O banks—one per side of the FG484 package.

The 3.3 V supply can also be used to provide the  $V_{PUMP}$  programming voltage.  $V_{PUMP}$  can be provided to the chip during programming by applying a FlashPro3 programmer to the J10 interface and selecting  $V_{PUMP}$  from the FlashPro v4.0 (or later) programming software. The  $V_{PUMP}$  voltage can also be provided directly to the chip from the board. Leave the JP39 jumper in place to apply the 3.3 V supply to the  $V_{PUMP}$  pin (U17 on the FG484 package).

**Note:** If both FlashPro3 and the board are selected to provide  $V_{PUMP}$ , the connection on the board will override, as FlashPro3 will detect that a voltage is available, issue an information message in the programming software, and then tristate the  $V_{PUMP}$  output pin, allowing the board to provide all the power.

The board must be powered up during programming, as the chip needs its core voltages provided, and  $V_{JTAG}$  must be detected by the FlashPro3 programmer before it can set its JTAG signal voltages to the correct level.

USB has its own dedicated 5 V power supply, all components of which (including the regulator U3) are marked on the circuit board in a boxed area to indicate which components on the PCB are associated with which tasks. A green LED (D5) representing 5 V supply availability is located at the top middle of the board.

The external +9 V power supply is rated at 2 A maximum. In the first of the schematics shown in “Board Schematics” on page 125, it can be seen that the 3.3 V supply is rated at 5 A maximum. The derived power supplies of 1.5 V and 2.5 V are each rated at 2 A maximum, and the USB 5 V power supply is rated at 500 mA, as shown in Figure B-3 on page 128. As such, the derived supplies cannot all be working at their maximum current outputs simultaneously. The maximum ratings are given for the individual regulator ICs and cannot be added together.

Both U1 (LM2678S-3.3) and U3 (LM2674M-5.0) are rated for an input voltage range of +8 V to +40 V, so a wide range of power supplies can be used with the board with no concern about over-voltage conditions occurring from inadvertent usage of the wrong power supply. However, the user should take care to ensure that the voltage provided is positive at the center pin of the J16 connector and grounded on the outside.

**Note:** Greater heating of the regulator chips will be observed with higher voltages. It is therefore recommended that only the included power supply or an equivalent substitute be used with the Development Kit. The included power supply has been rated for this board, including any Actel daughter cards that may be attached to the board.

## Programming the Development Kit with a FlashPro3 Programmer

The same board is used for all CoreMP7 Development Kits. The COREMP7-E600-DEV-KIT board is fitted with a M7A3PE600-FG484 device, and the COREMP7-1000-DEV-KIT board is fitted with a M7A3P1000-FG484 device. Further, there are two additional variations of the CoreMP7 Development Kit: COREMP7-E600-DEV-KIT-FP3 and COREMP7-1000-DEV-KIT-FP3. The only difference between these two is the designator -FP3, which indicates that the kit includes the FlashPro3 programmer.

### Connecting the FlashPro3 Programmer to the Board

**To connect the FlashPro3 programmer to the board:**

1. Connect the FlashPro3 programmer to your computer via the USB cable.
2. Follow the instructions in the *FlashPro User's Guide* (software v4.0 or later) for installing the software and connecting to FlashPro3. The amber (yellow) power LED on the FlashPro3 should be illuminated at this stage. If it is not, recheck the procedure given in the *FlashPro User's Guide* until you obtain steady illumination of the amber power LED.
3. Make sure the board power switch SW1 is in the OFF position and only the red external power LED is illuminated on the board.
4. Connect the FlashPro3 programmer to the board via the 10-pin programming cable supplied with the FlashPro3 programmer. The connector to use on the board is labeled FP3\_JTAG (J10) and has a keyed header. The pin 1 location on the cable, indicated by the red ribbon running along the side of the cable, will be on the left side as it enters the board.

After connecting the FlashPro3 programmer, you can verify communication by checking **Device Info** in the FlashPro software. The M7A3P/E details will be shown in the software log window. If you suspect a JTAG communication problem, try changing the V<sub>JTAG</sub> voltage. To overcome noise, higher values usually work better, but all values should work with the supplied programming cable (6" in length) connected to just one board.

### Programming or Reprogramming the Example Design

On the Development Kit CD, you will find a *Designer* directory containing a STAPL file for programming the target design. Select the *TOP\_M7A3PE6.STP* file (for M7A3PE600 parts) or the *TOP\_M7A3P1K.STP* file (for M7A3P1000 parts) from the CD and use that as the STAPL file in the FlashPro software. Selecting **Program** will erase, program, and verify the part.

## **Jumpers for Isolating Switches, LEDs, and Other Components from the FPGA**

Many jumpers are provided on the board to allow the user to disconnect various switch combinations and LEDs from the FPGA I/O banks. All such jumpers are shown in the schematic in [Figure B-8 on page 133](#) and are labeled on the top-layer silkscreen as JP\*, where \* is a number. All jumpers are also labeled with the FPGA I/O pin number to which they are connected; e.g., JP29, for the TX0 connection of the RS-232 transmitter to the FPGA, is labeled “F18,” which indicates that it is connected to pin F18. Similarly, SW4 has a jumper above it, JP3, that is labeled “T5,” indicating that SW4 is connected to pin T5 of the FPGA when the jumper is in place.

Disconnecting jumpers JP2–JP9 causes the push button switches (SW3–SW10, respectively) to be disconnected from the FPGA so that I/O pins T4, T5, R6, R5, U2, U3, P6, and P7 can be used for other purposes. Disconnecting the eight jumpers, JP10–JP17, causes the eight LEDs (D9–D16) to be disconnected from FPGA I/O pins R4, P5, R2, T2, P2, N2, N6, and N7, respectively.

The push-button switch SW2 (labeled RESET#), meant for applying a reset pulse, is connected to pin W15, a chip-wide global. Again, all labeling is clearly shown on the silkscreen. This flexibility is useful for experimentation with designs of your own choosing and in connecting other external equipment to the board for development purposes.

## LED Connections

Eight LEDs are connected to the device via jumpers. If the jumpers are in place, the device I/O can drive the LEDs. The LEDs change based on the output as follows:

- A '1' on the output of the device lights the LED.
- A '0' on the output of the device switches off the LED.
- An unprogrammed or tristated output may show a faintly lit LED.

**Note:** If the I/O voltage of Bank 5 (on A3PE, set by J6) or Bank 2 (A3P, set by J6) is not at least 2.5 V, the LEDs will not illuminate. A setting of 1.8 V on the voltage bank will cause extremely faint illumination.

Table 2-1 lists the jumper and device connection associated with each LED.

Table 2-1. LED Device Connections

LED	Jumper	Device Connection
D9	JP17	U9 pin N7
D10	JP16	U9 pin N6
D11	JP15	U9 pin N2
D12	JP14	U9 pin P2
D13	JP13	U9 pin T2
D14	JP12	U9 pin R2
D15	JP11	U9 pin P5
D16	JP10	U9 pin P4

To use the device I/O for other purposes, remove the jumpers.



## Switch Connections

Eight switches are connected to the device via jumpers. If the jumpers are in place, the device I/O can be driven by the switches listed in [Table 2-2](#).

- Pressing a switch drives a '1' onto the associated device I/O pin. The '1' continues to be driven while the switch is in place.
- Releasing a switch drives a zero onto the device I/O pin.

[Table 2-2](#) lists the jumper and device connection associated with each switch.

Table 2-2. Switch Device Connections

Switch	Jumper	Device Connection
SW3	JP2	U9 pin T4
SW4	JP3	U9 pin T5
SW5	JP4	U9 pin R6
SW6	JP5	U9 pin R5
SW7	JP6	U9 pin U2
SW8	JP7	U9 pin U3
SW9	JP8	U9 pin P6
SW10	JP9	U9 pin P7

## CoreUARTapb RS-232 Implementation

The CoreMP7 Development Kit includes two RS-232 ports that can be used for communication between the embedded microprocessor and a common serial port, as found on a PC or other RS-232-compatible device. To use either of the RS-232 ports, jumpers must be in place to connect the FPGA to the on-board RS-232 transceiver. The jumpers used for the RS-232 connections can be found in [Table 2-3](#).

The primary RS-232 port (P2) has lines to support RTS/CTS flow control in addition to TXD and RXD.

**Note:** Currently, CoreUARTapb does not support hardware RTS/CTS handshaking. If this functionality is needed, it must be implemented in software. The default configuration has a jumper shorting RTS0 and CTS0 together, disconnecting them from the FPGA I/Os and creating a loopback connection, similar to the implementation found on the secondary RS-232 port.

Table 2-3. RS-232 Connections

Signal	Jumper	Device Connection
TX0	JP29	U9 pin B11
RX0	JP32	U9 pin G21
RTS0	JP30	U9 pin K17
CTS0	JP33	U9 pin J19
TX1	JP31	U9 pin C11
RX1	JP34	U9 pin K18

## Core10/100 Ethernet Implementation

Core10/100 is an Ethernet Media Access Controller (MAC) that connects Local Area Networks (LANs) at data rates of 10 or 100 Mbps (see Figure 2-3). It has a Media Independent Interface (MII) for physical connection and implements Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms, per IEEE 802.3. Ethernet is a common standard used in computer, communications, industrial, and other applications.

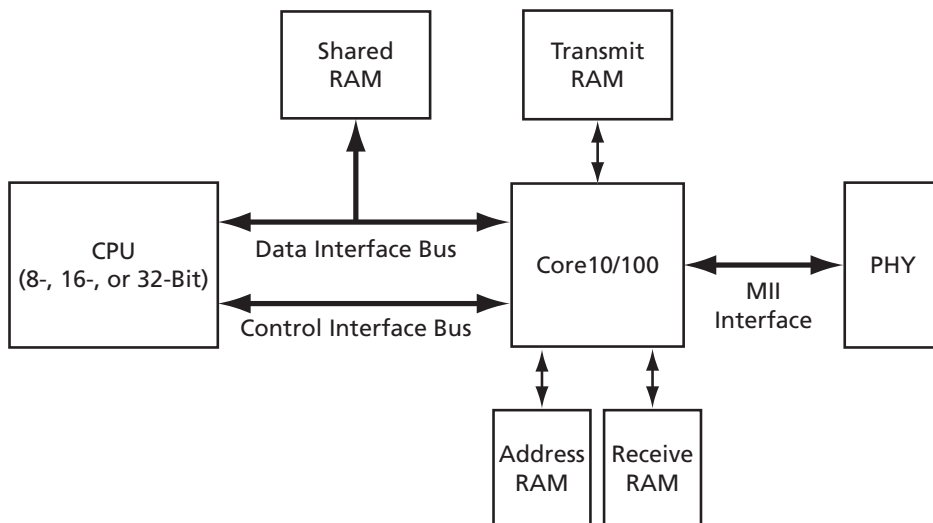


Figure 2-3. Overview of a Typical Core10/100 System

Detailed Core10/100 information is available in the *Ethernet Media Access Controller Core10/100* datasheet at [http://www.actel.com/ipdocs/Core10100\\_DS.pdf](http://www.actel.com/ipdocs/Core10100_DS.pdf).

The MII interface to Core10/100 works with most Ethernet PHY chips. Due to the analog requirements of an Ethernet PHY, such cannot be implemented in an Actel FPGA.

The CoreMP7 Evaluation Board supports dual Ethernet connections. An AM79C874VI from Advanced Micro Devices (AMD) is used for each PHY (U19 and U20). See Table 2-4 on page 20 for details on the connections between the FPGA and each PHY via J26 and J27, which are connection/disconnection points for PHY0 and PHY1, respectively.

**Note:** The dual Core10/100 Ethernet interfaces are only populated on boards based on the M7A3P1000 device.

Table 2-4. 10/100 Ethernet Connections

PHY Signal	Jumper	Device Connection (J26/J27)
MDIO	1	U12/U12
MDC	3	T12/T12
TXD0	5	V10/V12
TXD1	7	U9/V11
TXD2	9	U10/R12
TXD3	11	T10/R11
TX_EN	13	AB7/AA9
TX_ER	15	AB6/AA10
EXT_IN_CLK_0	17	Y7/AA8
RXD0	19	Y6/AA7
RXD1	21	U9/AB9
RXD2	23	V8/AB8
RXD3	25	AA6/W9
RX_DV	27	AA5/W8
RX_ER	29	AB5/Y10
EXT_IN_CLK_1	31	AB4/W10
COL	33	AA4/U11
CRS	35	Y4/T11
RST#	37	W15/W15
N/C	39	N/C

## USB

The CoreMP7 Evaluation Board includes a Fairchild Semiconductor USB1T11AM USB transceiver. The USB standard specifies support for multiple device connections, allowing up to 127 unique devices. Further, the Fairchild transceiver supports the transmitting and receiving of serial data at both full-speed (12 Mbps) and low-speed (1.5 Mbps) data rates. Implementation of the Serial Interface Engine (SIE) is required to use the USB interface present on the Evaluation Board. Information on the SIE can be found on the USB Implementers Forum at <http://www.usb.org>.

## CompanionCore CAN 2.0B Implementation

The CAN bus is a communication standard with multi-master capability, error detection and correction, and broad industry acceptance. The CAN bus was designed for the automobile industry, but CAN has recently been appearing in non-traditional applications. The CAN bus comprises two signals to which all networked devices are connected, thus allowing communication between multiple devices. The reliability and error detection is handled by a series of arbitrations, (not) acknowledges, and CRC checks.

Table 2-5 details the connections between the M7A3P/E FPGA and the onboard CAN transceiver (U18). The CoreMP7 Evaluation Board is also equipped with LEDs (D18 and D19) connected to the TXD and RXD lines of the CAN bus. Therefore, when data is being transmitted or received, the respective LED will blink.

If network termination is needed (typically for CAN baud rates greater than 100 kbps), shorting JP38 (CAN\_TERM) inserts a 120  $\Omega$  resistor between CAN-H and CAN-L.

Table 2-5. CAN Device Connections

CAN Signal	Jumper	Function / Device Connection
CAN_TXD	JP35	U9 pin K20
CAN_RXD	JP36	U9 pin J22
CAN_TERM	JP37	Enables termination
CAN_SHLD	JP38	Enables shield ground

## Clock Circuits

The CoreMP7 Evaluation Board has two clock circuits: a 48 MHz oscillator and a 32 kHz oscillator.

### 48 MHz Oscillator

The 48 MHz oscillator on the board is a 30 ppm–stability crystal module that provides more than adequate performance and can be connected to a general purpose I/O (pin W12) or a chip-wide global (pin W17) using a jumper.

### 32 kHz Oscillator

The 32 kHz oscillator on the board is a 30 ppm–stability crystal module that will provide enough accuracy to perform RTC calculations and is hardwired to a chip-wide global (pin V16).

## Memory

### Flash

The CoreMP7 Evaluation Board includes two STMicroelectronics M29W800DT Flash memory chips, totaling 2 MB, which can be arranged in either a  $1\text{M} \times 16$  or a  $512\text{K} \times 32$  configuration. The Flash memory is intended for use as executable program storage for the embedded microprocessor; however, it can also be used as nonvolatile memory for the storage of system constants and parameters.

### SRAM

The CoreMP7 Evaluation Board includes two GSI Technology GS8001BT Synchronous SRAM modules, totaling 2 MB, which can be arranged in either a  $1\text{M} \times 16$  or a  $512\text{K} \times 32$  configuration. The SRAM memory is used for the embedded microprocessor stacks (both hardware and software) and for dynamic system data.

## Headers

There are three headers (J11, J12, and J13) present on the CoreMP7 Evaluation Board intended for use as general purpose I/O. These pins are tied to the various chip-wide global signals in the I/O banks as well as dedicated general purpose I/Os. See the schematics in [“Board Schematics” on page 125](#) for further information.

## Test Points

All test points on the board are fitted with small test loops. These test points are labeled on the silkscreen as TP1, TP2, etc. All such test points are also labeled on the silkscreen with the voltage expected to be observed at that test point or the I/O pin to which the test point is connected. Each voltage will be either 3.3 V, 2.5 V, 1.5 V, or GND. When measuring the voltage at a test point with a DVM (digital voltage multimeter), the ground lead should be connected to a test point labeled GND, and the voltage lead should be connected to the voltage to be tested. All voltage labels on the board are relative to a 0 V ground reference (GND).

## Board Layers

The complete board design and manufacturing files are included on the Development Kit CD. The board file is in Allegro format, which will allow a user to create the appropriate Gerbers and other board views as needed. Pictures of the board layers are also included in [“Signal Layers” on page 139](#). For your convenience, high-resolution PDFs of these layers are also provided on the Development Kit CD.

The board is fabricated with six copper layers. The layers are arranged as follows, from top to bottom:

- Layer 1 – Top signal layer
- Layer 2 – Ground plane
- Layer 3 – Signal layer 3
- Layer 4 – Signal layer 4
- Layer 5 – Power plane
- Layer 6 – Bottom signal layer

Refer to the diagrams in [“Signal Layers” on page 139](#).





---

# Setup and Self Test

This chapter outlines how to set up and test the CoreMP7 Evaluation Board.

## Software Installation

The CoreMP7 Development Kit includes the Libero IDE software suite (version 7.0). For Libero IDE software installation instructions, refer to the *Actel Libero IDE / Designer Installation and Licensing Guide for Software v6.1* at [http://www.actel.com/documents/install\\_ug.pdf](http://www.actel.com/documents/install_ug.pdf).

The CoreMP7 Development Kit also includes the Actel SoftConsole GNU-based C compiler and debugger, which can be used to program and debug the CoreMP7 program memory through the FlashPro3.

## Hardware Installation

FlashPro3 is required to use the CoreMP7 Development Kit. For software and hardware installation instructions, refer to the *FlashPro v3.3 User's Guide* at <http://www.actel.com/documents/flashproUG.pdf>. FlashPro3 is also used with SoftConsole to program and debug the Flash program memory on the CoreMP7 Evaluation Board.

If you are using the ARM RealView Developer Kit, you will need to use the ARM RealView ICE Micro Edition (RVI-ME) supplied with it to program and debug the Flash program memory on the CoreMP7 Evaluation Board. For software and hardware installation instructions, refer to the documentation included on the ARM RealView installation CDs.

## Programming the Test File

To retest the evaluation board at any time, use the test program to reprogram the board. Use the *TEST\_M7A3PE6.stp* file with an M7A3PE600-FG484 fitted on the board. Use *TEST\_M7A3P1K.stp* with an M7A3P1000-FG484 fitted on the board.

The test design is currently implemented for the M7A3PE600 die size. It is possible to recompile the design for other device sizes. For information about retargeting the device, refer to the *Designer User's Guide* at <http://www.actel.com/documents/designerUG.pdf>. The design files are available under *SelfTest* on the Development Kit CD.

For instructions on programming the device using FlashPro3, refer to the *FlashPro User's Guide* at <http://www.actel.com/documents/flashproUG.pdf>.

The Flash memory on the board can be programmed using either FlashPro3 (if you are using SoftConsole) or the RVI-ME (if you are using the RealView Developer Kit). For information on programming the memory with RealView and the RVI-ME refer to *ARM Application Note #110: Flash Programming with RealView Debugger* at <http://www.arm.com/pdfs/AN110.zip>.



---

# Actel CoreMP7 Design Flow

The CoreMP7 design flow consists of the two paths, shown in [Figure 4-1 on page 28](#):

- FPGA development – the creation of the CoreMP7 system based on the Actel M7 FPGAs
- Executable code development – the creation of software programs that will execute on the embedded microprocessor core

The CoreMP7 design flow has five main components:

- CoreMP7 system creation
- FPGA design creation and verification
- FPGA design implementation
- FPGA programming
- Microprocessor design creation and programming

## CoreMP7 System Creation

CoreConsole is a system-level development tool and IP deployment platform that greatly simplifies the task of assembling and connecting IP for implementation in Actel FPGAs. It enables you to select IP components from a database supplied by Actel and graphically “stitch” them together to build a processor-based System-Level Integration (SLI) design. When the design is complete, the RTL (and other files needed to implement the design) can be generated and imported into the familiar and proven design flow of the Actel Libero IDE software. CoreConsole also generates a testbench for the SLI design that you can build to assist in verification.

Refer to the [CoreConsole User's Guide](http://www.actel.com/documents/CoreConsole_UG.pdf) at [http://www.actel.com/documents/CoreConsole\\_UG.pdf](http://www.actel.com/documents/CoreConsole_UG.pdf) for more information on using CoreConsole.

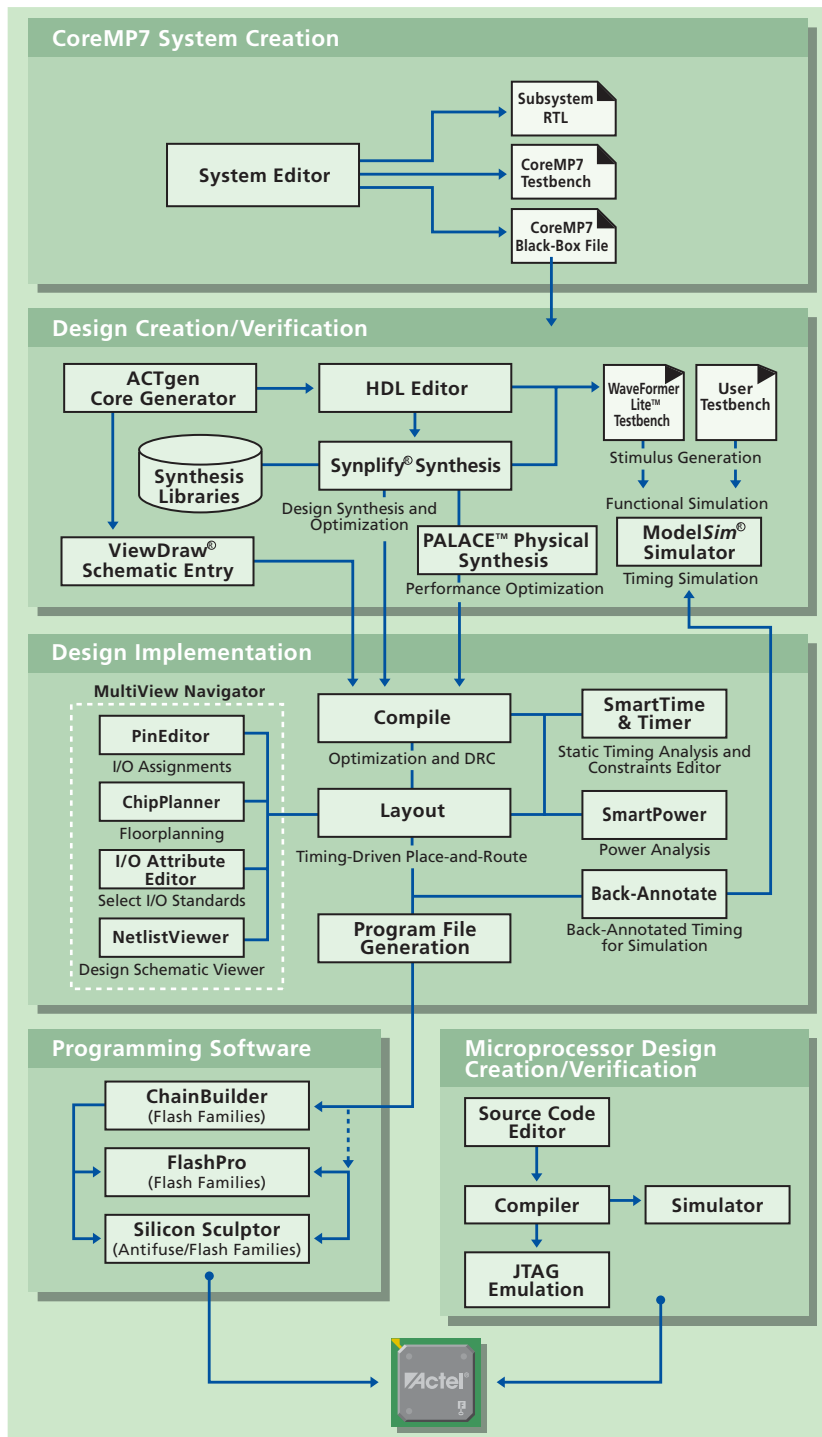


Figure 4-1. Design Flow Paths

# FPGA Design Creation and Verification

Design entry consists of writing HDL or capturing a schematic representation of the design and performing functional simulations with a testbench.

## Design Capture

For schematic capture, Libero IDE uses ViewDraw® for Actel, which includes a schematic editor. The schematic editor provides a graphical entry method to capture designs. ViewDraw for Actel is the Libero IDE integrated schematic entry vehicle, supporting mixed-mode entry, in which HDL blocks and schematic symbols can be mixed.

The ViewDraw WIR file is automatically created after using the **Save + Check** command. This file is used to create the structural HDL netlist.

For more information on using ViewDraw for Actel, refer to the *Libero IDE User's Guide for Software v7.2* at [http://www.actel.com/documents/libero\\_ug.pdf](http://www.actel.com/documents/libero_ug.pdf).

## Adding SmartGen Macros

Use the SmartGen Macro Builder to instantly create customized macros, then use ViewDraw to add these macros to a schematic. Alternatively, add the SmartGen macros in the HDL file.

## Creating and Adding Symbols for HDL Files

Schematic users can encapsulate an HDL block within a block symbol.

### To create a symbol:

1. Right-click the block in the Design Hierarchy window of Libero IDE.
2. Click **Create Symbol**. Libero IDE generates a symbol for the selected HDL block.

The macro is accessible from the components list in ViewDraw for Actel.

## Testbench Generation

To run a simulation, it is necessary to create a testbench and associate it with a project. WaveFormer Lite™ from SynaptiCAD™ is the Libero IDE integrated testbench generator. WaveFormer Lite fits perfectly into Libero IDE, automatically extracting signal information from HDL design files and producing HDL testbench code that can be used with any standard VHDL or Verilog simulator.

WaveFormer Lite generates VHDL and Verilog testbenches from drawn waveforms.

## Pre-Synthesis Simulation

Functional simulation verifies that the logic of a design is functionally correct. Simulation is performed using the Libero IDE integrated simulator, ModelSim® for Actel, which is a custom edition of ModelSim PE integrated into Libero IDE.

ModelSim for Actel is an OEM edition of the Model Technology™ Incorporated (MTI) tools. ModelSim for Actel supports VHDL or Verilog, but it can only simulate one language at a time. It only works with Actel libraries and is supported by Actel.

## Synthesis and Netlist Generation

After entering the design source, synthesize it to generate a netlist. Synthesis transforms the behavioral HDL source into a gate-level netlist and optimizes the design for a target technology. For more detailed information on the above topics, refer to the *Libero IDE User's Guide for Software v7.2* at [http://www.actel.com/documents/libero\\_ug.pdf](http://www.actel.com/documents/libero_ug.pdf).

## FPGA Design Implementation

During design implementation, Actel Designer performs place-and-route on the design.

### Place-and-Route

Start Designer from Libero IDE to place-and-route the design.

### Timing Simulation

Perform timing simulation on the design after place-and-route in Designer. Timing simulation requires information extracted and back-annotated from Designer.

### Optional Tools

The tools listed in Table 4-1 provide optional functions that are not required in a basic design. Use these tools to perform static timing analysis and power analysis, customize I/O placements and attributes, and view the netlist. Perform the post-layout (timing) simulation after place-and-route.

Table 4-1. Designer User Tools

Designer User Tool	Function
SmartTime	Static timing analysis
SmartPower	Power analysis
ChipEdit	Customize I/O and logic macro placement
PinEdit	Customize I/O placements and attributes
Netlist Viewer	View your netlist and trace paths

For more information on the tools described above, refer to the *Designer User's Guide for Software v7.2* at [http://www.actel.com/documents/designer\\_ug.pdf](http://www.actel.com/documents/designer_ug.pdf).

## FPGA Programming Software

Program the device with programming software and hardware from Actel or with a supported third-party programming system. Refer to the *Designer User's Guide for Software v7.2*, *Silicon Sculptor User's Guide*, and *FlashPro User's Guide* for information about programming an Actel device.

These guides can be found at <http://www.actel.com/techdocs/manuals/default.asp>.

## Microprocessor Design Creation and Programming

There are a large number of third party ARM7 program development tools that can be used with CoreMP7 for the development of software programs that run on the processor. Actel offers several, including the SoftConsole tools (included with the CoreMP7 Development Kit) and the RealView Developer Kit (RVDK). SoftConsole is available for free, and the RVDK can be licensed from Actel for an annual license fee. Although the RVDK has an annual license fee, the RealView C compiler generates significantly more efficient code for CoreMP7 than the SoftConsole GCC compiler.

ARM RealView Developer Kit provides a fully integrated software solution with leading-edge tools for creating efficient software to run on any ARM processor. Servicing all major market segments, RealView Developer Kit provides flexible software tools to meet present and future requirements.





---

# Quickstart Tutorial

This tutorial illustrates a Verilog CoreMP7 design for the CoreMP7 Evaluation Board. This design is created in Actel CoreConsole 1.1, Libero IDE v7.1, and ARM RealView Developer Kit. The steps involved are as follows:

## Actel CoreConsole 1.1

- “Step 1 – Creating the Basic CoreConsole Project”
- “Step 2 – Building the Subsystem within CoreConsole”
- “Step 3 – Reviewing and Generating the CoreConsole Design”

## Actel Libero IDE v7.1

- “Step 1 – Create a New Project”
- “Step 2 – Perform Pre-Synthesis Simulation”
- “Step 3 – Synthesize the Design in Synplify”
- “Step 4 – Perform Post-Synthesis Simulation”
- “Step 5 – Implementing the Design with Actel Designer”
- “Step 6 – Perform Timing Simulation with Back-Annotated Timing”
- “Step 7 – Generating the Programming File”
- “Step 8 – Programming the Device”

## ARM RealView Developer Kit

- “Step 1 – Creating a RealView Project”
- “Step 2 – Compiling the Source Files”
- “Step 3 – Debugging: Simulating/Executing the Compilation”

## Actel CoreConsole 1.1

This tutorial provides step-by-step instructions on how to create a CoreConsole project and generate a CoreConsole design. The tutorial consists of three steps: “[Step 1 – Creating the Basic CoreConsole Project](#)”, “[Step 2 – Building the Subsystem within CoreConsole](#)” on page 43, and “[Step 3 – Reviewing and Generating the CoreConsole Design](#)” on page 46.

**Note:** Before you begin this tutorial, make sure the CoreConsole software is installed.

### Step 1 – Creating the Basic CoreConsole Project

In Step 1, you learn the basic features of CoreConsole by creating a basic CoreConsole project. You will use the Actel CoreConsole IP Deployment Platform tool to develop a skeleton CoreMP7 system. This system can be simulated and synthesized; however, it is too basic for practical use and will be extended in “[Step 2 – Building the Subsystem within CoreConsole](#)”.

#### **To create the CoreConsole project:**

1. Double-click the **Actel CoreConsole 1.1** icon on your desktop to start the program, or select **Start > Programs > CoreConsole > Actel CoreConsole 1.1**.
2. From the **File** menu, select **New**. The New Design window displays, as shown in [Figure 5-1](#).

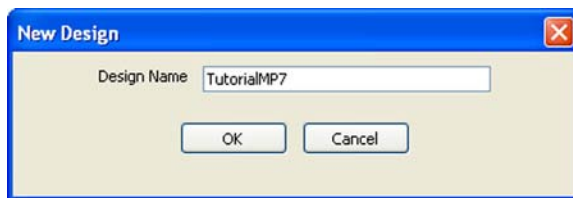


Figure 5-1. New Design Window in CoreConsole

3. Enter your **Design Name**. For this tutorial, name your design “TutorialMP7”.
4. Click **OK** to create your design project.

#### **To add components to your CoreConsole project:**

1. Under the **Components** tab, in the “Components available for selection” section, click **CoreMP7**.
2. Click the **Add** button in the “Selected Component's Details” section. The CoreMP7 component appears in your design.
3. In the “Components available for selection” section, click **CoreMP7Bridge**.
4. Click the **Add** button in the “Selected Component's Details” section.
5. Following the same process as in steps 4–5, add the component **CoreAHB**.

6. Once all three components have been added to the design, it should resemble [Figure 5-2](#).

**Note:** Some of the components are overlapping. To arrange the components neatly, select **Auto Layout** from the **Actions** menu.

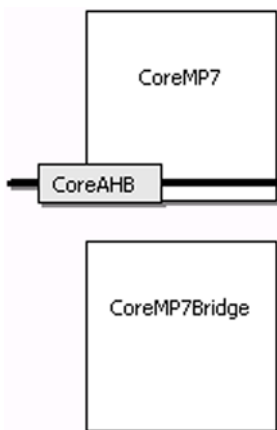


Figure 5-2. CoreConsole Schematic Window before Auto Layout

7. Following the process in steps 4–5, add the **CoreMemCtrl** component. When it appears in the schematic window, you can drag it to the right of CoreMP7 for neater appearance.

**To connect components within your CoreConsole project:**

1. From the **Actions** menu, select **Auto Stitch**. This displays the Auto Stitching window, as shown in Figure 5-3.

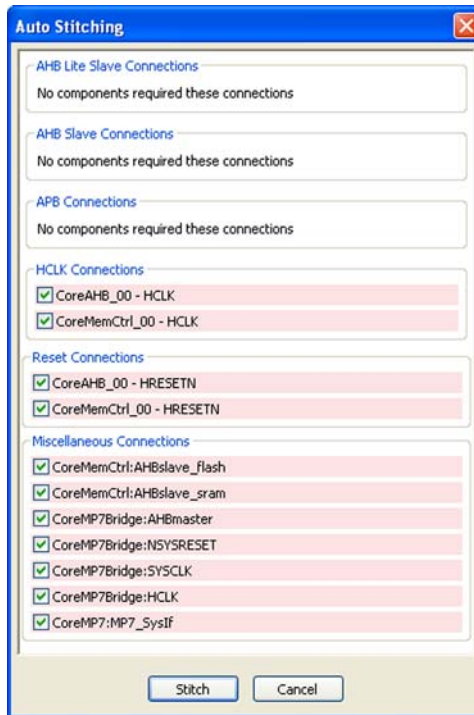


Figure 5-3. CoreConsole Auto Stitching Window

Auto Stitching connects the critical components of the system together. However, you will still need to connect most of the top-level signals manually. Manually connecting signals gives you finite control over the microprocessor's memory map.

2. Confirm that stitching has been enabled for **CoreMP7**, **CoreMP7Bridge**, **CoreAHB**, and **CoreMemCtrl**, as shown in Figure 5-3. Then click the **Stitch** button.

3. Once Auto Stitching is complete, your design should resemble Figure 5-4.

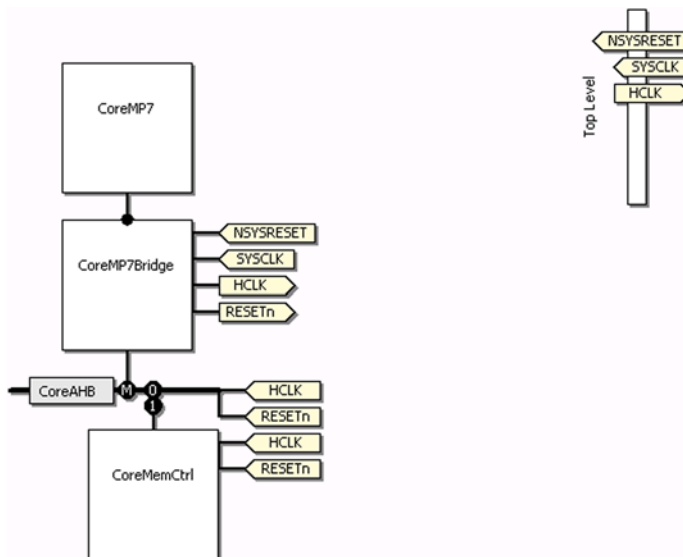


Figure 5-4. CoreConsole after Auto Stitching

The steps in the next section show you how to manually connect signals to the system's top level. The first set of instructions walks you through connecting the ARM7 JTAG interface to the top level. The second set of instructions brings the memory bus (both data and address) to the top level to interface with the Flash and SRAM modules.

4. To make additional connections, float your mouse over one of the components, such as **CoreMP7Bridge**. An options toolbar appears underneath the selected component, as shown in Figure 5-5.



Figure 5-5. Component Options Toolbar

5. Click the **Connect** icon, the first icon from the left, which resembles a power plug. When you have done this, the Configuring Connection dialog box appears, as shown in [Figure 5-6](#).

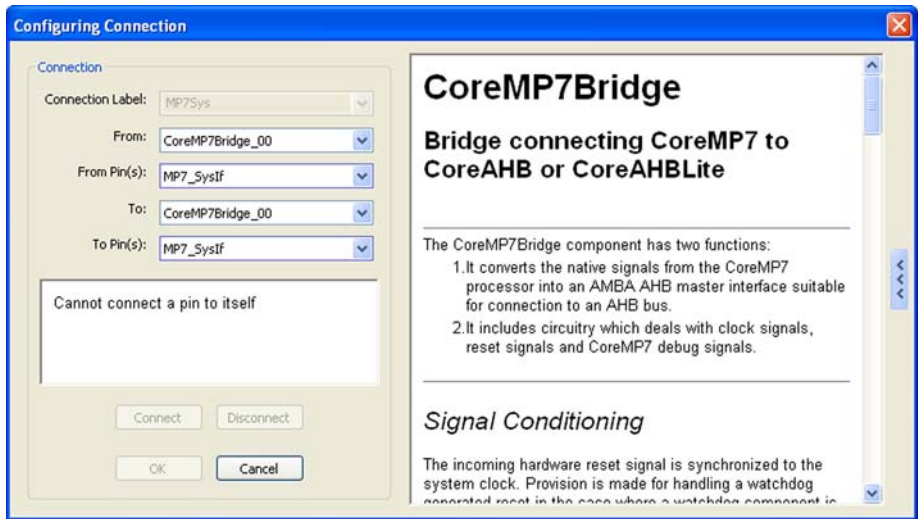


Figure 5-6. CoreConsole Configuring Connection Dialog Box

6. **CoreMP7Bridge** should automatically be selected in the **From** field. If it is not selected, select it from the drop-down menu. Then select **RV\_ICE\_If** from the **From Pin(s)** drop-down menu, as shown in Figure 5-7.

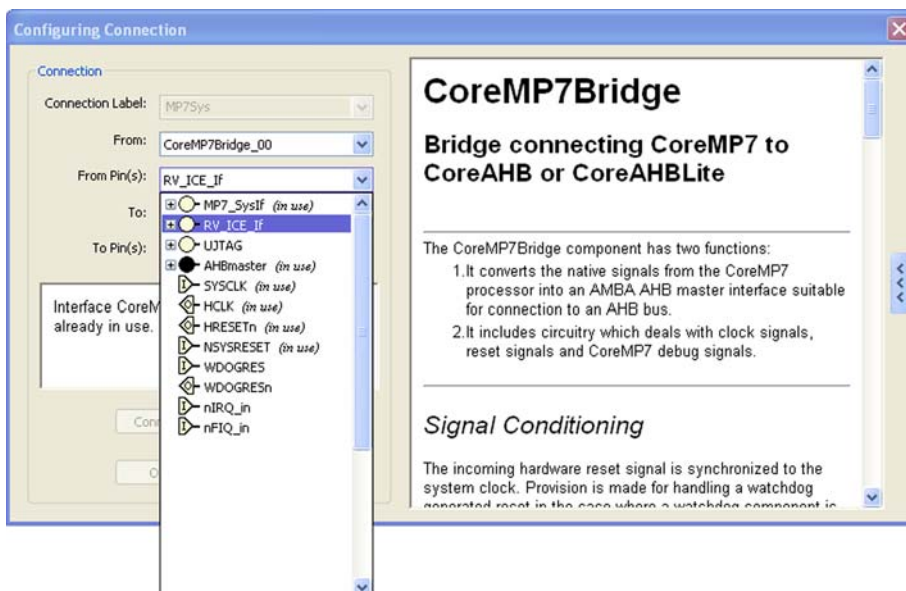


Figure 5-7. Selecting the CoreMP7Bridge RV\_ICE\_If Pins

7. Select **Top Level** in the **To** drop-down menu.
8. Enter the signal name "RV" for **Connection Label** and click **Connect**.

9. Click **OK**. Your schematic should resemble the one in [Figure 5-8](#).

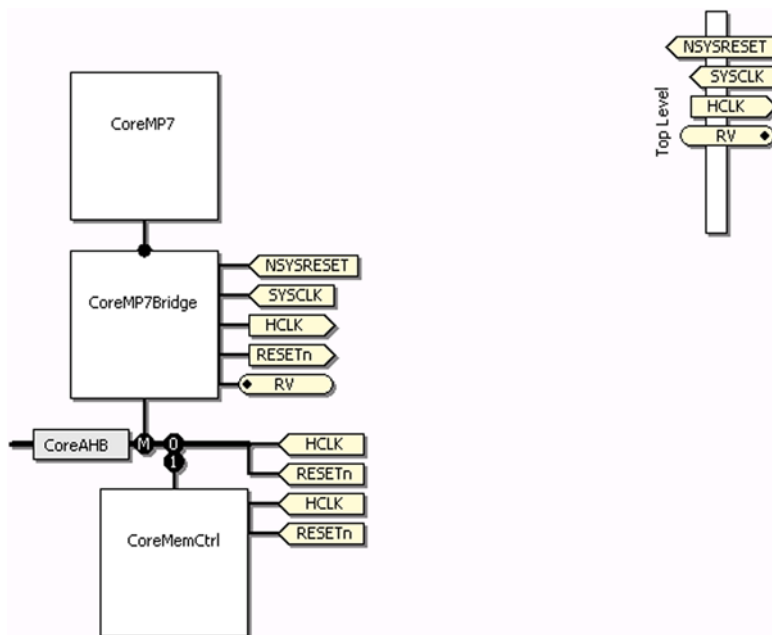


Figure 5-8. CoreConsole Schematic after Connecting ARM JTAG Interface

10. Click the **Connect** icon on the floating options menu for the **CoreMemCtrl** component.



11. From the Configuring Connection dialog box, select **ExternalMemoryInterface** in the **From Pin(s)** drop-down menu, as shown in Figure 5-9.

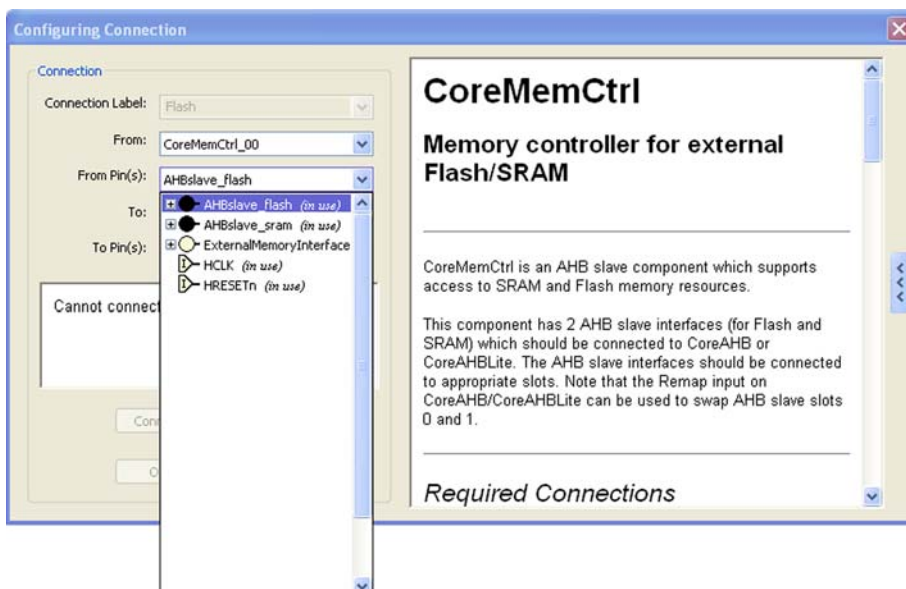


Figure 5-9. Selecting the CoreMemCtrl ExternalMemoryInterface Pins

12. Select **Top Level** in the **To** drop-down menu.
13. Enter the signal name "Mem" for **Connection Label** and click **Connect**.

14. Click **OK**. Your schematic should resemble the one in [Figure 5-10](#).

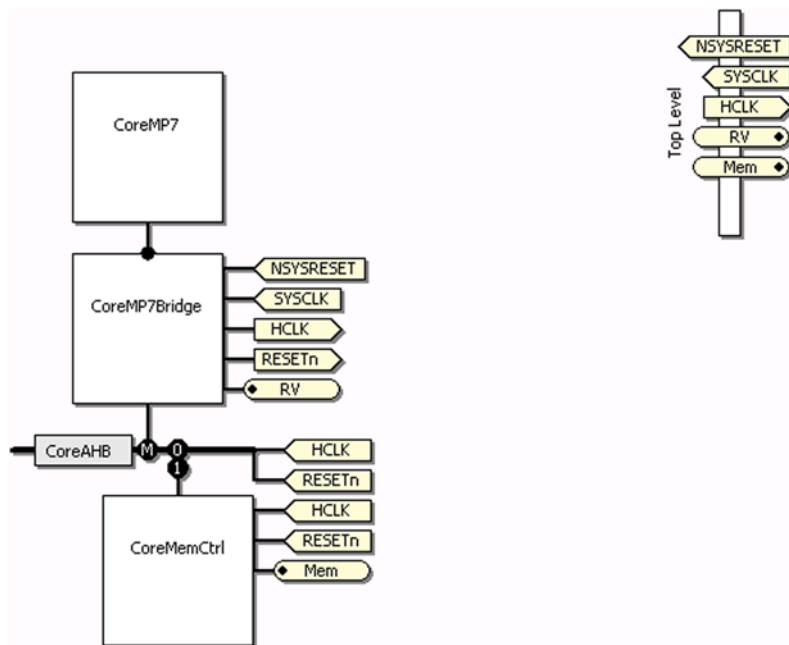


Figure 5-10. CoreConsole Schematic after Connecting the External Memory Interface

## Step 2 – Building the Subsystem within CoreConsole

Before you begin Step 2, you should know how to stitch components together, as taught in Step 1. For more information on stitching components together, review the instructions in “[Step 1 – Creating the Basic CoreConsole Project](#)” on page 34 or the *CoreConsole User Guide*.

### Adding CoreUARTapb to the System

In this section, you will add a common CoreUARTapb component to the subsystem. CoreUARTapb has an APB interface (as opposed to the high-speed AHB interface); therefore, a CoreAPB component is required and will be implemented through a bridge.

1. Add the **CoreAHB2APB**, **CoreAPB**, and **CoreUARTapb** components. For neater appearance, move CoreAHB2APB to the right of CoreMP7Bridge, with CoreAPB below the bridges and CoreUARTapb below CoreAPB.
2. Connect the components as follows:
  - Connect CoreAHB2APB through its AHBslave interface to CoreAHB via the AHBmslave12 interface.
  - Connect CoreUARTapb through its APBslave interface to CoreAPB via the APBmslave3 interface.
  - Connect the CoreUARTapb TX signal to Top Level with the connection name UART\_TX.
  - Connect the CoreUARTapb RX signal to Top Level with the connection name UART\_RX.
3. Once completed, select **Auto Stitch** from the **Actions** menu. Confirm that Auto Stitching is configured to operate on CoreAHB2APB and CoreUARTapb, then click the **Stitch** button. CoreConsole will then connect the HCLK and nRESET pins to the components you added, and connect CoreAHB2APB to CoreAPB as a master.

- Once completed, your CoreConsole schematic should look similar to Figure 5-11.

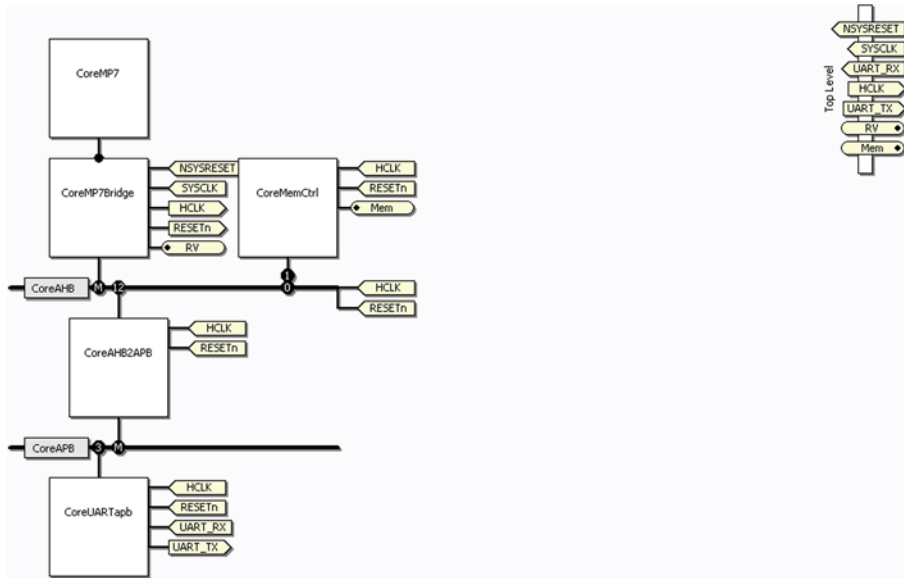


Figure 5-11. CoreConsole Schematic after Connecting CoreUARTapb

**To add the I/O block and system control registers to the system:**

- Add the **CoreGPIO** and **CoreRemap** components. For neater appearance, move CoreGPIO and CoreRemap to the right of CoreUARTapb.
- Connect the components as follows:
  - Connect CoreGPIO through its APBslave interface to CoreAPB via the APBmslave2 interface.
  - Connect CoreRemap through its APBslave interface to CoreAPB via the APBmslave15 interface.
  - Connect the CoreRemap CoreRemapDef pin to Top Level using the signal name ReMapDef.
  - Connect the CoreGPIO dataIn pin to Top Level using the signal name keyPadIn.
  - Connect the CoreGPIO dataOut pin to Top Level using the signal name ledOut.
- Once completed, select **Auto Stitch** from the **Actions** menu.

4. Confirm that Auto Stitching is configured to operate on CoreGPIO and CoreRemap, then click the **Stitch** button.  
CoreConsole will then connect the HCLK and nRESET pins to the components you added.
5. Once completed, your CoreConsole schematic should look similar [Figure 5-12](#).

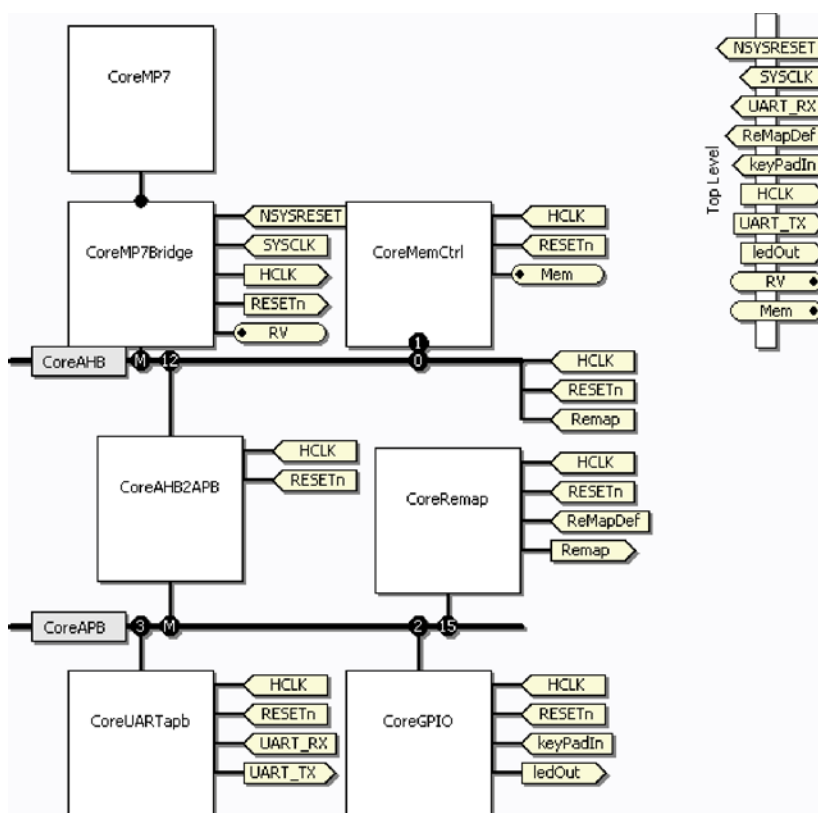


Figure 5-12. CoreConsole Schematic after Connecting CoreGPIO and CoreRemap

## Step 3 – Reviewing and Generating the CoreConsole Design

Even though you can manually view the design connections in a system of this size, this section explores features in CoreConsole for reviewing your design connections.

### **To review the CoreConsole design connections:**

1. From the **View** menu, select **Connections**. The Connections dialog box appears.
2. Examine the connections within the system for accuracy.
3. Click a connection. The appropriate Connection dialog box displays. If necessary, you can make modifications to the connection.
4. Click **OK** to close the Connections window.

### **Displaying Linked Connections**

Another useful tool is **Show Linked Connections**, which enables you to see all of your linked connections at once. This feature is enabled by default, but you can change this from **System Options > Options**.

In the schematic window, position your mouse cursor over HCLK on the Top Level bar and notice all the highlighted linked connections. You can perform this action with all of the system signals.

## Modifying Configuration Settings

Prior to generating the source code necessary for Actel Libero IDE, you must modify the configuration settings.

1. Position your mouse cursor over the **CoreMP7** component and click the **Configure** button (the second button from the left, with the binary digits).

The Configuring CoreMP7 dialog box displays, as shown in [Figure 5-13](#).

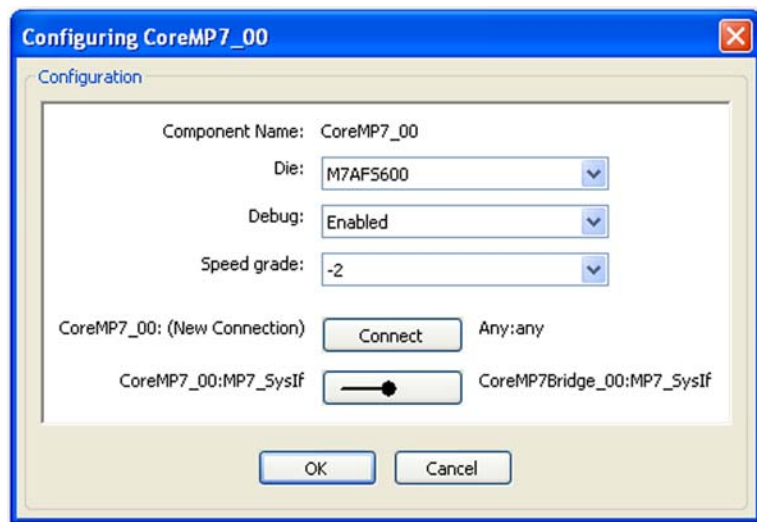


Figure 5-13. Configuring CoreMP7 Dialog Box

2. From the drop-down menu for **Die**, select **M7A3PE600**. This is the device populated on the CoreMP7 Evaluation Board.
3. Click **OK**.

Within the Configuring CoreMP7 dialog box, you can disable the JTAG debug interface, which allows you to select the "fast" version of CoreMP7.

4. Invoke the Configuration dialog boxes for the **CoreMP7Bridge** and **CoreUARTapb** components.
5. In the **Device family** field, select **ProASIC3E** from the drop-down menu and click **OK**.
6. Select the **Generate** tab within the design manager.
7. Select the HDL language preference. This tutorial is based on Verilog, so make sure **Verilog** is selected.
8. Click the **Save & Generate** button.

Before exiting CoreConsole, wait until both progress bars have reached 100% (Figure 5-14). This process can take up to 45 seconds, depending on system complexity and PC resources.

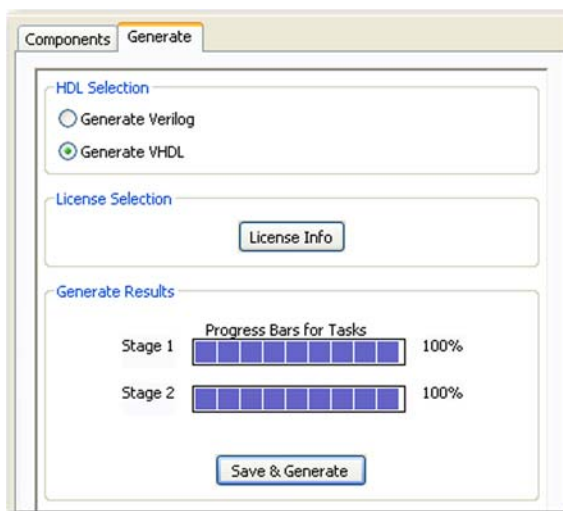


Figure 5-14. CoreConsole Generate Tab

Once the generation phase has successfully completed, you can import your CoreConsole project (i.e., HDL source files and the ARM7 black box) into Actel Libero IDE.



# Actel Libero IDE v7.1

## Step 1 – Create a New Project

This step uses the Libero IDE HDL Editor to enter an Actel CoreMP7 Verilog design.

### To create the Libero IDE Verilog project:

1. Double-click the **Libero IDE** icon on your desktop to start the program.
2. From the **File** menu, select **New Project**. This displays the New Project Wizard, shown in Figure 5-15.

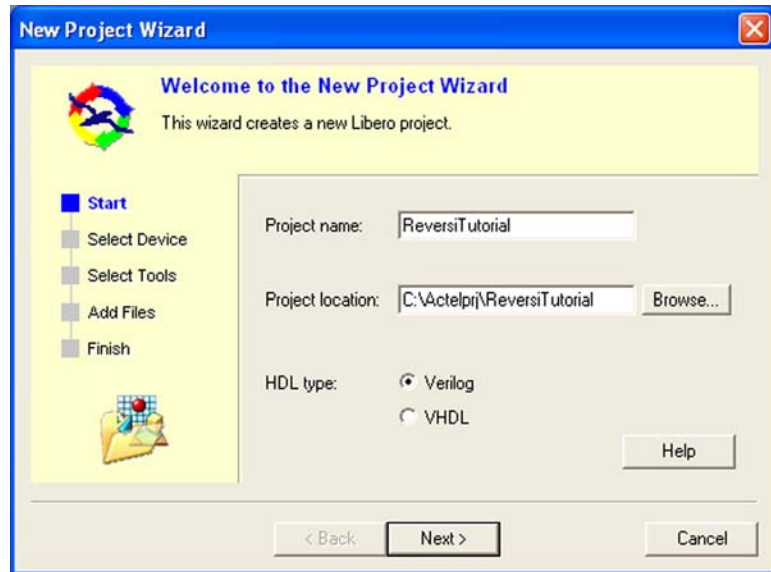


Figure 5-15. New Project Wizard in Libero IDE

3. Enter your **Project name**. For this tutorial, name your project “ReversiTutorial”.
4. Select your **HDL type**. For this tutorial select **Verilog**.
5. If necessary, in the **Project location** field, click **Browse** to navigate to *C:\Actelprj*. Click **Next** to continue.

6. Select your project **Family**, **Die**, and **Package**. For this tutorial, select **ProASIC3E**, the **M7A3PE600** die, and **484 FBGA** for the package (Figure 5-16).

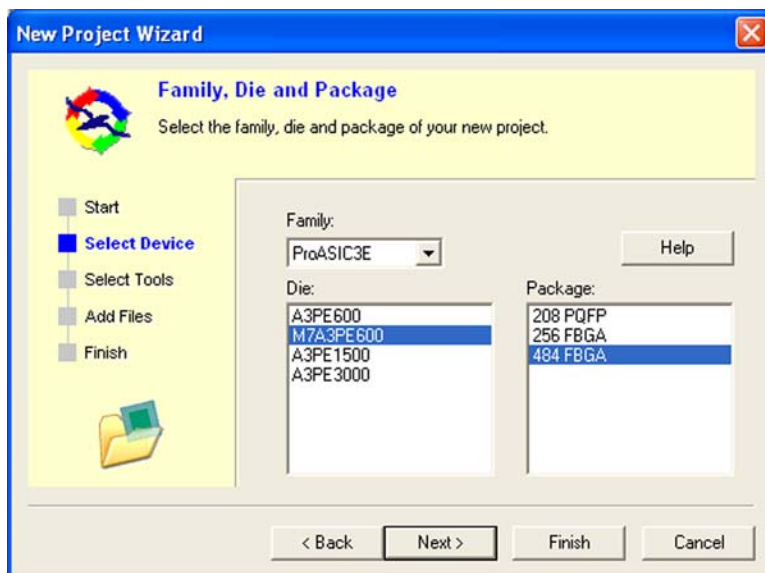


Figure 5-16. Select ProASIC3E, M7A3PE600, and 484 FBGA

7. Click **Next** to select integrated tools in the New Project Wizard (Figure 5-17).



Figure 5-17. Selecting Integrated Tools in the Libero IDE New Project Wizard

8. Click the **Restore Defaults** button to use the default tools included with Libero IDE.

9. Click the **Add** button to add a different Synthesis, Simulation, or Stimulus tool. If you wish to add a tool, Libero IDE opens the **Add Profile** dialog box (Figure 5-18).

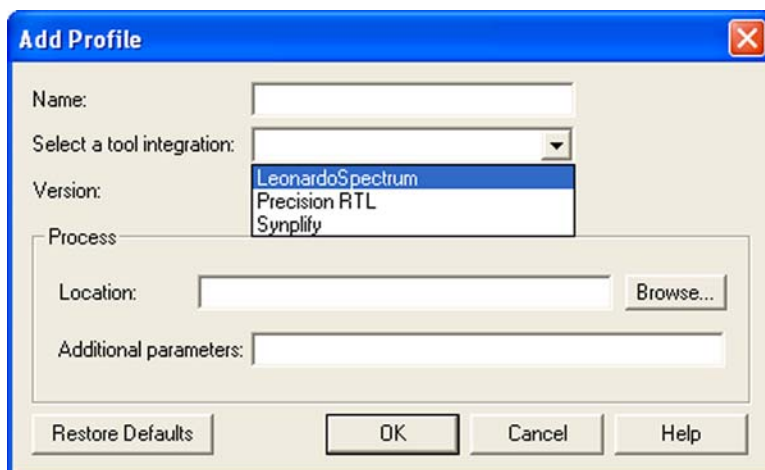


Figure 5-18. Add Profile Dialog Box in Libero IDE

10. Name your profile, select a tool from the list of Libero IDE supported tools, and **Browse** to the location of your tool. Click **OK** to return to the New Project Wizard.
11. After you have selected your tools, click **Next** to continue.

12. Click **Add Files** in the New Project Wizard to add existing project design files. Include any ACTgen cores, CoreConsole Projects, Block Symbol, Schematic, Verilog Source, Implementation, or Stimulus files (Figure 5-19).

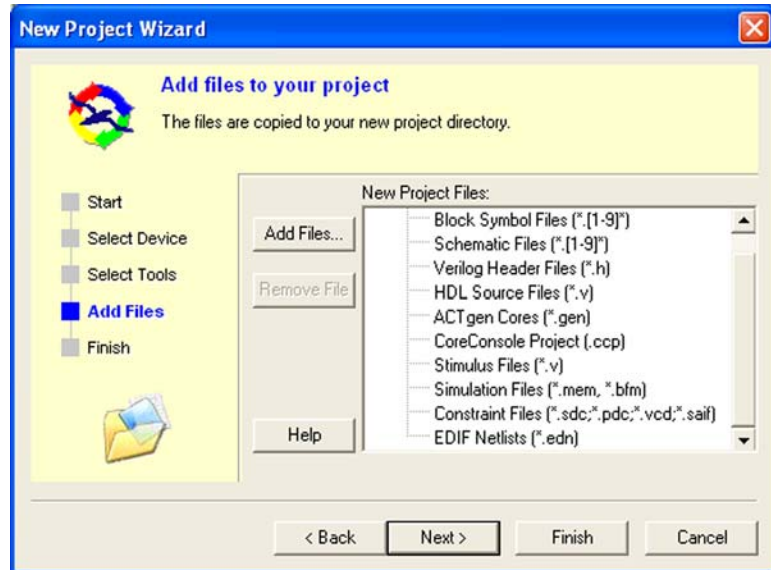


Figure 5-19. Add Files in the Libero IDE New Project Wizard

13. Select the CoreConsole Project file type and click **Add Files**. Browse to your CoreConsole Project created earlier (assuming default installation it will be located in the *C:\CoreConsole\Libero IDE Export\TutorialMP7\* directory) and select the *TutorialMP7.ccp* file, then click **Add**.

The CoreConsole project will be displayed in the project wizard (Figure 5-20). You can add as many files as you like this way. For this project, only the CoreConsole file will be imported with this method.

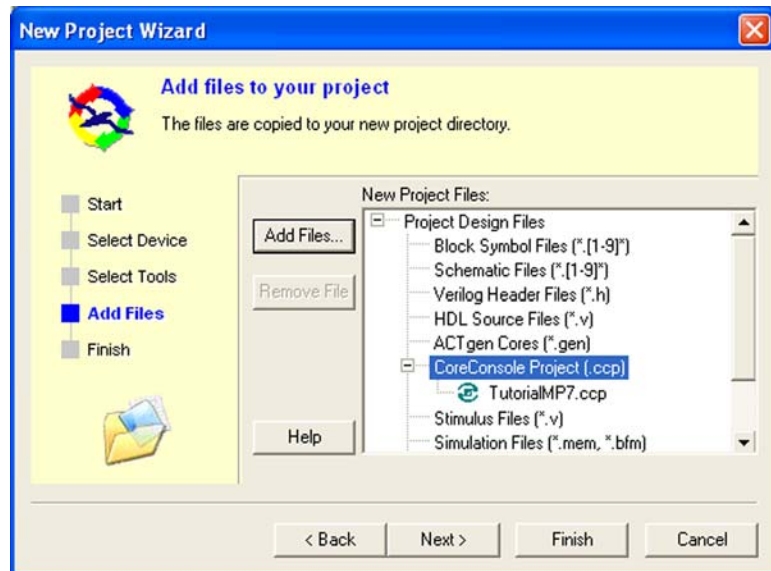


Figure 5-20. CoreConsole Project Added to the New Libero IDE Project

14. Review your project information. Click **Finish** to close the Wizard and create your new project (Figure 5-21). Click **Back** to return to any step of the Wizard and correct information in your project.

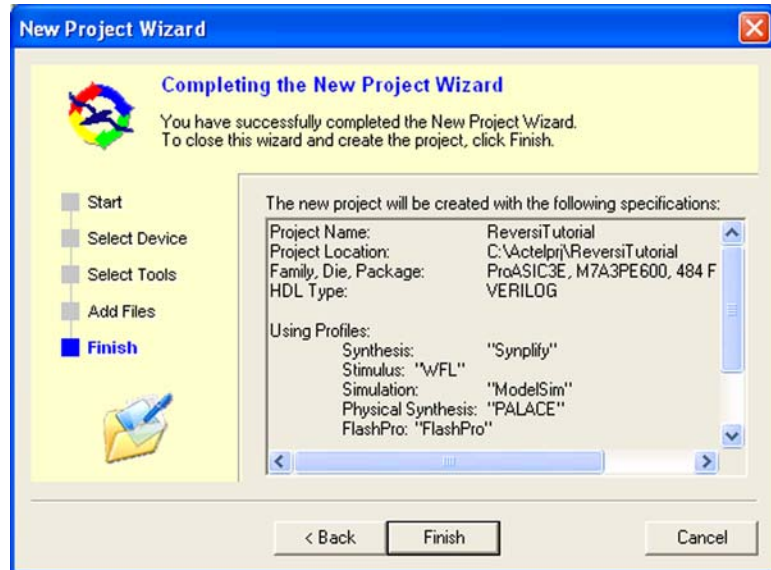


Figure 5-21. Summary in New Project Wizard

Your Libero IDE project exists, but you must add some top-level code or source to the project—such as a schematic, SmartGen core, or Verilog module—before you can run synthesis.

**To add a Verilog top-level HDL file to the project:**

1. From the **File** menu, select **Import Files**. Navigate to the `\Tutorial\FPGA` directory on the CD-ROM included with the CoreMP7 Development Kit and select the `TutorialTop.v` file. Click **Import**.
2. Click on the **Design Hierarchy** tab located at the bottom of the Libero IDE Design File Manager, as shown in Figure 5-22.

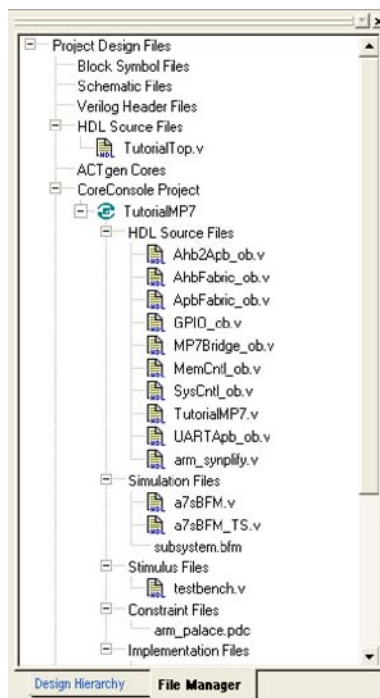


Figure 5-22. Libero IDE Design File Manager



3. From the **Design Hierarchy**, right-click the *TutorialTop.v* file and select **Set as Root** (see [Figure 5-23](#)). This sets the project's top-level file to the module contained in the source file just imported.

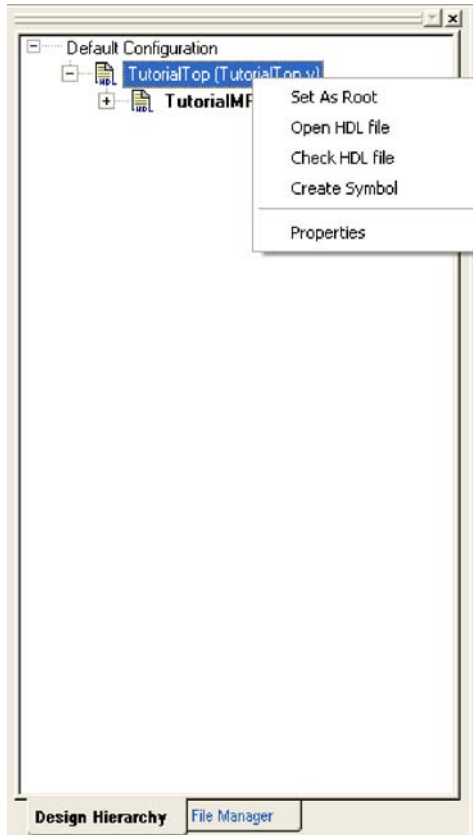


Figure 5-23. Libero IDE Design Hierarchy Viewer

4. Click the **File Manager** tab to return to the Libero IDE Design File Manager.

## Step 2 – Perform Pre-Synthesis Simulation

The next step is simulating the RTL description of the design. First, you must create a top-level testbench to provide a stimulus for the design. Keep in mind you will not be able to simulate CoreMP7 but will rely on the Bus Functional Model (BFM), which is a cycle-accurate model of the embedded ARM7TDMI-S processor.

### To create the top-level testbench:

1. From the **File** menu, select **File** and then **New**. This opens the New dialog box, shown in Figure 5-24.

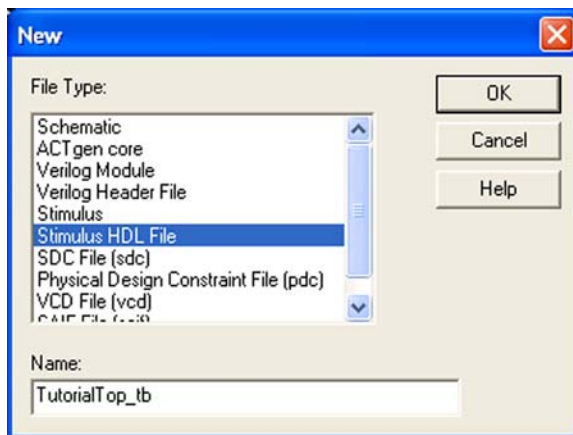


Figure 5-24. New File Dialog Box

2. Select **Stimulus HDL File** in the File Type field, enter “TutorialTop\_tb” in the **Name** field, and click **OK**. The HDL Editor opens. Enter the following text in a Verilog file, or if this document is open in an electronic form, copy and paste it from here. Alternatively, this file is provided in the `\Tutorial\FPGA` folder on the CD-ROM, and you can import the testbench as you did with the top-level source file.

```
`timescale 1ns/100ps

module testbench();
    parameter SYSCLK_PERIOD = 100; // 10MHz

    reg SYSCLK;
    reg NSYSRESET;

    wire ICE_nSRST;
```

```
pullup (weak1) p1 (ICE_nSRST);

initial
begin
    SYSCLK = 1'b0;
    NSYSRESET = 1'b0;

    // Release system reset
    #(SYSCLK_PERIOD * 4)
    NSYSRESET = 1'b1;

    #(SYSCLK_PERIOD * 100000);

    $stop;
end

// SYSCLK signal
always @(SYSCLK)
    #(SYSCLK_PERIOD / 2)
    SYSCLK <= !SYSCLK;

// Instantiate module to test
TutorialTop TutorialTop_0 (
    .SYSCLK(SYSCLK),
    .NSYSRESET(NSYSRESET),
    .RemapDefault(1'b0),
    .HIGH(),
    .LOW(),
    .FLASH_BYTEN(),
    .FLASH_CSN(),
    .FLASH_OEN(),
    .FLASH_RPN(),
    .FLASH_WEN(),
    .SRAM_ADSC(),
    .SRAM_ADSP(),
    .SRAM_ADV(),
    .SRAM_BYTEN(),
```

```
.SRAM_BYTE_WEN(),
.SRAM_CLK(),
.SRAM_CSN(),
.SRAM_GLOBAL_WEN(),
.SRAM_OEN(),
.SRAM_PWRDWN(),
.MEM_ADDR(),
.MEM_DATA(),
.SW(8'b0),
.LED(),
.RX0(1'b0),
.TX0(),
.ICE_nTRST(),
.ICE_TCK(1'b0),
.ICE_TDI(1'b0),
.ICE_TMS(1'b1),
.ICE_VTref(),
.ICE_TDO(),
.ICE_RTCK(),
.ICE_nSRST(ICE_nSRST),
.ICE_DBGACK(),
.ICE_DBGRQ()
);

endmodule
```

3. From the **File** menu, click **Save**. The testbench file now appears in the Libero IDE Design Manager. Libero IDE lists *TutorialTop\_tb.v* under Stimulus Files, as shown in [Figure 5-25](#).

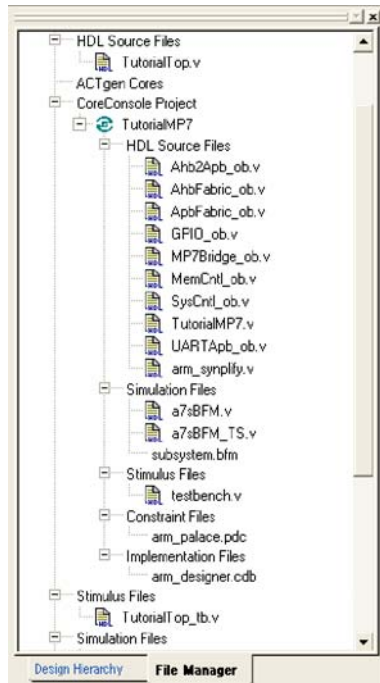


Figure 5-25. Libero IDE File Manger with Stimulus

4. Check the HDL in the file before you continue. Under the File Manager tab (Figure 5-26), right-click *TutorialTop\_tb.v* and select **Check HDL**. This checks the syntax of *TutorialTop\_tb.v*. Before moving to the next section, modify the code if you find any errors.

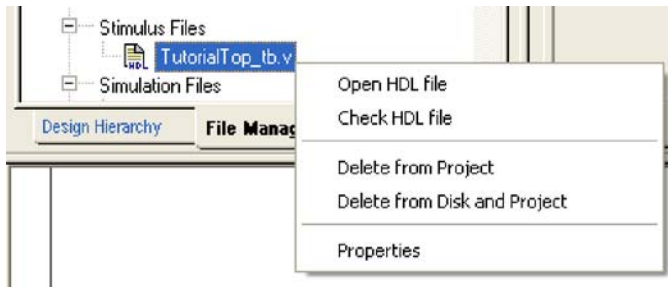


Figure 5-26. Check HDL Option from File Manager

**To perform a pre-synthesis simulation:**

1. From the **Design Hierarchy** tab, right-click the *TutorialTop.v* file and select **Organize Stimulus**, as shown in Figure 5-27.

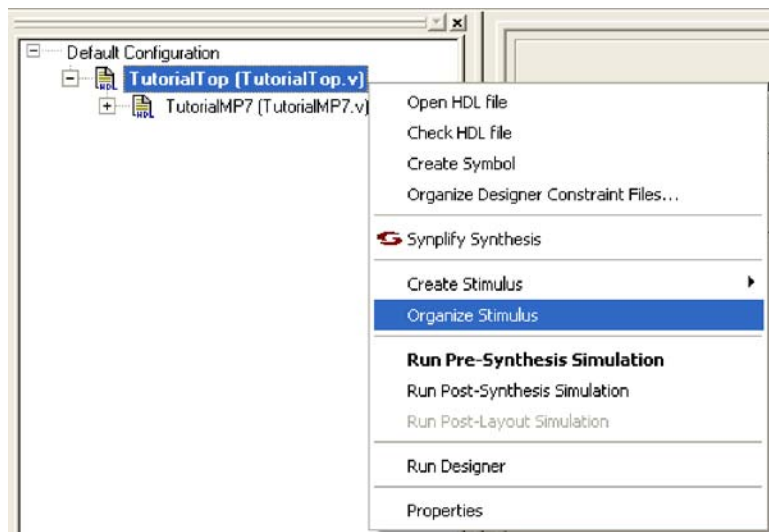


Figure 5-27. Design Hierarchy Context Menu

The Organize Stimulus dialog box appears, as shown in [Figure 5-28](#).

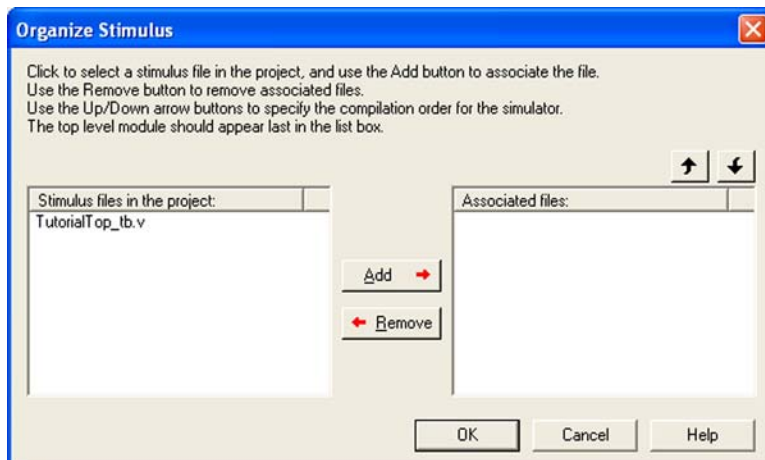


Figure 5-28. Organize Stimulus Dialog Box

2. Select *TutorialTop\_tb.v* from the Stimulus files in the project list box and click **Add** to add the file to the Associated files list.
3. Click **OK**. Stimulus icons in the Design Flow window turn green to notify you that there is a testbench file associated with the project.
4. Right-click the **Simulation** icon in the Libero IDE Design Flow window and select **Options**, as shown in [Figure 5-29](#).

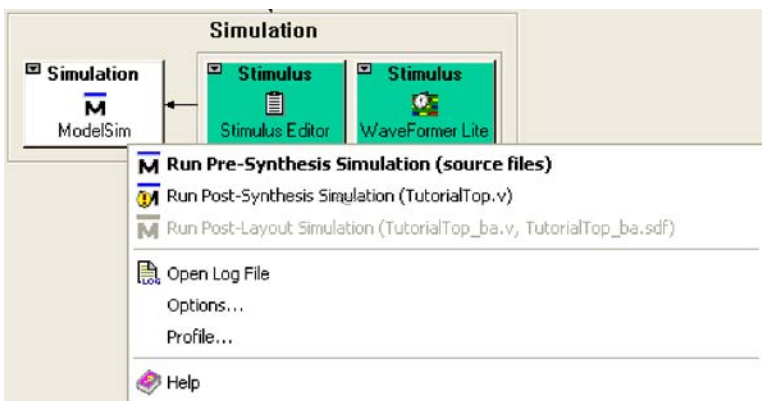


Figure 5-29. Simulation Context Menu

The Project Settings: Simulation options window appears (Figure 5-30).

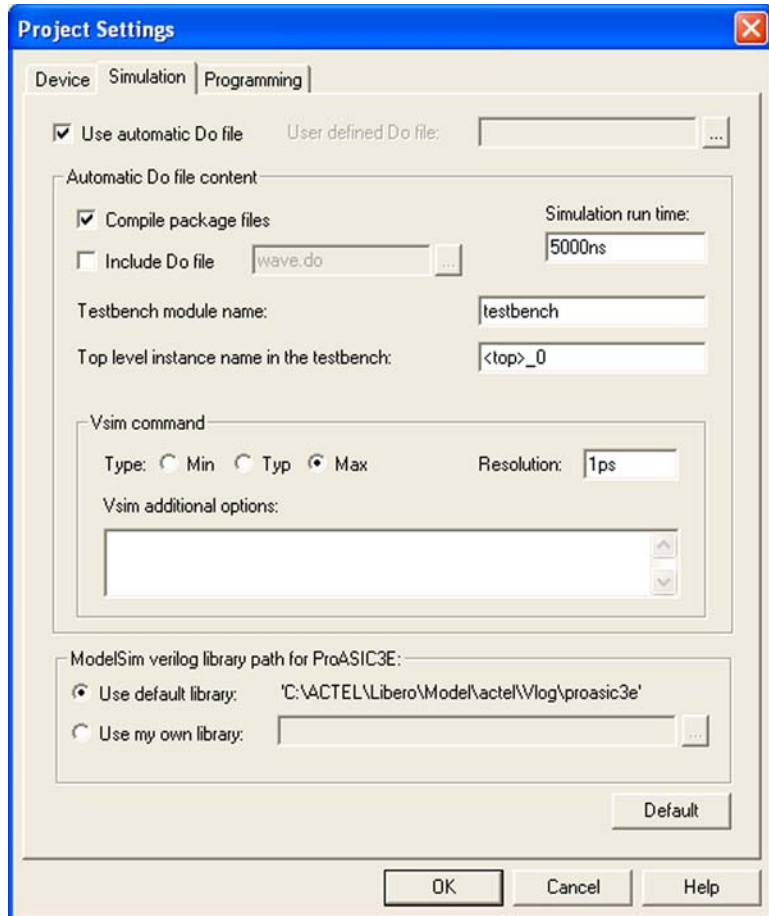


Figure 5-30. Simulation Project Settings

5. Change the **Simulation run time** from **1000ns** to **5000ns**, as shown in Figure 5-30, and click **OK**. Changing the run time allows the default BFM test scripts to complete without having to invoke additional run time from within the ModelSim simulator.



6. Click the **Simulation** icon in the Design Flow window, or right-click *TutorialTop.v* in the **Design Hierarchy** and select **Run Pre-Synthesis Simulation**, as shown in [Figure 5-31](#).

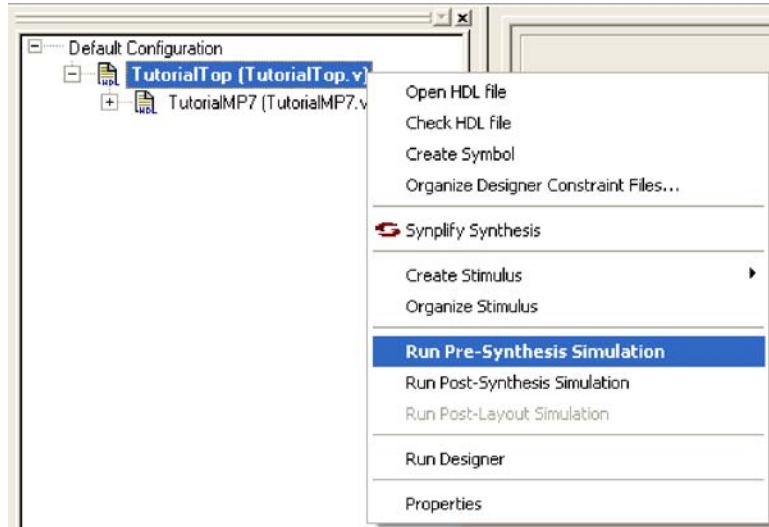
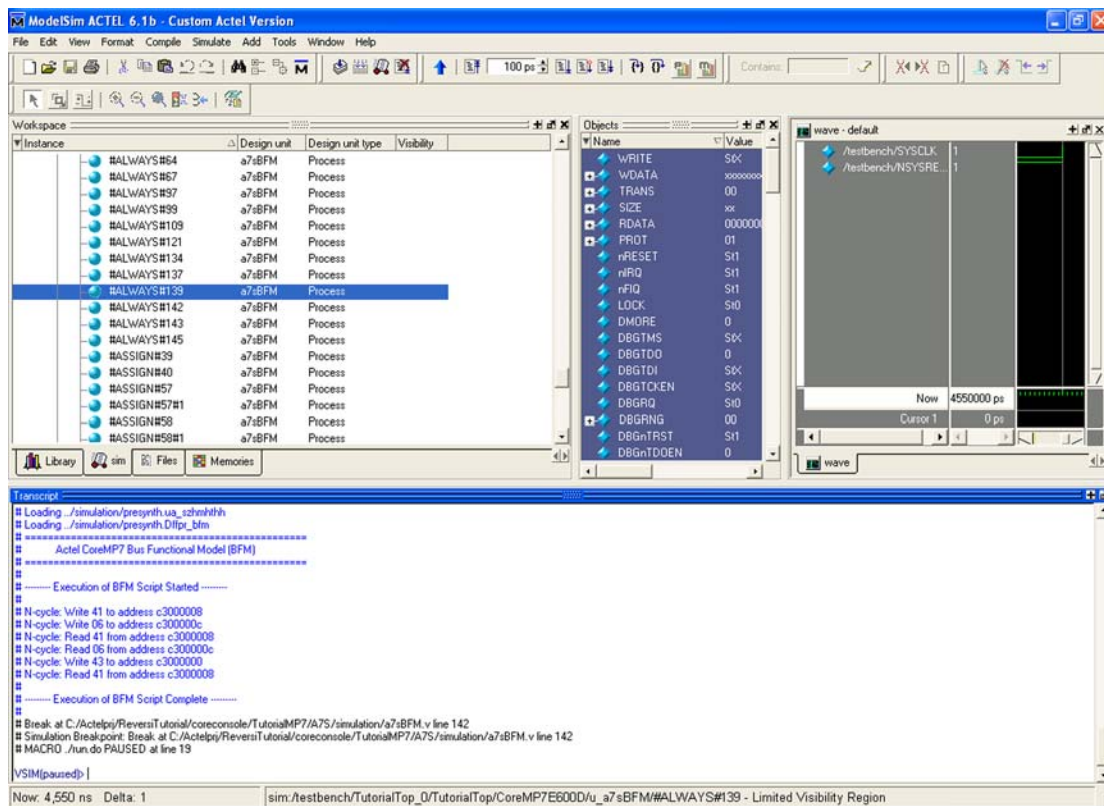


Figure 5-31. Running Pre-Synthesis Simulation from the *TutorialTop.v* Context Menu

CoreMP7 Development Kit User's Guide



Once the compilation completes, the simulator simulates for the default time period of 5000 ns, and a wave window, shown in Figure 5-33, opens to display the simulation results. The default wave window currently contains only the SYSCLK and NSYSRESET signals. You will expand this shortly. The results of the default BFM scripts can also be viewed in the ModelSim log window (as shown in Figure 5-32 on page 66). The successful reads and writes confirm that CoreMP7 is connected properly from the top level down to the various busses.

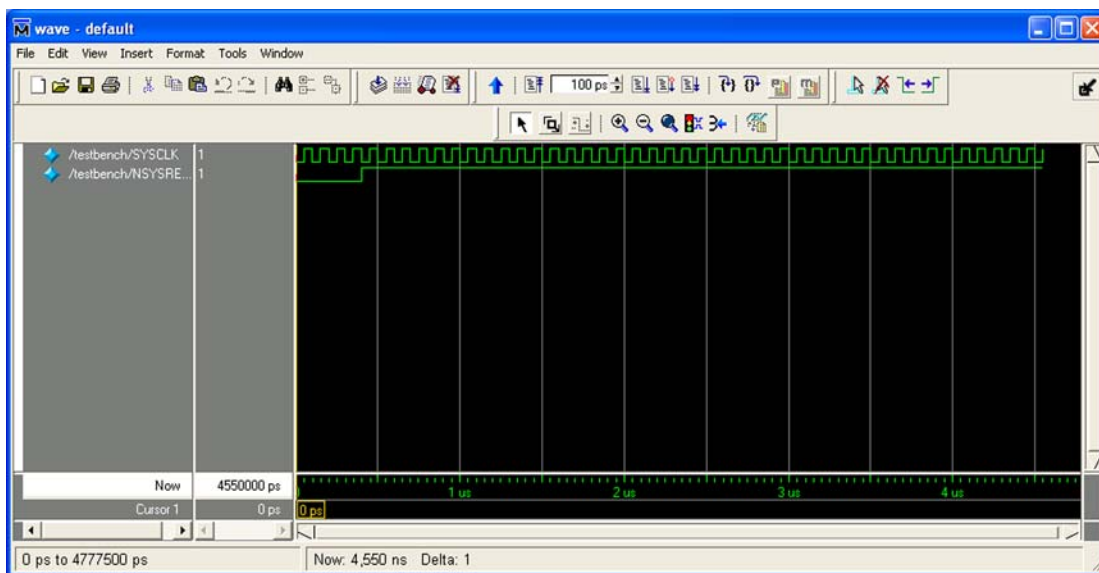


Figure 5-33. ModelSim Wave Window

7. To add the CoreMP7 signals to the ModelSim Wave window, navigate to the **CoreMP7E600D** instance in the ModelSim workspace, which can be found under the following hierarchy (shown in Figure 5-34):

TutorialTop\_0 > TutorialTop > CoreMP7E600D

Drag the CoreMP7E600D instantiation to the ModelSim wave window. The instantiation's signals will appear.

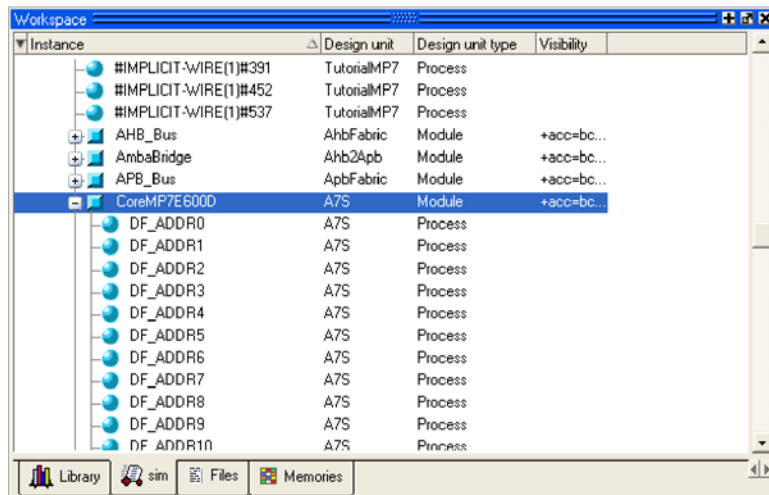


Figure 5-34. ModelSim Workspace Window

8. In the ModelSim Transcript (Log) window, type **restart**. This will bring up the Restart dialog box (shown in Figure 5-35). Click the **Restart** button. This will reset the simulation to the beginning so that logging of the CoreMP7 signals occurs.

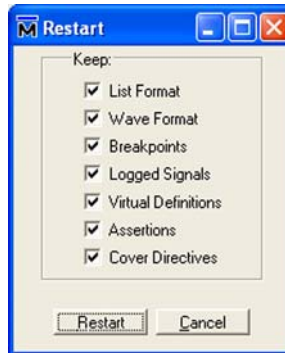


Figure 5-35. ModelSim Restart Dialog Box

9. Within the ModelSim Transcript window, type **run-all**. This will re-run the default testbench and BFM scripts. The results should be the same as the previous run. Notice the ModelSim Wave window—the CoreMP7 signals now have waveforms associated with them.
10. Undock the ModelSim Wave window and maximize it, then select **Zoom Full** from the **View > Zoom** menu. Examining the ADDR, RDATA, WDATA, WRITE, SIZE, SYSCLK, nRESET, and NSYSRESET signals allows the re-creation of the BFM scripts and validates the results. The corresponding signals have been grouped together and are shown in Figure 5-36.

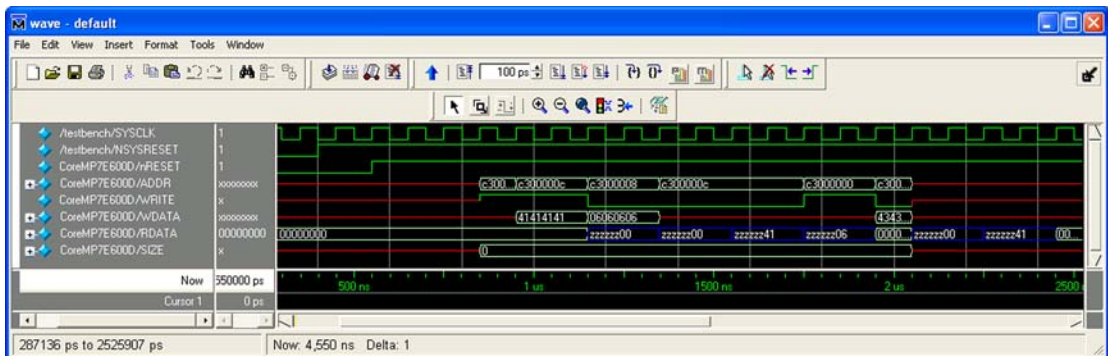


Figure 5-36. ModelSim Wave Window with CoreMP7 Signals

11. In the ModelSim window, select **File** then **Quit** to close the window.

## Step 3 – Synthesize the Design in Synplify

The next step is to generate an EDIF netlist by synthesizing the design in Synplify. For HDL designs, Libero IDE launches and loads the Synplify Synplicity synthesizer with the appropriate design files.

### To create an EDIF netlist for the design using Synplify:

1. In Libero IDE, click the **Synplify Synthesis** icon in the Design Flow window, or right-click the *TutorialTop.v* file in the Design Hierarchy and select **Synplify Synthesis**. This launches the Synplify synthesis tool with the appropriate design files, as shown in [Figure 5-37](#).

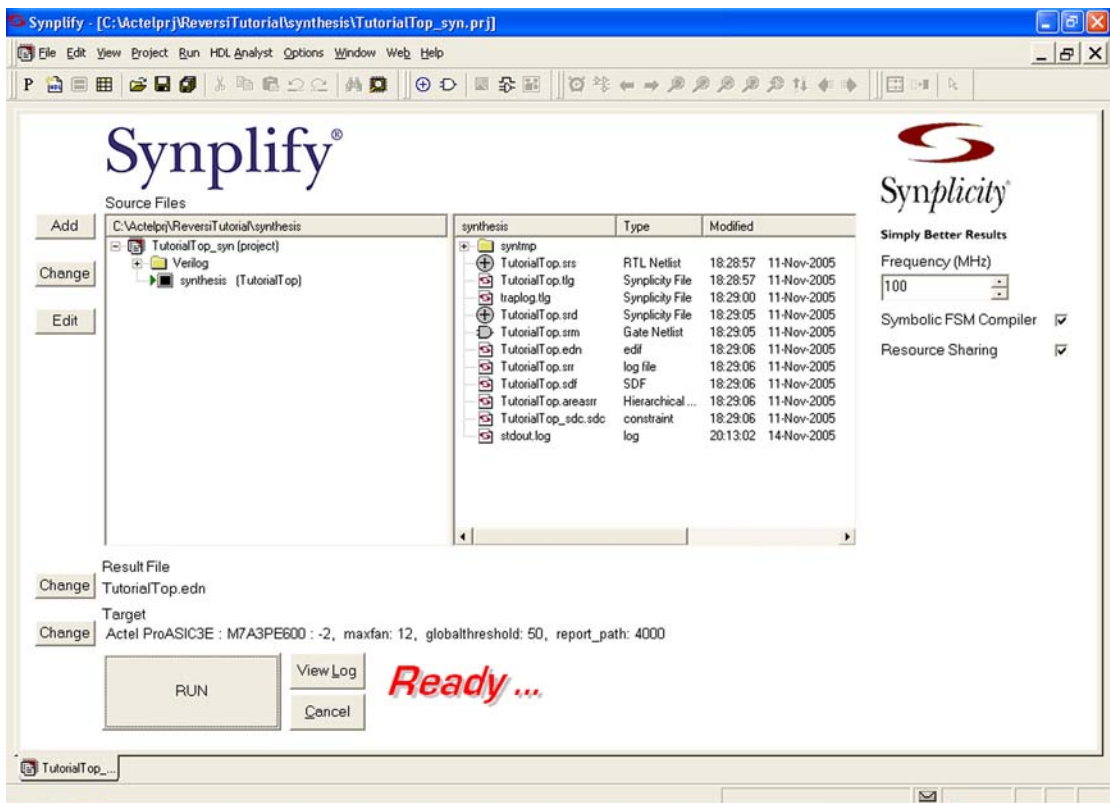


Figure 5-37. Synplify Synthesis Main Window

2. From the **Project** menu, select **Implementation Options**. This displays the options for the Implementation dialog box, as shown in Figure 5-38 for the M7A3PE600.

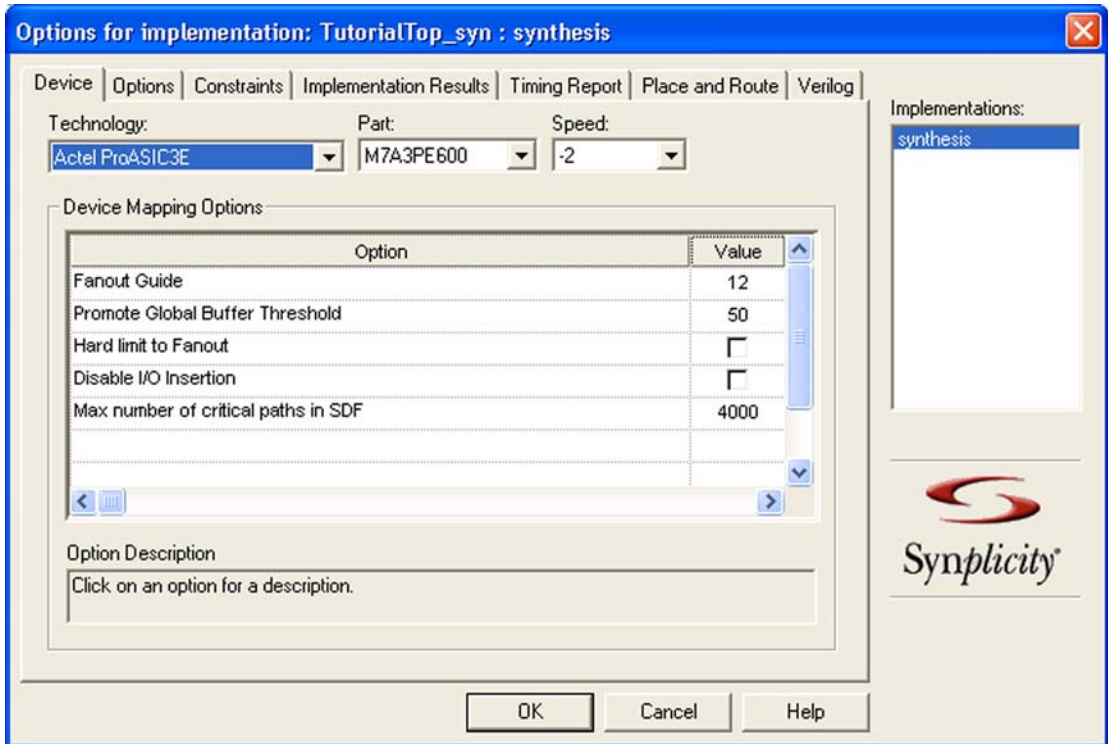


Figure 5-38. Implementation Options Dialog Box for the M7A3PE600

3. Set (confirm) the following in the dialog box:
  - **Technology:** Actel ProASIC3E (set automatically by Libero IDE)
  - **Part:** M7A3PE600
  - **Fanout Guide:** 12 (default)
  - **Hard limit to Fanout:** Off (default)
4. Accept the default values for each of the other tabs in the Options for Implementation dialog box and click **OK**.
5. In the Synplify main window, click **Run**. Synplify compiles and synthesizes the design into a netlist called *TutorialTop.edn*. This netlist is then automatically translated by Libero IDE into a Verilog netlist called *TutorialTop.v*.

The resulting EDIF and Verilog files are displayed under Implementation Files in the Libero IDE File Manager.

6. If any errors appear after you click the **Run** button, edit the file using the Synplify editor. To edit the file, double-click the file name in the Synplicity window. Any changes made here are saved to the original design file in Libero IDE.
7. Save and close Synplify. From the **File** menu, click **Exit** to close Synplify. Click **Yes** to save any settings made to the *TutorialTop\_syn.prj* file in Synplify.

## Step 4 – Perform Post-Synthesis Simulation

The next step is simulating the Verilog netlist of the design using the Verilog testbench created in “[Step 2 – Perform Pre-Synthesis Simulation](#)” on page 58.

1. Click the **Simulation** icon in the Libero IDE Design Flow window, or right-click the *TutorialTop.v* file in the Design Hierarchy tab and select **Run Post-Synthesis Simulation**. This launches the ModelSim simulator, which compiles the source files and testbench.

Once the compilation completes, the simulator runs for 5000 ns and the Wave window displays the simulation results. Verify that the read/write results of the executed BFM scripts are correct.

2. Follow the same sequence as in “[Step 2 – Perform Pre-Synthesis Simulation](#)” on page 58, beginning with step 7, to add and verify the internal CoreMP7 signals of the BFM.
3. Scroll in the Wave window to verify that the CoreMP7 system works correctly. Use the zoom buttons to zoom in and out as necessary.

## Step 5 – Implementing the Design with Actel Designer

After creating and simulating the design, the next phase is implementing the design using the Actel Designer software (performing place-and-route).

1. From the Libero IDE **File** menu, select **Import Files**. Navigate to the *\Tutorial\FPGA* directory on the Development Kit CD-ROM and select the *TutorialTop\_PinConstraints.pdc* file. It might be necessary to change the file type to PDC to view this file. Click **Import**. The file will now be listed under Constraint Files in the Libero IDE File Manager.



- Click the Designer **Place & Route** button in the Libero IDE Design Flow window, or right-click *TutorialTop.v* in the **Design Hierarchy** tab and select **Run Designer**. Designer reads in the design file (Figure 5-39).

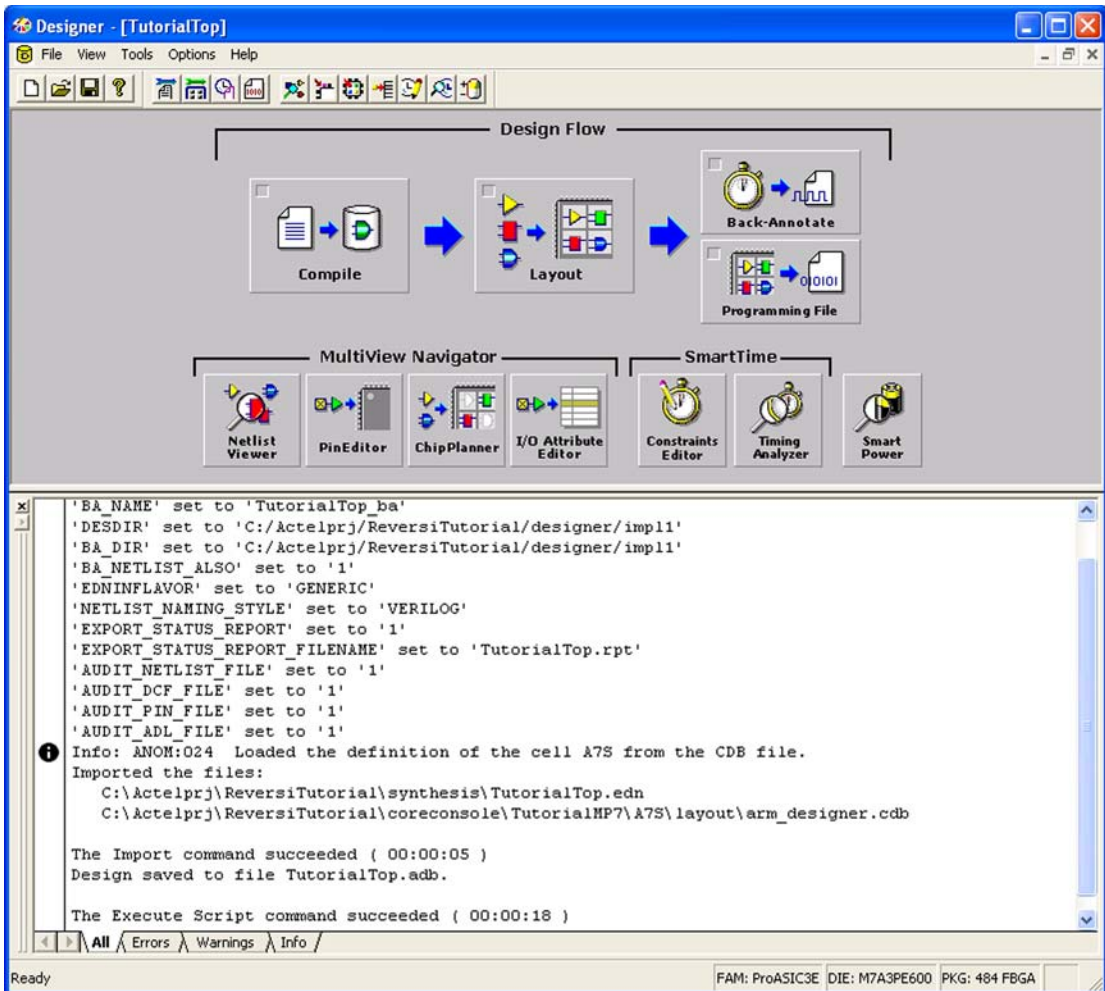


Figure 5-39. Actel Designer GUI

The Device Selection Wizard opens (Figure 5-40).

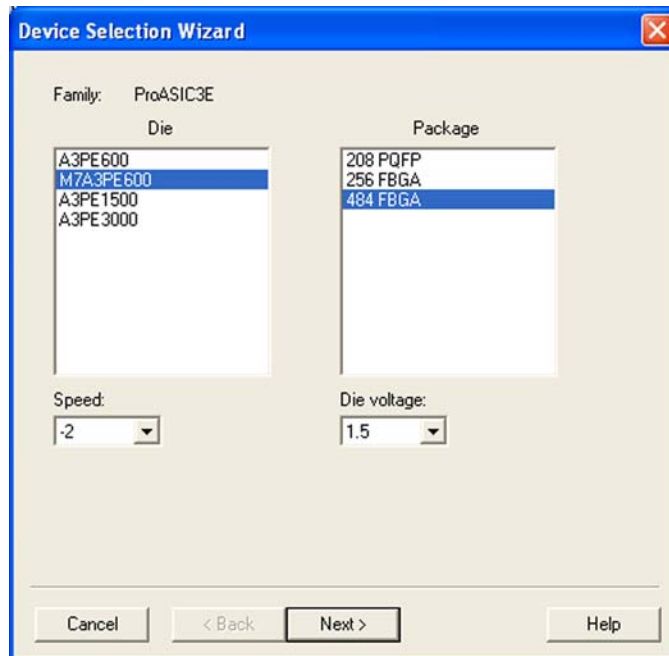


Figure 5-40. Device Selection Wizard for M7A3PE600

3. Select **M7A3PE600** in the **Die** field and **484 FBGA** in the **Package** field. Accept the default **Speed** grade and **Die voltage** and click **Next**.
4. Use the default I/O settings and click **Next**.
5. Use the default **Junction Temperature** and **Voltage** setup and click **Finish**.
6. From the Designer **File** menu, select **Import Source Files**.

This displays the Import Source Files dialog box (Figure 5-41). Click the **Add** button, navigate to the Libero IDE project's `\constraint` directory, and add the *TutorialTop\_PinConstraints.pdc* file (it may be necessary to change the file type to view the PDC file). Once the file has been added, click **OK**.

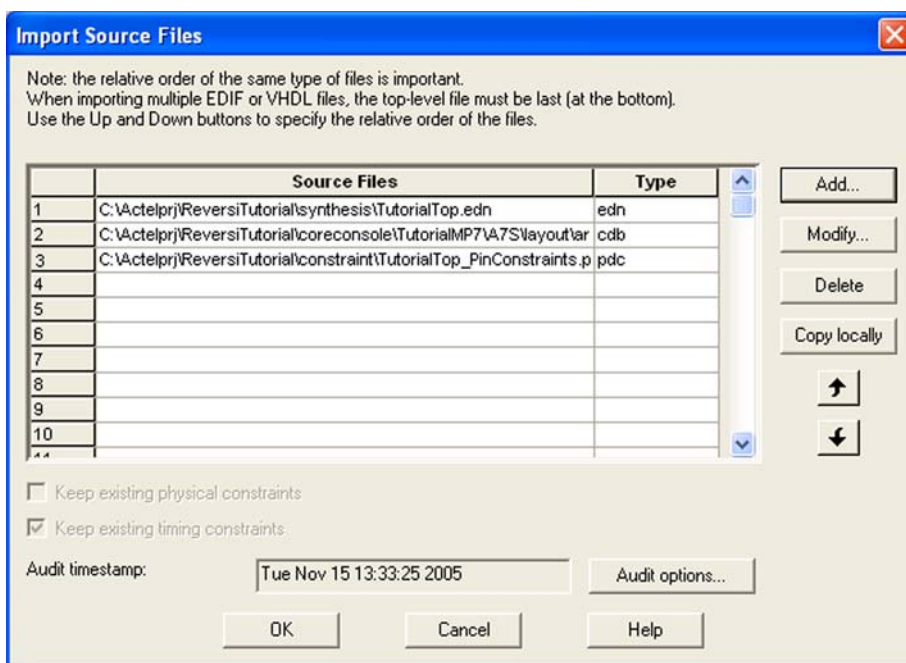


Figure 5-41. Import Source Files Dialog Box in Designer

7. When the EDIF Import Options dialog box appears, as in [Figure 5-42](#), click **OK**. This will re-import the source files (all three of them) into Designer.

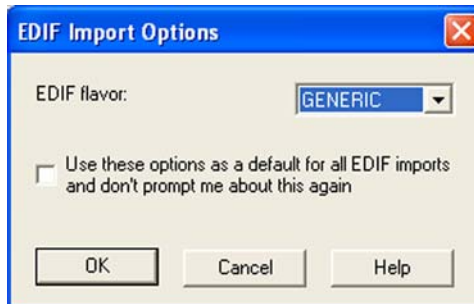


Figure 5-42. EDIF Import Options Dialog in Designer

8. Click the **Compile** icon. Leave the default Compile settings ([Figure 5-43](#)) and click **OK**.

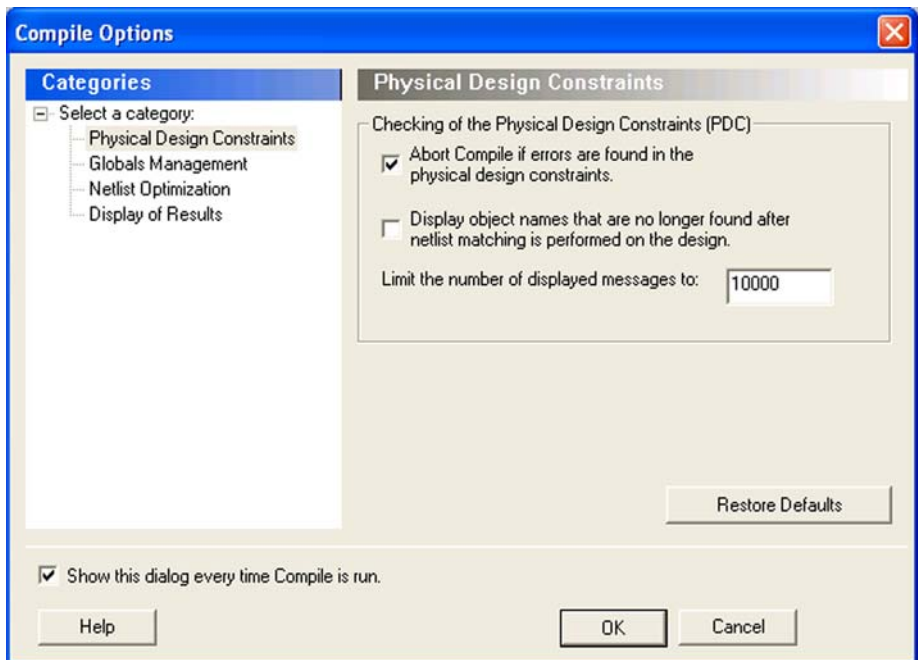


Figure 5-43. Compile Options Window

Designer compiles the design and shows the utilization of the selected device. Also, note that the Compile icon in Designer turns green once the compile has successfully completed.

9. Once the design compiles successfully, use the I/O Attribute Editor tool to verify the pin assignments imported from the pin constraints file. Alternatively, the I/O Attribute Editor can be used to create pin assignments. Click the **I/O Attribute Editor** to open the tool. It opens in the MultiView Navigator user interface, as shown in Figure 5-44.

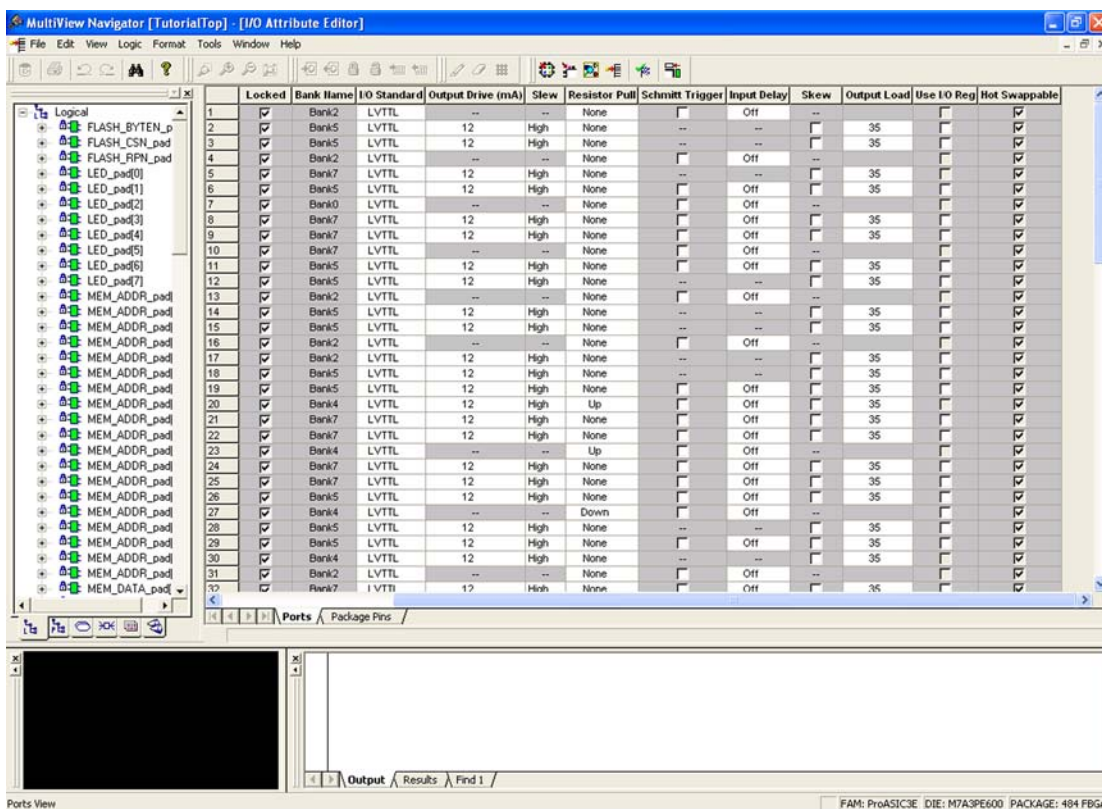


Figure 5-44. I/O Attribute Editor in MultiView Navigator

10. Verify that the correct pins have been assigned to all of the signals. If changes are made, select **Commit** from the **File** menu and then close the I/O Attribute Editor.

*Optional:* After successfully compiling the design, use the Designer Tools to view the pre-layout static timing analysis with SmartTime, set the timing constraints in SmartTime, analyze the

static and dynamic power with SmartPower, and use the ChipPlanner to assign modules. Click the appropriate icons to access these tools.

For more information on these functions, refer to the Designer or Libero IDE online help.

11. In Designer, click **Layout**. This opens the Layout Options dialog box, shown in [Figure 5-45](#).

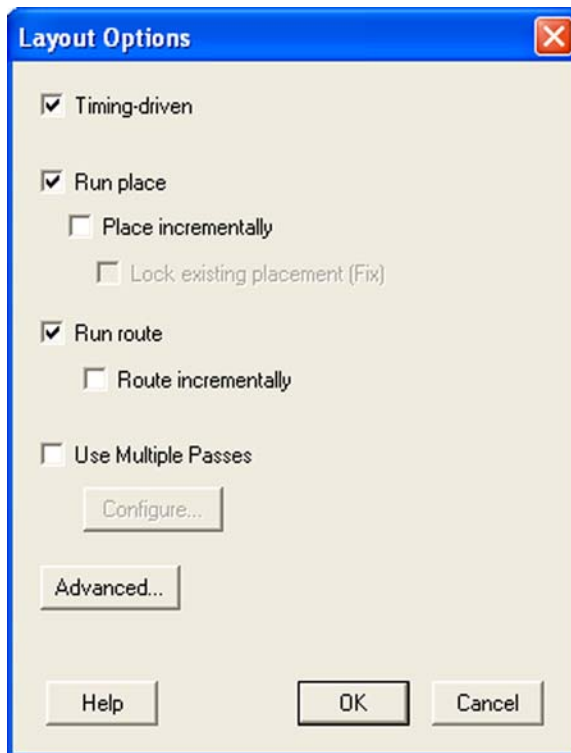


Figure 5-45. Layout Options Dialog Box

12. Click **OK** to accept the default layout options. This runs place-and-route on the design. The Layout icon turns green to indicate that the layout has successfully completed.

13. From Designer, click **Back-Annotate** in the Design Flow window. This opens the Back-Annotate dialog box, shown in Figure 5-46.

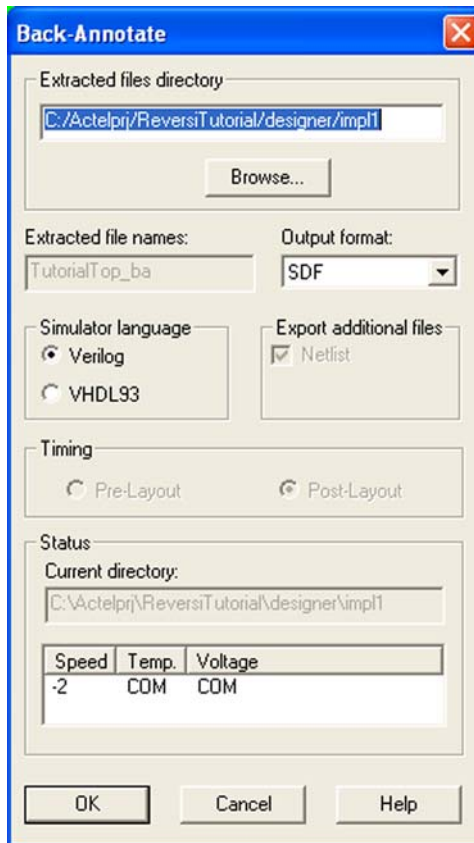


Figure 5-46. Back-Annotate Dialog Box

14. Accept the default settings and click **OK**. The Back-Annotate icon turns green.
  15. Save and close Designer. From the **File** menu, click **Exit**. Click **Yes** to save the design before closing Designer. Designer saves all the design information in an ADB file.
- The file *TutorialTop.adb* appears under the Designer Files tab of the File Manager. To re-open the file, right-click it and select **Open in Designer**.

## Step 6 – Perform Timing Simulation with Back-Annotated Timing

After completing place-and-route and back-annotation of the design, perform a timing simulation with the ModelSim HDL simulator.

### **To perform a timing simulation:**

1. Click the **Simulation** icon in the Libero IDE Design Flow window, or right-click the *TutorialTop.v* file under the **Design Hierarchy** tab and select **Run Post-Layout Simulation**.
2. This launches the ModelSim Simulator, which compiles the back-annotated Verilog netlist file and testbench.

Once the compilation completes, the simulator runs for 5000 ns and the Wave window displays the simulation results. Verify that the read/write results of the executed BFM scripts are correct.

3. Follow the same sequence as in “[Step 2 – Perform Pre-Synthesis Simulation](#)” on page 58, beginning with step 7, to add and verify the internal CoreMP7 signals of the BFM.
4. Scroll in the Wave window to verify that the CoreMP7 system works correctly. Use the zoom buttons to zoom in and out as necessary.



## Step 7 – Generating the Programming File

1. Open the *TutorialTop.adb* file in Designer and click the **Programming File** button in the Design Flow window, which opens the FlashPoint window (Figure 5-47).

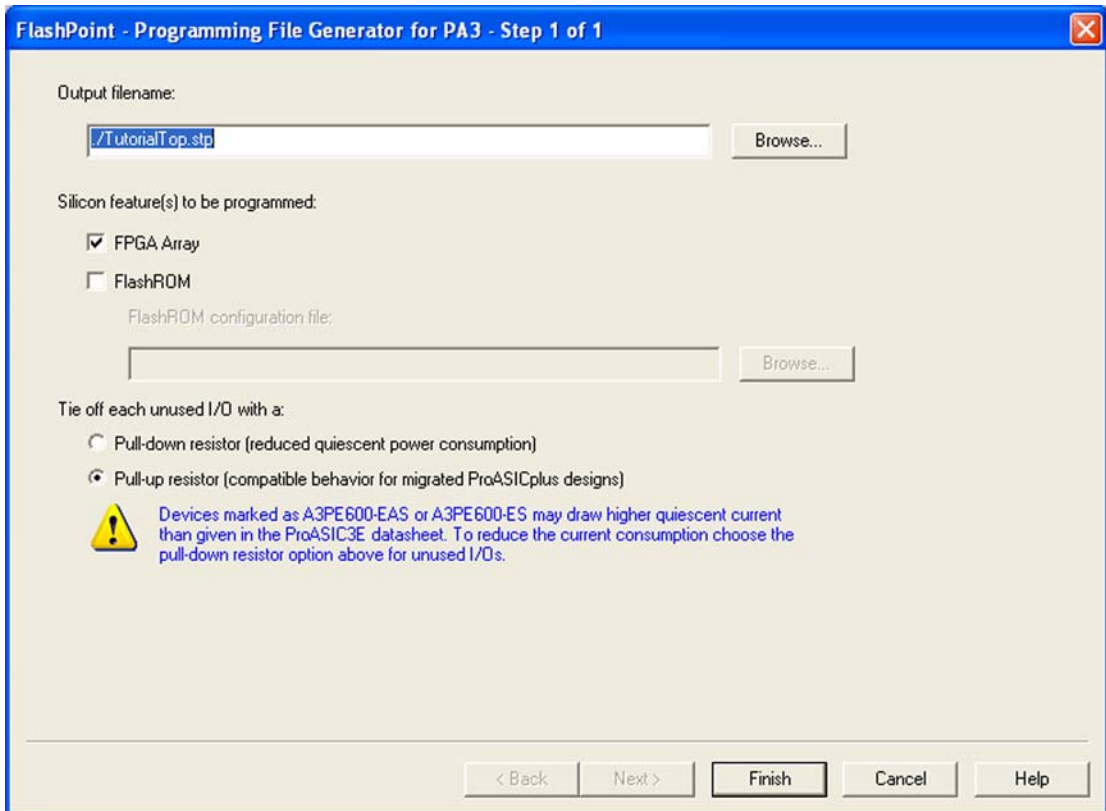


Figure 5-47. Flash Point Dialog Box

2. Click **Finish**. The programming file is generated and saved in the *\designer\impl1* folder. The **Programming File** icon in the Designer Design Flow window should now be green, indicating that programming file generation has been successfully completed.
3. Save and close Designer. From the **File** menu, click **Exit**. Click **Yes** to save the design before closing Designer.

## Step 8 – Programming the Device

After generating the programming file, program the device using an Actel FlashPro3 programmer. Before performing any action with the FlashPro3 programmer, it must be properly set up. Connect the FlashPro3 USB cable to your PC USB port, connect the ribbon cable to the programming header on the target board, and turn on the power switch on the board.

1. Click the **FlashPro Programming** button in the Libero IDE Design Flow window, or right-click *TutorialTop.v* under the Design Hierarchy tab and select **Run FlashPro**.

FlashPro opens (Figure 5-48).

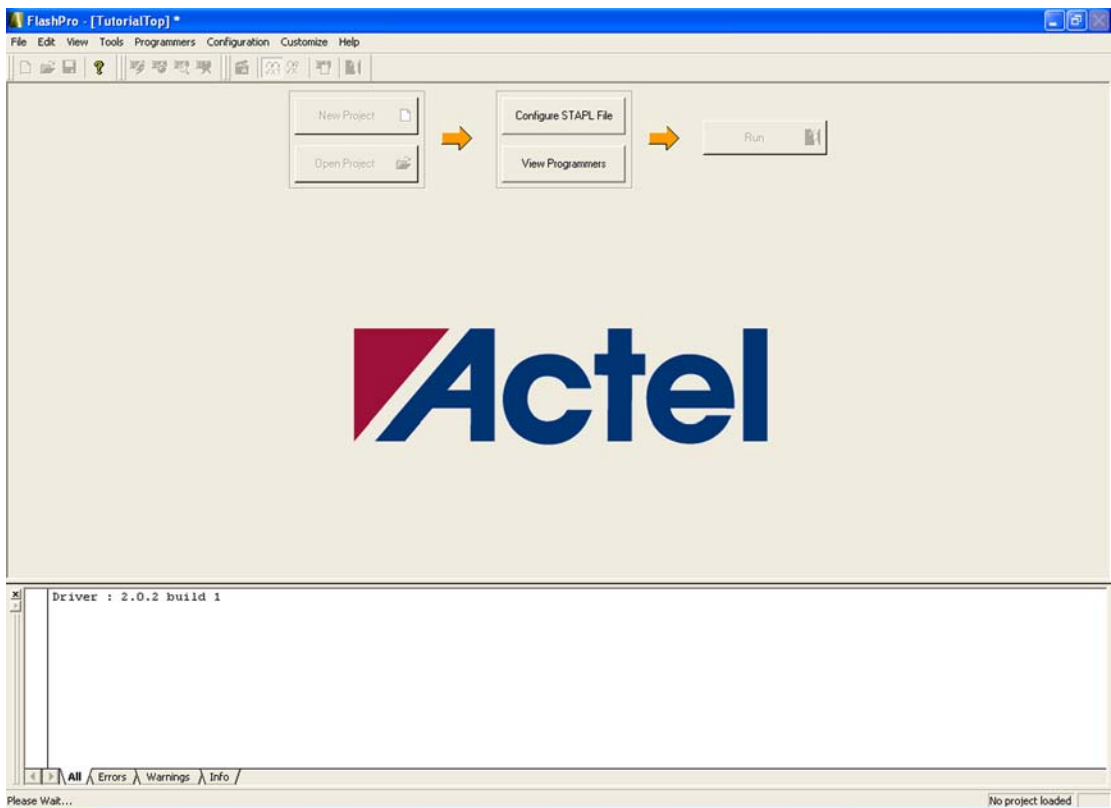


Figure 5-48. FlashPro Desktop – Prior to Locating Programmer(s)

FlashPro establishes a communication channel with the FlashPro3 programmer(s) attached to the PC (Figure 5-49).

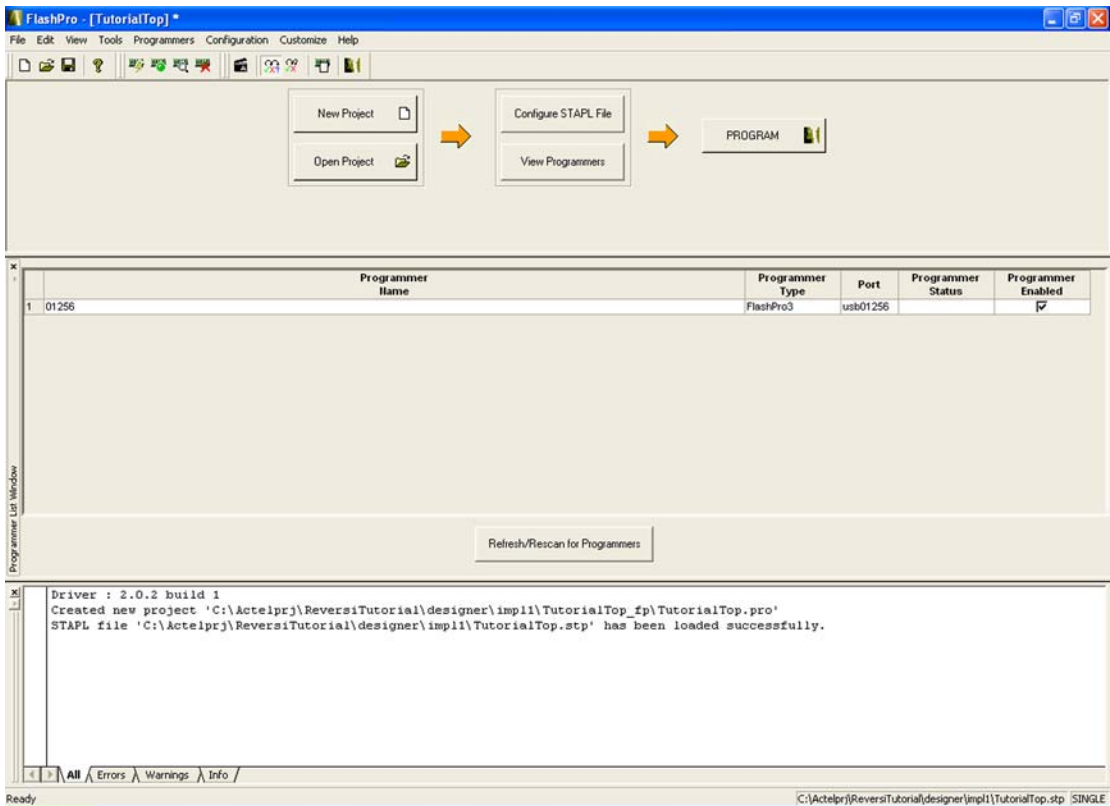


Figure 5-49. FlashPro Desktop – After Locating Programmer(s)

- When launching FlashPro from within Libero IDE, the project STAPL file is automatically loaded and configured. To verify the STAPL file being used to program the device, click the **Configure STAPL File** button.

The STAPL Configuration window (Figure 5-50) appears.

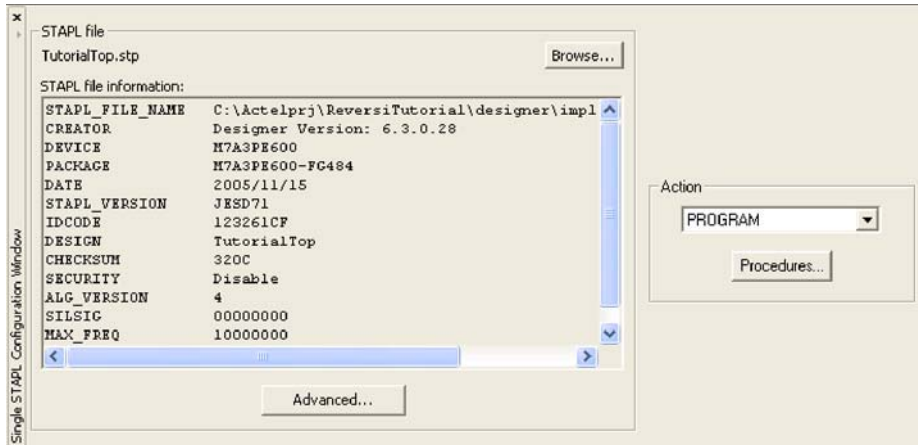


Figure 5-50. STAPL Configuration Window

- Here the programming file may be changed by clicking the **Browse** button and navigating to the new programming file. Various **Actions** may be performed using the drop-down selections. For this tutorial, leave it set to **PROGRAM** (default).

For more information on these functions, refer to the FlashPro online help.

To return to the Programmer List window (Figure 5-51), click the **View Programmers** button.



Figure 5-51. Programmers List Window

4. Verify that the attached programmer's check box is selected under the **Programmer Enabled** heading, then click the **Program** button.
5. Programming will take approximately two and a half to three minutes to complete. Under the **Programming Status** heading, a progress bar will appear. Alternatively, the log window may also be viewed (Figure 5-52).

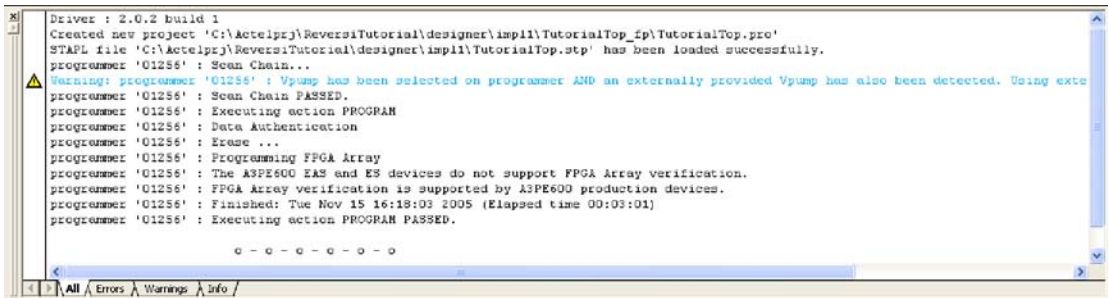


Figure 5-52. FlashPro Log Window

6. Once programming has completed, select **Exit** from the **File** menu. Answer **Yes** when prompted to save the project. This will return you to Libero IDE.
7. From the Libero IDE **File** menu, select **Exit**. If prompted to save your project, answer **Yes**, as this completes the Libero IDE FPGA portion of this tutorial.

## ARM RealView Developer Kit – Actel Edition

The RealView Developer Kit, available from Actel (separately from the CoreMP7 Development Kit), contains an integrated project manager and file editor suitable for creating and developing embedded projects. In this section, we'll create the executable source code to be run on CoreMP7. No coding will be required in this section, as we'll use source code included with the CoreMP7 Development Kit.

### Step 1 – Creating a RealView Project

#### *To create a RealView project:*

1. Copy the contents from the `\Tutorial\MPU` directory on the CoreMP7 Development Kit CD-ROM to the `C:\CoreMP7\Tutorial\Source` directory, which must be created.
2. Launch the **RealView Debugger 1.8** program located under **Start > ARM > RealView Developer Suite 2.2**.

Prior to launching RealView Debugger, the RealView ICE Micro Edition must be connected to the PC via the USB port. For information on installing and configuring the drivers for the RealView ICE Micro Edition, see the *RealView ICE – Micro Edition User's Guide* on the RealView Developer Kit – Actel Edition CD-ROM.

3. From the **Project** menu, select **New Project**. This displays the Create New Project dialog box, as shown in [Figure 5-53](#).

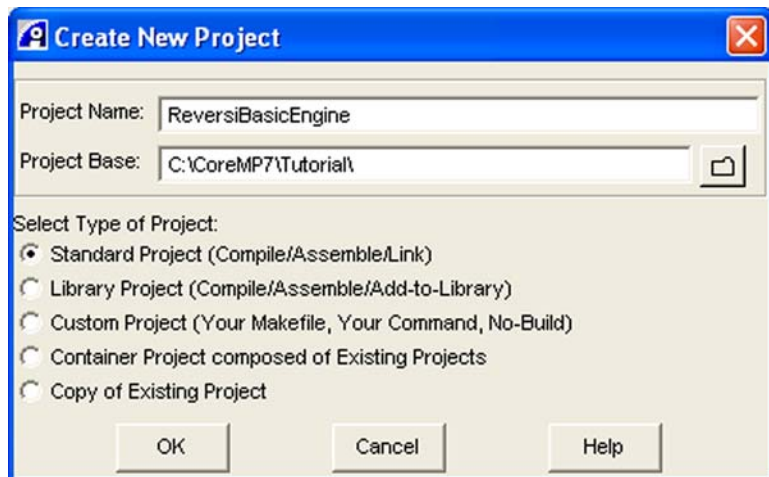


Figure 5-53. RealView Create New Project Dialog Box

4. Click the **Navigate** button (the folder to the right of **Project Base**), select **<Select Dir...>** from the context menu, and browse to the project directory: *C:\CoreMP7\Tutorial*. Click **Select**.
5. Ensure the **Standard Project** radio button is selected, and enter “ReversiBasicEngine” in the **Project Name** field. Click **OK**. This displays the Create Standard Project dialog box, shown in Figure 5-54.

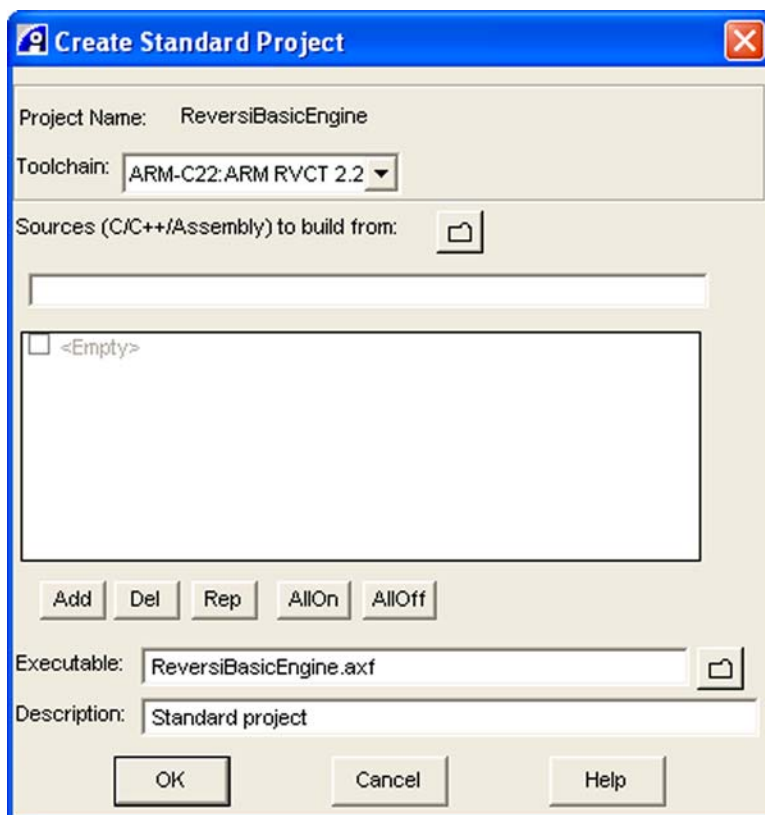


Figure 5-54. RealView Create Standard Project Dialog Box

6. Again, click the navigate button (to the right of **Sources to build from**) to open the Select Source Files for Project dialog box (Figure 5-55).

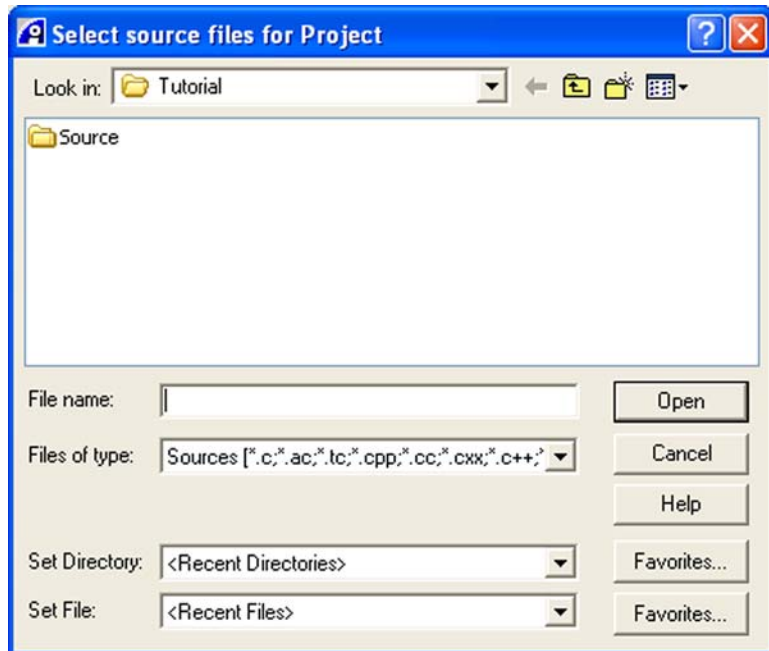


Figure 5-55. RealView Select Source Files for Project Dialog

7. Navigate to the `C:\CoreMP7\Tutorial\Source` directory and press CTRL + A to select all the files within the directory.
8. Click **Open**.



9. Click **OK** in the Create Standard Project dialog box. The Project Properties window now appears (Figure 5-56).

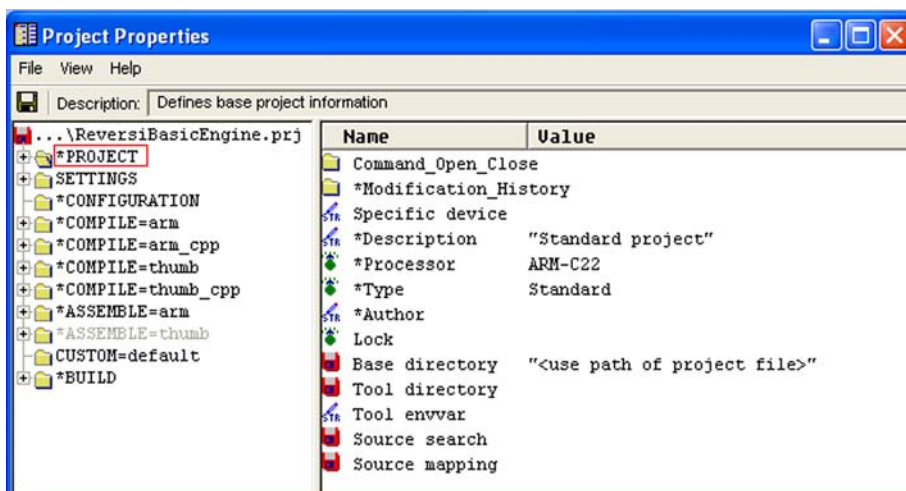


Figure 5-56. RealView Project Properties

10. Click the **CONFIGURATION** entry in the left-hand window pane to view the available project build variants.

11. Right-click the **Active config** entry in the **CONFIGURATION** pane and select **DebugRel** from the context menu, as shown in Figure 5-57.

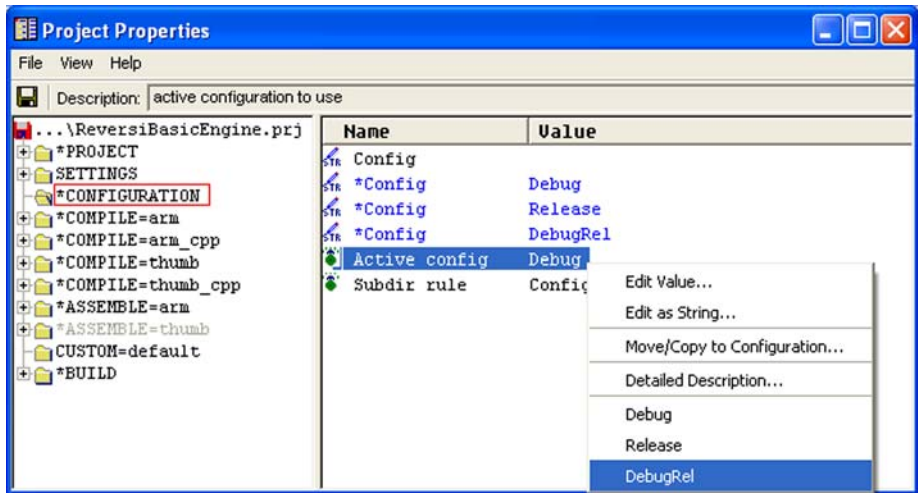


Figure 5-57. RealView Project Properties Configuration Options

The Configuration settings enable you to build your application program in different ways. They define the target configurations used in the build model. The most common target configurations are a Debug build, with debug information and no code optimization, and a Release build, with less debug information and high optimization.

This group can also be used to set up different optimization levels—for example, a DebugRel configuration with higher optimization than Debug but lower than Release. Another example is multiple variants of your application using different device drivers.

See the RealView documentation available at [http://www.arm.com/documentation/Trace\\_Debug/index.html](http://www.arm.com/documentation/Trace_Debug/index.html) or refer to online help for further information on this subject.

12. Expand the **BUILD** entry by clicking the plus sign to the left of its folder, and then select the **Link Advanced** sub-menu component.

13. Right-click the **Scatter file** entry in the **Link Advanced** pane and select **Edit as Filename** from the context menu, as shown in Figure 5-58.

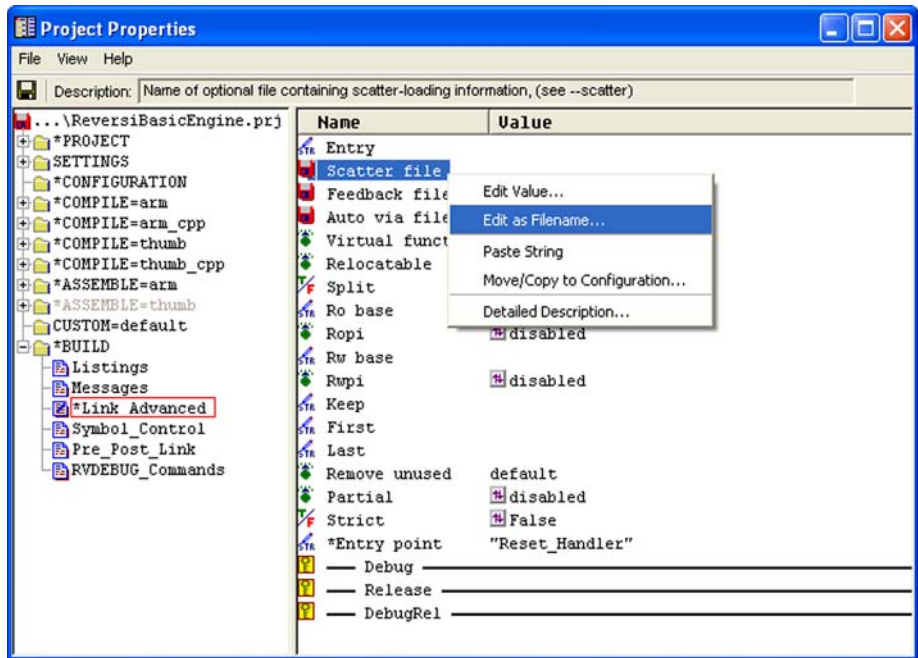


Figure 5-58. RealView Project Properties Build Options

14. Browse to the `..\CoreMP7\Tutorial\Source` directory and select the `CoreMP7DevKit.scf` scatter map description file.

The scatter file is used to tell the linker where to load files or objects residing in memory. For detailed reference information on the linker and scatter-loading, refer to the [ARM Developer Suite Linker and Utilities Guide](#).

The benefit of using a scatter description file is that all the (target-specific) absolute addresses chosen for your devices, code, and data are located in one file, making maintenance easy. Furthermore, if you decide to change your memory map (e.g., if peripherals are moved), you do not need to rebuild your entire project. You only need to re-link the existing objects.

15. Also within the **Link Advanced** menu, set the **Entry point** to **Reset\_Handler**.
16. From the **File** menu, select **Save and Close**. This will return you to the RealView Debugger desktop where you might see the compiler attempt to compile the source files with a makefile, which fails due to missing files. Ignore this error/warning.

1. From the **Build** menu, select **Build**. Select **Yes** if you are asked if you would like to Rebuild All. The output of the Build pane appears with several messages generated as a result of the attempted build, as shown in [Figure 5-59](#). If errors were present in your source code, they would be listed with the corresponding filename, line number, and a brief description of the error. If an error is found, the Code pane of the RealView Debugger opens the relevant source file with an arrow pointing to and highlighting the line of source code the first error message is referencing. Correct the error, save the modified source file, and then rebuild the project, and continue this process until no errors are present upon Compile.



- ### Step 3 – Debugging: Simulating/Executing the Compilation

A simulator attempts to model the entire behavior of a processor in software running on your personal computer. No matter the speed of your PC, there is no simulator that can simulate the microprocessor's real-time behavior. Further, there is no external world communication between the simulator and your target system. Stimulus files must be created and used to simulate external events. On the other hand, emulators typically replace the processor on the target board and interface directly with the external world. The emulator provides the user with all the features of simulation plus the capabilities of interfacing with the external world and running at full system speed. Emulators exist in two forms: Debug Modules and In-Circuit Emulation (ICE).

92

The ICE approach is slightly different. The ICE interfaces directly to the On-Chip Debug system within the actual processor. This provides an interface for complete control of the target processor—typically JTAG, but sometimes a proprietary interface, already embedded into the target processor.

The exact electrical and timing characteristics of the target system are achieved when using the On-Chip Debug system, whereas the Debug Module approach may provide additional features and access to internals of the target. For that reason, simulators are best suited for the testing of algorithms.

### Configuring the Simulator / On-Chip Debugger

The frontend tools for performing emulation are exactly the same. The only differences between simulation and emulation are the initial setup steps, specifically steps 3–6. The alternative steps for performing emulation are discussed in “Alternative Steps for Using the On-Chip Debugger” on page 101.

1. Click the **Src** tab in the RealView Debugger Debug window. The Code pane shows that there is currently not a target connected to the debugger.
2. Click the **Connect to Target** link to launch the Connection Control dialog.
3. Expand the **Server > localhost** branches in the name tree, then right-click **new\_arm** and select **Configure Device Info**, as shown in Figure 5-60.

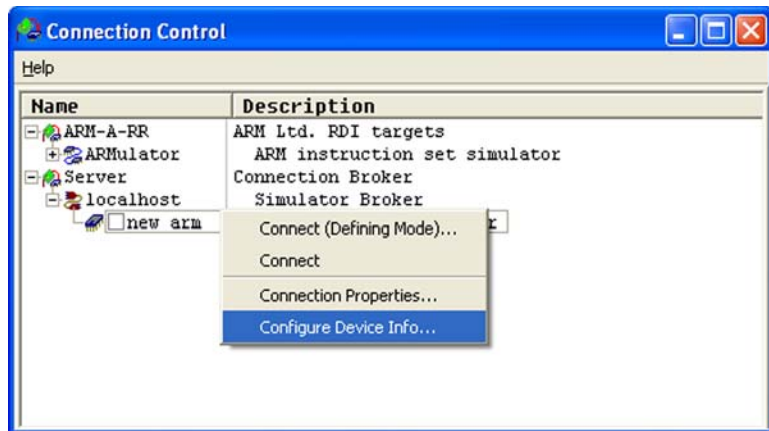


Figure 5-60. RealView Debugger Connection Control Dialog Box

4. The ARMulator Configuration dialog box appears. Select the **ARM7TDMI-S** processor, as shown in Figure 5-61. Click **OK** to return to the Connection Control dialog box.

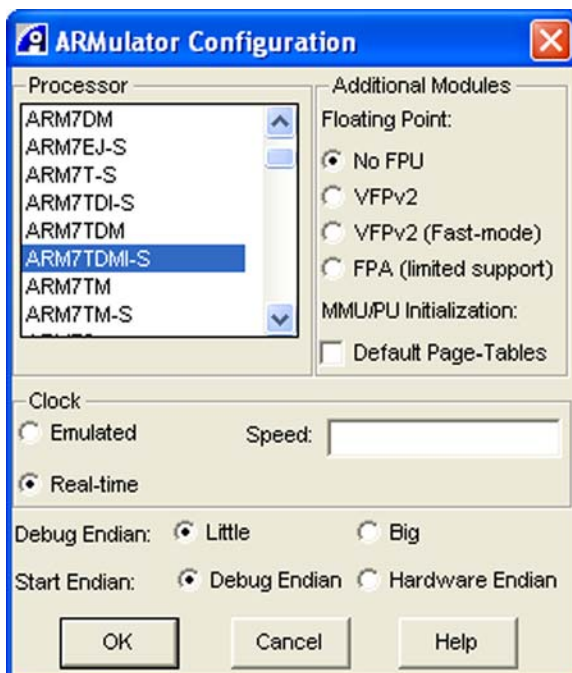


Figure 5-61. ARMulator Configuration Dialog Box

5. Select the **new\_arm** check box under the name tree. A new simulation object, Simarm\_1, will be instantiated, as shown in Figure 5-62.

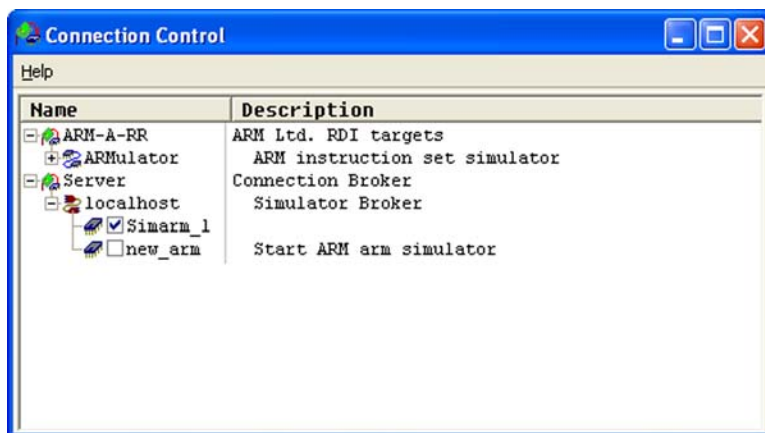


Figure 5-62. RealView Connection Control with Simulation Object

6. The RealView Debugger is now connected to the ARM7TDMI-S Instruction Set Simulator target. You may now close or minimize the Connection Control dialog box.
7. The RealView Debugger Code pane prompts for the loading of the recently built image to the target. Click the **Load** link to load the image. If the Code pane is not prompting you to load an image, click the **Code** tab at the bottom of the Code pane.

- The code is now loaded into the target. The current point of execution is identified by the red box, as shown in [Figure 5-63](#). The code currently being displayed is the basic initialization (or bootloader) code, which is typically written in assembly language.

```

Reset_Handler  FUNCTION

; Branch to C Library entry point

IMPORT    __main
BL        Setup_Stacks
LDR      r12,=__main
BX       r12                ; branch to __main
ENDFUNC

IMPORT ||Image$$RAM_EXEC$$ZI$$Limit||
EXPORT __user_initial_stackheap

heap_base    DCD    ||Image$$RAM_EXEC$$ZI$$Limit||

__user_initial_stackheap FUNCTION
LDR      r0,heap_base
MOV      pc,r0
ENDFUNC

Setup_Stacks
LDR      R0, =0xDEADBEEF
LDR      R1, =USR_STACK
STR      R0, [R1]

MOV      R0, #Mode_IPQ:0R:I_Bit:0R:F_Bit

```

Figure 5-63. RealView Debugger Code Pane with Loaded Target

## Debugging the Design

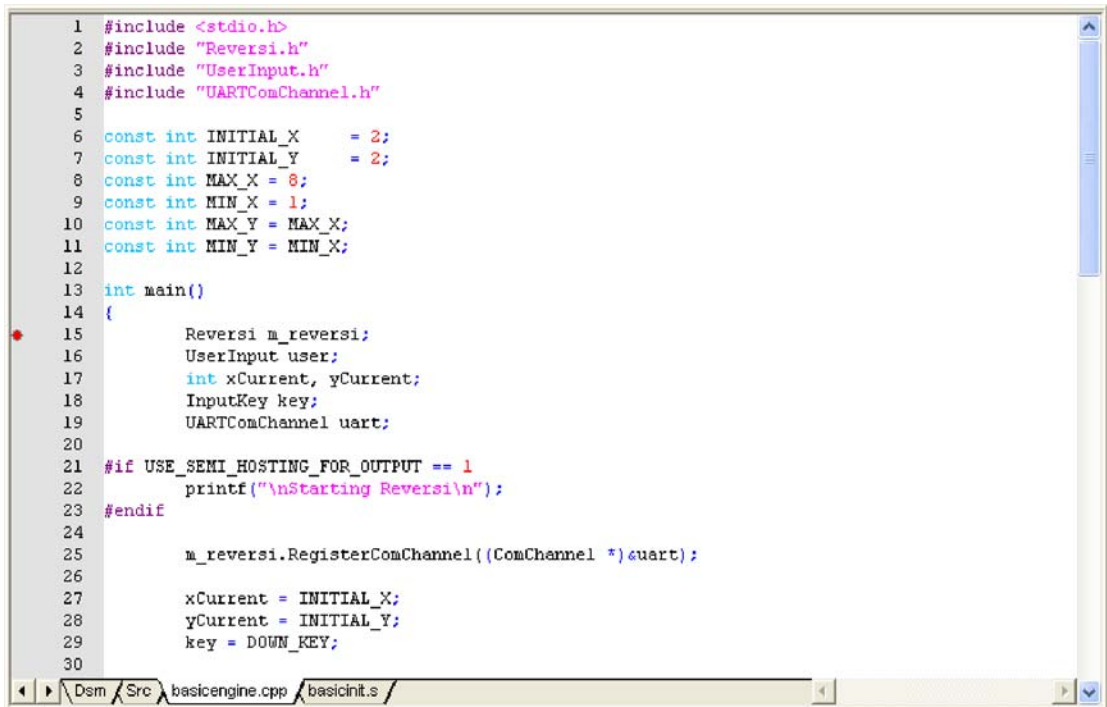
This section will touch on the basics of debugging from the simulation point of view. The primary goal of this section is to learn the very basic features of the debugger, as their application in the following example is rudimentary.

- Open the *basicengine.cpp* source file found in the `..\CoreMP7\Tutorial\Source` directory using the **Open** option under the **File** menu.
- Select **Show Line Numbers** from the **Advanced** sub-menu of the **Edit** menu to display line numbers in the margins of the source code.
- Set a breakpoint on line 15, the “`Reversi m reversi;`” statement, by double-clicking to the left of the line number. A red breakpoint marker will appear to the left of the line number, as shown in [Figure 5-64 on page 97](#).

A *breakpoint* is a user-defined stopping point in a program that is inserted for debugging purposes. Breakpoints are a method embedded developers use to gain information about a



program during its execution. During the *break*, the developer can examine the internal contents of the processor, memory, registers, etc. to ensure proper operation.



```

1  #include <stdio.h>
2  #include "Reversi.h"
3  #include "UserInput.h"
4  #include "UARTComChannel.h"
5
6  const int INITIAL_X    = 2;
7  const int INITIAL_Y    = 2;
8  const int MAX_X        = 8;
9  const int MIN_X        = 1;
10 const int MAX_Y        = MAX_X;
11 const int MIN_Y        = MIN_X;
12
13 int main()
14 {
15     Reversi m_reversi;
16     UserInput user;
17     int xCurrent, yCurrent;
18     InputKey key;
19     UARTComChannel uart;
20
21     #if USE_SEMI_HOSTING_FOR_OUTPUT == 1
22         printf("\nStarting Reversi\n");
23     #endif
24
25     m_reversi.RegisterComChannel((ComChannel *) &uart);
26
27     xCurrent = INITIAL_X;
28     yCurrent = INITIAL_Y;
29     key = DOWN_KEY;
30

```

The screenshot shows a code editor window with a red dot indicating a breakpoint on line 15. The file path at the bottom is \Dsm\Src\basicengine.cpp\basicinit.s.

Figure 5-64. RealView Code Pane with Breakpoint

4. Select **Run** from the **Debug** menu (or use the F5 shortcut key). The cursor (red box) will now be present on line 15 (where the breakpoint was set).
5. Open the *position.cpp* source file and set a breakpoint on line 125, the “m\_board[4][4] = White;” statement.

- Right-click **m\_board** and select **Watch** from the context menu. The m\_board array will be added to the watch window, as shown in Figure 5-65.

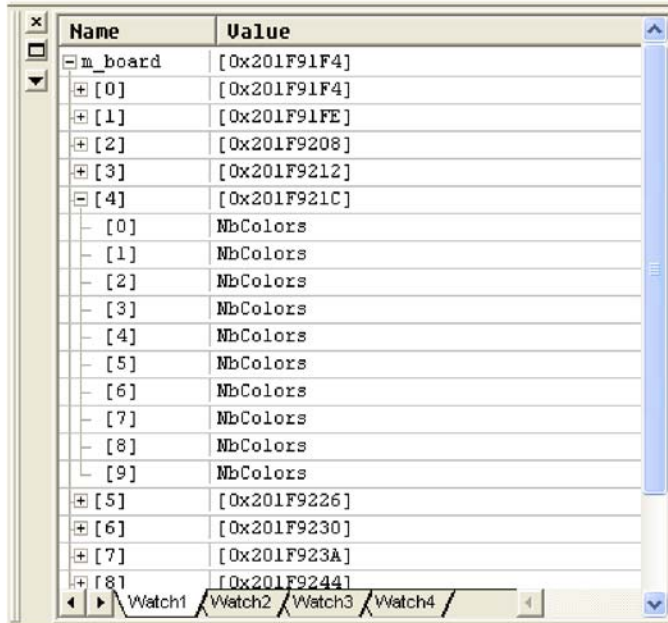


Figure 5-65. RealView Debugger Watch Window

A *watch* is a variable or expression that you require the debugger to display at every step or breakpoint so that you can see how its value changes. The Watch pane is part of the RealView Debugger Code window and displays the watches you have defined.

7. Within the watch window, expand the **m\_board** variable and then expand the **[4]** sub-component (a dimension of the *m\_board* array).
8. Select **Step Into** from the **Debug** menu (or use the F11 shortcut key). The cursor (red box) will now be present on line 126. Looking at the watch window, you can see that the *m\_board*[4][4] contents have changed from the default “nbColors” to the value “White.”
9. Continuing to single-step (or **Step Into**) three more times, you will see changes in the [5][4], [5][5], and [4][5] components of the *m\_board* multi-dimensional array.
10. Select **Registers** from the **View** menu. The Register window is displayed; you may choose to “dock” it into the RealView Debugger, as shown in [Figure 5-66](#).

The *Register window* displays the contents of the internal processor registers at every step or breakpoint so that you can see how its value changes. The register contents may also be modified through this interface by simply double-clicking the value and modifying the field.

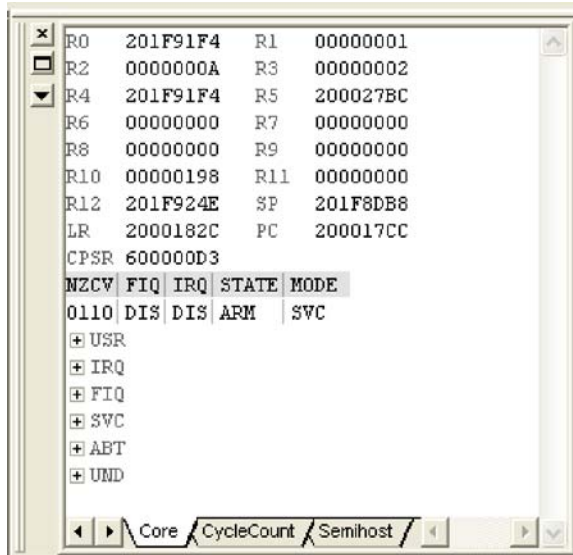


Figure 5-66. RealView Debugger Register Window

11. Right-click one of the <<NoAddr>> values in the RealView Debugger Memory window and select **Set Start Address**. Enter the value **0xC2000000**. The Memory window will now display the memory segment contents beginning with the aforementioned address, as shown in [Figure 5-67](#).

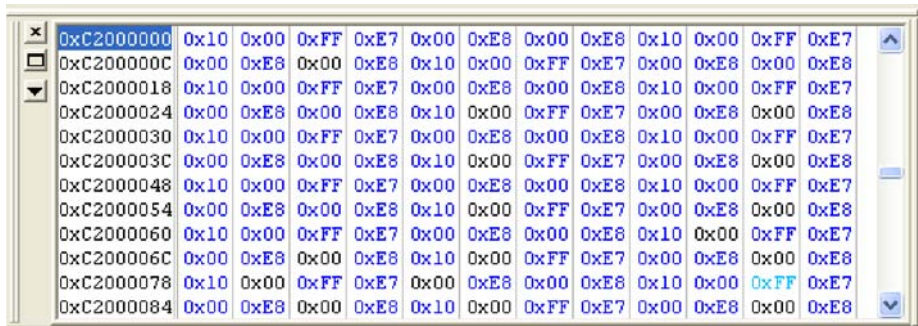


Figure 5-67. RealView Debugger Memory Window

The *Memory window* displays the contents of the memories (both Flash and SRAM), as well as the memory-mapped peripheral configuration registers, at every step or breakpoint so that you can see how its value changes. The memory contents may also be modified through this interface by simply double-clicking the value and modifying the field.

12. If you are debugging with the emulator, you can modify the memory contents at 0xC2000000 and twiddle the LEDs on the CoreMP7 Evaluation Board, thus showing that you have direct access to the on-chip peripherals.
13. If you wish to run the complete Reversi (also known as Othello) game on the CoreMP7 Development Kit, continue with [“Running the Reversi Game via the On-Chip Debugger” on page 104](#). Otherwise, end the debugging session by selecting **Disconnect** from the **Target** menu.

## Alternative Steps for Using the On-Chip Debugger

1. Place a spare jumper on pin #37 and #39 of the J11 header (pins #AB15 and #AB17). Configure the part to boot from the on-board synchronous SRAM.
2. Connect the RVI-ME to the CoreMP7 Evaluation Board through the ARM\_JTAG header and power up the board.
3. Expand the **ARM-ARM-USB** branch to reveal **RVI-ME** in the name tree. Right-click **RVI-ME** and select **Configure Device Info**, as shown in Figure 5-68. This launches the RVConfig utility.

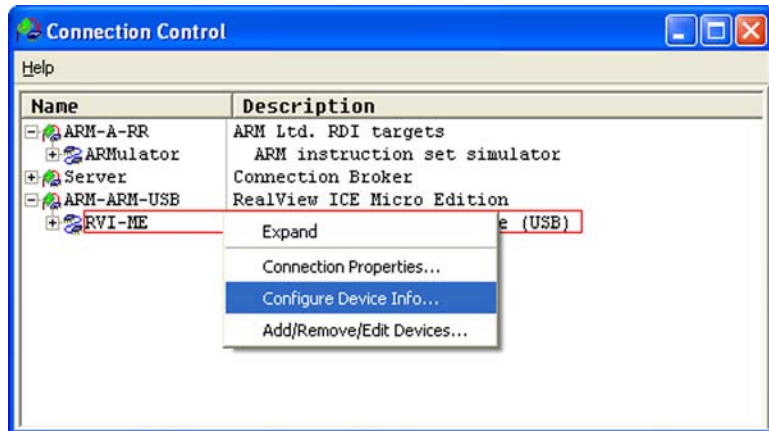


Figure 5-68. RealView Debugger RVI-ME Configure Device

4. Click the **Auto Configure Scan Chain** button in the RVConfig utility. After a few seconds, the ARM7TDMI-S processor will be identified, as shown in Figure 5-69.

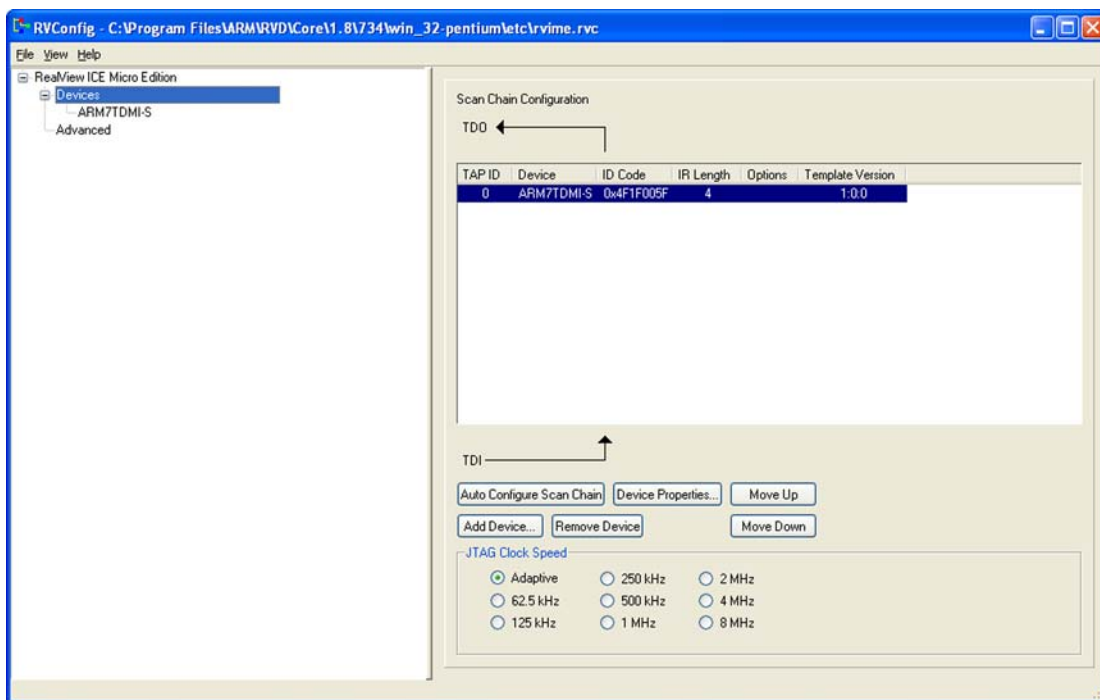


Figure 5-69. RealView Debugger RVConfig Dialog Box

5. From the **File** menu, select **Save**. Then select **File > Exit** and return to the Configuration Control dialog box.
6. Select the **ARM7TDMI-S** check box under the Name tree. It may be necessary to expand the **RVI-ME** branch first, as shown in [Figure 5-70](#).

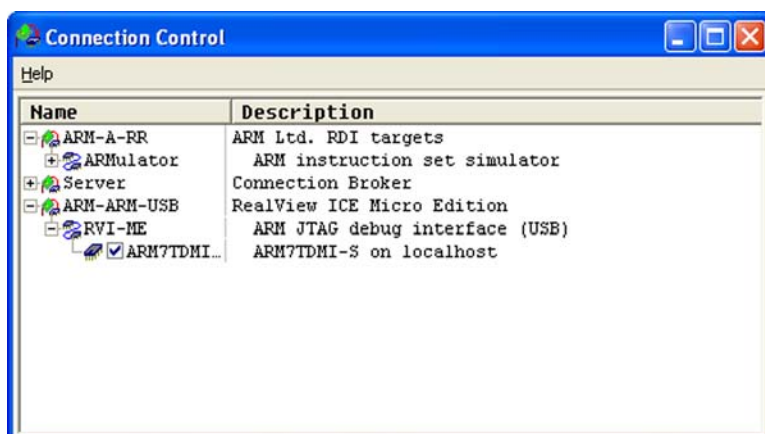


Figure 5-70. RealView Debugger Connection Control with On-Chip Debugger Target

7. The RealView Debugger is now connected to the ARM7TDMI-S via the On-Chip Debugger. You may now close or minimize the Connection Control dialog box.
8. Continue with the Quickstart Tutorial with step 7 on [page 95](#).

## Running the Reversi Game via the On-Chip Debugger

1. Reload the *ReversiBasicEngine.axf* file by selecting **Reload Image to Target** from the **Debug** menu.
2. Connect the RS-232 connector P2 to the COM1 port of your PC using a standard straight-through serial cable.
3. From the **Debug** menu, select **Run** (or use the F11 shortcut key). There will be an indicator in the RealView Debugger status bar which indicates that the target is *Running* with a small progress bar.
4. Minimize the RealView Debugger tool.
5. Navigate to the *..\Tutorial\Demo* directory on the Tutorial CD-ROM and double-click the *ReversiGUI.exe* executable.
6. Pressing SW6, SW7, SW8, SW9, or SW10 will commence the game.
7. To end the game, select **Stop Execution** from the **Debug** menu (or use the SHIFT + F5 shortcut key).
8. Disconnect from the target by selecting **Disconnect** from the **Target** menu. It is now safe to close the RealView Debugger.

## Reversi Background

The object of the game is to own more pieces than your opponent when the game is over. The game is over when neither player has a move. Usually, this means the board is full. On your turn, you place one piece on the board with your color facing up. You must place the piece so that your opponent's piece, or a row of your opponent's pieces, is flanked by your pieces. All of the opponent's pieces between your pieces are then turned over to become your color.

## Game Controls

- SW6: Moves the placement box up one unit
- SW7: Moves the placement box down one unit
- SW8: Places a piece
- SW9: Moves the placement box to the left one unit
- SW10: Moves the placement box to the right one unit



# M7A3PE600 and M7A3P1000 FG484

## Package Connections

Due to the comprehensive and flexible nature of M7 ProASIC3/E device user I/Os, a naming scheme is used to show the details of each I/O. The name identifies the I/O bank to which the I/O belongs as well as the pairing and pin polarity for differential I/Os.

I/O nomenclature: Gmn or IOuxwBy

Gmn is only used for I/Os that also have CCC access, i.e., global pins.

- G: Global
- m: Global pin location associated with each CCC on the device: A (northwest corner), B (northeast corner), C (east middle), D (southeast corner), E (southwest corner), and F (west middle)
- n: Global input MUX and pin number of the associated global location m: either A0, A1, A2, B0, B1, B2, C0, C1, or C2
- u: I/O pair number in the bank, starting at 00 from the northwest I/O bank and proceeding in a clockwise direction
- x: P (positive) or N (negative) for differential pairs, or R (regular – single-ended) for I/Os that support single-ended and voltage-referenced I/O standards only. U (positive-LVDS only) or V (negative-LVDS only) restricts the I/O differential pair from being selected as an LVPECL pair.
- w: D (differential pair), P (pair) or S (single-ended). D if both members of the pair are bonded out to adjacent pins or are separated only by one GND or NC pin, P if both members of the pair are bonded out but do not meet the adjacency requirement, or S if the I/O pair is not bonded out. For differential pairs, adjacency for ball grid packages means only vertical or horizontal. Diagonal adjacency does not meet the requirements for a true differential pair.
- B: Bank
- y: Bank number [0..3] for M7 ProASIC3 and [0..7] for M7 ProASIC3E. The bank number starts at 0 from the northwest I/O bank and proceeds in a clockwise direction.

Figure A-1 on page 106 and Table A-1 on page 107 are extracted from the *ProASIC3 Flash Family FPGAs with Optional Soft ARM Support* and *ProASIC3E Flash Family FPGAs with Optional Soft ARM support* datasheets and provide package connections for the M7A3PE600 and M7A3P1000 devices.

Pinouts for other devices in the FG484 family may be found on the Actel website:

*ProASIC3 Flash Family FPGAs with Optional Soft ARM Support* datasheet at [www.actel.com/documents/PA3\\_DS.pdf](http://www.actel.com/documents/PA3_DS.pdf)

*ProASIC3E Flash Family FPGAs with Optional Soft ARM Support* datasheet at [www.actel.com/documents/PA3E\\_DS.pdf](http://www.actel.com/documents/PA3E_DS.pdf)

These datasheets are included on the CoreMP7 Development Kit CD. However, always refer to the website for the most recent datasheet.

## 484-Pin FGBGA Package

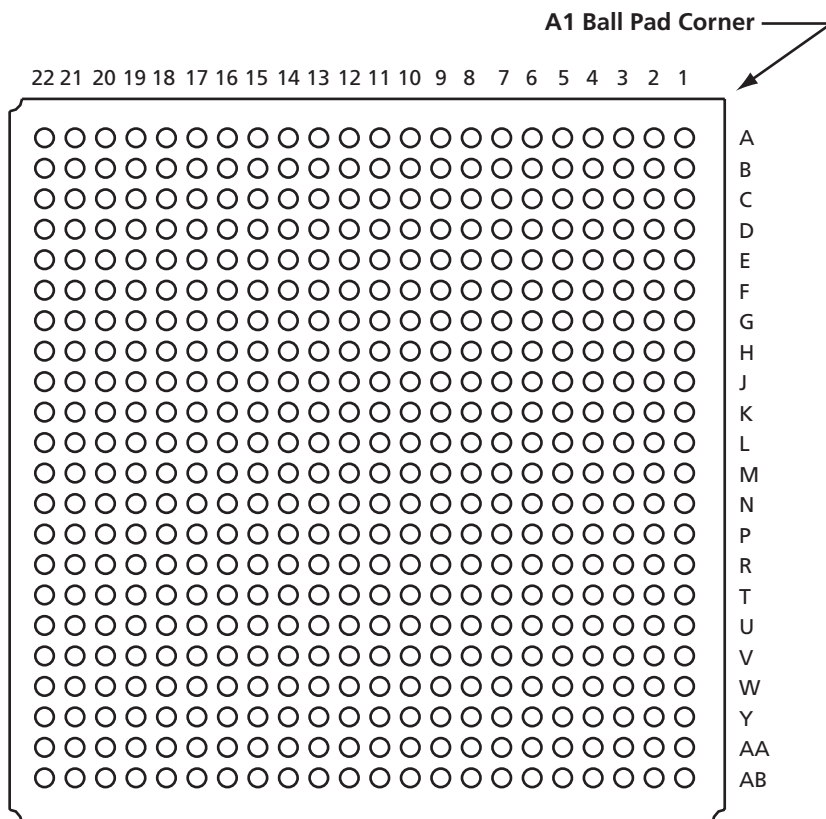


Figure A-1. 484-Pin FGBGA Package Layout

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections

Pin Number	M7A3P1000 Function	M7A3PE600 Function
A1	GND	GND
A2	GND	GND
A3	V <sub>CCI</sub> B0	V <sub>CCI</sub> B0
A4	IO07RSB0	IO06NDB0V1
A5	IO09RSB0	IO06PDB0V1
A6	IO13RSB0	IO08NDB0V1
A7	IO18RSB0	IO08PDB0V1
A8	IO20RSB0	IO11PDB0V1
A9	IO26RSB0	IO17PDB0V2
A10	IO32RSB0	IO18NDB0V2
A11	IO40RSB0	IO18PDB0V2
A12	IO41RSB0	IO22PDB1V0
A13	IO53RSB0	IO26PDB1V0
A14	IO59RSB0	IO29NDB1V1
A15	IO64RSB0	IO29PDB1V1
A16	IO65RSB0	IO31NDB1V1
A17	IO67RSB0	IO31PDB1V1
A18	IO69RSB0	IO32NDB1V1
A19	NC	NC
A20	V <sub>CCI</sub> B0	V <sub>CCI</sub> B1
A21	GND	GND
A22	GND	GND
AA1	GND	GND
AA2	V <sub>CCI</sub> B3	V <sub>CCI</sub> B7
AA3	NC	NC
AA4	IO181RSB2	IO03NDB0V0

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
AA5	IO178RSB2	IO03PDB0V0
AA6	IO175RSB2	IO07NDB0V1
AA7	IO169RSB2	IO07PDB0V1
AA8	IO166RSB2	IO11NDB0V1
AA9	IO160RSB2	IO17NDB0V2
AA10	IO152RSB2	IO14PDB0V2
AA11	IO146RSB2	IO19PDB0V2
AA12	IO139RSB2	IO22NDB1V0
AA13	IO133RSB2	IO26NDB1V0
AA14	NC	NC
AA15	NC	NC
AA16	IO122RSB2	IO30NDB1V1
AA17	IO119RSB2	IO30PDB1V1
AA18	IO117RSB2	IO32PDB1V1
AA19	NC	NC
AA20	NC	NC
AA21	V <sub>CCI</sub> B1	V <sub>CCI</sub> B2
AA22	GND	GND
AB1	GND	V <sub>CCI</sub> B7
AB2	GND	NC
AB3	V <sub>CCI</sub> B2	NC
AB4	IO180RSB2	NC
AB5	IO176RSB2	GND
AB6	IO173RSB2	IO04NDB0V0
AB7	IO167RSB2	IO04PDB0V0
AB8	IO162RSB2	V <sub>CC</sub>
AB9	IO156RSB2	V <sub>CC</sub>
AB10	IO150RSB2	IO14NDB0V2

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
AB11	IO145RSB2	IO19NDB0V2
AB12	IO144RSB2	NC
AB13	IO132RSB2	NC
AB14	IO127RSB2	V <sub>CC</sub>
AB15	IO126RSB2	V <sub>CC</sub>
AB16	IO123RSB2	NC
AB17	IO121RSB2	NC
AB18	IO118RSB2	GND
AB19	NC	NC
AB20	V <sub>CCI</sub> B2	NC
AB21	GND	NC
AB22	GND	V <sub>CCI</sub> B2
B1	GND	NC
B2	V <sub>CCI</sub> B3	NC
B3	NC	NC
B4	IO06RSB0	GND
B5	IO08RSB0	GAA0/IO00NDB0V0
B6	IO12RSB0	GAA1/IO00PDB0V0
B7	IO15RSB0	GAB0/IO01NDB0V0
B8	IO19RSB0	IO05PDB0V0
B9	IO24RSB0	IO10PDB0V1
B10	IO31RSB0	IO12PDB0V2
B11	IO39RSB0	IO16NDB0V2
B12	IO48RSB0	IO23NDB1V0
B13	IO54RSB0	IO23PDB1V0
B14	IO58RSB0	IO28NDB1V1
B15	IO63RSB0	IO28PDB1V1
B16	IO66RSB0	GGB1/IO34PDB1V1

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
B17	IO68RSB0	GBA0/IO35NDB1V1
B18	IO70RSB0	GBA1/IO35PDB1V1
B19	NC	GND
B20	NC	NC
B21	V <sub>CCI</sub> B1	NC
B22	GND	NC
C1	V <sub>CCI</sub> B3	NC
C2	IO220PDB3	NC
C3	NC	GND
C4	NC	GAB2/IO133PDB7V1
C5	GND	GAA2/IO134PDB7V1
C6	IO10RSB0	GNDQ
C7	IO14RSB0	GAB1/IO01PDB0V0
C8	V <sub>CC</sub>	IO05NDB0V0
C9	V <sub>CC</sub>	IO10NDB0V1
C10	IO30RSB0	IO12NDB0V2
C11	IO37RSB0	IO16PDB0V2
C12	IO43RSB0	IO20NDB1V0
C13	NC	IO24NDB1V0
C14	V <sub>CC</sub>	IO24PDB1V0
C15	V <sub>CC</sub>	GBC1/IO33PDB1V1
C16	NC	GBB0/IO34NDB1V1
C17	NC	GNDQ
C18	GND	GBA2/IO36PDB2V0
C19	NC	IO42NDB2V0
C20	NC	GND
C21	NC	NC
C22	V <sub>CCI</sub> B1	NC

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
D1	IO219PDB3	NC
D2	IO220NDB3	IO131NDB7V1
D3	NC	IO131PDB7V1
D4	GND	IO133NDB7V1
D5	GAA0/IO00RSB0	IO134NDB7V1
D6	GAA1/IO01RSB0	VMV7
D7	GAB0/IO02RSB0	V <sub>CCPLA</sub>
D8	IO16RSB0	GAC0/IO02NDB0V0
D9	IO22RSB0	GAC1/IO02PDB0V0
D10	IO28RSB0	IO15NDB0V2
D11	IO35RSB0	IO15PDB0V2
D12	IO45RSB0	IO20PDB1V0
D13	IO50RSB0	IO25NDB1V0
D14	IO55RSB0	IO27PDB1V0
D15	IO61RSB0	GBC0/IO33NDB1V1
D16	GBB1/IO75RSB0	V <sub>CCPLB</sub>
D17	GBA0/IO76RSB0	VMV2
D18	GBA1/IO77RSB0	IO36NDB2V0
D19	GND	IO42PDB2V0
D20	NC	NC
D21	NC	NC
D22	NC	NC
E1	IO219NDB3	IO127NDB7V1
E2	NC	IO127PDB7V1
E3	GND	NC
E4	GAB2/IO224PDB3	IO128PDB7V1
E5	GAA2/IO225PDB3	IO129PDB7V1
E6	GNDQ	GAC2/IO132PDB7V1

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
E7	GAB1/IO03RSB0	V <sub>COMPLA</sub>
E8	IO17RSB0	GNDQ
E9	IO21RSB0	IO09NDB0V1
E10	IO27RSB0	IO09PDB0V1
E11	IO34RSB0	IO13PDB0V2
E12	IO44RSB0	IO21PDB1V0
E13	IO51RSB0	IO25PDB1V0
E14	IO57RSB0	IO27NDB1V0
E15	GBC1/IO73RSB0	GNDQ
E16	GBB0/IO74RSB0	V <sub>COMPLB</sub>
E17	IO71RSB0	GBB2/IO37PDB2V0
E18	GBA2/IO78PDB1	IO39PDB2V0
E19	IO81PDB1	IO39NDB2V0
E20	GND	IO43PDB2V0
E21	NC	IO43NDB2V0
E22	IO84PDB1	NC
F1	NC	NC
F2	IO215PDB3	NC
F3	IO215NDB3	V <sub>CC</sub>
F4	IO224NDB3	IO128NDB7V1
F5	IO225NDB3	IO129NDB7V1
F6	VMV3	IO132NDB7V1
F7	IO11RSB0	IO130PDB7V1
F8	GAC0/IO04RSB0	VMV0
F9	GAC1/IO05RSB0	V <sub>CCI</sub> B0
F10	IO25RSB0	V <sub>CCI</sub> B0
F11	IO36RSB0	IO13NDB0V2
F12	IO42RSB0	IO21NDB1V0



Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
F13	IO49RSB0	V <sub>CCI</sub> B1
F14	IO56RSB0	V <sub>CCI</sub> B1
F15	GBC0/IO72RSB0	VMV1
F16	IO62RSB0	GBC2/IO38PDB2V0
F17	VMV0	IO37NDB2V0
F18	IO78NDB1	IO41NDB2V0
F19	IO81NDB1	IO41PDB2V0
F20	IO82PPB1	V <sub>CC</sub>
F21	NC	NC
F22	IO84NDB1	NC
G1	IO214NDB3	IO123NDB7V0
G2	IO214PDB3	IO123PDB7V0
G3	NC	NC
G4	IO222NDB3	IO124PDB7V0
G5	IO222PDB3	IO125PDB7V0
G6	GAC2/IO223PDB3	IO126PDB7V0
G7	IO223NDB3	IO130NDB7V1
G8	GNDQ	V <sub>CCI</sub> B7
G9	IO23RSB0	GND
G10	IO29RSB0	V <sub>CC</sub>
G11	IO33RSB0	V <sub>CC</sub>
G12	IO46RSB0	V <sub>CC</sub>
G13	IO52RSB0	V <sub>CC</sub>
G14	IO60RSB0	GND
G15	GNDQ	V <sub>CCI</sub> B2
G16	IO80NDB1	IO38NDB2V0
G17	GBB2/IO79PDB1	IO40NDB2V0
G18	IO79NDB1	IO40PDB2V0

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
G19	IO82NPB1	IO45PDB2V1
G20	IO85PDB1	NC
G21	IO85NDB1	IO48PDB2V1
G22	NC	IO46PDB2V1
H1	NC	IO121NDB7V0
H2	NC	IO121PDB7V0
H3	V <sub>CC</sub>	NC
H4	IO217PDB3	IO124NDB7V0
H5	IO218PDB3	IO125NDB7V0
H6	IO221NDB3	IO126NDB7V0
H7	IO221PDB3	GFC1/IO120PPB7V0
H8	VMV0	V <sub>CCI</sub> B7
H9	V <sub>CCI</sub> B0	V <sub>CC</sub>
H10	V <sub>CCI</sub> B0	GND
H11	IO38RSB0	GND
H12	IO47RSB0	GND
H13	V <sub>CCI</sub> B0	GND
H14	V <sub>CCI</sub> B0	V <sub>CC</sub>
H15	VMV1	V <sub>CCI</sub> B2
H16	GBC2/IO80PDB1	GCC1/IO50PPB2V1
H17	IO83PPB1	IO44NDB2V1
H18	IO86PPB1	IO44PDB2V1
H19	IO87PDB1	IO49NPB2V1
H20	V <sub>CC</sub>	IO45NDB2V1
H21	NC	IO48NDB2V1
H22	NC	IO46NDB2V1
J1	IO212NDB3	NC
J2	IO212PDB3	IO122PDB7V0

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
J3	NC	IO122NDB7V0
J4	IO217NDB3	GFB0/IO119NPB7V0
J5	IO218NDB3	GFA0/IO118NDB6V1
J6	IO216PDB3	GFB1/IO119PPB7V0
J7	IO216NDB3	V <sub>CC</sub> PLF
J8	V <sub>CC</sub> I <sub>B</sub> 3	GFC0/IO120NPB7V0
J9	GND	V <sub>CC</sub>
J10	V <sub>CC</sub>	GND
J11	V <sub>CC</sub>	GND
J12	V <sub>CC</sub>	GND
J13	V <sub>CC</sub>	GND
J14	GND	V <sub>CC</sub>
J15	V <sub>CC</sub> I <sub>B</sub> 1	GCC0/IO50NPB2V1
J16	IO83NPB1	GCB1/IO51PPB2V1
J17	IO86NPB1	GCA0/IO52NPB3V0
J18	IO90PPB1	V <sub>CC</sub> PLC
J19	IO87NDB1	GCB0/IO51NPB2V1
J20	NC	IO49PPB2V1
J21	IO89PDB1	IO47NDB2V1
J22	IO89NDB1	IO47PDB2V1
K1	IO211PDB3	NC
K2	IO211NDB3	IO114NDB6V1
K3	NC	IO117NDB6V1
K4	IO210PPB3	GFA2/IO117PDB6V1
K5	IO213NDB3	GFA1/IO118PDB6V1
K6	IO213PDB3	V <sub>CC</sub> PLF
K7	GFC1/IO209PPB3	IO116NDB6V1
K8	V <sub>CC</sub> I <sub>B</sub> 3	GFB2/IO116PDB6V1

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
K9	V <sub>CC</sub>	V <sub>CC</sub>
K10	GND	GND
K11	GND	GND
K12	GND	GND
K13	GND	GND
K14	V <sub>CC</sub>	V <sub>CC</sub>
K15	V <sub>CC</sub> I <sub>B</sub> 1	GCB2/IO54PPB3V0
K16	GCC1/IO91PPB1	GCA1/IO52PPB3V0
K17	IO90NPB1	GCC2/IO55PPB3V0
K18	IO88PDB1	V <sub>CC</sub> PLC
K19	IO88NDB1	GCA2/IO53PDB3V0
K20	IO94NPB1	IO53NDB3V0
K21	IO98NDB1	IO56PDB3V0
K22	IO98PDB1	NC
L1	NC	IO114PDB6V1
L2	IO200PDB3	IO111NDB6V1
L3	IO210NPB3	NC
L4	GFB0/IO208NPB3	GFC2/IO115PDB6V1
L5	GFA0/IO207NDB3	IO113PPB6V1
L6	GFB1/IO208PPB3	IO112PDB6V1
L7	V <sub>COM</sub> PLF	IO112NDB6V1
L8	GFC0/IO209NPB3	V <sub>CC</sub> I <sub>B</sub> 6
L9	V <sub>CC</sub>	V <sub>CC</sub>
L10	GND	GND
L11	GND	GND
L12	GND	GND
L13	GND	GND
L14	V <sub>CC</sub>	V <sub>CC</sub>

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
L15	GCC0/IO91NPB1	V <sub>CCI</sub> B3
L16	GCB1/IO92PPB1	IO54NPB3V0
L17	GCA0/IO93NPB1	IO57NPB3V0
L18	IO96NPB1	IO55NPB3V0
L19	GCB0/IO92NPB1	IO57PPB3V0
L20	IO97PDB1	NC
L21	IO97NDB1	IO56NDB3V0
L22	IO99NPB1	IO58PDB3V0
M1	NC	NC
M2	IO200NDB3	IO111PDB6V1
M3	IO206NDB3	IO115NDB6V1
M4	GFA2/IO206PDB3	IO113NPB6V1
M5	GFA1/IO207PDB3	IO109PPB6V0
M6	V <sub>CCPLF</sub>	IO108PDB6V0
M7	IO205NDB3	IO108NDB6V0
M8	GFB2/IO205PDB3	V <sub>CCI</sub> B6
M9	V <sub>CC</sub>	GND
M10	GND	V <sub>CC</sub>
M11	GND	V <sub>CC</sub>
M12	GND	V <sub>CC</sub>
M13	GND	V <sub>CC</sub>
M14	V <sub>CC</sub>	GND
M15	GCB2/IO95PPB1	V <sub>CCI</sub> B3
M16	GCA1/IO93PPB1	GDB0/IO66NPB3V1
M17	GCC2/IO96PPB1	IO60NDB3V1
M18	IO100PPB1	IO60PDB3V1
M19	GCA2/IO94PPB1	IO61PDB3V1
M20	IO101PPB1	NC

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
M21	IO99PPB1	IO59PDB3V0
M22	NC	IO58NDB3V0
N1	IO201NDB3	NC
N2	IO201PDB3	IO110PDB6V0
N3	NC	V <sub>CC</sub>
N4	GFC2/IO204PDB3	IO109NPB6V0
N5	IO204NDB3	IO106NDB6V0
N6	IO203NDB3	IO106PDB6V0
N7	IO203PDB3	GEC0/IO104NPB6V0
N8	V <sub>CCI</sub> B3	VMV5
N9	V <sub>CC</sub>	V <sub>CCI</sub> B5
N10	GND	V <sub>CCI</sub> B5
N11	GND	IO84NDB5V0
N12	GND	IO84PDB5V0
N13	GND	V <sub>CCI</sub> B4
N14	V <sub>CC</sub>	V <sub>CCI</sub> B4
N15	V <sub>CCI</sub> B1	VMV3
N16	IO95NPB1	V <sub>CC</sub> PLD
N17	IO100NPB1	GDB1/IO66PPB3V1
N18	IO102NDB1	GDC1/IO65PDB3V1
N19	IO102PDB1	IO61NDB3V1
N20	NC	V <sub>CC</sub>
N21	IO101NPB1	IO59NDB3V0
N22	IO103PDB1	IO62PDB3V1
P1	NC	NC
P2	IO199PDB3	IO110NDB6V0
P3	IO199NDB3	NC
P4	IO202NDB3	IO105PDB6V0

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
P5	IO202PDB3	IO105NDB6V0
P6	IO196PPB3	GEC1/IO104PPB6V0
P7	IO193PPB3	V <sub>COMPLE</sub>
P8	V <sub>CCI</sub> B3	GNDQ
P9	GND	GEA2/IO101PPB5V2
P10	V <sub>CC</sub>	IO92NDB5V1
P11	V <sub>CC</sub>	IO90NDB5V1
P12	V <sub>CC</sub>	IO82NDB5V0
P13	V <sub>CC</sub>	IO74NDB4V1
P14	GND	IO74PDB4V1
P15	V <sub>CCI</sub> B1	GNDQ
P16	GDB0/IO112NPB1	V <sub>COMPLD</sub>
P17	IO106NDB1	V <sub>JTAG</sub>
P18	IO106PDB1	GDC0/IO65NDB3V1
P19	IO107PDB1	GDA1/IO67PDB3V1
P20	NC	NC
P21	IO104PDB1	IO64PDB3V1
P22	IO103NDB1	IO62NDB3V1
R1	NC	NC
R2	IO197PPB3	IO107PDB6V0
R3	V <sub>CC</sub>	IO107NDB6V0
R4	IO197NPB3	GEB1/IO103PDB6V0
R5	IO196NPB3	GEB0/IO103NDB6V0
R6	IO193NPB3	VMV6
R7	GEC0/IO190NPB3	V <sub>CCPLE</sub>
R8	VMV3	IO101NPB5V2
R9	V <sub>CCI</sub> B2	IO95PPB5V1
R10	V <sub>CCI</sub> B2	IO92PDB5V1

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
R11	IO147RSB2	IO90PDB5V1
R12	IO136RSB2	IO82PDB5V0
R13	V <sub>CCI</sub> B2	IO76NDB4V1
R14	V <sub>CCI</sub> B2	IO76PDB4V1
R15	VMV2	VMV4
R16	IO110NDB1	TCK
R17	GDB1/IO112PPB1	V <sub>PUMP</sub>
R18	GDC1/IO111PDB1	TRST
R19	IO107NDB1	GDA0/IO67NDB3V1
R20	V <sub>CC</sub>	NC
R21	IO104NDB1	IO64NDB3V1
R22	IO105PDB1	IO63PDB3V1
T1	IO198PDB3	NC
T2	IO198NDB3	NC
T3	NC	GND
T4	IO194PPB3	GEA1/IO102PDB6V0
T5	IO192PPB3	GEA0/IO102NDB6V0
T6	GEC1/IO190PPB3	GNDQ
T7	IO192NPB3	GEC2/IO99PDB5V2
T8	GNDQ	IO95NPB5V1
T9	GEA2/IO187RSB2	IO91NDB5V1
T10	IO161RSB2	IO91PDB5V1
T11	IO155RSB2	IO83NDB5V0
T12	IO141RSB2	IO83PDB5V0
T13	IO129RSB2	IO77NDB4V1
T14	IO124RSB2	IO77PDB4V1
T15	GNDQ	IO69NDB4V0
T16	IO110PDB1	GDB2/IO69PDB4V0



Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
T17	V <sub>JTAG</sub>	TDI
T18	GDC0/IO111NDB1	GNDQ
T19	GDA1/IO113PDB1	TDO
T20	NC	GND
T21	IO108PDB1	NC
T22	IO105NDB1	IO63NDB3V1
U1	IO195PDB3	NC
U2	IO195NDB3	NC
U3	IO194NPB3	NC
U4	GEB1/IO189PDB3	GND
U5	GEB0/IO189NDB3	IO100NDB5V2
U6	VMV2	GEB2/IO100PDB5V2
U7	IO179RSB2	IO99NDB5V2
U8	IO171RSB2	IO88NDB5V0
U9	IO165RSB2	IO88PDB5V0
U10	IO159RSB2	IO89NDB5V0
U11	IO151RSB2	IO80NDB4V1
U12	IO137RSB2	IO81NDB4V1
U13	IO134RSB2	IO81PDB4V1
U14	IO128RSB2	IO70NDB4V0
U15	VMV1	GDC2/IO70PDB4V0
U16	TCK	IO68NDB4V0
U17	V <sub>PUMP</sub>	GDA2/IO68PDB4V0
U18	TRST	TMS
U19	GDA0/IO113NDB1	GND
U20	NC	NC
U21	IO108NDB1	NC
U22	IO109PDB1	NC

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

<b>Pin Number</b>	<b>M7A3P1000 Function</b>	<b>M7A3PE600 Function</b>
V1	NC	V <sub>CCI</sub> B6
V2	NC	NC
V3	GND	NC
V4	GEA1/IO188PDB3	IO98NDB5V2
V5	GEA0/IO188NDB3	GND
V6	IO184RSB2	IO94NDB5V1
V7	GEC2/IO185RSB2	IO94PDB5V1
V8	IO168RSB2	V <sub>CC</sub>
V9	IO163RSB2	V <sub>CC</sub>
V10	IO157RSB2	IO89PDB5V0
V11	IO149RSB2	IO80PDB4V1
V12	IO143RSB2	IO78NPB4V1
V13	IO138RSB2	NC
V14	IO131RSB2	V <sub>CC</sub>
V15	IO125RSB2	V <sub>CC</sub>
V16	GDB2/IO115RSB2	NC
V17	TDI	NC
V18	GNDQ	GND
V19	TDO	NC
V20	GND	NC
V21	NC	NC
V22	IO109NDB1	V <sub>CCI</sub> B3
W1	NC	GND
W2	IO191PDB3	V <sub>CCI</sub> B6
W3	NC	NC
W4	GND	IO98PDB5V2
W5	IO183RSB2	IO96NDB5V2
W6	GEB2/IO186RSB2	IO96PDB5V2

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

Pin Number	M7A3P1000 Function	M7A3PE600 Function
W7	IO172RSB2	IO86NDB5V0
W8	IO170RSB2	IO86PDB5V0
W9	IO164RSB2	IO85PDB5V0
W10	IO158RSB2	IO85NDB5V0
W11	IO153RSB2	IO78PPB4V1
W12	IO142RSB2	IO79NDB4V1
W13	IO135RSB2	IO79PDB4V1
W14	IO130RSB2	NC
W15	GDC2/IO116RSB2	NC
W16	IO120RSB2	IO71NDB4V0
W17	GDA2/IO114RSB2	IO71PDB4V0
W18	TMS	NC
W19	GND	NC
W20	NC	NC
W21	NC	V <sub>CCI</sub> B3
W22	NC	GND
Y1	V <sub>CCI</sub> B3	GND
Y2	IO191NDB3	GND
Y3	NC	V <sub>CCI</sub> B5
Y4	IO182RSB2	IO97NDB5V2
Y5	GND	IO97PDB5V2
Y6	IO177RSB2	IO93NDB5V1
Y7	IO174RSB2	IO93PDB5V1
Y8	V <sub>CC</sub>	IO87NDB5V0
Y9	V <sub>CC</sub>	IO87PDB5V0
Y10	IO154RSB2	NC
Y11	IO148RSB2	NC
Y12	IO140RSB2	IO75NDB4V1

Table A-1. M7A3PE600 and M7A3P1000 FG484 Package Connections (Continued)

<b>Pin Number</b>	<b>M7A3P1000 Function</b>	<b>M7A3PE600 Function</b>
Y13	NC	IO75PDB4V1
Y14	V <sub>CC</sub>	IO72NDB4V0
Y15	V <sub>CC</sub>	IO72PDB4V0
Y16	NC	IO73NDB4V0
Y17	NC	IO73PDB4V0
Y18	GND	NC
Y19	NC	NC
Y20	NC	V <sub>CCI</sub> B4
Y21	NC	GND
Y22	V <sub>CCI</sub> B1	GND

---

## Board Schematics

This appendix provides illustrations of the CoreMP7 Evaluation Board.

### Top-Level View

Figure B-1 on page 126 illustrates a top-level view of the CoreMP7 Evaluation Board. Figure B-2 on page 127 shows a bottom-level view of CoreMP7 Evaluation Board.

### CoreMP7 Schematics

The rest of this appendix shows the following illustrations of the CoreMP7 Evaluation Board:

Figure B-3 on page 128: Main 1.5 V, 2.5 V, and 3.3 V Power Supplies

Figure B-4 on page 129: Flash and Synchronous SRAM Memories

Figure B-5 on page 130: M7A3PE600 FPGA – I/O Banks 0–2

Figure B-6 on page 131: M7A3PE600 FPGA – I/O Banks 3–5

Figure B-7 on page 132: M7A3PE600 FPGA – I/O Banks 6–7

Figure B-8 on page 133: LEDs and Push-Button Switches

Figure B-9 on page 134: FPGA I/O Expansion Headers

Figure B-10 on page 135: USB Interface

Figure B-11 on page 136: RS-232 and CAN Interfaces

Figure B-12 on page 137: Ethernet 0 Interface

Figure B-13 on page 138: Ethernet 1 Interface



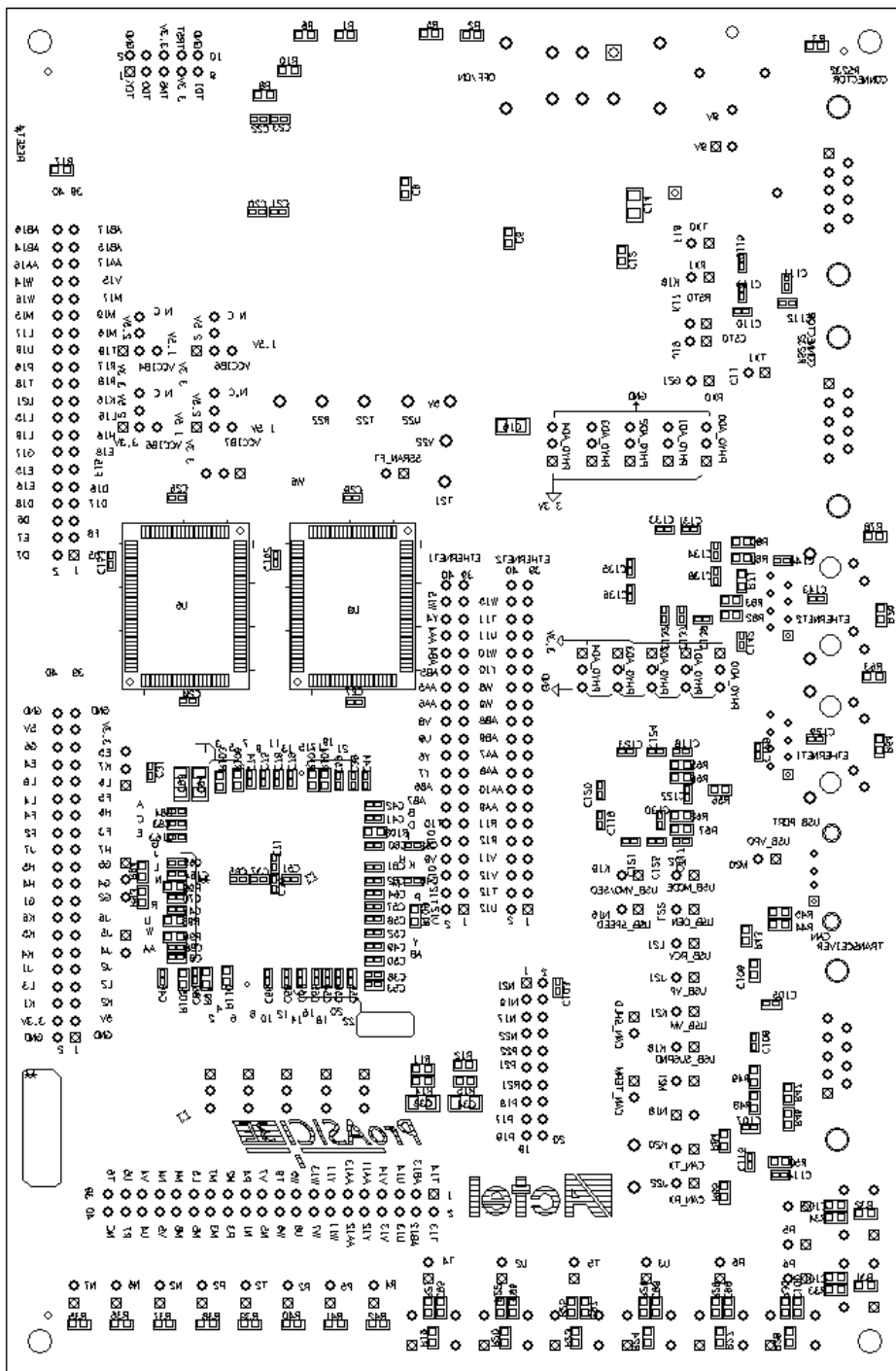
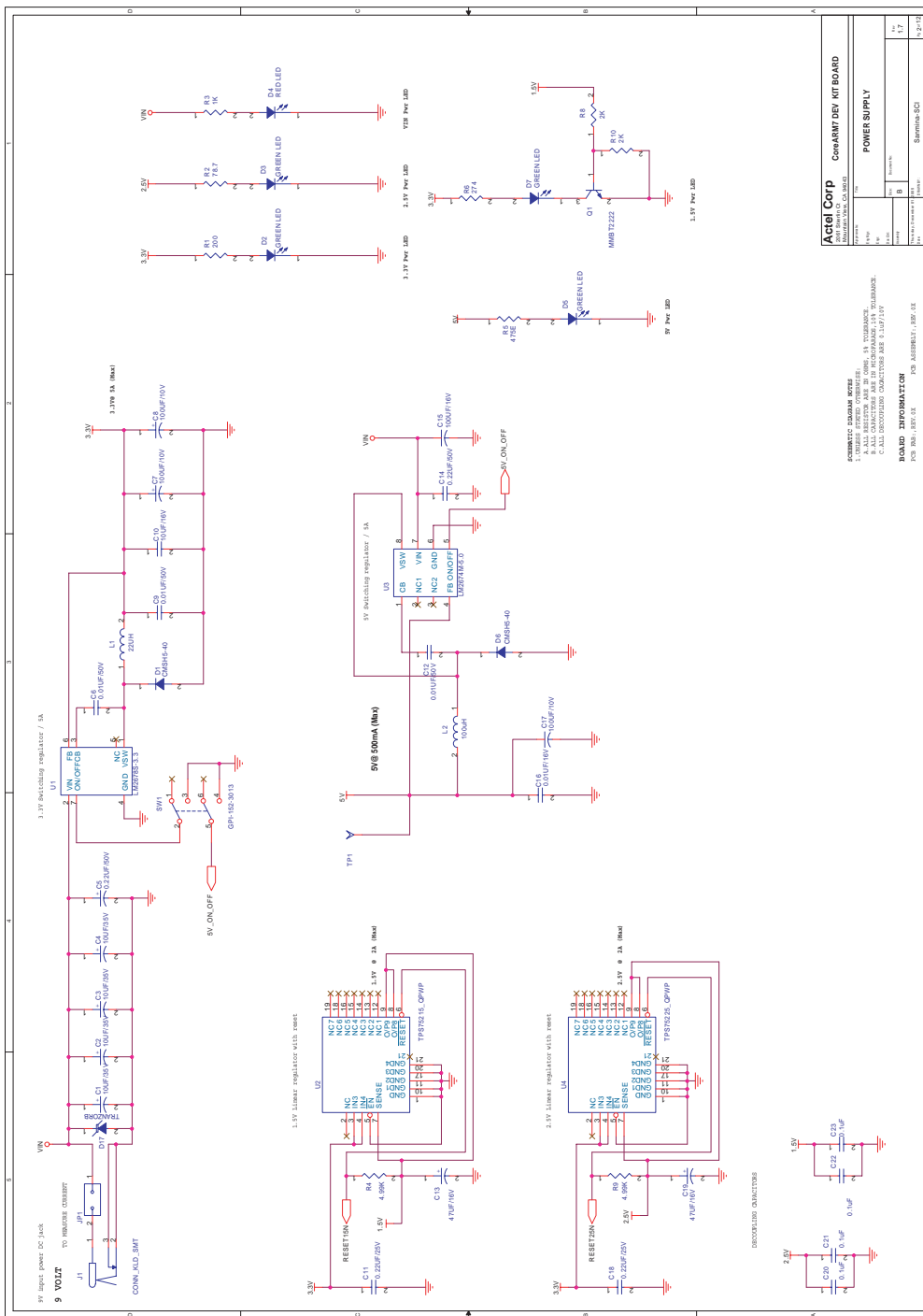
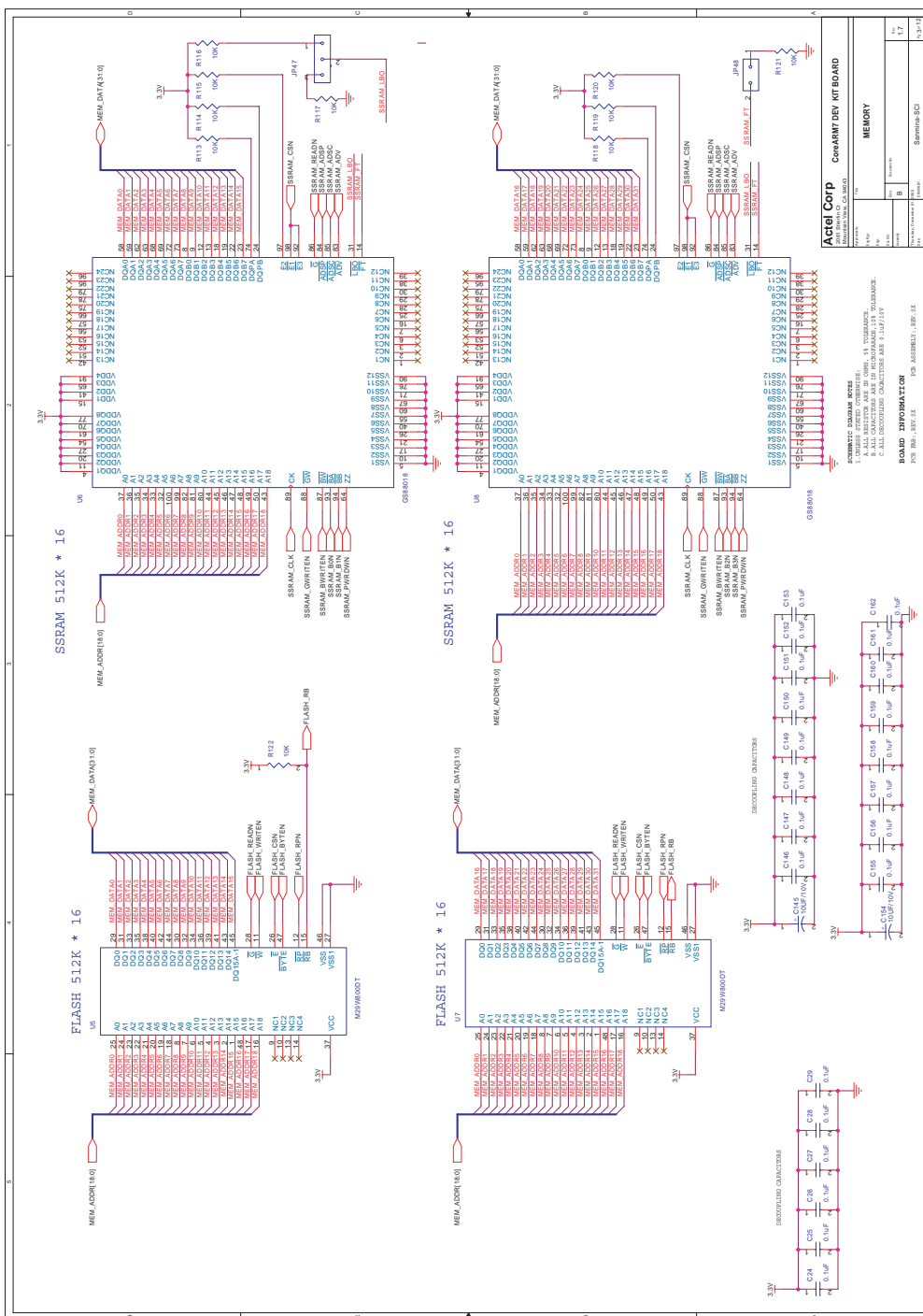


Figure B-2. Bottom-Level View of CoreMP7 Development Board



### Figure B-3. Main 1.5 V, 2.5 V, and 3.3 V Power Supplies







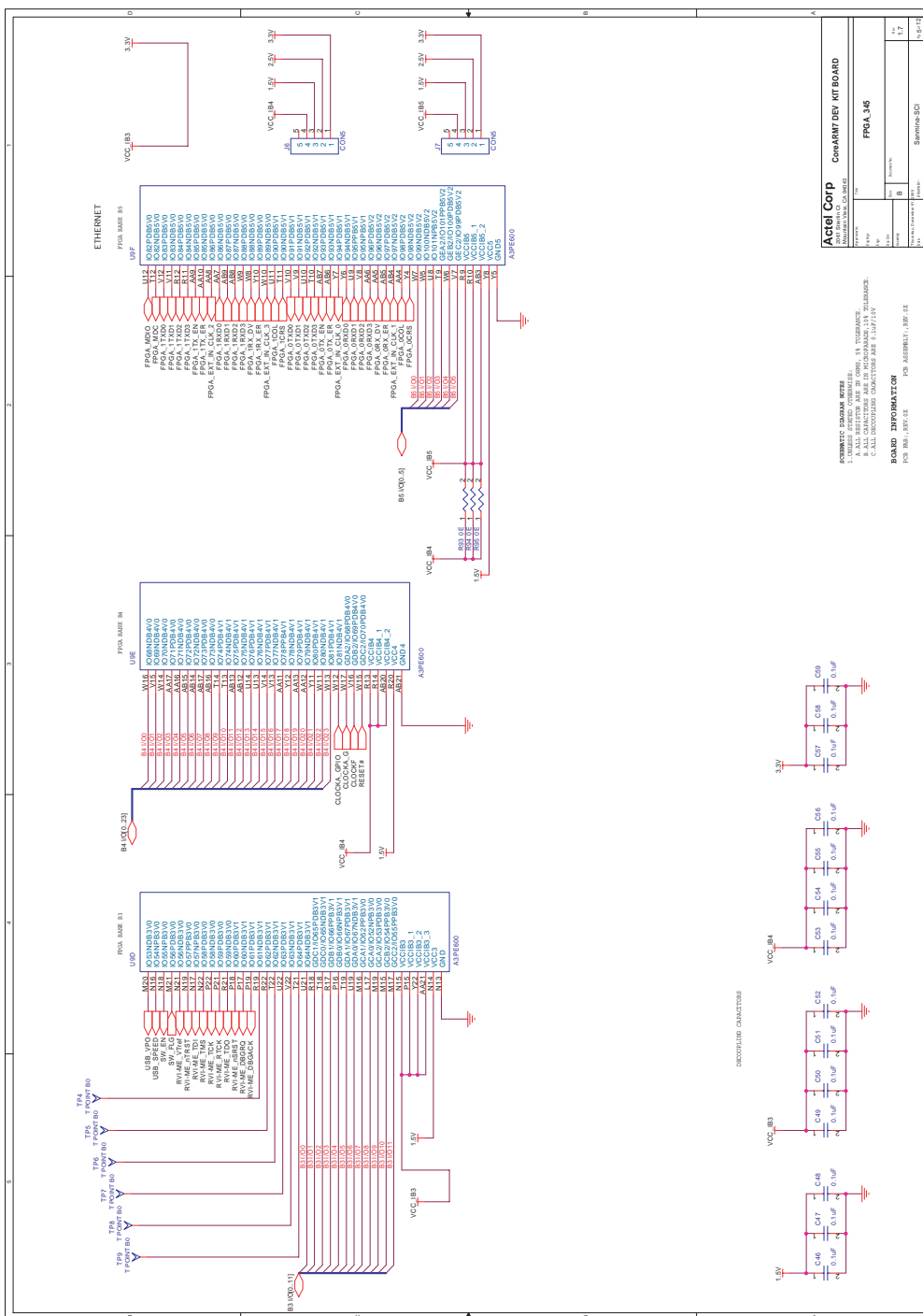
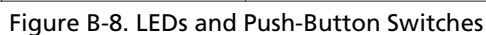
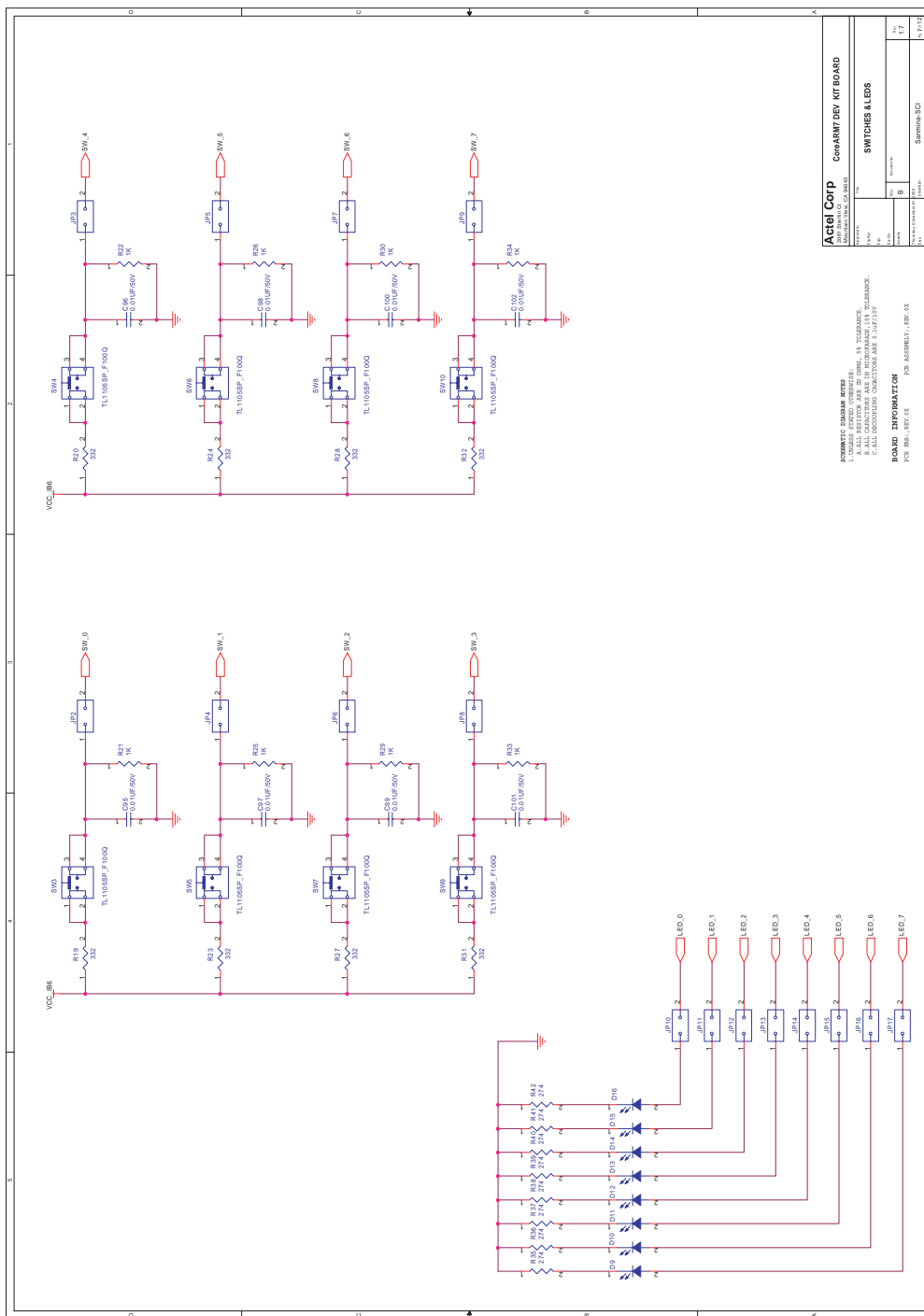


Figure B-6. M7A3PE600 FPGA – I/O Banks 3–5







### Figure B-9. FPGA I/O Expansion Headers



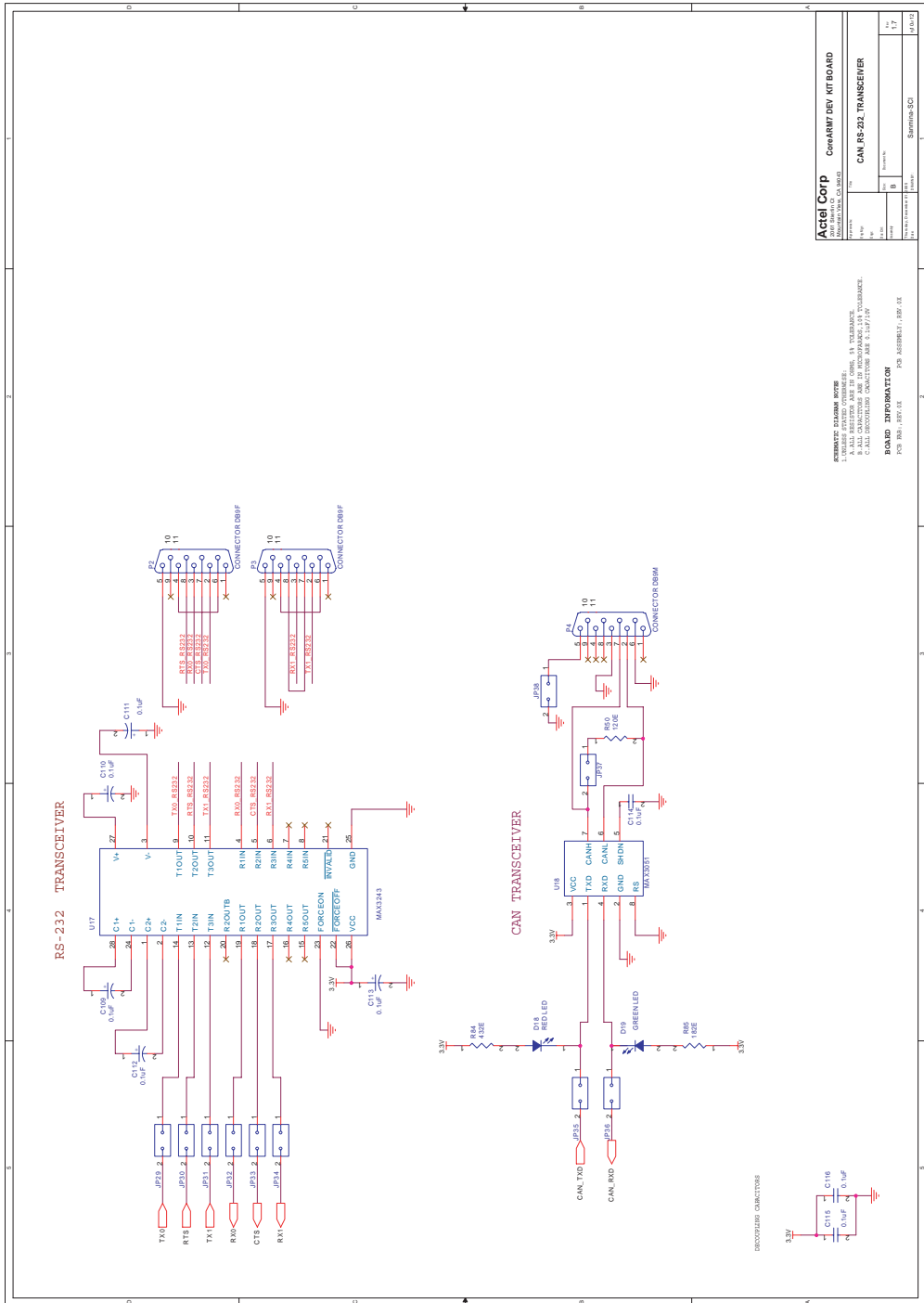
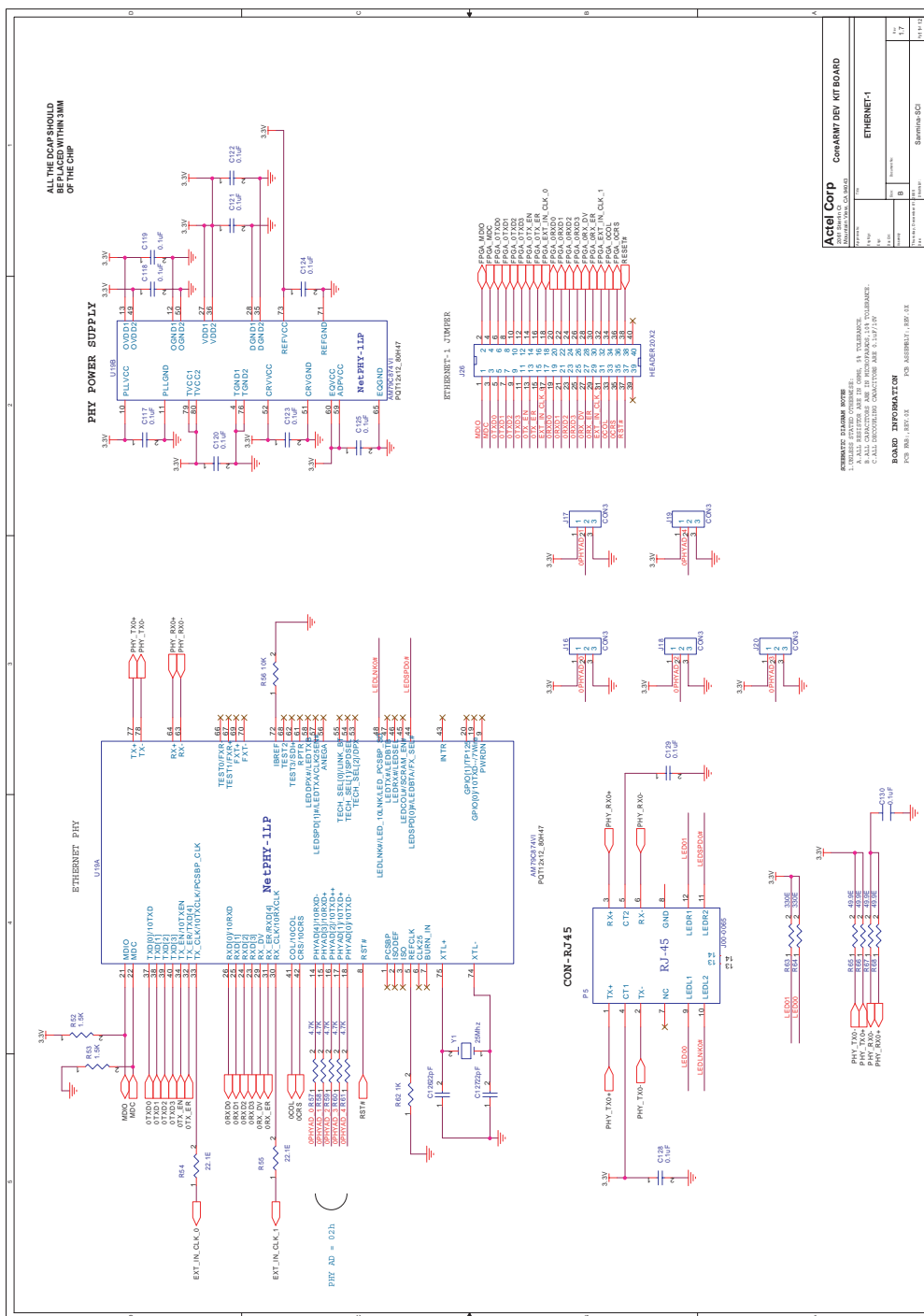


Figure B-11. RS-232 and CAN Interfaces





### Figure B-12. Ethernet 0 Interface

### Figure B-13. Ethernet 1 Interface

---

# Signal Layers

The CoreMP7 Evaluation Board is a six-layer board. The board has the following copper layers:

[Figure C-1 on page 140](#): Layer 1 – Top Signal Layer

[Figure C-2 on page 141](#): Layer 2 – Ground Plane

[Figure C-3 on page 142](#): Layer 3 – Signal Layer 3

[Figure C-4 on page 143](#): Layer 4 – Signal Layer 4

[Figure C-5 on page 144](#): Layer 5 – Power Plane

[Figure C-6 on page 145](#): Layer 6 – Bottom Signal Layer

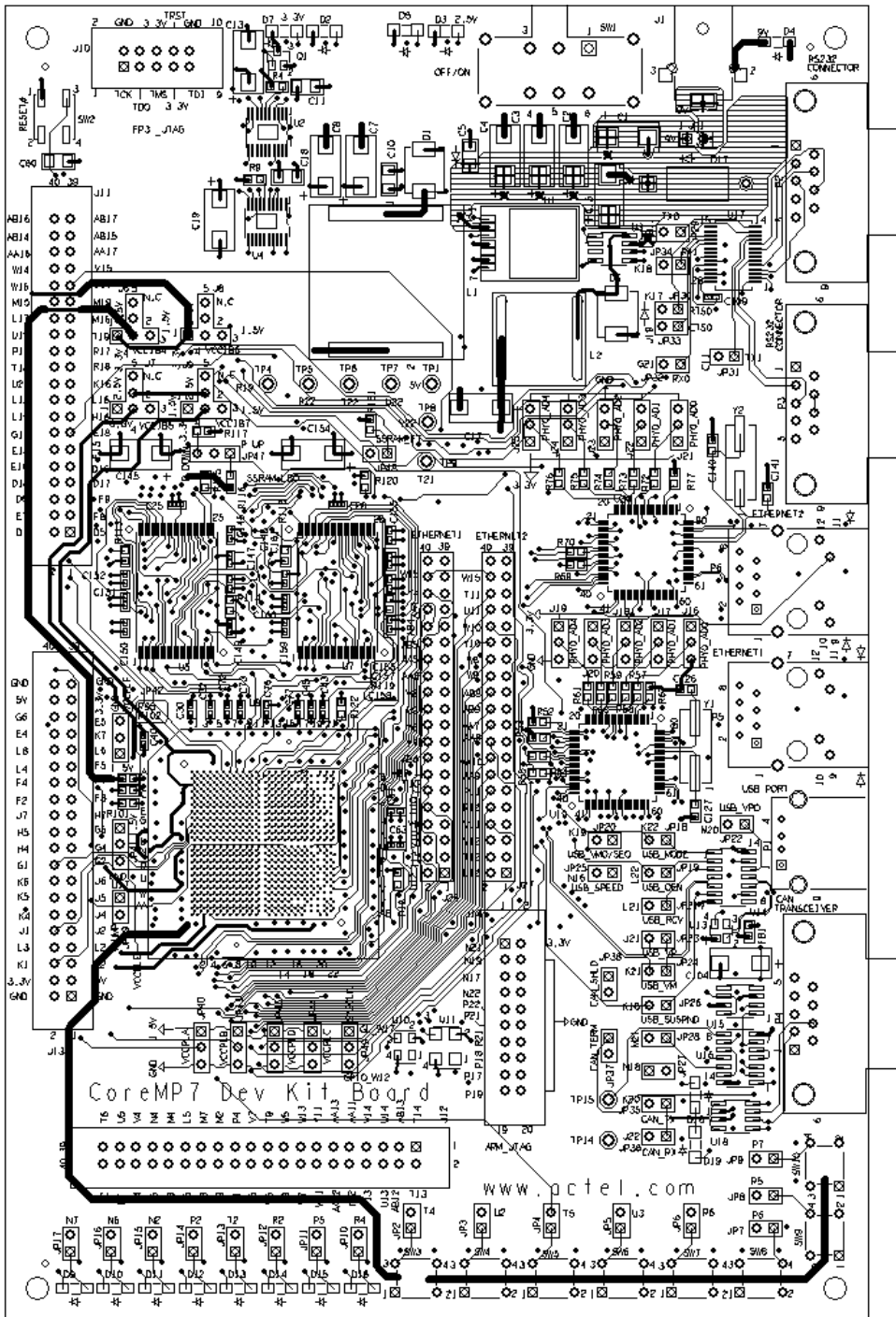


Figure C-1. Layer 1 – Top Signal Layer

Figure C-2. Layer 2 – Ground Plane (Blank)

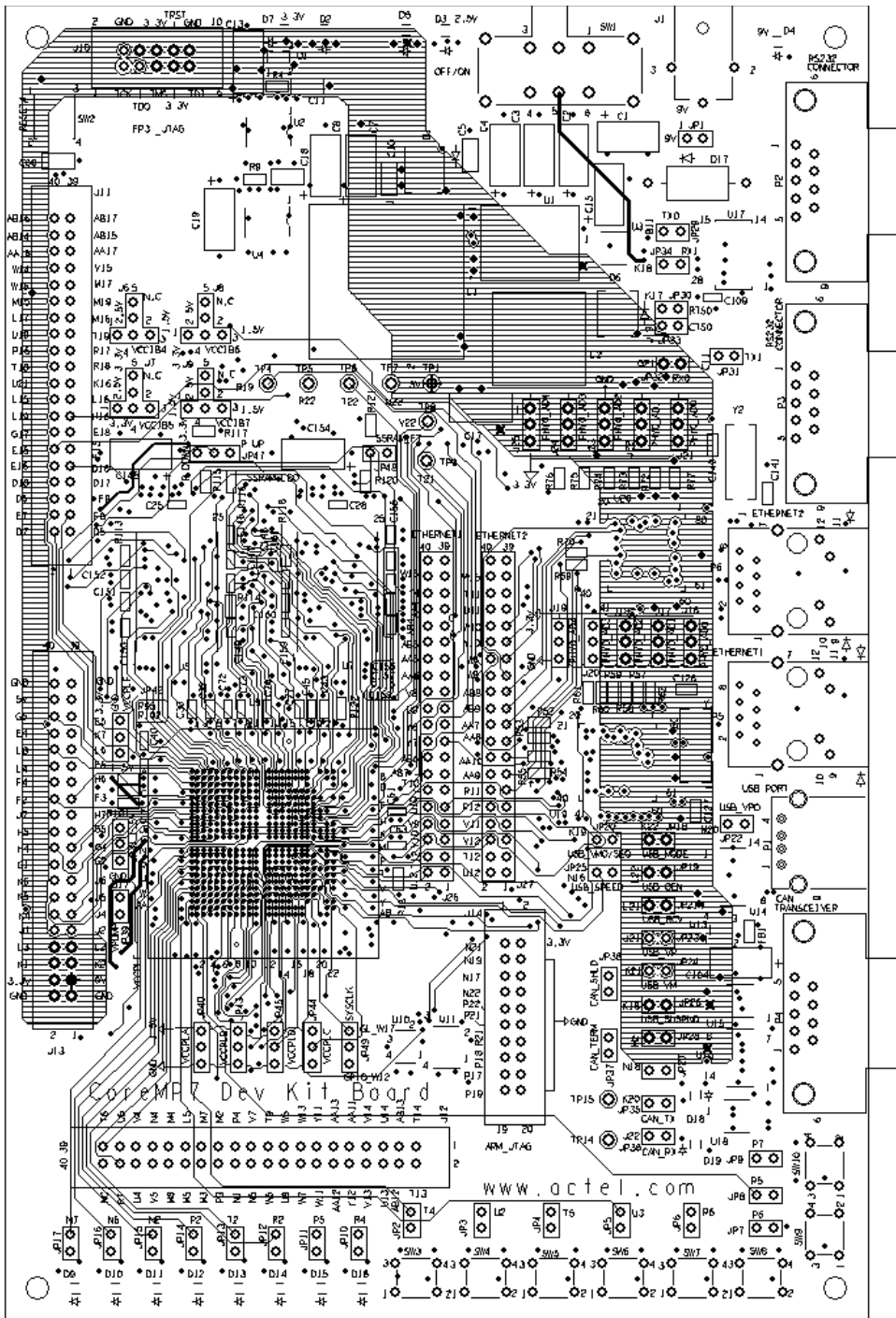


Figure C-3. Layer 3 – Signal Layer 3

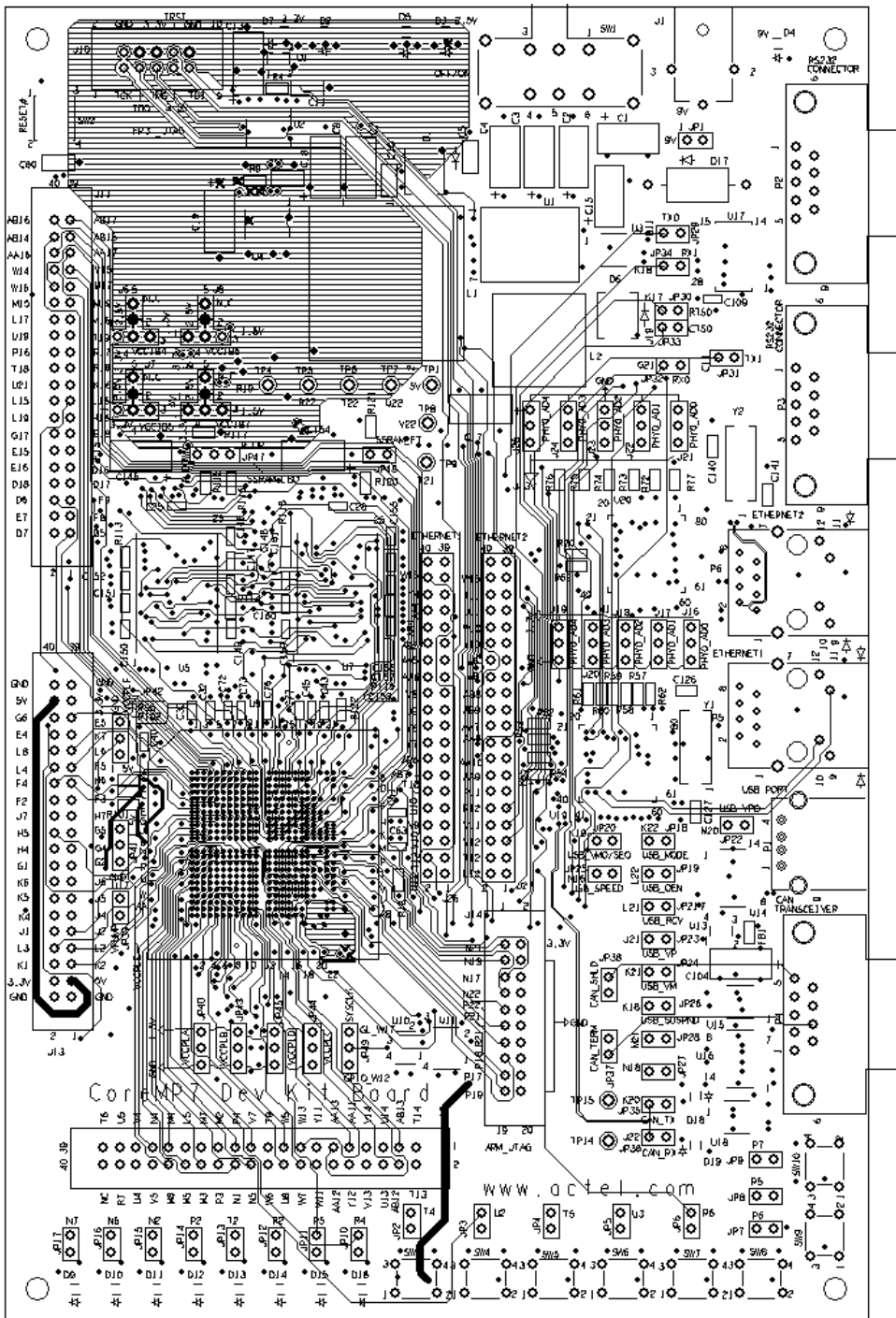


Figure C-4. Layer 4 – Signal Layer 4

Figure C-5. Layer 5 – Power Plane (Blank)



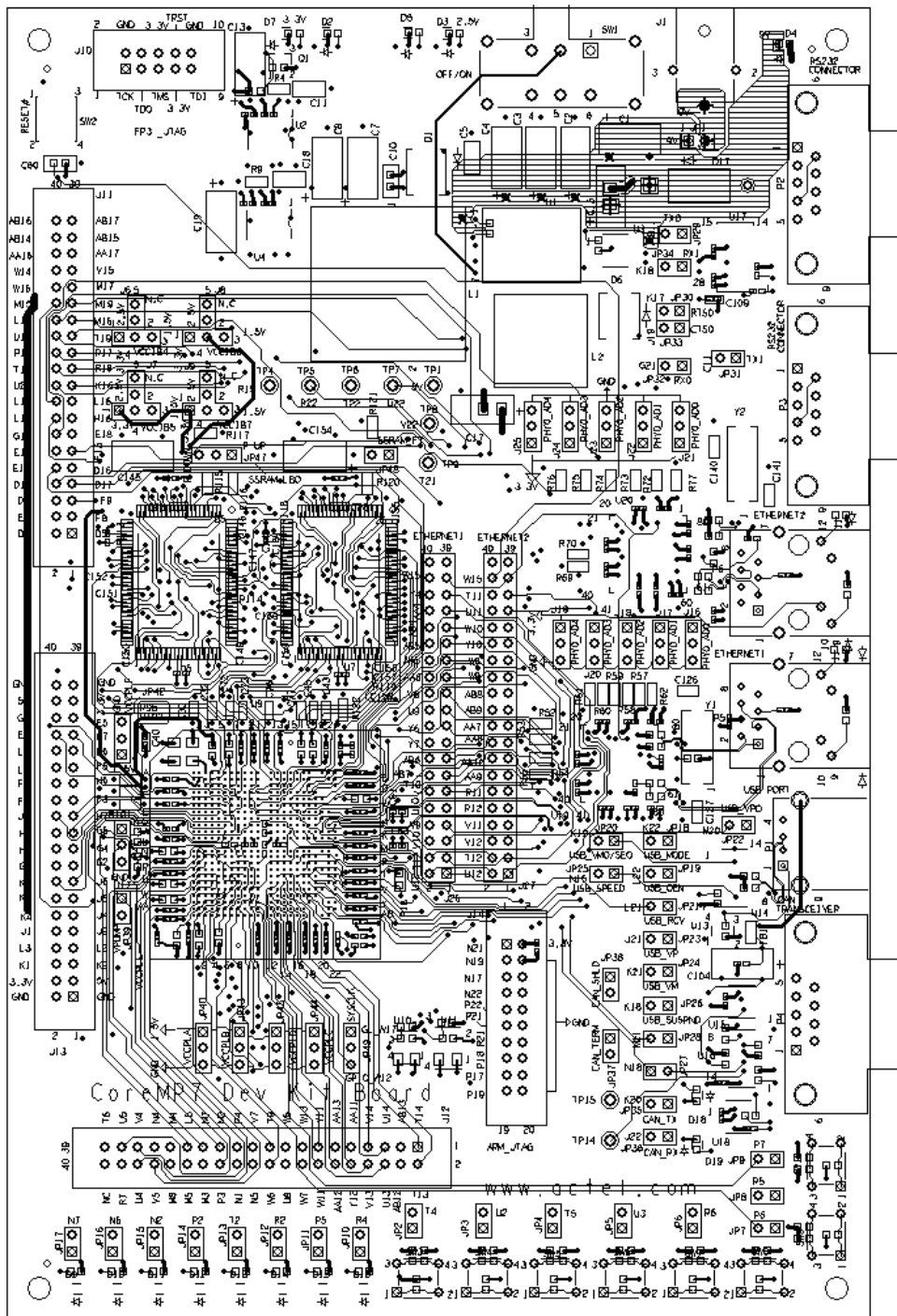


Figure C-6. Layer 6 – Bottom Signal Layer



---

# Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

## Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**

From Southeast and Southwest U.S.A., call **650.318.4480**

From South Central U.S.A., call **650.318.4434**

From Northwest U.S.A., call **650.318.4434**

From Canada, call **650.318.4480**

From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**

From Japan, call **650.318.4743**

From the rest of the world, call **650.318.4743**

Fax, from anywhere in the world **650.318.8044**

## Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

## Actel Technical Support

Visit the [Actel Customer Support website \(www.actel.com/custsup/search.html\)](http://www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

## Website

You can browse a variety of technical and non-technical information on Actel's [home page](http://www.actel.com), at [www.actel.com](http://www.actel.com).

## Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

### Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is [tech@actel.com](mailto:tech@actel.com).

### Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

**650.318.4460**

**800.262.1060**

Customers needing assistance outside the US time zones can either contact technical support via email ([tech@actel.com](mailto:tech@actel.com)) or contact a local sales office. [Sales office listings](#) can be found at [www.actel.com/contact/offices/index.html](http://www.actel.com/contact/offices/index.html).

---

# Index

10/100 Ethernet 19

## A

Actel

- electronic mail 148
- telephone 148
- web-based technical support 147
- website 147

assumptions 5

## B

board

- CAN 21
- clocks 21
- description 9
- Ethernet 19
- headers 22
- jumpers 15
- layers 22
- LEDs 16
- memory 22
- PLLs 11
- power supplies 12
  - block diagram 12
- programming 14
- RS-232 18
- schematics 125
- self test 25
  - programming 25
- switches 17
- test points 22
- testing 25
  - programming 25
- top-level view 10
- usage 9
- USB 20

## C

CAN 21

clocks 21

CompanionCore 21

contacting Actel

- customer service 147

- electronic mail 148

- telephone 148

- web-based technical support 147

Core10/100 19

CoreMP7 evaluation board 9

CoreUART 18

customer service 147

## D

design flow 27

- design creation 29

- implementation 30

- microprocessor 31

- programming 31

- system creation 27

- verification 29

development kit contents 7

## E

Ethernet 19

example design 14

## F

FPGA package connections 105

## H

hardware 9

- description 9

- installation 25

headers 22

## ***J***

jumpers 15

## ***K***

kit contents 7

## ***L***

LEDs 16

## ***M***

memory 22

## ***P***

PLLs 11

power supplies 12

    block diagram 12

product support 147–148

    customer service 147

    electronic mail 148

    technical support 147

    telephone 148

    website 147

programming 14

## ***R***

RS-232 18

## ***S***

schematics 125

self test 25

    programming 25

setup 25

software

    installation 25

switches 17

system requirements 7

## ***T***

technical support 147

test points 22

testing 25

    programming 25

## ***U***

USB 20

## ***W***

web-based technical support 147



***For more information about Actel's products, visit our website at  
<http://www.actel.com>***

***Actel Corporation*** • 2061 Stierlin Court • Mountain View, CA 94043 USA  
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

***Actel Europe Ltd.*** • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL • United Kingdom  
Phone +44 (0) 1276 401 450 • Fax +44 (0) 1276 401 490

***Actel Japan*** • EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan  
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • [www.jp.actel.com](http://www.jp.actel.com)

***Actel Hong Kong*** • Suite 2114, Two Pacific Place • 88 Queensway, Admiralty Hong Kong  
Phone +852 2185 6460 • Fax +852 2185 6488 • [www.actel.com.cn](http://www.actel.com.cn)

50200075-0/8.06

