XILINX®

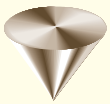
ONLINE

HARDWARE & PERIPHERALS

USER GUIDE

TABLE OF CONTENTS

INDEX



GO TO OTHER BOOKS

XSTEP™

0401305

Contents

Chapter 1 FPGA Design Demonstration Board

Chapter 3 XChecker Cable and Logic Probe

FPGA Design Demonstration Board

The FPGA Demonstration Board supports the following Xilinx FPGAs:

- XC2000, XC2000L
- XC3000, XC3100, XC3000A, XC3000L
- XC3100A
- XC4000, XC4000A, XC4000H
- XC5200

The FPGA Demonstration Board is a stand-alone board for experimenting and developing prototypes with FPGAs using Xilinx FPGA architecture. The FPGA Demonstration Board allows you to become familiar with all the Xilinx FPGA device families and the XACT*step* Development System.

The FPGA Demonstration Board comes with an XC3020APC68 and an XC4003APC84 part. You can configure the demonstration board either with the XChecker™ cable (slave serial mode) or the onboard 17XXX (master serial mode). It has the following features:

- One socket for an XC2000/XC3000 device
- One socket for an XC4000 device
- One 17XXX socket for each FPGA
- An XChecker/Download cable header for each FPGA
- Daisy-chain configuration with the XC4000 device at the head of the chain
- 8 DIP switches to set up the XC4000 and XC2000/XC3000 FPGAs, as shown in Table 1-1.

Table 1-1 DIP Switch Configuration

XC2000/ XC3000	XC4000 (SW2	Switch
INP	PWR	1
MPE	MPE (multiple configurations)	2
SPE	SPE (single configuration)	3
M0	M0	4
M1	M1	5
M2	M2	6
MCLK	RST	7
DOUT	INIT	8

- 16 I/O lines that connect the two FPGAs
- An external relaxation oscillator for the XC2000/XC3000
- The XC4000 OSC4 library symbol, which uses pin 19 of the XC4003A to drive the XC3000 TCLKIN on pin 11 of the XC3020A
- The XC4000 OSC4,which uses pin 13 to drive the XC2000/XC3000 alternate clock buffer (BCLKIN) on pin 43
- 8 DIP switches which set logic input levels; switch outputs, which drive both FPGAs; closing switches, which drive signals to logic 1's
- Program, Reset, and Spare Pushbutton switches, which are common to both FPGAs
- XC2000/XC3000 displays that use eight LED bars in one row and one 7-segment LED (in Figure 1-1)
- XC4000 displays that use eight LED bars in one row and two 7-segment LEDs, shown in Figure 1-1
- Space for an optional +5-V regulator for battery operation
- Space for an optional crystal oscillator
- Headers for FPGA probe points
- Prototype area on PC board

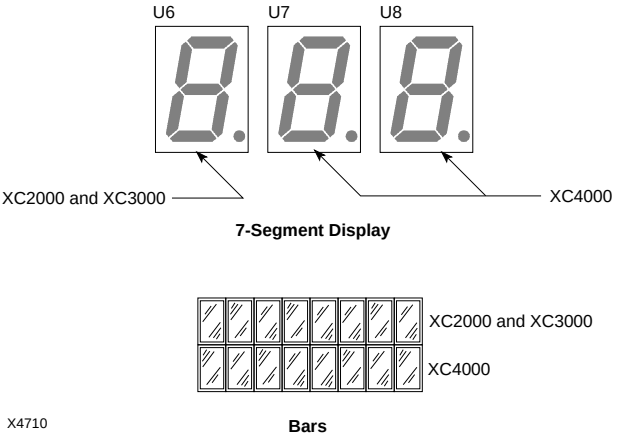
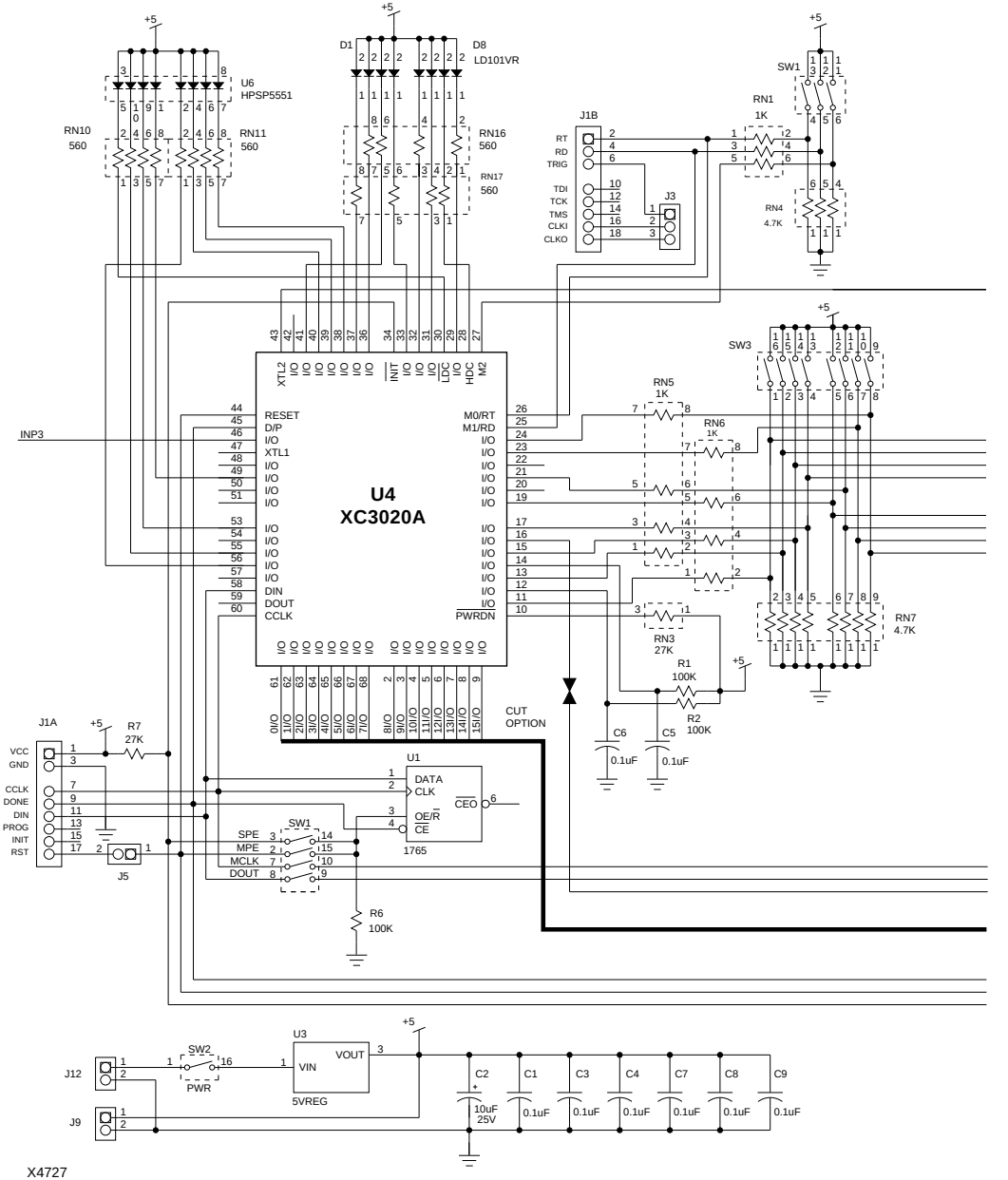
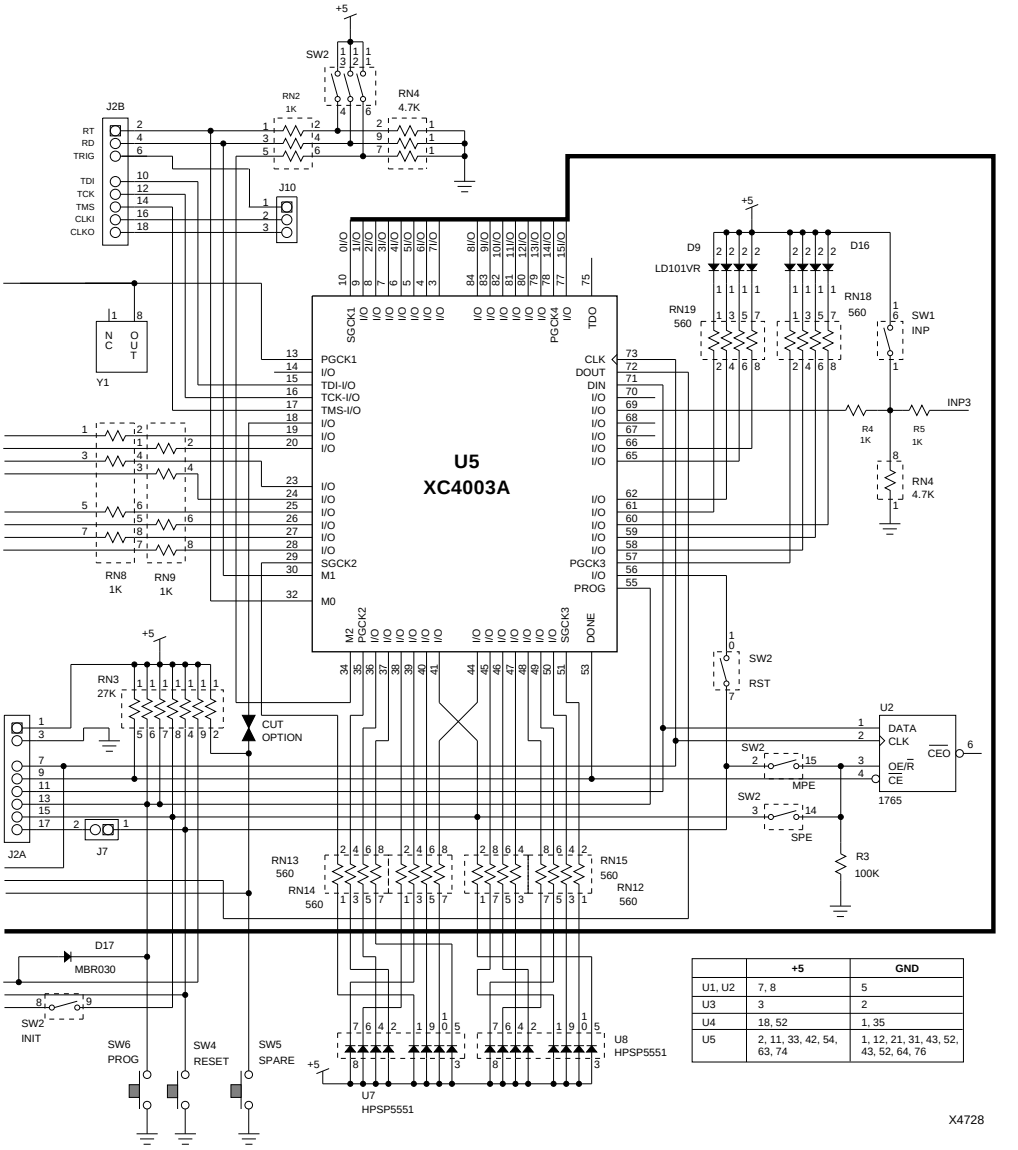


Figure 1-1 FPGA Demonstration Board Displays

FPGA Demonstration Board Components

Figure 1-2 shows the schematic of the FPGA Demonstration Board. Figure 1-3 shows the component layout of the FPGA Demonstration Board. Descriptions of the important board components and their board reference designator follow. General components are listed first, then the XC4003A components, followed by the XC3020A components.


U4
XC3020A
RESET
D/P
XTL1
DIN
DOUT
CCLK
M0/RT
M1/RD
PWRDN
CUT
OPTION
INP3
U6
HPSP5551
RN10
560
RN11
560
D1
D8
LD101VR
RN16
560
RN17
560
J1B
RT
RD
TRIG
TDI
TCK
TMS
CLKI
CLKO
J3
SW1
RN1
1K
RN4
4.7K
SW3
RN5
1K
RN6
1K
RN7
4.7K
RN3
27K
R1
100K
R2
100K
C6
C5
0.1uF
J1A
VCC
GND
CCLK
DONE
DIN
PROG
INIT
RST
R7
27K
J5
U1
DATA
CLK
CEO
OE/R
CE
1765
SPE
MPE
MCLK
DOUT
R6
100K
J12
J9
SW2
PWR
U3
VIN
VOUT
5VREG
C2
10uF
25V
C1
C3
C4
C7
C8
C9
0.1uF
X4727



	+5	GND
U1, U2	7, 8	5
U3	3	2
U4	18, 52	1, 35
U5	2, 11, 33, 42, 54, 63, 74	1, 12, 21, 31, 43, 52, 64, 76

Figure 1-2 FPGA Demonstration Board Schematic

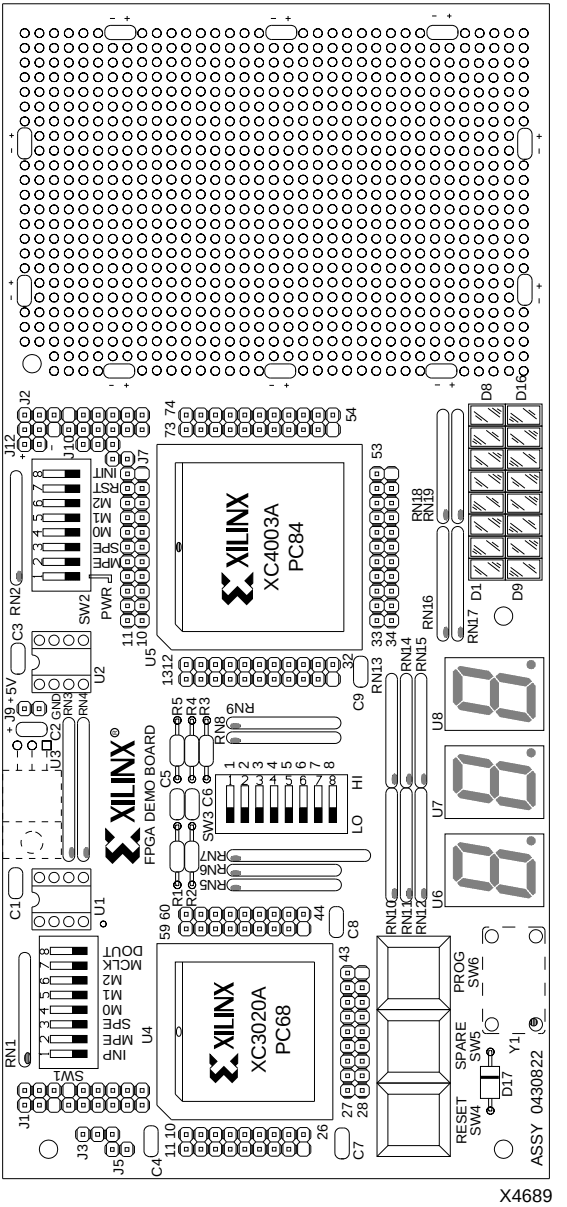


Figure 1-3 FPGA Demonstration Board Component Layout

FPGA Board General Components

This section describes the general and common components that are found on the FPGA Demonstration Board.

+5-V Power Connector (J9)

A regulated +5-volts and ground connected to the FPGA Demonstration Board through connector J9. Pin 1 (square pad) is +5 V and pin 2 is ground. The power supply should provide at least 250 mA of current to drive the LED displays.

Unregulated Power Input (J12)

The unregulated power input provides a way to power the FPGA Demonstration Board from an unregulated source, such as a 9-V battery or an a.c. adapter. The input should be 7VDC - 12VDC at 250 mA, typically. You must consider the power dissipation requirements of the U3 voltage regulator if the voltage input is greater than 9V.

The J12 unregulated power input provides two holes to connect the unregulated power source. The hole with the square pad, marked with a "+" is the positive input. The other hole, marked with a "-" is circuit ground. The positive input is connected through the power on-off switch SW2–1 to U3–1, which is the optional +5-V regulator. U3 must be installed to use this input.

+5-V Regulator Option (U3)

You can install a three terminal +5-V regulator, such as the LM2940CT shown in Figure 1-4, powers the demonstration board from an unregulated power supply, such as a +9-V battery. Pin 1 (square pad) is V_{in}, pin 2 is ground, and pin 3 is +5-V out.

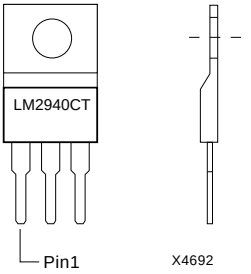


Figure 1-4 LM2940CT +5-V Regulator

RESET Pushbutton (SW4)

When you press the RESET pushbutton it can apply an active-Low Reset signal to the FPGAs and configuration PROMs, depending on how the Reset signal routing is configured. Reset is normally pulled High through a 27K-ohm resistor.

SPARE Pushbutton (SW5)

The SPARE pushbutton applies an active-Low signal to the XC3020A on pin 16, and to the XC4003A on pin 18. You can isolate these pins from the switch by using the trace-cut options on the solder side of the board. The trace-cut options appear as point-to-point triangles; the trace-cut option for the XC3020A is under its socket and the trace-cut option for the XC4003A is under R3. The SPARE signal is pulled High through a 27K-ohm resistor.

PROG Pushbutton (SW6)

The PROG pushbutton applies an active low signal to the DONE/PROGRAM input on the XC3020A FPGA socket at pin 45 and to the PROGRAM input on the XC4003A FPGA socket at pin 55. The PROG signal is normally pulled High through a 13.5K-ohm resistor.

Eight General-Purpose Input Switches (SW3)

Eight switches connect to eight general-purpose inputs on both the XC3020A and the XC4003A FPGAs. These switches provide logic input to the FPGAs. An FPGA input pin is set to a logic "1" when a

switch is on, and a logic "0" when a switch is off. See Figure 1-5.

The FPGA pins connected to this switch are intended for use as inputs; however, each FPGA pin has a 1k-ohm resistor that isolates it from the switch so it is possible to define them as outputs. It is also possible to drive them from an external source by connecting that signal to the FPGA probe point header. Table 1-2 lists the FPGA pin connections.

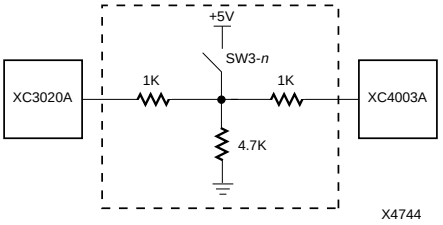


Figure 1-5 FPGA Demonstration Board General-Purpose Switch

Table 1-2 FPGA Pin Connections

Switch	XC3020A	XC4003A
SW3–1	11	19
SW3–2	13	20
SW3–3	15	23
SW3–4	17	24
SW3–5	19	25
SW3–6	21	26
SW3–7	23	27
SW3–8	24	28

7-Segment Displays (U6, U7, U8)

Three 7-segment displays, which are included with the leftmost display (U6) connect to the XC3020A FPGA, and the right two displays (U7 and U8) connect to the XC4003A.

Each LED segment is turned on by driving the corresponding FPGA pin 'LOW' with a logic '0.' The decimal point on U8 connects to the INIT pin of the XC4003A (pin 41), and serves as a programming error indicator. The decimal point should be on while the FPGA is in its internal clearing state, then it should remain off during configuration. If the decimal point comes back on, there has been a programming error.

The decimal points on U6 and U7 are tied to the LDC (Low during configuration) pins of the XC3020A and XC4003A, respectively. The decimal points are on while the FPGAs wait to be configured.

Table 1-3 shows the I/O pin definitions. Figure 1-6 shows the 7-segment display.

Table 1-3 7-Segment I/O Connections

Display Segment	XC3020A	XC4003A	XC4003A
	U6	U7	U8
a	38	39	49
b	39	38	48
c	40	36	47
d	56	35	46
e	49	29	45
f	53	40	50
g	55	44	51
decimal point	30	37	41

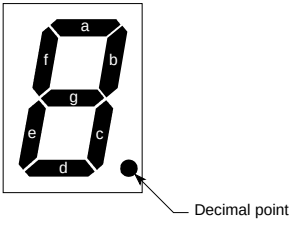


Figure 1-6 7-Segment Display

LED Indicators (D1-D8, D9-D16)

Eight LEDs connect to the I/O pins of each FPGA. D1 through D8 connect to the XC3020A, and D9 through D16 connect to the XC4003A. You can turn on an LED by driving its corresponding FPGA pin Low with a logic "0." Table 1-4 shows the pin connections for the LED indicators.

Table 1-4 LED Indicators for XC3020A and XC4003A

LED	XC3020A Pin	LED	XC4003A Pin
D1	37	D9	61
D2	36	D10	62
D3	41	D11	65
D4	33	D12	66
D5	32	D13	57
D6	31	D14	58
D7	28	D15	59
D8	29	D16	60

I/O Line Connections

There are 16 I/O lines that connect the XC3020A and XC4003A FPGAs, as shown in Table 1-5.

Table 1-5 I/O Line Connections for XC3020A and XC4003A Devices

I/O Line	XC3020A Pin	XC4003A Pin
0	61	10
1	62	9
2	63	8
3	64	7
4	65	6
5	66	5
6	67	4
7	68	3
8	2	84
9	3	83
10	4	82
11	5	81
12	6	80
13	7	79
14	8	78
15	9	77

Optional Crystal Oscillator (Y1)

You can add a standard 4-pin crystal oscillator to the FPGA Demonstration Board. The oscillator output drives the XC3020A XTL2 input at pin 43 and the XC4003A PGCK1 input at pin 13.

Prototype Area

The Prototype area is a 0.1-inch grid of holes where you can add additional circuitry to the demonstration board. A +5-V bus (component side) and a ground bus (solder side) are available on the perimeter of this area. There are also locations for filter capacitors.

XC4003A Components

This section describes the XC4003A components on the FPGA Demonstration Board.

XC4003A FPGA and Socket (U5)

The XC4003A FPGA occupies socket U5 on the demonstration board.

XC4003A Probe Points

All pins of the XC4003A connect to the headers that surround the FPGA socket. These pins provide convenient points for probing signals or making wirewrap connections to other circuitry, such as on the prototype area. Pin numbering increases from the inside row to the outside, counterclockwise. See the corners of each header for the starting number of that header.

XC4003A Configuration Switches (SW2)

The following sections describe each of the SW2 switches.

PWR — Power (SW2–1)

This switch turns the unregulated power input on or off to the +5-V regulator U3.

MPE — Multiple Program Enable (SW2-2)

With MPE turned on and SPE turned off, the configuration PROM (U2) is reset by the RESET pushbutton (SW4). Configuration mode must be set to master-serial. After a Reset or powerup, the first bitstream stored in the serial PROM is loaded into the XC4003A. Pressing RESET resets the serial PROM address pointer. Pressing PROG (SW6) loads the XC4003A with the first bitstream again. If you press PROG without pressing RESET, the XC4003A is loaded with the next bitstream that is stored in the serial PROM. The size of the serial PROM limits the number of bitstreams that can be sequentially loaded.

SPE — Single Program Enable (SW2-3)

With SPE turned on and MPE turned off, the configuration PROM (U2) is reset by the XC4003A's INIT output, which is driven Low whenever you press PROG (SW6). The first bitstream stored in the serial PROM is loaded into the XC4003A.

Note: MPE and SPE must not be on at the same time. MPE and SPE are only used in conjunction with the serial PROMs. The serial PROMs must be configured as OE/$\overline{\text{Reset}}$ to allow MPE and SPE to function properly.

M0, M1, M2 — Mode Pins (SW2-4,5,6)

These three switches must be on to configure the XC4003A using the XChecker/Download Cable. When these switches are on, the FPGA is in slave serial mode. To configure the XC4003A from the onboard serial PROM, these three switches must be off, placing the FPGA in master serial mode.

RST — Reset (SW2-7)

When this switch is on, it connects the RESET pushbutton (SW4) to XC4003A pin 56.

INIT — Initialize (SW2-8)

When this switch is on, it connects the XC3020A INIT pin to the XC4003A INIT pin. This connection is used to configure FPGAs in a daisy chain with the XC4003A at the head of the chain.

Note: INIT should only be used to configure FPGAs in a daisy chain.

XChecker/Download Cable Connector (J2)

Table 1-6 provides a detailed description of the J2 XChecker/ Download Cable connector.

Table 1-6 XChecker/Download Cable Connector (J2)

Pin	Name	Function	Pin	Name	Function
J2-1*	VCC	Supplies +5 V to XChecker cable.	J2-2	RT	Read Trigger allows XChecker to trigger a readback of the XC4003A. Connects to XC4003A pin 32.
J2-3*	GND	Supplies ground reference to XChecker cable.	J2-4	RD	Used by XChecker for readback data. Connects to XC4003A pin 30.
J2-5	N.C.		J2-6	TRIG	XChecker input that allows an external event to trigger readback of the XC4003A or output a burst of clocks to the XC4003A. Connects to tiepoint J10–1.
J2-7*	CCLK	Provides the clock during configuration or readback. Connects to XC4003A input pin 73.	J2-8	N.C.	
J2-9*	DONE	Indicates when configuration is complete. Connects to XC4003A output pin 53.	J2-10**	TDI	Inputs boundary-scan data to the XC4003A. Connects to XC4003A pin 15.
J2-11*	DIN	Provides configuration data during configuration. Connects to XC4003A DIN input pin 71.	J2-12**	TCK	Input boundary scan clock to the XC4003A. Connects to pin 16.
J2-13	PROG	Provides program pulse that causes the FPGA to configure. Connects to XC4003A PROG input pin 55.	J2-14**	TMS	Boundary scan mode input to the XC4003A. Connects to pin 17.

J2-15	PROG	Goes Low if CRC error occurs during configuration. Connects to XC4003A INIT pin 41.	J2-16	CLK1	A system clock input to XChecker to be controlled and output on CLK0. Connects to tiepoint J10-2.
J2-17	RST	Connects to jumper J7. If connected, allows XChecker to provide a Reset input (same as pressing the Reset button).	J2-18	CLK0	A system clock output controlled by XChecker. Used to single-step or burst clocks to the XC4003A. Connects to tiepoint J10-3.

The Download Cable supports pins marked with an asterisk (*).
Pins marked with a double asterisk (**) indicate boundary scan operations that XChecker does not support..

With the Download Cable connected, J2-9 provides both the DONE and PROG functions. Since the XC4003A requires a Program input that is separate from DONE, you must press the PROG button before configuring the XC4003A.

Jumper J7 and Tiepoints J10 (1-3)

Jumper J7 allows the XChecker signal RST on J2-17 to drive the reset line on the demonstration board. Tiepoint pins jumper the following XChecker signals into the circuit. Tiepoint J10-1 connects to TRIG on J2-6; Tiepoint J10-2 connects to CLK1 on J2-16; and, Tiepoint J10-3 connects to CLK0 on J2-18. See Table 1-6 for more details on the XChecker/Download Cable connections.

Serial PROM Socket (U2)

This serial PROM configures the XC4003A or the XC4003A and XC3020A connected in a daisy chain. The configuration mode must be the master serial mode to configure from the serial PROM.

XC3020A Components

This section describes the XC3020A components on the FPGA Demonstration Board.

XC3020A FPGA and Socket (U4)

The XC3020A FPGA occupies socket U4 on the demonstration board.

XC3020A Probe Points

All pins of the XC3020A FPGA connect to the headers that surround the FPGA socket. These pins provide convenient points for probing signals or making wirewrap connections to other circuitry, such as the prototype area. Pin numbering increases from the inside row to the outside, counterclockwise. See the corners of each header for the starting number of that header. Refer to Table 1-5 for information.

The XC3020A I/O pins 2 through 9 and 61 through 68 connect to XC4003A pins 3 through 10 and 77 through 84, respectively. The XC3020A pins share the XC4003A probe points header.

XC3020A Configuration Switches (SW1)

The following sections describe each of the SW1 switches.

INP — Input Switch (SW1-1)

This is an extra switch, which is connected to provide an extra logic input to the XC3020A pin 46 and the XC4003A pin 69. The FPGA input pins are set to a logic "1" when the switch is on and a logic "0" when the switch is off.

The FPGA pins connected to this switch are intended for use as inputs; however, they have a 1K-ohm resistor that isolates them from the switch, so it is possible to define them as outputs. It is also possible to drive them from an external source by connecting that signal to the FPGA probe point header. See Figure 1-7 for details.

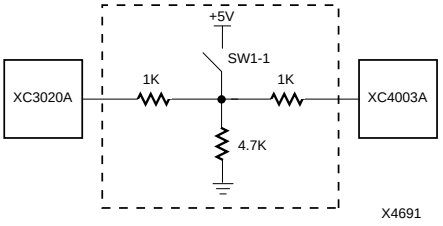


Figure 1-7 Configuration Switch SW1

MPE — Multiple Program Enable (SW1-2)

When MPE is on and SPE is off, the configuration PROM (U1) is reset by the RESET pushbutton (SW4). Configuration must be set to the master serial mode. After a Reset or powerup, the first bitstream stored in the serial PROM is loaded into the XC3020A FPGA. IF you press RESET, the serial PROM address pointer is reset. If you press PROG (SW6), the XC3020A is loaded with the first bitstream again. If you press PROG, and do not press RESET, then the XC3020A is loaded with the next bitstream stored in the serial PROM. The number of bitstreams that can be sequentially loaded is limited by the size of the serial PROM.

SPE — Single Program Enable (SW1-3)

When SPE is on and MPE is off, the configuration PROM (U1) is reset by the XC3020A's INIT output, which is driven Low whenever you press PROG (SW6). The first bitstream stored in the serial PROM is loaded into the XC3020A FPGA.

Note: MPE and SPE must not be on at the same time. MPE and SPE are only used in conjunction with the serial PROMs. The serial PROMs must be configured as OE/$\overline{\text{RESET}}$ to allow MPE and SPE to function properly.

M0, M1, M2 — Mode Pins (SW1-4,5,6)

To configure the XC3020A using the XChecker/Download Cable, these switches must be on, placing the FPGA in slave serial mode. To configure from the onboard serial PROM, these switches must be off to place the FPGA in master serial mode.

MCLK — Master Clock (SW1-7)

When this switch is on, it connects the XC4003A configuration clock (pin 73) to the configuration clock on the XC3020A (pin 60). This connection is used to configure FPGAs in a daisy chain with the XC4003A at the head.

DOUT — Data Out (SW1–8)

When this switch is on, it connects the XC4003A data out line (pin 72) to the data in line of the XC3020A. This connection configures FPGAs in a daisy chain with the XC4003A at the head.

Note: MCLK and DOUT should only be used to configure the FPGAs in a daisy chain.

XChecker/Download Cable Connector (J1)

Table 1-7 describes the pins of the XChecker/Download cable J1 connector.

Table 1-7 XChecker/Download Cable Connector J1

Pin	Name	Function	Pin	Name	Function
J1–1*	VCC	Supplies +5 V to the XChecker cable.	J1–2	RT	Allows XChecker to trigger a readback of the XC3020A. Connects to XC3020A pin 26.
J1–3*	GND	Supplies ground reference to XChecker cable.	J1–4	RD	Used by XChecker for readback data. Connects to XC3020A pin 25.
J1–5*	N.C.		J1–6	TRIG	XChecker input that allows an external event to trigger readback of the XC3020A or outputting a burst of clocks to the XC3020A. Connects to tiepoint J3–1.
J1–7*	CCLK	Provides clock during configuration or readback. Connects to XC3020A input pin 50.	J1–8	N.C.	

J1–9*	D/P	Starts configuration and indicates completion. Connects to XC3020A DONE/PROGRAM pin 45.	J1–10	N.C.	
J1–11*	DIN	Provides configuration data during configuration. Connects to XC3020A DIN input pin 58.	J1–12	N.C.	
J1–13	N.C.		J1–14	N.C.	
J1–15	N.C.		J1–16	CLKI	System clock input to XChecker to be controlled and output on CLKO. Connects to tiepoint J3–2.
J1–17	RST	Connects to jumper J5. If connected, allows XChecker to provide a Reset input (same as pressing Reset button).	J1–18	CLKO	System clock output controlled by XChecker. Used to single-step or burst clocks to the XC3020A. Connects to tiepoint J3–3.

The Download cables supports those pins with an asterisk (*).

Jumper J5 allows the XChecker signal RST on J1-17 to drive the reset line on the demonstration board. Tiepoint pins jumper the following XChecker signals into your circuit. Tiepoint J3-1 connects to TRIG on J1-6; Tiepoint J3-2 connects to CLK1 on J1-16; and, Tiepoint J3-3 connects to CLK0 on J1-18. See Table 1-7 for more information on the XChecker/Download Cable connections.

Serial PROM Socket (U1)

This serial PROM configures the XC3020A. You must use the master serial mode to configure from the serial PROM.

Relaxation Oscillator Components (R1 C5, R2 C6)

R1, C5 and R2, C6 are two RC networks that connect to the XC3020A at pins 12 and 14. These RC networks are for use in a relaxation

oscillator such as the circuit is shown in Figure 1-8.

With the components provided, R1 = R2 = 100k ohms and C5 = C6 = 0.1uF, the oscillator generates an output frequency of approximately 100Hz.

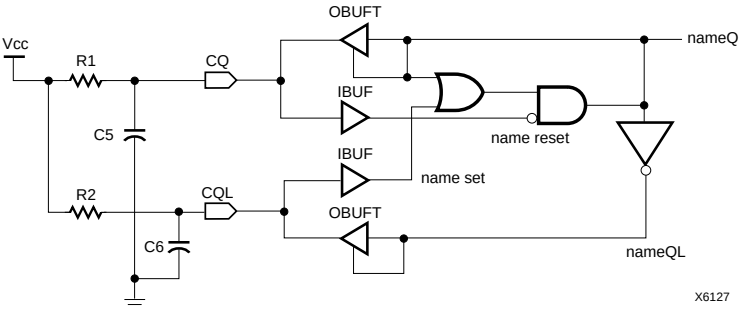


Figure 1-8 Relaxation Oscillator Schematic

Figure 1-9 shows the RC network.

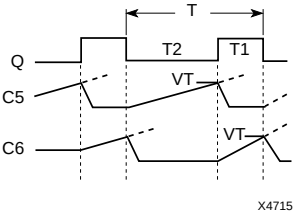


Figure 1-9 Network Calculation Formula

The formula for calculating the RC network is as follows:

T = T1 + T2 = N ((R1C5) + (R2C6))

where:

N = approximately 0.35 for TTl threshold
= approximately 0.75 for CMOS threshold

when the FPGA allows each capacitor to discharge during the opposite timing phase.

Mode Switch Settings

This section describes the SW1 and SW2 switch settings for configuring the XC3020A and XC4003A:

- From the XChecker/Download Cable
- From the serial PROM (single program)
- From the serial PROM (multiple program)
- In a daisy chain

Table 1-8 lists the names and positions of the SW1 and SW2 switches for configuring the XC3202A FPGA from the XChecker/Download cable.

Table 1-8 Configuring the XC3020A from the XChecker/ Download Cable

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	OFF	SW2–2	MPE	X
SW1–3	SPE	OFF	SW2–3	SPE	X
SW1–4	M0	ON	SW2–4	M0	X
SW1–5	M1	ON	SW2–5	M1	X
SW1–6	M2	ON	SW2–6	M2	X
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

Table 1-9 lists the names and positions of the SW1 and SW2 switches for configuring the XC4003A FPGA from the XChecker/Download cable.

Table 1-9 Configuring the XC4003A from the XChecker/ Download Cable

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	X	SW2–2	MPE	OFF
SW1–3	SPE	X	SW2–3	SPE	OFF
SW1–4	M0	X	SW2–4	M0	ON
SW1–5	M1	X	SW2–5	M1	ON
SW1–6	M2	X	SW2–6	M2	ON
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

When you configure both the XC3020A and XC4003A using the XChecker/Download Cable, configure the XC4003A FPGA first. If you configure the XC3020A first, it's configuration is lost when the XC4003A FPGA configures because the $\overline{\text{PROG}}$ signal connects directly to the XC4003A $\overline{\text{PROG}}$ input and through a diode to the XC3020A DONE/$\overline{\text{PROG}}$ input.

Table 1-10 lists the names and positions of the SW1 and SW2 switches for configuring the XC3020A FPGA from the serial PROM.

Table 1-10 Configuring the XC3020A from the Serial PROM (Single Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	OFF	SW2–2	MPE	X
SW1–3	SPE	ON	SW2–3	SPE	X
SW1–4	M0	OFF	SW2–4	M0	X
SW1–5	M1	OFF	SW2–5	M1	X
SW1–6	M2	OFF	SW2–6	M2	X
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

Table 1-11 lists the names and positions of the SW1 and SW2 switches for configuring the XC4003A FGPA from the serial PROM.

Table 1-11 Configuring the XC4003A from the Serial PROM (Single Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	X	SW2–2	MPE	OFF
SW1–3	SPE	X	SW2–3	SPE	ON
SW1–4	M0	X	SW2–4	M0	OFF
SW1–5	M1	X	SW2–5	M1	OFF
SW1–6	M2	X	SW2–6	M2	OFF
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

Table 1-12 lists the names and positions of the SW1 and SW2 switches for configuring the XC3020A FGPA from the serial PROM (multiple program).

Table 1-12 Configuring the XC3020A from the Serial PROM (Multiple Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	ON	SW2–2	MPE	X
SW1–3	SPE	OFF	SW2–3	SPE	X
SW1–4	M0	OFF	SW2–4	M0	X
SW1–5	M1	OFF	SW2–5	M1	X
SW1–6	M2	OFF	SW2–6	M2	X
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

Table 1-13 lists the names and positions of the SW1 and SW2 switches for configuring the XC4003A FGPA from the serial PROM (multiple program).

Table 1-13 Configuring the XC4003A from the Serial PROM (Multiple Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	X	SW2–2	MPE	ON
SW1–3	SPE	X	SW2–3	SPE	OFF
SW1–4	M0	X	SW2–4	M0	OFF
SW1–5	M1	X	SW2–5	M1	OFF
SW1–6	M2	X	SW2–6	M2	OFF
SW1–7	MCLK	OFF	SW2–7	RST	X
SW1–8	DOUT	OFF	SW2–8	INIT	OFF

X indicates don't care.

Table 1-14 lists the names and positions of the SW1 and SW2 switches for configuring the XC3020A and XC4003A FGPAs in a daisy-chain

from the XChecker/Download cable.

Table 1-14 Configuring the XC3020A and XC4003A in a Daisy Chain from the XChecker/Download Cable

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	OFF	SW2–2	MPE	OFF
SW1–3	SPE	OFF	SW2–3	SPE	OFF
SW1–4	M0	ON	SW2–4	M0	ON
SW1–5	M1	ON	SW2–5	M1	ON
SW1–6	M2	ON	SW2–6	M2	ON
SW1–7	MCLK	ON	SW2–7	RST	X
SW1–8	DOUT	ON	SW2–8	INIT	ON

X indicates don't care.

Table 1-15 lists the names and positions of the SW1 and SW2 switches for configuring the XC3020A and XC4003A FGPAs in a daisy-chain from the serial PROM (single program).

Table 1-15 Configuring the XC3020A and XC4003A in a Daisy Chain from the Serial PROM (Single Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	OFF	SW2–2	MPE	OFF
SW1–3	SPE	OFF	SW2–3	SPE	ON
SW1–4	M0	ON	SW2–4	M0	OFF
SW1–5	M1	ON	SW2–5	M1	OFF
SW1–6	M2	ON	SW2–6	M2	OFF
SW1–7	MCLK	ON	SW2–7	RST	X
SW1–8	DOUT	ON	SW2–8	INIT	ON

X indicates don't care.

Table 1-16 lists the names and positions of the SW1 and SW2 switches for configuring the XC3020A and XC4003A FPGAs in a daisy-chain from the serial PROM (multiple program).

Table 1-16 Configuring the XC3020A and XC4003A in a Daisy Chain from the Serial PROM (Multiple Program)

Switch	Name	Position	Switch	Name	Position
SW1–1	INP	X	SW2–1	PWR	X
SW1–2	MPE	OFF	SW2–2	MPE	ON
SW1–3	SPE	OFF	SW2–3	SPE	OFF
SW1–4	M0	ON	SW2–4	M0	OFF
SW1–5	M1	ON	SW2–5	M1	OFF
SW1–6	M2	ON	SW2–6	M2	OFF
SW1–7	MCLK	ON	SW2–7	RST	X
SW1–8	DOUT	ON	SW2–8	INIT	ON

X indicates don't care.

FPGA Demonstration Board Operation

Note: The information in this section applies to both the XC3020A and the XC4003A FPGAs. However, for clarity it only references the XC4003A FPGA.

Downloading with XChecker

You must follow the recommended design flow to assure proper operation. A demonstration design is supplied with the Xilinx FPGA Demonstration Board in the XACT\examples\core\litefpga directory for PCs and the $XACT/examples/core/litefpga for workstations.

Please read the text files that accompany these designs to acquaint yourself with the information.

If you just want to download a demonstration design, change to the XACT\examples\core\litefpga directory and refer to the "XChecker Cable and Logic Probe" chapter in this manual for more information.

You can also view or edit the demonstration designs supplied with the FPGA Demonstration Board.

Note: Make backups before making changes to any demonstration design files.

1. Produce a routed design, *design*.lca using a design entry tool and the appropriate place and route tool or XDE for manual implementation.

   If you want a global Reset signal in your XC4000 designs, you must include the Startup symbol in your design and select the location of the RESET pin. Attach pin 56 to an inverter and the GSR pin on the Startup symbol. GSR is active-High so you must include an inverter between the pad and the Startup symbol.

2. Generate a bitstream for the design, *design*.bit with the appropriate configuration options using the MakeBits program.

3. Optionally, create a PROM File.

4. Generate a PROM file (*design*.mcs, *design*.tek, or *design*.exo) using the MakePROM program. This step is optional since XChecker can use the *design*.bit file as input.

5. Connect XChecker to the target system.

   The XChecker cable draws its power from the target system through the V_{CC} and GND wires. Therefore, power to XChecker, as well as to the target FPGA, must be stable. You must not connect the XChecker pins to any signals before connecting V_{CC} and ground to the FPGA Demonstration Board.

   When you use XChecker to download, only one of the two-keyed connectors are needed.

6. Connect XChecker to J1 (for the XC3020A) and J2 (for the XC4003A) on the FPGA Demonstration Board.

7. Set the mode switches.

   When you use the XChecker cable, the M0, M1, and M2 switches must be on. This setting causes the device to be in the serial slave mode. Refer to Table 1-14 for the switch settings necessary to configure a daisy chain.

Starting XChecker

From within XDM (version 2.3 or later), select XChecker from the Verify menu. You can edit the xchecker.pro file if you desire. You can also start XChecker from the operating system prompt.

xchecker *design_name*

When you start XChecker with no options, XChecker selects the port where the cable is found and sets the baud rate to the maximum allowed by the platform.

XChecker indicates that the FPGA design is loading. When loading is complete, XChecker indicates that the Done pin went High. At this point, the loaded bit file functions as designed.

Loading with a Configuration PROM

If you already have a design burned in a PROM, skip to step 5. You can also view or edit the demonstration designs supplied with the FPGA Demonstration Board.

Note: Make backups before making changes to any demonstration design files.

1. Place and route the design.

   Produce a routed design, *design*.lca using a design entry tool and the appropriate place and route tool or XDE for manual implementation.

2. Generate a configuration bitstream for the design, *design*.bit with the appropriate configuration options using the MakeBits program.

3. Create a PROM file.

   Generate a PROM file (*design*.mcs, *design*.tek or *design*.exo) using the MakePROM program. See the MakePROM documentation in the *Development System Reference Guide* to create a PROM file. Then follow the instructions for burning a PROM in the "XPP Serial Configuration PROM Programmer" chapter in this manual.

Note: The XC17XXX PROMs must be programmed with the reset polarity set for active-Low.

4. Place the PROM on the FPGA Demonstration Board.

After you have a PROM that has a configuration bitstream burned into it, place it into the FPGA Demonstration Board with power off. Use socket U2 for XC4003A devices and for XC4003A and XC3020A devices in a daisy chain with the XC4003A at the head of the chain. Use socket U1 for XC3020A devices.

5. Set the mode switches.

   When you use the serial PROMs, the M0, M1, and M2 switches must be off. This setting causes the device to be in the active master serial mode. Set the MPE, SPE, and RST switches to the desired positions. Refer to Table 1-15 and Table 1-16 for switch settings required to configure a daisy chain.

6. Load the FPGA.

7. After you insert the PROM into the socket and set the configuration switches, apply power to the FPGA Demonstration Board.

   This step configures the FPGA; When the Done pin goes High, it indicates that the design logic has become active.

Demonstration Designs

The example design in $XACT/examples/core/litefpga for workstations and \XACT\examples\core\litefpga for PCs incorporates the ability of the XC4003A to build ROM out of function generators. The ROM macros store a sequence of patterns that are displayed on the 7-segment displays and the LED bar graphs of the FPGA Demonstration board.

The litefpga.mcs design file is a daisy chain of an XC4003A design and an XC3020A design. When you are ready to download the litefpga.mcs design, set up the FPGA Demonstration board as shown in Table 1-14.

The design schematics are available by calling the Xilinx Technical Support Hotline.

XPP/Serial Configuration PROM Programmer

The Xilinx PROM Programmer (XPP) supports the following Xilinx FPGAs:

- XC2000, XC2000L
- XC3000, XC3100, XC3000A, XC3000L
- XC3100A
- XC4000, XC4000A, XC4000H
- XC5200

When using Xilinx Serial Configuration PROMs to configure XC2000, XC3000, XC4000, and XC5200 devices, you can program them with the HW112 Serial Configuration PROM programmer, or one of the third-party PROM programmers listed in the *The Programmable Logic Data Book*. The HW112 connects to any serial port on a PC, or Sun, DEC, or Apollo workstation.

With XPP on the PC, you use the keyboard to control the HW112 programmer. On workstations you can use either the mouse or keyboard to control the PROM programmer. Since the XPP software is not licensed, you can install the software and programming unit on multiple PCs or workstations.

The HW112 supports the following PROMs in 8-pin DIP packages:

- XC1718D, XC1718L
- XC1736A, XC1736D, AM1736
- XC1765, XC1765D, XC1765L, AM1765
- XC17128, XC17128D
- XC17256D

To configure these PROMs in 20-pin PLCC packages, you need to use the HW112-PC20 adapter, which fits on top of the HW112. You can use the HW112-O8 adapter with small outline 8-pin (SO8) packages.

XPP supports the programming of multiple device types in a single session on PC platforms. You can program large configuration patterns of daisy-chained devices over different device types.

Programming Flow

To configure a PROM, you must first compile the LCA file into a BIT file using the MakeBits program in the XACT*step* Development System. When configuring a single FPGA, you can use the BIT file as input to the PROM programmer. For a daisy-chain of FGPAs, you must create a combined hexadecimal file using the MakePROM program. Then you would use XPP to download the data to the programming unit and program the device. Figure 2-1 shows the XPP programming design flow.

The MakePROM utility supports three hexadecimal file formats listed below; you can use any of these formats with XPP to program serial configuration PROMs.

- Intel MCS–86 Hexadecimal Object (MCS extension)
- Motorola EXORMACS (EXO extension)
- Tektronix Hexadecimal (TEK extension)

Additional information about the MakeBits and MakePROM programs is available in this manual.

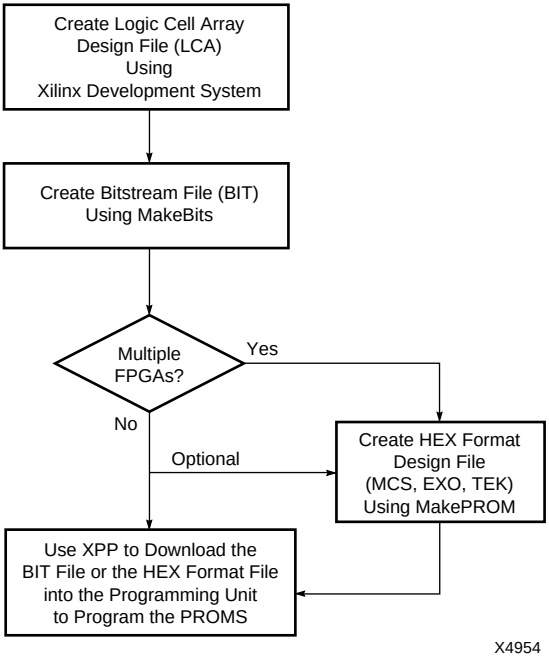


Figure 2-1 XPP Programming Process Overview

Programmer Setup

Note: Before you set up the programmer, make sure that you have the XPP software on your system. If it is not already installed, follow the installation instructions included with the XPP program.

Complete the following steps to set up the HW112:

1. Turn off the rocker switch on the rear panel of the programmer.
2. Connect an RS-232 cable (not supplied) between the PC/workstation serial port and the programmer serial port.
3. Connect the a.c. adapter to the power connector input and a.c. line source.
4. Alternatively, you can connect the HW112 power connector to a +9-V, 1-amp regulated supply with a 5mm O.D. x 2.1mm I.D. female power plug (not supplied).

Note: Be sure to check the power supply. The HW112 requires a +9-volt d.c., 1-amp power supply. You could damage the PROM programmer if you connect it to a different power source.

Figure 2-2 and Figure 2-3 depict the top and rear panels of the HW112 programmer.

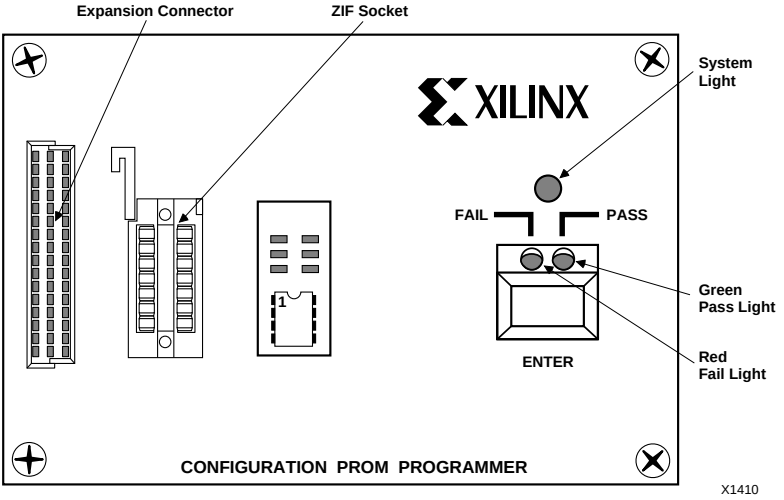


Figure 2-2 HW112 Programmer Top Panel

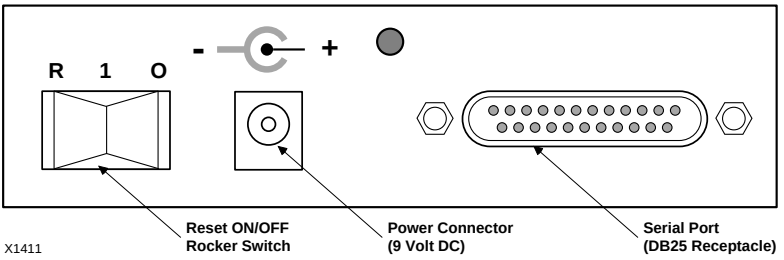


Figure 2-3 HW112 Programmer Rear Panel

Figure 2-4 illustrates the proper cable connections.

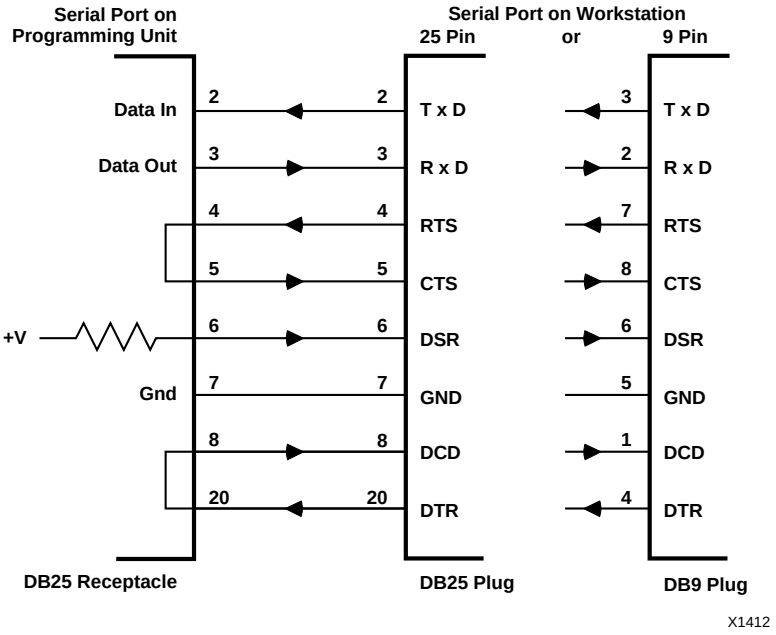


Figure 2-4 Serial Cable Connection

5. Turn on the programmer power switch before running the XPP software.

   After you power on the programmer, the self-test firmware takes a few seconds to test the hardware.

The red system light flashes during the power-on self-test, and then remains lit. A flashing red Fail light on the Enter button indicates a hardware problem. Refer to Table 2-1 to determine the nature of the problem.

Table 2-1 Serial PROM Programmer Troubleshooting

Fail Indicator Flashing	Problem
1 time	Microcontroller RAM Failure
2 times	Bad PROM Checksum
3 times	SRAM
4 times	D/A Converter

XPP Setup

This section describes environment variables and hardware configuration that you must define to use the XPP software and HW112 PROM Programmer.

Environment Variables

Before you run XPP, you must set three environment variables. For PCs, you must set three variables: MACHINE, PATH, and XACT. For workstations you only need to set the PATH and XACT environment variables.

MACHINE (PCs)

The MACHINE environment variable indicates the type of machine you are using. Specify NEC for NEC 9801 computers, IBMPC for PCs and other compatibles. The default is IBMPC.

For example, to set the MACHINE environment variable for an NEC 9801, enter the following string in the autoexec.bat file:

```
Set MACHINE=IBMPC
```

PATH

The operating system uses the PATH environment variable as a search path for the XPP program. Make sure that the XACT directory in which XPP resides is in the search path.

XACT

The XACT environment variable points to the directory where the program-dependent files are found.

For example, on the PC, enter the following string in the autoexec.bat file:

```
Set XACT=drive:\XACT
```

All XPP files, except log and text files, are searched for in a prescribed path, starting with the current directory, and then through the XACT environment and its subdirectories: files, data, and designs.

XPP Configuration

When you use XPP for the first time, you must configure the software to specify the serial port, baud rate, sound setting, device count, and device name. Follow the instructions on the screen to configure the software.

There are two methods for starting XPP. You can select XPP from the Verify menu in the Xilinx Design Manager (XDM), or type the following command at the prompt:

```
xpp
```

Port Name

For workstations there are three options for the serial port:

- /dev/ttyd0,
- /dev/sio2
- /dev/ttya

Consult your system administrator for the serial device name assigned to your system. Table 2-2 lists the default port settings for supported platforms.

For PCs, the two options for the serial port are COM1 and COM2. The default setting is COM1.

Table 2-2 Valid System Port Names

Machine	Serial Port
DEC 3100	/dev/ttyd0
Sun	/dev/ttya
Apollo	/dev/sio1
IBM PC	com1
DEC Alpha	/dev/tty00
RS6000	/dev/tty1
HP700	/dev/tty01

Baud Rate

XPP supports these baud rates; 1200, 2400, 9600, and 19200. The default baud rate is 9600.

Sound

Errors during programming cause a beep or other sound alert from the host when the sound setting is on (default). To turn the sound from off to on, or on to off, enter **y** at the prompt when you change the settings interactively.

Device Repetition Count

The device repetition count setting determines the default number of devices to be programmed or compared. You can override this setting during programming or comparing. The default device repetition count setting is one.

Device Name

XPP requires that the exact device name be specified in order to program a PROM or perform other XPP operations. You can specify one or more device names, using the full name of the PROM. For example, if you are only programming an XC1736D device in a session, you would enter **XC1736D** when prompted for a new device list.

If you were to program more than one device type (for example, XC1765, XC17128, and XC1736D), you would enter the following string when prompted for a new device list.

```
XC1765 XC17128 XC1736D
```

The default setting is one device type, XC1736D.

Note: Error messages that appear during configuration are explained in the XPP Error Messages section at the end of this chapter.

XPP saves the configuration information in a profile file called xpp.pro that is located in the current directory. These configuration settings are used each time you run the XPP program. After the first run of XPP, the initialization process is bypassed as long as the xpp.pro file is present in your search path.

If you want to change the configuration information, run XPP with the –s option. You can choose to save the saving the new settings in the xpp.pro file or, temporarily use the new settings.

Using XPP (PC Users)

You can use the PC version of XPP in two ways: interactive mode and batch mode. You use the interactive mode for engineering development. You use the batch mode to support manufacturing needs.

When you execute XPP, it reads the xpp.pro file and tries to establish communication with the HW112 programmer. If XPP cannot establish communication with the HW112 programmer, it aborts with an error message.

Once communication is established, XPP displays the configured device type, the bootstrap firmware version number, and then waits for you to press the ↵ key before displaying the Main Programming Menu.

Command Syntax

The command syntax for XPP is as follows:

`xpp [-ahs] [-b` *filename]* **`[-d`** *device] data_file*

Note: If you are using the interactive mode, *data_file* is also an optional parameter.

Descriptions of the XPP options and parameters follow.

–a Use ANSI Video Interface

This option signals XPP to use the ANSI video interface, which is primarily used on NEC 9801 computers because they depend on an ansi.sys driver to handle all screen I/O. Most MS–DOS computers do not require this option.

–h Display XPP Help Information

This option displays information about XPP, including program execution, options, files that are used and created by XPP, and environment variables.

–s Enter Setup Mode

This option allows you to change the configuration information in the xpp.pro file, which XPP reads upon execution.

–b Specify Data_File Name

This option specifies which data file to input to the PROM programmer.

–d Set the Device Type

The –d option allows you to specify the supported devices when invoking XPP. The device type specified by this option overrides the device type stored in the xpp.pro.

–polarity low/ high

This option sets the RESET Line Polarity of the following PROMs (XC1718D, XC1718L, XC1736D, XC1765, XC1765D, XC1765L, XC17128, XC17128D, and XC17256D)

The –polarity option applies only to those serial configuration PROMs with programmable RESET polarity. It specifies the PROM's RESET/$\overline{OE}$ line as an active Low or active High reset. RESET/$\overline{OE}$ is a dual-purpose signal with two functions, resetting the PROM (RESET) and enabling the output of the PROM (OUTPUT ENABLE). Changing the RESET signal polarity also changes the $\overline{OE}$ function's active state to the opposite polarity. The default is active High reset.

data_file LCA Design Data File

This option allows you to specify the design data file containing the LCA design that you want to process. The design data file can use formats from Intel (MCS), Tektronix (TEK), Motorola (EXO), or Xilinx (BIT or RBT). If you do not specify the data file, XPP searches for the design data file in this order: BIT, MCS, TEK, EXO.

Function Keys

There are three function keys (F1, F9, and F10)that the PC version of XPP supports only from the Main Programming Menu.

F1

F1 displays help for any highlighted menu selection, or any submenu appearing on the screen. Press the Escape (Esc) key to exit the help function and return to your previous location.

F9

F9 displays the SCREEN COLOR CONFIGURATION menu, where you can change the screen color.

```
SCREEN COLOR CONFIGURATION:

1) Monochrome
2) Color Palette 1
3) Color Palette 2
4) Color Palette 3

Select palette code (1-4) :
```

Type a number from 1 to 4 to select the desired palette. then press the ↵ key.

F10

F10 opens a DOS shell where you can enter or execute DOS commands without leaving XPP. In the DOS shell, enter **`exit`** at the prompt to return to XPP. You can only use the F10 key while in the Main Programming Menu.

Interactive Mode

Using the interactive mode, you can operate XPP from the Main Programming Menu, shown in Figure 2-5. Use the Up and Down arrow keys on the keyboard to scroll through the menu items. The current menu item is highlighted.

Press the ↵ key to select the highlighted menu item. Press the Escape (Esc) key to cancel any operations.

To exit the program from the main menu, enter an **x**. You can type an upper case or lower case letter

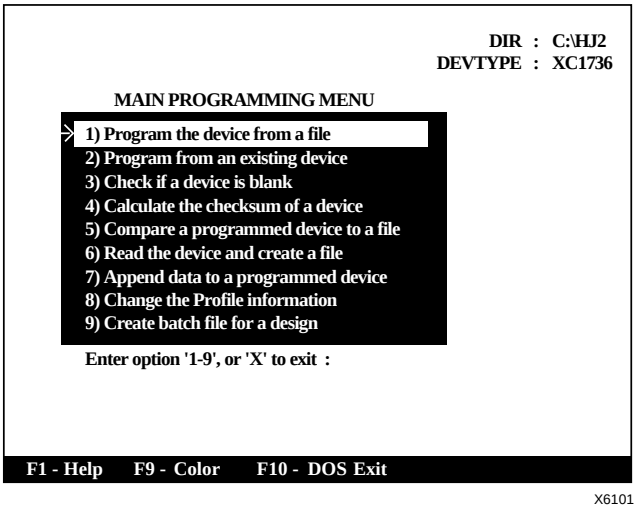


Figure 2-5 Main Programming Menu

Options

There are nine options that are available from the Main Programming Menu, as described in the following section.

Program the Device from a File

Use this option to program serial PROMs with the contents of an existing BIT file or hexadecimal file. Perform the following steps after selecting this option from the Main Programming menu:

1. Enter the full name of the BIT file or the hexadecimal file that you want to use. Include the file name extension (BIT, MCS, and so forth).

Note: You can type**?** and press the ↵ key to see the list of data files in the current directory.

Note: Make sure the device type you are using matches the device type list specified in the xpp.pro file. Use option 8 in the Main Programming menu to change the device type and remember that the XC17XX devices are one-time programmable. Multiple device type programming is only supported by option 1.

2. Press the ↵ key to use the default device repetition count, or enter the desired repetition number.
3. Insert a serial PROM into the Zero Insertion Force (ZIF) socket on the programmer.

   XPP permits you to ignore the condition of the device and continue programming. XPP checks if the device is blank before actually programming it; if it is not, a prompt appears. You can either continue with the same PROM, or try a different one.
4. Press either the ↵ key on the keyboard or the Enter button on the programmer to start the programming process.

   XPP prints the configuration data checksum onscreen immediately after reading the source file.

   If programming is successful, XPP issues a message onscreen and illuminates the programmer's green Pass light. If you are programming more than one PROM, XPP prompts for the next device, which must be of the same type.

   If programming is unsuccessful, XPP displays an error message onscreen and the HW112 Programmer Fail light flashes red. To correct this problem, repeat steps 3 and 4 and program a different PROM.

The following submenu, which allows you to set up Low or High enable on the RESET line, appears whenever you try to program a device with programmable RESET polarity.

```
RESET POLARITY MODE:

1) high enable <- default
2) low enable

Select polarity mode (1-2):
Press “Enter” to use default mode
```

Although you can program a large LCA configuration file into multiple device types, devices that have the reset polarity mode changed to an active Low logic level cannot be cascaded with XC1718, XC1736, or XC1736A devices. If you are mixing XC1718, XC1736, or XC1736A PROMs with other Xilinx serial PROMs, you cannot change the reset polarity mode of the larger devices.

If you are making several copies of a single PROM design, XPP prompts you for the next device, which must be of the same type.

If you are making several copies of a multiple PROM design, XPP programs all PROMs of the same type at one time. For example, if you are programming an LCA configuration pattern into an XC1765 and XC17128 PROM and you want ten copies, XPP would program ten XC1765 devices before prompting for the XC17128.

If a data file is large, such as an XC4013 configuration bitstream, it can require several PROMs to hold the complete design. These PROMs may or may not be of the same type. When one device fills, XPP prompts for the next device. XPP refers to the content of each PROM as a pattern, such as “pattern 1” and “pattern 2” for data that requires two PROM devices. The programming unit can only program one pattern at a time; therefore, all devices used to store the first pattern must be programmed before programming devices that will store the next pattern.

Program from an Existing Device

This option reads the data from a programmed serial PROM and programs the same data into a blank serial PROM, making a duplicate. Perform these steps after selecting this option:

1. Insert the programmed serial PROM into the ZIF socket and press the ↵ key.
2. Press the ↵ key again to accept the default device repetition count, or enter a different repetition number.

3. Remove the original serial PROM from the ZIF socket.
4. Insert the a blank serial PROM into the ZIF socket. Make sure the device type is defined in the xpp.pro file. Use option 8 from the Main Programming Menu if you need to change the device type.

   XPP verifies that the device is blank before it begins programming, so you can ignore the condition of the device and continue programming.
5. Press the ↵ key on the keyboard or the Enter button on the programmer when you are ready to program the device.

   If programming is successful, XPP displays a message and illuminates the green Pass light on the programmer. If you are programming more than one PROM, XPP prompts for the next device. At that time, remove the PROM from the socket and repeat steps 4 and 5 above.

   If programming is unsuccessful because the PROM has already been programmed or because it is faulty, XPP displays an error message and the HW112 Programmer Fail light flashes red. To correct this problem, repeat steps 4 and 5 with a different PROM.

   XPP displays the configuration data checksum immediately after reading the source file.

Check if a Device Is Blank

Use this option to determine if the PROM device is blank. Perform the following steps after selecting this option from the main menu:

1. Press the ↵ key to accept the default device repetition count, or enter a different repetition number.
2. Insert the serial PROM to be tested into the ZIF socket.
3. Press either the ↵ key on the keyboard or the Enter button on the programmer to start the programming process.
4. XPP displays the result of the check onscreen.
5. If the repetition count is set to more than 1, remove the tested device and repeat steps 1 through 4 for each additional device that you want to test.

Calculate the Checksum of a Device

Use this option to calculate the checksum of a device, not the checksum of a file. Perform the following steps after selecting this option from the main menu:

1. Press the ↵ key to use the default device repetition count, or enter a different repetition number.
2. Insert the serial PROM to be checked into the ZIF socket.
3. Press the ↵ key to start calculating the checksum.
4. XPP displays the checksum result onscreen.
5. If the default device repetition count is set to more than 1, remove the tested device and repeat steps 1 through 4 for each additional device that you want to test.

The checksum calculation technique is consistent throughout XPP, but might not be compatible with the checksum technique that the XPROM program uses (an earlier PROM programmer supplied by Xilinx) or software from other vendors.

Compare a Programmed Device to a File

Use this option to determine the integrity of the data within a device. This option compares the data from a BIT file or a hexadecimal file to the corresponding programmed data from a PROM. If the data file requires more than one PROM, XPP compares one device at a time to the source contents. XPP refers to the contents as a pattern. Perform the following steps after selecting this option from the main menu:

1. Enter the name of the BIT file or the hexadecimal file that you want to use. Include the file name extension (.bit, .mcs, .tek, .exo). If this data file is large, XPP displays the number of patterns that it produced.

Note: You can type**?** and press the ↵ key to see the list of data files in the current directory.

2. Press the ↵ key to accept the default device repetition count or enter a different repetition number.
3. XPP displays the pattern number to be compared. Select the appropriate device then insert it into the ZIF socket.

4. At this time, XPP prompts you to press the Enter key before it compares the device to the data file. Press either the ↵ key on the keyboard or the Enter button on the programmer to start the comparison.

   XPP indicates whether the data from the source file matches the data from the PROM. If the data in both sources do not match, XPP displays the number of bits that are different.

5. If applicable, XPP also prompts for the next device. Remove the present device and repeat steps 3 and 4.

   The programmer compares the file data to the PROM data until the data is exhausted or the number of mismatches reaches 255, at which time the process halts and you are returned to the Main Programming Menu.

6. To save the entire data file, including the different bits, use Option 6, "Read the device and create a file."

Read the Device and Create a File

This option reads a serial PROM and saves its contents as a text file. Perform the following steps after selecting this option from the main menu:

1. Insert the programmed serial PROM into the ZIF socket on the programmer.

2. Select one of the following output formats from the submenu that appears.

```
1   ASCII one's and zero's.
2   ASCII HEX characters.
3   ASCII .rbt format
4   Comparison against a file.
```

   If you select the fourth output format, XPP prompts you for the name of the BIT or hexadecimal file that is to be compared to the device. Enter the comparison file name, including the extension (for example, prom1.mcs, lca2.bit) and press the ↵ key.

If you selected the fourth output format in step 2, XPP writes the differences between the two files to the output file. A slash (/) indicates that the device bit is a zero, but the source file was a one. A period (.) indicates that the device bit is a one, but the source file was a zero.

3. At the next prompt, enter the output file name and press the ↵ key. You do not need to enter the extension because it defaults to TXT.
4. Insert the serial configuration PROM into the ZIF socket.
5. Press the ↵ key on the keyboard or press the Enter button on the programmer to start the read or output process.

Append Data to a Programmed Device

This option allows you to concatenate additional configuration data to a previously programmed PROM. This adds a second design file to a serial PROM that already contains a design. This step is usually done with a PROM that programs a daisy–chain of FPGAs. Perform the following steps after selecting this option from the main menu:

1. Enter the name of the design file that already exists in the PROM.
2. Enter the name of the design file including the filename extension (BIT, MCS, and so forth) that you want added to this PROM.

   If the add-on design file is too large for the existing space, an additional device is required to accommodate the rest of the design file. XPP indicates the number of patterns generated before programming starts.
3. Press the ↵ key on the keyboard to accept the default device repetition count or enter a different repetition number.

Note: Make sure that the device type specified in the xpp.pro file matches the inserted device. If the device types do not match, press the Escape (Esc) key to abort and use option 8 from the menu to change the profile.

4. Insert the previously programmed PROM into the ZIF socket. Either press the ↵ key on the keyboard or the Enter button on the programmer to start the concatenating process.

If programming is successful, XPP displays the status and illuminates the green Pass light on the programmer lights.

If programming is unsuccessful either because the PROM has already been programmed and you did not enter the name of the file it already contains, or because the PROM is faulty, XPP displays an error message and flashes the red Fail light on the programmer. To correct this problem, make sure that you correctly specified the name of the file already in the PROM, or repeat steps 3 and 4 and reprogram a different PROM.

When appropriate, XPP prompts for the next device. Remove the PROM from the socket and repeat step 4.

5. To concatenate the additional data at the right location, you must specify the existing data file in the programmed PROMs.

   With this information, XPP calculates the correct starting point for additional data. XPP prompts you to insert the last partially programmed device only if the originally programmed files used multiple PROMs.

Change the Profile Information

Use this option to change the following settings in the present profile: serial port, baud rate, sound on or off, device repetition count, and device list. After making changes, you can update the profile by entering **y** at the prompt; enter **n** or press the Escape (Esc) key to cancel.

XPP uses the most recent data even if you do not update the profile. If you change the baud rate, XPP automatically resets the programming unit and then sets the new baud rate. Press the ↵ key again to return to the main menu.

Creating a Batch File for a Design

Use this option to create a batch file, which simplifies the PROM programming process for manufacturing. After creating this batch file, you can execute it from the DOS prompt.

The settings in the current directory xpp.pro file determine the device repetition count, baud rate, communication port, and device list. Make sure the xpp.pro file contains the correct settings before running XPP in batch mode.

If you are programming more than one device type during the batch mode session, ensure that the device list contains the proper device names before starting.

Perform the following steps after selecting this option from the main menu:

1. Enter the name of the design file to program into the PROM (for example, design1.mcs).
2. Enter the name you want to give the batch file (for example, design1.bat).

   Be sure to specify the BAT extension for the name of the batch file when you are creating a batch file for a PC platform.

XPP creates the batch file with the necessary execution instructions.

Batch files that XPP creates are meant to be executed as created. Xilinx does not recommend that you edit these files.

Batch File Mode

Using the batch mode, you can configure XPP to bypass the Main Programming menu, and program PROMs with a preset design file. You must have the following files in the current directory to use XPP in the batch mode.

- The BAT file that was created using option 9
- The xpp.pro file with the desired settings
- The design file (DSF) specified in the BAT file

  At the beginning of execution, XPP loads the input file into the programmer and then prompts you for a device to program. Perform the following steps to continue this process.

1. Press the ↵ key to use the default device repetition count, or enter the desired repetition count.
2. Insert a blank serial PROM into the ZIF socket. Make sure the device is the same type as the type defined in the xpp.pro file. Use option 8 if you need to change the device type.

   XPP permits you to ignore the condition of the device and continue programming.

3. Press either the ↵ key on the keyboard or the Enter button on the programmer to start programming the device.

   If programming is successful, XPP displays the status and illuminates the green Pass light on the programmer. If programming is unsuccessful because the PROM is faulty or has already been programmed, XPP displays an error message and the HW112 Programmer Fail light flashes red.

When the device repetition count is reached, XPP terminates and automatically exits to DOS.

Pressing the Escape (Esc) key when there is a pause during execution, such as when XPP issues a prompt, also terminates the program and exits to DOS.

Using XPP (Workstation Users)

The workstation version of XPP provides two user interfaces: command-line entries and interactive mode. Typically, you use the command-line mode for production, as only a limited number of functions are available from the command line. You use the interactive mode for prototyping.

Configuration information for XPP is stored in the xpp.pro file. This file is created the first time XPP is executed and contains the port name, baud rate, device repetition count, and so forth that was used during this first session.

XPP reads the previously created configuration profile (xpp.pro) and tries to establish communication with the programmer. If XPP cannot establish communication with the programmer, it aborts with an error message. You must resolve the error before you can continue.

If there are no hardware or configuration problems with XPP, once you establish communication, the PROM Programmer displays the configured device type and the bootstrap firmware version number.

Command-Line Mode

The command-line mode is non-interactive. By specifying commands on the XPP command line, you can direct XPP to execute commands and exit without entering the interactive mode. The command-line mode is useful in a programming lab setting.

You can pre-set the port and device setting at the command line. You can also direct XPP to start with a short, interactive session for the setup command before executing the specified commands.

Command-Line Syntax

You invoke the command-line mode by typing in the following command string at the system prompt:

xpp *[command–line parameters] command [commands]*

Command-Line Parameters

There are three parameters that you can specify on the command line:

–help

This parameter displays information about XPP, including program execution, options, environment variables; and the files that XPP uses and creates.

–baud rate

This parameter sets the communication baud rate.

–batch name

This parameter specifies the name of the batch file.

–port name

This parameter specifies the communication port of your system that is connected to the PROM programmer.

–dev name

This parameter specifies the PROM device to use. Valid PROMs are XC1718D, XC1718L, XC1736A, XC1736D, 1736AMD, XC1765, XC1765D, XC1765L, 1765AMD, XC17128 XC17128D, and XC17256D.

–setup

This parameter puts you in the interactive mode for the Setup command.

Commands

The following commands are available from the command line.

program

This command allows you to program the device from a file. The syntax for this command is as follows:

```
program [–high|–low] design [numdev]
```

where:

–high programs RESET polarity of the device to High. This is the default.

–low programs RESET polarity of the device to Low.

design is the name of the design file to program into the device. The design file can be one of the following formats:

- .bit Xilinx binary bit file format generated by MakeBits
- .mcs Intel MCS-86 PROM file format
- .tek Tektronix hexadecimal PROM file format
- .exo Motorola EXORMACS PROM file format
- .rbt Xilinx ACSII bit file format

copy

This command allows you to program the device from an existing programmed device. The RESET polarity is the same as the existing device that XPP reads. The syntax for this command is as follows:

```
copy [numdev]
```

where:

numdev is a number, from 1 to 1000, of devices to be programmed with the design file. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

check

This command allows you to verify the status of a device. If the device is not blank, the command displays the checksum of the device. The syntax for this command is as follows:

```
check [numdev]
```

where:

numdev is a number, from 1 to 1000, of devices to be programmed with the design file. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

checksum

This command reads a programmed device, calculates and displays its checksum. The syntax for this command is as follows:

```
checksum [numdev]
```

where:

numdev is a number, from 1 to 1000, of devices to be programmed with the design file. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

compare

This command reads the specified design file and compares the data to a programmed device. You can direct the compared data to an output file or display the number of different bytes on the screen. The syntax for the compare command is as follows:

```
compare [–out name] design [numdev]
```

where:

–out *name* specifies the output file that has the recorded differences between the design file and the programmed device. The output file contains the ASCII bits that were read back from the device. The difference between the device and the design file are marked by two special characters:

- "/" device data is 1 and design data is 0

- "." device data is 0 and design data is 1

If you do not specify the –out parameter, XPP displays the number of bytes that are different between the design file and the device.

design is the name of the design file to program into the device. The design file can be one of the following formats:

- .bit Xilinx binary bit file format generated by MakeBits
- .mcs Intel MCS-86 PROM file format
- .tek Tektronix Hexadecimal PROM file format
- .exo Motorola XORMACS PROM file format
- .rbt Xilinx ASCII bit file format

numdev is a number, from 1 to 1000, of devices to be programmed with the design file. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

read

This command reads the device data into a specified file. The data can be formatted in different number bases in the file. The read command has the following syntax:

`read [–bin|–hex|–dec|–rbt] [–out name] [numdev]`

where:

- –Bin formats the data in binary characters. This is the default.
- –Hex formats the data in hexadecimal characters (base 16).
- –Dec formats the data in decimal characters (base 10).
- –Rbt formats the data in raw bits (ASCII) format.

–out *name* specifies that the data read from the device is to be written to an output file called *name*. The default is to write the data in binary format. If you do not use the –out option, XPP displays the data on the screen.

numdev is a number, from 1 to 1000, of devices to be programmed with the design file. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

append

This command appends new data to the specified device. Append adds the data from another design file to the end of the current programming in a device.

The append command has the following syntax:

append *design1 design2* [numdev]

where:

design1 is the name of the design file already programmed into the device. XPP reads the original design file to calculate where the current programming ends in the device. The design file must be in the current directory, or the file name must contain the full path to the file.

design2 is the name of the design file to be appended to the device. The design file must be in the current directory, or the file name must contain the full path to the file.

numdev specifies the number of devices to be used as the default number *numdev* for other commands. Enter a positive integer value from 1 to 1000. If you do not specify a value, XPP reads the value from the xpp.pro file. If there is no value in the xpp.pro file, XPP sets the value to 1.

setup

This command allows you to set and modify the programmer options. If you use this command with no parameters, XPP prompts for all required parameters.

Note: You can only specify one parameter when using the setup command on the XPP command line:

The syntax for this command is as follows:

setup [–dev *devices* –port *name* –count *numdev*
–baud *rate* –sound *mode*]

where:

–dev *devices* specifies the devices that are used. Enter a list of device names with at least one space separating the names.

–port *name* specifies the name of the serial port that connects to the HW112 PROM Programmer. Use one of the following port names:

- Apollo /dev/sio (Default) /dev/sio1 /dev/sio2
- RS6000 /dev/tty0 (Default) /dev/tty1
- DEC Alpha /dev/tty00 (Default) /dev/tty01
- Sun /dev/ttya (Default) /dev/ttyb
- HP700 /dev/ttya

–count *numdev* specifies the number of devices to be used as the default number *numdev* for other commands. Enter a positive integer value from 1 to 1000. If you do not specify a value, XPP reads the value from the xpp.pro file; and if there is no value in the xpp.pro file, the value is set to 1.

–baud *rate* specifies the serial communication baud rate. The HW112 PROM Programmer supports two baud rates, 9600 and 19200. If you do not specify a baud rate, XPP reads the value from the xpp.pro file. If no such value exists in the xpp.pro file, the default is 9600 baud.

–sound *mode* specifies whether a beep sounds when XPP encounters an error during programming. Enter one of two modes:

- on enables beep sound
- off disables beep sound

Examples

This section contains three examples that use the command-line syntax.

Example 1

To program five XC1736D devices with low reset polarity, using MCS file *xxx*.mcs, enter the following command string:

```
xpp –dev xc1736d program –low xxx 5
```

Example 2

To copy 15 XC1736D devices, enter the following command string:

```
xpp –dev xc1736d copy 15
```

Example 3

To specify a new port (on a Sun workstation), a baud rate, and change to the Interactive mode, enter the following command string:

```
xpp –port /dev/ttya –baud 19200
```

Note: Omitting the device type or input programming file invokes the interactive mode.

Interactive Mode

The interactive mode prompts you for commands and programmer settings. The interactive mode allows you to view help information for the available commands and settings. The commands available in this mode are the same as for the command line. Refer to the previous command descriptions for details.

Using the interactive mode, you use XPP from the Main Programming Menu, shown in Figure 2-6.

In the interactive mode, you can pre-set the port and device settings at the command line. You can also direct XPP to start immediately with the Setup command at the beginning of the interactive session.

Main Programming Menu

The Main Programming Menu remains on the screen throughout the session so you can select the appropriate command.

At the XPP? prompt enter the number of the option you want to use, from 1 to 8 or 'X'.

```
------------------------------------------------------------
                MAIN PROGRAMMING MENU

   1) program  - program the device from a file
   2) copy     - program from an existing device
   3) check    - check status of device
   4) checksum - calculate the checksum of a device
   5) compare  - compare the device to a design file
   6) read     - read the device data into a file
   7) append   - append new data to the device
   8) setup    - set the programmer options
   X) exit     - quit XPP
------------------------------------------------------------
Enter option '1-8' or 'X' to exit or type a command
For information on a specific  command, type 'help <command>'.

XPP ?
```

Figure 2-6 Main Programming Menu

Syntax

To run XPP in the interactive mode, do not specify any options after the command name. The syntax for starting XPP in the Interactive Mode is as follows:

```
xpp
```

The syntax for using the XPP online help is as follows:

```
help topic
```

For example, to get additional information about the Program command, type the following command:

```
help program
```

Interactive Commands

This section describes the interactive commands. You enter these commands at the `XPP ?` prompt.

baud 9600| 19200

This command allows you to change the baud rate.

count

This command changes the number of devices to program (device repetition count) and the default value in the xpp.pro file.

design design_name

This command specifies the default design name to use when programming.

device device_types

This command allows you to specify a list of devices to program. You can specify mixed device types in the same string.

help command

This command displays help information about the specified command. When you substitute the command variable with the keyword index, you get a list of commands for which help is available. Typing the following command string:

```
help index
```

generates a list of all XPP commands, as follows:

```
APPEND    BATCH     BAUD      CHECK   CHECKSUM
COMMANDS  COMPARE   COPY      HELP    MENU
PATH      PORT      PROGRAM   QUIT    READ
RESET     SAVE      SETUP     TUTOR
```

path dirs

This command sets up the searched path. XPP uses the searched path to search for a design file. XPP scans for files, in order, in all the specified directories.

port portname

This command specifies the communication port to which you connect the PROM programmer.

setup [–dev name] [–port name] [–count #] [–baud 9600|19200] [–sound on|off]

This command defines important communication parameters such as, device type, communications port, device count, baud rate, and beep (sound) mode.

sound [–on|–off]

This command sets the beep sound to either on or off when XPP detects a programming error.

reset

This commands resets the HW112 programmer.

append #devices designname

This command appends a new design into the devices.

check #devices

This command determines whether the devices are blank. If the device is not blank, XPP calculates its checksum.

compare [–n # –out name]

This command compares a device or a number of devices (specified by the –n# option) to a design file.

copy #devices

This command copies devices from a master device.

read [–bh –bin –dec –hex –rbt –n # –out name]

This command reads the contents of a device or some number of devices (–n# option) and allows you to save that information to a file in the specified format.

program [–high –low] design_name #devices

This command programs the devices from a design file.

Searching a Design File

When you specify a design file without a path or type, XPP scans the search path for a matching file. To specify a search path, set up the XACT environment as follows:

```
setenv XACT dir1:dir2:dir3
```

First, XPP scans for file types in the following order:

- *name*.bit
- *name*.mcs
- *name*.tek
- *name*.exo

Next, XPP scans for the complete file name in the following order:

- current directory
- data subdirectory
- designs subdirectory
- msg subdirectory
- next directory in the search path

Search path is disabled as the default. XPP only scans the current directory and its subdirectories. To enable the search path, enter the following command at the XPP prompt:

```
set use_path on
```

XPP Profile

The xpp.pro file stores the parameters set up in the current XPP session. XPP loads these stored parameters for the next session. XPP searches xpp.pro file using the same search path as a design file. To set up the HW112 PROM Programmer the first time, follow these steps:

1. Connect the HW112 Programmer to a serial port on your workstation and turn the power on.

Note: You can connect the HW112 to your workstation while your system is on.

When powered on, the HW112 runs a self–diagnostic. The FAIL LED blinks if there is an error.

2. At the system prompt, type the following command:

   `xpp setup`

3. Answer the questions to the setup command to establish a profile for the port name, baud rate, sound, and the default number of devices to be used.

   XPP saves this information in the xpp.pro file for subsequent sessions.

Error Messages and Recovery Techniques

This section describes the error messages that XPP can generate and provides suggestions for recovery.

`Bad Device in socket.`

The device in the socket is bad or the wrong type. Try another device or check the device type.

`Batch file` *filename* `is not located.`

The specified batch file could not be found.

`Bit file` *filename* `empty.`

There is no data in the bit file.

`Bit file` *filename* `is too small.`

The bit file is corrupted.

`Can't open bit file` *filename.*

The bit file could not be found. Check the search path.

`Can't open file` *filename.*

The file could not be found. Check the search path.

`Cannot create output file` *filename.*

The output file was not created. Check file name and file permissions.

`Cannot open log file` *filename.*

The log was not created. Check file name and file permissions.

`CEO didn't go low - Wrong device.`

Current device is the wrong type or is bad.

`Checksum Error.`

Data transmission problem. Check communication cable.

`Checksum error` *Expected x Actual y.*

Data transmission problem. Check communication cable.

`Communication failure on port` *name.*

Communication problem. Make sure the communication port name is valid and the hardware is properly connected. If using a network, make sure that you are the only one using the programmer.

`Communication line is broken.`

Communication problem. Make sure the hardware is properly connected. If using a network, make sure that you are the only one using the programmer.

`Datafile` *filename* `is corrupted.`

Data file is corrupted. Regenerate the data file.

`Device` *name*`.dsf is not located.`

The device–specific file (DSF) was not found. It must be located in the search path.

`Device failed during Verification.`

The device was programmed, but readback verification failed. The device may be bad.

`Failed to communicate with port` *name.*

Communication failure. Make sure the serial port name is valid and functional. If using a network, make sure that you are the only one using the programmer.

`Failed to establish communication.`

Communication failure. Make sure the serial port name is valid and functional. If using a network, make sure that you are the only one using the programmer.

`Failed to re-establish communication with the programmer.`

Communication failure. Make sure the serial port name is valid and functional. If using a network, make sure that you are the only one using the programmer.

`Failed to set to baudrate` *rate*.

Communication failure. Make sure baud rate is correct.

`File` *filename* `doesn't exist and it cannot be created.`

The file cannot be created. Check file name and file permission.

`File` *filename* `is empty.`

The specified file is empty.

`File` *filename* `is not located.`

File is not found. Check file name and search path.

`File` *filename* `is too large to process.`

The data file specified is too large to load in memory.

`Hardware D/A Converter failed.`

Programmer hardware problem. Turn programmer off and try again.

`Help file` *filename* `is not accessible.`

Help file is not accessible. Check file permission.

`Help file 'xpp.hlp' is not located.`

Help file was not found in the search path. xpp.hlp is installed in install_dir/files.

`Host software timed out.`

Turn off the programmer and try again. If you are connected to a network, make sure that you are the only one using the programmer.

`Initialization failure. Profile information can be changed by running 'xpp -s'.`

Internal communication failure. Make sure the serial port name is valid and functional and the baud rate is correct.

`Invalid Programmer command.`

Turn off the programmer and try again. If using a network, make sure that you are the only one using the programmer.

`Invalid baud rate` *rate*.

The specified baud rate is not supported by the system. Restart XPP with the –s option to change the baud rate.

`Invalid bit file format.`

The BIT file is corrupted. Regenerate the bit file from the LCA file.

`Invalid data in file.`

Hex file is corrupted. Regenerate the data file from the BIT file.

`Invalid data record in file` *filename*`, line` *number*.

Hex file is corrupted. Regenerate the data file from the BIT file.

`Invalid function on command line` *name*.

The command line only accepts these functions: program, append, check, compare, copy, and read.

`Invalid header for data packet.`

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Invalid host ACK for data packet.`

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Not enough memory in the programmer.`

The device–specific file (DSF) is too big. It may be corrupted. Reinstall the software and try again.

`PROM CRC test failed.`

Programmer hardware problem. Turn off the programmer and try again. If using a network, make sure that you are the only one using the programmer.

`Received` *number x hex number* `when expecting` *number* `from programmer.`

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Received NAK:` *number* `when expecting` *number* `from programmer.`

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Received only` *number* `bytes of data. Expected` *number*.

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Time out on SOH receiving.`

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Timeout on command` *name*.

Communication problem. If using a network, make sure that you are the only one using the programmer.

`Unexpected EOF in file` *filename*`, line` *number*.

Unknown PROM file format in *filename*; hex file is corrupted.

`Unexpected token` *number* `in file` *filename*`, line` *number*.

Unknown PROM file format in *filename*; hex file is corrupted.

XChecker Cable and Logic Probe

The XChecker™ cable and software supports the following Xilinx FPGAs:

- XC2000, XC2000L
- XC3000, XC3100, XC3000A, XC3000L
- XC3100A
- XC4000, XC400A, XC4000H
- XC5200

You can use XChecker to download, read back, verify design configuration data, and probe internal logic states of your designs.

This chapter describes the XChecker hardware and how to use the software on workstations and DOS-based PCs. Refer to the *Hardware Debugger User Guid*e for information about using the XChecker cable and software on windows for PCs.

The XChecker cable and software support the following capabilities.

- XChecker allows you to download a design to the FPGA on the target system.
- The optional +3-V adapter, which you can order separately, allows user target systems containing XC2000L and XC3000L low-voltage FPGA devices to interface with XChecker.
- XChecker can verify its configuration by comparing it to the original design after configuring an FPGA.
- You can probe an FPGA's internal logic with XChecker to debug your designs. Probing is the execution of a readback of all the configuration data and extracting the internal logic states of desired signals from it.

XChecker software supports all previous versions of parallel and serial download cables for download.

XChecker Hardware

The XChecker hardware is the cable assembly with internal logic, a test fixture, and a set of headers to connect the cable to your target system. Your system may require a DB-9/DB-25 adapter to connect to the host computer. Available as optional equipment is a +3-V Adapter for use with low-voltage parts.

Using XChecker requires a minimum of 512 kB of base memory for PC compatibles and a standard DB-9 or DB-25 RS-232 serial port. If you have a different serial port connection, you need to provide an appropriate adapter. Figure 3-1 shows the XChecker cable hardware and accessories.

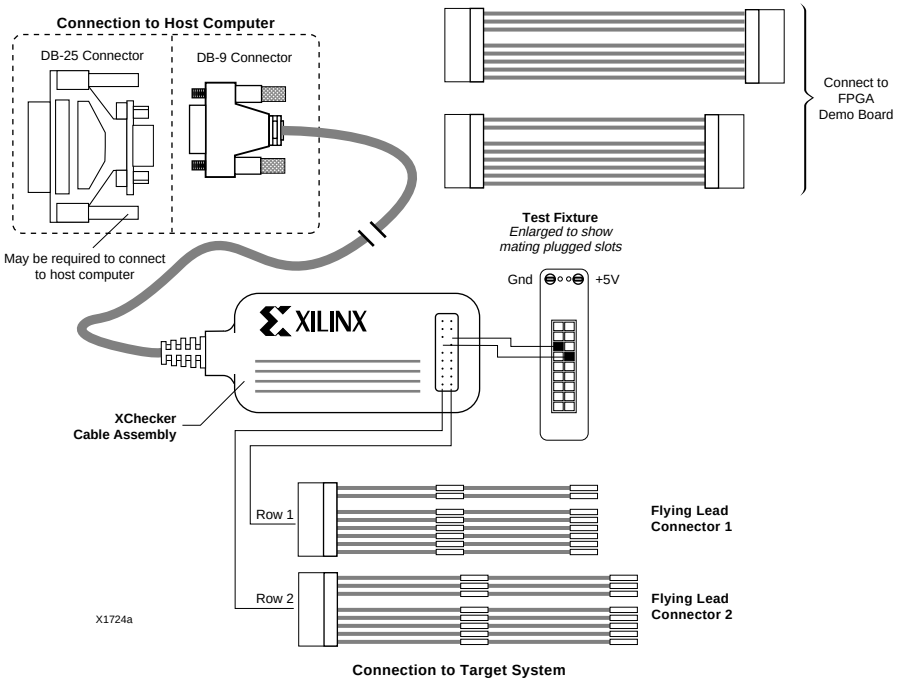


Figure 3-1 XChecker Hardware and Accessories

Figure 3-2 show top and bottom views of the XChecker cable.

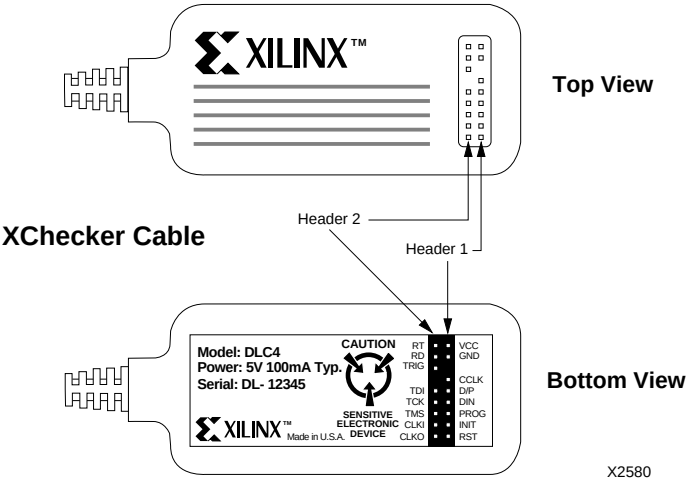


Figure 3-2 XChecker Cable

The cable assembly houses internal circuitry consisting of a Xilinx FPGA, a static RAM, and an oscillator circuit. The internal Xilinx FPGA functions as an interface between the XChecker software and the target FPGA. The static RAM stores the configuration data for download and readback. The oscillator circuit provides a system clock and allows download and readback of configuration data.

The optional +3-V adapter accepts V_{CC} supply voltages from the target system from +2.9 V to +5.25 V. The +3-V adapter contains a voltage step-up circuit that generates the +5-V supply voltage needed by XChecker.

Since the +3-V adapter can accept input voltages up to 5.25 V, there is no need to remove the adapter when moving the XChecker cable between low-voltage systems and higher +5-V systems. Except for the voltage conversion, the +3-V adapter is completely invisible to the XChecker hardware or the target system.

You can use the XChecker cable with a single FPGA or several connected in a daisy chain to download configuration data. When used to read back data or as a logic probe, the XChecker cable can only be used with one device at a time.

The XChecker cable transmits configuration data to all target FPGAs at 921 kHz. Readback of configuration data from XC2000 and XC3000 devices is also at 921 kHz. However, with XC4000 and XC5200 devices, readback can be performed at three different clock rates: 921 kHz, approximately 2.75 MHz, and approximately 5.5 MHz.

Communication between the host system and the XChecker cable is dependent on host system capability. Table 3-1 lists the valid baud rates for the supported platforms.

Table 3-1 Valid Baud Rates

	Baud Rate			
Platform	9600	19200	38400	115.2K
IBM PC	X	X	X	X
NEC PC	X			
APOLLO	X	X	X	
DEC3100	X	X	X	
SUN	X	X	X	
DEC Alpha	X	X	X	
RS 6000	X	X	X	
HP 700	X	X	X	

X indicates supported baud rate

Using Download Cables with XChecker Software

Although you are encouraged to use the XChecker cable with the XChecker software to replace previous download cables and download programs, you can use XChecker software with previous download cables.

The software supports all previous download cables, parallel and serial. However, these previous cables can only download a configuration bitstream, they cannot do readback or verification.

If you use XChecker software with a download cable, keep the following points in mind.

- Previous versions of the download cable were designed to download XC3000 and XC2000 designs, not XC4000 or XC5200 designs. They do not have a PROG pin to initiate a re-program in XC4000 or XC5200 devices. They also do not have an INIT pin to check for cyclical redundancy check (CRC) errors during configuration.

Note: To use a parallel download cable to download designs to the XC4000 or XC5200 family of devices, you must manually toggle the PROG pin Low. PROG is active when it is Low.

- For the PC, the download cable is a parallel cable, requiring connection to the parallel port. (The XChecker cable is serial.)

There are only two situations when you might prefer using previous download cables instead of the new XChecker cable.

You might have circuit boards with header connectors keyed to match the previous cable headers. However, you could use the XChecker cable with its flying lead connectors. Simply match the labeled flying leads to the equivalent signals on your system.

You may have circuit boards where power consumption is a critical factor. (The XChecker cable requires about 100 mA; the parallel cable used with PCs draws less power from the target FPGA board.) In such cases, you may use either the XChecker software or the Download program to download the bitstream.

Preparing to Use the XChecker Cable and Software

A sample LCA design called XROMs comes with the XChecker software. It is in the \xact\examples \core\xchecker directory when you install the XChecker software. This directory contains the following files; xroms.lca, xroms.ll, xroms.rpt, xroms.bit, and xroms.xnf. These files were compiled for an XC4003PC84 device. If you are using a different device, you need to compile the file xroms.xnf for the part you are using.

You can use the XROMs design and the XC4000 Demonstration Board as examples. In this sample design, the 500-kHz clock is taken from the OSC4 block and made available as the output signal, TO_CLKI. The clock is then returned to the design by the input

signal, FROM_CLKO and fed to the 4-bit counter (CB4RE).

A schematic representation of the XROMs design is shown in Figure 3-3.

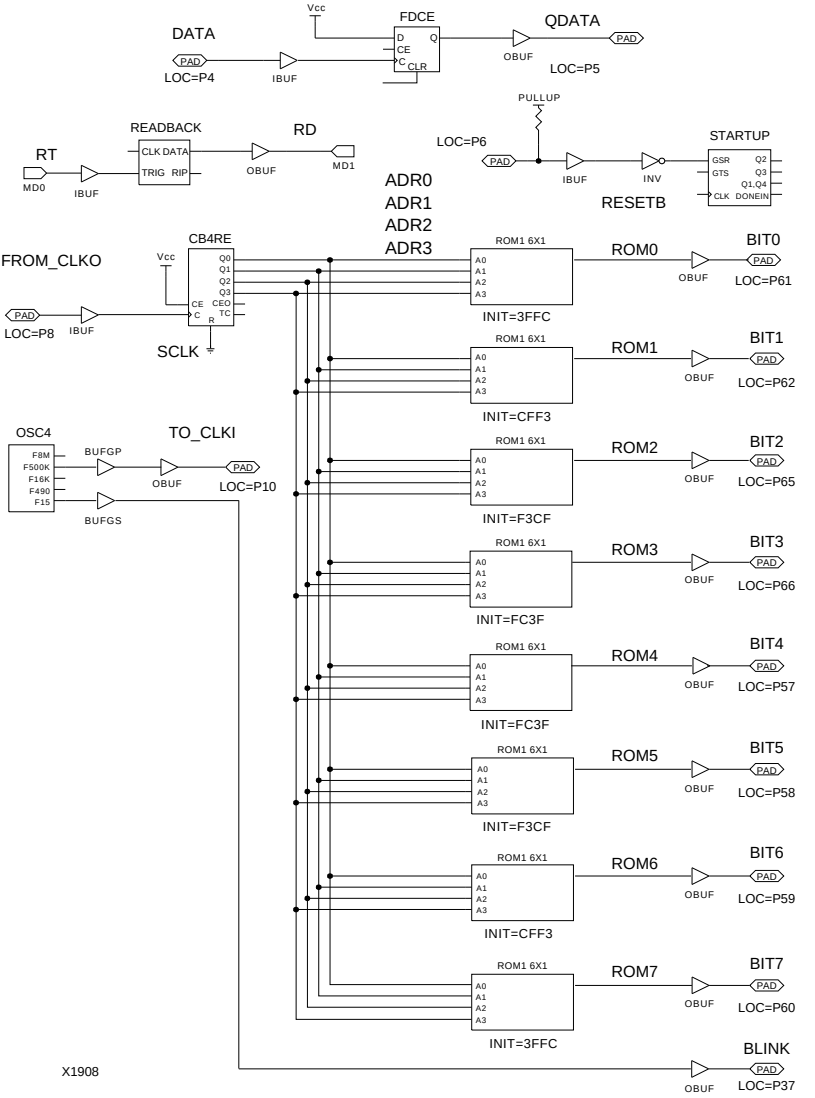


Figure 3-3 XROMs Sample LCA Design

Note: Special XC4000 and XC5200 Startup and Readback symbols are used; note their connections. These special symbols are found in the schematic editor library. Include these symbols in your XC4000 and XC5200 designs. Table 3-2 shows the pin locations for the XROMs design.

Table 3-2 XROMs Sample Design Pin Locations for XC4003APC84 FPGA

Pin Location	Pin Name
P61	: BIT0
P62	: BIT1
P65	: BIT2
P66	: BIT3
P57	: BIT4
P58	: BIT5
P59	: BIT6
P60	: BIT7
P37	: BLINK
P4	: DATA
P8	: FROM_CLKO
P5	: QDATA
P6	: RESETB
P10	: TO_CLKI

The counter in the design simply increments the addresses to the ROM cells which have been initialized to predetermined values from the schematic editor. Table 3-3 lists the predetermined values in the ROM cells.

Table 3-3 Contents of the Sample Design ROM Cells

Address	ROM Cell Contents							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0:	0	1	1	1	1	1	1	0
1:	0	1	1	1	1	1	1	0
2:	1	0	1	1	1	1	0	1
3:	1	0	1	1	1	1	0	1
4:	1	1	0	1	1	0	1	1
5:	1	1	0	1	1	0	1	1
6:	1	1	1	0	0	1	1	1
7:	1	1	1	0	0	1	1	1
8:	1	1	1	0	0	1	1	1
9:	1	1	1	0	0	1	1	1
10:	1	1	0	1	1	0	1	1
11:	1	1	0	1	1	0	1	1
12:	1	0	1	1	1	1	0	1
13:	1	0	1	1	1	1	0	1
14:	0	1	1	1	1	1	1	0
15:	0	1	1	1	1	1	1	0

If you do not want to use the sample design, follow these steps to create your own design, before using XChecker:

1. Create a design.
2. Generate a bitstream and set the configuration options.
3. Use the MakeLL option in Makebits to create a logic allocation file, *design*.ll that you can use for probing.

XChecker software can use a *design*.bit or a *design*.rbt file as input, so generating a PROM file (*design*.mcs, *design*.tek, or *design*.exo) is optional. Use MakePROM to generate a PROM file. For more information about PROM files, read the MakePROM chapter in the *Development System Reference Guide*.

Figure 3-4 illustrates the XChecker design flow.

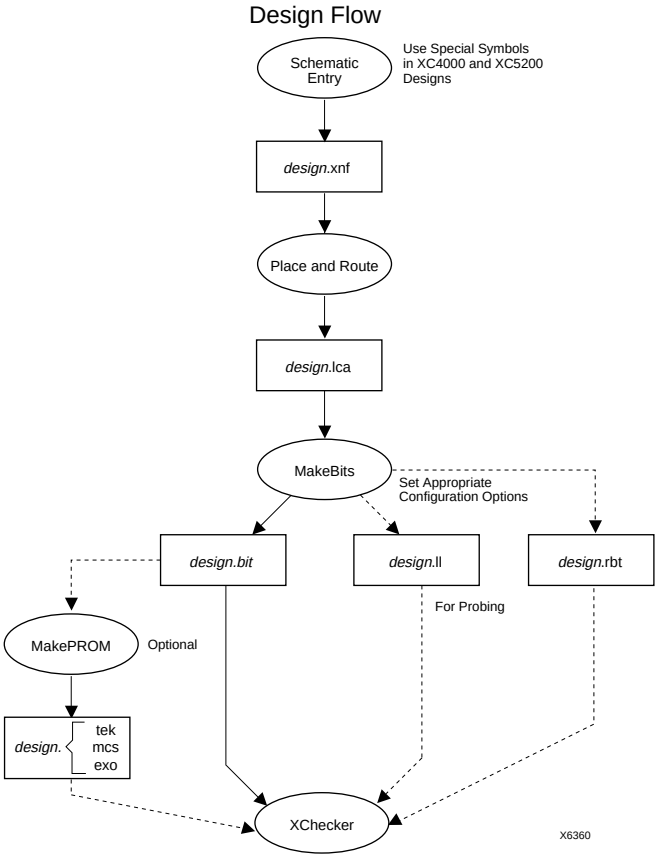


Figure 3-4 XChecker Design Flow (for DOS and UNIX)

Creating a Downloadable Design

If you have already created a configuration bitstream, please see the Using the XChecker Software section.

You can use any Xilinx-supported schematic editor to enter the design. For more information, refer to the appropriate interface user guide.

Special symbols or signals are not required for XC2000 and XC3000 designs, or for XC4000 and XC5200 designs intended for download only. XC4000 and XC5200 schematic symbols are required for the readback, verification, and logic probing of XC4000 and XC5200 designs. These symbols are provided with your schematic editor interface.

When you include the Readback symbol in the schematic with the appropriate connections, as shown in Figure 3-5, it assigns external pins to execute the readback functions RTRIG and RDATA.

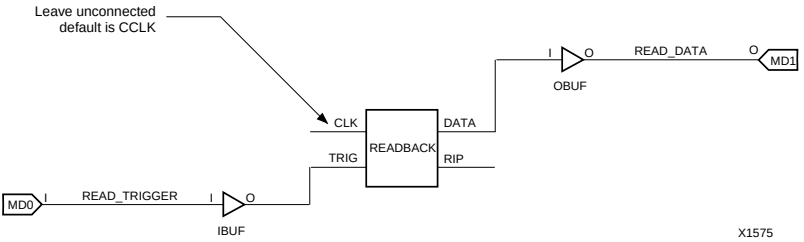


Figure 3-5 XC4000 and XC5200 Readback Symbol

Note: If you want RTRIG and RDATA to correspond to the M0 and M1 mode pins, use the special schematic symbols MD0 and MD1, as shown in Figure 3-5.

Also include the Startup symbol to select the location of the RESET pin and to set the polarity of the RESET signal to Low. The default polarity of the RESET pin for XC4000 and XC5200 parts is active-High, and XChecker software expects the RESET signal to be active-Low. Figure 3-6 shows the XC4000 Startup symbol.

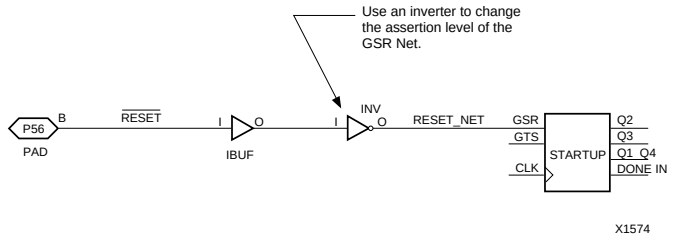


Figure 3-6 XC4000 Startup Symbol

Figure 3-7 shows the XC5200 Startup symbol.

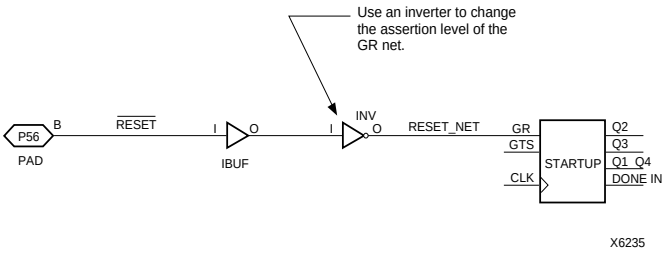


Figure 3-7 XC5200 Startup Symbol

Note: You do not need to use the MD0 and MD1 special pads to connect to the Readback symbol. You can use normal IPAD and OPAD primitives if you want the RTRIG and RDATA signals to be assigned to locations other than M0 and M1.

Note: The inverter in the Startup symbol implements the active-Low Reset asserted by XChecker, on the RST wire.

Generating a Bitstream

To download, use XMake to create a configuration bitstream.

XC4000 and XC5200 designs, or XC3000 designs used for probing, require specific MakeBits options. For this reason, specify the –m option, which runs all appropriate programs and stops before MakeBits.

Use MakeBits to generate a configuration bitstream for the design, *design*.bit with appropriate configuration options.

For XC4000 and XC5200 designs, enable readback of the device and enable a pull-up resistor for the DONE pin. Use MakeBits from the command line or from XDE.

From the command line, use MakeBits with the following options.

`MakeBits –f readcapture:enable donepin:pullup` *design*`.lca`

From XDE, select MakeBits from the Programs menu and select the following configuration options.

ReadCapture **Enable**
DonePin **PullUp**

To probe designs, you must create a logic allocation file, *design*.ll. The design.ll file provides bit locations of the values of RAM, I/O, latches, and flip-flops. To create a logic allocation file in XDE, select the MakeLL option in MakeBits from the Config menu.

If you are using MakeBits from the command line, specify the –l option. For more information about MakeBits, read the MakeBits chapter in the *Development System Reference Guide*.

Connecting the XChecker Cable

There are three simple steps for connecting the cable:

- Connect the cable to your host system RS-232 serial port.
- Perform cable self-check with the text fixture and the Diagnostics command.
- Connect the cable to your target system.

Connecting the Cable to Your Host System

The XChecker cable connects to your system RS-232 serial port. You may need a DB-9/DB-25 adapter, which accommodates most serial ports, so that you can connect the XChecker cable to your host system.

Performing Cable Self-Check

Use the test fixture and the Diagnostics command to perform a self–test. The test fixture is a small printed circuit card with a keyed header connector that fits onto the cable assembly. You use it only to validate the proper operation of the cable. Refer to the Diagnostics command description, later in this chapter.

The test fixture has connectors for +5 V and ground. It is necessary to connect +5 V and ground to these connectors when executing a diagnostic check since XChecker draws power from your target system, not from the host system.

Connection to Your Target System

You need appropriate pins on the target system for connecting the target system board to the header connection on the cable. These connectors must be standard 0.025–inch square male pins that have dedicated traces to the target FPGA control pins. You can connect to these pins with a header connector or a flying lead connector.

Note: The XChecker cable draws its power from the target system through V_{CC} and GND. Therefore, power to XChecker, as well as to the target FPGA, must be stable. Do not connect any signals before connecting V_{CC} and ground. See the Troubleshooting Guide section, later in this chapter.

Header Connector

The header connector is two standard 9-pin (8 signals, 1 key) header connectors that fit 0.025" square male pins. The pin order is listed in Table 3-4 and Table 3-5. These header connectors are keyed to assure proper orientation to the cable assembly.

Flying Lead Connectors

The flying lead connector is two flying lead header connectors with eight standard individual female connectors on one end that fit onto 0.025" square male pins. Each lead is labeled to identify the proper pin connection.

Cable Connections

Connections between the cable assembly and the target system use 16 leads. The cable has 14 signal connections, plus V_{CC} and ground. Every signal pin is not required for every function.

Once installed properly, the connectors provide power to the cable, transmit your system clock signal, allow download and readback of configuration data, and provide for logic probe sampling of configuration-related pins.

Each pin has a 100-Ω series resistor. You must provide an external pull-up resistor (approximately 10-50-kΩ) where indicated.

Table 3-4 describes the pin connections to the target circuit board.

Table 3-4 XChecker Cable Connections and Definitions

Name	Function	Connections		
		XC2000	XC3000	XC4000
VCC	Power – Supplies V_{CC} (5 V, 100 mA, typically) to the cable.	To target system V_{CC}		
GND	Ground – Supplies ground reference to the cable.	To target system ground		
CCLK	Configuration clock* – Provides configuration clock to the target system during configuration and readback.	To target system configuration clock		
D/P	Done/$\overline{\text{Program}}$* – Signals end of configuration for all families, and provides a reprogram pulse for XC2000/XC3000 parts.	Connect to the D/$\overline{\text{P}}$ pin with a 10-50-kΩ pull-up resistor.		Connect to the target system DONE pin with a 10-50-kΩ pull-up resistor.
DIN	Data In – Provides configuration data to the target system during configuration and is tristate at all other times.	Connect to the target system D_{IN}.		
PROG	$\overline{\text{Program}}$* – Provides a reprogram pulse for XC4000 parts.	Unconnected.		Connect to target system $\overline{\text{PROG}}$ with a 10-50-kΩ pull-up resistor.

Name	Function	Connections		
		XC2000	XC3000	XC4000
INIT	Initialize* – A status pin indicating the start of configuration for XC3000/XC4000 parts. For XC4000 devices, a logic zero on this pin during configuration indicates that a CRC error has occurred.	Unconnected.	Connect to the target system INIT with a 10-50-kW pull-up resistor.	
RST	Reset* – After configuration, this pin can drive Low to reset the target FPGA internal latches and flip-flops.	Connects to the target FPGA RESET pin with a 10-50-kW pull-up resistor.		User-programmable connection; requires a 10-50-kW pull-up resistor.
RT	Read Trigger* – Initiates a read back by causing a Low-to-High transition at the target FPGA RTRIG pin.	Connect to M0/RTRIG with a 10-50-kW pull-up resistor.		User-programmable connection; requires a 10-50-kW pul-lup resistor.
RD	Read Data* – Read back data from the target FPGA is read at this pin.	Connect to M1/RDATA through a 10-50-kW pull-up resistor (must be in slave serial configuration mode).		User-programmable connection; requires a 10-50-kW pull-up resistor.
TRIG	System Trigger* – A Low-to-High transition on this pin signals the XChecker to initiate a readback.	Connect to the target system read back trigger.		
TDI TCK TMS	Reserved*	Do not connect. These pins are reserved.		

Name	Function	Connections		
		XC2000	XC3000	XC4000
CLKI	Clock Input – Transmits system clock to XChecker. This clock must be from 120 kHz to 10 MHz. Connecting this pin to the clock of the target system allows synchronizing the triggering of readback to the target system clock.	Connect to source of target system clock (for synchronous readback and probing). See Figure 3-12.		
CLKO	Clock Output – The target system clock is on this pin. The clock can come from either the CLKI pin, or internally generated by the XChecker cable.	Connect to destination of target system clock (for synchronous readback and probing). See Figure 3-12.		

*These signals can be sampled and their logic states displayed. See the Status command in the Valid Commands in the Interactive Mode section.

Note: XChecker does not drive the configuration mode pins (M0, M1, M2) during configuration. You must externally specify the logic levels for these pins.

Table 3-5 lists the operation mode connections.

Table 3-5 Operation Mode Connections

Header	Pin Name	Download Only	Download, Readback, & Synchronous Logic Probe	Download, Readback, & Asynchronous Logic Probe
1	VCC	X	X	X
	GND	X	X	X
	CCLK	X	X	X
	D/P	X	X	X
	DIN	X	X	X
	PROG	X*	X*	X*
	INIT	X**	X**	X**
	RST	X	X	X
2	RT	N/C	X	X
	RD	N/C	X	X
	TRIG	N/C	Opt	Opt
	TDI	N/C	N/C	N/C
	TCK	N/C	N/C	N/C
	TMS	N/C	N/C	N/C
	CLKI	N/C	X	N/C
	CLKO	N/C	X	N/C

X = Connect as indicated in Table 3-5
N/C = Leave unconnected
* = Connect only to XC4000 and XC5200 devices
** = Connect only to XC3000/XC4000 devices
Opt = Optional. XC4000 devices can have the system trigger connected directly to the target FPGA RTRIG pin to latch the state of the device instead of waiting for the XChecker software to initiate the readback.

Connecting the Optional +3-V Adapter

When you connect the optional +3-V adapter, it is necessary to use normal ESD precautions. This adapter is static-sensitive and can be damaged by ESD energy.

Ensure that you are adequately grounded before connecting or using the +3-V adapter. Refer to Figure 3-8 and place the adapter on top of the XChecker case aligning the 18-pin female socket (J2) with the 18-pin male connector on XChecker. Grip the adapter board by the edges and gently press down until the adapter makes a solid connection to XChecker.

Figure 3-8 shows the optional +3-V adapter connected to the top of the XChecker cable.

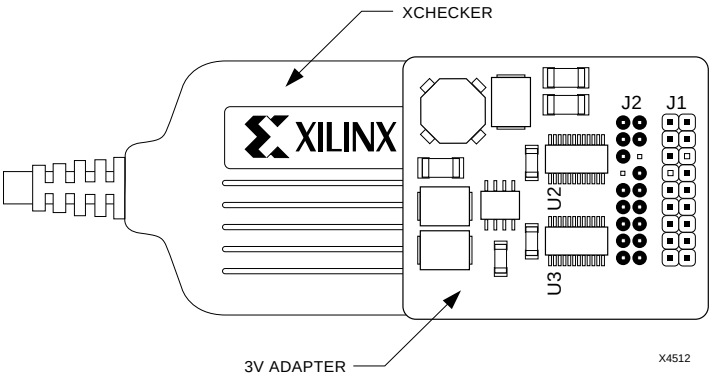


Figure 3-8 3-V Adapter Connected to XChecker Cable

Verifying the +3-V Adapter Operation

You can use the test fixture to test the +3-V adapter, by following these steps:

1. Plug the adapter onto XChecker, then plug the test fixture onto the J1 connector on the +3-V adapter.
2. Attach a power supply to the test fixture V_{CC} and GND terminals and set the voltage to about +3.3 V.

3. Invoke the XChecker software, then run the diagnostics by typing the following command:

   ```
   diag
   ```

   This command runs the diagnostics test. The test results should be identical to those obtained by running the same diagnostics test without the +3-V adapter connected to XChecker.

4. Attach the 18-signal flying wire or header block to the J1 connector on the +3-V adapter.

   XChecker operation is unchanged.

5. Attach the XChecker cable with the +3-V adapter to your target system.

Using the +3-V Adapter with XC2000L and XC3000L Parts

To use the +3-V adapter with XC2000L and XC3000L devices, follow these steps:

1. Attach the 18-signal flying wire or header block to the J1 connector on the +3-V adapter.

   XChecker operation is unchanged.

2. Attach the XChecker cable with the +3-V adapter to your target system.

The configuration of the XC2000L and XC3000L devices is the same as the XC2000, XC3000, XC3000A, and XC3100 devices.

The readback clock speed of the XC3000L devices has been slowed because of lower V_{CC} supply voltage. If the supply voltage of the target system is lower than +3 volts, you might see the following error message when reading back or verifying a configured XC3000L device.

```
XCHECKER? verify
Design design_name has 128 probeable signals.
Readback 1847 bytes of configuration.
Verifying datafile design_name...MISMATCHED Total
of 405 bits mismatched.
```

Connecting for Download

Connect the XChecker cable to the host system and your target FPGA device, as shown in Figure 3-9. This step only allows you to download the design. You cannot verify the design.

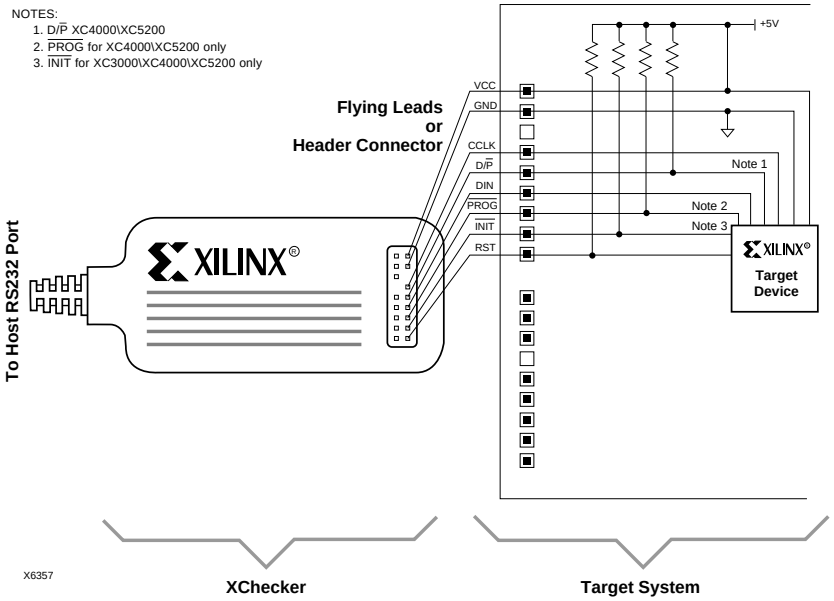


Figure 3-9 Downloading Configuration Data

If you are using the sample design, XROMs, you need a jumper between the device pins TO_CLKI (P10) and FROM_CLKO (P8) to provide a system clock. You can determine the location of these pins from the xroms.rpt file.

Note: The M0, M1, and M2 mode pin switches must be on to place the FPGA in slave serial mode.

Connecting for Verification

You can connect XChecker for download and verification or only verification of a configured FPGA. To download and verify an FPGA, connect XChecker as shown in Figure 3-10.

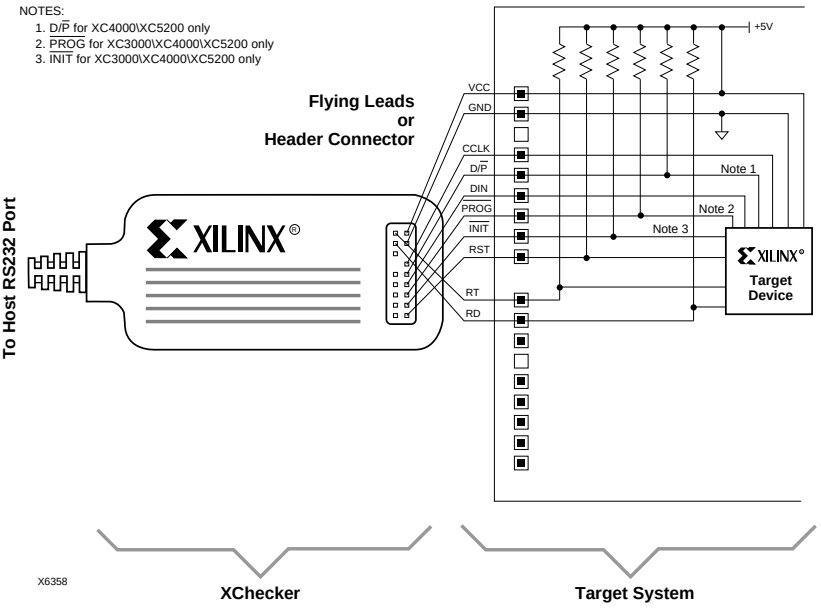


Figure 3-10 XChecker Cable Connections for Downloading and Verification

To verify a previously configured FPGA, connect XChecker as shown in Figure 3-11.

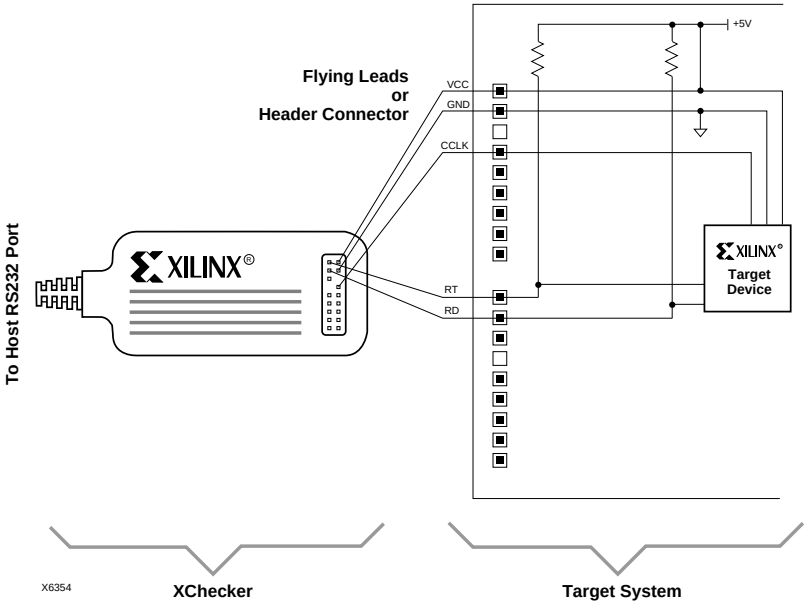


Figure 3-11 XChecker Cable Connections for Verification Only

Use the following guidelines when using XChecker for verification.

- If you are using the sample design, remember to place a jumper between the signals TO_CLKI and FROM_CLKO to provide a system clock. (Since verification of the configuration bitstream is intended, it makes no difference whether you connect the clock from the target system to XChecker.)
- Remember to connect XChecker RT and RD pins to the FPGA (programmable) RTRIG and $\overline{\text{RDATA}}$ pins. If you used the MD0 and MD1 primitives to place these signals, you can determine the pin locations (M0 and M1) using the pinout tables for the appropriate FPGA in the *The Programmable Logic Data Book*. If you used regular IPAD/OPAD primitives, consult the *design*.rpt file for pin locations.

Connecting for Synchronous Probing

Connect XChecker to the target FPGA, as shown in Figure 3-12. This step allows you to download, probe, and verify your design.

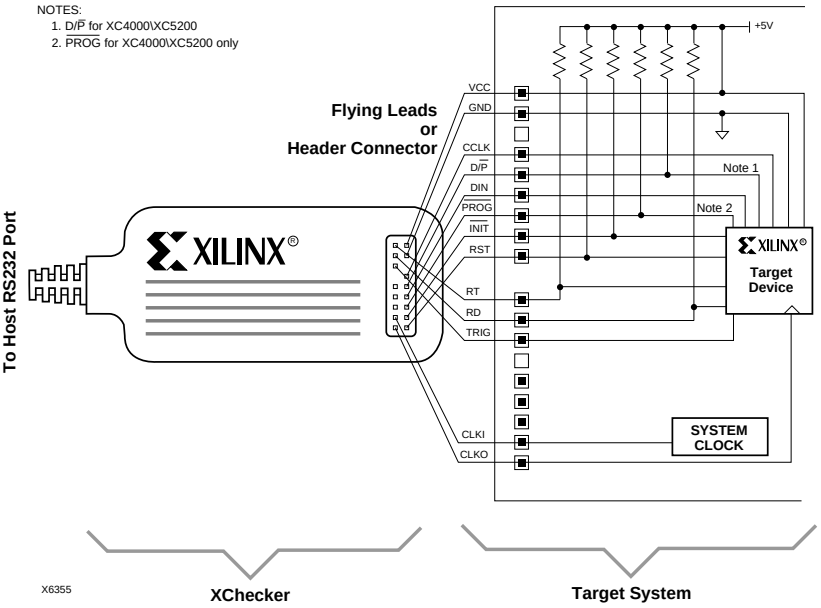


Figure 3-12 XChecker Cable Connections for Synchronous Probing

Note: XC3000 design internal logic probing must be synchronous, which means that XChecker interrupts your FPGA system clock, executes readback of the FPGA internal logic, and reapplies your target FPGA system clock.

Use the following guidelines.

- Be sure to connect the XChecker CLKO and CLKI pins, as shown in Figure 3-12, from the clock source to the XChecker CLKI and from the CLKO to your target FPGA. This connection allows XChecker to control the target FPGA system clock. If you are using the sample design, consult Table 3-2 (or the xroms.rpt file) and connect a jumper between the signals FROM_CLKO and TO_CLKI to the XChecker CLKO and CLKI pins, respectively.

- To use an external trigger such as the terminal count of a counter or some other condition in your target board to initiate a readback, make sure to connect this signal to the XChecker TRIG pin. You can also use an internal trigger (pressing a key on the keyboard) to initiate a readback.
- Make sure to connect the XChecker RT and RD pins to the FPGA RTRIG and RDATA pins, respectively. If you used the MD0 and MD1 symbols, consult the pinout tables for the appropriate part in the *The Programmable Logic Data Book*. If you used normal PADs, consult your *design*.rpt file.

Connecting for Asynchronous Probing

Connect XChecker to the target FPGA, as shown in Figure 3-13. This step allows the target system to run while a readback is executed. (You can use the sample design in these examples.) Pay particular attention to the clock and trigger connections.

- Be sure to provide a system clock. If you are using the sample design, provide a jumper between the TO_CLKI and FROM_CLKO signals to leave the system running. There is no need to provide the system clock to XChecker because readback is executed independently from the system clock.
- Connect an external trigger to the XChecker TRIG pin. If you are using the XROMs sample design, connect the BIT3 signal (pin P66 if you are using the sample design bitstream for an XC4003PC84) directly to the XChecker TRIG pin.

Note: Connecting the system trigger to the XChecker TRIG pin will insert one extra clock before the readback is initiated.

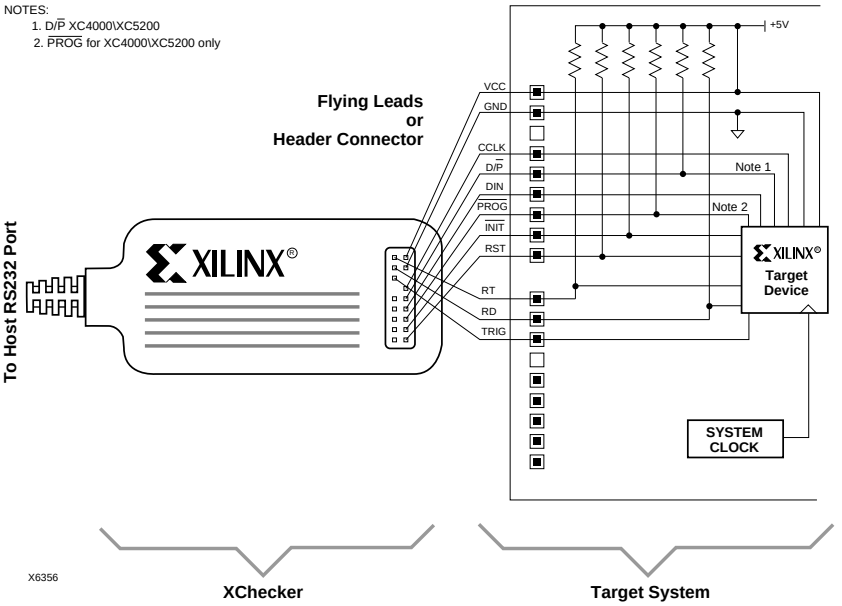


Figure 3-13 XChecker Cable Connections for Asynchronous Probing

Using the XChecker Software

This section describes the XChecker files and commands.

XChecker Files

You must become familiar with the following files, which are used by the XChecker software.

design.bit

The *design*.bit file contains the configuration information for the target FPGA design. This file is generated by using MakeBits and must be located in the same directory in which you started the XChecker software.

design.ll

The *design*.ll file is the logic allocation file containing the bitstream positions of flip-flops, latches, RAM and CLB outputs, IOB inputs, and IOB registered outputs. This file must be located in the same directory in which you started the XChecker software.

If you intend to probe the target FPGA internal logic states, refer to the MakeLL command in the "MakeBits" chapter of the *Development System Reference Guide*.

design.rbt

The *design*.rbt file is the ASCII equivalent of the *design*.bit file. You can use the *design*.rbt to configure a target FPGA.

xchecker.pro

The xchecker.pro file contains the default values for all XChecker options: part, design, baud, and port. These option values are updated at the end of every XChecker session. For XChecker to recognize an xchecker.pro file, it must be located in the same directory in which you started the XChecker software.

parttype.ll

The part type files are generic logic allocation files for all of the part types that might be targets, such as 4005pc84.ll. They are provided by the XChecker software and are needed for verification and probing of the bitstream.

batch_file.cmd

The batch files are text files used to execute commands in the batch mode, and the extension ".cmd" is required.

design.exo, design.mcs, design.tek

These design files contain the configuration information for each design. These files are optional since the XChecker software can take a BIT file as input, but are required for daisy chains. You create these files in MakePROM.

Invoking XChecker

You can start XChecker using any of the following methods:

- Command line entry from the system shell. Only download and readback are supported. You cannot interactively probe the internal logic.
- Command line entry from XDM.
- Interactive commands from the system shell. This mode offers additional commands for download and readback and also allows you to probe the internal logic states of the target FPGA device.

Downloading

You can download a design after connecting the XChecker cable to the host system and target FPGA. To download the XROMs sample design, enter the following command at the operating system prompt.

```
xchecker xroms
```

Note: The device you are downloading with the XChecker cable must be in serial slave mode.

When you do specify any options, the XChecker software selects the port where the cable is connected and sets the baud rate to the maximum allowed by the platform. You can modify the communication port and baud rate by changing the appropriate settings in the xchecker.pro file.

1. To download in an interactive mode, enter the following command at the system prompt.

   ```
   xchecker
   ```

   You see the following message on the screen:

   ```
      Xilinx (R) XChecker 5.2.0 - Download/Readback
   LCA via download cable
      Copyright (C) Xilinx, Inc. 1991-1995. All
   rights reserved.
   Cable ID type is 'SERIAL-Readback'
      Cable is connected to 'com1'
      Baud rate is 115200
   ```

2. To load the sample design, enter this command string:

   load xroms

   The following message appears on the screen:

   ```
   About to download 'xroms.bit'.
   Press ENTER key to continue or Q to quit:
   ```

3. After you press the Return key, the following message displays:

   ```
   Total of 11875 bytes transmitted.
   DONE signal went high.
   Transmitting time = 1.81 secs
   ```

   You can also access XChecker from within XDM. To do so, select XChecker from the Verify menu, or type **xchecker** at the command line.

   The total bytes transmitted, the time required, the baud rate, and the port used varies depending on your selection of part type and platform.

Note: Connecting the XChecker cable to download only a bitstream does not allow verification of the bitstream after configuration. You cannot use the –v option to do readback.

Verifying

After you have properly configure an FPGA, you can verify its configuration and compare it to your original design.

In most applications, verification is not needed, but this feature can be helpful with designs that experience extremely unstable or noisy V_{CC} conditions, or if your design standards require that stored data be read back periodically. Refer to the *Development System User Guide* for information about Readback and Verification implementations.

To download and verify the XROMs sample design, enter the following command string at the operating system prompt.

xchecker –v xroms

Specifying the –v option causes the XChecker cable to download the file *design*.bit to the target FPGA and initiate a readback immediately after the configuration data has been downloaded. See the waveforms in Figure 3-14.

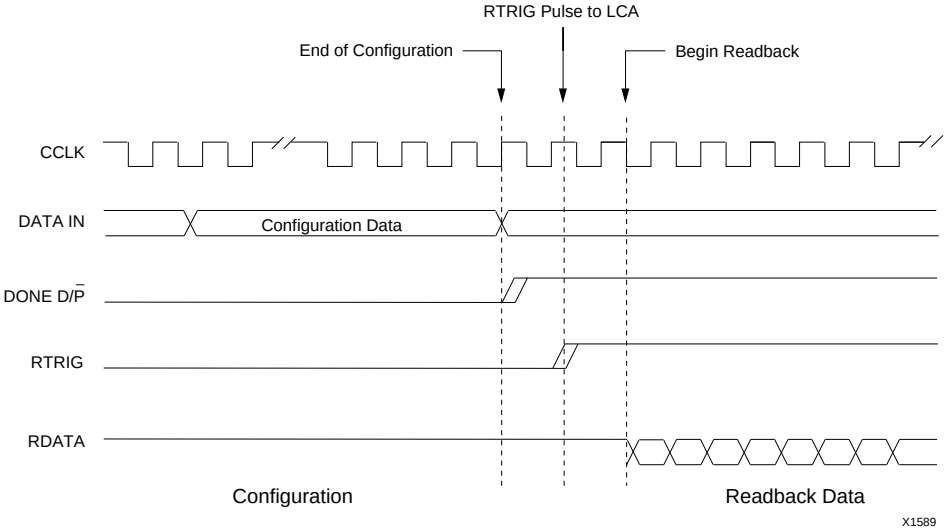


Figure 3-14 Waveforms for Verification After Download

To execute a readback after the device has been in operation, use the interactive commands, as follows:

```
xchecker
```

This command invokes the interactive mode, and the `XCHECKER ?` prompt appears.

`XCHECKER ?` **`load`** *design_name*

The Load command downloads *design*.bit to the target FPGA. If you want a readback after the target FPGA is in operation, you can execute the Verify command.

`XCHECKER ?` **`verify`** *design_name*

This command initiates a readback, and compares the data to the *design*.bit file. See Figure 3-15.

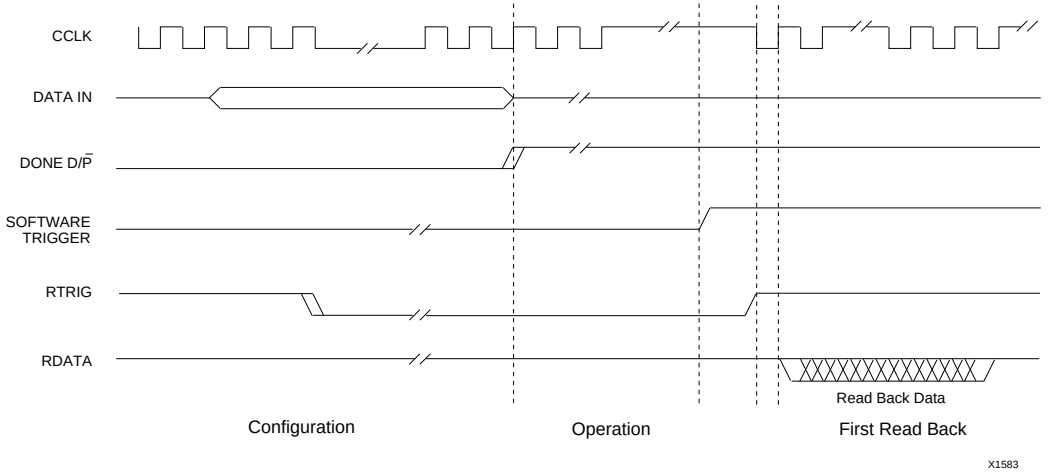


Figure 3-15 Waveforms for Verification Some Time After Download

Probing

To probe the FPGA internal logic, execute a readback of all the configuration data and extract the internal logic states. You might want to probe the internal logic to debug your designs using XChecker as an in-circuit emulator.

XC2000 and XC3000 Designs

You must stop the FPGA system clock while readback is in progress to obtain a static "snapshot" of the internal logic states. XChecker controls the FPGA system clock if appropriate clock connections have been made. See Figure 3-12.

Example 1

Suppose you want to determine the eight consecutive values of a design, s_0, s_1, s_2, and s_3, after an interrupt to a microprocessor in the board occurs. The interrupt signal is IR, connected to the XChecker TRIG pin.

1. Enter the xchecker command at the operating system prompt.

   xchecker

2. To load the design, enter the following command.

   XCHECKER ? **load** *design*

   This step downloads *design*.bit to the target FPGA. After the target FPGA is in operation, probing internal flip-flops and latches is possible.

   For instance, the following command would group s_0 through s_3 into a signal named s_bus:

   ```
   group s_bus s_0 s_1 s_2 s_3
   ```

3. Set the display mode for s_bus to hexadecimal.

   XCHECKER ? **pick –hex s_bus**

4. Apply the internal system clock to the target FPGA, after a trigger has been acknowledged, at a rate of 2.75 MHz.

   XCHECKER ? **clock –int –speed 3**

5. Set the readback trigger (IR) to be external and detected as a Low-to-High transition at the cable TRIG pin.

   XCHECKER ? **trigger –auto –clock 32 16 –timeout 3**

   After doing a readback, and upon detecting a trigger, XChecker generates 32 system clocks and initiates a readback. Then, if several snapshots are read back, 16 clocks are inserted between each readback. If a trigger is not received within three seconds after the Readback command is executed, the XChecker system times out and does not execute a readback.

6. Use the Readback command as follows:

   XCHECKER ? **readback 8**

   XChecker issues 32 system clocks on the CLKO pin (per the Trigger command) and then reads back eight consecutive bitstreams (snapshots) after acknowledging the trigger and inserting eight internally generated system clocks between each snapshot. See the waveforms in Figure 3-16.

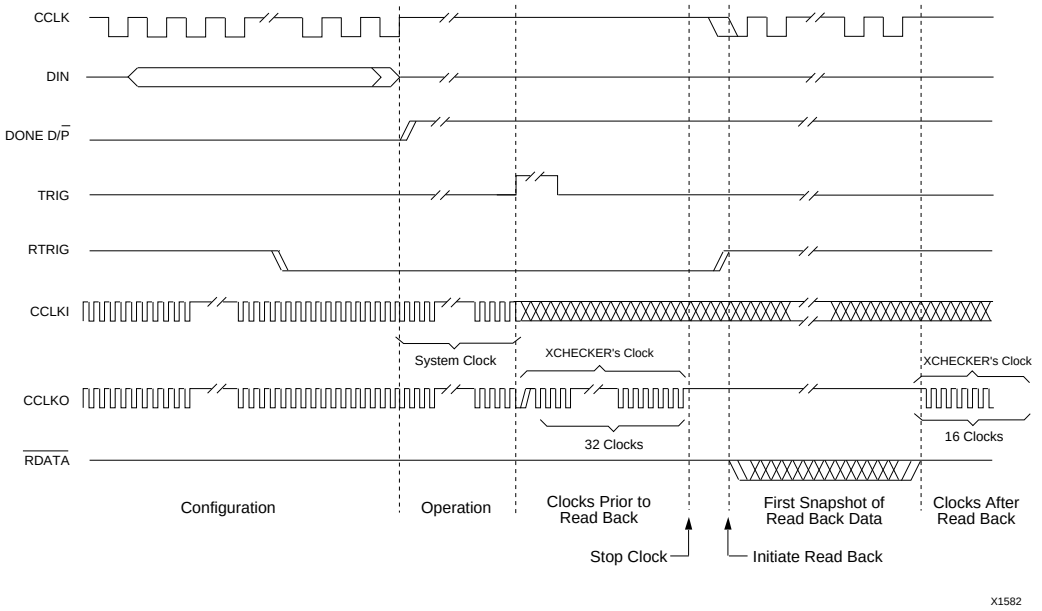


Figure 3-16 Waveforms for Collecting 1-of-16 Snapshots

7. Display the node values.

   XCHECKER ? **show -sn 8 8**

   This step e Show displays the node values, starting with snapshot 8 and ending with snapshot 16 as follows:

```
    S
    B
    u
    s
8:  A
9:  B
10: C
11: D
12: E
13: F
14: 0
15: 1
16: 2
```

Example 2: Single-Step Analysis

Suppose you want to debug a design containing a state machine. You have the input variables A, B, C, and D, and the outputs signals LOAD and READ. You want to stop the state machine, initialize it and single step the clock, executing a readback in each state.

1. Type the XChecker command without options at the system shell. to enter the interactive mode.

   xchecker

   In the interactive mode, all XChecker functions are available.

2. Load the design using the Load command.

   XCHECKER ? **load 3kstate**

3. Add the desired signals to the display list. The default format is binary.

   XCHECKER ? **pick A B C D LOAD READ**

4. Now, set up the trigger. Use your keyboard to initiate a readback (the manual option).

   XCHECKER ? **trig –man**

   You must determine the clock source to your system. You can choose between the clock already in your board (–ext), which has been fed to XChecker, or one generated by XChecker (–int). For this example, the XChecker clock is used at speed 1 (921 kHz).

5. Set the clock source.

   XCHECKER ? **clock –int –speed 1**

6. To single step clock in the design, executing one readback between clock sequences, use the following sequence.

clock –stop	stop the clock
reset	initialize state machine
readback	read one snapshot
clock 1	send one clock
readback –a	append to previous snapshot
clock 10	send 10 clocks
readback –a	append to previous snapshot
clock –resume	resume clock

7. To show the collected data for this sequence, use the Show command.

   ```
   XCHECKER ? show –vdata
   ```

   The collected data appears as follows.

   ```
        LR
        OE
        AA
   ----ABCDDD
     0:000010
     1:010110
     2:011001
   ```

XC4000 and XC5200 Designs

You can read back an XC4000 or an XC5200 FPGA with two methods, as follows:

- Synchronous probing

  You can stop the system clock and execute a readback as is done with the XC3000 and XC2000 devices. This method is called synchronous probing, as each readback is synchronized to the system clock.

- Asynchronous probing

  You can keep the system clock running while the readback is in progress. The XC4000 and XC5200 devices latch their internal logic states in special latches when a readback is requested. Therefore, the system clock need not be stopped to collect coherent data. This method is called asynchronous probing because readback is independent of the system clock.

Synchronous Probing

In the following examples you pass control of the system clock to XChecker to synchronize readback.

You can use any combination of clock (target or XChecker) and trigger (manual, none, or auto). In the following examples, the focus is not on the clock but on the use of the Trigger command. Use the Clock command to select either the internal XChecker clock or the external target system clock.

Using the Keyboard to Initiate a Readback

To use the keyboard to initiate a readback, follow these steps:

1. Enter the following command at the system shell.

   ```
   xchecker
   ```

2. Use the Load command to download the XROMs design.

   ```
   XCHECKER ? load xroms
   ```

3. Now, group the signals ROM 0-7 into a more convenient name such as “pattern.”

   ```
   XCHECKER ? group pattern ROM7 ROM6 ROM5 ROM4 ROM3 ROM2 ROM1 ROM0
   ```

4. Prepare to display the new signal “pattern” in binary format.

   ```
   XCHECKER ? pick pattern
   ```

5. Next, set up the trigger for the subsequent readback.

   ```
   XCHECKER ? trig –manual –reset –clock 0 1
   ```

   This step causes XChecker to reset the FPGA before readback and inserts one clock pulse between each subsequent readback.

6. To readback 16 snapshots, enter the following command:

   ```
   XCHECKER ? readback 16
   ```

   XChecker waits for a Return key signal before starting a readback. After the readback is completed, it displays the “pattern” signal on the screen with the Show command. The default display format is a vertical listing of signal data.

7. Use Show command with the –hdata option to display the signal.

   ```
   XCHECKER ? show –hdata
   ```

   The following pattern corresponds to the contents of the 16 ROM cells beginning at address 0.

```
p
a
t
t
e
r
n
0:01111110
1:01111110
2:10111101
3:10111101
4:11011011
5:11011011
6:11100111
7:11100111
8:11100111
9:11100111
10:11011011
11:11011011
12:10111101
13:10111101
14:01111110
15:01111110
```

Using an External Trigger to Initiate a Readback

To use an external trigger, you need a toggle switch or other means to provide a Low-to-High transition at the XChecker TRIG pin.

For the following examples, connect the BIT3 signal (pin P66 if you are using the sample design bitstream for an XC4003PC84) directly to XChecker's TRIG pin. The BIT3 signal has only one Low-to-High transition when the address (ADR0-3) increments from 9 to 10 (hex 9 to hex A), as shown in the following pattern. Therefore, because XChecker inserts one clock after the TRIG occurs, using BIT3 as a trigger should always produce the contents of the ROM address 11 to be read back in the BIT 0-7 signals.

Note: XChecker inserts one clock after acknowledging a TRIG before a readback is initiated.

```
76543210
0:01111110
1:01111110
2:10111101
3:10111101
4:11011011
5:11011011
6:11100111
7:11100111
8:11100111
9:11100111
10:11011011<--  Low-to-High in BIT3 signal at address 10
11:11011011<--  Signals at address 11 read
12:10111101
13:10111101
14:01111110
15:01111110
```

Follow these steps to use an external trigger to initiate a readback:

1. Enter the Xchecker command at the system shell.

   xchecker

2. Download the XROMs design.

   XCHECKER ? **load xroms**

3. Group the signals ROM 0-7 into a more convenient name such as, pattern.

   XCHECKER ? **group pattern ROM7 ROM6 ROM5 ROM4 ROM3 ROM2 ROM1 ROM0**

4. Prepare to display the new “pattern” signal in binary format, which is the default.

   XCHECKER ? **pick pattern**

5. Set up the trigger for the subsequent readback. An external trigger on the TRIG pin initiates a readback.

   XCHECKER ? **trig –auto**

   There are no clocks issued before or after the snapshot is collected, because the clock parameter was not used with the Trig command.

6. Use the Readback command to readback one snapshot.

   ```
   XCHECKER ? readback
   ```

   XChecker waits for the Low-to-High transition of BIT3 and initiates a readback.

7. Use the Show command to display the “address” and “pattern” signals on the screen.

   ```
   XCHECKER ? show –hdata
   ```

   The pattern displays as follows:

   ```
   a  p
   d  a
   d  t
   r  t
   e  e
   s  r
   s  n
   0  101111011011
   ```

   The first four bits correspond to address 11 (hex B) and the last eight to the contents of the ROM cells at that address (pattern).

Initiating a Readback Without a Trigger

To initiate a readback without using a trigger, follow these steps:

1. Enter the Xchecker command at the system shell:

   ```
   xchecker
   ```

2. Download the XROMs design.

   ```
   XCHECKER ? load xroms
   ```

3. Group the ROM 0-7 signals into the name pattern.

   ```
   XCHECKER ? group pattern ROM7 ROM6 ROM5 ROM4 ROM3
   ROM2 ROM1 ROM0
   ```

4. Prepare to display the new “pattern” signal in binary format, which is the default.

   ```
   XCHECKER ? pick pattern
   ```

5. Set up the trigger for the subsequent readback.

```
XCHECKER ? trig –none –clock 8 1 –reset
```

With trig –none, an external trigger is not expected for a readback. XChecker resets the target device and inserts eight clocks before the first readback and one after that using the command in step 6.

6. To readback 16 snapshots, use the Readback command.

```
XCHECKER ? readback 16
```

The readback sequence starts immediately since trigger –none was selected.

7. Display the signal pattern on the screen, use the Show command.

```
XCHECKER ? show –vdata
```

Since the Readback command issued eight clocks before the first snapshot, the pattern corresponds to the contents of the eight ROM cells beginning at address 8 and wrapping around to address 0.

The pattern displays as follows:

```
p
a
t
t
e
r
n
0: 11100111
1: 11100111
2: 11011011
3: 11011011
4: 10111101
5: 10111101
6: 01111110
7: 01111110
8: 01111110
9: 01111110
10:10111101
11:10111101
12:11011011
13:11011011
14:11100111
15:11100111
```

Asynchronous Probing

The following example uses the BIT3 signal as an external trigger with the –auto option to illustrate a feature of the XC4000 and XC5200 FGPA. XC4000 and XC5200 devices latch the internal logic state in special latches upon receiving a Low-to-High transition at the RTRIG pin. With XC4000 and XC5200 devices, it is not always necessary to connect the CLKI and CLKO pins to the target system.

Observe that the BIT3 signal experiences only one Low-to-High transition when the address (ADR 0-3) increments from 9 to 10 (hex 9 to hex A). Using BIT3 as a trigger always produces the contents of the ROM address 11 (hex B) to be read back in the signals BIT 0-7.

1. Enter the Xchecker command at the system prompt:

   xchecker

2. Download the XROMs design by entering this command string:

   XCHECKER ? **load xroms**

3. Group the signals ROM outputs (ROM 0-7) into a more convenient name, “pattern,” and the address signals (ADR 0-3) into “address.”

   XCHECKER ? **group pattern ROM7 ROM6 ROM5 ROM4 ROM3 ROM2 ROM1 ROM0**

   XCHECKER ? group address ADR3 ADR2 ADR1 ADR0

4. Prepare to display the new “pattern” and “address” signals in binary format, which is the default.

   XCHECKER ? **pick address pattern**

5. Set up the trigger for the subsequent readback. An external trigger initiates a readback.

   XCHECKER ? **trig –auto**

6. To capture one snapshot, use the Readback command.

   XCHECKER ? **readback**

   The readback occurs on the next Low-to-High transition of BIT3.

7. To display the “address” and “pattern” signals on the screen, use the Show command.

```
XCHECKER ? show –hdata
```

The pattern displays as follows:

```
a  p
d  a
d  t
r  t
e  e
s  r
s  n
0: 101111011011
```

The first four bits correspond to “address” (hex B) and the last eight bits correspond to the contents of the ROM cells at that address “pattern.”

Probing RAM Bits in an XC4000 Part

Probing internal RAM bits is different than probing other signals since internal RAM bits are not shown in the *design*.ll file.

You can probe the value of internal RAM bits with either of the following methods:

- You can probe the output net of the RAM cell and address the RAM bit in question.
- You can probe the individual RAM bit(s) by specifying the “bit name” at the Probe command.

Of the two methods, the first one is easier to implement. However, the second method allows you to probe any RAM bit at any time, without having to address it.

Example of Probing Individual RAM Bits

You must use the EditLCA program in XDE to probe individual RAM bits to determine the CLB location of the RAM cell. You use the CLB location to identify the RAM bit.

For example, you have a design that implements a RAM using the XACT Unified Libraries element RAM16X4 (available from your schematic library). Suppose that the output nets of the RAM are called *my_ram 0* through *my_ram 3*. As an example you may want to probe the bits at address 7. PPR implements this RAM macro using

two CLBs. To find which CLBs PPR used to implement the RAM, do the following steps:

1. Invoke XDE by typing the following command:

   xde

2. Select the design.
3. Select the EditLCA command from the Programs menu.

   (Refer to the "XDE" Chapter in the *Development System Reference Guide* for more information.)

4. Find the output nets of the RAM by using the Findnet command from the Screen menu.
5. At the prompt, enter the names of the output nets. Use the wildcard symbol (*) to represent all outputs.
6. Determine the CLB position of the RAM.

   The mouse points to the CLB where the output net is located. The location of the CLB (row and column number) displays at the command line. An example of a CLB location is CLB_R2C3.

7. Use the CLB location to identify the desired bits. The following notation identifies RAM bits in a CLB.

   CLB_LOCATION.M F_or_G *bit_number*

   An example of CLB_LOCATION is CLB_R2C3. The character “M” is mandatory to indicate that a memory bit follows. F_or_G are the characters “F” or “G,” depending on the type of function generator that implemented the RAM. In a CLB implementing two 16x1 RAMs, use “F” for nets sourced by the X pin and “G” for nets sourced by the Y pin. Omit this character in CLBs implementing 32x1 RAMs. The *bit_number* is a number from 0 to 15.

For example, suppose that in your 16x4 RAM, the output nets are sourced as follows:

my_ram0 sourced by CLB_R2C3.X

my_ram1 sourced by CLB_R2C3.Y

my_ram2 sourced by CLB_C3R3.X

my_ram3 sourced by CLB_C3R3.Y

The bits at address 7 are identified as follows:

BC.MF7 for bit 0 at address 7

BC.MG7 for bit 1 at address 7

BC.MF7 for bit 2 at address 7

BC.MG7 for bit 3 at address 7

In another example design, where CLB_CxRy is part of a 32 x m (0-31) RAM, the notation for bit *n* is as shown below.

CLB_CxRy.M*n*

where *n* is any number from 0 to 31.

8. Use the Pick command to add the desired bits to the "pick set."

   pick –add BC.MF7 BC.MG7

Displaying Readback Data in the Viewlogic Viewwave Environment

You can display readback data from the target system in the Viewlogic Viewwave environment. You use the XChecker Export command in the interactive mode. The following sequence is an example of a typical display session.

1. Download a bitstream file:

   load *design*

2. Select the signals that you want to probe:

   probe –add tog1 tog2 tog3

3. Select signals that you want to display:

   pick –add tog1 tog2 tog3

4. Read back three snapshots:

   ```
   readback 3
   ```

5. Write the results to a GEN file:

   ```
   export –v design
   ```

6. Open the GEN file in your waveform viewer.

There is no time reference, because XChecker reports sets of readback data, not real-time information.

Command-Line Options

This section describes the XChecker command-line options. The data files are configuration bitstream files in BIT or RBT format, or any one of the supported PROM formats, TEK, EXO, or MCS. When you do not specify any options or data files, the XChecker system defaults to the interactive mode. If you do not specify any options but you do specify a data file, a default set of options specified in the xchecker.pro file sets the port.

The command-line syntax is as follows:

`xchecker options` *datafile*

Note: You can abbreviate all options to the minimum number of distinctive characters in the option name.

Commands, options, and signal names are not case-sensitive.

–batch Batch Mode Operation

Syntax –batch *bat_file*.cmd

Abbreviation b

The Batch option executes commands in batch mode. The *bat_file* must have a ".cmd" extension and contain valid XChecker commands, including interactive commands. You can add comments to files by using the # symbol, either on the command line or on a new line.

–h The Help Option

Syntax –help

Abbreviation h

The Help option displays command line usage information.

–pa Specify Part Type

Syntax –part *parttype*

Abbreviation pa

The Specify Part Type option defines the part to be used such as, 3090pc84, 3020APC68, or 4005PC156. This option is only required when the configuration information is stored in a PROM file (MCS, TEK, or EXO) and you want to do readback, verification, or probing.

–po Specify Port Name

Syntax –port *portname*

Abbreviation po

The Specify Port Name option identifies the port connection for the XChecker cable. If you do not specify this option, the default option AUTO, searches for the cable connected to any port, parallel or serial. Valid ports for supported platforms are listed in Table 3-6.

Table 3-6 Valid Ports for the XChecker Cable

Platform	Communication Ports			
IBM PC	com1	com2	lpt1*	lpt2*
NEC	com 1			
Apollo	/dev/sio1	/dev/sio2		
DEC3100	/dev/ttyd0	/dev/ttyd1		
Sun	/dev/ttya**	/dev/ttyb**		
DEC Alpha	/dev/tty00	/dev/tty01		
RS6000	/dev/tty0	/dev/tty1		
HP700	/dev/tty00	/dev/tty01		

* Use with the parallel download cable only (no Readback).
**ttya and ttyb must be readable and writable to ensure a proper connection.

–v Verify Download and Readback

Syntax –v

Abbreviation v

The Verify Download and Readback option executes a download and readback of the current FPGA design for verification. XChecker reads the configuration from the FPGA and compares it to the original bitstream. If you do not specify this option, readback is not executed after configuration.

Interactive Mode Commands

This section describes the XChecker interactive mode commands. To use the interactive mode commands, you enter **`xchecker`** at the system prompt.

Note: You can abbreviate the commands using the least number of distinctive characters, as with the command line options, but you must use at least two characters. You can repeat the previous command using either an equal sign, "=," or an exclamation point, "!."

Batch — Execute in Batch Mode

Syntax batch *bat_file*.cmd

Abbreviation bat

The Batch command executes commands in a batch mode. The *bat_file* must have a ".cmd" extension and contain valid XChecker commands. Use the pound sign, "#" to precede comment lines in the batch file.

Examples

The following examples show two methods of using the Batch command from the XChecker prompt:

```
batch bat_file.cmd
< bat_file.cmd
```

Baud — Specify Baud Rate

Syntax baud *baud_rate*

Abbreviation bau

The Baud command specifies a communication baud rate. At initialization, the fastest baud rate for your host system is automatically selected. Table 3-7 lists the valid baud rates.

Table 3-7 Valid Baud Rates

Platform	Baud Rate			
	9600	19200	38400	115200
IBM PC	X	X	X	X
NEC PC	X			
Apollo	X	X	X	
DEC3100	X	X	X	
Sun	X	X	X	
DEC Alpha	X	X	X	
RS 6000	X	X	X	
HP 700	X	X	X	

X indicates don't care

Clock — Specify Clock Source

Syntax [–int | –ext] [–speed *1/3/5/11*] [–stop | –resume] *nclocks*

Abbreviation cl

The Clock command specifies the source of the system clock for the target FPGA. The system clock connects to the CLKO pin. Use this clock to single-step the target FPGA during hardware debug.

The following options are available for the Clock command:

–int Internal Clock

The internal XChecker clock that is used during single-stepping.

–ext External Clock

The external system clock that is used during single-stepping. Use this option if you need a clock speed other than the ones provided with XChecker. Clock frequencies must be in a range from 120 kHz to 10 MHz and applied at the CLKI pin.

–speed x Clock Speed

The Speed option specifies the speed of the clock inside XChecker by

selecting the –int option. You can select this clock to be on the CLKO pin. Specify the desired frequency with one of the variables in the following list.

1	~0.921	MHz
3	~2.75	MHz
5	~5.50	MHz
11	~11.0	MHz

The Speed option specifies the system clock rate during single stepping as well as the readback clock rate. XChecker limits XC4000 and XC5200 readback to 5.5 MHz. If you select variable 11 for an internal burst clock, XChecker bursts the clocks at 11 MHz on the CLKO pin between readback snapshots and slows it to 5.5 MHz during actual snapshots. When performing readback with XC2000 and XC3000 devices, only variable 1 is valid (0.921 MHz).

–stop Stop the Clock

This option stops the clock on the CLKO pin with a logic High. Use this option with the Resume option to single-step the target FPGA system clock.

–resume Reapply the Clock

This option reapplies the clock on the CLKO pin. With this option, the XChecker system becomes a transparent buffer between the CLKI and CLKO pins, with a delay of approximately 100 ns. Use this option with the Stop option to single-step the system clock.

Variables

The following variable is available for the Clock command:

nclocks Specify the Number of System Clocks

The variable *nclocks* specifies the number of clocks the system issues to single-step the target FPGA. Valid values for *nclocks* range from 0 to 32767. You must specify this variable without any options. *Nclocks* is only valid if you used the Clock command with the Stop option.

The following example is the sequence for single-stepping an FPGA design.

`clock -stop`	Stop the system clock provided on the CLKO pin
`clock -speed 3`	Set the system clock (CLKO pin) to 2.75 MHz
`clock 32`	Send 32 clocks from the CLKO pin
`clock 8`	Send eight more clocks from the CLKO pin
clock –resume	Reapply free–running system clock on the CLKO pin

Note: The XChecker system introduces approximately a 100 ns delay between the CLKI and CLKO pins. Also, when single-stepping a target device during debug, XChecker applies the system clock in bursts of up to 127 clocks. To apply more than 127 clocks, the XChecker system pauses between bursts.

Browse — Scan Data Display

Syntax browse

Abbreviation br

The Browse command only scans displayed data on the PC when you use the Show command. The screen shows 24 lines of data and the prompt "more." Enter **y** to see another 24 lines of data, or **n** to quit the Browse command.

Diagnostics — Perform Cable Check

Syntax diag *num*

Abbreviation dia

The Diagnostics command performs a check of the XChecker cable hardware. To execute this command, place the test fixture on the XChecker cable connecting the fixture to V_{CC} and ground. The Diagnostics command verifies whether data can be transmitted and received at the different baud rates and if the internal FPGA and RAM are operational.

The variable *num* represents the number of times to repeat the diagnostic tests (default is 1).

Note: This command requires about one minute to execute.

Exit — Terminate Session

Syntax exit

Abbreviation exi

The Exit command terminates the current XChecker session, asks you whether to save current program options in the xchecker.pro file, and returns you to the system shell.

Export — Save Readback Data

Syntax export [–v] *save_file*

Abbreviation exp

The Export command saves the current readback data to the file *save_file*. XChecker writes the file in one of two formats; a binary file or a Viewlogic generic (GEN) file. The binary format is useful when you want to display saved readback data in subsequent XChecker sessions. Use the GEN file to display readback data in Viewwave format. Use the Import command to retrieve readback data.

–v Save as Viewlogic Generic File

This option saves readback data to a Viewlogic GEN file, which you can later display in the Viewwave environment.

Group — Define/Name a Signal Group

Syntax group *group_name signal_list*

Abbreviation gr

The Group command defines a group name for a signal list. The variables *group_name* and *signal_list* specify the name of the new group and the list of signals to be grouped. You can use the "*" wildcard. The signal names are displayed in 24-line segments. Enter **y** to scroll forward to the next 24 lines. Enter **n** to exit the Group command.

Examples

The following examples show various uses of the Group command.

```
group address a0 a1 a2 a3
```

This example displays the signals a0, a1, a2, and a3 as a single signal called address, when you use the Pick and Show commands.

```
group data d*
```

This example groups all signals in the *design*.ll file beginning with the character "d" in the data group name.

Help — Online Help

Syntax help *topic*

Abbreviation he

The Help command displays online help for the topic requested in 24-line segments. Enter **y** to scroll forward to the next 24 lines. Enter **n** to exit Help.

Import — Retrieve Data

Syntax import *save_file*

Abbreviation im

The Import command retrieves the binary information that the Export command stores in the *save_file* file. Use this command to display readback data from previous XChecker sessions.

Load — Download Design to LCA

Syntax load [–v] *design*

Abbreviation loa

The Load command downloads the specified design to your FPGA, or FPGA daisy-chain. With the –v option, XChecker reads back the design after download to verify the current configuration data. The variable *design* specifies the file that contains the configuration information. Valid file names are *design*.bit, *design*.rbt, *design*.tek, *design*.exo, or *design*.mcs).

The following option is available for the Load command

–v Verify Download and Readback

The –v option executes a download and readback of the current FPGA design for verification. XChecker reads back the configuration data from the FPGA and compares it to the original bitstream. If you do not specify this option, readback is not executed after configuration.

Note: Do not use this option for daisy-chained FPGAs because XChecker only reads the first device.

Log — Send Screen Display to File

Syntax log –out *filename string*

Abbreviation log

The Log command sends the screen output to the *filename* file. Use this command to capture the output of a Readback or a Show command.

There is one option for the Log command.

–out

The –out option closes any previous log file, opens a new one, and places the string at the beginning of the file.

There is one variable for the Log command:

string

Use the variable *string* to insert your comments into the log file, which normally only captures the screen display.

List — List Matching File Names

Syntax ls [–s] *spec*

Abbreviation ls

The List command lists the directory entries of matching file names. You can use the asterisk "*" as a wildcard.

There is one option for the List command.

–s

The –s option expands the search for matching files in the directory indicated by the XACT environment variable.

Part — Specify Part Type

Syntax part *parttype*

Abbreviation pa

The Part command specifies the FPGA part type in use. Specifying the part type is only necessary if you use a PROM file as input and you want to probe the device. XChecker searches for a *parttype*.ll file in the current directory, then in the directory indicated by the XACT environment variable.

Pick — Specify Signal and Display Format

Syntax pick [–bin | –oct | –hex | –dec] [–add | –delete | –all] *signal_list* [–clear]

Abbreviation pi

The Pick command selects the list of signals and display format that the Show command will display (called the display set). The signals defined by *signal_list* are a sub-set of the signals in the *design*.ll file; or you can define them using the Probe command. You can use this command to modify previous display settings. Using the Pick command without any options lists the signal names that have been selected. The list appears in 24-line segments. You can scroll forward by entering **y**. To exit the Pick command, enter an **n**.

The following options are available for the Pick command:

–bin

The –bin option displays signals in binary format

–oct

The –oct option displays signals in octal format

–hex

The –hex option displays signals in hexadecimal format

–dec

The –dec option displays signals in decimal format

–add| –delete| –all

Use the –add and –delete options to add or delete a signal or list of signals to/from the display set. Use the –all option to add or remove all signals to and from the display set.

–clear

The –clear option is equivalent to –delete –all

There is one variable for the Pick command.

–signal_list

Use this variable to define the sub-set of signals in the *design*.ll file

Examples

The following three examples show various uses of the Pick command.

```
pick –bin –add address data
```

This example adds the signals named *address* and *data* to the display set and displays the signals in binary format.

```
pick –hex bus
```

This example adds the group of signals named *bus* to the display set and displays them in hexadecimal format. (See the Group command.)

```
pick –delete –all
```

This example removes all signals from the display set.

Port — Specify Download/Readback Port

Syntax port *portname*

Abbreviation po

The port command specifies the download/readback port. Table 3-8 lists the valid entries; the ports listed in bold face are the defaults. If the port is defined as Auto, all ports are scanned to search for a cable.

Table 3-8 Valid Ports for the XChecker Cable

Platform	Communication Ports			
IBM PC	com1	com2	lpt1*	lpt2*
NEC	com 1			
Apollo	/dev/sio1	/dev/sio2		
DEC3100	/dev/ttyd0	/dev/ttyd1		
Sun	/dev/ttya	/dev/ttyb		
DEC Alpha	/dev/tty00	/dev/tty01		
RS6000	/dev/tty0	/dev/tty1		
HP700	/dev/tty00	/dev/tty01		

* Use with the parallel download cable only (no Readback).
**ttya and ttyb must be readable and writable to ensure a proper connection.

Probe — Define Signals to be Probed

Syntax probe [–add | –delete] s*ignal_list* [–all] [–clear]

Abbreviation pr

The Probe command adds or removes the signals (including RAM bits) that you want to probe (the probe set). These signals must be valid signal names in the *design*.ll file. If you use the Probe command without any options, then it only displays the signals that have been selected. The signal names display in 24-line segments. To scroll forward, enter **y**. To exit the Probe command, enter **n**.

The following options are available for the Probe command.

–add| –delete

Use these options to add or delete the specified signals. The default is –add.

–all

Use the –all option to specify all the signals contained in the file.

–clear

Clear is equivalent to –delete –all.

There is one variable for the Probe command:

–signal_list

A variable you use to define the sub-set of signals in the *design*.ll file

Examples

The following three examples show various uses of the Probe command.

```
probe –add –all
```

This example adds all the signals in the *design*.ll file to the probe set.

```
probe –add address mux
```

This example adds the signals "address" and "mux" to the probe set.

```
probe –delete add
```

This example removes the signal “add” from the probe set.

Note: You must only use valid node names that are found in the *design*.ll file. These node names may differ from the original schematic names; the software may add a hierarchical prefix. You must specify the complete hierarchical name when using the probe command.

Quit — Terminate Session

Syntax quit

Abbreviation qu

The Quit command terminates the current XChecker session and asks you whether to save current program options in the xchecker.pro file.

Readback – Read Back Data Snapshots

Syntax readback [–a] [–c] #*snapshot*

Abbreviation rea

The Readback command reads back data snapshots from your FPGA to verify either the bitstream or logic states. The default is to read only the contents of flip-flops. To view the snapshots, use the Show command after the Readback command.

There are two options for readback.

–a

The –a (Append) option lets you add additional snapshots. For example, if you executed a Readback command and specified five snapshots and used the –a option to specify five additional snapshots, then executing a Show command would display ten snapshots.

–c

The –c option (Readback the configuration bitstream) allows you to readback the bitstream. The difference between this option and the Verify command is that the –c option only reads back the configuration bitstream; it does not compare this bitstream to the original configuration data.

There is one variable for the Readback command.

Snapshot

The *snapshot* variable indicates the maximum number of snapshots to be read back and collected. You can use this variable with or without the –c option. A snapshot corresponds to one complete set of readback data, and only signals that you defined with the Probe command are included in the snapshot.

Reset — Reset Target LCA/Cable

Syntax reset [–cable]

Abbreviation res

The Reset command resets the internal Logic of the target FPGA or resets the XChecker cable. The default is to reset the target FPGA.

There is one option for the Reset command.

–cable

The –cable option reprograms the XChecker cable's internal FPGA. It re-initializes the cable, including setting the correct baud rate. This option is useful in the event of power glitches that could affect proper cable operation.

With this option, you could remove power from the target system, then restore power, while running XChecker; the Reset command re-initializes the cable to the proper settings.

Note: You must specify the location of the RESET pin for XC4000 and XC5200 parts and connect the XChecker cable to that pin. The default polarity of the RESET pin for XC4000 and XC5200 parts is active-High; the XChecker software expects the RESET signal to be active-Low. See Figure 3-6 for the symbol and Table 3-4 for pin connections.

Save — Save Option Settings

Syntax save

Abbreviation sa

The Save command saves the settings of four interactive command results in the xchecker.pro file; baud rate (Baud command), design name (Load command), device type (Parttype command) and port name (Port command).

At initialization, XChecker reads the xchecker.pro file to set up the defaults for the current session. This file must be in the current directory or in the XACT environment search path. XChecker updates the profile information at the end of every session. The xchecker.pro file is created when you exit from your first XChecker session.

Settings — Display Settings

Syntax settings

Abbreviation se

The Settings command provides a listing of the following information; the port name, the baud rate, the type of cable, the design name, the part type and package type, the clock source, and hardware trigger status. It also lists the number of clocks for the first and subsequent snapshots, the number of signals defined in the probe list, and the number of signals defined in the display list.

Show — Display Readback Mode

Syntax show [–signal] *xy* [–snapshot] *nm* [–tabular | –vdata] text | –hdata]

Abbreviation sh

The Show command displays the readback data in the format you specified in the Pick command. The Show command displays the data 24 lines at a time. To scroll forward, enter **y**. To exit the Show, command, enter **n**.

The following options are available for the Show command.

–signal

The –signal option selects the number of signals to be displayed, beginning with the *x* signal and ending with the *y* signal. This option is useful for selecting a set of signals to display without changing the display set.

–snapshot

The –snapshot option allows you to define the number of the first snapshot to show, specified by the *n* variable, and the total number of snapshots to show, specified by the *m* variable.

Note: One snapshot corresponds to one complete set of readback data from the target FPGA.

–vdata

The –vdata option displays vertically, from top to bottom, the signal data. This is the default mode.

–hdata

The –hdata option displays horizontally, from left to right, the signal data.

There are four variables for the Show command:

x

The *x* variable works with the –signal option and defines the first signal name in the display set.

y

The *y* variable works with the –signal option and defines the last signal name in the display set.

n

The *n* variable works with the –snapshot option and defines the first snapshot to display with the Show command.

m

The *m* variable works with the –snapshot option and defines the total number of snapshots to display with the Show command.

Examples

The following example shows how to use the Display and Show commands.

```
display –bin BUS DATA S0 S1 S2 S3
show –vdata –snap 51 4
```

This example results in the BUS and DATA signal groups and the S0, S1, S2, and S3 signals to display in binary tabular (vertical) format,

starting with snapshot 51 and showing a total of four snapshots ending with snapshot 54, as shown in Table 3-9.

Table 3-9 Snapshot Examples

Snapshot	BUS	DATA	S0	S1	S2	S3
51	0000001	01001	0	1	0	1
52	1000101	10101	1	1	1	0
53	1010110	11101	0	0	0	1
54	1101011	11111	0	0	0	1

Status — Show Logic Levels of XChecker Pins

Syntax status

Abbreviation st

The Status command displays the logic level of the RST, INIT, D/$\overline{P}$, $\overline{PROG}$, CCLK, TRIG, RT, and RD XChecker pins. The TRIG pin status is registered, so the Status command display shows TRIG High if at any time a trigger has been acknowledged.

Sys —Temporarily Exit to Operating System

Syntax sys

Abbreviation none

The Sys command allows you to temporarily exit from XChecker to the operating system prompt. Enter **`exit`** to return to XChecker.

Trigger — Select Trigger for Readback

Syntax trigger [–auto –manual –none] –clock] *xy* [–timeout] [–reset]

Abbreviation tr

The Trigger command selects the triggering event for the next readback. You must execute a Trigger command before running a Readback command. When a Readback command is executed, XChecker waits for a trigger to initiate a readback. Once a trigger is acknowledged by the XChecker cable, the software issues the number

of clocks that you specified with –clock option and initiates a readback operation.

Note: With XC4000 and XC5200 devices, you can directly connect the system trigger to the target FPGA TRIG input of the readback symbol pin which latches the state of the device instantly instead of waiting for the XChecker software to initiate a readback.

There are three triggers for the Trigger command:

–auto

The –auto trigger is a Low-to-High transition on the XChecker TRIG pin.

–manual Manual Trigger

The –manual trigger defines the trigger as pressing the ↵ key on the host system keyboard.

–none

The –none trigger initiates a readback immediately after you issue a Readback command, without expecting a trigger, or stopping the system clock.

There are three options for the Trigger command:

–clock

The –clock option causes a pause between the readbacks of data. It indicates that, after acknowledging a trigger, the XChecker software issues *x* clocks before the first readback is initiated and *y* clocks between each snapshot. This option is functional only if the CLKO pin is connected to the target FPGA.

–timeout

The –timeout option indicates that readback is cancelled if a trigger is not received within *t* seconds (*t* must be an integer between 0 and 32767).

–reset

The –reset option resets the target FPGA, before starting a readback, by sending a pulse on the RESET line.

Examples

The following examples explain various uses of the Trigger command:

```
trigger –auto –clock 2 5
readback 10
```

In this example, XChecker waits for a Low-to-High transition at the TRIG pin, stops the system clock (puts CLKO High), and then issues two system clocks on the CLKO pin. Next, XChecker initiates a readback by sending a Low-to-High pulse on the RT pin. After the first snapshot is read back, five system clocks are issued, and the whole process repeats nine more times. XChecker reapplies the free–running system clock after readback finishes.

```
trigger –manual
readback 1
```

In this example, XChecker stops the system clock and initiates a readback immediately after you press the ↵ key, collecting one snapshot. XChecker reapplies the system clock is reapplied the readback finishes.

```
trigger –none
readback 2
```

In this example, XChecker initiates two consecutive readbacks immediately after you issue the Readback command.

Verify — Verify Target FPGA Bitstream

Syntax verify *filename*.bit

Abbreviation ve

The Verify command compares the current bitstream in the target FPGA to the bitstream in *filename*.bit to verify correctness. This command differs from the Load command with the –v option in that the configuration bitstream is not downloaded; it is only read back and compared.

Troubleshooting Guide

This section is a simple guide to understanding the more common issues you might encounter when configuring FPGAs with the XChecker cable. These issues are likely to fall into three groups; communication, improper connections, and improper or unstable V_{CC}.

- Communication

  This section describes several issues that involve the integrity of the bitstream and the configuration clock that XChecker transmits to the target FPGAs.

- Improper Connections

  This section involves assigning configuration pins to invalid signals or voltage levels.

- Improper or Unstable V_{CC}

  This section describes several causes of incorrect configuration sequences and incorrect handshaking signals from the FPGA.

Communication

Observing the following guidelines should minimize the communication difficulties that can occur between XChecker and the target FPGA.

Do not attach extension cables to the target FPGA side of the XChecker cable; this can compromise configuration data integrity and cause checksum errors.

Make sure that any noise or ringing in the configuration clock (CCLK) is not above 20% of V_{CC} for a logic zero or below 70% of V_{CC} for a logic one.

Attach the XChecker cable configuration leads firmly to the target FPGA.

Use the verify feature to assure integrity of the configuration data. You can do this from the command line with the –v option or in the interactive mode by specifying the verify command.

On XC3000A devices, INIT can go Low.

With XC4000 and XC5200 devices, use the Status command to monitor the INIT pin. INIT can go Low on XC3000A, XC4000, and XC5200 devices. If CRC is enabled, a logic zero at the INIT pin signals the occurrence of a corrupted frame of data.INIT can go Low on XC4000 and XC5200 devices even when CRC is not enabled. You can only enable CRC for XC4000 and XC5200 devices while running the MakeBits program.

Improper Connections

Always make sure that XChecker leads are connected properly for the mode of operation desired. (Refer to Table 3-4 and Table 3-5.)

Note: Connecting the XChecker leads to the wrong signal will cause permanent damage to XChecker internal hardware. You must connect V_{CC} to +5 V and gnd to ground.

Use the Status command to display the logic state of the leads. This helps determine connectivity between the target system and the XChecker cable. For workstations, you must have read and write permissions to the port to which you connect the XChecker cable. XChecker might issue a message stating that the cable is not connected to port tty*x*. When you see this message, follow the check list below:

- The board must have the power on, since XChecker uses power from the board.
- Check the device driver using the following command string:

```
ls –l /dev/ttya /dev/ttyb
```

The result should be the following:

```
crw-rw-rw-  1 root12,0 month date time /dev/ttya
crw-rw-rw-  1 root12,1 month date time /dev/ttyb
```

- Reconnect the XChecker cable to another valid port.
- Read the /etc/ttyab file. There should be two lines, as follows:

```
ttya   ‘‘/usr/etc/getty std.9600’’  unknown   off
local secure
ttyb   ‘‘/usr/etc/getty std.9600’’  unknown   off
local secure
```

If you use a port to connect a modem or a remote login, you cannot use that port. The port must be on. Consult your System Administrator if the information the /etc/ttyab file if different than what is listed in the aforementioned list.

Improper or Unstable V_{CC}

Never connect the control signals to XChecker before V_{CC} and ground. Xilinx recommends the following sequence:

1. Turn off power to the target system.
2. Connect V_{CC}, ground, and then the signal leads,
3. Turn on power to the target system.

Warning: As with any CMOS device, the input/output pins of the internal FPGA should always be at a lower or equal potential than the rail voltage to avoid internal damage.

Make sure V_{CC} rises to a stable level within 10ms. Stable V_{CC} should be between 4.75 V and 5.25 V.

In the event of power glitches, use the interactive Reset command with the –c option (Cable option) to reconfigure the XChecker internal FPGA and use the interactive Load command to reconfigure the target FPGA.

Warning Messages

This section describes the warning messages that XChecker may generate. There is a suggested workaround that follows each message.

```
Warning  001  Current design does not have
ReadCapture option set. Snapshot readback is
incorrect.
```

To verify the design or probe the internal logic, set the following configuration options in MakeBits (from XDE).

For the XC4000 and XC5200 parts:

SyncToDone:	**No**
ReadCapture:	**Enable**
ReadClk:	**CCLK**

DoneActive: **C1 C2 C3 C4**
OutputsActive: **C2 C3 C4**

For the XC3000 and XC2000 parts:

Read: **CMD**

Refer to the "MakeBits" chapter in the *Development System Reference Guide* for details.

```
Warning  002  Current design does not have
Readback clock set to CCLK. Snapshot readback is
incorrect.
```

To verify the design or probe the internal logic, set the following configuration options in MakeBits (from XDE).

For the XC4000 and XC5200 parts:

SyncToDone: **No**
ReadCapture: **Enable**
ReadClk: **CCLK**
DoneActive: **C1 C2 C3 C4**
OutputsActive: **C2 C3 C4**

For the XC3000 and XC2000 parts:

Read: **CMD**

Refer to the "MakeBits" chapter in the *Development System Reference Guide* for more details.

```
Warning  003  Current design does not have DONE
set to be active before or at IOB enable. Device
may not be configured correctly.
```

For XC4000 and XC5200 parts, set the following configuration options in MakeBits (from XDE).

For the XC4000 and XC5200 parts:

DoneActive: **C1 C2 C3 C4**

Refer to the "MakeBits" chapter in the *Development System Reference Guide* for details.

For the XC3000 and XC2000 parts:

Read: **CMD**

Refer to the "MakeBits" chapter in the *Development System Reference Guide* for more details.

```
Warning  004  Current design may not have one of
the following options set up correctly. Please
verify them with XDE.
```

For the XC4000 and XC5200 parts:

SyncToDone:	**No**
ReadCapture:	**Enable**
ReadClk:	**CCLK**
DoneActive:	**C1 C2 C3 C4**
OutputsActive:	**C2 C3 C4**

To verify the design or probe the internal logic, set the following configuration options in MakeBits (from XDE).

For the XC3000 and XC2000 parts:

Read:	**CMD**

Refer to the "MakeBits" chapter in the *Development System Reference Guide* for more details.

```
Warning  005  LL file filename.ll is not found.
Design verification will not work correctly.
```

The *filename*.ll file must be in the current directory or in the directory XACT\DATA. (For workstations, the XACT directory is referred to as the “Xilinx_dir” in the installation notes.) Check the read permissions for your directory and for the *filename*.ll file. Make sure that your XACT environment variable points to the XACT directory.

Error Messages and Recovery Techniques

This section describes the error messages that XChecker may generate. Following each error message, there is a suggested workaround.

```
Error   001   Command file bat file.cmd is not found.
```

Make sure that the command file you specified is in the current directory or the environment search path. Make sure that the command file has the ".cmd" extension

```
Error   002   Internal Error – Command table
```

```
syntax error Cmd=valid_command.
```

This is an internal program error that normally should not occur. Try entering the command sequence again. If the error persists, try reinstalling your XChecker software. If the error reappears, call Xilinx Technical Support. Be prepared to duplicate the error and reference specific files or examples.

```
Error   003   Diagnostic is not supported on this
cable.
```

The Diagnostics command, as well as other readback and verification commands are only supported for the XChecker cable.

```
Error   004   Not enough memory.
```

XChecker requires a minimum of 512 kB free base RAM. If you are using a PC, make sure no other program or TSR (Terminate and Stay Resident application, such as mouse drivers, network drivers, or window drivers etc.) is running simultaneously with XChecker. To free memory, type EXIT at the DOS command line.

Error 010 Cannot open output file *filename*.

Check available disk space. Current directory or file must have write permission.

Error 011 Cannot create output file *filename*.

Check available disk space. Current directory or file must have write permission.

Error 012 Cannot open input file *filename*.

Make sure the *filename* file exists in your working directory or in the environment search path. Current directory or file must have write permission.

Error 013 File *filename* is not found.

The *filename* file does not exist in the current directory or search path. Make sure that the *filename* file exists in your working directory or in the environment search path.

Error 021 Help file *XChecker.hlp* is not accessible

Make sure that the XACT environment variable points to the XACT directory in the PC (the XACT directory in the PC is equivalent to the

"Installation Directory" on the workstations). Also make sure that xchecker.hlp is in the directory XACT\MSG. If you cannot find xchecker.hlp in the XACT\MSG directory, you must reinstall the XChecker software.

`Error 022 No help for command` *command entered.*

Help is not available for the specified command. Refer to the Interactive Mode Commands section in this chapter for help.

`Error 023 Cannot save configuration to` *filename.pro.*

Check available disk space. Current directory or file must have write permission.

`Error 024 Invalid command at line` *line number.*

Check the file xchecker.pro in your current directory for illegal commands. Delete the xchecker.pro file. XChecker creates a new profile when you exit from the session.

`Error 030 Ambiguous command.`

Enter the minimum unique characters that identify the command or enter complete commands with no abbreviations.

`Error 031 Invalid command.`

The command you entered is illegal. Refer to the Interactive Mode Commands section in this chapter for help.

`Error 032 Invalid number of arguments.`

Refer to the Interactive Mode Commands section in this chapter for help.

`Error 033 Invalid option` *selected option.*

Refer to the Command-Line Options and the Interactive Mode Commands sections in this chapter for help.

`Error 034 Invalid value given to` *parameter.*

Refer to the Command-Line Options and the Interactive Mode Commands sections in this chapter for help.

`Error 035 Value is required for` *command entered.*

Refer to the Interactive Mode Commands section in this chapter for help.

```
Error   050   System Error Messages
```

System error codes are usually a string of messages generated by your operating system.

```
Error   051   System file error code.
```

System error codes are usually a string of messages generated by your operating system.

```
Error   101   Cable is not initialized.
```

Reissue the Reset command with the –c option, or cycle power to XChecker and then issue the Reset command with the –c option. See the Improper or Unstable V_{CC} section in this chapter.

```
Error   102   Cable is not located.
```

No cable has been recognized at any port. Make sure there is power to your board and to XChecker. If you are using the test fixture, you must connect V_{CC} and ground to it. XChecker draws power from your target system, not from your host computer. Also make sure the RS-232 connector is firmly attached.

For the PC, remember that your serial port needs to be set to the following IRQ lines and I/O addresses:

COM1 IRQ4 at address 03f8

COM2 IRQ3 at address 02f8

```
Error   103   Invalid port name.
```

Refer to the XChecker Hardware section in this chapter for help.

```
Error   104   Invalid baud specified.
```

Refer to the XChecker Hardware section in this chapter for help.

```
Error   105   Cable is not reset.
```

Cycle power to the cable. Use the Reset command with the –c option.

```
Error   107   Communication line is broken.
```

Run the Reset command with the –c option. Also make sure there is power to your board and to the XChecker cable. Check all power and port connections.

```
Error   108   Communication checksum error.
```

Check for induced noise in your target system or from your target system into the XChecker connections. Do not use cable extensions. The XChecker cable length is tested to produce minimal noise levels. Remember that a logic High must be 80-100% of V_{CC} and a Logic Low must be 0-25% of V_{CC}.

```
Error   109   Cable has no power.
```

Make sure there is power from your target system to XChecker. XChecker draws power from an external source, not from the host computer.

```
Error   110   Communication time-out.
```

XChecker has not received an expected signal; for example, a system trigger to initiate readback or data coming from readback. Make sure that the selected options for trigger and readback are what you intended. Check all connections. For XC4000 and XC5200 devices, make sure that you used the Readback symbol in your schematics and check the location of RTRIG and RDATA. For XC3000 and XC2000 devices, check the location of the signals RTRIG and RDATA in your pinouts. Consult your APR/PPR report files for signal locations.

```
Error   111   DONE did not go high as expected.
```

The design has not been configured properly. Make sure that you compiled your design for the correct part type and package. Make sure that the target FPGA has been set up for serial slave mode. Check all connections. Remember that connections to the FPGA differ slightly across the device families. Check the bitstream options in MakeBits and ensure that you have selected a pull-up for the DONE pin.

```
Error   112   Cannot reset DONE signal.
```

XChecker cannot set the DONE signal to Low. Make sure DONE is pulled up, not tied High. Check the corresponding package pinouts.

```
Error   113   DONE went high earlier than
expected.
```

Make sure that you compiled your design for the correct part type and package. Check for noise in the CCLK line. Refer to the Troubleshooting Guide section.

```
Error   114   DONE went high later than expected.
```

Make sure that you compiled your design for the correct part type and package. Check for noise in the CCLK line. Refer to the Troubleshooting Guide section.

```
Error   120   Cannot communicate to the cable.
```

Run the Reset command with the –c option. Also, ensure that there is power to your board and to the XChecker cable. Check all connections. Make sure the RS-232 connector is firmly attached.

Error 121 Cable datafile *filename* is empty.

Run the Reset command with the –c option. Make sure that the XACT environment variable points to the XACT directory on the PC. (On workstations, the XACT directory is equivalent to the "Xilinx_Directory" referenced in the installation notes).

Error 122 Cable datafile *filename* has invalid format.

On PCs, make sure that the XACT environment variable points to the XACT directory. On the workstations, the XACT directory is equivalent to the "Xilinx_Directory" referenced in the installation notes.

Error 122 Can't open cable datafile *filename*.

On PCs, make sure that the XACT environment variable points to the XACT directory. On the workstations the XACT directory is equivalent to the "Xilinx_Directory" referenced in the installation notes.

Error 123 No XChecker cable is connected to the port *portname*.

Ensure that there is power to your board and to XChecker. You must connect V_{CC} and ground to the test fixture, if you are using it. XChecker draws power from your target system, not from the host computer. Ensure that the RS-232 connector is firmly attached.

On PCs, your serial port (for use with XChecker and serial cables) must be set to the following IRQ lines and I/O addresses:

COM1	IRQ4 at address 03f8
COM2	IRQ3 at address 02f8

```
Error   124   No XChecker cable is connected to
the system.
```

Ensure that there is power to your board and to XChecker. You must connect V_{CC} and ground to the test fixture, if you are using it. XChecker draws power from your target system, not from the host computer. Ensure that the RS-232 connector is firmly attached.

On PCs, your serial port (for use with XChecker and serial cables) must be set to the following IRQ lines and I/O addresses:

COM1	IRQ4 at address 03f8
COM2	IRQ3 at address 02f8

```
Error   125   Fail reading cable status.
```

Try using the Reset command with the cable option. Ensure that there is power to your board and to XChecker. Check all connections.

```
Error   126   Unsupported command for this
cable.
```

See the Interactive Mode Commands section for valid with the XChecker cable. If you are using the previous parallel or serial download cables, you can only use the Load command to download.

```
Error   128   Read only number of bits received.
```

Check all connections. Check for noise that may be induced into your target system or from your target system into the XChecker connections. Do not use cable extensions. The XChecker cable length is tested to produce minimal noise levels. Remember that a logic High must be 80-100% of V_{CC} and a Logic Low must be 0-25% of V_{CC}.

```
Error   130   Invalid baud rate. Current baud
rate is baud rate.
```

See Table 3-1 for the valid baud rates for your computer.

```
Error   131   Missing baud rate. Current baud
rate is baud rate.
```

See the Interactive Mode Commands section in this chapter for correct command usage.

```
Error   132   Block number: Communication Error.
```

Check for noisy connections. Check power stability. You may want to use the Reset command with the cable option to re-initialize the cable, then try to read back the data.

`Error 133 Block %d: Communication Timeout.`

XChecker has not received an expected block of data. Check all connections. Check for power stability and for contention with the control signals that can cause a readback abort.

`Error 134 Cannot communicate with port` *port name.*

Check this manual for supported ports. See the Port command. When using the XChecker cable with a PC the serial port needs to be set to the following IRQ lines and I/O addresses:

COM1 IRQ4 at address 03F8

COM2 IRQ3 at address 028

`Error 135 Invalid port name` *port name.*

Refer to Table 3-6 for supported ports. See the Port command.

`Error 140 Datafile` *filename* `has invalid format.`

XChecker only supports the following formats: RBT and BIT (Xilinx formats), MCS (Intel format), TEK (Tektronix format), and EXO (Motorola format).

`Error 141 Datafile` *filename* `is empty.`

The specified datafile is either empty or contains invalid data. XChecker supports the following formats: RBT and BIT (Xilinx formats), MCS (Intel format), TEK (Tektronix format), and EXO (Motorola format).

`Error 142 Datafile` *filename* `is not found.`

The *filename* file does not exits in the current directory or search path. Check your environment search path to make sure that it contains the directory where *filename* is.

`Error 143 Can't open datafile` *filename.*

The file *filename* does not exits in the current directory or search path. Check your environment search path to make sure that it contains the directory where *filename* is located.

`Error 150 Only` *number* `bytes read.`

Check all connections. Do not use cable extensions. The XChecker cable length is tested to produce minimal noise levels. Remember that a logic High must be 80-100% of V_{CC} and a Logic Low must be 0-25% of V_{CC}.

`Error 180 Cannot create export file` *filename*.

Check available disk space. Current directory or file must have write permission.

`Error 181 Cannot open export file` *filename*.

Check available disk space and file accessibility such as, write privileges to the current directory.

`Error 182 Missing design name to export.`

You must specify a name to create an export file. Check for correct command usage.

`Error 183 Missing design name to import.`

You must specify a name to open an import file. Check for correct command usage.

`Error 190 Missing .ll filename.`

You must specify a file name for the .ll file.

`Error 191 .ll file` *filename.ll* `is not found.`

The *filename.ll* file must be in the current directory or in the xact\data directory on PCs. On workstations the XACT directory is referred to as the "Xilinx_dir" in the installation notes. Check read permissions to your directory and to the *filename*.ll file. Check your XACT environment variable to make sure that it points to the XACT directory.

`Error 192 Cannot open .ll file` *filename.ll*.

The *filename.ll* file must be in the current directory or in the xact\data directory on PCs. On workstations the XACT directory is referred to as the "Xilinx_dir" in the installation notes. Check read permissions to your directory and to the *filename*.ll file. Check your XACT environment variable to make sure that it points to the XACT directory.

```
Error   202   Invalid number of clocks.
```

The valid number of clocks range from 1 to 32767. Refer to the Clock command description for the correct usage.

```
Error   210   Not enough memory for loading
design datafile.
```

XChecker requires a minimum of 512 kB free base RAM. If you are using a PC, make sure that no other programs or TSR (Terminate and Stay Resident application such as, mouse drivers, network drivers, window drivers) are running simultaneous with XChecker. To free memory, at the DOS command line, type **exit**.

```
Error   211   Not enough memory for bitmap
result.
```

XChecker requires a minimum of 512 kB free base RAM. If you are using a PC, make sure that no other programs or TSR (Terminate and Stay Resident application such as, mouse drivers, network drivers, window drivers) are running simultaneous with XChecker. To free memory, at the DOS command line, type **exit**.

```
Error   220   No part type is defined.
```

The part type specified in your design or by you (using the Part command) is invalid. Check the *The Programmable Logic Data Book* for valid part types and packages.

Error 240 Data is corrupted. Bad checksum in file *PROM file*, line *linenumber*.

The input (hexadecimal) file is corrupted. Try regenerating the input file using MakePROM. Remember that supported hexadecimal formats are: MCS (Intel format), EXO (Motorola format), and TEK (Tektronix format).

Error 241 Invalid data character in file *PROM file*, line *linenumber*.

The input (hexadecimal) file is corrupted. Try regenerating the input file using MakePROM. Remember that supported hexadecimal formats are: MCS (Intel format), EXO (Motorola format), and TEK (Tektronix format).

Error 242 Invalid datafile format in file *PROM file*, line *linenumber*.

XChecker supports the following formats: BIT (Xilinx format), MCS (Intel format), EXO (Motorola format), and TEK (Tektronix format).

`Error 243 Internal Error` *errorname* `in file` *filename*`, line` *linenumber*.

An illegal character has been detected in *filename*. This could be caused by an illegally formatted PROM file (supported PROM formats are TEK, EXO, and MCS) or simply a corrupted data or command file.

`Error 250 Readback configuration contains all` *0/1*.

Check all connections. Particularly, check the connections to the XChecker RT and RD pins. Refer to Table 3-4 for pin functions and proper connections. Check MakeBits options: for XC3000 and XC2000 devices make sure that you have selected readback upon command from MakeBits (Read option must be Cmd); for XC4000 and XC5200 devices make sure that ReadCapture is Enabled. Check the power to the target FPGA.

`Error 251 No data in readback buffer.`

You must execute a Readback command before you can show any data.

`Error 260 Signal name` *name* `is not defined.`

Execute a Probe command (without arguments) to display the available signal names. You can also check the *design*.ll file for this information. Include the complete net names, with their hierarchical prefixes. Remember that valid signal names are: user-defined net names connected to flip-flops, latches, and memory outputs; user-defined group names and IOB names.

`Error 262 Signal name` *name* `is not user defined.`

Execute a Probe command (without arguments) to display the available signal names. You can also check the *design*.ll file for this information. Include the complete net names, with their hierarchical prefixes. Remember that valid signal names are: user-defined net names connected to flip-flops, latches, and memory outputs, user-defined group names, and IOB names.

```
Error   263   DONE signal went High early.
```

Make sure that you compiled your design for the correct part type and package. Check for noise in the CCLK line. Refer to the Trouble Shooting Guide section for more information.

```
Error   264   DONE signal did not go High.
```

The design has not been configured properly. Make sure that you compiled your design for the correct part type and package. Make sure that you set the target FPGA for the serial slave mode. Check all connections. Remember that connections to the FPGA differ slightly across the different families. Check the bitstream options in MakeBits and verify that you selected a pull-up for the DONE pin.

```
Error   265   INIT signal is low.
```

The design has not been configured properly. Make sure that you compiled your design for the correct part type and package.

Make sure that you set the target FPGA for the serial slave mode. Check all connections. Remember that connections to the FPGA differ slightly across the different families. Check for noisy signals that may corrupt the data or CCLK going to the target FPGA.

```
Error   266   Probe list is empty.
```

If you have executed a Probe command with the Clear option, you must also execute a Probe command to add signals to the probe list. Check the *design*.ll file to make sure that it contains your design's net names.

```
Error   267   Display list is empty.
```

You must execute a Pick command before you can show the readback data. Refer to the Pick and Show command descriptions in this chapter.

```
Error   268   Group list is empty.
```

Refer to the Interactive Mode Commands section in this chapter for proper use of the Group command.

Index

Z

Trademark Information

XILINX®, XACT, XC2064, XC3090, XC4005, and XC-DS501 are registered trademarks of Xilinx. All XC-prefix product designations, XACT-Floorplanner, XACT-Performance, XAPP, XAM, X-BLOX, X-BLOX plus, XChecker, XDM, XDS, XEPLD, XPP, XSI, BITA, Configurable Logic Cell, CLC, Dual Block, FastCLK, HardWire, LCA, Logic Cell, LogicProfessor, MicroVia, PLUSASM, SMARTswitch, UIM, VectorMaze, VersaBlock, VersaRing, and ZERO+ are trademarks of Xilinx. The Programmable Logic Company and The Programmable Gate Array Company are service marks of Xilinx.

IBM is a registered trademark and PC/AT, PC/XT, PS/2 and Micro Channel are trademarks of International Business Machines Corporation. DASH, Data I/O and FutureNet are registered trademarks and ABEL, ABEL-HDL and ABEL-PLA are trademarks of Data I/O Corporation. SimuCad and Silos are registered trademarks and P-Silos and P/C-Silos are trademarks of SimuCad Corporation. Microsoft is a registered trademark and MS-DOS is a trademark of Microsoft Corporation. Centronics is a registered trademark of Centronics Data Computer Corporation. PAL and PALASM are registered trademarks of Advanced Micro Devices, Inc. UNIX is a trademark of AT&T Technologies, Inc. CUPL, PROLINK, and MAKEPRG are trademarks of Logical Devices, Inc. Apollo and AEGIS are registered trademarks of Hewlett-Packard Corporation. Mentor and IDEA are registered trademarks and NETED, Design Architect, QuickSim, QuickSim II, and EXPAND are trademarks of Mentor Graphics, Inc. Sun is a registered trademark of Sun Microsystems, Inc. SCHEMA II+ and SCHEMA III are trademarks of Omation Corporation. OrCAD is a registered trademark of OrCAD Systems Corporation. Viewlogic, Viewsim, and Viewdraw are registered trademarks of Viewlogic Systems, Inc. CASE Technology is a trademark of CASE Technology, a division of the Teradyne Electronic Design Automation Group. DECstation is a trademark of Digital Equipment Corporation. Synopsys is a registered trademark of Synopsys, Inc. Verilog is a registered trademark of Cadence Design Systems, Inc.

Xilinx does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx devices and products are protected under one or more of the following U.S. Patents: 4,642,487; 4,695,740; 4,706,216; 4,713,557; 4,746,822; 4,750,155; 4,758,985; 4,820,937; 4,821,233; 4,835,418; 4,853,626;

4,855,619; 4,855,669; 4,902,910; 4,940,909; 4,967,107; 5,012,135; 5,023,606; 5,028,821; 5,047,710; 5,068,603; 5,140,193; 5,148,390; 5,155,432; 5,166,858; 5,224,056; 5,243,238; 5,245,277; 5,267,187; 5,291,079; 5,295,090; 5,302,866; 5,319,252; 5,319,254; 5,321,704; 5,329,174; 5,329,181; 5,331,220; 5,331,226; 5,332,929; 5,337,255; 5,343,406; 5,349,248; 5,349,249; 5,349,250; 5,349,691; 5,357,153; 5,360,747; 5,361,229; 5,362,999; 5,365,125; 5,367,207; 5,386,154; 5,394,104; 5,399,924; 5,399,925; 5,410,189; 5,410,194; 5,414,377; RE 34,363, RE 34,444, and RE 34,808. Other U.S. and foreign patents pending. Xilinx, Inc. does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. Xilinx assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Xilinx products are not intended for use in life support appliances, devices, or systems. Use of a Xilinx product in such applications without the written consent of the appropriate Xilinx officer is prohibited.