XAPP938 (v1.0) March 28, 2007

Dynamic Bus Mode Reconfiguration of PCI-X and PCI Designs

Authors: John Ayer and Jameel Hussein

Summary

The Xilinx LogiCORE™ solution for dynamic bus mode reconfiguration of PCI® and PCI-X designs requires a different bitstream for each mode when targeting Virtex™-4 or Virtex-5 devices. To be fully compliant, the FPGA must recognize the bus type and select the appropriate bitstream for the core to operate in the correct mode. This application note discusses how to dynamically reload the FPGA after power-up, using a CPLD when the initial bitstream is not compatible with the bus mode. The reference design is fully verified and tested using the Virtex-4 Development Kit for PCI and PCI-X designs (ML455). FPGA and CPLD design files for this board are included in the reference design file.

Overview

Xilinx offers four cores for PCI and PCI-X designs. The appropriate core choice is based on the target device and design requirements. Each core has a primary version number (shown in Table 1) followed by a revision or build number.

Table 1: **Available LogiCORE Products for PCI-X and PCI Solutions**

Core	Primary Version	Device(s) Supported	Bus Mode Reconfiguration Support
PCI	v3	Virtex-4, Virtex-II Pro, Virtex-E, Virtex, Spartan-3E, Spartan-3, Spartan-IIE, Spartan-II	No
PCI	v4	Virtex-5	Yes
PCI-X	v5	Virtex-4, Virtex-II Pro, Virtex-E	Yes
PCI-X	v6	Virtex-5	Yes

More information about the current versions of these cores is available in the LogiCORE data sheets for PCI and PCI-X section of the PCI/PCI-X lounge.

The LogiCORE products support the dynamic reconfiguration option required for PCI-X compliance. The dynamic reconfiguration technique can be used with any PCI-X design for any of the supported devices shown in Table 1. However, this application note focuses on Virtex-4 and Virtex-5 designs. The Virtex-4 core (v5) and Virtex-5 core (v6) have slight variations in the user application ports. This application note describes interfacing to each of these cores to accomplish full PCI-X compliance. The Virtex-5 core (v4) also supports the dynamic reconfiguration option. Designers using Virtex-5 devices have the option of using the standard core (v4) to achieve 66 MHz PCI operation and the version 6 core for PCI-X operation. Both version 5 and version 6 of the PCI-X solution only support 33 MHz PCI operation.

PCI-X Device Initialization and Add-In Card Requirements

PCI-X device and add-in cards must be able to recognize the PCI or PCI-X bus mode and width. To be fully compliant, the device must adjust its operation to match the bus mode and width dynamically. Designers are encouraged to review Chapter 6 of the *PCI-X Protocol Addendum to the PCI Local Bus Specification v2.0a* for detailed information on these requirements.

The source bridge resource indicates to the PCI/PCI-X device the bus width through REQ64# at the rising edge of reset. If the bridge asserts REQ64# at the rising edge of RST#, it indicates a 64-bit bus. If the host does not assert REQ64# at the rising edge of RST#, the bus is 32 bits wide. Figure 1 shows the host indicating to all devices on the bus segment that the bus is 64 bits wide.

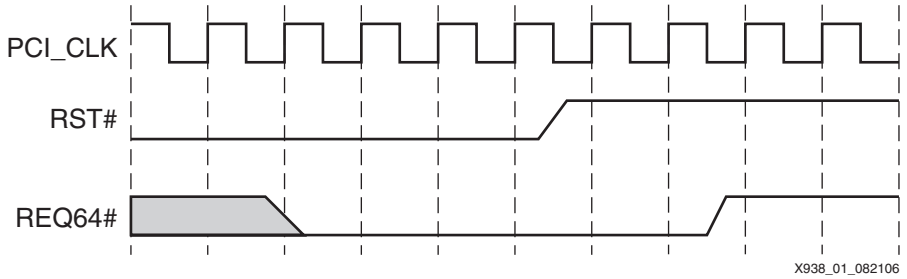

Figure 1: **PCI Bus Width Indication**

From the time that power is valid to the rising edge of RST#, the PCI specification guarantees a minimum of 100 ms. This parameter is called T_{PVRH}. The FPGA must configure within this time to see the bus width and bus mode initialization pattern. For detailed information on all bus timing parameters, see Chapter 4 of the *PCI Local Bus Specification v3.0*.

Along with bus-width initialization, the bridge resource must inform all connected devices of the bus mode. In general, the choice is either PCI or PCI-X bus mode, but this can be further segmented depending on the clock frequency. The bridge resource indicates the bus mode by driving a certain pattern on PERR#, DEVSEL#, STOP#, and TRDY# during the deassertion of RST#. This is similar to the method used to indicate the bus width using REQ64#. Table 2 shows a subset of the initialization pattern for PCI and PCI-X devices.

Table 2: **Initialization Pattern for PCI and PCI-X Devices**

PERR#	DEVSEL#	STOP#	TRDY#	Mode	Min Clock Frequency (MHz)	Max Clock Frequency (MHz)
1	1	1	1	Conventional 33 MHz	0	33
1	1	1	1	Conventional 66 MHz	33	66
1	1	1	0	PCI-X Mode 1	50	66
1	1	0	1	PCI-X Mode 1	66	100
1	1	0	0	PCI-X Mode 1	100	133

The solution for the PCI-X standard is designed to operate up to 133 MHz and does not differentiate between the different mode patterns. The following is an example of the logic:

```
is_pcix_mode = PERR# and DEVSEL# and (not STOP# or not TRDY#)
```

By using timing analysis, designers ensure that the user application and core meet the maximum system timing requirements.

Figure 2 is an example of a PCI-X initialization pattern along with REQ64#, indicating a 64-bit bus.

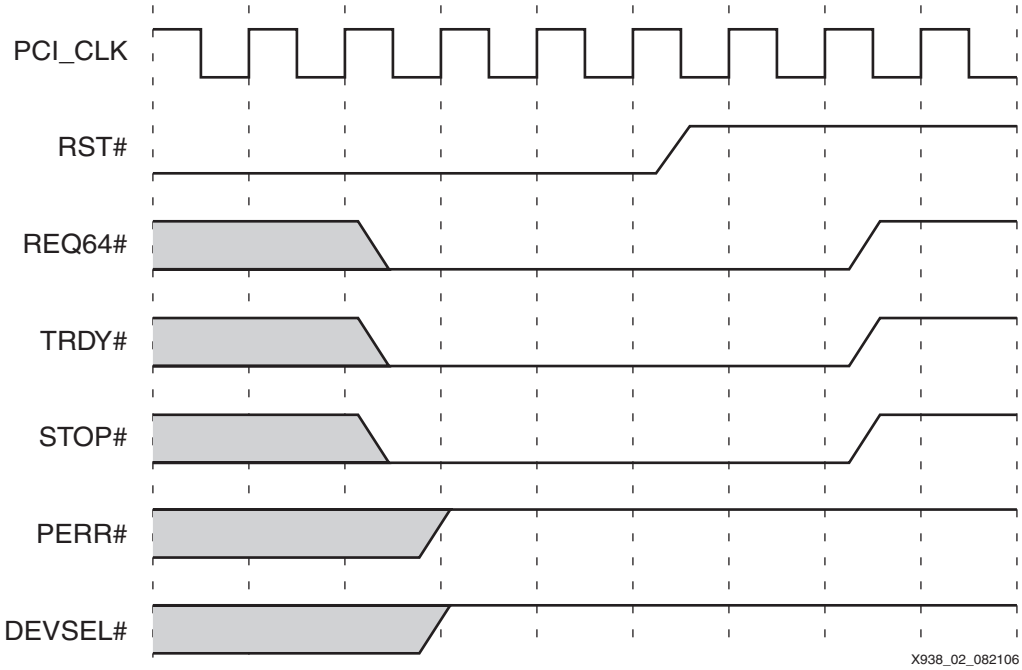

Figure 2: **PCI-X Initialization Pattern**

PCI-X add-in cards indicate their mode and maximum speed capability using the M66EN and PCIXCAP signals. The use of these signals is beyond the scope of this application note. For more information on M66EN and PCIXCAP, refer to section 6.2 of the *PCI-X Protocol Addendum to the PCI Local Bus Specification v2.0a.*

PCI-X Operation Modes

The LogiCORE solution for the PCI-X standard operates in one of the following modes (each one requires a separate bitstream):

- PCI mode only at 33 MHz
- PCI-X mode only, up to 133 MHz
- Dual mode - PCI at 33 MHz and PCI-X at 66 MHz (Virtex-II and Virtex-II Pro devices only)

The bitstreams for the different modes of operation are differentiated by using the correct combination of wrapper, UCF, and LogiCORE files. In the majority of cases, the same user application can work with either mode. To create two bitstreams, synthesize with different wrapper files and implement with different UCF and core files. The reference design files ("Reference Design," page 11) help accomplish these tasks. For more information on the wrapper, UCF, and LogiCORE files used to create the different bitstreams are found in Chapter 3 of the *Getting Started Guide for PCI-X* delivered with the LogiCORE files.

A CPLD controls loading the correct bitstream from an onboard PROM when two bitstreams are used for both PCI and PCI-X operation. This method is outlined below.

Dynamic Board Reconfiguration

To support both PCI and PCI-X modes of operation, a default bitstream is loaded at power-up, and the FPGA is configured. If the core determines it is in the wrong bus mode, a new bitstream must be loaded. A single bitstream designed for 64-bit buses can support both 32-bit and 64-bit PCI/PCI-X buses. The bitstreams differ only by the core's mode of operation.

At power-up, a default bitstream is loaded into the FPGA. The bridge source on the PCI bus drives the initialization pattern when RST# is deasserted. At this point, the LogiCORE for PCI-X knows both the bus mode and the bus width. If the core is in the wrong mode for the bus, it asserts the user application output signal – RTR. The RTR signal indicates that the core must

be reconfigured. The RTR signal is also used to inform the CPLD to reload the FPGA with the new bitstream. Table 3 describes RTR and other relevant PCI-X user output signals.

Table 3: **Core Reconfiguration Output Signals for PCI-X**

Signal	Type	Description
RTR	Output	RTR is an output signal used in dual configuration designs. When RTR is asserted by the interface, the user application must reconfigure the FPGA device with an alternate bitstream.
PCIX_EN	Output	When the PCIX_EN output signal is asserted High, the core is operating in PCI-X mode.
PCIW_EN	Output	When the PCIW_EN output signal is asserted High, the core is operating in 64-bit mode.

This reconfiguration process must occur within 2^{25} clock cycles on a PCI bus and 2^{26} clock cycles on a PCI-X bus. These values are guaranteed by the respective specification and are called T_{RHFA} in each specification. This parameter indicates the time from RST# deassertion until the first valid configuration transaction. Because the FPGA is being configured after RST# deasserts, no bus transactions should occur during this time. Even though the FPGA does not see the RST# assertion, the core is guaranteed by design to present in a known good state.

Since a new design does not see a RST# deassertion on the bus, the bus width at power-up is unknown. To resolve this problem, the core outputs PCIW_EN to indicate the initial bus width as an input to the CPLD reconfiguration design. The CPLD remembers the state of PCIW_EN and forces the core into the correct bus width by forcing certain inputs after the new bitstream is loaded. Depending on the version of the core, the bus width is set in one of two ways. The newer Virtex-5 cores have input ports to set the bus width. The bus width in the Virtex-4 core (v5) for PCI-X is set using two bits of the 512-bit configuration vector.

Virtex-5 Cores: (v6) for PCI-X Designs and (v4) for PCI Designs

The newer cores supporting Virtex-5 designs have input ports allowing the user to set both the bus mode and bus width (Table 4).

Table 4: **Virtex-5 LogiCORE Reconfiguration Userapp Signals**

Signal	Type	Description
BM_DETECT_DIS	Input	**Bus Mode Detect Disable:** Active High. When asserted, this signal forces the core to start in the mode indicated by the BM_MANUAL_PCI port.
BM_MANUAL_PCI	Input	**Bus Mode Manual PCI:** When BM_DETECT_DIS is asserted, the LogiCORE for PCI-X designs samples this input to determine the operating mode. If this input is set to 1, then the core operates in PCI mode. If this input is set to 0, then the core operates in PCI-X mode. If BM_DETECT_DIS is not asserted, then this input has no effect on the core.
BW_DETECT_DIS	Input	**Bus Width Detect Disable:** Active High. When asserted, it forces the core to ignore the bus width initialization pattern and sample the BW_MANUAL_32B input instead.
BW_MANUAL_32B	Input	**Bus Width Manual as 32 Bit:** When BW_DETECT_DIS is asserted, the core assumes it is plugged into a 32-bit bus and the input is set to 1. Setting the input to 0 forces the core to assume it is plugged into a 64-bit capable bus. If BW_DETECT_DIS is not asserted, then this input has no effect on the core.

Virtex-4, Virtex-II Pro, Virtex-E (v5) Core for PCI-X Designs

The configuration file parameters used as part of a reconfiguration design are shown in Table 5. These bits are found in the 512-bit configuration vector input to the LogiCORE for PCI-X designs.

Table 5: **Configuration Vector Parameters**

Parameter	Bit	Description
Bus Mode Detect Disable	504	Setting this bit to 1 forces the LogiCORE for PCI-X to start in the mode set by bit 505.
Bus Mode Manual as PCI	505	If bit 504 is set to 1, setting this bit to 1 forces the core to operate in PCI mode. Setting this bit to 0 forces the core to operate in PCI-X mode. If bit 504 is not set, then this bit has no effect on the core.
Bus Width Detect Disable	502	Setting this bit to 1 forces the core to ignore the bus width initialization pattern and use the width indicated by the Bus Width Manual as 32 bit or bit 503 instead.
Bus Width Manual as 32-bit	503	If bit 502 is set to 1, setting this bit to 1 forces the core to assume it is plugged into a 32-bit bus. Setting this bit to 0 forces the core to assume it is plugged into a 64-bit capable bus. If bit 502 is not set, then this bit has no effect on the core.

Setting BM_DETECT_DIS (v4 and v6 core) or bit 504 (v5 core) does not disable the ability of the core to recognize the bus mode. The RTR output is asserted to the user application if the bitstream loaded is in the wrong mode for the bus. When using two bitstreams, one for PCI operation and one for PCI-X operation, BM_DETECT_DIS or bit 504 must be set to 1. The "Design Setup" section discusses these signals.

Board Setup

To reconfigure bitstreams, certain board design requirements are necessary. This application note uses the setup on the ML455 board. This board meets the basic requirements, including an FPGA that supports the core for PCI-X, a CPLD for the dynamic reconfiguration design, and a method of storing two bitstreams. Figure 3 shows the connections between the Virtex-4 FPGA, CPLD, and Platform Flash on the ML455 board.

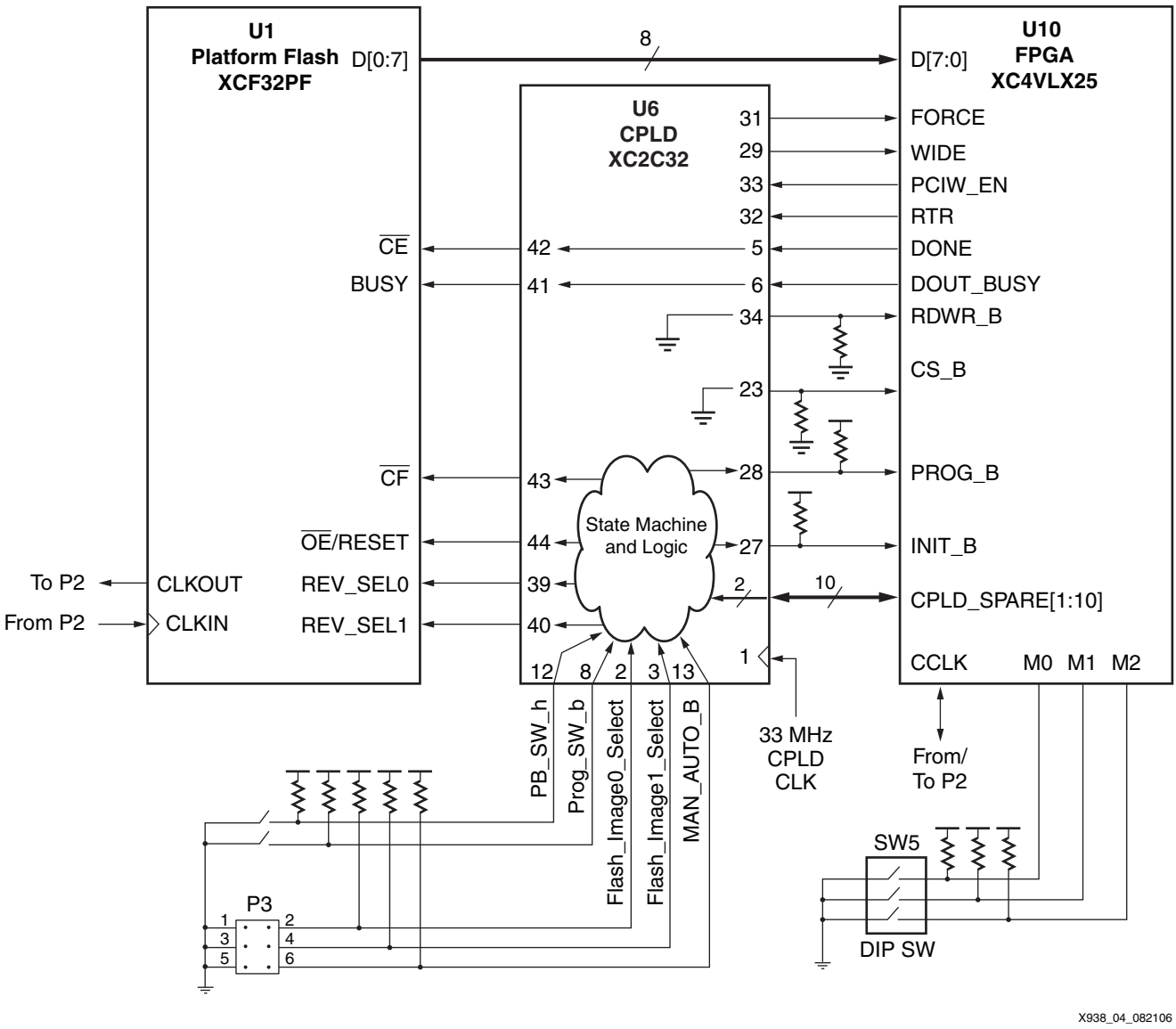

Figure 3: **ML455 Board Connections for Dynamic Bitstream Reconfiguration**

Detailed information on the ML455 board is available in the Virtex-4 Development Kit User Guide for PCI/PCI-X at: http://www.xilinx.com/bvdocs/userguides/ug084.pdf.

The FORCE and WIDE signals are used to drive the configuration vector bits 502 and 503, respectively. These signals are discussed in the "Design Setup" section.

Design Setup

This section describes a scenario using dynamic board reconfiguration to ensure the correct core bitstream is loaded. This design is tested using the ML455 board. "Reference Design," page 11 details the Virtex-4 (v5) core on the ML455 board. The design setup for using the Virtex-5 (v4) and (v6) cores varies slightly depending on the core chosen. Most of the differences are in the connections for the PCI-X bus mode detect and bus width manual signals in the user application design. The same CPLD design can be used with either core.

CPLD Dynamic Reconfiguration Design Ports

When a reconfiguration is necessary, the dynamic reconfiguration design communicates with the onboard PROM. The inputs and outputs of the design are shown in Table 6.

Table 6: **Inputs/Outputs of the CPLD Dynamic Reconfiguration Design**

Signal	Type	Description
CLK	Input	**PCI/PCI-X System Clock**: PCI/PCI-X clock from the PCI edge connector.
FPGA_RTR	Input	**FPGA_RTR**: Assertion of this signal begins the reconfiguration process. This signal is connected to the RTR output of the LogiCORE for PCI-X.
PCIW_EN	Input	**PCI/PCI-X Bus Width**: Bus width indicator user application output from the core for PCI/PCI-X. When the PCIW_EN output signal is asserted, the core is operating in 64-bit mode.
MAN_AUTO_B(1)	Input	**Manual Override**: Allows manual override to force bitstream selection using the inputs FLASH_IMAGE0_SELECT and FLASH_IMAGE1_SELECT.
FLASH_IMAGE0_SELECT(1)	Input	**Revision Select Bit 0**: Manual override bitstream revision select bit 0.
FLASH_IMAGE1_SELECT(1)	Input	**Revision Select Bit 1**: Manual override bitstream revision select bit 1.
FPGA_DONE	Input	**FPGA Done Output**: Configuration DONE pin output from FPGA. Indicates when configuration is complete.
DOUT_BUSY	Input	**FPGA Busy Output**: Only used for readback in Virtex-4 devices. Otherwise, driven Low.
PROG_SW_B(1)	Input	**PROG Push Button**: Onboard PROG push button to manually restart configuration of FPGA.
PB_SW_H(1)	Input	**Board Push Button**: Used as reset for FPGA and PROM to manually restart configuration of FPGA.
EDGE_RST_I_B	Input	**PCI/PCI-X System Reset**: PCI/PCI-X system reset from PCI edge connector.
CE	Output	**PROM CE Input**: Tied to FPGA_DONE. When High, puts PROM into lower-power standby mode.
BUSY	Output	**PROM Busy Input**: Tied to DOUT_BUSY. When asserted, causes PROM to pause configuration.
FORCE	Output	**Force**: When asserted High, causes the core to use the WIDTH output to know the PCI/PCI-X bus width.
WIDTH	Output	**Width**: Indicates the bus width to the core upon reconfiguration. Setting this output to 1 indicates a 32-bit bus width, and setting it to 0 indicates a 64-bit bus width.
REV_SEL0	Output	**Revision Select 0**: Bitstream revision select bit 0 output to the PROM.
REV_SEL1	Output	**Revision Select 1**: Bitstream revision select bit 1 output to the PROM.
PROG_B	Output	**FPGA PROG_B Input**: When asserted Low, the FPGA clears its configuration memory and restarts the configuration process.
INIT_B	Output	**FPGA INIT_B Input**: Can be used to delay configuration of the FPGA.
CF	Output	**PROM CF Input:** Assertion resets PROM an initiates configuration sequence.
OE_RESET	Output	**PROM OE/RESET Input**: Holds PROM in reset when deasserted Low.
FPGA_RDWR_B	Output	**FPGA RDWR_B Input**: Indicates direction of FPGA SelectMap data bus.
FPGA_CS_B	Output	**FPGA CS_B Input**: Chip select to enable the SelectMap data bus.

Notes:

1. These ports are used on the ML455 development board. When manual bitstream selection and push buttons are not available on the board, these ports are optional.

CPLD Dynamic Configuration Design Control State Machine

The CPLD design assumes that the bus starts in either PCI or PCI-X mode. Depending on the assumed start-up mode, the state machine reconfigures only once to the other mode when FPGA_RTR is asserted. For example, if the CPLD design assumes the default bus mode is

PCI, and the bus actually starts in PCI-X mode, it will trigger the PROM to reconfigure the FPGA. Otherwise, nothing will happen. Figure 4 shows the main control state machine for the CPLD design.

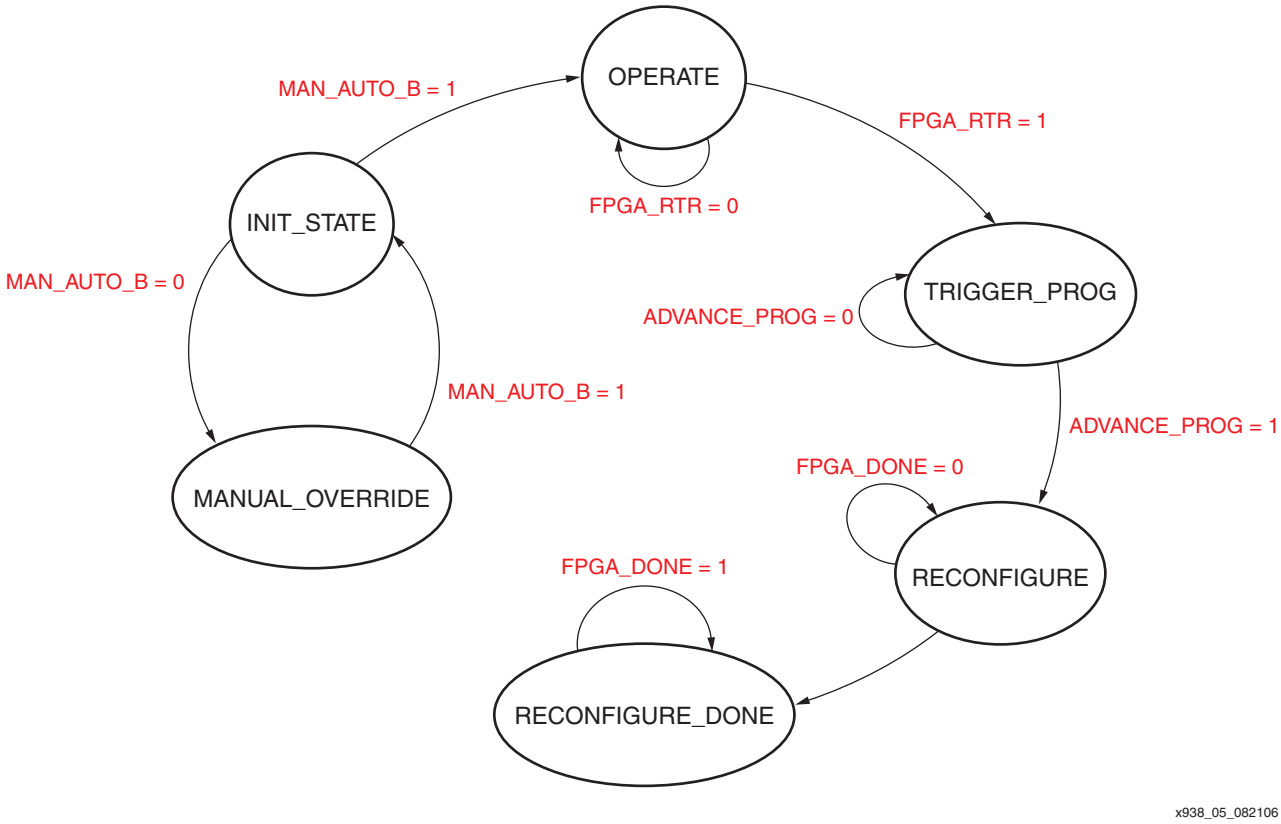

Figure 4: **CPLD Control State Machine**

INIT_STATE

INIT_STATE is the initial state after power-on or after reset is applied. From this state, the reference design allows the user to manually override the design and force the loading of a particular bitstream revision through board jumpers. The ML455 board supports this feature. Otherwise, there is not a need for a transition to the Manual Override state.

OPERATE

The OPERATE state monitors the FPGA_RTR. An asserted FPGA_RTR indicates that the core is in the wrong bus mode and the FPGA needs to be reconfigured. FGPA_RTR is an input to the CPLD design and is connected to the RTR output of the core. If FPGA_RTR asserts, then the state machine transitions to the next state and triggers the reconfiguration sequence; otherwise, it remains in the OPERATE state.

TRIGGER_PROG

TRIGGER_PROG is the first state of the reconfiguration sequence. Here, the state machine waits for 300 ns while PROG_B is asserted Low to the FPGA. The FPGA requires PROG to be asserted a minimum of 300 ns. Once this is completed, it transitions to the next state where the new bitstream is loaded.

RECONFIGURE

During the RECONFIGURE state, the new bitstream is loaded into the FPGA. This state monitors the FPGA_DONE input, which is the FPGA DONE pin, to ascertain when the FPGA reconfiguration is complete.

RECONFIGURE_DONE

RECONFIGURE_DONE is the final state in the reconfiguration sequence. Once the device is reconfigured, the bus mode should not change unless a system reset occurs, causing a transition back to the INIT_STATE.

MANUAL_OVERRIDE

The MANUAL_OVERRIDE state is optional. It is entered if the MAN_AUTO_B input to the CPLD design is Low. This allows users to force the bitstream revision number to be loaded based on board jumpers or switches. The ML455 board it supports this feature prototyping or debugging.

Generating the FORCE and WIDE Output

Once the FPGA is reconfigured, the CPLD design must store the width of the bus based on the PCIW_EN input from the FPGA. There are two steps to setting the bus width of the newly reconfigured bitstream, regardless of whether it is in PCI or PCI-X mode. During the reconfiguration sequence, when the control state machine is in the OPERATE state, a register is defined and captures the state of PCIW_EN. The value is then assigned to the WIDE output of the CPLD design as shown in the following Verilog example code:

```
reg reg_wide;

  always @(posedge CLK or posedge RST)
  begin: wide_reg
    if (RST) reg_wide <= 1'b0;
    else if(state == `OPERATE)reg_wide <= PCIW_EN ;
  end

  assign #DLY WIDE = reg_wide;
```

Next, the CPLD design must only assert the FORCE output to the FPGA if a reconfiguration occurs. In other words, the design should not force the bus width if the bus starts in the same mode expected by the FPGA design. To accomplish this, a register is set when the control state machine moves through the RECONFIGURE state and remains set unless a reset occurs as shown in the Verilog example code below.

```
reg reg_force;

  always @(posedge CLK or posedge RST)
  begin: force_reg
    if (RST) reg_force <= 1'b0;
    else if(state == `RECONFIGURE_DONE)
      reg_force <= 1'b1 ;
  end

  assign #DLY FORCE = reg_force;
```

Depending on the core being used, the user application design must be modified to use the FPGA inputs WIDE and FORCE.

Modifying the Core User Files

LogiCORE (v5) for PCI-X Designs

When using the Virtex-4 core (v5) follow this section to set up a user application design.

The user must generate two designs resulting in two bitstreams. One bitstream is for PCI mode, the other for PCI-X mode.

In the PCI mode bitstream, configuration bit 504 and bit 505 must be set to 1, as shown in the following Verilog example code:

```
// Bus Mode Detect Disable
assign CFG[504] = 1'b1;
// Bus Mode Manual as PCI
assign CFG[505] = 1'b1;
```

The PCI-X mode bitstream uses the following settings:

```
// Bus Mode Detect Disable
assign CFG[504] = 1'b1;
// Bus Mode Manual as PCI
assign CFG[505] = 1'b0;
```

After the bitstream is loaded, the design expects the bus to be in the mode set by these bits. If not, the core asserts RTR to signal it is in the wrong mode. Each design must also drive the bus width detect and bus width manual bits as follows:

```
// Bus Width Detect Disable
assign CFG[502] = FORCE; // FORCE is FPGA input
// Bus Width Manual As 32-Bit
assign CFG[503] = !WIDE; // WIDE is FPGA input
```

As shown in Figure 3, the signals FORCE and WIDE are inputs to the FPGA. They are controlled by the CPLD as discussed in the section "Generating the FORCE and WIDE Output." In this example, WIDE is inverted because, in the CPLD example, design WIDE is High if the bus is in 64-bit mode. The core expects configuration bit 503 to be Low if the core should be in 64-bit mode and High if it should be in 32-bit mode.

Each design provides the RTR and PCIX_EN output signals to the CPLD as shown in Figure 3.

LogiCORE (v4) for PCI or LogiCORE (v6) for PCI-X

The Virtex-5 cores (v4 and v6) do not use the configuration vector used by the Virtex-4 core (v5).

The user must generate two designs resulting in two bitstreams. One bitstream is for PCI mode, the other for PCI-X mode. For the PCI mode bitstream, the core (v6) for PCI-X can be used in PCI 33 MHz mode, and the core (v4) for PCI can be used in PCI 66 MHz mode.

In the PCI mode bitstream, the core user application inputs BM_DETECT_DIS and BM_MANUAL_PCI must be tied to logic 1, as shown in the following Verilog example.

```
// Bus Mode Detect Disable
assign BM_DETECT_DIS = 1'b1;
// Bus Mode Manual as PCI
assign #DLY BM_DETECT_DIS = 1'b1;
```

The PCI-X mode bitstream uses the following settings:

```
// Bus Mode Detect Disable
assign BM_DETECT_DIS = 1'b1;
// Bus Mode Manual as PCI
assign BM_DETECT_DIS = 1'b0;
```

After the bitstream is loaded, the design expects the bus to be in the mode set by these bits. If not, the core asserts RTR to signal it is in the wrong mode. Each design must also drive the bus width detect and bus width manual bits as follows:

```
// Bus Width Detect Disable
assign BW_DETECT_DIS = FORCE; // FORCE is FPGA input
// Bus Width Manual As 32-Bit
assign BW_MANUAL_32B = !WIDE; // WIDE is FPGA input
```

As shown in Figure 3, the signals FORCE and WIDE are inputs to the FPGA. They are controlled by the CPLD as discussed in the section "Generating the FORCE and WIDE Output." In this example, WIDE is inverted because, in the CPLD example design, WIDE is High if the bus is in 64-bit mode. The core expects the BW_MANUAL_32B to be Low if the core is in 64-bit mode, and High if it is in 32-bit mode.

Reference Design

The reference design files (VHDL and Verilog source code) for the CPLD are available in a downloadable ZIP file from the Xilinx website at:

www.xilinx.com/bvdocs/appnotes/xapp938.zip

The PCI-X standard portion of the reference design files use the Virtex-4 core (v5) for PCI-X. Designers can apply the techniques discussed in Design Setup when targeting the Virtex-5 (v4) core for PCI or the Virtex-4 core (v6) for PCI-X. The CPLD design can be used with either core as it is not specific to the FPGA device type.

This design is fully tested and verified on the ML455 board. The reference design also includes wrappers for the (v5) core for PCI-X, showing the changes necessary to the configuration vector as discussed in the section "Modifying the Core User Files." The reference design will be updated to include the Virtex-5 (v4) and (v6) core files when testing is complete.

ML455 Development Board for PCI/PCI-X Designs

The ML455 is a 3.3V, 64-bit board. This board is not a Universal board and must not be plugged into 5V PCI slots. Chapter 1 of the ML455 User Guide (UG084) describes how to identify a 3.3V system board slot.

M66EN and PCIXCAP

Users must know how to set the ML455 board to indicate to the system if it is a PCI-X or PCI card. As described in Chapter 6 of the *PCI-X Protocol Addendum to the PCI Local Bus Specification, Rev 2.0*, the combination of M66EN and PCIXCAP advertise the ability of the card to the system controller. The ML455 board allows users to customize the board through the following settings:

M66EN – 66 MHz Enable

P1.B49 is wired to two-pin header pin P9.1. With the P9 jumper shunt removed, M66EN has a 0.01 µF capacitor to GND. Placing the jumper shunt across pins 1 and 2 of P9 shorts M66EN to GND.

- M66EN = GND indicates 0 to 33 MHz operation.
- M66EN = Open indicates 33 MHz to 66 MHz operation. M66EN is pulled up on the system board.

PCIXCAP – PCI-X Capability

P1.B38 is wired to 3-pin header P8 (center pin).

- P8.1 is wired to GND through a 10 KΩ pulldown resistor.
- P8.2 is wired to P1.B38 and a 0.01 µF capacitor to GND.
- P8.3 is wired to GND.
- A jumper shunt across P8 pins 1 and 2 indicates that the card is PCI-X 66 capable.
- No jumper shunt across P8 indicates that the card is PCI-X 133 capable.
- A jumper shunt across P8 pins 2 and 3 indicates that the card is not PCI-X capable (i.e., is PCI, not PCI-X).

The terms "P1.B49" and "P1.B38" refer to pins defined on the PCI edge connector. For more information on the pin designations and the jumper settings, please refer to the ML455 User Guide (UG084).

LEDs

This reference design uses the ML455 onboard LEDs to indicate the mode, reconfiguration, and clock pulse status (Table 7).

Table 7: **LED Definitions**

LED Signal	Board Designation	Description
LED0	USER1 D1	**PCI-X Enable**: LED is on when core is in PCI-X mode.
LED1	USER2 D2	**Bus Width**: LED is on when core is in 64-bit mode.
LED2	USER3 D3	**RTR**: LED is on when the core needs to be reconfigured.
LED3	USER4 D4	**Clock Pulse**: LED blinks noticeably when clock is live.

Design Files

The reference design is available in both VHDL and Verilog. The directory structure of the design files is shown in Figure 5. Already implemented bit files are also provided to allow for quick download and use on the ML455 board. However, all necessary files including the PCI-X wrapper and UCF files are provided to re-create the bit files. The actual LogiCORE for PCI-X is not included in the application note ZIP file because users must obtain a license to access this core. This can be done by visiting the PCI-X Lounge.

After the LogiCORE license is obtained, users can generate a core using the CORE Generator tool. If the generated core file, pcix_core.ngc, is dropped into the `xapp938/<Verilog` or `VHDL>/v5_pcix_core_files/src/xpci` directory, then the PCI-X design can be implemented by using the `run_xilinx.bat` or `run_xilinx.csh` scripts in the `/example/xilinx/` directories. The result will be both a bit file for PCI-X and PCI for the PROM on the ML455 board.

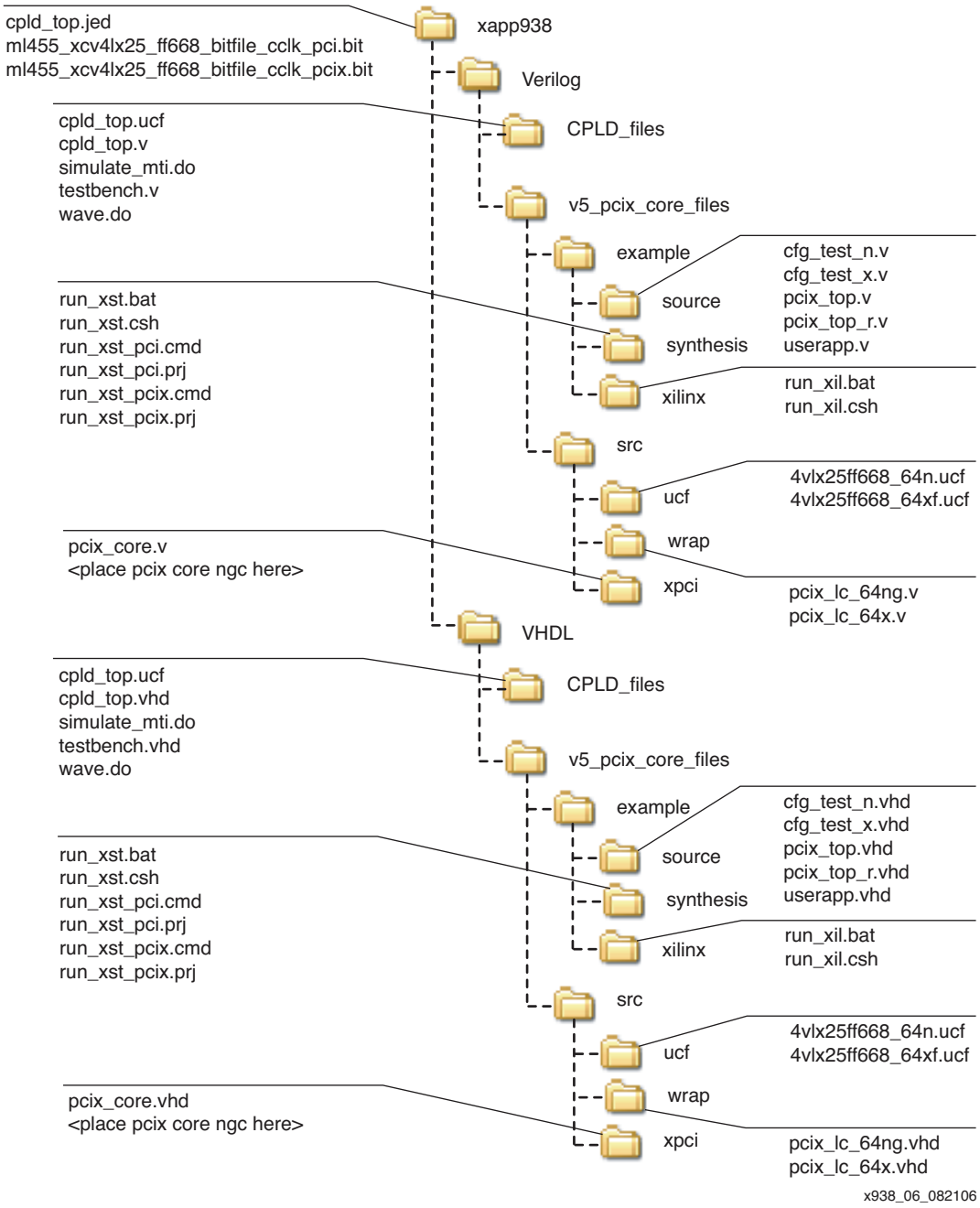

Figure 5: **Directory Structure of Reference Design**

The CPLD Dynamic Reconfiguration Design files are available in the directory `xapp938/Verilog/CPLD_files`. This design can be implemented by creating an ISE Project through the ISE Project Navigator targeting a XC2C32-4VQ44 device. Implementing this project results in programming files for the ML455 CPLD. A simulation testbench and a simulation script file for ModelSim is provided in the `xapp938/<Verilog or VHDL>/CPLD_files` directory. In ModelSim, target this directory and run the `simulate_mti.do` file to run the simulation.

Verification Example

Users can quickly verify the design using the ML455 board and the bitstreams provided. The reference design was tested on different systems, including a Dell Precision 670 workstation (with a 64-bit PCI-X slot). This section describes how to bring up the board in this Dell system and verify the design. The easiest way to determine if the correct bitstream is loaded is by using a PCI bus tool to read the configuration space of the board once it is powered up. For simplicity, the configuration bitstreams each have a unique device ID as shown in Table 8.

Table 8: **Device IDs for Available Bitstreams**

Bitstream	Device ID
`ml455_xcv4lx25_ff668_bitfile_cclk_pci.bit`	2222
`ml455_xcv4lx25_ff668_bitfile_cclk_pcix.bit`	1111

The device ID is located in bits `31:16` of the first DWORD in the PCI configuration header at address `0x00`. This is a simple way to verify the correct bitstream is loaded.

The CPLD design as provided assumes the bus mode will be PCI and will load the PCI bitstream at power-up. If the bus turns out to be in PCI-X mode, then the core will notify the CPLD by asserting the FPGA_RTR signal indicating a new bitstream needs to be loaded into the FPGA. The CPLD will then trigger a reconfiguration process and load the PCI-X design.

Using the ML455 User Guide as a reference:

1. Program the onboard PROM to have the PCI bitstream as revision 0 and the PCI-X bitstream as revision 1.
2. Program the CPLD with the provided CPLD bitstream.
3. Remove the jumper shunt from P8 on the ML455 board to indicate that the board is capable of PCI-X 133 operation.
4. Plug the board into the PCI-X 133 slot and power on the system. The following steps will then occur:
    a. Power is applied to the board and the PCI bitstream is loaded.
    b. The system recognizes the board as PCI-X 133 capable due to the PCIXCAP setting and starts the bus in PCI-X mode.
    c. The core recognizes it is in the wrong bus mode and asserts the RTR user application output, which is an input to the CPLD design.
    d. The CPLD reconfigures the FPGA with the PCI-X bitstream.
    e. The device ID of the ML455 board indicates the PCI-X mode bitstream is loaded by showing a value of `1111`.

If the ML455 board is changed by applying a jumper shunt across P8 pins 2 and 3 causing PCIXCAP to be pulled to ground, it indicates that the card is not PCI-X capable. Since the PCI mode bitstream is only capable of running up to 33 MHz, a jumper shunt should be placed across P9, shorting M66EN to ground. Now, powering up the system with the same design files will result in the following:

1. Power is applied to the board and the PCI bitstream is loaded.
2. The system recognizes the board as PCI capable due to the PCIXCAP setting and starts the bus in PCI mode.
3. The system recognizes the board supports 0 to 33 MHz PCI operation due to M66EN.
4. The core recognizes the bus is in PCI mode, which matches the loaded bitstream and does not assert RTR to the user.
5. The device ID of the ML455 board indicates the PCI mode bitstream is loaded by showing a value of `2222`.

Various tools can be used to probe PCI configuration space. A simple one to use on a Windows system is the PciTree tool available at: http://www.pcitree.de. Using this tool, the device ID of the board can be verified.

Conclusion

The LogiCORE for PCI-X is capable of running in both PCI and PCI-X modes. For a PCI-X design to be fully compliant, it must be able to operate in both PCI-X mode and PCI mode. This application note describes how users can use a CPLD to dynamically reconfigure the FPGA to support PCI-X and full PCI backwards compatibility in their PCI-X system when two separate bitstreams are required – one for each mode.

References

PCI-X Electrical and Mechanical Addendum to the PCI Local Bus Specification, Revision 2.0a

LogiCORE for PCI-X v5.0 User Guide (UG160)

PCI Local Bus Specification, Revision 3.0

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/28/07	1.0	Initial Xilinx release.