XSB-300E Parallel Port Interface

October 20, 2003 (Version 1.0) Application Note by D. Vanden Bout

Summary

This application note describes the default parallel port interface circuit that is programmed into the XC9572XL CPLD on the XSB-300E Board. It also discusses how to change the parallel port interface to support other features.

The Default Parallel Port Interface

Listing 1 shows the VHDL code for the default parallel port interface that is programmed into the XC9572XL CPLD on the XSB-300E Board. This interface provides two functions:

- It transfers configuration bitstreams from the PC to the SpartanIIE FPGA.
- It lets the PC and the SpartanIIE communicate through the parallel port after the FPGA is configured.

How the VHDL implements these functions is described below.

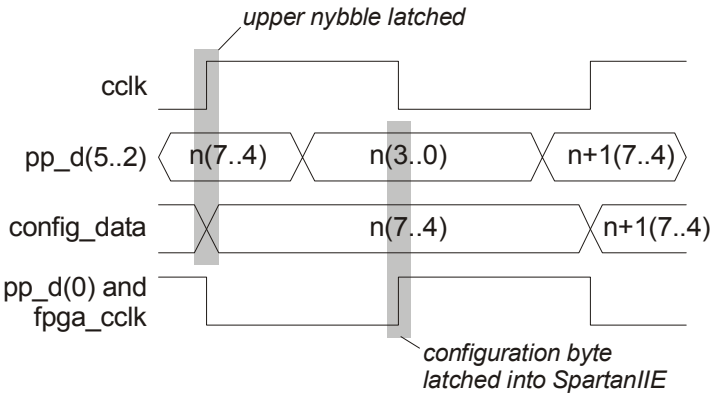
Lines 40 and 41 just give more meaningful names to some input signals.

The main functionality of the CPLD is defined by the process starting on line 43. It begins by setting the default values of some signals on lines 45–48. The JTAG circuitry of the SpartanIIE FPGA is kept quiescent so it cannot interfere by holding its clock pin low on line 49.

The circuitry that actually controls the configuration of the SpartanIIE device is described on lines 50–58. On line 50, the CPLD pulls down the M0 mode pin of the SpartanIIE device to set the FPGA into the Slave Parallel configuration mode (the M1 and M2 mode pins of the SpartanIIE are hard-wired on the XSB-300E PCB). The PROGRAM pin for the SpartanIIE is connected to data line D7 of the parallel port on line 51. A low level on D7 will initiate the configuration of the SpartanIIE and its DONE pin will go low. The low level on the DONE pin activates the functions described on lines 55–58. Low levels are placed on the chip-select and write-strobe of the SpartanIIE on lines 56 and 57, respectively. This enables the writing of byte-wide configuration data into the SpartanIIE. These pins are released after DONE goes high because they become general-purpose I/O pins after configuration is completed.

The configuration data bytes arrive as two four-bit nybbles over data lines D2–D5 (line 40). The upper nybble of each configuration byte is stored in the config_data register on the rising edge of the cclk (lines 88–93). The inverse of parallel port data line D0 drives the internal cclk signal (line 52), so the data is latched into the config_data register on the falling edge of D0.

The upper nybble in the config_data register is concatenated with the lower nybble on the D2–D5 data lines to form a complete byte of configuration data (line 58). The configuration clock for the SpartanIIE is the inverse of the clock that controls the config_data register (line 53). So the configuration data is latched into the SpartanIIE on the falling edge of cclk. The overall process of getting byte *n* of configuration data into the SpartanIIE looks like this:



Once all the configuration data enters the SpartanIIE, it will raise its DONE pin. This activates the circuitry described on lines 59–83 that allows the SpartanIIE to communicate with the PC parallel port through the

CPLD. The chip-select and write-strobe of the SpartanIIE are used to receive a clock and reset signal from the PC to the FPGA (lines 60 and 61, respectively). These signals control state machines that are loaded into the SpartanIIE.

The FPGA outputs an address to ther CPLD to select one of the clauses in the case statement on lines 62–82. As can be seen on line 41, the address is composed of the lower three bits of the peripheral address bus and the value on the INIT# pin of the FPGA (which becomes a general-purpose I/O pin after the FPGA is configured.)

If the CPLD address output by the FPGA is 0000, then the circuitry described by statements 63–67 is activated. This circuitry creates a pathway through the CPLD whereby the PC can send and receive the I^2C clock and data signals through the data and status pins of the parallel port to the FPGA. In this case, the FPGA must be configured to pass these signals onto the SCL and SDA signals that go to the I^2C peripherals on the XSB-300E Board (i.e., the programmable clock and the video decoder chips). This parallel port➔CPLD➔FPGA pathway allows the XSTOOLs software utilities to program the oscillator frequency and the video decoder options.

If the CPLD address output by the FPGA is 0001, then the circuitry described by statements 68–72 is activated. This circuitry creates a pathway through the CPLD whereby the PC can send and receive the clock, data and chip-select signals through the data and status pins of the parallel port to the FPGA. The FPGA must be configured to pass these signals onto the CCLK, CDTI, CDTO and CSN# pins of the stereo codec on the XSB-300E Board. This parallel port➔CPLD➔FPGA pathway lets the XSTOOLs software utilities program the stereo codec options.

If the CPLD address output by the FPGA is 0010, then the circuitry described by statements 73–75 is activated. This circuitry creates a pathway through the CPLD whereby the PC can send and receive data through the data and status pins of the parallel port to the FPGA. The FPGA must be configured to assemble complete address and data fields from these signals (using a state machine driven by the clock from line 60) which it can then use to access various peripherals on the XSB-300E Board. This pathway is used, for example, so the GXSLOAD/XSLOAD software utilities can upload/download the SRAM and SDRAM on the board.

If the CPLD address output by the FPGA is 0011, then the circuitry described by statements 76–78 is activated. This circuitry lets the PC force a value on the lower seven bits of the peripheral data bus using the data pins of the parallel port. The FPGA can operate upon this value and return a result through the parallel port status pins using three bits of the peripheral address bus. This pathway is used by the GXSPORT/XSPORT software utilities to apply test vectors to simple designs loaded into the FPGA and receive results.

The CPLD does not open a pathway between the parallel port and the rest of the XSB-300E Board when the CPLD address is 01XX (line 79). Addresses in this range can be used for new CPLD modes of operation.

If the FPGA places a high level on its INIT# pin, then the CPLD circuitry defined by lines 80–81 is activated. Bit A19 of the peripheral address bus is passed through the CPLD to a status pin of the parallel port while the rest of the parallel port remains isolated from the board. This circuitry is used solely to pass a status signal that reports the health of the XSB-300E Board back to the GXSTEST/XSTEST software utility running on the PC.

Changing the Parallel Port Interface

The parallel port interface is stored in the nonvolatile Flash of the XC9572XL CPLD on the XSB-300E Board. Any design you load into the CPLD will become active as soon as the XSB-300E Board powers up. So it is possible to load a faulty interface design into the CPLD that makes it impossible to program the SpartanIIE even after you cycle the power. The only solution is to explicitly reprogram the CPLD with a functional interface using GXSLOAD. Then the XSB-300E Board will function correctly again.

All you need to do when changing the parallel port interface is to add a new clause to the case statement on lines 62–83. Just select one of the unused CPLD addresses and add the VHDL code for your additional circuitry there.

When generating a new interface for the CPLD, you must set the USERCODE signature register to the four-character string `<4>!`. The XSTOOLs utilities look for this signature in the CPLD to verify that a valid interface is present.

Listing 1: VHDL code for the default CPLD parallel port interface.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use IEEE.std_logic_arith.all;
4
5  entity dwnldpar is
6     port(
7        -- parallel port data and status pins
8        pp_d:            in     std_logic_vector(7 downto 0);
9        pp_s:            out    std_logic_vector(5 downto 3);
10
11       -- FPGA configuration pins
12       fpga_m:          out   std_logic_vector(0 downto 0);  -- config. mode select (out)
13       fpga_program_n:  out    std_logic;    -- active-low config. initiate (out)
14       fpga_cclk:       out    std_logic;    -- config. clock (out)
15       fpga_cs_n:       out    std_logic;    -- active-low chip-select (out)
16       fpga_write_n:    out    std_logic;    -- active-low write-enable (out)
17       fpga_init_n:     inout  std_logic;    -- config. initialization (in)
18       fpga_done:       in     std_logic;    -- config. done (in)
19       fpga_tck:        out    std_logic;    -- JTAG clock (out)
20
21       -- peripheral bus
22       pb_d:            inout  std_logic_vector(7 downto 0); -- config. data (out)
23       pb_a:            inout  std_logic_vector(19 downto 0) -- address bus (in)
24
25    );
26 end entity dwnldpar;
27
28
29 architecture arch of dwnldpar is
30    constant LO: std_logic := '0';
31    constant HI: std_logic := '1';
32    constant HIZ: std_logic := 'Z';
33    constant SLAVE_PARALLEL_MODE: std_logic_vector(0 downto 0) := "0";
34    signal cclk: std_logic;
35    signal config_data, nybble: std_logic_vector(3 downto 0);
36    signal cpld_addr: std_logic_vector(3 downto 0);
37
38    begin
39
40    nybble    <= pp_d(5 downto 2);          -- data from PC to board
41    cpld_addr <= fpga_init_n & pb_a(2 downto 0);   -- selects CPLD profile
42
43    process(pp_d,pb_a,nybble,config_data,cclk,fpga_done,cpld_addr)
44    begin
45       fpga_cs_n       <= HIZ;
46       fpga_write_n    <= HIZ;
47       pb_a            <= (others=>HIZ);
48       pb_d            <= (others=>HIZ);
49       fpga_tck        <= LO;              -- deactivate FPGA JTAG circuit
50       fpga_m          <= SLAVE_PARALLEL_MODE;-- set FPGA config mode
51       fpga_program_n  <= pp_d(7);         -- FPGA PROGRAM# comes from parallel port
52       cclk            <= not pp_d(0);     -- internal configuration clock
53       fpga_cclk       <= not cclk;        -- FPGA configuration clock
54
55       if fpga_done=LO then-- FPGA is not configured
56          fpga_cs_n    <= LO;              -- enable writing of config. data
57          fpga_write_n <= LO;
58          pb_d         <= config_data & nybble;   -- two nybbles of config data
59       else   -- FPGA is configured
60          fpga_cs_n    <= not pp_d(1);     -- clock for data interchange
61          fpga_write_n <= pp_d(6);         -- reset for data interchange FSM
62          case conv_integer(unsigned(cpld_addr)) is
```

```
63              when 0 => -- I2C programming is selected by the FPGA
64                 pb_d(0)   <= not pp_d(1);  -- I2C clock to SAA7114 & osc chips
65                 pb_d(1)   <= pp_d(6);      -- I2C data to SAA7114 & osc chips
66                 pp_s(3)   <= pb_a(3);      -- I2C clock from SAA7114 & osc chips
67                 pp_s(4)   <= pb_a(4);      -- I2C data from SAA7114 & osc chips
68              when 1 => -- stereo codec programming is selected by the FPGA
69                 pb_d(0)   <= not pp_d(1);  -- config. clock to codec chip
70                 pb_d(1)   <= pp_d(6);      -- config. data to codec chip
71                 pb_d(3)   <= pp_d(5);      -- chip-select to codec chip
72                 pp_s(4)   <= pb_d(2);      -- config. data from codec chip
73              when 2 => -- data interchange interface
74                 pb_d(5 downto 2) <= pp_d(5 downto 2); -- data nybble from PC to FPGA
75                 pp_s(5 downto 3) <= pb_a(5 downto 3); -- data bits from FPGA to PC
76              when 3 => -- GXSPORT/XSPORT interface
77                 pb_d(6 downto 0) <= pp_d(6 downto 0);
78                 pp_s(5 downto 3) <= pb_a(5 downto 3);
79              when 4 | 5 | 6 | 7 =>  -- currently undefine CPLD modes
80              when others => -- CPLD is not selected by the FPGA
81                 pp_s(5)      <= pb_a(19); -- for reporting GXSTEST status
82           end case;
83        end if;
84
85     end process;
86
87     -- gather 4-bit data from parallel port
88     process(cclk)
89     begin
90        if rising_edge(cclk) then
91           config_data <= nybble;
92        end if;
93     end process;
94
95  end architecture arch;
```

Listing 2: User-constraint file for CPLD pin assignments.

```
#
# pin assignments for the XC9572XL CPLD chip on the XSB Board

# peripheral bus
net pb_d<0>      loc=p2;       # data bit D0 (in/out)
net pb_d<1>      loc=p4;       # data bit D1 (in/out)
net pb_d<2>      loc=p5;       # data bit D2 (in/out)
net pb_d<3>      loc=p6;       # data bit D3 (in/out)
net pb_d<4>      loc=p7;       # data bit D4 (in/out)
net pb_d<5>      loc=p8;       # data bit D5 (in/out)
net pb_d<6>      loc=p9;       # data bit D6 (in/out)
net pb_d<7>      loc=p10;  # data bit D7 (in/out)
net pb_a<0>      loc=p1;       # address bit A0 (in/out)
net pb_a<1>      loc=p64;  # address bit A1 (in/out)
net pb_a<2>      loc=p63;  # address bit A2 (in/out)
net pb_a<3>      loc=p62;  # address bit A3 (in/out)
net pb_a<4>      loc=p61;  # address bit A4 (in/out)
net pb_a<5>      loc=p60;  # address bit A5 (in/out)
#net pb_a<6>     loc=p59;  # address bit A6 (in/out)
#net pb_a<7>     loc=p58;  # address bit A7 (in/out)
#net pb_a<8>     loc=p45;  # address bit A8 (in/out)
#net pb_a<9>     loc=p44;  # address bit A9 (in/out)
#net pb_a<10>    loc=p57;  # address bit A10 (in/out)
#net pb_a<11>    loc=p43;  # address bit A11 (in/out)
#net pb_a<12>    loc=p56;  # address bit A12 (in/out)
#net pb_a<13>    loc=p46;  # address bit A13 (in/out)
#net pb_a<14>    loc=p47;  # address bit A14 (in/out)
#net pb_a<15>    loc=p52;  # address bit A15 (in/out)
#net pb_a<16>    loc=p51;  # address bit A16 (in/out)
#net pb_a<17>    loc=p48;  # address bit A17 (in/out)
#net pb_a<18>    loc=p42;  # address bit A18 (in/out)
net pb_a<19>     loc=p50;  # address bit A19 (in/out)
#net pb_oe_n     loc=p12;  # active-low output enable (out)
#net pb_we_n     loc=p49;  # active-low write enable (out)

# parallel port
net pp_d<0>      loc=p33;  # data pin D0 (in)
net pp_d<1>      loc=p32;  # data pin D1 (in)
net pp_d<2>      loc=p31;  # data pin D2 (in)
net pp_d<3>      loc=p27;  # data pin D3 (in)
net pp_d<4>      loc=p25;  # data pin D4 (in)
net pp_d<5>      loc=p24;  # data pin D5 (in)
net pp_d<6>      loc=p23;  # data pin D6 (in)
net pp_d<7>      loc=p22;  # data pin D7 (in)
net pp_s<3>      loc=p34;  # status pin S3 (out)
net pp_s<4>      loc=p20;  # status pin S4 (out)
net pp_s<5>      loc=p35;  # status pin S5 (out)

# FPGA configuration interface
net fpga_m<0>    loc=p36;  # config. mode select (out)
net fpga_program_n  loc=p39;  # active-low config. initiate (out)
net fpga_cclk    loc=p16;  # config. clock (out)
net fpga_cs_n    loc=p15;  # active-low chip-select (out)
net fpga_write_n loc=p19;  # active-low write-enable (out)
#net fpga_bsy    loc=p18;  # config. busy (in)
net fpga_init_n     loc=p38;  # config. initialization (in)
net fpga_done    loc=p40;  # config. done (in)
net fpga_tck     loc=p13;  # JTAG clock (out)

#net flash_ce_n     loc=p11;  # Flash RAM chip-enable (out)

#net cpld_clk    loc=p17;  # CPLD clock (in)
```