XSA Parallel Port Interface

March 17, 2004 (Version 1.1)

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 XSA 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 XSA Board. This interface provides two functions:

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

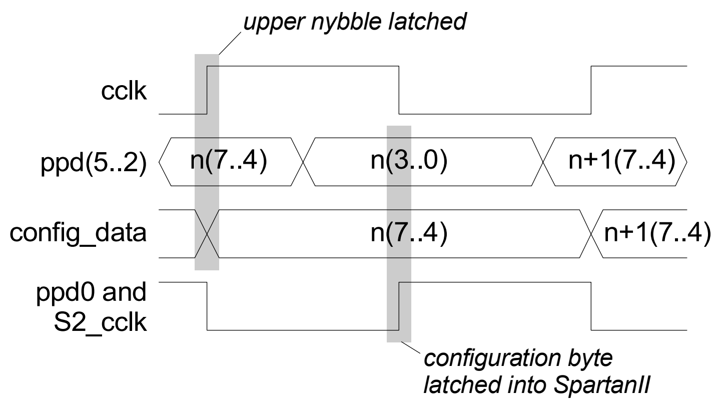
How the VHDL implements these functions is described below.

Lines 39–42 disable other chips and functions on the XSA Board so they cannot interfere with the configuration of the SpartanII device. The JTAG circuitry of the SpartanII FPGA is kept quiescent by holding its clock pin low on line 42. The Flash RAM is disabled by pulling its chip-enable pin high on line 43. The Flash chip-enable is allowed to float once the SpartanII device is configured (as indicated when the SpartanII DONE pin goes high) so the SpartanII can take control of the Flash chip-enable.

The circuitry that actually controls the configuration of the SpartanII device is described on lines 45–62. The CPLD pulls down the M0 mode pin of the SpartanII device to set the FPGA into the Slave Parallel configuration mode (the M1 and M2 mode pins of the SpartanII are hard-wired on the XSA PCB). The PROGRAM pin for the SpartanII is connected to data line D7 of the parallel port on line 48. A low level on D7 will initiate the configuration of the SpartanII and its DONE pin will go low. The low level on the DONE pin forces low levels on the chip-select and write-strobe of the SpartanII on lines 49-50. This enables the writing of byte-wide configuration data into the SpartanII. 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. The upper nybble of each configuration byte is stored in the config_data register on the rising edge of the cclk (lines 57–62). The inverse of parallel port data line D0 drives the internal cclk signal, 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. This configuration data is passed to the SpartanII on line 53. The configuration clock for the SpartanII is the inverse of the clock that controls the config_data register (line 52). So the configuration data is latched into the SpartanII on the falling edge of cclk. The overall process of getting byte *n* of configuration data into the SpartanII looks like this:



Once all the configuration data enters the SpartanII, it will raise its DONE pin. This will raise the chip-select and write-strobe of the SpartanII (lines 49–50) and disable further writing of configuration data. It will also make the CPLD release control of the SpartanII configuration data input pins (lines 54 and 55).

These pins become general purpose I/O pins after DONE goes high and the FPGA can use them to access data from the Flash or to drive the seven-segment LED. If the SpartanII is not accessing the Flash (i.e., the Flash chip-enable, fceb, is high) then the CPLD will light up the decimal-point segment of the LED to indicate that the SpartanII is currently configured while releasing the remaining data lines (line 54). But if the SpartanII enables the Flash, then the CPLD releases all the data lines to prevent contention with the FPGA (line 55).

Lines 65 and 66 describe how the CPLD connects the parallel port data lines to the general-purpose I/O pins of the SpartanII. Line 65 connects the eight parallel port data lines to the SpartanII so it can receive data from the PC. (The lower two data bits are inverted to mimic the parallel port connection used by the XS95, XS40 and XSV Boards.) These same FPGA pins are used by the SpartanII to connect to address pins A8–A13, the write-enable and the output-enable of the Flash. Therefore, the CPLD releases these pins to avoid contention when the SpartanII lowers the Flash chip-enable and begins to access the Flash.

On line 66 the CPLD connects three of the parallel port status lines to the SpartanII so the FPGA can transfer data back to the PC. The SpartanII also has a direct connection to a fourth status line. (The CPLD uses the fifth status line for the TDO output of its JTAG interface, so it is not available for general-purpose transfer of data to the PC.)

Finally, the CPLD connects the output of the programmable oscillator on the XSA Board to a dedicated clock input of the SpartanII (line 69).

The default parallel port interface implements the following connections between the SpartanII and the parallel port, LED and programmable oscillator.

Pin Function	SpartanII Pin#
Parallel Port Data Pin D0	50
Parallel Port Data Pin D1	48
Parallel Port Data Pin D2	42
Parallel Port Data Pin D3	47
Parallel Port Data Pin D4	65
Parallel Port Data Pin D5	51
Parallel Port Data Pin D6	58
Parallel Port Data Pin D7	43
Parallel Port Status Pin S3	40
Parallel Port Status Pin S4	29
Parallel Port Status Pin S5	28
Parallel Port Status Pin S6	78
LED Segment S0	67
LED Segment S1	39
LED Segment S2	62
LED Segment S3	60
LED Segment S4	46
LED Segment S5	57
LED Segment S6	49
LED Segment DP	44
Programmable Oscillator	88

Changing the Parallel Port Interface

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

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.

There are probably only three things you might want to change in the default parallel port interface:

1. You might want to give the SpartanII access to the decimal-point of the seven-segment LED instead of having the CPLD use it to display the configuration status of the FPGA. To do this, change the bitstring on line 54 to "ZZZZZZZZ". Note that once you do this you will no longer get any visual indication of whether your SpartanII bitstreams were downloaded correctly into the FPGA.

2. You might want to use all eight parallel port data pins to send data to the SpartanII. Therefore you will have to prevent any data presented on bit D7 from affecting the PROGRAM pin of the SpartanII or it will erase its configuration. You could connect the PROGRAM pin to a constant logic '1' and then you would have to interrupt the power to the XSA Board to force a reconfiguration.

3. You might want to change the clock that is passed to the SpartanII on line 69. You can pass the programmable oscillator output through a divider and on to the SpartanII. Or you could drive the SpartanII clock input with one of the parallel port data pins to do single-stepping of a design. Or you could have the CPLD release the SpartanII clock input entirely so you can drive it with an external clock source.

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

```
library ieee;
use ieee.std_logic_1164.all;

entity dwnldpar is
   port(
      -- parallel port data and status pins
      ppd: in std_logic_vector(7 downto 0);
      pps: out std_logic_vector(5 downto 3);

      -- programmable oscillator
      clk: in std_logic;

      -- Spartan2 FPGA pins
      S2_tck: out std_logic; -- driver to Spartan2 JTAG clock
      S2_cclk: out std_logic;-- driver to Spartan2 config clock
      S2_progb: out std_logic;   -- driver to Spartan2 program pin
      S2_csb: out std_logic; -- driver to Spartan2 config. chip-select
      S2_wrb: out std_logic; -- driver to Spartan2 config. write strobe
      S2_initb: in std_logic;-- input from Spartan2 init pin
      S2_done: in std_logic; -- input from Spartan2 done pin
      S2_d: out std_logic_vector(7 downto 0); -- drivers to Spartan2 data pins and 7-seg LED
      S2_m: out std_logic_vector(0 downto 0); -- Spartan2 config mode pins
      S2_clk: out std_logic; -- clock output to Spartan2
      S2_ppd: out std_logic_vector(7 downto 0);  -- parallel port data pins to Spartan2
      S2_pps: in std_logic_vector(5 downto 3);-- parallel port status pins from Spartan2
      fceb: inout std_logic  -- Flash chip-enable
   );
end entity dwnldpar;


architecture arch of dwnldpar is
   constant LO: std_logic := '0';
   constant HI: std_logic := '1';
   constant SLAVE_PARALLEL_MODE: std_logic_vector(0 downto 0) := "0";
   signal cclk: std_logic;
   signal config_data: std_logic_vector(3 downto 0);
   component pullup port(O: out std_logic); end component;

   begin
   -- disable other chips on the XSV Board so they don't interfere
   -- during the configuration of the Spartan2 FPGA
   S2_tck    <= LO; -- deactivate Spartan2 JTAG circuit
   fceb      <= HI when S2_done=LO else 'Z';  -- disable Flash during config.

   -- connect Spartan2 configuration pins
   S2_m      <= SLAVE_PARALLEL_MODE;-- set Spartan2 config mode pins
   u0: pullup port map(O=>S2_progb);-- place a pullup on the Spartan2 PROGRAM pin
   S2_progb  <= LO when ppd(7)=LO else 'Z';   -- programming pulse comes from parallel port
   S2_csb    <= LO when S2_done=LO else 'Z';  -- enable writing of data
   S2_wrb    <= LO when S2_done=LO else 'Z';  -- during Spartan2 config. phase
   cclk      <= not ppd(0);
   S2_cclk   <= not cclk;
   S2_d      <= (config_data & ppd(5 downto 2)) when S2_done=LO else
                "ZZZZZZ1Z" when fceb=HI else     -- show Spartan2 config status on LED-DP
                "ZZZZZZZZ";  -- release control of Spartan2 Flash/LED pins

   process(cclk)
   begin
      if(cclk'event and cclk=HI) then
         config_data <= ppd(5 downto 2);
      end if;
   end process;
```

```

    -- connect the parallel port data and status pins to the Spartan2
    S2_ppd    <= (ppd(7 downto 2) & not(ppd(1 downto 0))) when fceb=HI else "ZZZZZZZZ";
    pps       <= S2_pps;

    -- send the programmable oscillator clock to the Spartan2
    S2_clk    <= clk;

end architecture arch;
```

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

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

# Spartan2 FPGA connections to CPLD
net S2_clk       loc=p42;
net S2_tck       loc=p13;
# net S2_dout    loc=p18;
# net S2_din     loc=p2;
net S2_wrb       loc=p19;
net S2_csb       loc=p15;
# net S2_initb   loc=p38;
net S2_done      loc=p40;
net S2_progb     loc=p39;
net S2_cclk      loc=p16;
net S2_m<0>      loc=p36;
net S2_d<0>      loc=p2;
net S2_d<1>      loc=p4;
net S2_d<2>      loc=p5;
net S2_d<3>      loc=p6;
net S2_d<4>      loc=p7;
net S2_d<5>      loc=p8;
net S2_d<6>      loc=p9;
net S2_d<7>      loc=p10;
net S2_ppd<0>    loc=p45;
net S2_ppd<1>    loc=p44;
net S2_ppd<2>    loc=p57;
net S2_ppd<3>    loc=p43;
net S2_ppd<4>    loc=p56;
net S2_ppd<5>    loc=p46;
net S2_ppd<6>    loc=p49;
net S2_ppd<7>    loc=p12;
net S2_pps<3>    loc=p1;
net S2_pps<4>    loc=p64;
net S2_pps<5>    loc=p63;

# Flash RAM
# net fd<0>      loc=p2;
# net fd<1>      loc=p4;
# net fd<2>      loc=p5;
# net fd<3>      loc=p6;
# net fd<4>      loc=p7;
# net fd<5>      loc=p8;
# net fd<6>      loc=p9;
# net fd<7>      loc=p10;
# net fa<0>      loc=p1;
# net fa<1>      loc=p64;
# net fa<2>      loc=p63;
# net fa<3>      loc=p62;
# net fa<4>      loc=p61;
# net fa<5>      loc=p60;
# net fa<6>      loc=p59;
# net fa<7>      loc=p58;
# net fa<8>      loc=p45;
# net fa<9>      loc=p44;
# net fa<10>     loc=p57;
# net fa<11>     loc=p43;
# net fa<12>     loc=p56;
# net fa<13>     loc=p46;
# net fa<14>     loc=p47;
# net fa<15>     loc=p52;
# net fa<16>     loc=p51;
```

```
# net fa<17>     loc=p48;
# net frstb      loc=p50;  # Flash reset
# net foeb       loc=p12;  # Flash output-enable
# net fweb       loc=p49;  # Flash write-enable
net fceb         loc=p11;  # Flash chip-enable

# DIP and pushbutton switches
# net dipsw<1>   loc=p47;
# net dipsw<2>   loc=p52;
# net dipsw<3>   loc=p51;
# net dipsw<4>   loc=p48;

# 7-segment LEDs
# net s<0>       loc=p10;
# net s<1>       loc=p2;
# net s<2>       loc=p9;
# net s<3>       loc=p8;
# net s<4>       loc=p5;
# net s<5>       loc=p7;
# net s<6>       loc=p6;
# net dp         loc=p4;

# programmable oscillator
net clk          loc=p17;

# parallel port
net ppd<0>       loc=p33;
net ppd<1>       loc=p32;
net ppd<2>       loc=p31;
net ppd<3>       loc=p27;
net ppd<4>       loc=p25;
net ppd<5>       loc=p24;
net ppd<6>       loc=p23;
net ppd<7>       loc=p22;
net pps<3>       loc=p34;
net pps<4>       loc=p20;
net pps<5>       loc=p35;
```