Parallel Cable III Emulator for the XSB-300E Board

October 4, 2003 (Version 1.0) Application Note by D. Vanden Bout

Summary

This application note shows how to configure the XC9572XL CPLD on the XSB-300E Board so its parallel port interface emulates the functions of the Xilinx Parallel Cable III. This lets you use the Xilinx WebPACK JTAG tools with the XSB-300E Board through its simple 25-wire downloading cable.

Why Emulate the Parallel Cable III?

Xilinx WebPACK software contains tools for downloading and testing SpartanIIE FPGAs through their their JTAG interface. One of the ways these tools access the FPGA is through a Parallel Cable III connected from the FPGA to the parallel port of a PC. A schematic for the Parallel Cable III (henceforth referred to as PCBLIII) is shown in Figure 1.

The SpartanIIE FPGA on the XSB-300E Board is accessed from the PC parallel port through a simple 25-wire cable that connects to an XC9572XL CPLD on the XSB-300E Board. The CPLD can be programmed to pass signals from the parallel port to and from the JTAG pins of the SpartanIIE device. The XSB-300E Board is supplied with a default CPLD configuration that lets you download bitstreams to the SpartanIIE using the GXSLOAD utility provided by XESS.

This application note describes an alternate circuit that allows the CPLD to emulate the JTAG functions of the PCBLIII. By loading this circuit into the CPLD, you can use all the Xilinx downloading and testing tools with the XSB-300E Board through the simple downloading cable provided by XESS.

VHDL for the Parallel Cable III Emulator

Listing 1 shows the VHDL code for the PCBLIII emulator 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 using the JTAG interface.
- After the SpartanIIE FPGA is configured and its DONE pin goes high, the JTAG interface is used to readback and/or test the FPGA.

How the VHDL implements these functions is described below.

Line 29 outputs a high logic level to status pin S3 of the parallel port. The Xilinx software checks for a high level on this status pin which indicates that power is being supplied to the PCBLIII.

The Xilinx software also checks for the presence of the PCBLIII by looping a signal from parallel port data pin D6 back through two of the status pins S5 and S7. Line 34 handles the loop from D6 back to S5. The CPLD cannot pass D6 to S7, however, because its TDO pin is already attached to S7. Therefore, the shunt on jumper JP1 has to be moved to the **xi** position to manually connect D6 and S7.

Line 36 drives the mode pins of the SpartanIIE FPGA to set it in the slave-serial configuration mode. This doesn't really do anything since the SpartanIIE will be programmed through its JTAG interface. Line 37 holds the /PROGRAM pin of the SpartanIIE at a high logic level to prevent accidental erasure of the FPGA configuration.

Lines 41-43 connect parallel port data pins D2, D1, and D0 to the SpartanIIE TMS, TCK, and TDI JTAG pins if D3 is low. The TMS, TCK and TDI inputs are allowed to float when D3 is high.

Line 46 passes the FPGA JTAG TDO signal back to the PC through status pin S4 of the paralle port. S4 is driven low when data pin D4 is low.

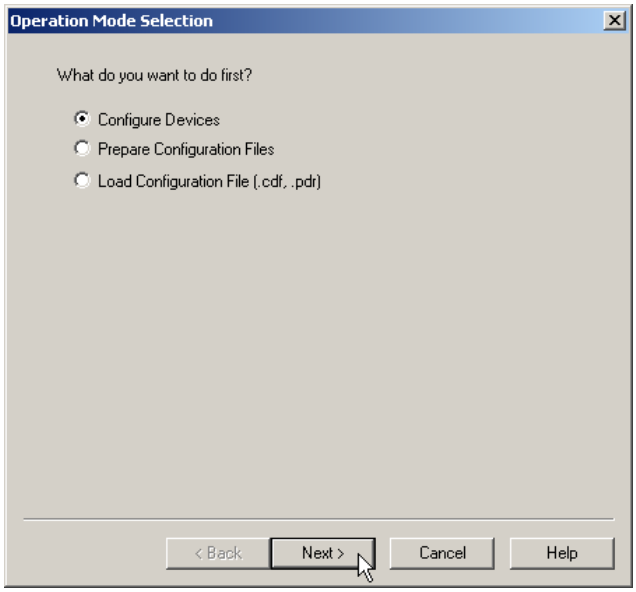
The PCBLIII emulator I/O pin assignments for the CPLD on the XSB-300E Board are shown in Listing 2.

Using the Parallel Cable III Emulator

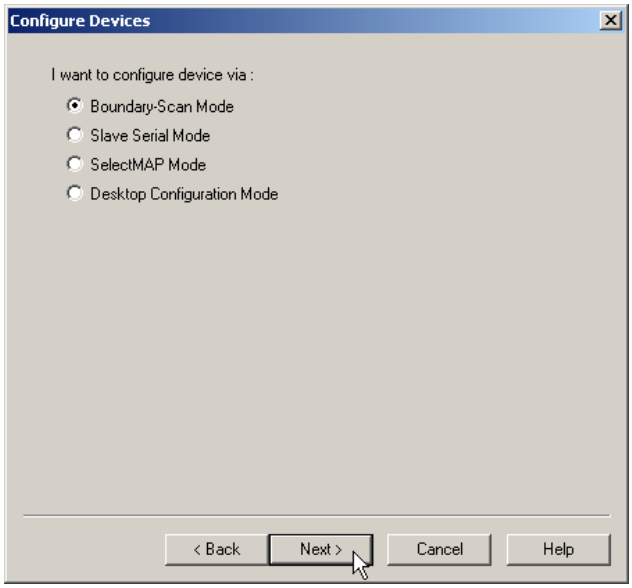
First, connect the XSB-300E Board to the parallel port of a PC through the simple 25-wire cable provided by XESS. Make sure the shunt on jumper JP1 is in the **xs** position. Then download the piiijtag.svf file into the XC9572XL CPLD using the GXSLOAD tool from XESS. Now move the shunt on JP1 to the **xi** position. At this point, the bitstream downloading portion of the PCBLIII emulator is active.

Next, double-click the Configure Device (iMPACT) icon in the Process pane of the WebPACK **Project Navigator** window. (This discussion assumes you already have a SpartanIIE design synthesized and implemented in WebPACK 5.2.).

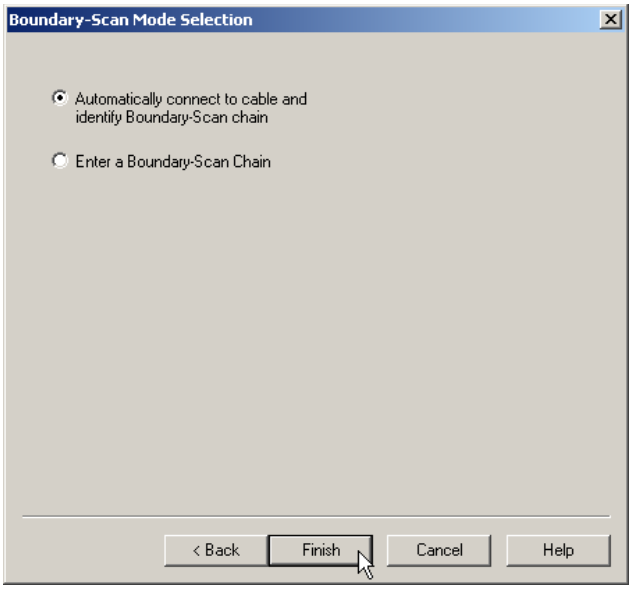
In the **Operation Mode Selection** window, select the Configure Devices option since we are going to configure the FPGA on the XSB-300E Board (i.e., download a bitstream file to it). Then click on the Next button.



Now the **Configure Devices** window appears. Previously, we programmed the CPLD on the XSB-300E Board so it would support programming of the FPGA through its boundary-scan (i.e. JTAG) pins. So select the Boundary-Scan Mode option and click on the Next button.

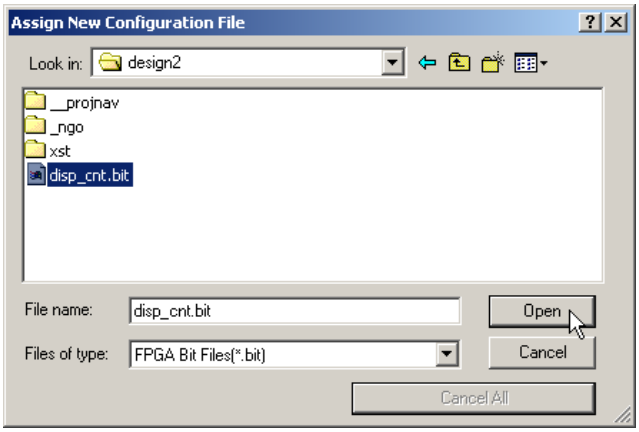


Boundary-scan mode allows the configuration of multiple FPGAs connected together in a chain. To do this, the iMPACT software needs to know the types of the FPGAs in the chain. We know there is just a single FPGA on the XSB-300E Board and we could easily describe this to iMPACT. But iMPACT can also probe the boundary-scan chain and automatically identify the types of the FPGAs. This is even easier, so we select the automatic identification option and click on the Finish button.

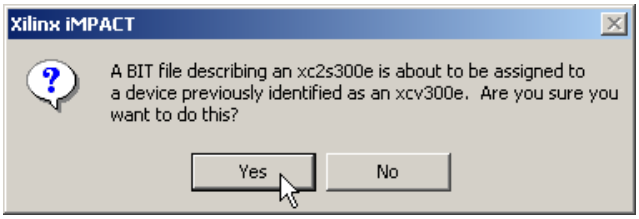


The iMPACT software will probe the boundary-scan chain and find there is a single FPGA in it. Now we need to tell iMPACT what bitstream file we want to download into this FPGA. Click on the OK button to proceed.

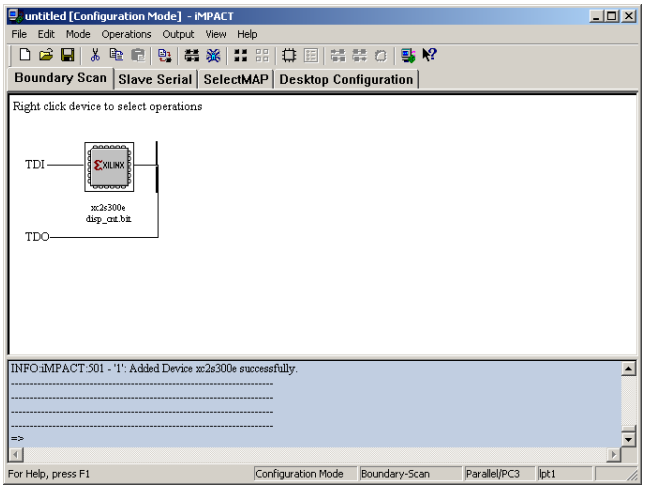
The **Assign New Configuration File** window now appears. Go to the folder where the bitstream file is located and highlight it. Then click on the Open button.



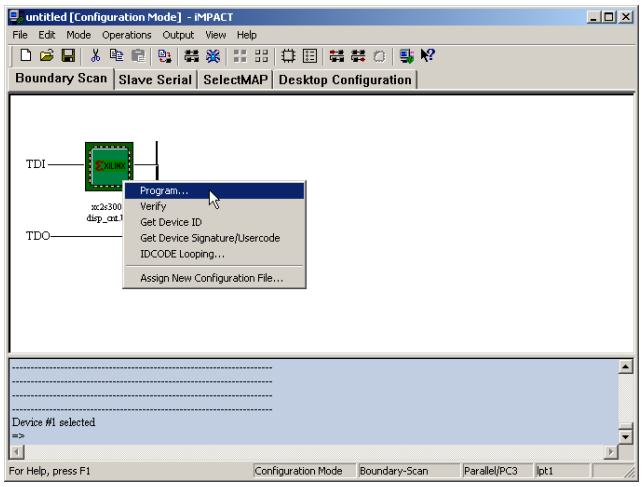
Now iMPACT may warn us that the bitstream file we selected is for an XC2S300E FPGA but it detected an XCV300E FPGA in the boundary-scan chain. The XC2S300E FPGA was developed as a low-cost variant of the XCV300E FPGA and they both share the same boundary-scan identification code. We know the XSB-300E Board has an XC2S300E FPGA that is compatible with the bitstream file, so we can click on the Yes button and move on.



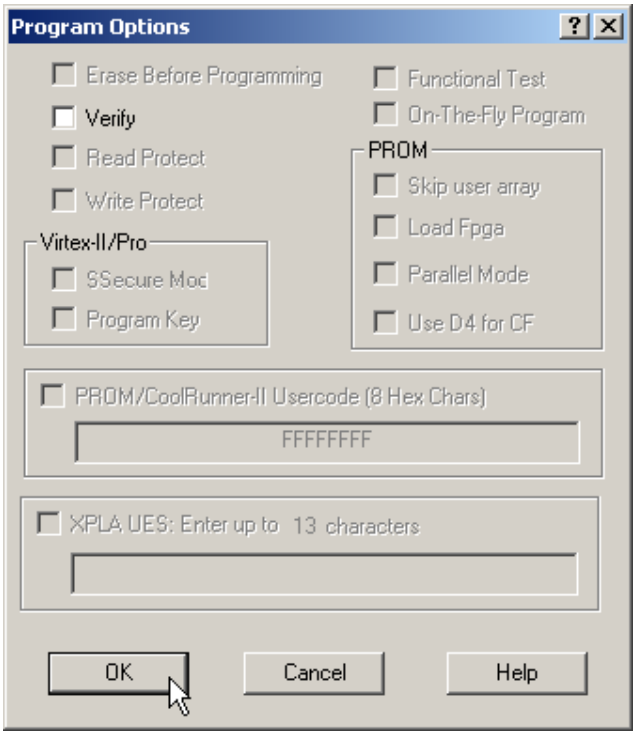
Now the main **iMPACT** window shows a boundary-scan chain consisting of a single XC2S300E FPGA.



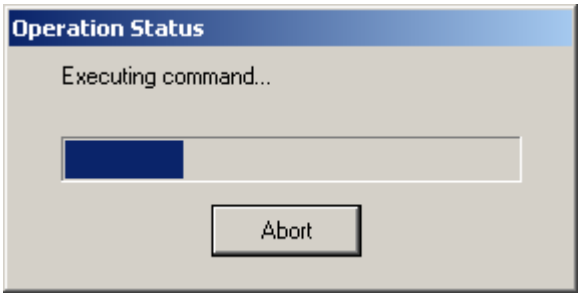
Now right-click on the xc2s300e icon and select the Program… item on the pop-up menu.



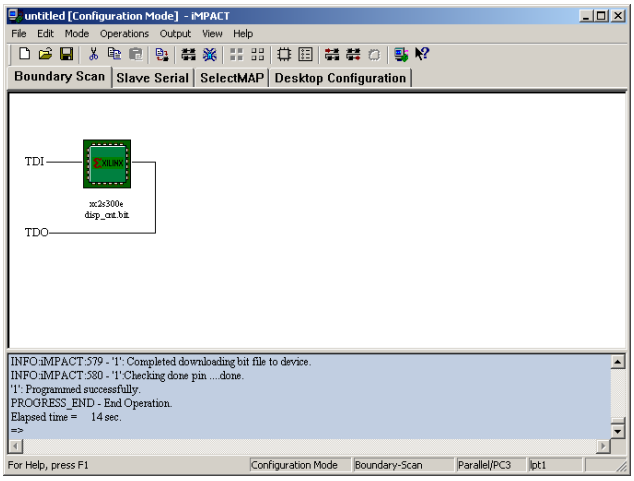
The **Program Options** window will appear. All we need to do at this point is click on the OK button to begin loading the bitstream file into the FPGA.



The progress of the bitstream download will be displayed. The download operation should complete within twenty seconds.



After the download operation completes, we can check the status messages in the bottom pane of the **iMPACT** window to see the FPGA was configured successfully.



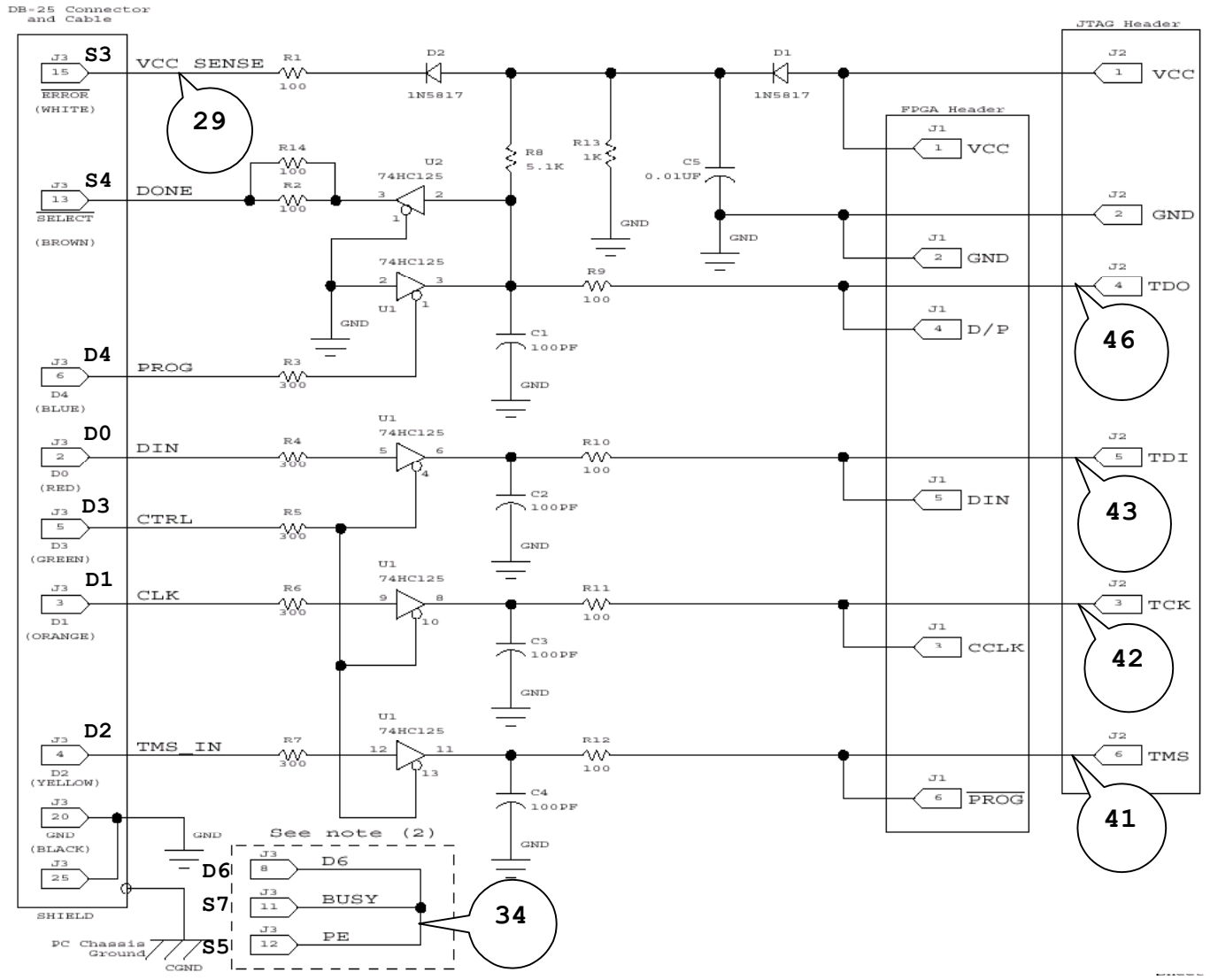


Figure 1: Xilinx Parallel Cable III schematic. The line numbers of the VHDL code in Listing 1 associated with each schematic element are shown.

Listing 1: VHDL code for the Parallel Cable III emulator.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity piiijtag is
5     port(
6        -- parallel port data and status pins
7        ppd:          in std_logic_vector(6 downto 0);
8        pps:          out   std_logic_vector(5 downto 3);
9
10       -- Spartan2E FPGA pins
11       S2_tck:       out   std_logic; -- driver to Spartan2E JTAG clock
12       S2_tms:       out   std_logic; -- driver to Spartan2E JTAG mode input
13       S2_tdi:       out   std_logic; -- driver to Spartan2E JTAG serial data input
14       S2_tdo:       in std_logic; -- input from Spartan2E JTAG serial data output
15       S2_prog_n: out   std_logic; -- driver to Spartan2E /PROGRAM pin
16       S2_m:         out   std_logic_vector(0 downto 0)  -- Spartan2E config mode pins
17    );
18 end piiijtag;
19
20 architecture arch of piiijtag is
21    constant NO:  std_logic := '0';
22    constant YES: std_logic := '1';
23    constant LO:  std_logic := '0';
24    constant HI:  std_logic := '1';
25    constant SLAVE_SERIAL_MODE: std_logic_vector(0 downto 0) := "1";
26 begin
27
28    -- the XSB power status is sent back through the parallel port status pin 3
29    pps(3)     <= HI;     -- tell the PC that the VCC for the XSB board is OK
30    -- the cable is detected by sending data through data pin 6 and returning
31    -- it on status pins 5 and 7.  Status pin 7 is used by the JTAG TDO
32    -- pin of the XC9500XL CPLD on the XSB Board, so place a shunt at position "xi"
33    -- of jumper JP1 to make this connection.
34    pps(5)     <= ppd(6);
35
36    S2_m       <= SLAVE_SERIAL_MODE; -- set Spartan2E config mode pins
37    S2_prog_n  <= HI;                    -- prevent any accidental reconfiguration
38
39    -- drive the Spartan2E JTAG pins from the parallel port when tristate
40    -- control pin (parallel port data pin 3) is low.
41    S2_tms     <= ppd(2) when ppd(3)=LO else 'Z';
42    S2_tck     <= ppd(1) when ppd(3)=LO else 'Z';
43    S2_tdi     <= ppd(0) when ppd(3)=LO else 'Z';
44
45    -- the JTAG TDO output is sent back through the status pin
46    pps(4)     <= S2_tdo when ppd(4)=HI else LO;
47
48 end 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_tms    loc=p18;
# net S2_din    loc=p2;
# net S2_wr_n   loc=p19;
net S2_tdo    loc=p19;
# net S2_cs_n   loc=p15;
net S2_tdi    loc=p15;
# net S2_init_n    loc=p38;
# net S2_done   loc=p40;
net S2_prog_n   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_a<0>   loc=p1;
# net S2_a<1>   loc=p64;
# net S2_a<2>   loc=p63;
# net S2_a<3>   loc=p62;
# net S2_a<4>   loc=p61;
# net S2_a<5>   loc=p60;
# net S2_a<6>   loc=p59;
# net S2_a<7>   loc=p58;
# net S2_a<8>   loc=p45;
# net S2_a<9>   loc=p44;
# net S2_a<10>     loc=p57;
# net S2_a<11>     loc=p43;
# net S2_a<12>     loc=p56;
# net S2_a<13>     loc=p46;
# net S2_a<14>     loc=p47;
# net S2_a<15>     loc=p52;
# net S2_a<16>     loc=p51;
# net S2_a<17>     loc=p48;
# net S2_rst_n     loc=p50;# Flash reset
# net S2_oe_n   loc=p12;# Flash output-enable
# net S2_we_n   loc=p49;# Flash write-enable
# net S2_ce_n   loc=p11;# Flash chip-enable

# 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;
```

```
63  # net fa<3>     loc=p62;
64  # net fa<4>     loc=p61;
65  # net fa<5>     loc=p60;
66  # net fa<6>     loc=p59;
67  # net fa<7>     loc=p58;
68  # net fa<8>     loc=p45;
69  # net fa<9>     loc=p44;
70  # net fa<10>    loc=p57;
71  # net fa<11>    loc=p43;
72  # net fa<12>    loc=p56;
73  # net fa<13>    loc=p46;
74  # net fa<14>    loc=p47;
75  # net fa<15>    loc=p52;
76  # net fa<16>    loc=p51;
77  # net fa<17>    loc=p48;
78  # net frst_n    loc=p50;# Flash reset
79  # net foe_n     loc=p12;# Flash output-enable
80  # net fwe_n     loc=p49;# Flash write-enable
81  # net fce_n     loc=p11;# Flash chip-enable
82
83  # DIP and pushbutton switches
84  # net dipsw<1>     loc=p47;
85  # net dipsw<2>     loc=p52;
86  # net dipsw<3>     loc=p51;
87  # net dipsw<4>     loc=p48;
88
89  # 7-segment LEDs
90  # net s<0>   loc=p10;
91  # net s<1>   loc=p2;
92  # net s<2>   loc=p9;
93  # net s<3>   loc=p8;
94  # net s<4>   loc=p5;
95  # net s<5>   loc=p7;
96  # net s<6>   loc=p6;
97  # net dp   loc=p4;
98
99  # programmable oscillator
100 # net clk    loc=p17;
101
102 # parallel port
103 net ppd<0>   loc=p33;
104 net ppd<1>   loc=p32;
105 net ppd<2>   loc=p31;
106 net ppd<3>   loc=p27;
107 net ppd<4>   loc=p25;
108 # net ppd<5>    loc=p24;
109 net ppd<6>   loc=p23;
110 # net ppd<7>    loc=p22;
111 net pps<3>   loc=p34;
112 net pps<4>   loc=p20;
113 net pps<5>   loc=p35;
114
```