XSV Flash Programming and Virtex Configuration

July 5, 2001 (Version 1.1)

Application Note by D. Vanden Bout

Summary

This application note describes the circuits that let the XC95108 CPLD program the Flash on the XSV Board and then configure the Virtex FPGA with the data stored in the Flash.

Using Flash with the XSV Board

The 16 Mb Flash chip on the XSV Board can be used to store configurations for the Virtex FPGA. When used in this way, there are three steps used to set up the Flash:

1. Configure the XC95108 CPLD with a programming circuit that connects the Flash to the parallel port.

2. Program the Flash by passing the Virtex configuration bitstream through the parallel port.

3. Configure the CPLD with a configuration circuit that will, upon power-up of the XSV Board, load the Virtex with the bitstream stored in the Flash.

The Flash Programming Interface

Listing 1 and Listing 2 show the VHDL code and pin assignments for the CPLD circuit that connects the Flash to the parallel port. This circuit is simply an interface that allows the PC to read and write the Flash using only four data bits and two control signals. The PC uses this simple interface to control the higher-level Flash programming functions such as erasing the Flash sectors before they are programmed with new data.

The Flash programming circuit performs the following functions:

- It collects six successive nybbles from the parallel port and concatenates these into a 24-bit Flash address.

- It collects two successive nybbles from the parallel port and concatenates these into a byte of Flash data.

- It writes the data into the Flash at the given address and then loops back to await the arrival of another set of address and data nybbles.

- While gathering the nybbles for an address, it also reads the byte of Flash data from the previous loop and passes it to the parallel port as two successive nybbles.

How the VHDL implements these functions is described below.

Lines 10-32 of Listing 1 define the interface for the circuit. It uses six data pins of the parallel port: four for passing data and address nybbles, one for a synchronizing clock to drive the state machine in the CPLD, and one as a reset for the state machine. Four status pins of the parallel port are used to send data nybbles back to the PC and to report the current state of the CPLD state machine. The CPLD also interfaces to the address, data, and control pins of the Flash chip and the PROGRAM pin of the Virtex FPGA.

The eight states of the Flash programming state machine are defined on lines 43-53. Six states are used to gather the 24-bit Flash address in nybble chunks, and then two more states are used to collect the Flash data.

Line 63 makes the CPLD pull the Virtex PROGRAM pin low so it stays in its unconfigured state. This tristates the pins of the Virtex so it can't interfere with the programming of the Flash chip. Line 64 makes sure the reset of the Flash chip is released so it can be programmed.

Lines 65-67 just rename the parallel port I/O with more understandable names that reflect their underlying functions.

The main process for the Flash programming state machine begins on line 71. Lines 74-81 just set the default values for the outputs from the state machine.

Lines 87-132 implement the six states that concatenate nybbles from the parallel port into a 24-bit address. During each of these states, the nybble from the parallel port is placed into the appropriate slot in the address register. The current state is also reported back to the PC through the parallel port status lines in states load_a20, load_a8, load_a4, and load_a0. The Flash programming code in the PC uses this information to make sure it is in sync with the state machine.

However, during states load_a16 and load_a12 the status lines are used to carry the nybbles of a data byte from the Flash. The location of this data is stored in the next_addr register in state load_a20 (line 92). This address appears on the Flash address lines at the start of state load_a16 and persists until the next_addr register is written to again. During states load_a16 and load_a12, the Flash chip-enable and output-enable lines are forced low and the upper and lower nybbles of the Flash data at the given address are passed through the parallel port (lines 103 and 113, respectively). The Flash programming code in the PC gathers these nybbles and assembles the byte of Flash data.

Lines 134-152 implement the two states that concatenate two data nybbles into a byte of data that is written into the Flash at the address loaded during the previous six states. The actual write occurs in the second half of state load_d0 when the clock is low. this gives the address time to settle from the previous cycle before the write occurs. When the clock goes high to end the write pulse, the state machine transfers to state load_a20. Note that when state load_a20 is first entered, line 90 ensures that the Flash data lines are still carrying the same value as they were in state load_d0. This ensures the data hold time for the Flash.

The process on lines 163-177 updates the state, address, and data registers on the rising clock edge. A reset from the parallel port will clear the data register and send the state machine to the load_a20 state to start another Flash address cycle. Note that the reset will not clear the address register. This allows the PC to read the Flash without writing it by forcing a reset after state load_a0. When the state machine returns to the load_a16 and load_a12 states, the PC can read the Flash data at the address that was loaded during the previous loop. This would not be possible if the address register was cleared by a reset.

The process on lines 181-188 updates the register that drives the Flash address lines. (The connection of this register to the Flash address lines is done on line 190.) The address lines change on the falling clock edge. This ensures the address lines are stable before any potential write operation is initiated on the next rising clock edge.

The Virtex-Flash Configuration Circuit

Listing 3 and Listing 4 show the VHDL code and pin assignments for the CPLD circuit that configures the Virtex FPGA with the bitstream programmed into the Flash. This circuit is simply increments an address counter which reads out the next byte of Flash data and strobes it into the Virtex. When the Virtex signals that it is completely configured, then the CPLD ceases operations. How the VHDL implements these functions is described below.

Lines 10-37 of Listing 3 define the interface for the circuit. It uses the programmable oscillator on the XSV Board as the main clock. The CPLD also interfaces to the address and control pins of the Flash chip so it can fetch the bytes of the Virtex configuration bitstream. (It doesn't need to access the Flash data pins since these are already directly connected to the configuration data inputs of the Virtex chip on the XSV Board.) The CPLD stuffs the bitstream into the Virtex using the configuration control pins.

Line 54 merely renames the V_dout pin of the Virtex to V_busy since the Virtex will use this signal to indicate when it is busy storing a byte of configuration data. Line 57 causes the CPLD to output the code onto the mode pins of the Virtex that place it in the SelectMAP configuration mode. In this mode, the Virtex chip accepts bytes of configuration data on the rising edge of the configuration clock as long as its chip-select and write-enable are active.

Lines 61-64 set the Flash control pins so it can output the data bytes of the Virtex bitstream. The CPLD releases its control of these pins when the Virtex signals that the configuration process is done (V_done=HI).

The Flash chip has an access time of 85 ns while the XSV Board oscillator can run as fast as 100 MHz. Lines 69-76 implement a counter that divides the oscillator frequency by 16 and uses the slower clock to drive the configuration of the Virtex.

After power is applied to the XSV Board, the Virtex FPGA needs some time to settle before configuration starts. Lines 80-90 create a power-on timeout counter and a reset signal that is active until the counter reaches zero. Then the reset is removed and the configuration starts. Line 8 of Listing 4 ensures that the timeout counter in the CPLD is initialized to the 11…1 state upon power-up of the XSV Board.

Lines 95-96 use the power-on reset to lower the PROGRAM pin of the Virtex when the board powers up. The PROGRAM signal goes high after the power-on timeout expires and the Virtex configuration starts.

Lines 100-107 select the Virtex chip for configuration when the PROGRAM pin is high and the Virtex is not indicating a configuration error by pulling its INIT pin low. The internal chip-select signal is inverted and drives the Virtex chip-select and write-enable pins on lines 112-113. The CPLD releases control of these pins when the configuration process is done.

The process on lines 120-129 controls the fetching of configuration data from the Flash. The Flash address register is set to zero while the Virtex is held in its reset state with the PROGRAM pin pulled low. After the PROGRAM pin goes high and configuration starts, the Flash address is incremented on every clock cycle as long as the Virtex chip is selected and the Virtex is not signaling a configuration error (INIT=HI) or that it is busy with a previous byte of configuration data (BUSY=LO). The value in the address counter is passed to the Flash chip address pins on line 133.

Listing 1: VHDL code for the Flash programming interface.

```
--
-- XC9500 CPLD design which controls the loading of the XSV Flash
-- with data from the PC parallel port.
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity flash is
	generic
	(
		ADDR_LEN: positive := 21		-- number of address bits for XSV FLASH
	);
	port
	(
		-- parallel port data and status pins
		ppd: in std_logic_vector(5 downto 0);	-- data nybble, clk, reset from par. port
		pps: out std_logic_vector(6 downto 3);	-- status nybble to parallel port

		-- Flash data, address, and control pins
		d: inout std_logic_vector(7 downto 0);		-- data bus to XSV FLASH
		a: out std_logic_vector(ADDR_LEN-1 downto 0);	-- address bus to XSV FLASH
		ceb: out std_logic;				-- chip-enable for XSV FLASH
		oeb: out std_logic;				-- output-enable for XSV FLASH
		web: out std_logic;				-- write-enable for XSV FLASH
		resetb: out std_logic;				-- reset for XSV FLASH

		-- Virtex FPGA pins
		V_progb: out std_logic				-- Virtex PROGRAM pin
	);
end flash;


architecture flash_arch of flash is

	constant LO : std_logic := '0';
	constant HI : std_logic := '1';
	constant NO : std_logic := '0';
	constant YES: std_logic := '1';

	-- states for the state machine that programs the Flash
	type flash_state_type is
	(
		load_a20,		-- load address nybble A23-A20
		load_a16,		-- load address nybble A19-A16, read data nybble D7-D4
		load_a12,		-- load address nybble A12-A15, read data nybble D3-D0
		load_a8,		-- load address nybble A8-A11
		load_a4,		-- load address nybble A4-A7
		load_a0,		-- load address nybble A0-A4
		load_d4,		-- load data nybble D4-D7
		load_d0			-- load data nybble D0-D3
	);

	signal flash_state, next_flash_state: flash_state_type;
	signal clk, reset: std_logic;
	signal nybble: std_logic_vector(3 downto 0);
	signal addr, next_addr: std_logic_vector(ADDR_LEN-1 downto 0);
	signal addr_reg, next_addr_reg: std_logic_vector(23 downto 0);
	signal data_reg, next_data_reg: std_logic_vector(3 downto 0);

begin
	V_progb<= LO;	-- keep Virtex in reset state so it doesn't interfere
```

```
64          resetb <= HI;       -- remove Flash reset so the chip is enabled
65          reset  <= ppd(0);   -- Flash prog. state machine reset from LSB of parallel port data
66          clk    <= ppd(1);   -- state machine clock from next bit of parallel port data
67          nybble <= ppd(5 downto 2); -- Flash data nybble from parallel port data
68
69          -- this process directs the state transitions of the Flash programming
70          -- state machine and sets the control outputs for each state
71          process(addr,addr_reg,d,data_reg,nybble,ppd,flash_state)
72          begin
73                -- the following statements set the default values for the outputs
74                oeb <= HI;    -- Flash chip data pin drivers disabled
75                ceb <= HI;    -- Flash chip disabled
76                web <= HI;    -- no write operations to Flash chip
77                d   <= (others=>'Z');            -- no data driven into the Flash chip
78                pps <= "1111";                   -- illegal state reported on status pins
79                next_addr <= addr;               -- Flash address does not change
80                next_addr_reg <= addr_reg;
81                next_data_reg <= data_reg;       -- Flash data does not change
82
83                -- now use the current state to determine the outputs and the
84                -- next state for the Flash programming state machine
85                case flash_state is
86
87                      when load_a20 =>
88                            -- load Flash address bits A23-A20 and output the
89                            -- last complete Flash address that was assembled previously
90                            d <= data_reg & nybble;    -- complete data byte written to Flash
91                            next_addr_reg(23 downto 20) <= nybble;  -- store A23-A20
92                            next_addr <= addr_reg(ADDR_LEN-1 downto 0);    -- output last addr
93                            pps <= "0000";      -- report current state through parallel port
94                            next_flash_state <= load_a16;     -- go to next state
95
96                      when load_a16 =>
97                            -- load Flash address bits A19-A16, read the contents
98                            -- from the previous Flash address, and send the upper
99                            -- nybble of the Flash data back through the parallel port
100                           next_addr_reg(19 downto 16) <= nybble;  -- store A19-A16
101                           ceb <= LO;                -- enable Flash
102                           oeb <= LO;                -- read Flash
103                           pps <= d(7 downto 4);     -- send upper data nybble back to PC
104                           next_flash_state <= load_a12;     -- go to next state
105
106                     when load_a12 =>
107                           -- load Flash address bits A15-A12, read the contents
108                           -- from the previous Flash address, and send the lower
109                           -- nybble of the Flash data back through the parallel port
110                           next_addr_reg(15 downto 12) <= nybble;  -- store A15-A12
111                           ceb <= LO;                -- enable Flash
112                           oeb <= LO;                -- read Flash
113                           pps <= d(3 downto 0);     -- send lower data nybble back to PC
114                           next_flash_state <= load_a8;      -- go to next state
115
116                     when load_a8  =>
117                           -- load Flash address bits A11-A8
118                           next_addr_reg(11 downto 8)  <= nybble;  -- store A11-A8
119                           pps <= "0011";      -- report current state through parallel port
120                           next_flash_state <= load_a4;      -- go to next state
121
122                     when load_a4  =>
123                           -- load Flash address bits A7-A4
124                           next_addr_reg(7 downto 4)   <= nybble;  -- store A7-A4
125                           pps <= "0100";      -- report current state through parallel port
126                           next_flash_state <= load_a0;      -- go to next state
127
128                     when load_a0  =>
```

```
129                         -- load Flash address bits A3-A0
130                         next_addr_reg(3 downto 0)   <= nybble;  -- store A3-A0
131                         pps <= "0101";      -- report current state through parallel port
132                         next_flash_state <= load_d4;      -- go to next state
133
134                   when load_d4  =>
135                         -- output the assembled address to the Flash and load the
136                         -- upper nybble of data that will be written to the Flash
137                         next_addr <= addr_reg(ADDR_LEN-1 downto 0); -- output complete addr
138                         ceb <= LO;    -- enable the Flash
139                         next_data_reg <= nybble;   -- store upper data nybble from par port
140                         d <= data_reg & nybble;    -- output data to the Flash
141                         pps <= "0110";      -- report current state through parallel port
142                         next_flash_state <= load_d0;      -- go to the next state
143
144                   when load_d0  =>
145                         -- now get the lower nybble of data from the parallel port
146                         -- and write the complete byte to the Flash during the
147                         -- second half of the clock phase
148                         ceb <= LO;    -- keep the Flash enabled
149                         web <= clk;  -- write goes low during second half of clock cycle
150                         d <= data_reg & nybble;    -- complete data byte written to Flash
151                         pps <= "0111";      -- report current state through parallel port
152                         next_flash_state <= load_a20;     -- go back to the start
153
154                   when others =>
155                         -- return the state machine to the initial state if it
156                         -- ever gets into an erroneous state
157                         next_flash_state <= load_a20;
158
159             end case;
160       end process;
161
162       -- update the programming machine state and other registers
163       process(reset,clk)
164       begin
165             if (reset=HI) then
166                   -- asynchronous reset sets state machine to initial state
167                   -- and clears data register
168                   flash_state <= load_a20;
169                   data_reg <= (others=>'0');
170
171             elsif (clk'event and clk=HI) then
172                   -- update the machine state and other registers on rising clock edge
173                   flash_state <= next_flash_state;
174                   addr_reg <= next_addr_reg;
175                   data_reg <= next_data_reg;
176             end if;
177       end process;
178
179       -- output Flash addresses one-half cycle early.  This gives the Flash
180       -- address time to settle and activate the appropriate location for writing.
181       process(reset,clk)
182       begin
183             -- change Flash address during the second half of the clock cycle
184             -- before the machine changes states
185             if (clk'event and clk=LO) then
186                   addr <= next_addr;
187             end if;
188       end process;
189
190       a <= addr;    -- output address to the Flash chip
191
192 end flash_arch;
```

Listing 2: Pin assignments for the Flash programming interface.

```
#
# pin assignments for the XC95108 CPLD chip on the XSV Board
#

# Virtex FPGA
net V_progb         loc=p11;

# Flash RAM
net resetb          loc=p3;
net ceb             loc=P46;
net oeb             loc=p42;
net web             loc=p43;
net d<0>            loc=p32;
net d<1>            loc=p33;
net d<2>            loc=p34;
net d<3>            loc=P35;
net d<4>            loc=P36;
net d<5>            loc=P37;
net d<6>            loc=P39;
net d<7>            loc=P40;
net a<0>            loc=p16;
net a<1>            loc=p17;
net a<2>            loc=p18;
net a<3>            loc=p19;
net a<4>            loc=p20;
net a<5>            loc=p23;
net a<6>            loc=p24;
net a<7>            loc=p25;
net a<8>            loc=p27;
net a<9>            loc=p28;
net a<10>           loc=p29;
net a<11>           loc=p30;
net a<12>           loc=p49;
net a<13>           loc=p50;
net a<14>           loc=p52;
net a<15>           loc=p53;
net a<16>           loc=p54;
net a<17>           loc=p55;
net a<18>           loc=p56;
net a<19>           loc=p58;
net a<20>           loc=p59;

# parallel port
net ppd<0>          loc=p77;
net ppd<1>          loc=p74;
net ppd<2>          loc=p72;
net ppd<3>          loc=p70;
net ppd<4>          loc=p68;
net ppd<5>          loc=p67;
# net ppd<6>        loc=p66;
# net ppd<7>        loc=p65;
# net ppc<0>        loc=p79;
# net ppc<1>        loc=p78;
# net ppc<2>        loc=p73;
# net ppc<3>        loc=p71;
net pps<3>          loc=p76;
net pps<4>          loc=p60;
net pps<5>          loc=p61;
net pps<6>          loc=p64;
# net pps<7>        loc=p63;
```

Listing 3: VHDL code for the Virtex-Flash configuration circuit.

```
--
-- XC9500 CPLD design which controls the configuration of the XSV Virtex
-- with data from the Flash chip.
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity config is
	generic
	(
		ADDR_LEN: positive := 21		-- number of Flash address bits
	);
	port
	(
		clk		: in std_logic;		-- clock from DS1075 prog. osc.

		-- Flash address and control pins
		a		: out std_logic_vector(ADDR_LEN-1 downto 0); -- Flash address
		ceb		: out std_logic;	-- Flash chip-enable
		oeb		: out std_logic;	-- Flash output-enable
		web		: out std_logic;	-- Flash write-enable
		resetb		: out std_logic;	-- Flash reset

		-- Virtex configuration pins
		V_progb		: out std_logic;	-- Virtex PROGRAM pin
		V_cclk		: out std_logic;	-- Virtex config clock
		V_csb		: out std_logic;	-- Virtex config chip-select
		V_wrb		: out std_logic;	-- Virtex config write-enable
		V_initb		: in std_logic;		-- Virtex config init status
		V_dout		: in std_logic;		-- Virtex config busy status
		V_done		: in std_logic;		-- Virtex config done status
		V_m		: out std_logic_vector(2 downto 0)	-- Virtex config. mode pins

	);
end config;

architecture config_arch of config is
	constant LO		: std_logic := '0';
	constant HI		: std_logic := '1';
	constant FLOAT		: std_logic := 'Z';

	signal clk_cnt			: std_logic_vector(3 downto 0);
	signal cclk			: std_logic;
	signal programb, cs		: std_logic;
	signal addr, next_addr		: std_logic_vector(ADDR_LEN-1 downto 0);
	signal poweron_reset		: std_logic;
	signal poweron_cnt		: std_logic_vector(19 downto 0);
	signal V_busy			: std_logic;
	signal button_progb		: std_logic;
begin

	V_busy <= V_dout;		-- give this signal a better name

	-- set Virtex mode to SelectMAP so it can be configured from Flash
	V_m		<= "110";

	-- Flash is enabled for reading while Virtex is not yet configured
	-- and then the Flash pins float when configuration is done
	oeb		<= LO when (V_done=LO) else FLOAT;
	ceb		<= LO when (V_done=LO) else FLOAT;
```

```
63          web          <= HI when (V_done=LO) else FLOAT;      -- disable Flash writes
64          resetb       <= HI;                                  -- remove Flash reset
65
66          -- generate configuration clock for Virtex from the XSV clock.
67          -- The XSV clock could be as much as 100 MHz, so divide by 16
68          -- to exceed the access time of the Flash.
69          process(clk,clk_cnt)
70          begin
71                if(clk'event and clk=HI) then
72                      clk_cnt <= clk_cnt + 1;
73                end if;
74          end process;
75          cclk   <= clk_cnt(3);      -- internal configuration clock
76          V_cclk <= cclk;            -- also send config. clock to Virtex
77
78          -- Apply reset when the power to the XSV Board is first applied.
79          -- Remove the power-on reset after the counter reaches 0.
80          process(poweron_cnt,cclk)
81          begin
82                if(cclk'event and cclk=HI) then
83                      if(poweron_cnt = 0) then
84                            poweron_reset <= LO;       -- remove reset when timeout expires
85                      else
86                            poweron_cnt <= poweron_cnt - 1;
87                            poweron_reset <= HI;
88                      end if;
89                end if;
90          end process;
91
92          -- initiate Virtex configuration by lowering the /PROGRAM pin
93          -- during the initial power-on reset and then raising it when
94          -- the power-on timeout expires and the manual program control is high
95          programb <= not(poweron_reset);
96          V_progb <= programb;
97
98          -- Select the Virtex for configuration as long as the /PROGRAM pin
99          -- is not held low and the INIT pin is not low.
100         process(V_initb,cclk,programb)
101         begin
102               if(programb = LO) then
103                     cs <= LO;
104               elsif(cclk'event and cclk=HI) then
105                     cs <= V_initb;
106               end if;
107         end process;
108
109         -- Select the Virtex for configuration by lowering its chip-select
110         -- and write inputs when the internal chip-select is high.  Then
111         -- float these pins after the Virtex configuration is done.
112         V_csb <= not(cs)    when (V_done=LO) else FLOAT;
113         V_wrb <= not(cs)    when (V_done=LO) else FLOAT;
114
115         -- increment the Flash address so the next byte of configuration
116         -- data is presented to the Virtex.  Stop incrementing if the
117         -- Virtex is not selected, signals a config. error (INIT=0), or
118         -- is busy.  Reset the address counter to zero whenever the
119         -- /PROGRAM pin goes low and a new configuration sequence begins.
120         process(addr,cs,V_initb,V_busy,cclk)
121         begin
122               if(cclk'event and cclk=HI) then
123                     if((cs=HI) and (V_initb=HI) and (V_busy=LO)) then
124                           addr <= addr + 1;
125                     elsif(programb = LO) then
126                           addr <= (others=>LO);
127                     end if;
```

```
128             end if;
129       end process;
130
131       -- pass the Flash address out to the Flash chip.  Float the address
132       -- lines once configuration is done.
133       a <= addr                 when (V_done=LO) else (others=>FLOAT);
134
135 end config_arch;
```

Listing 4: Pin assignments for the Virtex-Flash configuration circuit.

```
#
# pin assignments for the XC95108 CPLD chip on the XSV Board
#

# set all the bits in the initial state of the power-on
# counter so we get the maximum timeout interval
inst poweron_cnt_reg<*> INIT=S;

# Virtex FPGA
net V_dout          loc=p6;
net V_wrb           loc=p7;
net V_csb           loc=p8;
net V_initb         loc=p9;
net V_done          loc=p10;
net V_progb         loc=p11;
net V_cclk          loc=p12;
net V_m<0>          loc=p13;
net V_m<1>          loc=p14;
net V_m<2>          loc=p15;

# Flash RAM
net resetb          loc=p3;
net ceb             loc=P46;
net oeb             loc=p42;
net web             loc=p43;
net a<0>            loc=p16;
net a<1>            loc=p17;
net a<2>            loc=p18;
net a<3>            loc=p19;
net a<4>            loc=p20;
net a<5>            loc=p23;
net a<6>            loc=p24;
net a<7>            loc=p25;
net a<8>            loc=p27;
net a<9>            loc=p28;
net a<10>           loc=p29;
net a<11>           loc=p30;
net a<12>           loc=p49;
net a<13>           loc=p50;
net a<14>           loc=p52;
net a<15>           loc=p53;
net a<16>           loc=p54;
net a<17>           loc=p55;
net a<18>           loc=p56;
net a<19>           loc=p58;
net a<20>           loc=p59;

# programmable oscillator
net clk             loc=p22;
```