4Mbit Virtual SPROM

XAPP079 September, 1997 (Version 1.2) Application Note

Summary

This application note describes the design of a very low cost, CPLD-based virtual SPROM for downloading programming information to the Xilinx high density XC4000-Series FPGAs.

Xilinx Family

XC9500

Introduction

Typically, designers of embedded applications use serial PROMs (SPROMs) to store and download the configuration data for XC4000-Series FPGAs. SPROMs yield faster configuration rates and reduce the amount of circuitry required to program FPGAs. As FPGA densities continue to grow, so do their configuration memory requirements; the SPROM program memory requirements for a high density XC4000-Series FPGA can be 512K bits or more. The virtual serial PROM (VSPROM) solution described in this application note uses a low cost XC9536 CPLD, a standard byte-wide EPROM, and an on-board crystal oscillator to support embedded programming of high density XC4000-Series FPGAs.

Design Description

Figure 2 shows the schematic of the circuitry used to configure an XC4025E FPGA using the VSPROM design. The XC4025E requires 422,128 configuration bits, which are supplied by U4, a 64Kx8 (512K bits) UV EPROM. U1 is an XC9536-15PC44 CPLD, used to read the data from the EPROM and download it to the FPGA. An on-board crystal oscillator clocks the VSPROM design and supplies configuration clocks (**CCLKs**) to the FPGA. The FPGA is connected in **SLAVE SERIAL MODE** (see the Xilinx Data Book for information on XC4000-Series configuration modes). You can use this reference design as-is, or you can modify the design as required.

The XC9536 acts as an intelligent state machine monitoring the FPGA's **INIT** and **DONE** pins while pumping serial configuration data into the FPGA's **DIN** pin. Figure 1 shows the state diagram. The configuration process is started on the falling edge of the FPGA **INIT** pin. Data is continually read and shifted into the FPGA until the rising edge of the **DONE** pin. If **INIT** goes low during configuration, the XC9536 will pulse the FPGA's **/PROGRAM** pin low, reset the state machine, and consequently restart the configuration process.

Figure 3 shows the ABEL code describing the VSPROM design. Bits **d7** through **d0** are data bus interface pins to the EPROM. Data registers **data7** through **data0** are internal nodes used to latch and shift the incoming configuration data from the EPROM. The data registers are broken into two busses of four bits each: **busA** and **busB**. State **SHIFT_A** shifts **busA** while loading EPROM data into **busB**. State **SHIFT_B** shifts **busB** while loading EPROM data into **busA**. This provides a continual stream of data into the FPGA with no interruption in **CCLK**.

The state machine enters the **LASTCLKS** state on the rising edge of the **DONE** pin to provide the last few **CCLKs** to the FPGA. This is required to bring the FPGA out of configuration and enable its I/O. Finally, the state machine enters the **DONE** state to complete the process.

CCLK from the XC9536 to the FPGA is output-enabled to ensure that the FPGA doesn't receive stray clocks while the VSPROM state machine is in the **DONE** state. Since the **CCLK** output goes into a high impedance condition while in the **DONE** state, resistor **R6** is tied to ground to ensure that **CCLK** is always in a known state.

Design Implementation

This VSPROM design is fully verified with both functional and timing vectors. Figure 4 shows the timing simulation waveforms created with Viewlogic's ViewSim™. Configuration rates up to 10MHz were successfully simulated. The XC9536 CPLD was programmed with a Xilinx HW-130 V4.0 programmer, and the EPROM was programmed with a generic EPROM programmer. The XC9536 is an IEEE 1149.1 compliant ISP CPLD, and therefore could be programmed in-system via the JTAG port, TDI, TCK, TMS, and TDO.

To verify the design, XACTstep was used to implement a simple Johnson Counter in the XC4025E FPGA. The design was placed and routed using default settings. The PROM File Formatter utility in XACTstep produced a byte wide 64Kx8 (512K bits) MCS EPROM file using default settings for starting address and loading direction. The design was assembled on a generic prototyping board and configuration rates up to 5 MHz were verified. The XC4025E FPGA used in this application note requires 422,128 program bits. Therefore, only address lines 0 through 15 were required. A total of 19 address lines (4Mbits addressable space) are available for different configurations.

Table 1 shows the EPROM and address line requirements for each XC4000-Series device that requres more than 256Kbits of SPROM memory (see Xilinx Data Book, June 16, 1997 for more information). Use the Xilinx PROM File Formatter in XACTstep to determine the approximate EPROM size for daisy chains. If required, connect up to 3 additional address lines. Finally, the XC9536 may be used in mixed voltage systems (ie 5v/3.3v) by connecting it's Vccio pin to the appropriate voltage source. See the XC9500 Data Sheets for more information.

Conclusion

This Virtual SPROM application note provides an ultra low cost solution to support embedded programming of high density XC4000-Series FPGAs.

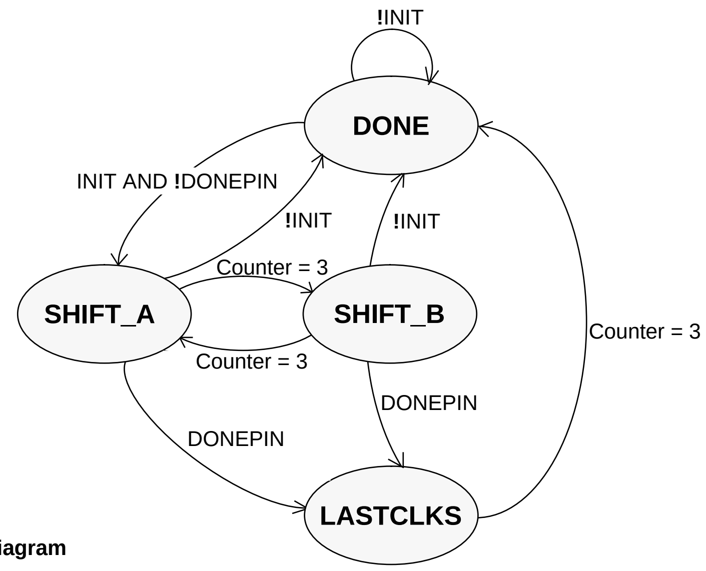

Figure 1: State Diagram

FPGA Target Device	Req. Program Bits	Req. EPROM Bits	CPLD Requirements
XC4010EX/XL	283,376	283,424	XC9536 (using design as shown)
XC4013EX/XL	393,580	393,632	XC9536 (using design as shown)
XC4020E	329,264	329,312	XC9536 (using design as shown)
XC4020EX/XL	521,832	521,880	XC9536 (connect address lines 0 thru 16 to standard 1M EPROM)
XC4025E	422,128	422,176	XC9536 (using design as shown)
XC4028EX/XL	668,132	668,184	XC9536 (connect address lines 0 thru 16 to standard 1M EPROM)
XC4036EX/XL	832,480	832,528	XC9536 (connect address lines 0 thru 16 to standard 1M EPROM)
XC4044EX/XL	1,014,876	1,014,928	XC9536 (connect address lines 0 thru 16 to standard 1M EPROM)
XC4052EX/XL	1,215,320	1,215,368	XC9536 (connect address lines 0 thru 17 to standard 2M EPROM)
XC4062EX/XL	1,433,812	1,433,864	XC9536 (connect address lines 0 thru 17 to standard 2M EPROM)
XC4085EX/XL	1,924,940	1,924,992	XC9536 (connect address lines 0 thru 17 to standard 2M EPROM)

Table 1: EPROM and CPLD Requirements

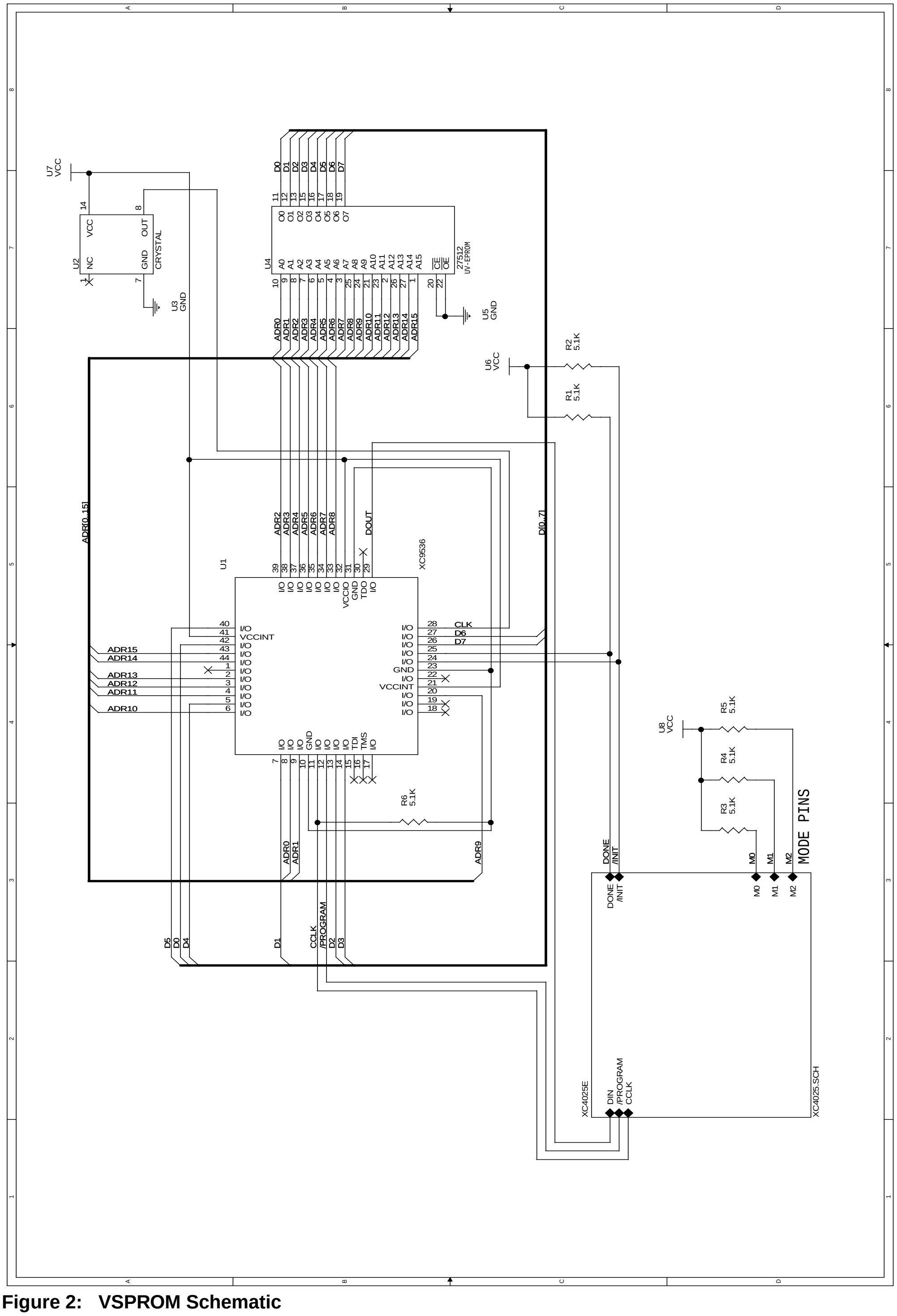

Figure 2: VSPROM Schematic

```
MODULE VSPROM

TITLE ‘ Virtual 4Mbit SPROM to configure XC4K FPGAs.
VERSION: 1.2

DATE: 9/97’

“inputs

init, donepin, clk           pin 24,25,28;
d7..d0                       pin 26,27,40,5,14,13,7,42;

“outputs

adr18..adr0, program         pin 18,1,22,43,44,2,3,4,6,20,33,34,35,36,37,38,39,9,8,12 istype ‘reg’;

cclk                         pin 11;
dout                         pin 29;

“nodes

data7..data0, count1, count0,
s1, s0                       node istype ‘reg’;
outen, reset                 node istype ‘com, keep’;

declarations

counter = [count1..count0];
address = [adr18..adr0];
busA = [data3..data0];
busB = [data7..data4];
inA = [d3..d0];
inB = [d7..d4];
inAB = [d7..d0];

XEPLD PROPERTY ‘LOGIC_OPT OFF outen, reset’;
XEPLD PROPERTY ‘PWR LOW program’;
Z, C, X = .Z., .C., .X.;
@DCSET;

“State values
DONE =^b00; SHIFT_A =^b11; SHIFT_B =^b10; LASTCLKS =^b01;

VSPROM = [s1..s0];

equations

VSPROM.clk = clk;
address.clk = clk;
counter.clk = clk;
busA.clk = clk;
busB.clk = clk;
program.clk = clk;
program.ar = !init & s1;

cclk = !clk;
cclk.oe = outen;

counter := !reset & (counter + 1);

 when reset then address := 0;
     else when ((VSPROM == SHIFT_A) & (counter == 3) & !reset)
            then address := address + 1; else address := address;

 when (VSPROM == DONE) then reset = 1;
     else
 when (((VSPROM == SHIFT_A) # (VSPROM == SHIFT_B)) & donepin) then reset = 1;
     else reset = 0;

 when (VSPROM == DONE) then
     {outen = 0;
     busA := inA;
     busB := inB;
     program := 1;
     dout = data0;}
```

Figure 3: ABEL Code for VSPROM

```
     when (VSPROM == SHIFT_A) then
            {outen = 1;
            program := 1;
            busB := inB; data0 := data1; data1 := data2;
            data2 := data3; dout = data0;}
 else
     when (VSPROM == SHIFT_B) then
            {outen = 1;
            program := 1;
            busA := inA; data4 := data5; data5 := data6;
            data6 := data7; dout = data4;}
 else
     when ((VSPROM == SHIFT_A) & (counter == 3)) then dout = data0;
 else
     when ((VSPROM == SHIFT_B) & (counter == 3)) then dout = data4;
 else
     when (VSPROM == LASTCLKS) then
            {program := 1;
            outen = 1;}


state_diagram VSPROM;
state DONE:
if (!init) then DONE;
   else if (!donepin) then SHIFT_A;
     else DONE;
state SHIFT_A:
if (!init) then DONE;
   else if (donepin) then LASTCLKS;
     else if ((VSPROM == SHIFT_A) & (counter == 3)) then SHIFT_B;
       else SHIFT_A;
state SHIFT_B:
if (!init) then DONE;
   else if (donepin) then LASTCLKS;
     else if ((VSPROM == SHIFT_B) & (counter == 3)) then SHIFT_A;
       else SHIFT_B;
state LASTCLKS:
if ((VSPROM == LASTCLKS) & (counter == 3)) then DONE;
   else LASTCLKS;
end;
```

Figure 3 - Continued

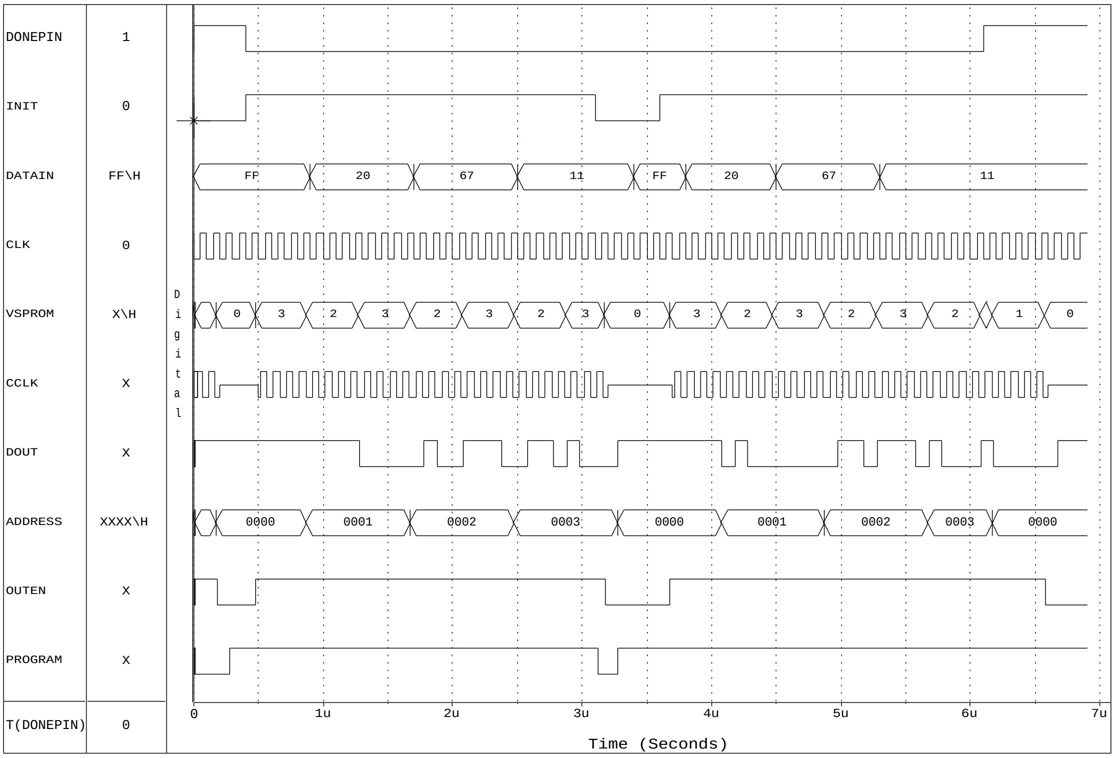

Figure 4: Simulation Waveforms