

Summary

This application note describes the design of a very low cost, CPLD-based virtual SPROM downloader for programming the Xilinx high density XC4000-Series FPGAs in embedded applications.

Xilinx Family

XC7300, 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 XC7336 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 XC7336-15PC44 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 XC7336 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 **DONE** 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 XC7336 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, implemented with XABEL-CPLD. 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 XC7336 is output enabled to ensure that the FPGA does not get stray clocks while the 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 were successfully simulated up to 10 MHz. The XC7336 CPLD was programmed with a Xilinx HW-130 V4.0 programmer, and the EPROM was programmed with a generic EPROM programmer.

To verify the design, XACT-STEP v6.0.1 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 XACT-STEP 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.

Designs Using More Than 1 Mbit of EPROM

The VSPROM design can be used as shown (using an XC7336 CPLD) for FPGAs up to the XC4044XL. This

design can easily be expanded to 4 Mbits by removing the pin declarations in the ABEL file, adding two address lines, and targeting an XC9536-15PC44 CPLD. The pin declarations must be removed because the design's existing pinout is not compatible with the XC9536PC44. The fitter will automatically assign pinouts to achieve the best performance.

Table 1 shows the VSPROM requirements for all XC4000-Series devices. To determine the approximate EPROM size

for daisy chains, use the Xilinx PROM File Formatter in XACTstep.

Conclusion

This Virtual SPROM application note provides an ultra low cost solution to support embedded programming of high density XC4000-Series FPGAs. The XC7336 CPLD can be used for FPGAs up to the XC4044XL and the XC9536 can be used for FPGAs requiring more than 1 Mbit of EPROM.

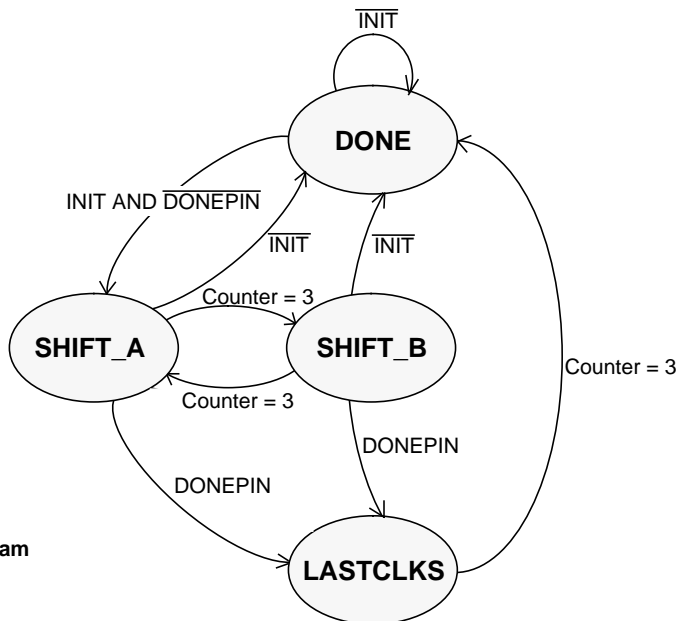
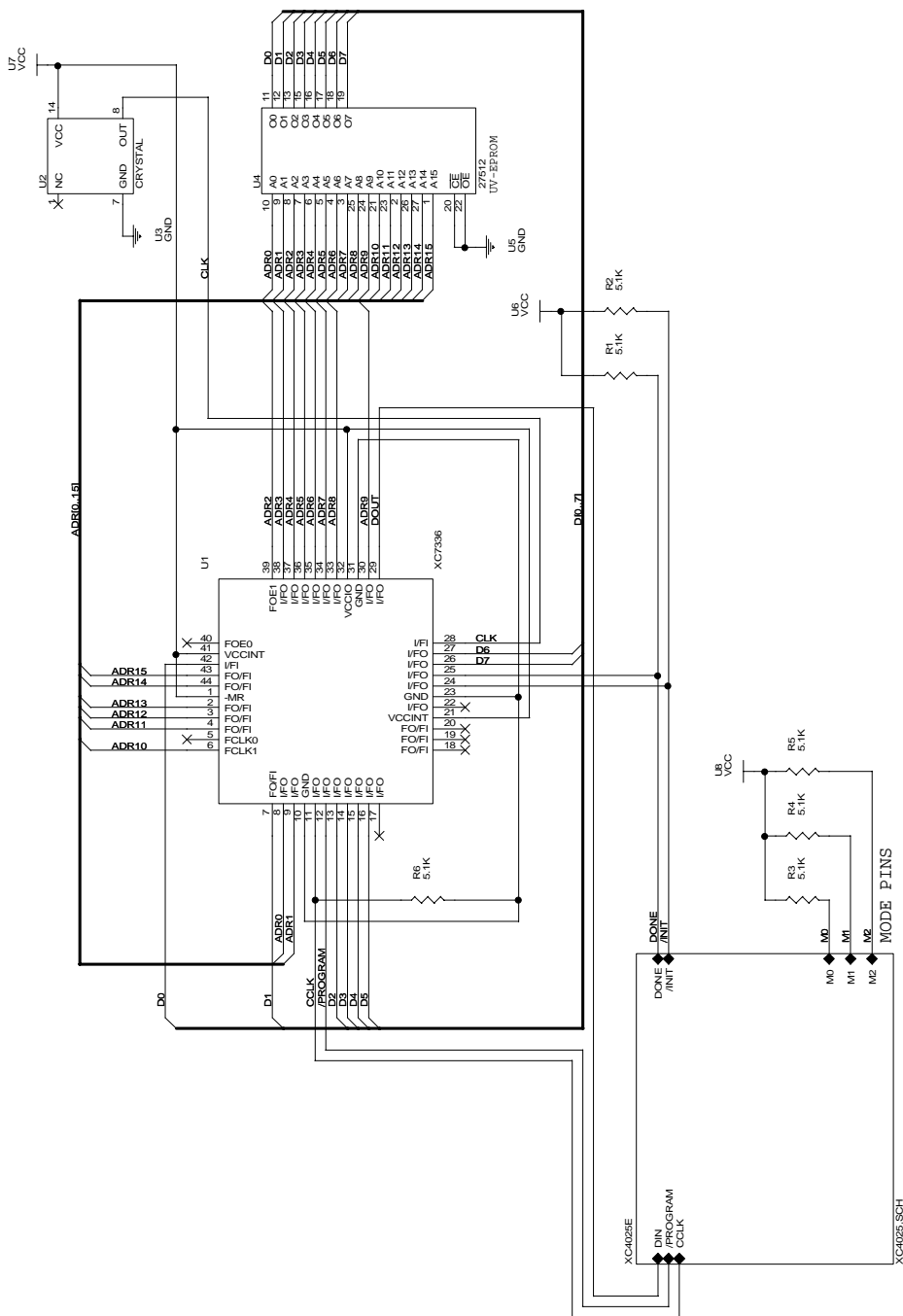


Figure 1: State Diagram

FPGA Target Device	Req. Program Bits	Req. EPROM Bits	CPLD Requirements
XC4003E	53,936	53,984	XC7336 (using design as shown)
XC4005E/L	94,960	95,008	XC7336 (using design as shown)
XC4006E	119,792	119,840	XC7336 (using design as shown)
XC4008E	147,504	147,552	XC7336 (using design as shown)
XC4010E/L	178,096	178,144	XC7336 (using design as shown)
XC4013E/L	247,920	247,968	XC7336 (using design as shown)
XC4020E	329,264	329,312	XC7336 (using design as shown)
XC4025E	422,128	422,176	XC7336 (using design as shown)
XC4028EX/XL	668,132	668,180	XC7336 (using design as shown)
XC4036EX/XL	832,480	832,528	XC7336 (using design as shown)
XC4044XL	1,014,876	1,014,924	XC7336 (using design as shown)
XC4052XL	1,215,320	1,215,368	XC9536 (add 1 additional address line)
XC4062XL	1,433,812	1,433,860	XC9536 (add 1 additional address line)

Table 1: EPROM and CPLD Requirements for Each FPGA

Figure 2: VSPROM Schematic



```
MODULE VSPROM

TITLE ` Virtual 1Mbit SPROM to configure XC4K FPGAs.
DATE: 2/97'

`inputs
init, donepin, clk          pin 24,25,28;
d7..d0                      pin 26,27,16,15,14,13,7,42;

`outputs
adr16..adr0, program        pin 22,43,44,2,3,4,6,30,33,34,35,36,37,38,39,9,8,12  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 = [adr16..adr0];
busA = [data3..data0];
busB = [data7..data4];
inA = [d3..d0];
inB = [d7..d4];
inAB = [d7..d0];

XEPD PROPERTY `LOGIC_OPT OFF outen, reset';
XEPD 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;}

else
```

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

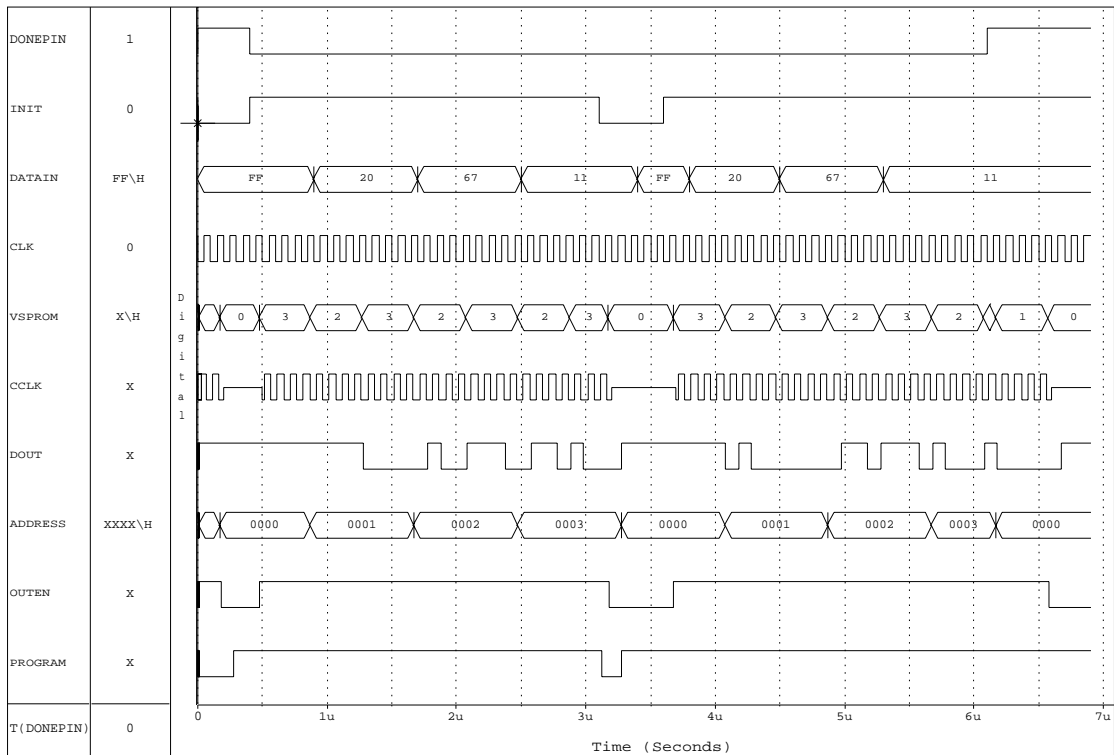


Figure 4: Simulation Waveforms