

079



---

XAPP079 September, 1997 (Version 1.2)

# 4Mbit Virtual SPROM

4MBIT VIRTUAL SPROM

# 이용 안내서

---

**PLDWorld.com**

- Inner Secret Devices -

<http://www.pldworld.com/>


---

## 4Mbit Virtual Serial PROM

### 요약:

본 응용서는 Xilinx의 대용량 XC4000 시리즈 FPGA에 프로그램 정보 (Bitstream)를 다운로드하기 위해 매우 저렴한 비용으로 할 수 있는 설계에 대해서 기술하고 있는데 이는 Xilinx XC9500 CPLD에 기반한 Virtual SPROM이다.

아이콘 키

 유용한 정보

### 개요

일반적으로 Embedded Application을 설계하는 엔지니어들은 XC4000 시리즈 FPGA에 Configuration Data를 다운로드하고 저장할 목적으로 Serial PROM (SPROM)을 사용한다. SPROM은 좀더 빠른 Configuration 속도를 제공하고 FPGA를 프로그램 하는데 요구되는 회로소자의 개수를 줄여준다. FPGA의 용량이 점차적으로 커짐에 따라 그것들을 Configuration할 메모리 요구량도 또한 커졌다; 대용량의 XC4000 시리즈 FPGA를 위한 SPROM의 프로그램 메모리 요구량은 512K bit이거나 그 이상이 될 수도 있다. 본 응용서에서 설명되는 Virtual Serial PROM (VSPROM)은 매우 저렴한 XC9536 CPLD와 일반적인 byte-wide EPROM, 그리고 대용량 XC4000 시리즈 FPGA의 Embedded 프로그래밍을 지원하기 위해 사용되는 보드상의 크리스탈 오실레이터를 사용한다.

### 회로 설명

그림 2는 VSPROM 디자인을 사용하여 XC4025E FPGA를 Configuration하기 위한 회로도이다. XC4025E는 422,128 Configuration Bit를 요구하는데 이는 64Kx8 (512K bits) 용량의 UV EPROM인 U4에 의해서 공급된다. U1은 EPROM에 들어있는 데이터를 읽어서 FPGA로 내용을 다운로드 하는데 사용되는 XC9536-15PC44C CPLD이다. 보드상의 크리스탈 오실레이터는 VSPROM 디자인과 FPGA의 Configuration Clock (CCLKs)을 공급한다. FPGA는 **SLAVE SERIAL MODE**로 연결되어 있다. (XC4000 시리즈의 Configuration 모드에 대한 정보는 Xilinx Data Book을 참조할 것.) 사용자는



본 참조 디자인을 그대로 사용하거나, 또는 요구되어지는 사양으로 변경하여 사용할 수 있다.

XC9536 은 FPGA 의 **DIN** 핀으로 Configuration Data 를 직렬로 보내는 동안 FPGA 의 **INIT** 와 **DONE** 핀을 지능적으로 감시하는 State Machine 으로서 행동한다. 그림 1 에 보여지는 것은 바로 그 상태도이다. Configuration 과정은 FPGA 의 INIT 핀의 Falling Edge 로 시작된다. Data 는 연속적으로 읽혀져서 **DONE** 핀이 Rising Edge 가 될 때까지 FPGA 로 이동한다. 만약 **INIT** 가 Configuration 동안 low 를 유지한다면, XC9536 은 FPGA 의 **/PROGRAM** 핀에 low 펄스를 가하고 State Machine 을 리셋할 것이고, 그 결과로서 Configuration 과정을 재 시작한다.

그림 3 은 VSPROM 디자인의 ABEL 코드이다. 비트 **d7** 부터 **d0** 는 EPROM 과의 데이터 버스 인터페이스 핀이다. Data Register **data7** 부터 **data0** 는 EPROM 으로부터 받아들이는 Configuration 데이터를 전달하거나 래치하는데 사용되는 내부 노드이다. Data Register 는 각 4 비트를 갖는 두개의 버스로 나뉘어 진다: **busA** 와 **busB**. **SHIFT\_A** 상태는 **busB** 로 EPROM 데이터를 로딩하는 동안 **busA** 로 천이 된다. **SHIFT\_B** 상태는 **busA** 로 EPROM 데이터를 로딩하는 동안 **busB** 로 천이 된다. 이는 CCLK 의 중단없이 FPGA 내로 데이터의 연속적인 흐름을 제공한다.

이 State Machine 은 FPGA 로 마지막 몇 개의 **CCLKs** 가 제공되도록 **DONE** 핀의 Rising Edge 일 때 **LASTCLKS** 상태가 입력된다. 이것은 FPGA 에게 Configuration 완료와 I/O 의 활성화를 유지하도록 요구되어진다. 마지막으로 State Machine 은 Configuration 과정이 완전하다고 **DONE** State 에 입력한다.

XC9536 에서 FPGA 로 향하는 **CCLK** 는 VSPROM State Machine 이 **DONE** 상태인 동안 FPGA 에 잘못된 Clock 이 도달하지 않도록 보장하기 위해 출력 가능한 상태가 된다. **CCLK** 출력이 **DONE** State 동안 High Impedance 상태가 된 이후, 저항 **R6** 은 GND 로 묶인다.

## 회로 제작

본 VSPROM 디자인은 기능적으로 또 Timing Vector 로 완벽하게 검증되었다. 그림 4 는 Viewlogic 사의 ViewSim 으로 작성된 Timing Simulation 파형이다. Configuration 속도는 10MHz 까지 성공적으로 Simulation 되었다. XC9536 CPLD 는 Xilinx HW-130 v4.0 Programmer 로 프로그램 하였고, EPROM 은 일반적인 EPROM Programmer 로 프로그램 하였다. XC9536 은 IEEE 1149.1 을 준수하는 ISP CPLD 이므로, JTAG 포트 (TDI, TCK, TMS, TDO)로 직접 시스템상에서 프로그램이 가능하다.

회로 검증을 위해 XACTstep M1 을 사용하여 XC4025E FPGA 에 간단한 Johnson Counter 를 구성하였다. 회로는 기본 설정치로 Place & Route 를 하였다. XACTstep M1 내에 있는 PROM File Formatter 로 시작번지와 Loading 방향을 기본 설정치로 하여 byte wide 64Kx8 (512K bits)인 MCS EPROM 파일을 만들었다. 본 회로는 일반적인 표준보드상에 조립을 하였고 Configuration 속도는 5MHz 까지 검증되었다. 본 응용

서에서 사용한 XC4025E FPGA 는 422,128 Program bit 를 필요로 한다. 그러므로 어드레스 라인은 0 ~ 15 까지만 사용한다. 전체 19 개의 어드레스 라인 (4Mbit 까지 지정 가능한 영역)은 다른 Configuration 시 사용 가능하다.

표 1 은 EPROM 과 256Kbit 의 SPROM 보다 큰 용량을 요구하는 각각의 XC4000 시리즈 FPGA 에서 필요한 어드레스 라인을 나타낸다. (좀더 자세한 정보는 Xilinx Data Book 을 참조할것) Daisy Chain 을 위한 대략적인 EPROM 용량의 산출은 XACTstep 내의 Xilinx PROM File Formatter 를 사용한다. 만약 Daisy Chain 을 요구한다면 3 개까지의 추가적인 어드레스 라인을 연결한다. 마지막으로 XC9536 은 필요한 전압을 Vccio 핀에 연결하면 Mixed Voltage System 내 (예: 5v/3.3v)에서 사용될 수도 있다. 좀더 자세한 정보는 XC9500 Data sheet 를 참조할것.

## 결론

이 Virtual SPROM 응용서는 고용량 XC4000 시리즈 FPGA 의 Embedded Programming 을 지원하는 초 저비용 솔루션을 제공한다.

## 회로도 및 관련자료

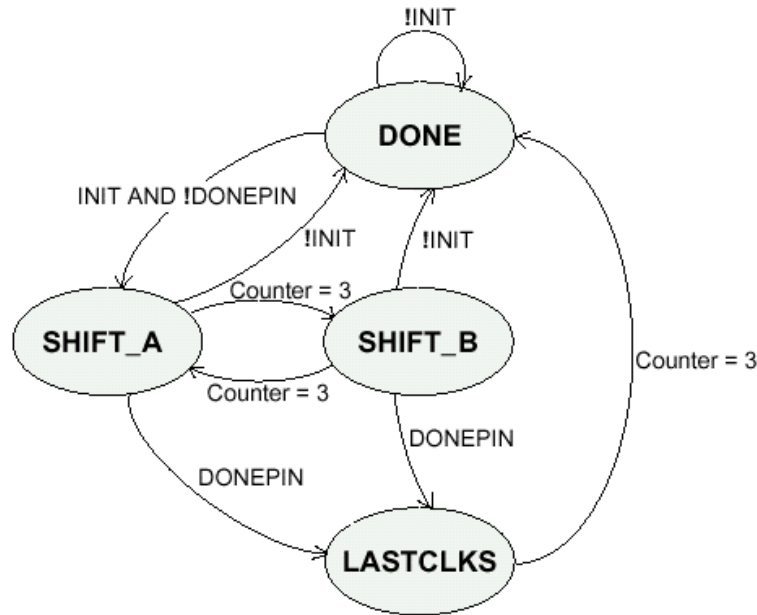


그림 1 State Diagram

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)

표 1 EPROM and CPLD Requirements

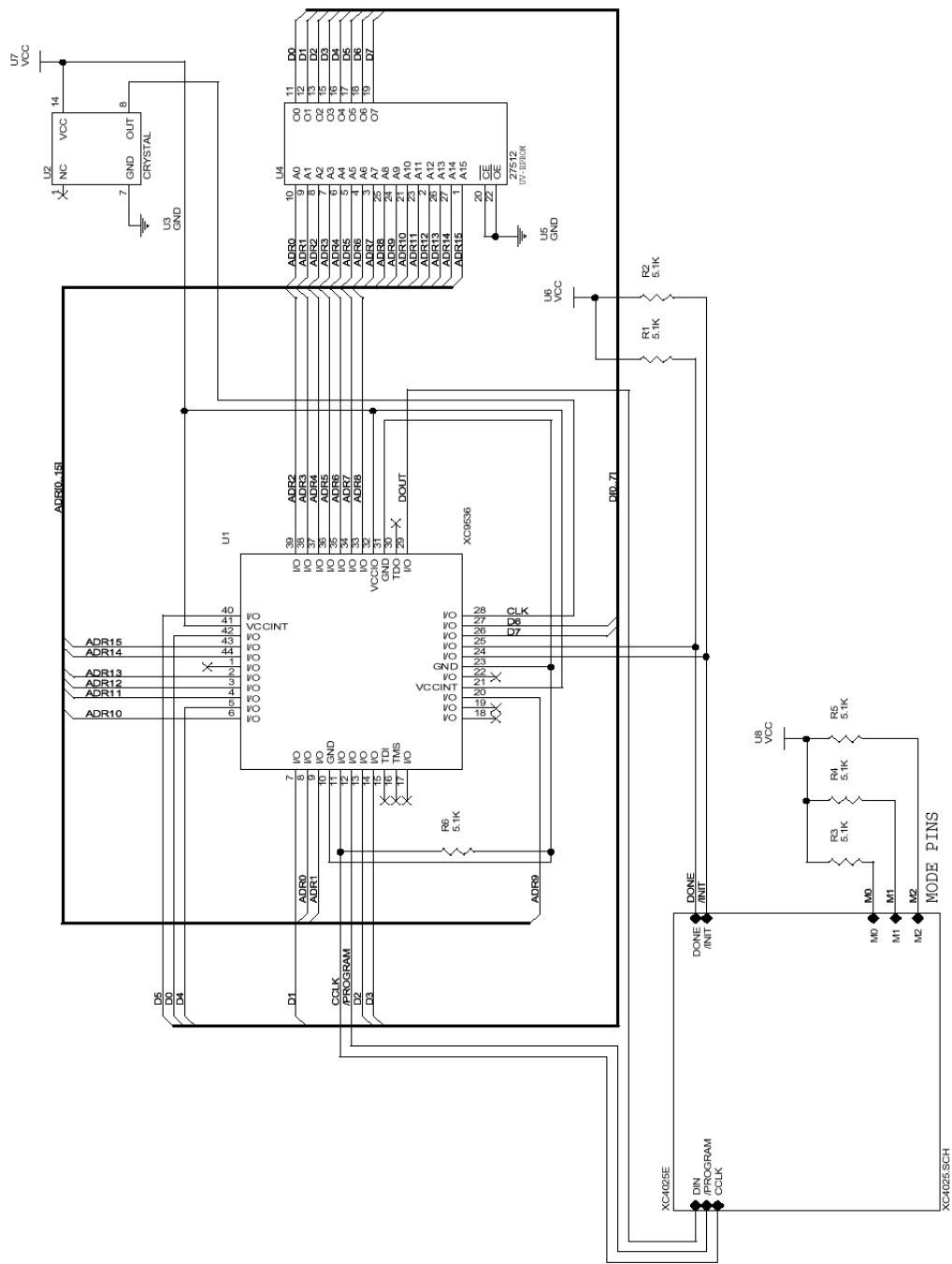


그림 2 VSPROM Schematic

HERE IS A "PLDWORLD.COM" ...

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;



HERE IS A "PLDWORLD.COM" ...

```
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
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:
```

HERE IS A "PLDWORLD.COM" ...

```
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;

test_vectors(
[donepin, init, inAB, clk] -> [VSPROM, cclk, busB, busA, dout, address, outen, counter, program])
[1, 0, ^b11111111, C] -> [DONE, C, ^b1111, ^b1111, 1, 0, 0, 0, 1];
[1, 1, ^b11111111, C] -> [DONE, C, ^b1111, ^b1111, 1, 0, 0, 0, 1];

//Shift busA - Vector 3
[0, 1, ^b11111111, C] -> [SHIFT_A, C, ^b1111, ^b1111, 1, 0, 1, 0, 1];
[0, 1, ^b11111111, C] -> [SHIFT_A, C, ^b1111, ^b0111, 1, 0, 1, 1, 1];
[0, 1, ^b11111111, C] -> [SHIFT_A, C, ^b1111, ^b0011, 1, 0, 1, 2, 1];
[0, 1, ^b11111111, C] -> [SHIFT_A, C, ^b1111, ^b0001, 1, 0, 1, 3, 1];

//Shift busB - Vector 7
[0, 1, ^b11111111, C] -> [SHIFT_B, C, ^b1111, ^b0000, 1, 1, 1, 0, 1];
[0, 1, ^b10100011, C] -> [SHIFT_B, C, ^b0111, ^b0011, 1, 1, 1, 1, 1];
[0, 1, ^b10100011, C] -> [SHIFT_B, C, ^b0011, ^b0011, 1, 1, 1, 2, 1];
[0, 1, ^b10100011, C] -> [SHIFT_B, C, ^b0001, ^b0011, 1, 1, 1, 3, 1];

//Shift busA - Vector 11
[0, 1, ^b10100011, C] -> [SHIFT_A, C, ^b0000, ^b0011, 1, 1, 1, 0, 1];
[0, 1, ^b10100011, C] -> [SHIFT_A, C, ^b1010, ^b0001, 1, 1, 1, 1, 1];
[0, 1, ^b10100011, C] -> [SHIFT_A, C, ^b1010, ^b0000, 0, 1, 1, 2, 1];
[0, 1, ^b10100011, C] -> [SHIFT_A, C, ^b1010, ^b0000, 0, 1, 1, 3, 1];

//Shift busB - Vector 15
[0, 1, ^b10100011, C] -> [SHIFT_B, C, ^b1010, ^b0000, 0, 2, 1, 0, 1];
[0, 1, ^b11111111, C] -> [SHIFT_B, C, ^b0101, ^b1111, 1, 2, 1, 1, 1];
[0, 1, ^b11111111, C] -> [SHIFT_B, C, ^b0010, ^b1111, 0, 2, 1, 2, 1];
[0, 1, ^b11111111, C] -> [SHIFT_B, C, ^b0001, ^b1111, 1, 2, 1, 3, 1];
```

HERE IS A "PLDWORLD.COM" ...

```
//Shift busA - Vector 19
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b0000, ^b1111, 1 , 2 , 1, 0 , 1 ];
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b0111, 1 , 2 , 1, 1 , 1 ];
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b0011, 1 , 2 , 1, 2 , 1 ];
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b0001, 1 , 2 , 1, 3 , 1 ];

//Shift busB - Vector 23
[0 , 1 , ^b11111111, C] -> [SHIFT_B , C , ^b1111, ^b0000, 1 , 3 , 1, 0 , 1 ];
[0 , 1 , ^b00000001, C] -> [SHIFT_B , C , ^b0111, ^b0001, 1 , 3 , 1, 1 , 1 ];
[0 , 1 , ^b00000001, C] -> [SHIFT_B , C , ^b0011, ^b0001, 1 , 3 , 1, 2 , 1 ];
[0 , 1 , ^b00000001, C] -> [SHIFT_B , C , ^b0001, ^b0001, 1 , 3 , 1, 3 , 1 ];

//Shift busA - Vector 27
[0 , 1 , ^b00000001, C] -> [SHIFT_A , C , ^b0000, ^b0001, 1 , 3 , 1, 0 , 1 ];

//Error during configuration -> INIT goes Low - Go back to DONE..
[1 , 0 , ^b11111111, C] -> [DONE , C , ^b1111, ^b0000, 0 , 0 , 0, 0 , 0 ];
[1 , 0 , ^b11111111, C] -> [DONE , C , ^b1111, ^b1111, 1 , 0 , 0, 0 , 1 ];

//Testing different possibilities of INIT/DONEPIN..
[1 , 1 , ^b11111111, C] -> [DONE , C , ^b1111, ^b1111, 1 , 0 , 0, 0 , 1 ];
[1 , 0 , ^b11111111, C] -> [DONE , C , ^b1111, ^b1111, 1 , 0 , 0, 0 , 1 ];
[0 , 0 , ^b11111111, C] -> [DONE , C , ^b1111, ^b1111, 1 , 0 , 0, 0 , 1 ];

//Start again - Shift busA - Vector 33
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b1111, 1 , 0 , 1, 0 , 1 ];
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b0111, 1 , 0 , 1, 1 , 1 ];
[0 , 1 , ^b11111111, C] -> [SHIFT_A , C , ^b1111, ^b0011, 1 , 0 , 1, 2 , 1 ];

//Finished shifting data -> Give last clocks - Vector 36
[1 , 1 , ^b11111111, C] -> [LASTCLKS , C , X , X , X , 0 , 1, 0 , 1 ];
[1 , 1 , ^b11111111, C] -> [LASTCLKS , C , X , X , X , 0 , 1, 1 , 1 ];
[1 , 1 , ^b11111111, C] -> [LASTCLKS , C , X , X , X , 0 , 1, 2 , 1 ];
[1 , 1 , ^b11111111, C] -> [LASTCLKS , C , X , X , X , 0 , 1, 3 , 1 ];

//Back to DONE.. - Vector 40
[1 , 1 , ^b11111111, C] -> [DONE , C , ^b0000, ^b0000, 0 , 0 , 0, 0 , 1 ];
[1 , 1 , ^b11111111, C] -> [DONE , C , ^b1111, ^b1111, 1 , 0 , 0, 0 , 1 ];

END;
```

그림 3 ABEL Code for VSPROM

HERE IS A "PLDWORLD.COM" ...

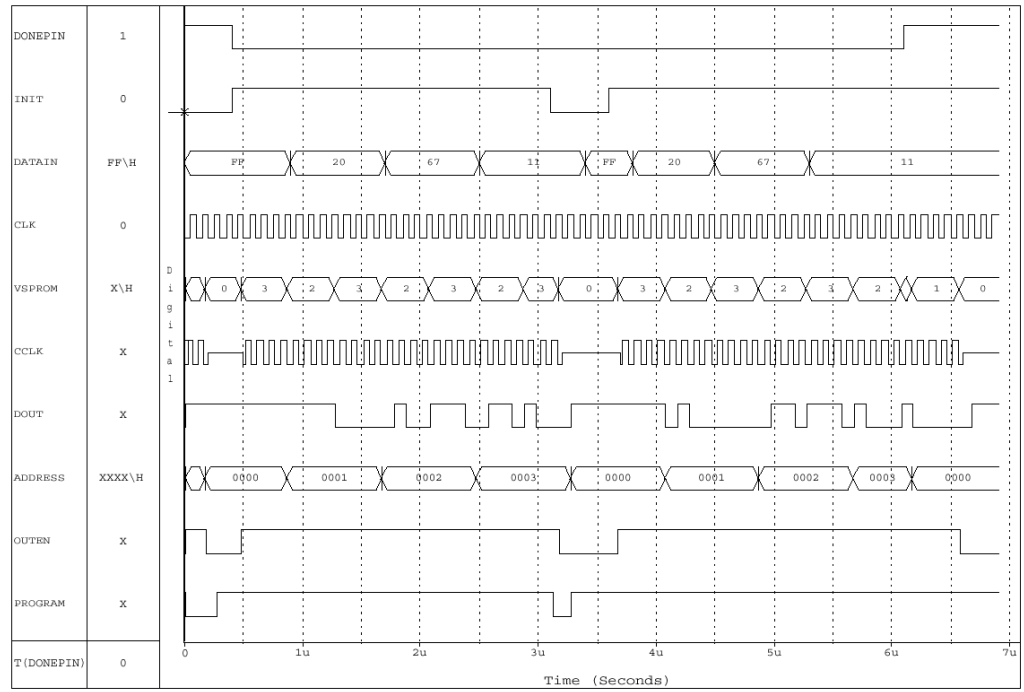


그림 4 Simulation Waveforms