Building Crosspoint Switches with CoolRunner-II CPLDs

XAPP380 (v1.0) June 5, 2002

Summary

This application note provides a functional description of VHDL source code for a N x N Digital Crosspoint Switch. The code is designed with eight inputs and eight outputs in order to target the 128-macrocell CoolRunner™-II CPLD device but can be easily expanded to target higher density devices. To obtain the VHDL source code described in this document, go to section **VHDL Code**, page 5 for instructions.

Introduction

Crosspoint switches are the building blocks of most telecommunication systems, including digital cross-connects, media gateways, and add-drop multiplexers. They offer high levels of performance, flexibility, and reprogrammability needed to quickly design efficient, scaleable systems.

The crosspoint switch reference design described in this application note is similar to the National Semiconductor crosspoint switch design. Its non-blocking architecture utilizes N-independent N:1 multiplexers to allow each output to be independently connected to any input and any input to be connected to any or all outputs. The double-row latch architecture utilized in this design allows switch reprogramming to occur in the background during operation. Activation of the new configuration occurs with a single configuration pulse. Each output can be individually disabled and set to a high impedance state, allowing flexible expansion to larger switch array sizes. Additionally, voltage translation can be done automatically within the CPLD. The block diagram of a sample 8 * 8 switch is shown in Figure 1.

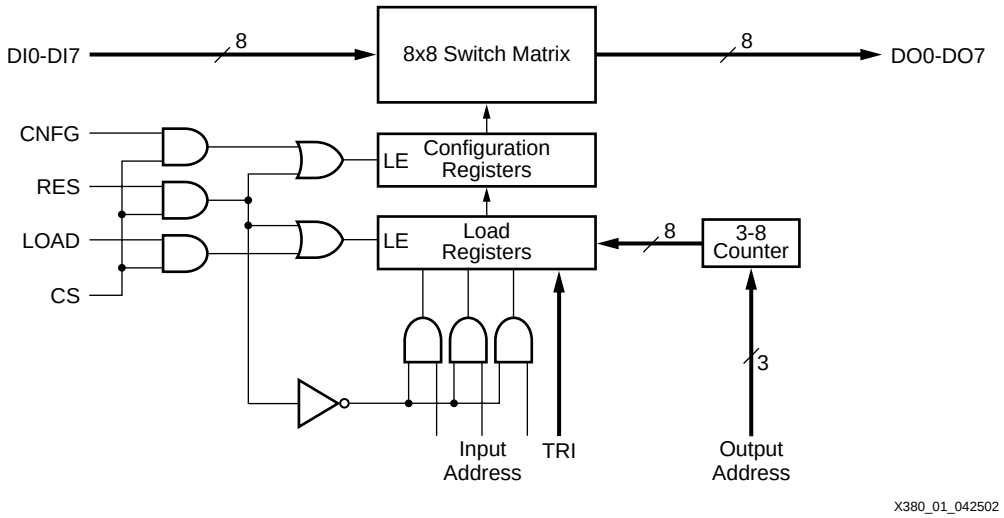

Figure 1: **Crosspoint Switch Block Diagram**

Configuring the Switch

Data controlling the switch matrix and output mode are stored in two sets of eight 4-bit registers. Three bits in each register identify the input to be connected to that particular output. The remaining bit controls whether the output is active or tri-state. A particular register in the first set, the LOAD REGISTERS, is selected by a 3-bit word placed on the output address bus. Data to be written into the load registers--consisting of the 3-bit address of the input to be connected to that output and the output enable control bit--are placed on the input address bus. Input data is stored in the load registers at the low-to-high transition of the LOAD input pin with chip select (CS) high. The contents of the load registers are transferred to the second set of CONFIGURATION REGISTERS at the low-to-high transition of the CNFG input signal. This causes the state of the entire switch matrix to be set to the selected configuration.

The entire crosspoint may be placed in an initializing state, with all outputs connected to input 0 and all outputs either enabled or tri-state. Outputs will be enabled if TRI is low and in tri-state if it is high. Apply a high-going pulse to the RES input pin with CS high to complete the initialization.

In summary, to configure an output, one must first place the 3-bit address of the output on the output address bus together with the 3-bit input bus address to be connected to that output and the output enable (TRI) control bit. Make chip select (CS) true, and then provide a high-going pulse to the LOAD input pin. If the above steps are repeated for each output to be configured, the entire crosspoint matrix can then be configured with the data held in the load registers. To implement the configuration, apply a high-going pulse to the CNFG input pin, transferring the contents of the load registers to the configuration registers. This will configure all of the crosspoints. Waveforms illustrating this will be shown in the **Design Verification** section.

Expanding the Switch Size

This VHDL code was designed for easy expansion to larger array sizes. Two generic parameters N and M, were used in the entity declaration. N is the data bus width and M is the address bus width. Users can easily alter the number of parameters to achieve the array sizes that meet their application needs. Here is a sample of the VHDL code:

```
Entity digital_crosspoint_switch is
  -- N = data bus width, M = address bus width
  Generic( N: natural := 8; M: natural :=3);
```

Figure 2 and Figure 3 show how to expand the number of inputs and outputs to increase the switch sizes beyond a single CPLD capacity.

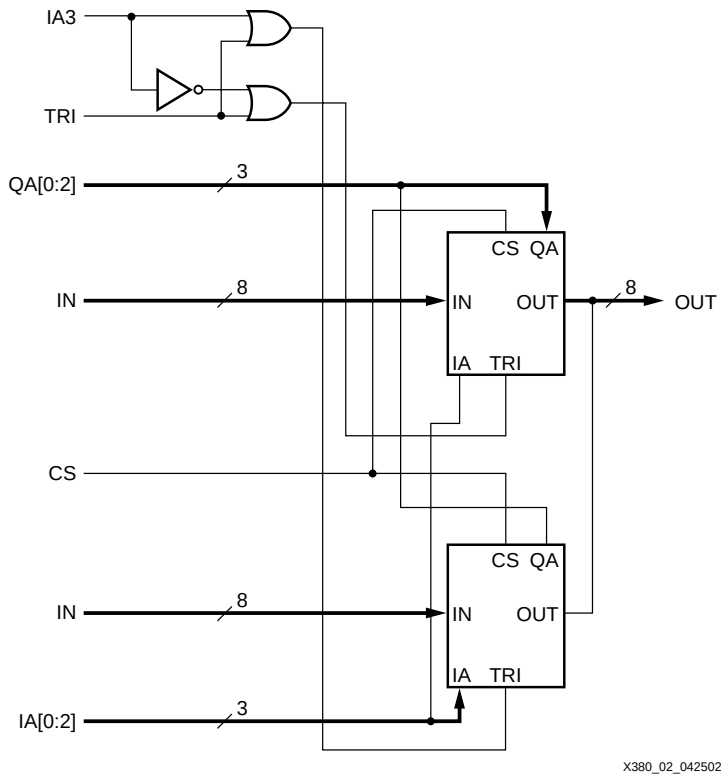

Figure 2: **Input Port Expansion**

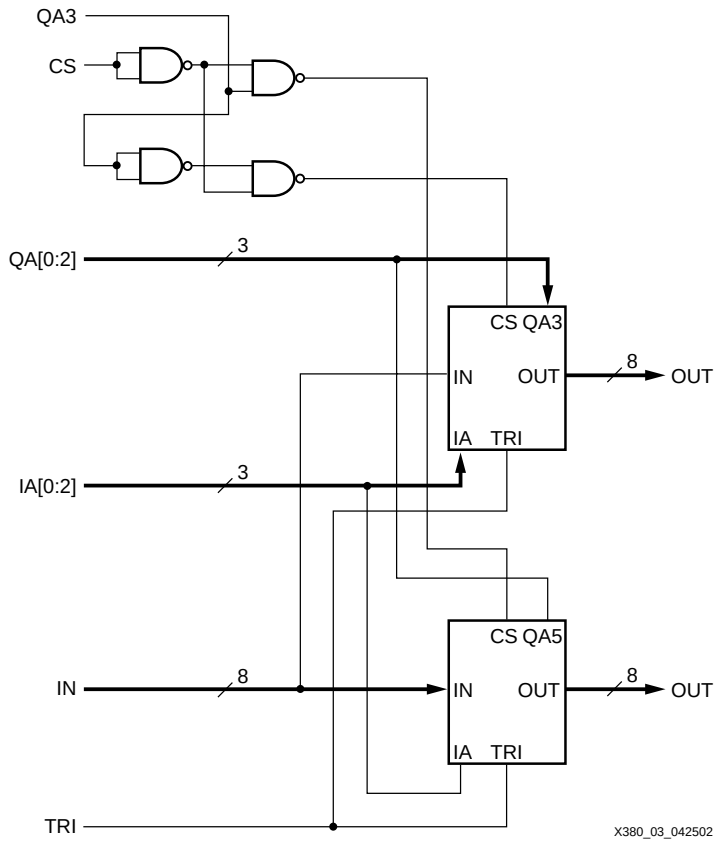

Figure 3: **Output Port Expansion**

The 8 * 8 crosspoint switch design was targeted for a 1.8V, 128-macrocell CoolRunner-II CPLD (XC2C128-VQ100) and a modified 16 * 16 switch was targeted for 256-macrocell part (XC2C256-VQ100). The utilizations are shown in Table 1 and Table 2.

Table 1: **Crosspoint Switch XC2C128 Utilization**

Resource	Available	Used	Utilization
Function Blocks	8	6	75%
Macrocells	128	61	47.66%
Product Terms	448	165	36.83%
I/O Pins	80	27	33.75%

Table 2: **Crosspoint Switch XC2C256 Utilization**

Resource	Available	Used	Utilization
Function Blocks	16	16	100%
Macrocells	256	193	75.39%
Product Terms	896	485	54.13%
I/O Pins	80	45	56.25%

Design Verification

Design verification for this Crosspoint Switch was completed using the ModelSim XE simulator in Project Navigator. The design was verified both functionally and with the post-fit timing model.

Figure 4 illustrates the crosspoint switch operation. The outputs were initially tri-state configured such that each output port connected to DI0 in the background. The CNFG pulse activated the new configuration and switched DI0 to all outputs.

The second half of the waveform shows two new configuations with the order DO following DI (01010101 -> 01010101) for the first configuration, and a reverse order connection (01010101 -> 10101010) activated from the background on the fly.

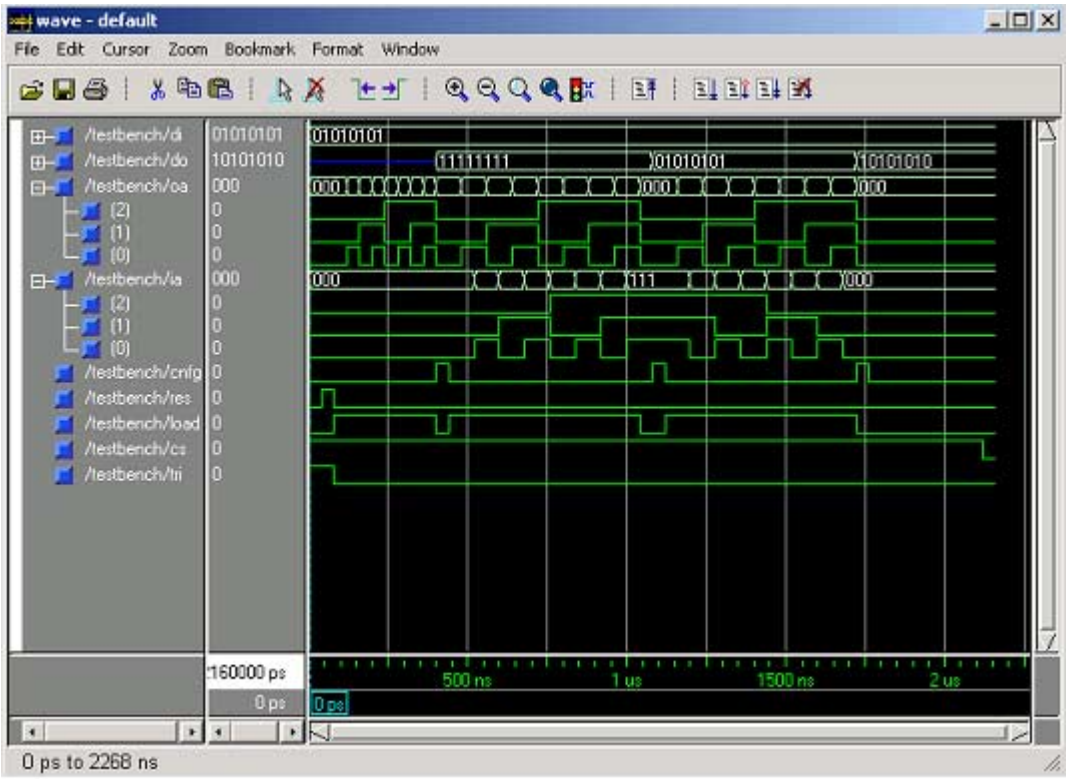

Figure 4: **Crosspoint Switch Simulation Waveform**

VHDL Code

VHDL source code and test benches are available for this design. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design has not been verified on hardware (as opposed to simulations), and it should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this design will meet your requirements, or that the operation of the design will be uninterrupted or error free, or that defects in the design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the design in terms of correctness, accuracy, reliability or otherwise.

XAPP380 - **http://www.xilinx.com/products/xaw/coolvhdlg.htm**

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/05/02	1.0	Initial Xilinx release.