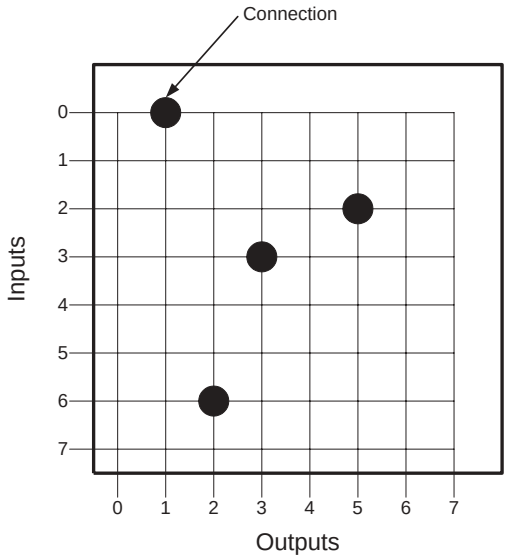
A 32 x 32 Crossbar Switch Implementation Using the Lattice ispLSI® 5384VE Device

Introduction

Crossbar switches are widely used today in a variety of applications including network switching, parallel computing and various telecommunications applications. There are off-the-shelf devices available that implement standard crossbar configurations. By using CPLDs to implement crossbar switches, design engineers have the flexibility to customize the switch to suit their specific design goals, as well as obtain switch configurations not available with off-the-shelf parts. Additionally, the use of In-System Programmable (ISP™) devices allows the switch to be re-configured if design modifications become necessary. This document addresses the implementation of a 32 x 32 Non-blocking Crossbar switch in the Lattice ispLSI 5384VE Complex Programmable Logic Device (CPLD) architecture. The design modifications for implementing a 64 x 64 crossbar switch are also given.

Figure 1. 8-Port Unidirectional Switch



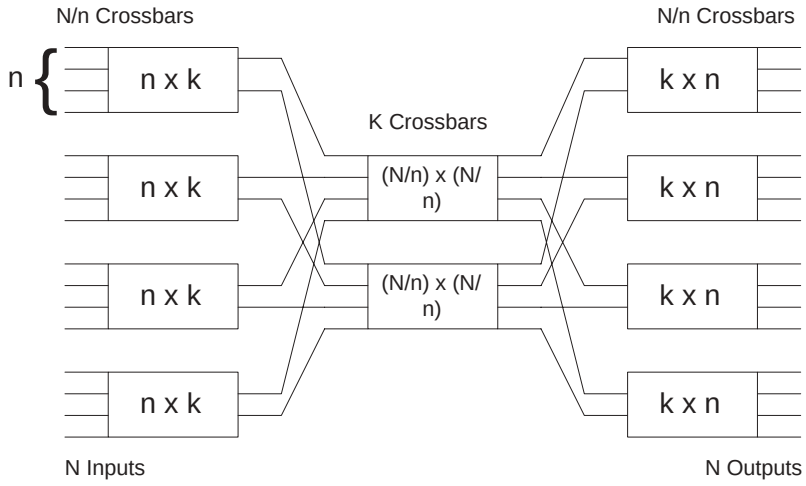
Crossbar Switches

A crossbar switch, also known as a crosspoint switch, is defined as a switch with n input lines and n output lines (n port switch). The switch has n^2 intersections, called crosspoints, where an input line and output line can be electrically connected (see Figure 1).

As can be seen from Figure 1, the number of crosspoints grows as the square of the number of lines into the switch. If we assume that a switch port does not connect to itself, the number of crosspoints needed is given by n(n-1)/2. For n=32, there are 496 crosspoint connections.

Splitting the crossbar switch into several smaller switches and interconnecting them can dramatically reduce the number of crosspoints. This technique is called space division switching. There is a penalty inherent in this technique known as blocking. Blocking can occur when two switch inputs attempt to access the same intermedi-

Figure 2. Space Division Switch



ate switch channel. Figure 2 illustrates space division switching.

As can be seen from Figure 2, only half the number of switch inputs can transmit at any given time. The total number of crosspoints needed for the switch in Figure 2 can be calculated. For the first stage, there are N/n crossbars with nk crosspoints for a total of Nk. For the second stage there are k crossbars with (N/n)2 crosspoints. The third stage has the same number of crosspoints as the first. Adding all three stages gives:

$$\text{Number of crosspoints} = 2kN + k(N/n)2$$

For 32 inputs, N=32, k=4, n=8. This gives 320 crosspoints, which is a 35 percent reduction in the number of crosspoints needed for a non-partitioned crossbar switch. The percent reduction in the number of crosspoints is proportional to n2.

32 x 32 Crossbar Switch Architecture

This ispLSI 5384VE 32 x 32 (32 port) crossbar switch design is similar to the National Semiconductor CLC018 8 x 8 Digital Crosspoint Switch, which is used for serial digital video routing, telecom/datacom switching, and ATM SONET. The block diagram of the switch is shown in Figure 3.

The switch has 32 inputs that connect to any or all 32 outputs via the 32 x 32 switch matrix. The switch matrix is constructed with thirty-two 32-input multiplexers. Each output supports individual tristate control. The MUX select lines (switch interconnects) and tristate control are configured using a set of double buffered configuration registers. The LOAD Registers are loaded for each port individually by asserting the LOAD and CS signals. The Output Address lines are decoded to select the port's LOAD register, and the input address lines are latched along with the TRI signal. The latched input address lines drive the port's MUX select lines, and TRI drives the port's tristate control. After the LOAD registers have been configured, the CNFG and CS signals are asserted, simultaneously configuring all 32 ports. This double buffering scheme prevents any data from being lost while the switch interconnects are updated. Two reset modes are supported. Broadcast reset results in all switch out-

Figure 3. Crossbar Switch Block Diagram

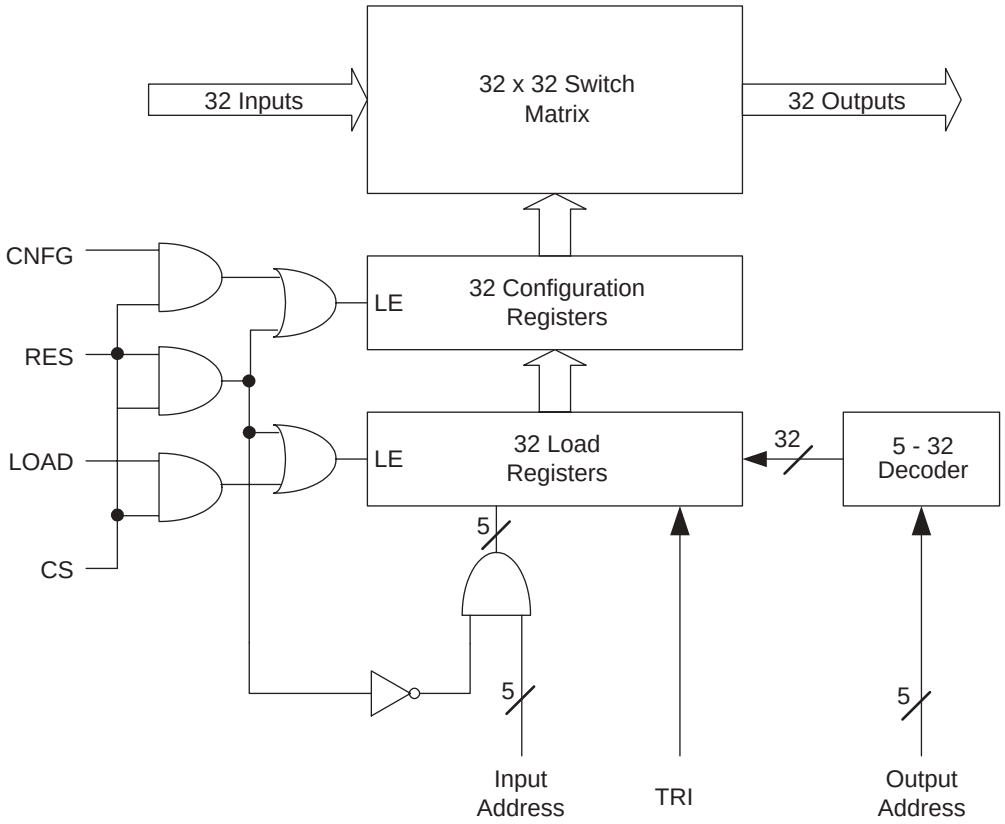
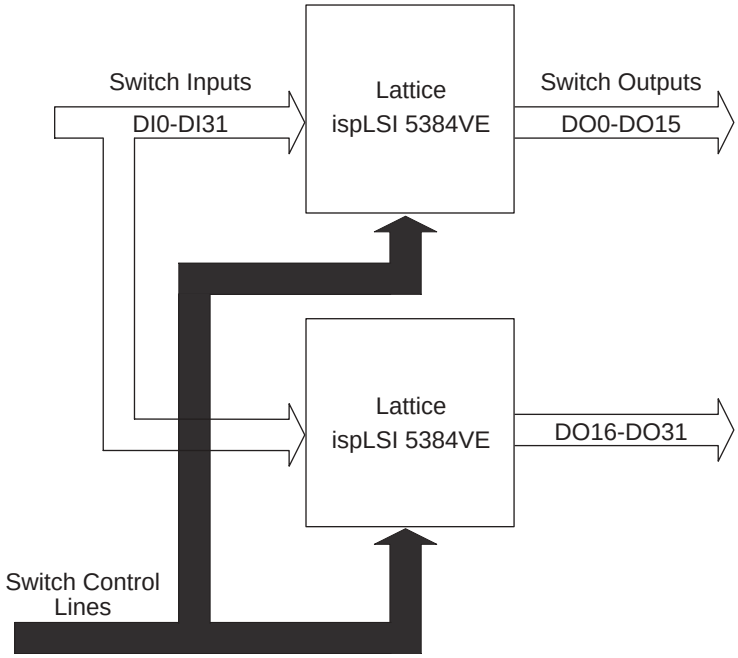


Figure 4. 32 x 32 Crossbar Switch Implementation



puts being set to select port 0. Broadcast is initiated by asserting RES and CS simultaneously. Tristate reset results in all outputs being disabled. Tristate reset is initiated when TRI is asserted along with RES and CS.

Implementation

This design is ideally suited to the Lattice ispLSI 5384VE device due to the large number of inputs required for each MUX. For the 32 x 32 switch, 32 x 32-input MUXes are required. The Lattice ispLSI 5384VE has 68 inputs into each Generic Logic Block (GLB), which allows the MUXes to be implemented in a single level of logic. Because this design uses a double buffered switch configuration scheme, a significant amount of device resources must be devoted to the switch control. Each port requires 12 configuration registers. For 32 ports, this totals 384 macrocells, not including the required decoding logic. Because of this, two ispLSI 5384VE CPLDs are required to implement a 32 x 32 crossbar switch (Figure 4). However, the entire implementation still takes only one level of logic.

This implementation requires 305 macrocells per device, or 79% utilization, leaving room for additional logic. The switch design was coded in VHDL allowing for the number of ports or control logic to be reconfigured simply by making modifications to the VHDL source code.

64 x 64 Implementation

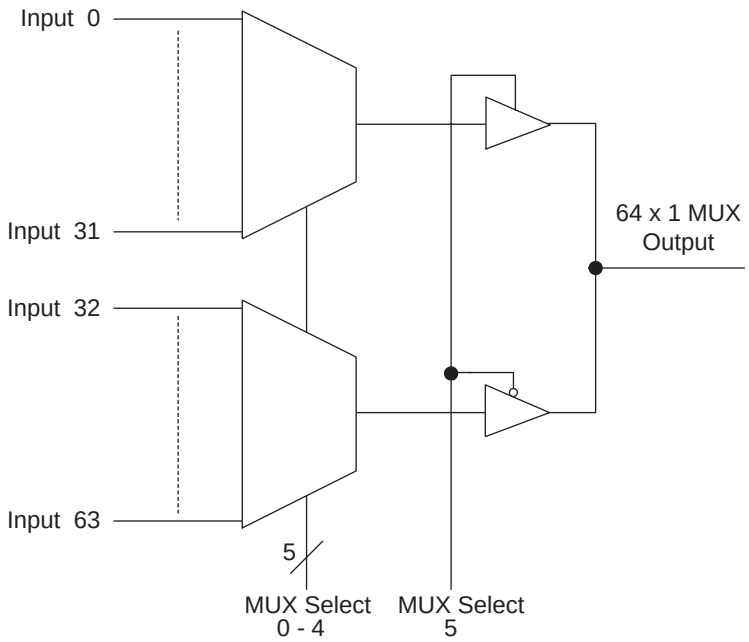
The flexibility of the ispLSI 5384VE does not limit the maximum number of ports to 32. A 64 x 64 Crossbar Switch can be implemented using four ispLSI 5384VE devices, all in a single level of logic. This can be accomplished in the following manner. Although there are 68 inputs into each GLB, the maximum number of product terms allowed for a GLB output is 35. It would take two GLB levels to directly implement a 64 x 1 MUX. However, because the ispLSI 5384VE supports tristate control on all I/Os, a 64 x 1 MUX can be constructed in a single GLB level. This is accomplished by externally tying the outputs of two 32 x 1 MUXes together and using the tristate control on the I/Os as the MSB of the MUX select lines (Figure 5)

By inverting the tristate control between the two MUX outputs, only one MUX can drive the output pin at any given time, thus avoiding the possibility of contention.

Conclusion

Clearly, the Lattice ispLSI 5384VE is the superior choice for implementing crossbar switches due to its Big Fast Wide (BFW) GLB structure. No other CPLD can boast a one-level crossbar switch implementation with up to 64 inputs and outputs. This is achieved in the ispLSI 5384VE

Figure 5. 64-Input MUX



due to the large number of GLB inputs (68/GLB) and GLB product terms (160/GLB). In using the ispLSI 5384VE and coding the design in VHDL, the system designer is not limited to the standard feature set available with off-the-shelf solutions. Because this 32 x 32 implementation requires only 79% device utilization, there are sufficient resources left available to implement other types of standard logic and control functions.

Source File

The source code for this design example may be downloaded from the Lattice web site at www.latticesemi.com.

Technical Support Assistance

Hotline:	1-800-LATTICE (Domestic) 1-408-826-6002 (International)
e-mail:	techsupport@latticesemi.com