

## Summary

This application note will help designers understand the XC9500 architecture and how to get the best performance from these devices.

## Xilinx Family

XC9500

## Introduction

To get the best performance from any CPLD, the designer must be aware of its internal architecture and how the various device features work together. This application note provides useful examples and practical details for creating successful designs. These design techniques apply to all XC9500 devices because the architecture is uniform across the family.

## XC9500 Architecture

The XC9500 architecture is comprised of multiple identical function blocks internally connected by a fully populated FastCONNECT switch matrix. The XC9500 function block has 18 macrocells per block and supports pin-to-pin speeds as fast as 5 ns, with clock rates up to 125 MHz. I/O signals can interface with 5 volt, 3.3 volt, or both levels.

Figure 1 shows the XC9500 architecture. Note the regular structure of high speed function blocks centrally connected by the FastCONNECT matrix and surrounded by pins. Signals enter and exit on the pins, form logic operations within the function blocks, and form connections and logic operations within FastCONNECT. Each of these features is discussed in the following sections to highlight key functionality.

## Interconnect Within Function Blocks

Function blocks (FBs) have 36 input sites. The FBs receive signals from the FastCONNECT matrix and input pins. The logic blocks generate 18 signals per FB from the 18 macrocells in each block, and each macrocell signal can drive its own dedicated I/O pin or feedback by entering the FastCONNECT matrix. Additional high speed local paths exist within the FB.

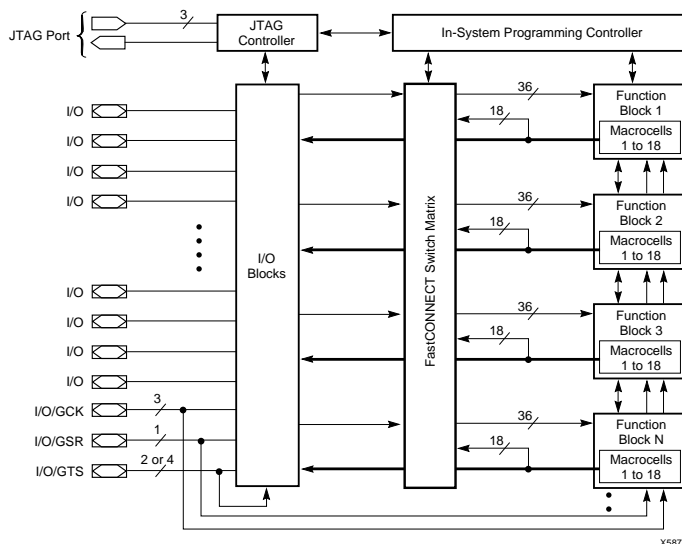


Figure 1: XC9500 Architecture

The FastCONNECT Switch Matrix

The FastCONNECT switch matrix attaches high speed signals to the function blocks. It also connects every macrocell output to the function blocks through a fully populated cross-point switch. This high degree of connectivity is a key factor that allows the designer to make design changes even after a device is mounted on a PC board.

Function Blocks

The function blocks, shown in Figure 2, are groups of 18 macrocells. Each FB has 90 product terms which can be assigned to any of the 18 macrocells. This provides optimum logic flexibility within the function block and supports pin-locked designs. The highest possible performance is attained by the software assigning a uniform five product terms per macrocell. The macrocell outputs can then drive output pins as well as provide inputs to both the FastCONNECT matrix and the FB in which it resides.

The Macrocell

In the default mode, there are 5 product terms that OR together driving the D input to the macrocell flip-flop, as shown in Figure 3. The most common arrangement

includes an Exclusive-OR gate capable of performing parity, full addition, or logical inversion.

Another configuration exports product terms to a neighboring macrocell, increasing that macrocell's available product terms. Product term exporting is shown in Figure 3.

The XC9500 flip-flops can be configured as D- or T-type. This permits the building of efficient counters using only a few gates to drive the state transitions. Table 1 summarizes the number of product terms needed to build common logic functions; most datapath functions require one or fewer macrocells per bit.

Table 1: Macrocell/Product Term Allocation

Data Operation	P-Term Used
Shift Register	2
Counters	2-4
n:1 Mux	n
Adder	6
Exclusive-OR	2
Storage registers	1

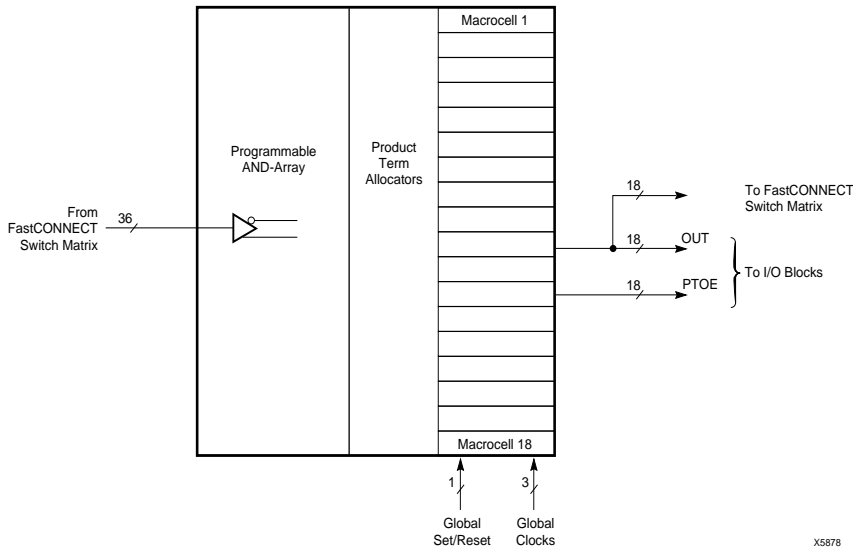
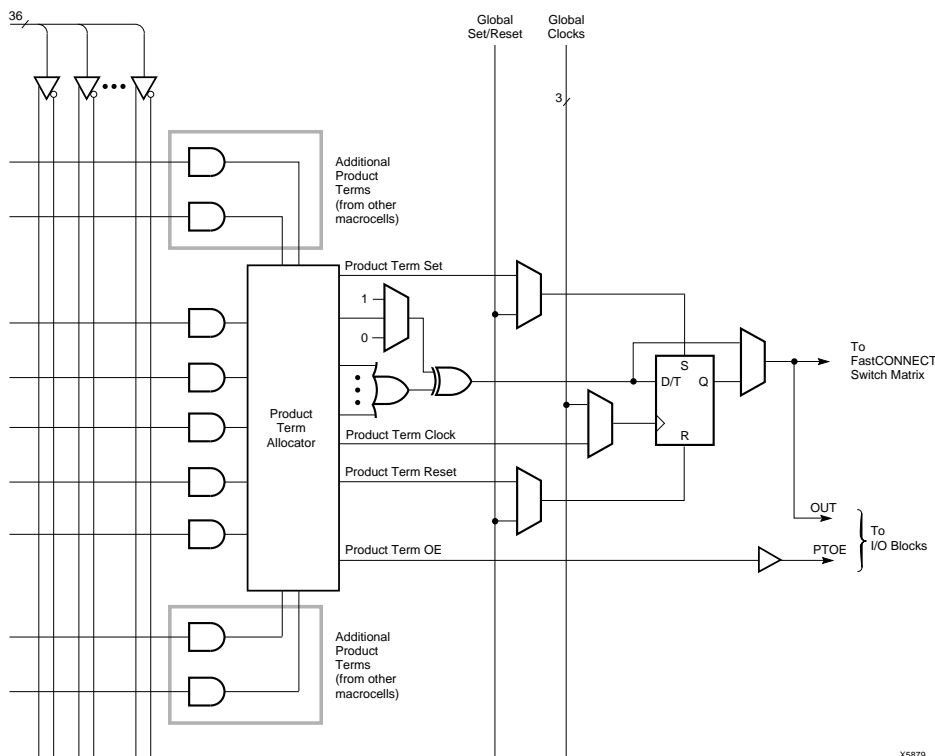


Figure 2: XC9500 Function Blocks



X5879

**Figure 3: XC9500 Macrocells**

**Table 2** shows the pin compatibility of the XC9500 family. Designs can easily be migrated to larger or smaller devices. In many cases, greater density with equivalent speed can be gained by using larger parts. If a design is initially targeted to a smaller device, the same design can be moved into a larger device, if additional capability is required. This capability allows designers maintain their pin

assignments, even when designs must be moved to a larger capacity device.

Moving designs from larger to smaller devices can also be accomplished, while keeping the original pinout, if the smaller device has enough resources to contain the design.

**Table 2: XC9500 Available Packages and Device I/O Pins**

Package	XC9536	XC9572	XC95108	XC95144	XC95180	XC95216	XC95288	XC95432	XC95576
44-Pin PLCC	34								
44-Pin VQFP	34								
84-Pin PLCC		69	69						
100-Pin PQFP		72	81	81					
100-Pin TQFP		72	81						
160-Pin PQFP			108	133	133	133			
208-Pin HQFP					166	166	168		
352-Pin BGA						166	192		
432-Pin BGA								232	232

Note: These numbers do not include the dedicated JTAG pins.

## Automatic Software

The following design examples are created in ABEL. Typically, designers won't designate specific function mapping into XC9500 designs. However, designers occasionally like to control how a solution is implemented, and in that case, these methods may be of interest.

Boolean operators used by ABEL are ! for invert, # for OR, and & for AND. Combinatorial logic expressions are formed with an equal sign, with operands and operators located on the right hand side of the expression.

Flip-flops are formed by writing expressions for the specific control pins of the flip-flop. The D-input is a special case, represented by the compound symbol "D:". Clock inputs are determined by the syntax *flip-flop\_name.clk*, and reset inputs are designated by *flip-flop\_name.rst*.

An ABEL design file contains a header section including optional documentation sections and mandatory declaration of inputs, outputs, global signals, and any user preferred arrangement of functions.

### Logic AND

The FastCONNECT switch matrix is capable of combining signals with a wire-AND function. Signals entering the FastCONNECT matrix are assigned to function block inputs, and multiple signals, may form a wired-AND function, which reduces the macrocell logic requirements. This feature increases both the logic capacity and available signal inputs to the Function Blocks.

### Gates

The following expressions show the basic logic operations.

```
ABAR = !A;
AORB = A#B;
AANDB = A&B;
ANORB = !(A#B);
ANANDB = !(A&B);
AEXORB = A$B;
AEXNORB = A!$B;
```

### Multiplexers and Decoders

Using the above methods, compound expressions are formed to build logic functions. Using A0 to A3, B0 to B3, and SEL (select) as inputs, a multiplexer is described as follows:

```
DAT0 = SEL&A0 # !SEL&B0;
DAT1 = SEL&A1 # !SEL&B1;
DAT2 = SEL&A2 # !SEL&B2;
DAT3 = SEL&A3 # !SEL&B3;
```

The approach extends to larger multiplexers. The previous example uses one macrocell per data bit and leaves behind two unused product terms in each macrocell. To take

advantage of four product terms per macrocell, the implementation expands as follows:

```
DAT0 = S1&S0&D0 # S1&!S0&C0 #
      !S1&S0&B0 # !S1&!S0&A0;
DAT1 = S1&S0&D1 # S1&!S0&C1 #
      !S1&S0&B1 # !S1&!S0&A1;
DAT2 = S1&S0&D2 # S1&!S0&C2 #
      !S1&S0&B2 # !S1&!S0&A2;
DAT3 = S1&S0&D3 # S1&!S0&C3 #
      !S1&S0&B3 # !S1&!S0&A3;
```

Very high speed decoders can be built in the macrocells to form SRAM select signals, but do not use all of the macrocell product terms or the flip-flop in most cases. Decoders are formed as follows:

```
DEC0 = !A3&!A2&!A1&!A0;
DEC1 = !A3&!A2&!A1&A0;
DEC2 = !A3&!A2&A1&!A0;
```

### Registers

Simple registers are formed as follows:

```
A:= DATAINPUT;
A.CLK = CLOCK;
A.RST = RESET;
```

This describes a D-type flip-flop with its input tied to a signal named DATAINPUT, its clock tied to a signal called CLOCK, and its reset input tied to a signal called RESET.

### Shift Registers

Cascading registers results in a shift register as follows:

```
A:=DATAINPUT;
B:=A;
C:=B;
D:=C;
A.CLK = CLOCK;
B.CLK = CLOCK;
C.CLK = CLOCK;
D.CLK = CLOCK;
A.RST = RESET;
B.RST = RESET;
C.RST = RESET;
D.RST = RESET;
```

This shift register uses four macrocells. If the signals designated A,B,C,D are declared as outputs, they will appear somewhere at the pins of an XC9500 device. If A,B,C, and D are declared as nodes (internal points), the software implements them within the macrocells.

### Counters

Counters can be built in a number of ways. The most efficient method is to have the macrocell flip-flops configured as T-type flip-flops. The following equations form T-type flip-flops; they add logic to load, hold, increment, and clear the flip-flops. Note the compact vector notation:

```

module Tcount
  title '4 bit counter with load and clear'
  D0..D3pin;
  Q3..Q0pin istype 'reg_T';
  CLK, I0, I1pin;
  Data = [D3..D0];
  Count = [Q3..Q0];
  Mode = [I1,I0];
  Clear= [0,0];
  Load = [1,0];
  Inc= [1,1];
  equations
    Count.T = ((Count.q+1) & (Mode == Inc)
    # (Data)& (Mode == Load)
    # ( 0 )& (Mode == Clear))
    $ Count.q
    Count.C = CLK;
  end

```

## Comparators

Comparators are easily handled by the XC9500 macrocell. Single bit comparators do not use all available macrocell product terms, and therefore a more efficient method is to implement four bits at a time, to generate multiple compares per macrocell:

```

COMP = !B1&!B0&!A1&!A0 + B1&!B0&A1&!A0
       !B1&B0&!A1&A0 + B1&B0&A1&A0

```

Several COMP signals can be gated together to detect equality across larger groups of bits. Each group of four bits uses 4 function block inputs and several four-bit comparisons can occur per function block. Another macrocell then forms the composite function of all the bit comparisons, as needed. **Figure 4** shows this technique expanded to a 10-bit comparator which is commonly used on the most significant address lines of a 32-bit microprocessor's address lines.

## Parity

Similar to comparisons, parity can be calculated with multiple data bits per macrocell. The first three bits are calculated using four product terms ORed together. This result is

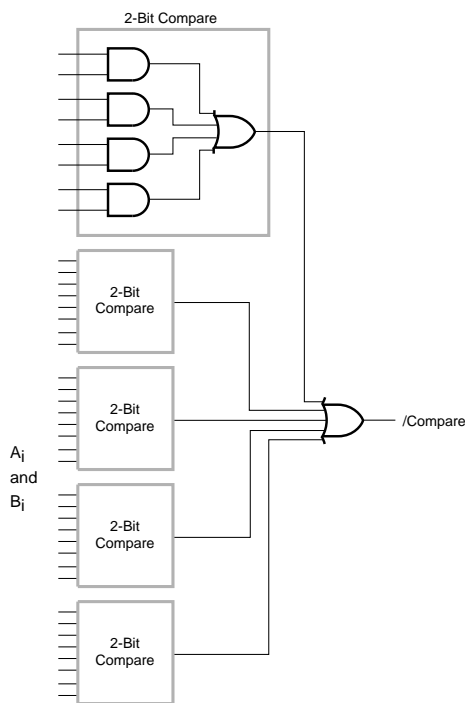
then delivered to the macrocell Exclusive-OR gate where a fourth data variable is introduced.

## Latches

Occasionally, designers need a transparent latch within an XC9500 device. This latch is formed by feeding the macrocell combinatorial logic back upon itself as shown in the following equation:

$$Q = \text{ENA} \& \text{DATA} + !\text{ENA} \& Q + Q \& \text{DATA};$$

The signal Q&DATA is included to make the Q output glitch free.



**Figure 4: 10-Bit Comparator**

## Practical Considerations for XC9500 Designs

By following a few simple rules, XC9500 devices can easily interface with systems using 3.3 volt and 5 volt devices. Also, these devices behave much better if standard high performance printed circuit board techniques are used (as with all high-speed devices) so a small checklist is provided here for those rules.

### Mixed Voltage Operation

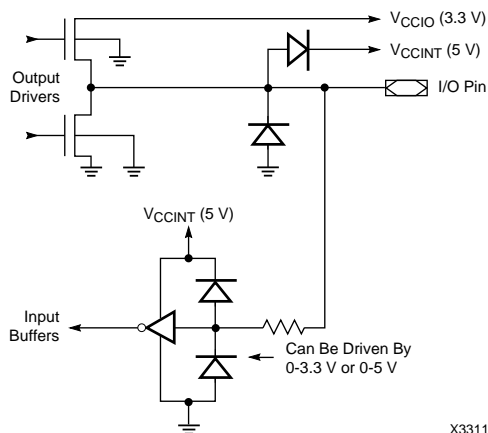
XC9500 CPLDs support mixed voltage systems combining both 3.3 volt and 5 volt components as shown in **Figure 6**. The XC9500 family contains both logic and level shifting functions in a single programmable device. This eliminates the need for discrete level translation buffers. The XC9500 devices feature split power supply rails. The internal core logic always runs at 5 volts for the fastest possible performance. The output buffers can be powered by either 5 volts or 3.3 volts by connecting the I/O  $V_{CC}$  to a 3.3 volt or 5 volt supply. True TTL compatibility allows XC9500 CPLDs to drive and be driven by any combination of 3.3 and 5 volt logic without any performance penalty, even when the I/O  $V_{CC}$  pins are powered by 3.3 volts.

The XC9500 I/O structure is shown in **Figure 5**. Input protection diodes are connected to the internal 5 volt power supply rail and not to the output buffer supply rail. This allows the input to withstand a maximum voltage of >5 volts, even when the I/O power pins connect to 3.3 volts. Since both output transistors are N-channel devices, there is no parasitic diode to be forward biased if the output is in a 3-state condition and a 5 volt device is driving the

XC9500 I/O pin. Therefore, the device can operate on a bus that includes both 3.3 volt and 5 volt devices.

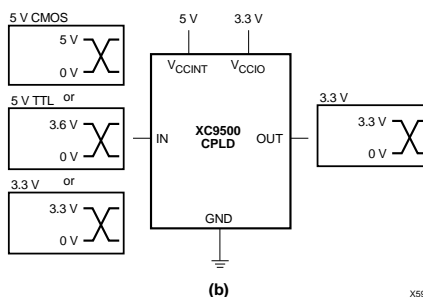
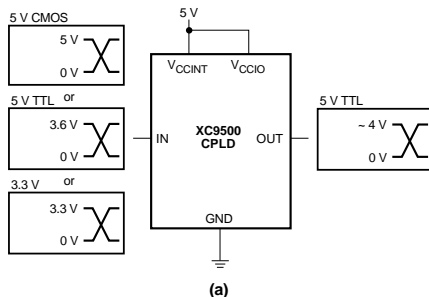
Because the input protection circuitry is powered by the 5 volt core logic supply, the device pins should not be driven externally until the 5 volt  $V_{CCINT}$  supply is greater than 3 volts. In mixed 3.3/5 volt systems, where other 3.3 volt logic may be driving XC9500 devices, this requirement can be easily met by powering the 3.3 volt supply at the same time (or after) the 5 volt supply.

XC9500 devices are TTL-compatible with 3.3 and 5 volt logic as shown in **Figure 7**. The 5 volt TTL logic input thresholds are  $V_{IH} = 2.0$  V and  $V_{IL} = 0.8$  V. XC9500 CPLDs drive HIGH greater than 2.4 volts and LOW below 0.4 volts at the rated output drive currents, with at least 400 mV noise margin.



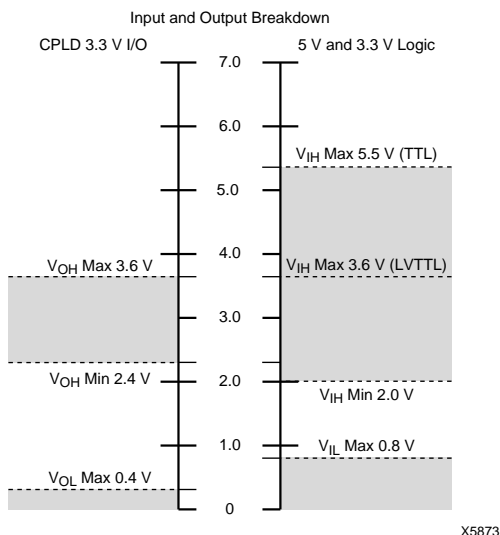
X3311

**Figure 5: XC9500 I/O Architecture**



X5901

**Figure 6: Typical Mixed Voltage System**



**Figure 7: Driving 3.3 volt and 5.0 volt Components**

## High Speed Design Considerations

XC9500 CPLDs are offered with pin-to-pin delays as fast as 5 ns, and the actual speed may be faster. Therefore, additional care should be taken to minimize noise so that adjoining devices will operate properly.

Many high speed designs also require high current drive outputs for handling capacitive loads. XC9500 CPLDs provide 24 mA drivers to eliminate the need for additional buffering and therefore the designer needs to manage the total current being switched to minimize possible ground rise problems.

As with other high speed logic devices, XC9500 CPLDs should use low inductance capacitors located as close as possible to the  $V_{CC}$  and GND pins on a PC board. Care should be taken to mount the devices so that the PC board interconnect traces are as close as possible to the target signal destinations.

## PC Board Layout Checklist:

Complying with the following checklist assures a successful design with XC9500 CPLDs:

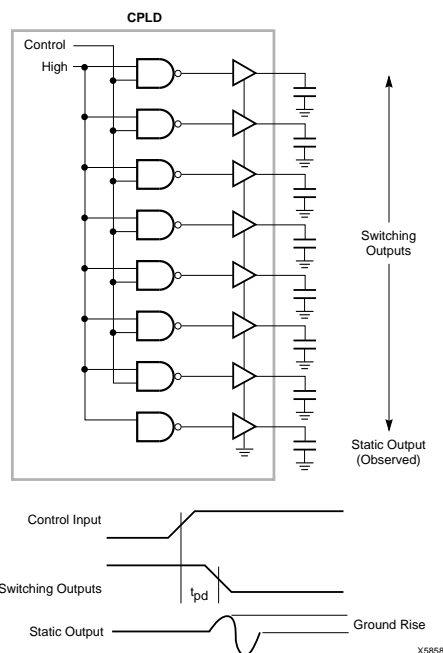
1. Tie unused inputs to ground.
2. Locate XC9500 CPLDs near the devices they drive (or are driven by) to minimize transmission line effects.
3. Use wide spacing between fast signal lines (particularly clocks) to minimize crosstalk.
4. Place power pins ( $V_{CC}$  and GND) on separate printed circuit board planes. Fast signals should reside on a different plane.

5. Decouple the device  $V_{CC}$  input with a 0.1  $\mu\text{f}$  capacitor. Directly connect each  $V_{CC}$  pin to the nearest ground plane. Low inductance, surface mounted capacitors are recommended.
6. Decouple the printed circuit board power inputs with 0.1  $\mu\text{f}$  ceramic (high frequency) and 100  $\mu\text{f}$  electrolytic (low frequency) filter capacitors.
7. Connect all device ground pins together.
8. Avoid using sockets to attach XC9500 CPLDs to the PCB. Direct soldered connection minimizes inductance and reduces ground rise. XC9500 CPLDs are specifically designed for direct PCB attachment.

## Managing Ground Rise

Designers must also be aware of additional factors that can affect the performance of fast, high-current drive systems. Possible voltage rise on device ground pins can affect the driven output levels and be sensed by the switching CPLD.

**Figure 8** shows how ground rise is typically observed. In this setup, multiple outputs are switched with a control variable, while one output is constantly being driven low and observed.



**Figure 8: Ground Rise Test**

As the multiple outputs switch, their in-rushing current converges at the ground pins of the device. Lead impedance causes the reference ground to develop a voltage higher than that which occurs before switching. The result is that

the static output being observed also develops an observable voltage swing.

All digital ICs have this property. No harm is caused to the system unless the voltage swing on the static output is capable of switching another circuit down the line. Problems can occur if the voltage swing is excessive. This effect is particularly significant if the static (quiet) signal is attached to another circuit's clock input.

Two factors contribute to this ground rise. First, the amount of capacitive load being driven is important because charge on this capacitance is the source of the in-rushing current. Second, the number of simultaneous switching outputs is a factor since each switching output adds to the total capacitance being discharged.

XC9500 devices are in symmetric packages with multiple ground pins. However, some designs may need more grounding, and therefore the XC9500 family includes user-programmable ground pins that allow the device I/O pins to be configured as additional grounds. Tying programmable ground pins to the external ground connection reduces system noise. The Xilinx XACTstep software can be used to connect unused macrocell outputs to ground.

The following checklist will help reduce unnecessary ground noise:

1. Only connect the essential outputs to I/O pins. Intermediate shift register bits and counter bits that need not drive outputs should remain buried.
2. Minimize the number of outputs switching simultaneously.
3. Two global clock or GTS inputs can be managed by delaying one of the signals to gain signal skew.
4. Using additional ground pins can lower ground rise effects. Unused pins configured as grounds can be tied directly to the PC board ground plane. This splits the current driven into heavily loaded ground pins and lowers the voltage rise.
5. Signal skewing can also reduce ground rise. This can be achieved by mixing ordinary and fast slew rate outputs. Only assign the fast slew rate to signals that require it.

## Conclusion

By using the techniques described in this application note, designers can achieve the highest-performance logic using the XC9500 family. The XC9500 family datasheet contains additional descriptions of the important system features.