CoolRunner Power-Saving Tips and Tricks

These techniques can lower your CoolRunner power consumption by 40%.

by Frank Wirtz
Staff Applications Engineer, Xilinx Inc.
frankw@xilinx.com

With the advent of Fast Zero Power technology and CoolRunner CPLDs, you can now create portable, high-performance, low-power, programmable devices, effortlessly. And, with some additional effort, you can reduce your power consumption by as much as 40%. However, to accomplish ultra low power reductions, you must first understand the mechanics of CPLD logic generation.

Xilinx has published a new application note, "Low Power Tips for CPLD Design" (XAPP346), that describes design techniques you can use to further reduce power consumption in CoolRunner CPLDs, which are already the lowest power-consuming CPLDs in the world. Here are some highlights from that application note.

Tips and Tricks for Reducing Power

The CoolRunner XPLA architecture gives you a flexible logic allocation tool that allows you to decrease power consumption by placing your logic in optimum locations. To use this tool, you need to understand the basic architecture of the device, and you need to know how the fixed geometry of the device determines both the speed-sensitive paths and the power-sensitive paths.

The following design implementation techniques are just a sampling of the information you can use to slash power consumption to a minimum:

Terminate!

You must properly terminate all inputs to a CMOS buffer. A single floating pin can result in an increase of quiescent current by 13mA. Slow input transitions will also cause unnecessary power use. Test data shows that input buffer power consump-

tion doubles if input rise time increases from 800ps to 5ns per input.

Congregate!

You can see how your design is implemented by reviewing your fitter report, and then adjust the fit to constrain your high frequency signals to a single logic block. This will decrease the distribution of high-speed nets and further decrease power consumption.

Modulate!

The application note details special clock considerations and explains how asynchronous clocking can provide low power benefits. Typically, asynchronous clocking increases power consumption. Modulation in this instance refers to only applying a clock signal to a register when it is required. Many designs have registers that infrequently change state, yet the clock signal is continually present and applied to the register. While asynchronous design techniques are usually discouraged, they do provide designers with additional flexibility when low power (or sometimes high speed) characteristics are required.

As an example of this technique, consider a counter circuit. In the case of a binary counter, not all of the registers change state on each significant clock edge. Designers can use a high speed clock for the LSBs of the counter, and then use a prescaled clock for the higher order bits, so the total amount of power required by the clock buffers is decreased.

Mixed Voltage Interfacing

When interfacing devices that have different VCC levels, consider the impact caused by under driving a CMOS input. Because a CMOS input buffer is comprised of at least two primary transistors, a P-channel pull up and an N-channel pull down, there exists a region of input voltage where both transistors are slightly on, and current flows from VCC to GND through these buffers. This causes power to be wasted, and since the output of this buffer may also be in the linear region, it can cause problems because other internal devices depend upon the output voltage of the buffer for driving their inputs.

In some cases, mixed voltage interfacing is necessary. Slight modifications to differing VCCs can drastically reduce power consumption in these instances. For example, the XPLA3 devices may be powered at 3.3V +10%, and 5V devices may be powered at 5V -10%. This changes the differential between Voh and Vih by 800mV per input, and will significantly reduce wasted power. However, examine the data sheet to ensure safe and reliable operating conditions.

Default System Conditions

Attention to default system operating conditions may provide an insight into ways you can further decrease power consumption. As an example, a CPLD may be interfaced to a CMOS microcontroller with programmable (polarity sensitive) interrupts. If it is necessary to interface a 3.3V CPLD to a 5V interrupt, system power can be saved by programming the microcontroller interrupt such that the system operates with the interrupt level normally low. This decreases the amount of time that the interrupt is active (high) which will reduce the overall amount of power consumed when under driving a CMOS input.

The Effects of Implementation Style

Implementation style affects power consumption. For example, consider how different types of counters are implemented; binary, Grey, and LFSR counters are created in different ways and require different amounts of resources. Keep in mind that a minimal number of changing signals will always deliver the lowest dynamic power solution.

Binary Counters

Typical binary counters will have their outputs changing state at a rate of:

$$\text{Percent of bits toggling} = \frac{2^{n+1} - 2}{n2^n} X\ 100$$

Where the number of bits of the counter = n

So, a typical 8-bit binary counter would have approximately 25% of its bits changing state for any single clock edge.

LFSR Counters

LFSR (Linear Feedback Shift Register) counters are wonderful solutions for FPGA users who need to keep look-up table fan-in to a minimum. However, because of the internal hardwired feedback of CPLDs, this type of counter consumes much more power than the other counter examples described here.

For example, an 8-bit LFSR counter has approximately 50% of its bits changing on average for any single clock edge. In comparison, an 8-bit binary counter changes at a 25% bit rate.

Grey Code Counters

Because of their characteristic step pattern of a single changing bit, Grey code counters offer designers the lowest power consumption of these three counter methods. The average bit change rate for an 8-bit Grey code counter is approximately 13% as defined by the equation:

$$\text{Percent of bits toggling} = \frac{1}{n} X\ 100$$

The Grey code design implementation is the most difficult, however, because next-state information must be coded for each count value.

The Bottom Line

Take full advantage of the CoolRunner low power benefits by downloading the free "Low Power Tips for CPLD Design" application note (in PDF format) from the Xilinx website at: www.xilinx.com/xapp/xapp346.pdf.