XAPP784 (v1.0) June 28, 2005

Bulletproof CPLD Design Practices

Summary

This application note consolidates more than 12 years of experience with CPLD customer issues. Check lists of best practices for robust design are presented so CPLD users can obtain best operation from Xilinx CPLDs.

Introduction

Robust design can be obtained with Complex PLDs by following a simple set of guidelines. Most of these guidelines are common sense for digital designers. However, we frequently encounter problems that could easily have been avoided by users if they had only check listed their designs in advance. The following lists are comprised of two sections: hardware practices and design software practices.

Hardware Practices

1. Attach all V_{CC} and GND pins to the printed circuit board correctly
2. Decouple all V_{CC} pins (V_{CCINT} and V_{CCIO})
3. Use strict synchronous design
4. No floating input pins
5. Sink current when driving LEDs
6. Terminate high speed outputs
7. Avoid pull-down resistors on pins
8. Avoid driving I/Os above V_{CCIO}
9. Remember to include JTAG stakes on the printed circuit board
10. No external JTAG termination for internally terminated pins
11. Assure sufficient regulator current is available
12. Power V_{CCINT} before V_{CCIO} (if possible)

Family Specific Hardware Practices

- CoolRunner-II: Use Schmitt triggers on slow moving input signals
- XPLA3: Understand JTAG Port Enable and use it correctly
- XC9500/XL/XV: Use low power mode on all noncritical signals

Design Software Practices

1. Do not disregard report file warnings
2. Use strictly synchronous design
3. Understand timing limits/warnings
4. Review the CPLD Fitter Report equations
5. Always configure I/Os properly (LVCMOS 3.3, LVCMOS 1.8, etc.)
6. Let the design software define the pinouts, if possible
7. There is solid value to design simulation

© 2005 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc.
All other trademarks are the property of their respective owners.

Checklist Discussion

We will now expand the discussion to detail both Hardware and Software Practices. Matching the sequence above, we will expand upon Hardware Practices first.

Hardware Practices

1. Attach all V_{CC} and GND pins to the printed circuit board correctly.

   Xilinx CPLDs do not contain any redundant V_{CC} and ground pins. The pins are supplied to best provide the power needs of the whole device. It would be impossible to determine which V_{CCINT} pins are applicable to sections of the device, so it is risky to omit connecting any of them. V_{CCIO} pins are clearly targeted to specific I/O sections of the die, but not connecting them can introduce sources of current leakage, as well as unbalance the rest of the device. Similarly, ground pins are a “group effort” and any missing attachments may degrade overall behavior. Floating pins can increase power and reduce ground bounce resilience. The simple rule: Connect all V_{CC} and GND pins is very clear and easy to do.

2. Decouple all V_{CC} pins (V_{CCINT} and V_{CCIO}).

   All Xilinx CPLDs are designed for high speed. To perform that feat, sufficient internal current must be provided at all times. The best way to provide for that is with external decoupling capacitors that provide additional surge current when internal paths switch quickly. In addition, decoupling removes extra sources of noise contamination that can enter through the power supply rails, which are common to other fast switching circuits. Both high and low frequency decoupling is required at each V_{CCINT} and V_{CCIO} pin. Details to help selection are available in XAPP112 for the XC9500/XL/XV families, and XAPP377 and XAPP378 for the CoolRunner families. For best decoupling, direct connection between the V_{CC} and GND lines is advised, as near to the device as possible. Underside attachment, directly below the pins/balls is frequently the best site.

3. Use strict synchronous design.

   Most college textbooks on sequential design praise the merits of synchronous design. Yet, many designers fail to heed the advice. Synchronous design is essentially simple. With synchronous design, there is a single clocking event. In a synchronous design, there is either a rising edge clock, or a falling edge clock, but not both. There are no asynchronous sets or resets, except a possible initial global signal that occurs but once, at the beginning. P-term clocks, p-term sets and resets are not often found in synchronous designed circuits.

   That being stated, the situation often arises where there are two regions of logic that are independently clocked from free running clocks of different speeds that must be interconnected. The best way to handle that is by making sure that each region (separate clock domain) is strictly synchronous except where the two interchange information (usually data). For those interfaces, a separate synchronizing resource needs to be designed to handle the interchange. In FPGAs, this is almost always a FIFO structure with independent read and write clocks. Having no memory to construct FIFOs from means that CPLDs need to use tiny FIFOs constructed from registers or alternately simple exchange registers where the ability to write from one side and read from the other is maintained. In some cases, this may need to be done asynchronously. Nonetheless, if a circuit failure occurs, you know exactly where to look for the problem – at that interface!

   In general, do not freely intermix rising and falling clocks from the same source, and do not freely intermix clocks from different clock sources.

4. No floating input pins.

   Floating inputs increase current on CPLDs, as a rule. They can add one to two mA per floating input to your I_{CC} budget, and also channel noise right into the heart of the CPLD. There are a number of ways to simply terminate “floaters”. The most flexible is to use “user programmable ground” which is available on XC9500/XL/XV and CoolRunner-II as a software switch. External attachment to ground is also fine, if the pins will never be used in future edits. Pulling the pins high (through large pull-ups is also fine, but more expensive). The software switch is free and easy!

5. Sink current when driving LEDs.

   All Xilinx CPLDs have N-channel FET outputs to ground. Typically, N-channel transistors have greater drive strength than P-channel transistors. The N-channel transistors are designed to accommodate larger currents, which are usually associated with LED drive. Specific values are in the Xilinx data sheets, and details on driving LEDs are in **XAPP805**. Best results are obtained with the pin attached to the LED cathode and its anode series terminated to V_{CCIO} for that bank. This also provides the brightest LED display.

6. Terminate High Speed outputs.

   All Xilinx CPLDs are fast. They are designed to give quick reaction to any input. Even slow switching inputs are given a fast response. Series termination is the cheapest, so at least plan for it. This can be done by identifying sites for “zero ohm” impedance that can be a PCB trace, that can be removed and resistance inserted, as needed. It is harder to insert them, than to remove them. Basic details are given in **XAPP115** for XC9500/XL/XV and **XAPP379** for CoolRunner parts.

7. Avoid pull-down resistors on pins.

   All Xilinx CPLDs include additional circuitry on an I/O pin beyond just the I/O buffer. This includes ESD as well as circuits that manage power up behavior. For example:

   a. XC9500 has High-Z during power on

   b. XC9500XL/XV has High-Z during power on, then a keeper latch

   c. XPLA3 has High-Z during power on, then a keeper “half latch”

   d. CoolRunner-II has High-Z during power on, then a keeper latch

   Pull-down resistors “fight” the internal pin electronics, which may misbehave due to the external pull-down. For the most predictable behavior, avoid pin pull-down resistors.

8. Avoid driving I/Os above V_{CCIO}.

   Its not a good idea to drive a pin voltage above its bank’s V_{CCIO}. Current can reverse flow into the regulator and affect its performance, and other devices attached to it. Overdrive also increases net device leakage during the overdrive period. Under some circumstances, it can limit the life of the CPLD. The maximum drive limits are defined in the device data sheet, so be sure to observe them.

9. Remember to include JTAG stakes on the printed circuit board.

   JTAG stakes are needed to test the CPLD when it is on a printed circuit board (PCB). Without them, testing becomes very difficult, if needed. At least including PCB sites for stake insertion is good insurance to assess any future problem. The sites can be loaded with stakes for using a JTAG test probe (like an iMPACT cable, or Parallel Cable IV). This capability permits using JTAG INTEST and EXTEST to observe internal behavior and external connections. Think of it as a JTAG safety net for your design.

10. No external JTAG termination for internally terminated pins.

    IEEE 1149.1 explicitly states that a JTAG compliant device must have TDI and TMS internally pulled up by the device. TCK and TDO have no requirements to permit multiple testing topologies to exist. However, externally terminating TDI and TMS can be detrimental to the JTAG chain effectiveness, particularly, if you attach pull-down resistors to TDI and TMS. Pull-downs will tend to place the TDI and TMS pins biased toward the switching region and make them more sensitive to noise. That will also increase the current being drawn by the device. External pull-ups can also increase overall current draw, as they are in parallel with the resistors supplied within the CPLD, thereby reducing the overall resistance.

11. Assure sufficient regulator current is available.

    CPLDs need sufficient current during power on. Sagging regulators can misconfigure a CPLD and severely impact its behavior. Power estimation tools are for steady state

behavior, not the power on behavior. If you select your regulator by targeting a value calculated from your design (using XPower, a power estimation equation, or spreadsheet), you may choose too low of a value. Regulator selection is best done experimentally, for a specific design, and should be investigated during the design process. A reasonable approach would be to select a pin compatible regulator where a higher rated one exists. Then, verify clean operation as the design evolves. Extra available power will also permit future design changes.

12. Power V_{CCINT} before V_{CCIO} (if possible).

    Xilinx CPLDs won't be damaged under any power sequence, but there is the possibility that internally created signals can propagate to other circuits during power up. To minimize this possibility, one easy way is to apply V_{CCINT} first, complete the configuration and then apply V_{CCIO}. That way, power is not applied to the I/O bank until after the internal configuration process has completed. Further details on Xilinx CPLD power up behavior is contained in **XAPP440**.

Family Specific Hardware Practices

CoolRunner-II: Use Schmitt Triggers on Slow Moving Input Signals

Slow noisy signals can aggravate system designs. High speed CoolRunner-II CPLDs will accept glitchy, slow signals and forward them right into your system, where they may cause other problems. This is easily resolved by simply invoking the input pin Schmitt Trigger capability, which will "sweeten up" poor signals and forward nice clean edges on to other devices. The design software and architecture permit doing this on a per pin basis, and it is easy to experiment with as your design proceeds.

CoolRunner XPLA3: Understand JTAG Port Enable and Use It Correctly

Port Enable is a CoolRunner XPLA3 pin that permits using the four JTAG pins as I/Os versus JTAG operation. Should you not need the JTAG pins for programming or testing, the four pins can be reclaimed, but Port Enable should be used correctly. See **XAPP343** for details.

XC9500/XL/XV: Use Low Power Mode on all Noncritical Signals

As discussed earlier, all Xilinx CPLDs are fast. This includes the sense amp based XC9500/XL/XV parts. More robust designs that draw less power and have reduced noise can be simply obtained by using the low power macrocell switches in the design software. There is a slight speed adder to using the feature, so it should be used only for the logic that need not operate at the top speed. Often, that can be 80% of the whole design! The added benefit is less power, lower noise and a touch of hysteresis added onto the lower power macrocells. The design software can do this automatically, so it is very easy to experiment with.

Design Software Practices

At this point, we switch our emphasis to the Software Design Practice details.

1. Do not disregard report file warnings.

   The Xilinx design software identifies potential problems at compile time, for your benefit. A key text report file is the <*design*>.rpt file, which you are free to inspect and understand. It is also called the CPLD report file, and it contains a resource summary, all the design equations, and a summary of the constraints chosen for the compile. This is a good place to look at exactly how the software mapped your design onto the CPLD architecture. It also provides a summary of warnings that should be read and understood.

   Whether you created your design with a schematic or HDL, the Xilinx fitter modifies the resulting netlist into an architecture specific netlist and a JEDEC bitstream. Along the way, it observes choices made, and reports back areas that may be of concern with its list of warnings. By examining the warnings, you can determine whether or not they may be a

problem for your design. For best results, it is a good idea to understand the warnings and take appropriate action, if needed.

2. Use strictly synchronous design.

   You may also notice that this one occurred under hardware practices. It occurs under the software practice section because you can observe whether your design has inadvertently become “asynchronous” along the way. With ambiguity in HDL descriptions, and interpretation differences among them, it is possible to think a design will be synchronous and discover otherwise later! The CPLD report file is a good place to inspect the exact mapping of the logic onto the architecture. Specifically, users can observe how many global clocks are being used, where and when various asynchronous “p-term” resources are used, and so on. Be sure to check for “gated p-term clocks”, which are troublesome. These are all good indicators of what happened to your high level design as the various optimizations progressed. Best results are modified at the HDL level to improve the “synchronous” behavior of the final design. The warnings are also a “plus.”

3. Understand timing limits/warnings.

   The CPLD Timing Report also gives a great performance summary of your design. In a different format than the CPLD Report file, this one splits out your timing for each path within the design, in a spreadsheet format. A high level summary gives upper limits on speed, identifies bottlenecks and summarizes warnings, so you may assess the damage or available extra speed available after the compile. Spending time understanding this report will have additional debug payoffs.

4. Review the CPLD Fitter Report equations.

   As mentioned earlier, the CPLD Fitter Report shows exactly how the design maps to the architecture. It tracks the JEDEC file. Seeing exactly how the architecture is handling important points within the design helps quickly isolate areas that need to be better understood or improved. Many times, just inspecting the summary of options at the bottom of the report shows key factors like low power usage, and an assortment of other simple switch settings in the design software.

5. Always configure I/Os properly (LVCMOS 3.3, LVCMOS 1.8, etc.)

   The CPLD doesn’t know what voltages you need to drive. Hence, you determine that by external connection, but also by informing the design software during your design. The software will set internal programming bits that will dictate the correct transistor configurations at the I/O pins, to get the proper drive at the pin. This is the designer’s responsibility, so be aware that it is a choice you must make.

6. Let the design software define the pinouts, if possible.

   Xilinx design software is written to make the best possible assignment of I/O pins on your behalf. What this means is that it will intentionally cluster logic within function blocks, to maximize reuse of signals. It will also attempt to avoid overusing function block inputs and logic resources, so there will be available resources to support future design changes, if needed. This is particularly critical for designs that will remain in “the field” for years, and possibly sustain many revisions as requirements change. Should you decide to override the software and make your own assignments, the responsibility to assure future editing capability rests on your shoulders. Remember that this design software has at least twelve years of many experienced developers behind it (at this point in time). Doing a better job than that team may take some skill!

7. There is solid value to design simulation.

   Design simulation offers the ability to assess your design under a model of your expected workload. It can identify timing issues that are missed by the timing and fitter report files, which only have a “static” model of the design. It can also be used to vary the environmental conditions. For instance, when running a design at cold temperature, the device tends to run faster. When running at hot temperature, it tends to run slower. It is possible to choose different speed grades for a given part, and run simulation with the

different speed grades to determine if a trend in the design behavior may be present at a model of “hot” or “cold.” Asynchronous behavior can often show new operation when simulated that way. In general, simulation can be a lot of work, but it is often less work than debugging something poorly designed.

Conclusions

Bulletproof design is nothing more than careful advanced planning. Designs that can withstand variation in voltage, temperature, noise and multiple environmental variables and perform well are great designs. NASA put a man on the moon with a checklist, so we know that procedure works. Day to day designing with programmable logic is really no different. The race does not go to the “swift,” but rather to the “methodical.”

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Other CPLD Data Sheets, Application Notes and White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/28/05	1.0	Initial Xilinx release.