XAPP379 (v1.1) August 1, 2002

High Speed Design with CoolRunner-II CPLDs

Summary

This application note describes methods which will produce consistently fast designs when used with Xilinx CoolRunner™-II CPLD family. More detail on this important new family of 1.8V CPLDs can be obtained from the Xilinx website (**www.xilinx.com**), where the family and individual part data sheets can be found. Additional application literature is also available. Of particular interest will be **XAPP375**, which discusses the timing of the CoolRunner-II CPLDs, and **XAPP376**, which discusses the basic operation of the macrocell and function block—the "logic engine" of the CoolRunner-II family.

Introduction

Time and Frequency

High speed CPLDs have changed roles substantially in today's systems due to both the mixed voltage interfacing and the nature of high performance microprocessor systems. In the past, with five volt systems, processors typically had small on-chip cache memories to feed their instruction bandwidth needs. Fast, external random access frequently relied on rapid memory and I/O decoding to supply instructions or pass data. Zero wait state performance required high speed CPLDs. Today, most instructions are passed in large blocks or frames and much data is of a streamed video nature. Most processors can achieve their speed needs without extremely fast propagation delay, but mandate a need for very fast clock rates. High speed throughput outranks fast random access in most applications.

It makes an interesting aside to review the nature of speed from a clocking viewpoint on a digital device. Flip flop switching speed is typically limited by the sum of setup time (T_{SU}) and clock to output time (T_{CO}), assuming a zero hold time relationship can be met. This results in the expression:

$$F_{MAX} = 1/(T_{SU} + T_{CO})$$

This is a bare flip flop. In the programmable logic world, it would be frequently burdened by an adder (for routing a signal to the flip flop) which is typically added into the T_{SU} value when determining F_{MAX}. Additional logic delay driving the D input is also accumulated in the new T_{SU}. Let's explore the sensitivity of the F_{MAX} expression:

(1) If $T_{SU} = T_{CO} = 5.0$ nsec, then $F_{MAX} = 1/10$ nsec = 100 MHz.

(2) If $T_{SU} = T_{CO} = 4.0$ nsec, then $F_{MAX} = 1/8$ nsec = 125 MHz.

(3) If $T_{SU} = T_{CO} = 2.0$ nsec, then $F_{MAX} = 1/4$ nsec = 250 MHz.

(4) If $T_{SU} = T_{CO} = 1.0$ nsec, then $F_{MAX} = 1/2$ nsec = 500 MHz.

In case (1), the clock cycle of 10 ns gives us an F_{MAX} of 100 MHz. Trimming this cycle by 2 ns as shown in case (2) gives us a cycle of 8 ns and a corresponding F_{MAX} of 125 MHz. In this instance, a change of 2 ns improved the maximum frequency by 25 MHz. Case (3) has a clock cycle of 4 ns, which gives us an F_{MAX} of 250 MHz. Case (4) then trims 2 ns off case (3), resulting in an F_{MAX} of 500 MHz. In this instance, a change of 2 ns improves the maximum frequency by 250 MHz, ten times better than the improvement made from case (1) to case (2). This demonstrates that even small improvements in T_{SU} will have a very significant effect upon F_{MAX} as you approach the lower limits.

T_{CO} is typically a flip flop specific "hard" specification, so much of the speed optimization for programmable logic is focused on reducing the setup time, T_{SU}, which is somewhat adjustable. Finding ways to shave a few picoseconds off key paths results in high performance designs. This fact will drive most of our high speed recommendations. The remaining recommendations will be driven by methods that preserve these gains with appropriate external signal conditioning.

Fast Combinational Logic Design Techniques

As suggested earlier, a key trick will be to identify signal paths within the CoolRunner-II CPLDs that reduce small time segments. Figure 1 shows the macrocell that is used on all CoolRunner-II CPLDs, and is the basic building block of all logic paths through the chip. Signals originate at input pins and the design software automatically assigns the paths needed to pass through the macrocells (entering the Advanced Interconnect Matrix –AIM), creating/encountering logic gates along the way. To simplify our explanation, we will strip away sections of the macrocell to expose the points of interest for various design techniques. Most techniques will use the product terms to form some kind of logical expression. We will look first at some key combinational functions, then some important sequential ones.

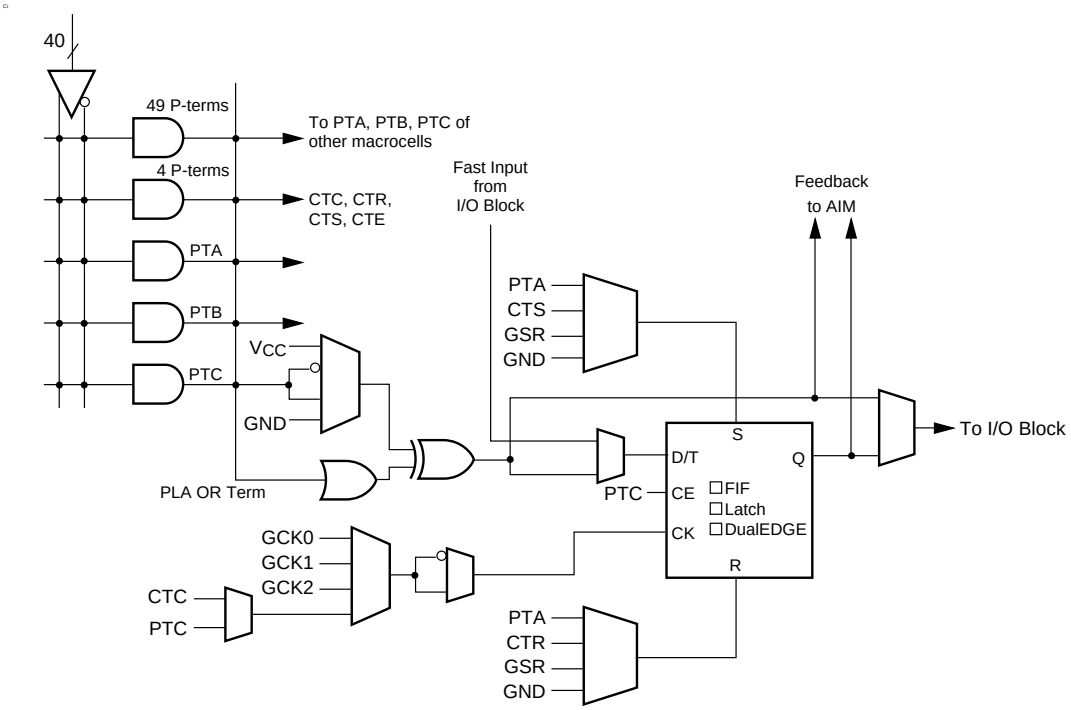

Figure 1: **CoolRunner-II Macrocell**

Figure 2 strips away the flip flop and several multiplexing circuits to highlight the Programmable Logic Array (PLA) and the product summing OR gate as well as the EX-OR gate which is valuable for counters, adders, comparators and parity circuits.

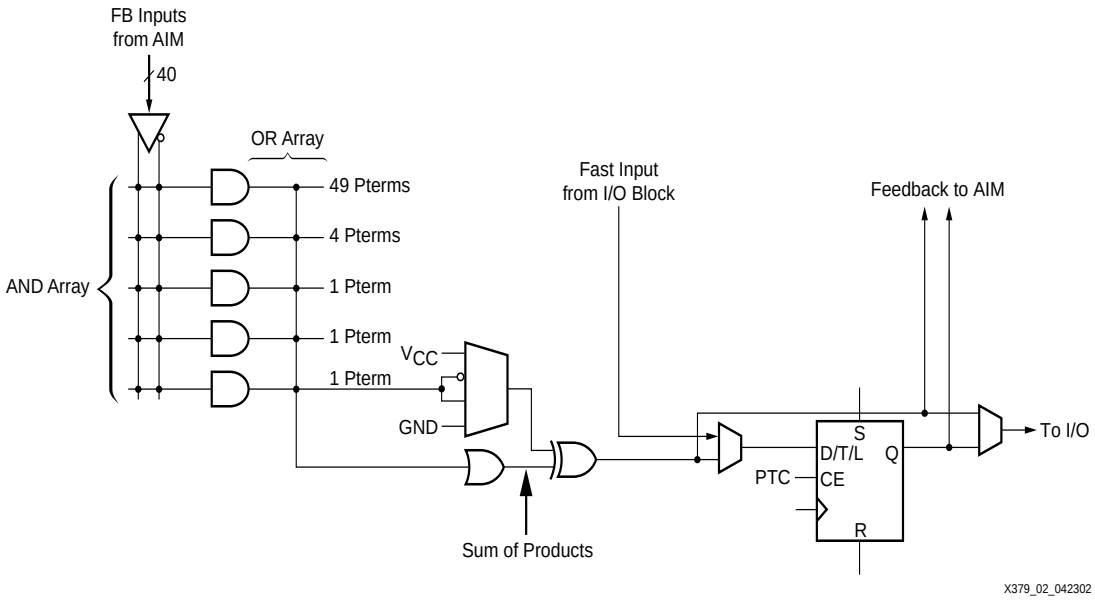

Figure 2: **Partially Stripped Macrocell**

Figure 3 further strips Figure 2 down to just two input legs of the OR gate, the first with a single product term assigned to it and the other driving two out of four inputs to a multiplexor that subsequently drives the other EX-OR input. The bold path is critical for high speed combinational signals.

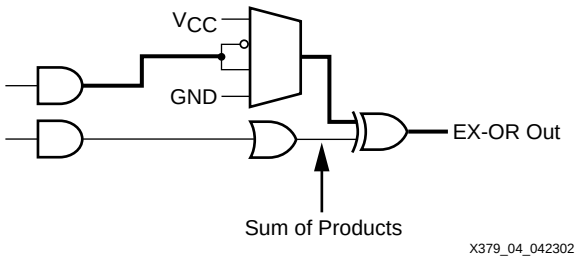

Figure 3: **Futher Exposed Macrocell Internals**

High Speed Decode

As mentioned earlier, microprocessors frequently decode modules of memory that exist external to the processor. This would be done using the bold path in Figure 3. The Function Block (FB) in CoolRunner-II can accommodate up to 40 input signals, which easily covers the standard 32 available in many of today's fast microprocessors. The additonal eight input signals can be direction qualifiers (R/W), Mem/I/O qualifiers, control strobes or clock phasing signals. In many systems, it is not required to resolve address selection down to one byte out of four gigabytes of the address space, so even more input signals are typically available. However, it should be noted that the CoolRunner-II can easily decode one byte out of four gigabytes when needed.

To perform the design, the user simply writes the combinational expression for an output signal in the chosen HDL: VHDL, Verilog or ABEL. The software does the rest.

Fast Equality Comparators

Combining both paths shown in Figure 3 permits a fast compare, where one operand is registered. What we seek to accomplish here is the creation of a signal that will be asserted when the value of a register is compared against several dynamic signals of the same number.

A standard CPLD approach would be to do bitwise compares and AND the respective bitwise compares into a combined signal, where any mismatch will create a logical zero and force the AND to be zero. All individual bitwise compares should create a logical one when they match. Any or several miscompares will create logical zeroes.

As a review, the EX-OR creates the complement of a bitwise compare in its standard configuration. Remember: A ⊕ B = A*/B +/A*B. The complement of this is: A*B + /A*/B. This can be accomplished with an EX-OR in two ways. First, the EX-OR can have its output complemented. This is the method we will show. The other way is complementing one input to the EX-OR, which produces the EX-NOR or “coincidence” function.

To achieve the fastest compare for a dynamically changing signal, the dynamic signal should be assigned to the bold path on Figure 3, and the static registered value will pass through the nonbold path on Figure 3. Either leg can be inverted to produce the appropriate bitwise compare signal. If the registered path is inverted, the signal occurs when the register feeds back through the AIM into the FB input signal polarity selection. Alternately, the registered value can arrive with intact polarity and the dynamic signal can pass along the bold Figure 3 path but take the inverted entry through the multiplexor before driving the top leg of the EX-OR. This results in one bitwise compare.

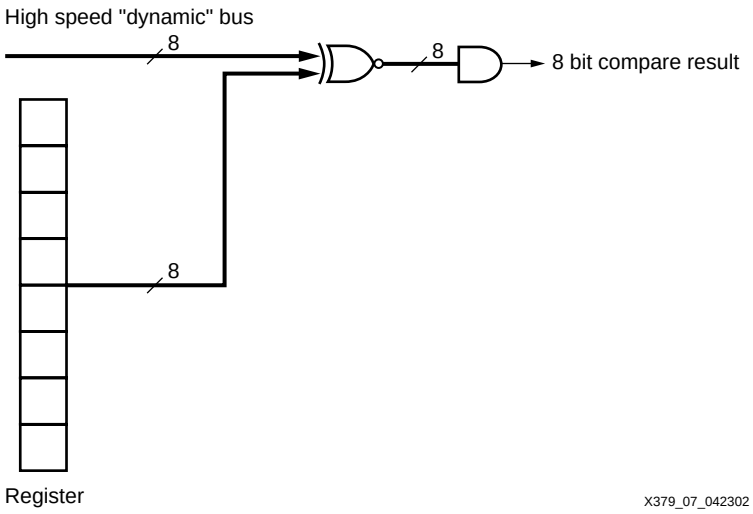

Figure 4: **High Speed Equality Comparator**

Expanding the idea just described, it is simply a matter of copying the circuit the required number of times, and ANDing the bitwise compares (Figure 4). Now, the AND can occur in two ways. If the signal is to reside within the CPLD, another pass through the AIM and a product term will be needed. Here, the only restriction for fast compare is the number of available inputs into the FB. If the compare signal is needed externally, it can be done by Wire-ANDing the bitwise compares at the pins which arrive through Open Drain output configurations; a pullup resistor does the rest. Small pullup resistors (1-5K) are typically faster than large ones, but they also pull more current: the classic speed/power tradeoff comes into play here.

Fast State Machines

State machines present an inherent speed limit due to the addition of feedback time delay and clock to output time to their flip flop setup times. The simplest state machine is probably the toggle (T) flip flop, which can operate at 416 MHz on the fastest CoolRunner-II parts. In this situation, there is no feedback and the setup time is always met, so the only roadblock is the clock to output time for the T flip flop. Toggle flip flops have value in that fast external clocks are frequently driven into CPLDs where they need to be divided down for other uses both inside and outside the CPLD. (This is one reason Xilinx CoolRunner-II CPLD datasheets also specify the F_{TOGGLE} value).

Shift Registers

Shift registers minimize the logic attached to the input of the flip flops, with a single product term being needed for simple connection between consecutive shifter bits (ie, Q_i to D_{i+1}). Given that the simple shifter will need to have only a single p-term for connection, it might as well be the fast one that bypasses the PLA OR term. This structure works fine for the serial in/serial out or serial in/parallel out shifter. It can't be used easily for the parallel in/serial out structure. Figure 5 shows connecting a couple of consecutive bits of a standard serial in/serial out shifter.

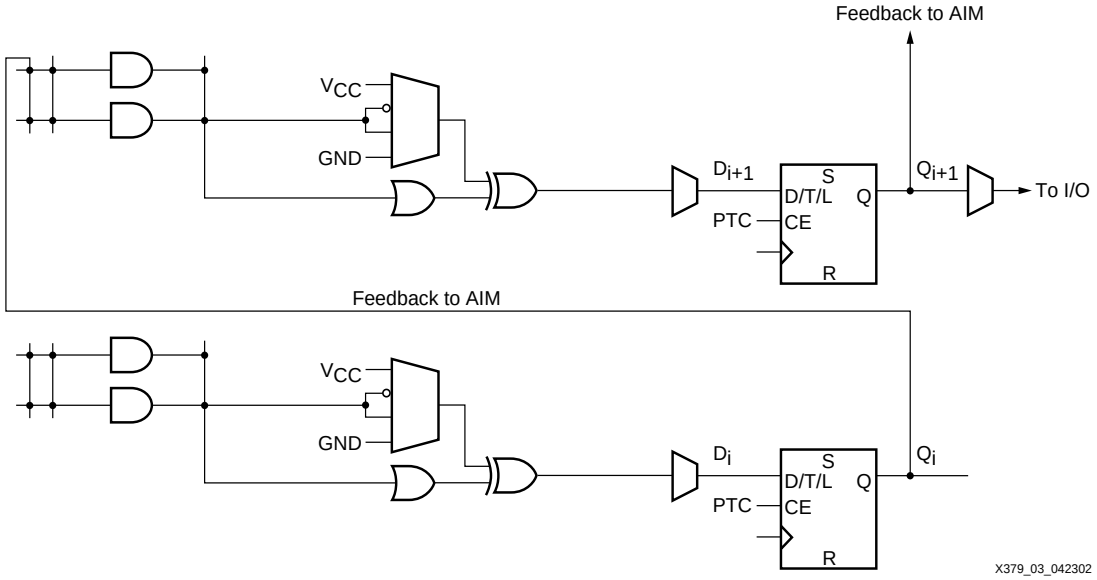

Figure 5: **Simple Shift Register**

Now that we have looked at a simple shift register and understand that the connection penalty of using a single p-term is a requirement for connection, let's see how this structure can be sped up by using a double data rate (DDR) shift register and some advanced CoolRunner-II features.

Double data rate operations have one data bit available on the rising edge of a clock and another available on the falling edge. This suggests using both edges of the clock to drive the shifter. One way to do this can be seen in Figure 6, where we dispense with showing the sum of products logic and simply show the overall result of tying consecutive flip flops together without exposing the underlying logic. In Figure 5, data arrives on the left through two input pins attaching to the fast input sites on the entry flip flops. Note that each is clocked by the opposite phase of a global clock. The flip flop outputs then feed a shift chain where the top half of the chain is clocked by the rising GCK edge and the bottom half is clocked by the falling edge of the same clock. This causes data arriving on the left to be interleaved between the top and bottom corresponding bits of the shifter. As the data sits in the shifter inside the CPLD, it has used the fast input feature and clocks to successfully split out the time interleaved aspect.

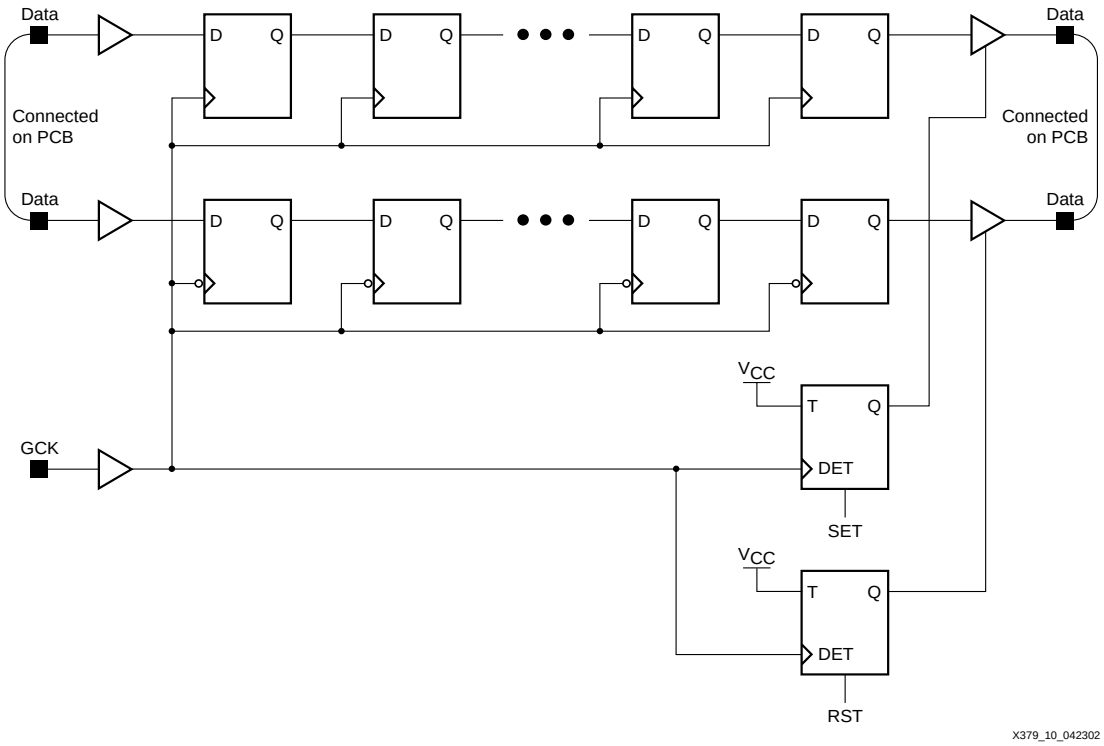

Figure 6: **Double Data Rate Shifter—One Version**

Proceeding to the right, we will re-interleave the data at the output pins by driving two separate tri-state buffers and connecting them on the printed circuit board. This interleave operation is accomplished by driving two tri-state nets from two toggle flip flops operating in the "dual edge triggered" mode. Note that the T flops are initialized out of phase with each other so there is no initial or subsequent collision. This capability is unique among CPLD products and results in the upper speed limit being that of the T flip flop toggle rates. Other versions of this design are currently being explored for speed, pin and power tradeoffs.

Binary Counters

High speed counters can be built up using just one product term per counter bit, which leaves their performance limited only by the time delay of the feedback path and the flip flop restrictions. Figure 7 shows this basic structure, demonstrating that each succeeding bit will grow in AND gate width as the counter size grows.

It should be kept in mind that the simplicity of the counter is a key factor in maintaining its speed. If the ability to load the counter, or to make it count up or down, is added, then its performance level will suffer from the added delay of passing signals through the sum-of-products OR term in the PLA.

The counter style shown in Figure 7 is appropriate in many counter applications where a simple large or small counter is needed. Note that the width of the AND gate grows to accommodate all lower order bits into the switching term for the current bit being designed. Figure 8 shows the basic CoolRunner-II CPLD AND gate, which is 40 inputs wide. For counters going up to 40 bits long, this is an appropriate style with no speed reduction as the counter grows to that limit. As the bit limit extends and an additional AND structure is built by stacking AND functions,however, a timing penalty will occur.

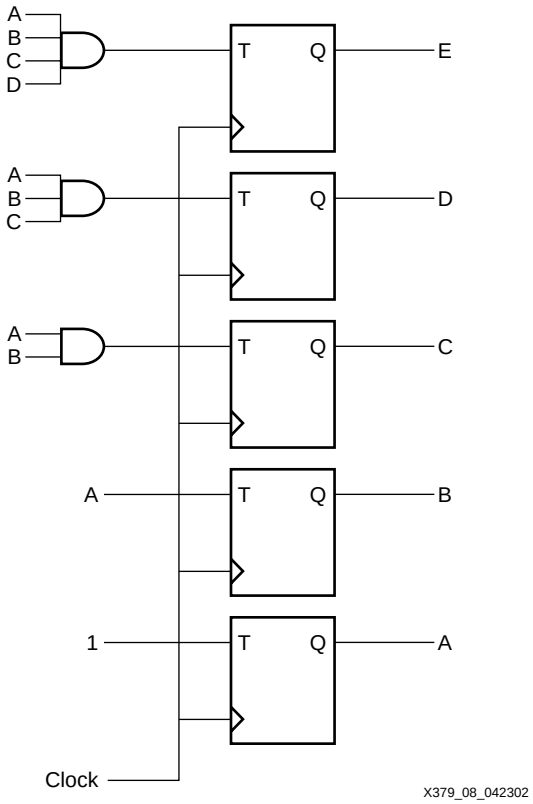

Figure 7: **Basic High Speed Counter Structure**

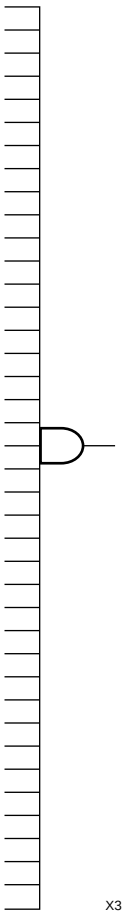

Figure 8: **40-Input CoolRunner-II Product Term**

Fast Johnson Counters

Returning to the theme of high speed counters, but with a different set of restrictions, we have the Johnson counter (sometimes dubbed the “moebius” counter). These counters operate at the shift speed of the CPLD, which is the upper FMAX limit, but deliver a strange “count” sequence in that the sequence is not binary consecutive. In practice, it is typically operated from a specific starting case by initializing with the asynchronous inputs. It will produce only 2N states as opposed to 2^N (where N is the number of flops). However, given its speed, for small counters that may be acceptable.

Software Considerations

In general, the software will assign these high speed paths automatically unless overriding restrictions, such as utilizing flip flops that employ the CE input, forbid it to do so. The CE input is seldom needed in CPLD designs, and in CoolRunner-II it is driven by the same product term that is tied to the faster leg on the EX-OR gate shown in Figure 1. Be aware that XST tends to create CEs, so turning this off in the fitter options is probably the best approach when seeking higher speed solutions. Also, because the default fitting option typically optimizes for density, it may be necessary to select the high speed option to ensure the fastest possible results.

Conclusions

Not all aspects of a design must be fast for the overall operation to meet its speed targets. This application note restricts its scope to common bottleneck functions that are readily handled by CoolRunner-II CPLDs. By approaching a design module by module and evaluating which particular modules need to be speed-streamlined in order to achieve the design’s overall speed requirements, the design can be speed-optimized both successfully and efficiently. Additional

high speed techniques will be discussed in future versions of this application note, so please check back for additions and revisions on the Xilinx website. More detail on low power design with CoolRunner-II CPLDs can be found in XAPP377 and detail on using the advanced features with the software can be found in XAPP378.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/15/02	1.0	Initial Xilinx release.
08/1/02	1.1	Minor changes.