Using Technology-Independent Intellectual Property *Are You Ready for 2*

By the year 2000, Xilinx will be producing devices containing more than 100,000 logic cells (2 million gates). You will soon have a canvas so broad that it will be difficult to paint all the landscape. We estimate that it will take a full 12 months and 20-30 engineers to fill a device of this density, if designing from scratch.

Clearly the solution to maintaining a trouble-free design cycle is to stop creating every design from scratch, and to start using technology-independent intellectual property. Just as it's easier to paint by numbers, it's easier to re-use blocks of logic. The use of intellectual property or core-based design modules is an essential component of a high-end "system on a chip" solution. The LogiCORE and AllianceCORE modules from Xilinx offer proven, pre-implemented, and fully verified cores that provide the fast time-to-market solutions you need.

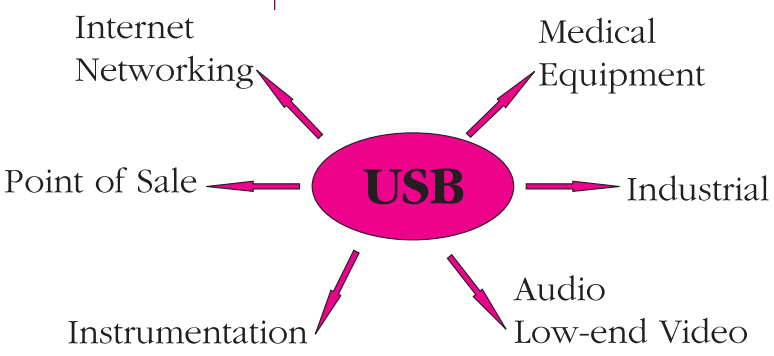
This article demonstrates the use of technology-independent intellectual property (IP) through the design of a Universal Serial Bus application.

Universal Serial Bus Case Study

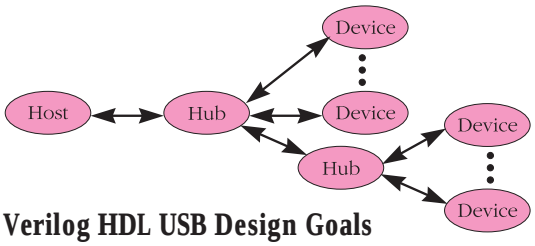
The Universal Serial Bus (USB) protocol was created to provide a standardized serial bus to be used in the personal and mobile computer markets. Just as PCI is becoming a standard parallel bus, USB is now becoming a standard PC serial bus for lower-bandwidth PC peripherals such as mice, keyboards, modems, and so on.

USB Protocol Description

The USB protocol uses a differential-twisted shielded pair for its physical medium. The signal coding is NRZI with bit stuffing, and has been designed to transmit data at two rates: 1.5 Mbps (low speed) and 12 Mbps (full speed). It can support up to 127 devices.



The USB architecture defines a host PC and "devices" or "functions" (such as keyboard or mouse) with "hubs" in the middle as necessary for fan-out.



Verilog HDL USB Design Goals

Mentor Graphics Inventra, a member of the Xilinx AllianceCORE partnership, has developed a family of USB functions and hub controller soft cores. These cores can be combined with application-specific back-end logic.

Inventra USB cores come in low speed and full speed versions for applications such as micro-controllers, audio, and generic user definable USB interfaces. These cores have been designed with the following goals in mind:

- A technology independent design methodology.
- Design implementation and mapping directed by synthesis timing constraints.
- Synthesis tools select state machines.
- Ability to re-use the bus interface with application-specific logic.

For this USB case study, an Inventra USB function controller core was selected. The USB function controller's hierarchy and logic was originally designed with its full-speed timing characteristics in mind. The USB function controller's hierarchy is illustrated in **Figure 1**.

While the full-speed USB Function Controller runs at 12 MHz, the design involves several blocks that run at 4X the basic rate, or 48 MHz. This 48 MHz clock is

Million Gates?

used to over-sample and drive both the data and the 12 MHz data clock from the serial NRZI USB data signal.

Controlling Timing

The design's logic was divided into different timing blocks as illustrated in **Figure 2**. This was done to understand the timing relationships both within and between blocks of logic.

By implementing the design using these timing blocks as the hierarchical boundaries, the Inventra engineers where able to clearly define timing specifications and constraints for the blocks of logic that needed to run at a rate of 48 MHz and those that needed to run only at 12 MHz. They were also able to control the timing interaction between these blocks.

By using the Xilinx Alliance series timing constraint capability, the global timing of 12 MHz was applied to the entire design. Then the more critical constraint of 48 MHz was applied to the flip-flops and I/O pads that needed to run at the 4X rate.

State Machines

The synthesis tools were also used to select the most efficient and highest performance state machines. The USB function controller consists of, among other things, five state machines. The Inventra engineers found that two of the state machines would not meet the system timing requirements if standard encoding methods were used for synthesis. By directing the synthesis tools to use "one hot" encoding for these two machines, performance requirements for these two blocks were met. To maintain the core's design re-use capability, the Verilog RTL code was not modified; only the synthesis options were changed.

Module Re-use

By developing the USB function controller as a reusable IP, Inventra is able to maintain a single version of RTL source code with a range of interfacing options for various back-end applications. This back-end application logic can also be developed using this technology independent methodology, allowing the modules to interface easily.

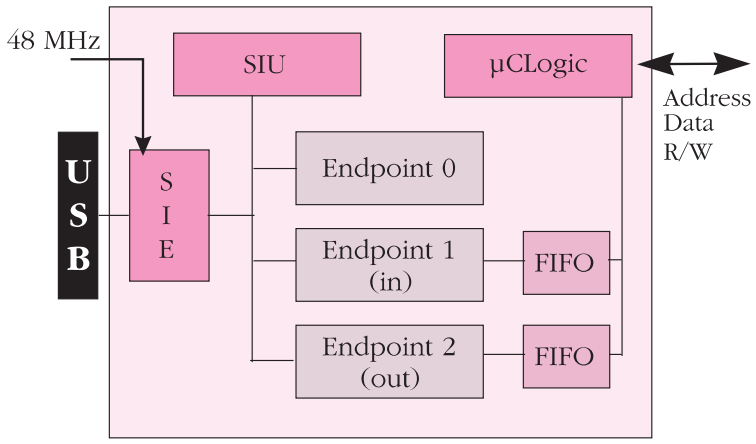


Figure 1:
USB Core Hierarchy

Summary

This USB IP development case study shows how generic Verilog RTL code was used to define the register transfer level functions while synthesis was used to produce the gate level implementation. The constraint files and directives are used to drive both the synthesis and the Alliance Series place and route tools.

In this technology-independent IP design flow, the Xilinx design tools used the new timing-driven placement and routing algorithms. The robust timing analysis tools were also used to verify that the timing requirements were met. There was no hand placement or routing, and no floor-planing was required.

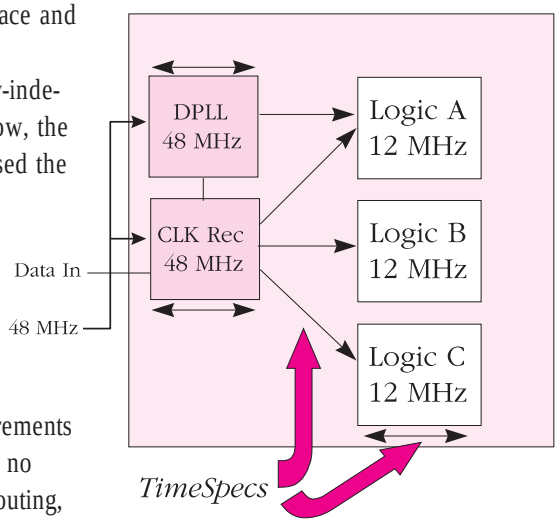


Figure 2:
Timing Blocks

The use of this technology independent methodology will become more and more important as device densities increase. Xilinx is committed to delivering not only the highest performance and highest density devices, but also to providing you with the tools you need to develop them quickly and easily.

For information on the Xilinx LogiCORE, AllianceCORE, or Mentor Graphics Inventra USB products, check out the Xilinx CORE Solutions web site at: http://www.xilinx.com/products/logicore/logicore.htm ◆