Understanding Parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs

Publish Date: Dec 06, 2011

Overview

This document is part of the

Multicore Programming Fundamentals Whitepaper Series

Multicore Programming Fundamentals Whitepaper Series

Parallel hardware is becoming a ubiquitous component in computer processing technology. Where uniprocessor once ruled, chip architectures have since evolved to allow engineers greater performance gains. Learn the difference between common parallel hardware architectures found in the marketplace today, including Multiprocessor, Hyperthreading, Dual-Core, Multicore and FPGAs.

Table of Contents

1. Multiprocessors

Multiprocessor systems contain multiple CPUs that are not on the same chip. Multiprocessor systems were made common in the 1990s for the purpose of IT servers. At that time they were typically processor boards that would slide into a rack-mount server. Today, multiprocessors are commonly found on the same physical board and connected through a high-speed communication interface.

Figure 1. The multiprocessor system has a divided cache and MMU with long-interconnects

Multiprocessor systems are less complex than multicore systems, because they are essential single chip CPUs connected together. The disadvantage with multiprocessor systems is that they are expensive because they require multiple chips which is more expensive than a single chip solution.

2. Hyperthreading

Hyperthreading is a technology that was introduced by Intel, with the primary purpose of improving support for multi-threaded code. Under certain workloads hyperthreading technology provides a more efficient use of CPU resources by executing threads in parallel on a single processor.

A hyperthreading equipped processor pretends to be two "logical" processors to the host operating system, allowing the operating system to schedule two threads or processes simultaneously. The advantages of hyperthreading are improved support for multi-threaded code, allows multiple threads to run simultaneously, and provides an improved reaction and response time.

Pentium 4 processors are an example of CPUs that implement hyperthreading.

3. Dual-Core and Multicore Processors

Dual-core processors are two CPUs on a single chip. Multicore processors are a family of processors that contain any number of multiple CPUs on a single chip, such as 2, 4, and 8. The challenge with multicore processors is in the area of software development. Performance speed-up is directly related to the how parallel the source code of an application was written through multi-threading.

Figure 2. The multicore processors share the cache and MMU with short interconnects

4. FPGAs

A FPGA (Field Programmable Gate Arrays) is a device that contains a matrix of reconfigurable gate array logic circuitry. When a FPGA is configured, the internal circuitry is connected in a way that creates a hardware implementation of the software application. Unlike processors, FPGAs use dedicated hardware for processing logic and do not have an operating system.

A single FPGA can replace thousands of discrete components by incorporating millions of logic gates in a single integrated circuit (IC) chip. The internal resources of an FPGA chip consist of a matrix of configurable logic blocks (CLBs) surrounded by a periphery of I/O blocks. Signals are routed within the FPGA matrix by programmable interconnect switches and wire routes.

Figure 3. FPGA allows the user to program gates into parallel hardware paths

Since FPGAs are simply huge fields of programmable gates, they can be programmed into many parallel hardware paths. FPGAs are truly parallel in nature so different processing operations do not have to compete for the same resources. Programmers can automatically map their solutions directly to the FPGA fabric. It allows the user to create any number of task-specific cores that all run like simultaneous parallel circuits inside one FPGA chip.

Hardware execution provides greater performance and determinism than most processor-based software solutions. The parallel nature of the logical gates on the FPGA allow for very high throughput of data. Once the code is compiled and running on the FPGA it will run without the jitter associated with software execution and thread prioritization typical to most common operating system and even present to a much smaller degree in real-time operating systems.

For more information, visit:
Optimizing your LabVIEW FPGA VIs: Parallel Execution and Pipelining

5. How LabVIEW Programs Parallel Hardware

LabVIEW has been multi-threaded since version 5.0 introduced in 1998. The dataflow nature of LabVIEW allows parallel code to easily map to parallel hardware. Therefore, it is an ideal development language for targeting multiprocessor, hypertheaded, and multicore processor systems.

In the case of programming FPGAs, LabVIEW generates VHDL code which is automatically compiled to a bitstream that can target Xilinx FPGAs.

6. More Resources on Multicore Programming

Multicore Programming Fundamentals Whitepaper Series

- www.ni.com/multicore
- Overcoming Multicore Programming Challenges with LabVIEW
- Differences between Multithreading and Multitasking
- Why Dataflow Programming Languages are Ideal for Programming Parallel Hardware?
- Learn more about NI LabVIEW