

Combining PLC and FPGA architectures

Information from National Instruments

Machine automation systems have become very complex, in order to achieve better throughput, faster changeover times, reduced waste, and lower downtime. Today's automation systems require various functionality, such as predictive maintenance, intelligent fault handling, fast response times, or custom control algorithms, in addition to logic or process control.

This functionality is difficult to implement in traditional PLC-based architectures. Today's machines need a combination of embedded and PLC technologies to deliver the expected functionality and performance. This article examines how a PLC works and explores FPGA technology with its parallel processing architecture, as well as applications for high-performance machine control.

PLC scanning architecture

A PLC typically operates in accordance with the IEC-61131-3 specification. This specification, developed by the International Electrotechnical Commission, details the guidelines for the operating system execution, data definitions, programming languages, and the instruction set a PLC uses. Unlike in the PC world, where one C program works well on any computer, in the PLC world, every vendor has its own flavour of an IEC 61131-3-compliant specification. However, the basic architecture of the system remains the same.

To illustrate, let us take a quick look at how a PLC operates. If you want to toggle a LED every second, your PLC program has three components to help you do this: program logic, memory map, and I/O scan. For program logic, you write out a value TRUE to a variable you call "LEDValue." This variable resides in the memory map. The I/O scan then transmits a binary equivalent of the value in memory to the physical I/O line, turning on or off your LED. Thus, the memory map acts as the link between your program logic and the I/O scan.

Because your program logic and I/O scan are decoupled, you can do a variety of things, such as force I/O on to the memory map when debugging in the field or updating a program on the fly when the I/O scan is still running. However, you need to keep certain things in mind when programming PLCs, such as not having any blocking function that may cause the program logic to indefinitely halt.

Sequential and reactive programming architectures

The scanning architecture in Fig. 1 is excellent for controlling sequential processes – turn the conveyor on, if the proximity sensor goes high, then turn the conveyor off and turn it on again after the proximity sensor goes low. However, modern machines are more complex and more reactive in nature. A typical packaging machine today devotes 20 % of the program to the logic for normal operation and 80% for fault handling. It needs a combination of sequential and reactive programming architectures to accomplish control for such complex machines.

Programmable automation controllers, such as NI CompactRIO, incorporate embedded technologies, such as field-programmable gate arrays (FPGAs), for implementing reactive architectures (see Fig. 2). In addition to the FPGA, there exists a floating point processor running a real-time operating system you can use to program sequential tasks, such as those in a PLC. PACs combine the best of embedded and PLC technologies to offer an ideal architecture for programming complex machines.

What is an FPGA?

An FPGA is a chip that consists of many unconfigured logic gates. Unlike the fixed,

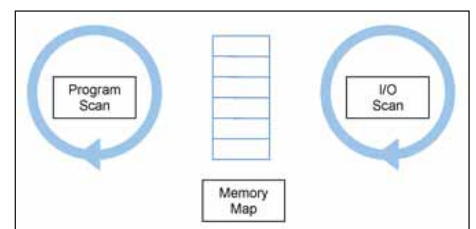


Fig. 1: Scanning architecture of a PLC.

vendor-defined functionality of an ASIC (application-specific integrated circuit) chip, you can configure and reconfigure the logic on FPGAs for specific applications. Engineers use FPGAs in applications where either the cost of developing and fabricating an ASIC is prohibitive, or the hardware must be reconfigured after it is placed into service. The flexible, software-programmable architecture of FPGAs offer benefits such as high-performance execution of custom algorithms, precise timing and synchronisation, rapid decision making, and simultaneous execution of parallel tasks. Today, FPGAs appear in a variety of devices, including instruments, consumer electronics, automobiles, aircraft, copy machines, and application-specific computer hardware.

The case for FPGA technology

While engineers often use FPGAs in industrial control products as an internal component, they

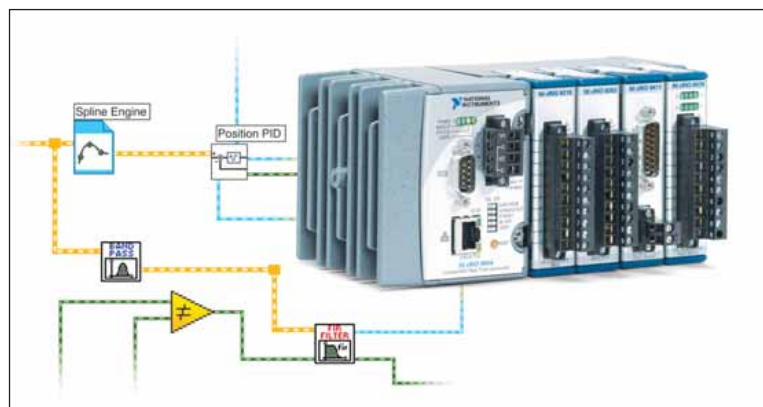


Fig. 2: Programmable automation controllers combine the best of both PLC and embedded technologies to solve complex machine automation problems.

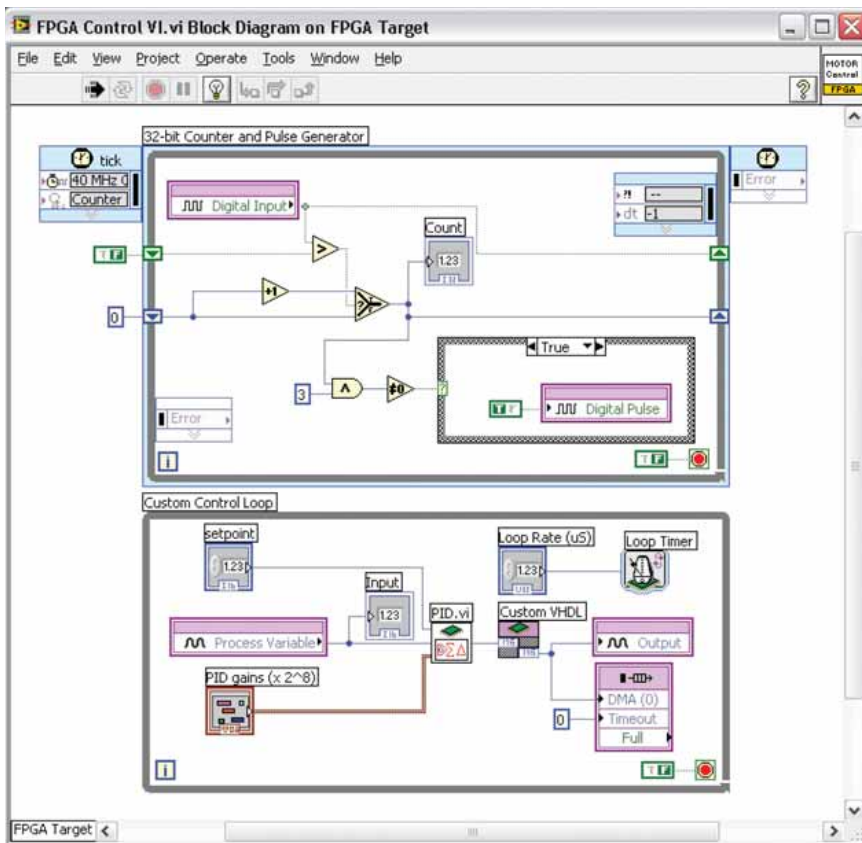


Fig. 3: Programming an FPGA is easy with the LabVIEW graphical development environment.

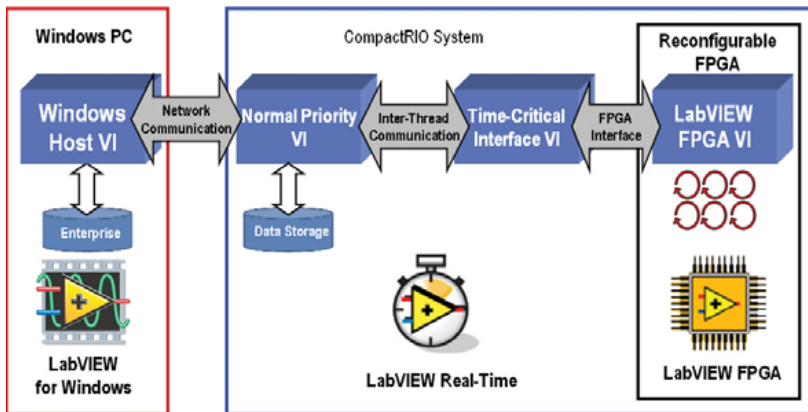


Fig. 4: Architecture of an FPGA-based PAC – NI CompactRIO.

have not had access to FPGA functionality. Defining FPGAs has historically required expertise using HDL programming or complex design tools used more by hardware design engineers than by control engineers. With PACs, such as CompactRIO, LabVIEW maybe be used - a high-level graphical development environment designed specifically for measurement and control applications, to create control systems that incorporate reactive behaviour with fast response rates. Because LabVIEW configures custom circuitry in the FPGA, systems can process and rapidly and deterministically generate synchronised analogue and digital signals, which you cannot achieve using the PLC execution architecture. With the FPGA, you can

execute digital logic at 20 MHz, analogue closed-loop control at 1 MHz, and achieve servo update rates of 200 kHz per axis for motion control.

What is LabVIEW?

NI LabVIEW is a graphical development environment engineers can use to design, prototype, and deploy high-performance monitoring and control applications with programmable automation controllers (PACs). LabVIEW helps engineers complement IEC 61131-3-based PLC systems by incorporating PC and embedded technologies into their applications and by providing HMI, SCADA, and enterprise communication capabilities. LabVIEW offers a

variety of programming architectures including finite state machines, discrete events, dynamic dataflow, continuous time and sequential logic – giving engineers the characteristics of continuous and reactive systems in the same application. Engineers can deploy LabVIEW programs to a variety of targets including FPGAs, real-time operating systems, Windows XP, WinCE, PalmOS, digital signal processors (DSPs), or any 32-bit microprocessor, depending on price-performance requirements.

Programming an FPGA-based PAC with LabVIEW

When you target an FPGA-based device, such as CompactRIO, Compact Vision System, or R Series PCI/PXI data acquisition devices, LabVIEW displays only the functions you can implement in the FPGA. The LabVIEW FPGA module functions palette includes typical LabVIEW structures and functions, such as while loops, for loops, case structures and sequence structures as well as a dedicated set of LabVIEW FPGA-specific functions for maths, signal generation and analysis; linear and nonlinear control; comparison logic, array, and cluster manipulation; occurrences; analogue and digital I/O; and timing. You can use a combination of these functions to define logic and embed intelligence onto your FPGA-based device.

Fig. 3 shows the block diagram for a 32-bit counter, pulse generator, and a PID control loop implemented in LabVIEW FPGA. Following the data flow makes it easy to understand the behaviour of this program. For example, to create the counter, you continuously read from a digital line and compare the line status from one iteration to the next. The counter increments by one when the digital line state changes.

Customising the counter or creating additional functionality is merely a matter of changing the LabVIEW code. Unlike traditional PC or PLC processors, FPGAs are parallel processors and adding additional loops to your application does not affect the performance of other independent loops. In this example, the counter and the PID loop execute in parallel on the FPGA – no multi-tasking as in PLCs or PCs.

The PID function block is just one example of intellectual property (IP) available for programming your FPGA in LabVIEW. For complex, high-speed monitoring and control applications, you can use signal processing IP from a variety of sources. The SoftMotion development module includes enhanced algorithms for motion control and built-in inputs for a variety of feedback devices including quadrature encoders. The digital filter design toolkit provides functions and interactive tools for design, analysis, and implementation of

digital filters within LabVIEW FPGA. Additionally, custom VHDL or Verilog IP can be linked into a LabVIEW FPGA block diagram using the HDL interface node.

Adding floating point capabilities to FPGA-based PACs

PACs, such as CompactRIO, have a combination of a floating point processor running a real-time operating system and an FPGA running logic in hardware. LabVIEW real-time can be used to create programs that execute on the floating point processor and communicate with the FPGA. The program running on the floating point processor can also be used to perform complex floating-point calculations, data logging, networking, email, file transfer protocol (FTP), remote web control, and any operations that do not fit within the FPGA fabric. LabVIEW real-time systems deliver deterministic processing engines for functions performed synchronously or asynchronously to the FPGA. For example, floating-point arithmetic, including spectral analysis or custom control algorithms are often performed in the LabVIEW real-time environment. You can store relevant data on a LabVIEW real-time system or transfer it to a Windows host computer for off-line analysis, data logging, or user interface displays. The architecture for this configuration is shown in Fig. 4.

Each FPGA-based device has flash memory available to store a compiled LabVIEW FPGA program and run the application immediately after power up of the device. In this configuration, as long as the FPGA has power, it runs the FPGA program, even if the host computer crashes or is powered down. This is ideal for programming safety power down and power up sequences when unexpected events occur.

Applications using FPGA-based PACs

When it comes to very high speed sorting, implementing custom control algorithms, such as model-free adaptive control, performing high-speed real-time analysis and control, or fast event response, FPGA-based PACs are an ideal choice for implementing machine control systems. For example, LabVIEW FPGA technology has been used to tightly synchronise the high-speed valve operation of a solid-state hydraulic actuator; it has been used to develop field-oriented control of a three-phase brushless permanent magnet motor; and FPGA-based CompactRIO has been used to acquire high-speed analogue data on multiple channels, run custom processing algorithms, and provide deterministic control to operate a verifiable fastener installation tool.

Incorporating embedded technologies, such as FPGAs, with parallel, reactive architectures effectively complement the sequential processing offered by PCs and PLCs for next generation automation systems.

Contact Richelle Glorioso, National Instruments,
Tel 0800 203 199,
ni.southafrica@natinst.co.za □