WHITE PAPER

FPGA

Five Ways to Build Flexibility into Industrial Applications with FPGAs

Intel® MAX® 10 and Cyclone® FPGAs help industrial designs adapt to changing requirements, and lower total solution costs.

Authors

Jason Chiang
Sr. Technical Marketing Manager
Intel Programmable Solutions Group

Stefano Zammattio
Product Marketing Manager
Intel Programmable Solutions Group

Table of Contents

Introduction

Programmable logic devices (PLDs) are a critical component in embedded industrial designs. PLDs have evolved in industrial designs from providing simple glue logic, to using FPGAs as a coprocessor. This technique allows for I/O expansion and off-loads the primary microcontroller (MCU) or digital signal processor (DSP) device in applications such as communications, motor control, I/O modules, and image processing.

As system complexity increases, FPGAs also offer the ability to integrate an entire SoC at a lower cost compared to discrete MCU, DSP, ASSP, or ASIC solutions. Whether used as a coprocessor or SoC, Intel® FPGAs offer the following advantages for your industrial applications:

- **Design Integration**—Simplify and reduce cost by using an FPGA as a coprocessor or SoC that integrates the intellectual property (IP) and software stacks on a single device platform
- **Reprogrammability**—Adapt industrial designs to evolving protocols, IP improvements, and new hardware features within one FPGA on a common development platform
- **Performance Scaling**—Enhance performance via embedded processors, custom instructions, and IP blocks within the FPGA to meet your system requirements
- **Obsolescence Protection**—Increase industrial product life cycles and provide protection against hardware obsolescence through long FPGA life cycles and device migration to new FPGA families
- **Familiar Tools**—Use familiar, powerful, and integrated tools to simplify design and software development, IP integration, and debugging

The following sections discuss these advantages in greater detail

Design Integration

Designers of modern industrial systems face many challenges, including system complexity, changing standards, performance requirements, and total system cost, as illustrated in Figure 1.

As a designer of industrial systems, you can determine whether to use an FPGA as a coprocessor (also referred to as an I/O companion or I/O hub device) or as a complete SoC solution. You can combine standard host processors with FPGAs on the same board, with the external host processor performing system processing. However, fixed function processors frequently lack the key interfaces, functionality, or performance for industrial applications. You can offload the processor by

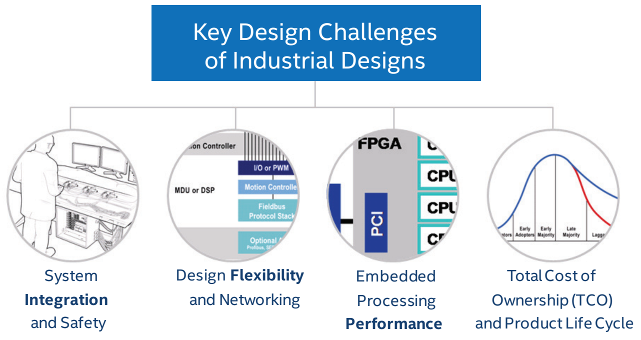

Key Design Challenges of Industrial Designs
System Integration and Safety
Design Flexibility and Networking
Embedded Processing Performance
Total Cost of Ownership (TCO) and Product Life Cycle

Figure 1. Key Industrial Design Challenges

moving some of the processing tasks inside the FPGA. Alternatively, you can integrate all processor functions on a single FPGA-based SoC platform to simplify design complexity and reduce overall system cost.

Many MCU or DSP solutions exceed bandwidth if required to run both motor control tasks and communications concurrently. As a result, you may be required to add another ASSP, MCU, or FPGA device, often as an expensive add-on card, if your current board cannot accommodate this extra device. Similarly, different fieldbus and evolving industrial Ethernet protocol standards may require a specific ASSP, MCU, or FPGA device for each protocol. This may be required because some standards require protocol-specific hardware (MAC) and a protocol-specific software stack. FPGAs allow you to integrate system functions in the coprocessor and to change your design as needed at any time.

Figure 2 illustrates a motion or motor control platform that takes advantage of Intel FPGAs as coprocessors for both the DSP offload engine and industrial networking. A motor controller sets the energy efficiency and accuracy of an electric motor through the control of speed and electrical current (translated into a torque setting). Similarly, motion control focuses on the precision of position and timing. In many cases, the electronic hardware is similar, and the control software, or algorithms, and I/O interfaces are the differentiating factors.

The example in Figure 2 shows a typical controller that depends on a primary MCU or DSP device (host processor) to run the algorithm, driving the power stage of a motor or motion controller. When the host processor reaches its performance limit, designers can increase the device clock speed to boost processor performance. However, there is a limit to the performance gains, and this method may also introduce other problems, such as the need to upgrade to faster memory, the performance of other hardware, and additional time required to optimize software.

In such cases, offloading some of the host processor functions to the FPGA coprocessor can provide relief, and using an FPGA for the communications provides you the flexibility to change with evolving standards like Industrial

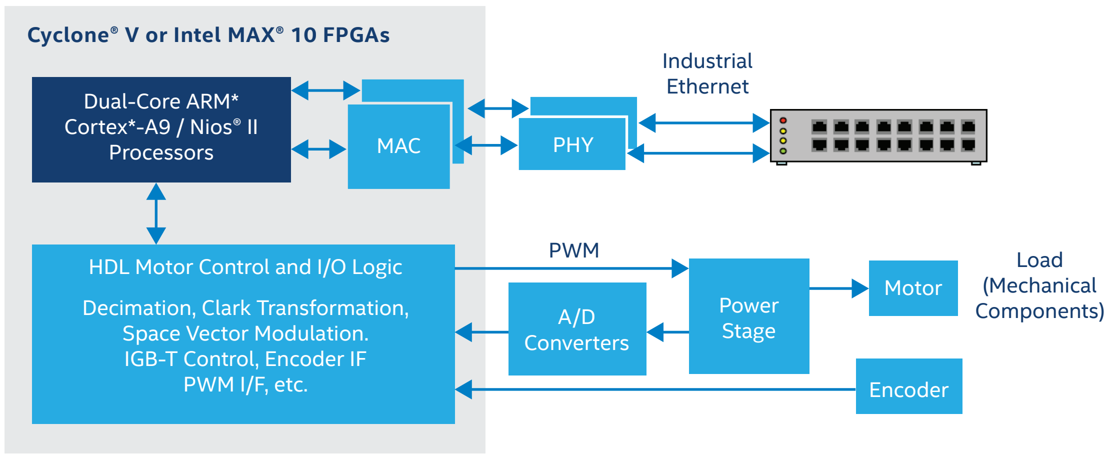

Cyclone® V or Intel MAX® 10 FPGAs
Dual-Core ARM* Cortex*-A9 / Nios® II Processors
MAC
PHY
Industrial Ethernet
HDL Motor Control and I/O Logic
Decimation, Clark Transformation, Space Vector Modulation. IGB-T Control, Encoder IF PWM I/F, etc.
PWM
A/D Converters
Power Stage
Motor
Load (Mechanical Components)
Encoder

Figure 2. FPGA as Motion/Motor Control Coprocessor

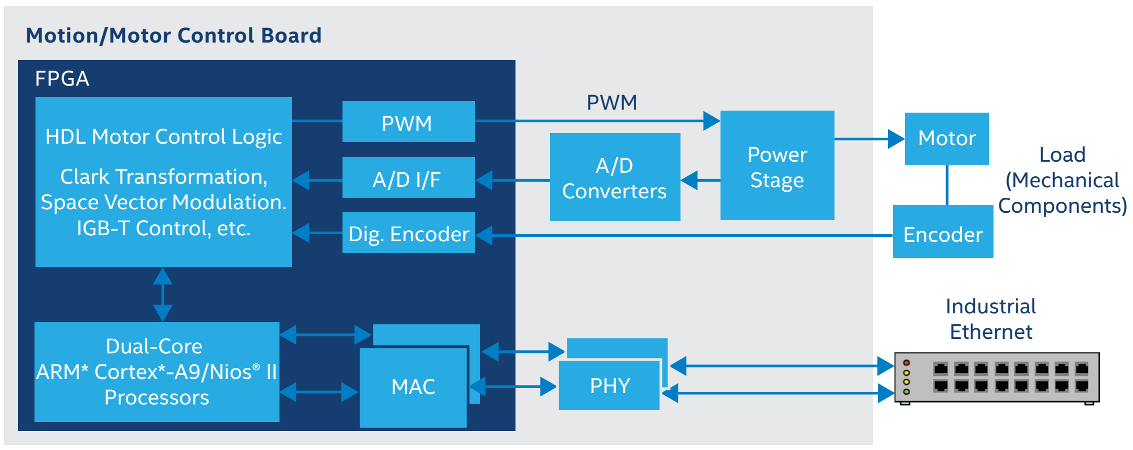

Motion/Motor Control Board
FPGA
HDL Motor Control Logic
Clark Transformation, Space Vector Modulation. IGB-T Control, etc.
PWM
A/D I/F
Dig. Encoder
PWM
A/D Converters
Power Stage
Motor
Encoder
Load (Mechanical Components)
Dual-Core ARM* Cortex*-A9/Nios® II Processors
MAC
PHY
Industrial Ethernet

Figure 3. FPGA as SoC Motion/Motor Control

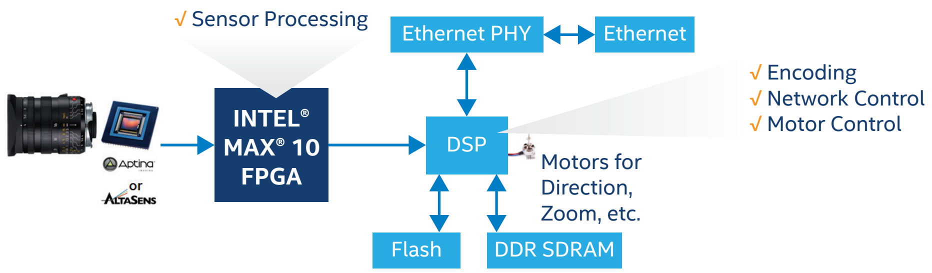


Figure 4. FPGA as Coprocessor—WDR IP Surveillance Camera

Ethernet protocols. You can then reprogram the FPGA and use the same hardware platform to meet your needs.

Integrating design components on a single SoC FPGA device platform further simplifies design complexity and reduces overall system cost. Figure 3 illustrates a simple industrial motor control system in which the FPGA now functions as the SoC, integrating DSP blocks, memory, video graphics controllers, motor encoders, and other components. You can simply add PHYs and other analog and power components to complete the design.

In addition, motor control applications often require a feedback mechanism to calculate current speed and position. Many optimized digital encoder interface IP cores are available only as IP for FPGAs, supporting the use of an FPGA for the interface. IP integration on the FPGA reduces board size, component count, assembly complexity, and stocking requirements. This integration increases system reliability with fewer components on the board. Intel FPGAs support many other system functions such as embedded processors, DSP blocks, LCD displays, and video processors.

Figure 4 illustrates another application example with the FPGA acting as a coprocessor in a video surveillance application. The video surveillance market is increasingly adopting wide dynamic range (WDR) camera sensors capable of distinguishing target objects from the background through adverse lighting conditions. Only FPGAs have the bandwidth to act as the coprocessor for the WDR image sensor pipeline (ISP), feeding the video stream to the DSP device for video encoding, such as H.264. A DSP device lacks the bandwidth and interfaces to handle WDR ISP, and lacks the performance to run additional surveillance functions, such as video analytics.

Alternatively, Figure 5 shows the FPGA as SoC in the video surveillance application. When used as a SoC, FPGAs enable you to integrate all components—ISP, video analytics, encoding, and networking —in a single FPGA device. This technique eliminates the need for a back-end DSP device, and provides a more compact and integrated design.

Figure 6 shows the block diagram for the example system implemented on the Cyclone® V SoC.

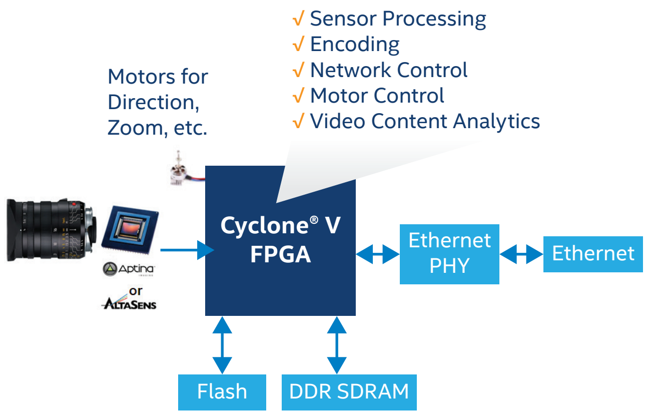


Figure 5. FPGA as SoC—WDR IP Surveillance Camera

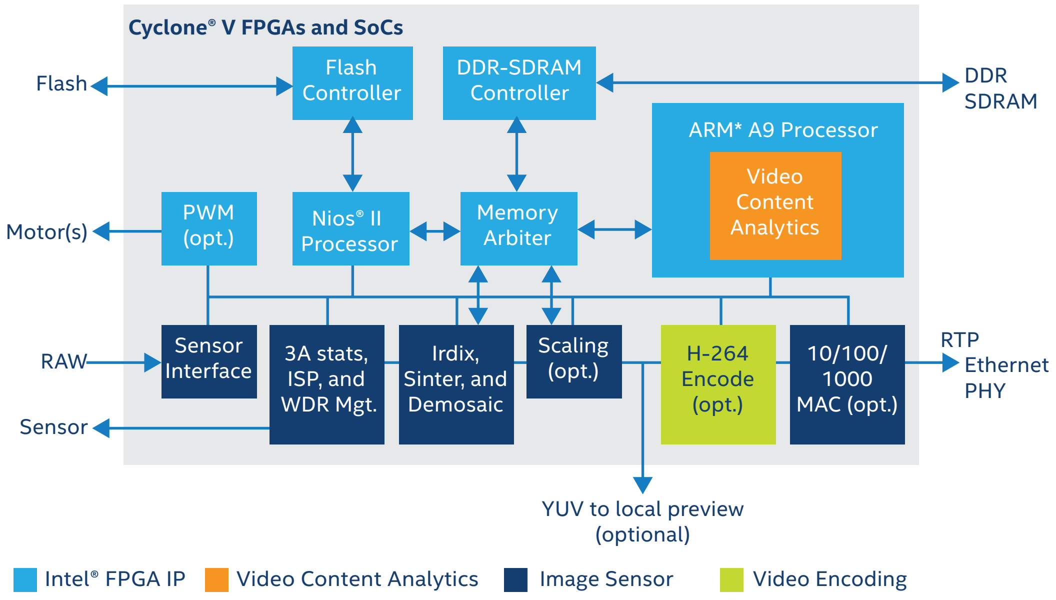


Figure 6. SoC Block Diagram—WDR IP Surveillance Camera

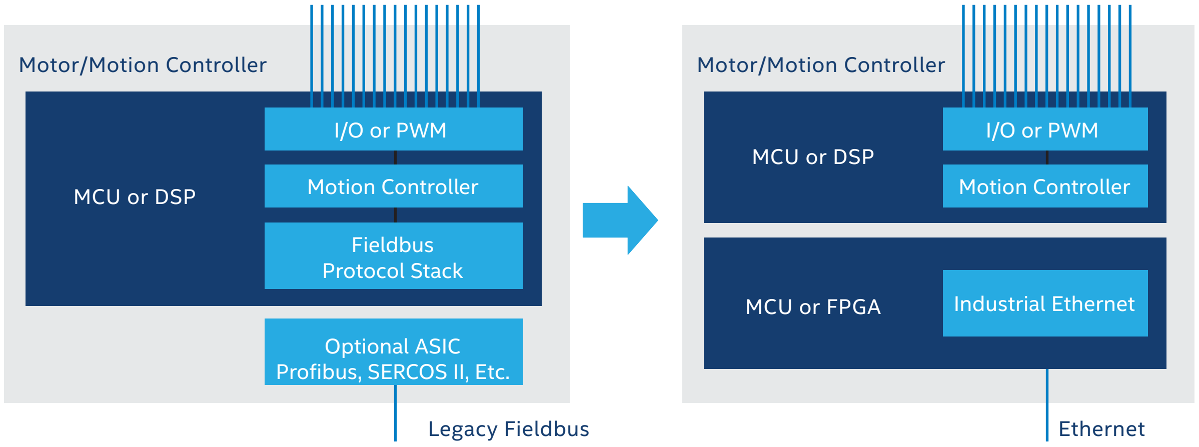


Figure 7. Fieldbus Migration to Industrial Ethernet

Reprogrammability

FPGA reprogrammability enables you to adapt to changing standards and support design reuse. Even if you target a single MCU, DSP, ASSP, or ASIC solution for your industrial application, many applications still require a separate device, such as an optional fieldbus-specific ASIC or FPGA, to handle features like industrial communications, as shown in Figure 7. When networking specifications or feature requirements change, you are often forced to create multiple PCBs to support different protocols and features, requiring additional software porting cycles on each platform. This can significantly increase the total solution cost.

Alternatively, you can use an FPGA as the communication coprocessor. You can design one communications subsystem, change the networking protocol at any time, and support multiple products on a single hardware platform. You can gain even more flexibility by integrating the main MCU or DSP control functions, multiple processors, and other IP and interfaces into a single FPGA design to create a smaller device footprint and save on space and cost.

With the ability to leverage one platform for multiple products, you can realize significant time to market advantages of several months or more because you have less hardware to develop and your software porting matrix is simplified.

Performance scaling

A key part of any industrial control system is the processing functions of a host/primary MCU, DSP, ASIC, or ASSP device. When performance is the design challenge, FPGAs provide the following ways to scale the processing performance, as illustrated in Figure 8.

- Use either a high-performance external processor along with one to multiple embedded processors inside the FPGA. You can also integrate all processing functions into the FPGA as the SoC.
- Add custom instructions in line with your processor code to accelerate specific processor instructions; floating point is a great example.
- Accelerate data transformation with application-specific hardware, like DSP blocks.

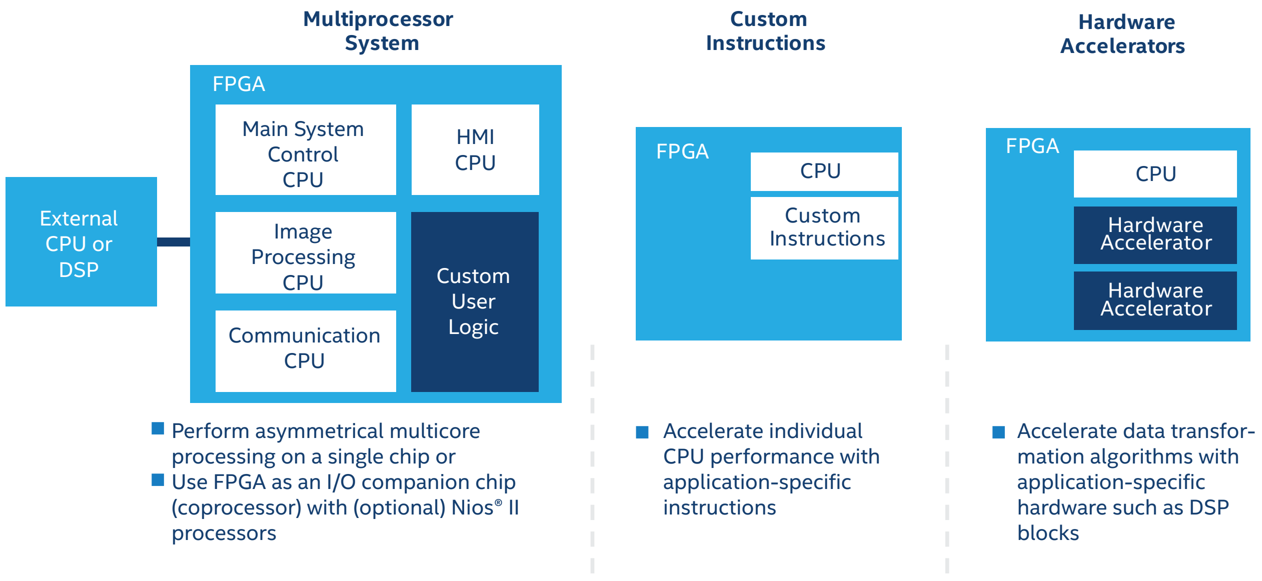

Multiprocessor System
FPGA
Main System Control CPU
HMI CPU
Image Processing CPU
Custom User Logic
Communication CPU
External CPU or DSP
Perform asymmetrical multicore processing on a single chip or
Use FPGA as an I/O companion chip (coprocessor) with (optional) Nios® II processors
Custom Instructions
FPGA
CPU
Custom Instructions
Accelerate individual CPU performance with application-specific instructions
Hardware Accelerators
FPGA
CPU
Hardware Accelerator
Hardware Accelerator
Accelerate data transformation algorithms with application-specific hardware such as DSP blocks

Figure 8. FPGA Performance Scaling Methods

Multicore processing

For flexible multiprocessor designs, you can choose from several implementations. Embedded industrial designers are commonly interested in asymmetrical coprocessing, with the FPGA as either the I/O companion chip or SoC. Asymmetric multiprocessing means that multifunction products can have a dedicated processor for each main function. This is especially suited to today's demanding applications, such as smart phones. Developers formerly built systems such as this with multiple processors on the PCB. Now you can accomplish this with dedicated processing blocks partitioned within a single FPGA, as shown in Figure 9.

An example of this type of application is a high-performance servo drive that requires a primary, high-performance processor (or multiple processors) to perform each main function. A dedicated processor executes the application code, a communications processor provides the fieldbus or Ethernet link, a graphics or image processor provides the display, and includes other custom logic and interfaces like digital motor encoders, PWM functions, and power control. You can integrate all of these functions into the FPGA either as a coprocessor or complete SoC.

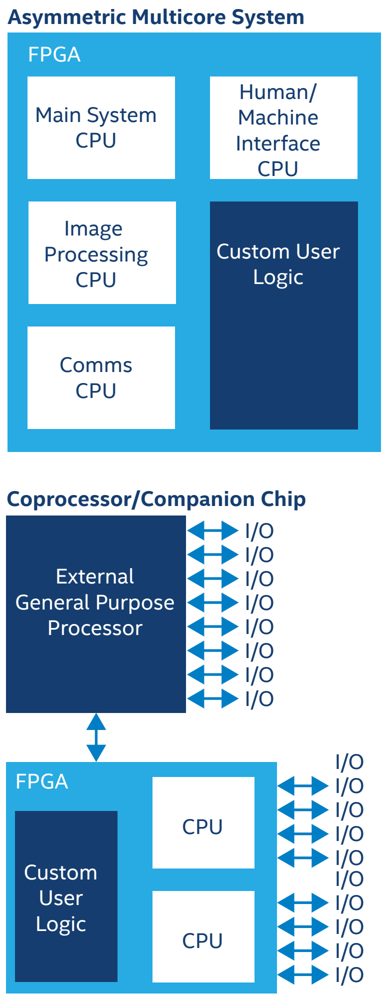

Asymmetric Multicore System
FPGA
Main System CPU
Human/ Machine Interface CPU
Image Processing CPU
Custom User Logic
Comms CPU
Coprocessor/Companion Chip
External General Purpose Processor
I/O
FPGA
Custom User Logic
CPU
CPU

Figure 9. FPGA as SoC Asymmetric Multicore and as Coprocessor

Custom instructions

You can scale processor performance by adding custom instructions in-line with the processor code. This technique accelerates specific processor instructions, as illustrated in Figure 10.

You can accelerate time-critical software algorithms by adding custom instructions to the embedded processor instruction set. The example in Figure 10 shows how you can add custom instruction logic to the arithmetic logic unit (ALU) of the Nios® II processor. Using custom instructions reduces a complex sequence of standard instructions to a single in-line instruction implemented in hardware. You can use this feature for a variety of applications. For example, you can optimize software inner loops for DSP, packet header processing, and computation-intensive applications. The Intel Quartus® Prime software provides a configuration GUI that supports up to 256 custom instructions to the Nios II processor. The example in Figure 10 uses a 64 Kilobyte (KB) cyclic redundancy check (CRC) buffer. The custom instruction can accelerate CPU performance by up to 27 times faster than software-only operation in a Nios II processor.

The Nios II processor single-precision, floating-point custom instructions are another good example of accelerating processor operations. These instructions accelerate FPGA performance significantly for division, multiplication, subtraction, and addition functions. Other processor architectures operate on similar principles. The actual performance acceleration of custom instructions may vary by processor and custom instruction.

- For more information about Nios II Floating-Point Custom instructions, refer to Using Nios II Floating-Point Custom Instructions Tutorial and the Nios II Custom Instruction User Guide.

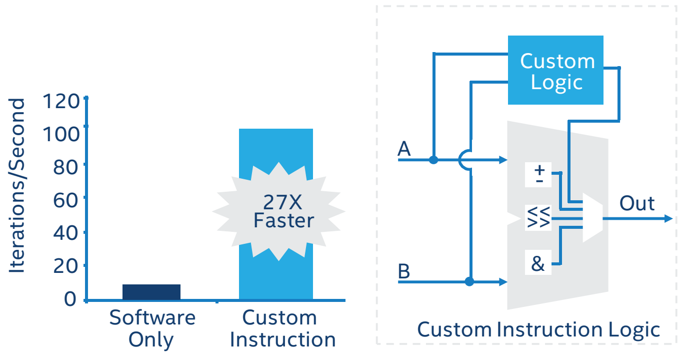
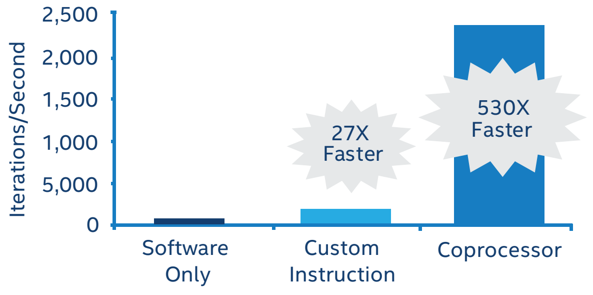


Figure 10. Custom Instruction Performance Gains

Hardware acceleration

In addition to custom instructions, you can use hardware accelerators, such as DSP blocks, video blocks, and other IP, to clear data bottlenecks. Figure 11 illustrates the use of concurrent, or parallel, data coprocessing to increase system performance by up to 530 times faster than the same Nios II processor system running only custom instructions. During concurrent data coprocessing, the processor CPU starts and stops the coprocessor, the coprocessor fetches data and stores results, and the CPU runs application code concurrently. This is ideal for block data operations, such as DSP functions often used in motor or motion control applications.



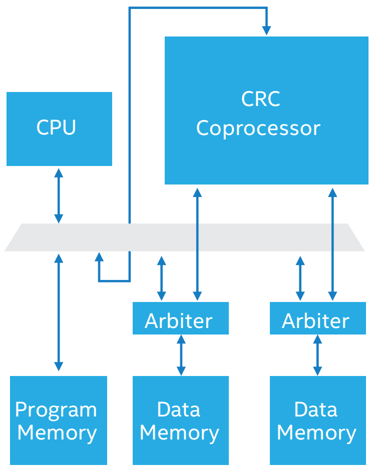


Figure 11. Hardware Accelerators Clear Processor Bottlenecks

Obsolescence protection

The long life cycles of FPGAs reduce the risk of product obsolescence. The life cycle of Intel FPGAs aligns well with the long life cycles of industrial equipment, thus enabling a consistent supply of devices, as shown in Figure 12.

Most MCU, DSP, or ASSP devices have significantly shorter life cycles than FPGAs because their vendors typically obsolete mature devices sooner than Intel. These types of devices are designed to fulfil specific applications for high-volume customers over a shorter period of time. In addition, although current ASIC devices in production may have a life cycle of 15 years or more, many are nearing their end-of-life, forcing designers to consider other long-term options such as FPGAs. Although new ASIC designs are in development, designers often cannot change these products fast enough to keep pace with evolving standards or new features required over time.

Conversely, FPGAs serve a wide range of applications and markets and are independent of any particular application for volume production. Therefore, it is cost effective for Intel to manufacture FPGAs over a longer period. You can better manage the stability of the supply chain, which may include many other semiconductor components. While using an FPGA platform for your design, you can update and change your designs, at any time. You can also reuse IP and port designs to newer FPGA families in a much less time than it takes to design a new MCU, DSP, ASSP, or ASIC.

Over time, Intel FPGAs help you improve the commercial viability of multiple product lines, while helping you contain the cost of product obsolescence.

Familiar tools

Intel provides embedded industrial designers with powerful and easy to use development tools—such as the Intel Quartus Prime design software, Intel FPGA IP library, Platform Designer (formerly Qsys), and the Eclipse-based Nios II Embedded Design Suite—that complement the FPGA hardware to streamline your design flow.

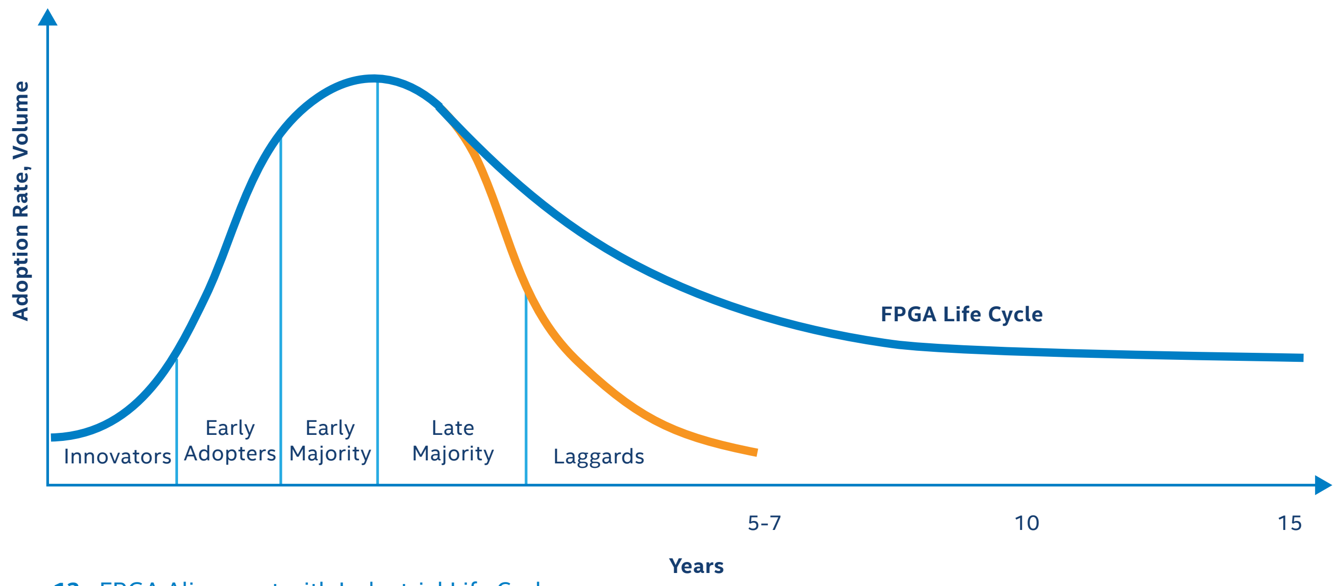


Figure 12. FPGA Alignment with Industrial Life Cycles

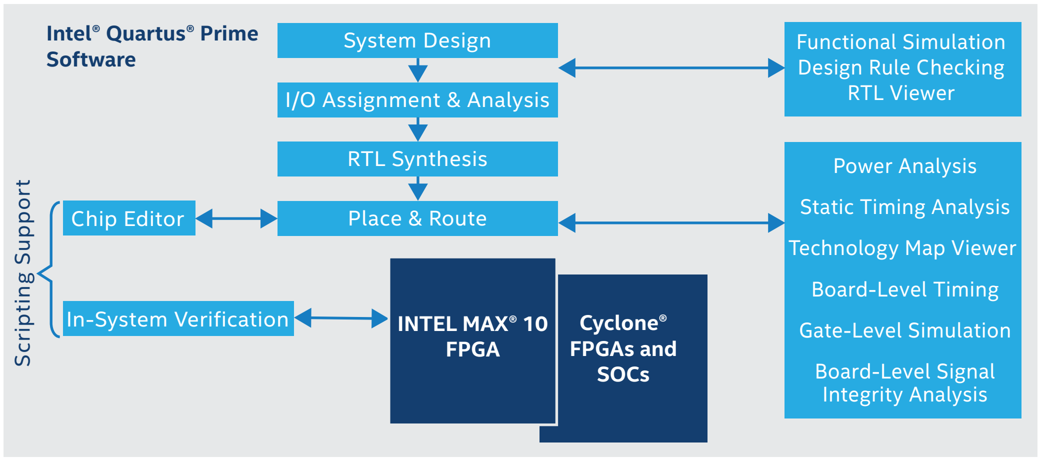


Figure 13. Intel Quartus Prime Design Flow

Intel Quartus Prime design software

The GUI-based Intel Quartus Prime software, available as a free web-edition or fully licensed version, addresses productivity and performance needs with a design flow methodology that includes system design and timing closure methodologies, in-system verification, and third-party EDA tool support, as shown in Figure 13.

The Intel Quartus Prime software accelerates your design flow with support for various design entry methods, scripting, incremental compilation, system-level integration, IP parameterization, I/O pin analysis, and synthesis options. At the verification and board levels, the Intel Quartus Prime software provides the Timing Analzyer, power analysis tool, a floorplanning chip planner, Signal Tap logic analyzer, register transfer level (RTL) viewer, and third-party verification support.

Getting started designing with the Intel Quartus Prime software is as easy as following three simple steps:

1. Run the New Project Wizard to quickly specify the project name, location, top-level entity, design files, target device, and optional third-party EDA tools for the project.
2. Complete the design, run timing analysis and synthesis to build netlist.
3. Compile the design to generate device programming files.

Figure 14 shows the Intel Quartus Prime main application window.

Intel FPGA IP library

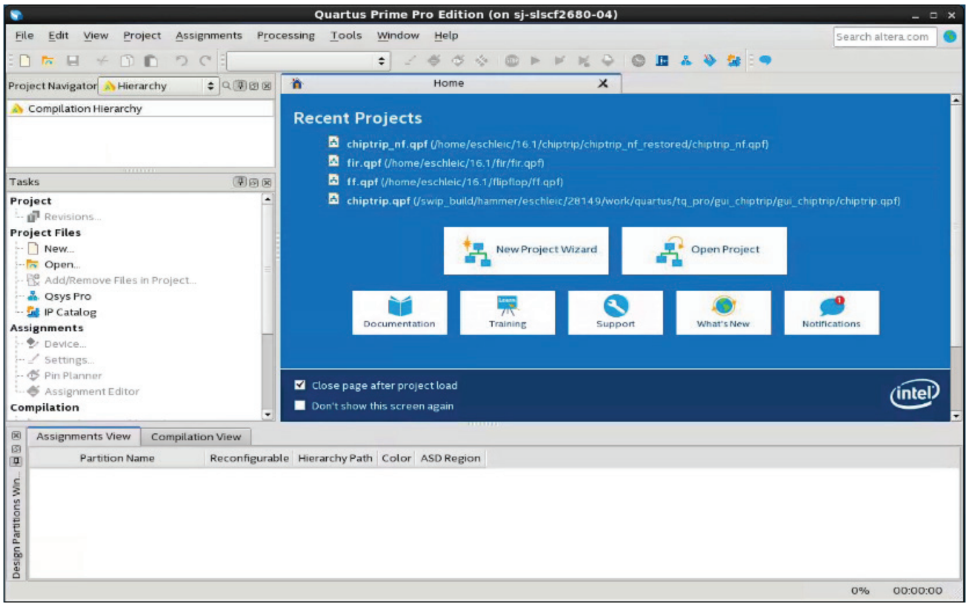
Intel and its third-party IP partners offer a broad portfolio of off-the-shelf, configurable IP cores optimized for Intel FPGAs. This IP includes components such as Intel's Nios II embedded processor, DSP modules, video IP suite, and many standard and popular interfaces like memory controllers, CAN, USB, and Ethernet. This licensed and unlicensed IP is delivered and installed with the Intel Quartus Prime design software. You can request partner IP directly from www.altera.com. The IP is modular, reusable, and easy to use and program into the FPGA via the Platform Designer. The Platform Designer also supports development and use of your own IP and interfaces.

In addition, Intel and its FPGA IP partners develop and deliver reference designs that show efficient solutions for common system design problems. You can download these reference designs directly from www.altera.com, use the automated request form, or you can contact partner IP providers directly.

For additional industrial IP available from Intel's IP partners, visit the Industrial Partners page on www.altera.com.

Platform Designer (formerly Qsys)

The Platform Designer is a system integration tool included as part of the Intel Quartus Prime software. The Plaform Designer captures system-level hardware designs at a high level of abstraction and


Quartus Prime Pro Edition (on sj-slscf2680-04)

Recent Projects

chiptrip_nf.qpf (/home/eschleic/16.1/chiptrip/chiptrip_nf_restored/chiptrip_nf.qpf)
fir.qpf (/home/eschleic/16.1/fir/fir.qpf)
ff.qpf (/home/eschleic/16.1/flipflop/ff.qpf)
chiptrip.qpf (/swip_build/hammer/eschleic/28149/work/quartus/tq_pro/gui_chiptrip/gui_chiptrip/chiptrip.qpf)

Partition Name	Reconfigurable	Hierarchy Path	Color	ASD Region

Figure 14. Intel Quartus Prime Software Main Application Window

simplifies the task of defining and integrating customized IP components. These components include verification IP cores, and other design modules. The Platform Designer facilitates design reuse by packaging and integrating your custom IP components with Intel and third-party IP components. The Platform Designer automatically creates interconnect logic from the high-level connectivity that you specify, thereby eliminating the error-prone and time-consuming task of writing HDL to specify system-level connections.

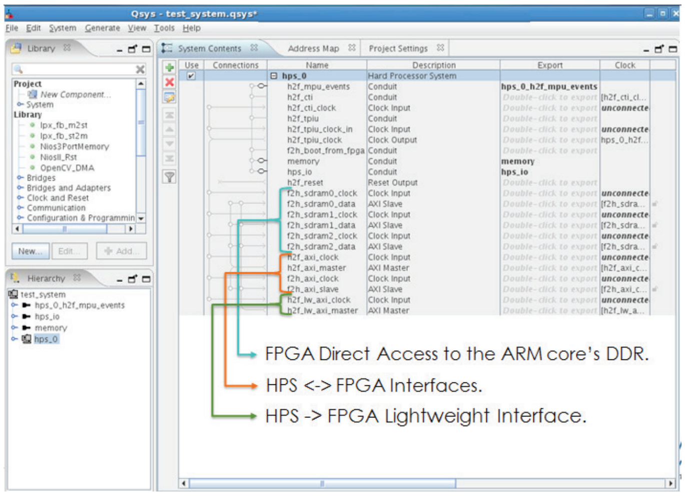
Using the Platform Designer, you can select the appropriate configuration options for each IP component. The Platform Designer, as shown in Figure 15, simplifies the process of customizing and integrating IP components into systems.

Standard Eclipse software tools

Typically, any system that requires a degree of control processing requires an embedded processor, especially if the processor must be contained within the SoC design. For those designers acquainted with software tools, Intel provides the Eclipse-based Nios II Embedded Design Suite, the Nios II embedded processor, and support for standard operating system (OS) and real-time operating system (RTOS) from a number of popular vendors.

With such familiar GUI-based development tools, the software team plays an integral part in the design flow. Your hardware and software team can capitalize on standard operating systems, board support packages (BSPs), and their application software expertise to port applications to run on one FPGA platform instead of across multiple MCU or DSP devices. You can apply previous software expertise with MCU or DSP programming to programming embedded processors—such as the Nios II embedded processor (using Eclipsed-based tools), the ARM* Cortex* M1, and the Freescale* ColdFire* V1 cores—all available for use with Intel FPGAs. The development tool flow and operating systems (such as Linux*) are very similar to those used in developing code for a discrete processor. Popular open-source operating systems such as Linux and eCOS are supported on FPGA-based processors, so you benefit from the active developer community creating new applications and features. These improvements and features can potentially save you many hours of effort in developing and supporting a product throughout its lifetime.

In addition, C code is portable across processor architectures. For example, The Nios II Software Build Tools (SBT) for Eclipse consist of a set of plug-ins based on the popular Eclipse framework and the Eclipse C/C++ development toolkit (CDT) plug-ins. The Nios II SBT for Eclipse provide a consistent development platform that works for all Nios II processor systems. These Eclipse tools improve software productivity for large software applications and team-based software design. You can accomplish all Nios II software development tasks within Eclipse, including creating, editing, building, running, debugging, and profiling programs.



Use	Connections	Name	Description	Export	Clock
✓		hps_0	Hard Processor System		
		h2f_mpu_events	Conduit	hps_0_h2f_mpu_events	
		h2f_cti	Conduit	Double-click to export	[h2f_cti_cl...
		h2f_cti_clock	Clock Input	Double-click to export	unconnecte
		h2f_tpiu	Conduit	Double-click to export	
		h2f_tpiu_clock_in	Clock Input	Double-click to export	unconnecte
		h2f_tpiu_clock	Clock Output	Double-click to export	hps_0_h2f...
		f2h_boot_from_fpga	Conduit	Double-click to export	
		memory	Conduit	memory	
		hps_io	Conduit	hps_io	
		h2f_reset	Reset Output	Double-click to export	
		f2h_sdram0_clock	Clock Input	Double-click to export	unconnecte
		f2h_sdram0_data	AXI Slave	Double-click to export	[f2h_sdra...
		f2h_sdram1_clock	Clock Input	Double-click to export	unconnecte
		f2h_sdram1_data	AXI Slave	Double-click to export	[f2h_sdra...
		f2h_sdram2_clock	Clock Input	Double-click to export	unconnecte
		f2h_sdram2_data	AXI Slave	Double-click to export	[f2h_sdra...
		h2f_axi_clock	Clock Input	Double-click to export	unconnecte
		h2f_axi_master	AXI Master	Double-click to export	[h2f_axi_c...
		f2h_axi_clock	Clock Input	Double-click to export	unconnecte
		f2h_axi_slave	AXI Slave	Double-click to export	[f2h_axi_c...
		h2f_lw_axi_clock	Clock Input	Double-click to export	unconnecte
		h2f_lw_axi_master	AXI Master	Double-click to export	[h2f_lw_a...

Figure 15. Platform Designer (formerly Qsys)

- For more information about the Nios II SBT, refer to the Nios II Gen2 Software Developer's Handbook, or training on www.altera.com.

Conclusion

Intel FPGAs provide the flexibility to adapt industrial designs to changing requirements and lower total solution costs. With a single FPGA, you can easily integrate part of your design into a single device, and then re-program your FPGA-based design, at any time—in both local and remote locations. This approach maximizes your design's ability to change with evolving standards, while minimizing the number of board designs required to support each protocol standard or each additional feature.

Intel FPGAs are ideal for parallel signal processing and therefore ideal for any system requiring a performance boost via hardware acceleration. Parallel hardware in the FPGA means no performance cost to add more controllers and features. You can accelerate performance with embedded processors and IP blocks on an FPGA used as either a coprocessor or SoC in your design.

Your software team can also capitalize on standard OS and BSP, and application software expertise and port applications to run on one FPGA platform instead of across multiple MCU or DSP devices. C code is portable across processor architectures.

One FPGA platform can support multiple product lines and provide a commercially meaningful, fast, and cost-efficient way to deploy solutions to market. The integration and flexibility benefits of FPGAs, such as the Intel® MAX® 10 or Cyclone family of FPGA devices, enable you to deliver products to market faster than with other technologies, thus maximizing market share, and extending the life cycle of your industrial designs.

Get started

If you're new to FPGAs and want to become familiar with FPGA soft embedded processors and development tools, the Intel MAX 10 Nios II Embedded Evaluation Kite (NEEK) (shown in Figure 16) is an excellent starting point. You can learn more about this kit at www.altera.com/products/boards_and_kits/dev-kits/partners/max-10-neek.html.

To prototype applications for industrial, automotive, consumer markets, the Terasic DE10-Lite board (with a 50K LE Intel MAX 10 FPGA and an Arduino UNO R3 expansion connector) provides a cost-effective solution. Learn more at www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=234&No=1021.

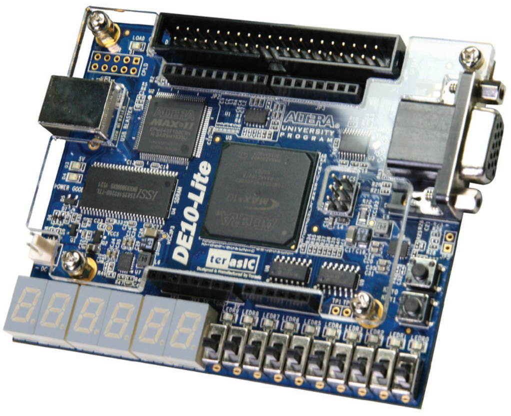


Figure 17. DE10-Lite Board

Where to get more information

For more information about Intel and MAX 10 FPGAs, visit **www.altera.com/products/fpga/max-series/max-10/overview.html**

For more information about Intel and Cyclone FPGAs, visit **https://www.altera.com/products/fpga/cyclone-series.html**

[1] www.altera.com/solutions/industry/industrial/overview.html
[2] www.altera.com/literature/wp/wp-01122-tco-industrial.pdf
[3] www.altera.com/literature/wp/wp-01133-ip-camera.pdf
[4] www.altera.com/education/webcasts/all/wc-2011-use-fpgas-industrial-applications.html

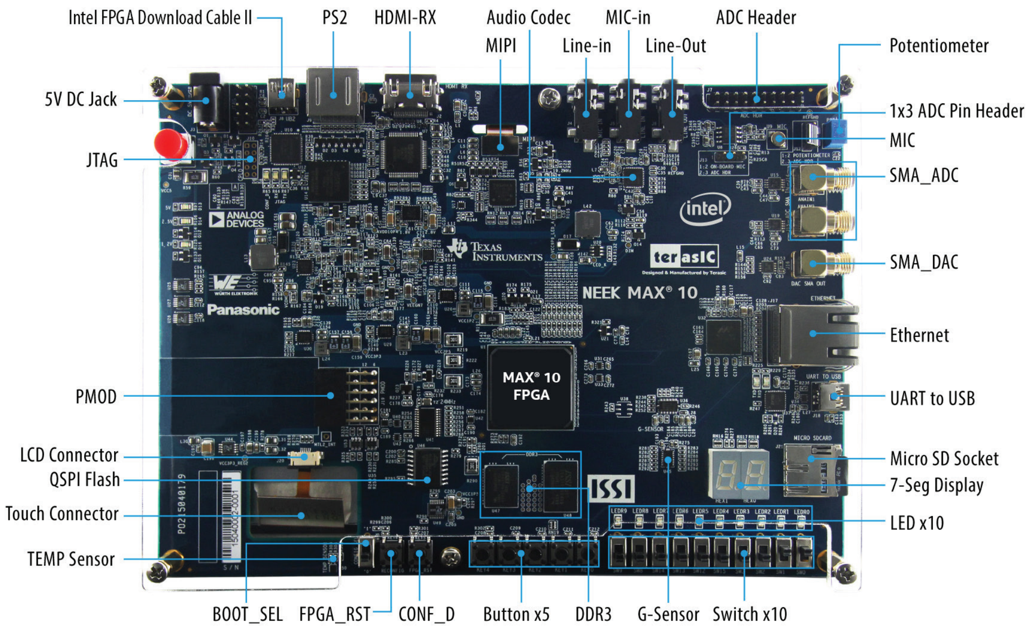


Figure 16. Intel MAX 10 NEEK