



FREEDOM OF CHOICE

Real evolutionary change occurs when innovative engineering minds have access to design tools that provide new ways to use technology and deliver possibilities in unrestricted ways...

The ongoing evolution of the electronics design industry is, in a fundamental and historical sense, directly coupled to advances in semiconductor device technology.



With each successive technology step, however, the real evolutionary change in product design occurs when the new device technology has been exploited to its full potential. This happens when innovative engineering minds have access to design tools that not only provide new ways to use the technology, but also deliver those possibilities in unrestricted and accessible ways.

The arrival of affordable microprocessors, originally conceived as a more effective way to produce pocket calculators, signaled the possibility of using software code to enhance and add functionality to products. However, it was the innovative application of those new devices using higher-level language tools that really created the revolution, by allowing a soft-centric approach to design.

By exploiting the full range of design possibilities, engineers were able to move large portions of hardware logic into the changeable software realm. The functional 'intelligence' that defines product value began its move into the soft domain, driven by engineers who harnessed the tools and technology to its full extent.

Today, the emergence of low-cost, high-capacity programmable devices such as FPGAs has delivered the next revolution in the way we design. These allow the intelligent portions of the design to encompass not only software in the traditional sense, but also 'soft' hardware implemented within an FPGA. By harnessing this technology beyond its basic potential as a mere container for glue logic, a product's unique functional aspects – and therefore its competitive advantage in the market – can be defined in the changeable soft realm.

Embedded system designers now have a new design canvas that can be used to create innovative new products. As a fully reconfigurable design platform to host a 'soft' design methodology, FPGAs provide the means to unprecedented levels of design freedom, thanks in part to a vast array of available IP cores that include logic blocks, peripherals and microprocessors. These can be changed at will as the design requirements evolve, critical design decisions can be locked down later in the design cycle, and both a product's software and hardware can be updated in the field.

When used to its full potential, an inclusively 'soft' approach to design delivers a new freedom to the product development process. Harnessing that capability to create products that offer true differentiation in the market means overcoming the implementation challenges the technology presents. The success of this is inexorably linked with the evolution of the design tools we use.

Introducing... limitations

In conventional design flows, the design opportunities offered by programmable devices invariably come with the legacy of increased complexity in the overall product development process. The new layer – or 'domain' – introduced into the product development toolchain is traditionally based on an embedded hardware development toolset that is both separate and disconnected from the existing design process. This not only creates significant complexity when integrating the separate FPGA-based sections into the overall product design, but also introduces a new set of design skills that you need to learn to make use of the embedded development tools.

The main reason for these challenges lies in the genealogy of conventional IDE (Integrated Development Environment) tools, which have their roots in the ASIC design world where HDLs

(Hardware Description Languages) are used to describe a chip's hardware design prior to manufacture. Using a conventional IDE toolchain means working with an isolated design environment and becoming an HDL specialist, while applying a working knowledge of the target FPGA's underlying architecture.

Most commonly, the IDE is a proprietary toolchain from the FPGA device vendors. These are developed to support each vendor's products and encourage sales, but are understandably devoid of support for competitors' products. Using a vendor IDE restricts the inherent flexibility and freedom of embedded hardware design by limiting the range of FPGA devices, vendor IP cores and design methodologies you can use. What's more, if you change the programmable device that's hosting your design you're back to square one.

In practice, the partisan nature of the development tools significantly compromises the available design choices that can be explored in the pursuit of innovation.

Vendor-neutral design


To open up the full range of design possibilities in today's embedded design process, what's needed is a programmable device development environment that allows you to easily target a wide variety of devices from different vendors. By creating a design environment from the ground up that is independent of vendor or device, the door opens to the full range of target possibilities, freeing embedded developers to choose the best possible device for the current design.

Such a system is also inherently neutral in the embedded design itself which allows the target device to be changed without compromising the validity of the source design files. As a result the target device can be changed at will and the final choice made much later in the design process, when its required specifications are more clearly understood. The embedded design process can proceed while all the device options are still available, removing the need to develop the hardware before serious embedded design can start.

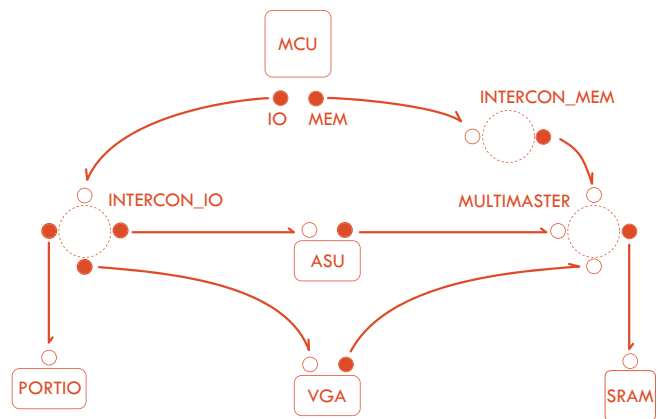
A design system that is independent of vendor also offers a greater range of IP cores by allowing you to simply change the target device to achieve compatibility with a desirable vendor-sourced core, such as a target-specific, high-performance microprocessor. If the system's device-neutral framework is harnessed to develop libraries of IP cores, the resulting blocks of soft hardware can exhibit the same neutral persona as the system. This introduces the compelling prospect of using a neutral 32-bit microprocessor core for the initial development work, then changing that IP block to a vendor-supplied core at a later time – with minimal impact on the work already done.

The success of this approach lies, in part, in the availability of system updates that add new device hardware support. The latest programmable devices can freely be used in your designs along with fully compatible IP cores from the system libraries. The latest 'soft' and discrete microprocessors, along with those embedded into the fabric of the FPGA, can also be supported because the system's embedded software toolchains are also updated.

Designing in a truly vendor-neutral product development environment delivers the freedom to easily explore all design possibilities. As the design progresses, the full choice of programmable device, microprocessors and IP cores can be exploited to create the most competitive product, without compromise.



Designing
in the soft
domain
provides
a blank
canvas for
innovation
and creativity.



High-level design capture systems that automatically deal with the underlying hardware complexity free designers to focus on innovation.

Clarity through abstraction

Along with restrictive device and IP possibilities of conventional embedded design flows, the arcane nature of HDL-based design capture – considered analogous to assembly language in complexity – demands specialist skills and distracts engineers from focusing on innovative design.

What's needed here is a system that raises the abstraction of the design process so that design engineers can work at a higher level, while using and expanding their existing design skills. The potential exists for high-level embedded design methodologies that allow engineers from all disciplines (hardware and software) to simply connect together pre-verified IP blocks in a way that is not only accessible, but hides the underlying complexity of the device architecture.

By offering design interfaces that are graphically based, or based on schematics, the process of embedded design is streamlined without the need for specialist HDL skills (although HDL entry can also be incorporated). When this system exists as a vendor-neutral embedded design environment, the process of changing processors, peripherals and memory potentially becomes simple block-level manipulation via a graphical interface.

The primary advantage of the system is that designers are free to focus on the essence of a product's development – its functional 'soft' intelligence. High-level interface choices easily capture the design intent, without complexity and distraction.

When incorporated in a system that supports familiar high-level design paradigms at a native level, the underlying hardware complexity is effectively hidden. Embedded hardware abstraction can include advanced IP interface systems such as the Wishbone bus and hardware interface IP cores that introduce transparent interconnect layers around processors, peripherals and memory. With the low-level interface architecture supporting design interfaces that reduce complexity, embedded system design can virtually become an exercise in 'joining the dots'.

Freedom through unity

Moving beyond a conventional IDE-based design flow can introduce a valuable expansion in the choices available to designers while providing innovative new ways to create embedded designs. This is provided by an embedded hardware design system that is independent of vendor or device and offers

a high level of design abstraction, but how that system connects into the rest of the design environment is also important.

For that to be fully effective, the design abstraction and open device choice needs to be reflected through the entire product development chain, so that the traditionally complex process of bringing all the design elements together (hardware, software and programmable hardware) becomes simplified and transparent.

A unified design system delivers this by bringing all of the design domains together into a single and cohesive application. Design models become single entities across all domains, allowing (for example) library IP blocks to encompass multiple device support, graphical models and hardware interface layers. Changes in the embedded design can be directly transferred to relevant domains using high-level automated processes such as FPGA pin optimization. This is possible because all parts of the design process share the same, single model of the design data.

Taking the unified vendor-independent design environment to its logical next step means encompassing those attributes in an associated hardware development platform. A hardware platform that is independent of vendor or device (possible through a system of plug-in FPGA boards) and communicates directly with the unified application can reflect the needed flexibility and open design choice into the physical world. The design iterations needed to refine and debug your design project can appear in real hardware, in real time, which also reduces the needed for design simulation.

When such a reconfigurable hardware platform unifies with the overall design system, the device and IP design choices you make can transparently flow through the whole design chain, from design capture through to physical hardware. The entire product development system is then unified and cohesive.

If the design environment also offers high levels of design abstraction, system developers are no longer restricted by the choice of devices and the method used to capture the design intent. The end result is an open approach to embedded system development that frees designers to focus on the essence of design – creating the functional device intelligence that provides true product differentiation. ●

Published: cieonline.com.co.uk, UK June '08 and EDN, US, June '08