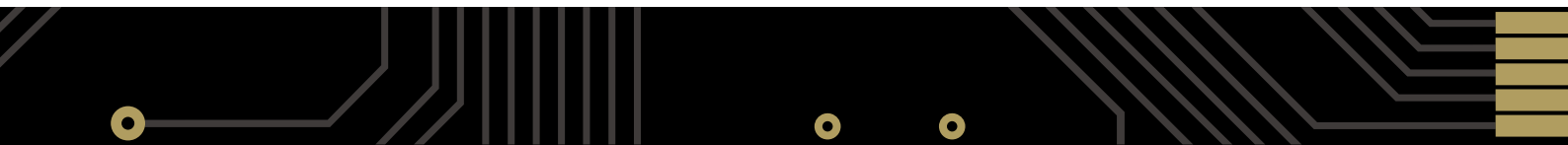




## Design abstraction – a practical view

---



The concept of applying a higher level of design abstraction to creative and engineering processes is so closely familiar that we probably take it for granted. From NC machines to SQL database systems, a high-level approach to capturing the design or operational intent of a system is universally accepted as the way it's done.

The benefits of course are substantial and easily justify the task of developing the specialist skills needed to use the highlevel systems. By inserting translation layers between the fundamental nuts and bolts of the system and the user, systems that raise the abstraction of creative processes dramatically reduce the level of complexity faced by the user and help to speed the overall process. Crucially, such systems also offer the opportunity for more users to 'play' in that arena, since a fundamental knowledge of the underlying technology is no longer essential.

In the electronic engineering world the obvious example is software development, where the move from using assembly code to high-level languages like C++ has revolutionized the way embedded software is created for processor-based systems. Any initial resistance based on the increased size of the resulting code was soon offset by larger, lower-cost system memory and faster processors, leaving software engineers to benefit from a reduction in complexity and faster development times. Thanks to C, the need to battle with arcane register stacks or reams of packed, sequential code is simply a thing of the past.

The way electronic products are developed and the technology we use has rapidly evolved however, and as history has shown, technology changes have a habit of shaking up the way things are normally done.

## Embedded design complexity

The advent of low cost, large scale FPGAs has opened a whole new range of possibilities for embedded system developers, where complete embedded systems – both the hardware and software – can be hosted within the programmable realm. While software is developed in a familiar way, the process of creating embedded hardware typically involves working with a hardware description language (HDL) such as VDL and Verilog, and applying a level of knowledge about the underlying hardware architecture.

Developing embedded hardware with these systems, in a design flow that has largely been adapted from the chip design industry, means acquiring specialist programming skills and dealing with a new level of design complexity. For embedded system developers with a limited knowledge of the hardware technology they are working with, an HDL solution imposes significant barriers to embedded hardware design.

As already evident in other design capture environments, raising the abstraction of what is a complex process should yield the benefits needed to free more engineers to create embedded hardware platforms. Unlike the systems that have benefited from doing this however, there is an important difference in the development process for embedded hardware; it does not exist in isolation, but is fundamentally linked to the other parts of the design process.

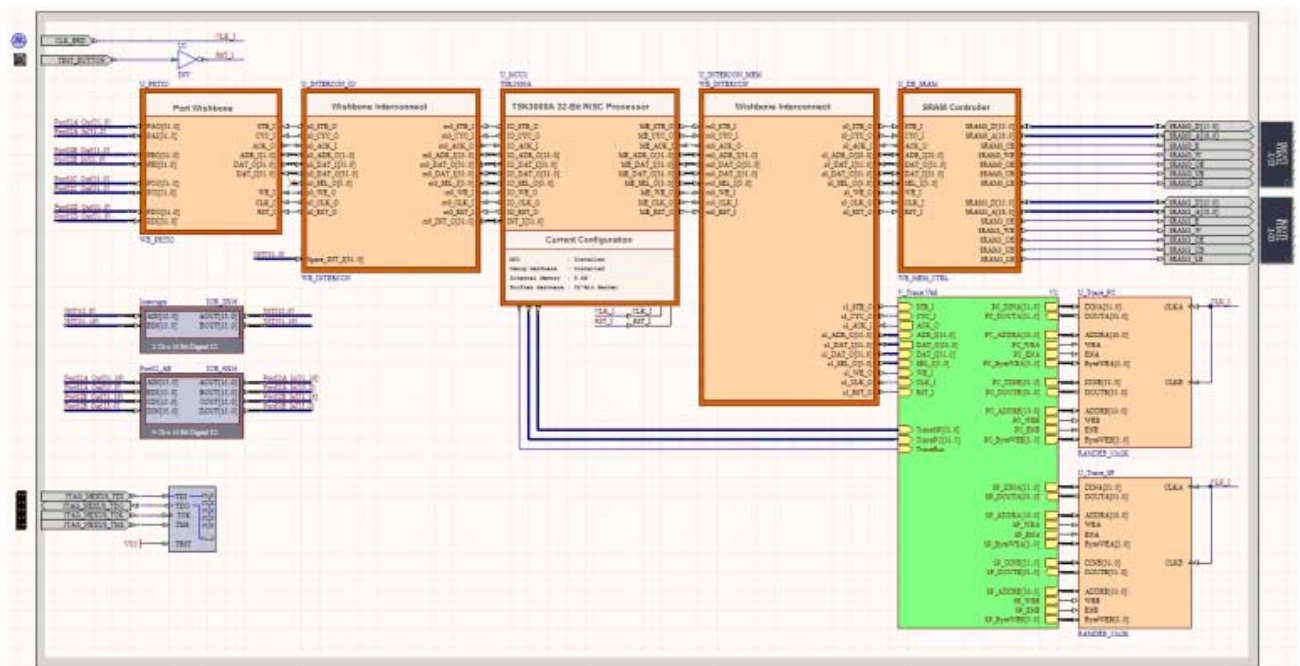
What this means in practice is that the embedded hardware configuration has a direct and inescapable link to both the design's software and its underlying physical hardware. For example, changing the type of embedded processor or peripheral function block is very likely to influence an FPGA's pin configuration, and the embedded software it hosts. As you might expect, the converse situation is equally true, where changing a hardware device such as the FPGA directly effects the embedded part of the design and how it responds to the embedded software.

High-level design approaches are needed to make embedded hardware design an accessible and connected part of the overall product development chain. There are a number of approaches to this, and in a similar way to the adoption of highlevel software languages, any increase in the size of the resulting embedded code is easily offset by more capable, larger devices. But while the introduction of abstraction or translation layers is likely to make the design capture process more user-friendly and efficient, the system integration issues are another matter.

Take for example a high-level embedded design system based on a variant of a conventional software language like C. Such a system inserts a C translation layer and offers software engineers with a working knowledge of hardware design the opportunity to create embedded hardware using a familiar approach. It is however a largely disconnected methodology for embedded hardware design that does not address the interdependent nature of the overall design process.

Raising the level of design abstraction in this way can satisfy a subset of software engineers, but tends to complicate the overall process by introducing a specialized development environment that is disengaged from the design development workflow. And just like conventional HDL entry, a C-based approach still requires software engineers to have a working knowledge of hardware design, and hardware engineers need to learn a specialized software language.

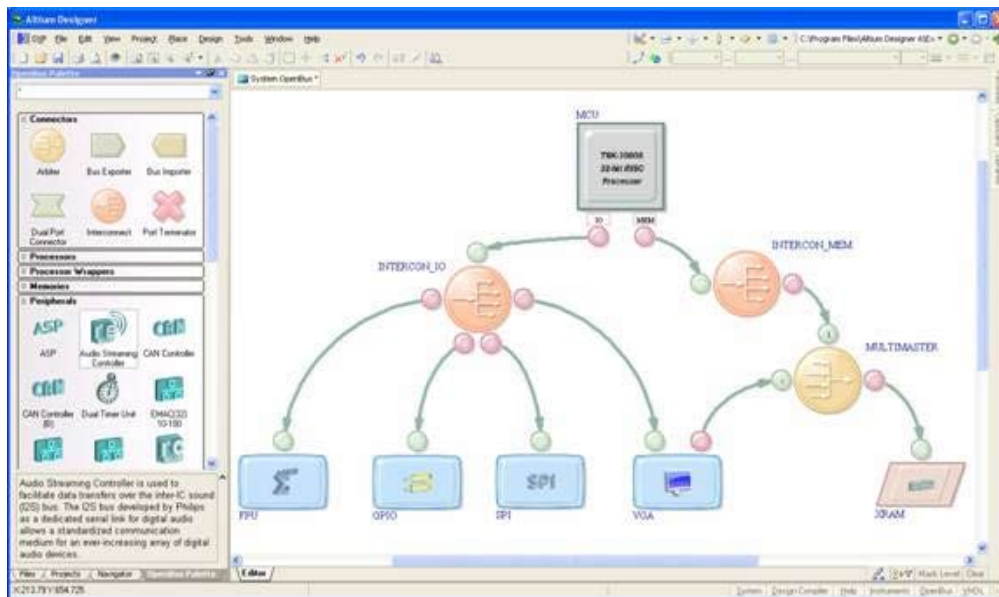
What's really needed is an embedded development system that raises the abstraction of the embedded (or 'soft') elements of the design process in ways that suit all engineers – system, software and hardware – and makes that process an intimately connected part of the overall product development workflow.



Embedded hardware design within a schematic interface. Wishbone bus IP components simplify processor to peripheral connections in an environment that's familiar to hardware engineers.

In such an approach the underlying gate-level source and FPGA architecture considerations are taken care of by the design abstraction layers within the system. With this system in place the potential then exists to introduce further levels of design abstraction that move one step further from a schematic approach, where the key elements of an embedded processor-based design are represented as simple graphical elements or icons.

These simplified elements are connected together in a simple flow chart approach, with individual bus interface connections – the 'wires' in effect – taken care of by configurable abstraction layers within the system. The design capture process in this case centers on a more software-centric approach of creating functional interactions between elements, rather than wiring hardware blocks together.



Embedded hardware design within a graphical icon-based environment. The underlying hardware complexity of the interconnections is handled by the system, opening up embedded hardware design to hardware-savvy software engineers.

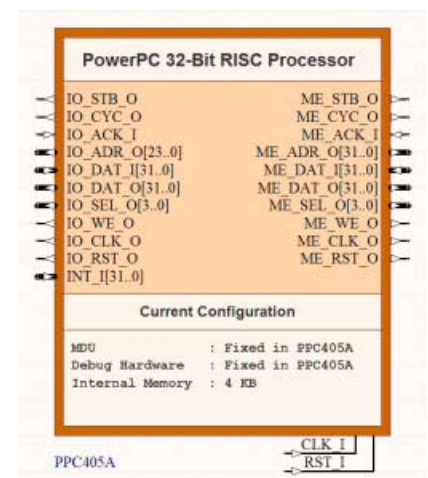
The common factor with these two approaches is that the underlying FPGA architecture has been hidden, or more accurately, taken care of by automated systems within the design environment. As such, the systems go beyond just the implementation of a translation layer to make design capture more accessible.

## The underlying nuts and bolts

One of the key elements that allow these high-level design interfaces to become truly practical is the nature of function blocks or components that are used to assemble a design. These are the raw materials of the high-level design system, and exist as collections of embedded IP blocks that can be placed from a system library. The IP blocks can be created to be independent of the FPGA device or vendor, further freeing the system from low-level architecture considerations. Along with core blocks of functional logic, this soft IP can include microprocessors, peripherals and memory, so everything needed is ready to go.

The possibilities of the system expand dramatically however when abstraction layers are implemented in the embedded hardware IP components themselves. For example, library IP parts featuring the Wishbone OpenBUS standard can be included to 'normalize' the interfaces between processors, memory and peripherals. These simplify bus interconnection systems to a point where the low level hardware architecture can be handled by the system rather than the designer.

Using this approach, library-based hardware interface cores can also be introduced that 'wrap' around pre-defined processors and peripherals, providing an isolating layer between the existing interface arrangement and a hardware Wishbone bus. These configurable hardware interface cores allow third-party or vendor IP blocks to be dropped into a design easily and changed at will. Changing to a different processor IP block causes minimal impact to the surrounding hardware, and when backed by compiler tool chains for all the supported processors, the embedded software can remain intact. Again, the system handles the hardware complexity, leaving the designer to implement the design's functionality via the high-level design interfaces.



A wishbone 'wrapper' component for the PowerPC 405 processor.

Simplifying the embedded hardware structure and design processes in this way means that complex processor-based embedded systems can be created and changed easily. The division between hardware and software is more flexible, allowing the partition to be moved throughout the design cycle rather than be locked at the beginning of the embedded development process. It also opens the possibility of implementing additional high-level systems that translate software algorithms into hardware co-processors, where the performance of the two options can be compared and selected accordingly.

When coupled with an underlying system that uses software abstraction layers to segregate and configure the supporting hardware, high-level design capture systems allow designers to work directly with the embedded elements that define the crucial product functionality. Furthermore, the hardware platform is no longer the first consideration and can be dealt with later, when the product's form and function have been developed to a mature state.

## Freedom though unity

The critical factor in the successful implementation of a high-level design approach is the design tool infrastructure that supports it, which needs to deliver the interconnection systems and design data management capability that makes the process viable in practice.

As the lines between hardware and software design blur, raising the abstraction level of design processes must be pervasive – or unified – across all stages of the design process. For example, placing a USB block in a design has ramifications at a schematic, board, programmable device and software level, so the abstracted IP block must inherently represent all these elements and transparently interconnect with universal bus systems.

A unified design system – one that brings together hardware, software and programmable hardware within a single application – makes such an approach possible by implementing a single design data model across all parts of the design processes. A block of functional IP that is stored independently or as a library part can contain all of the elements need to create a production-ready design – from PCB patterns right through to software drivers. When capturing the functional intent of a design, you just place and connect up a graphic symbol that represents the part. In the background though, all the elements of that part are imposed and connected across the entire unified design system.

At this level of design abstraction engineers from all disciplines are free to create designs quickly in a modular connect-the-boxes way, speeding the product development process and freeing designers to add value to the final product through innovative IP.

In addition, by raising the abstraction at a unified level, the traditionally complex process of bringing all of the elements of a design together – hardware, software and programmable hardware – is greatly simplified. The design is now abstracted as a whole rather than at different, specialized levels within each design discipline, which liberates cross-discipline processes such as FPGA pin optimization and software-to-hardware transition.



*A unified product development system makes high levels of design abstraction effective through the entire design process. This revolutionizes the way products can be created and opens embedded design to both hardware and software engineers.*

To simplify and streamline today's embedded design process, and make it accessible to more design engineers, the single step of introducing an abstraction layer that translates the design capture interface is not enough. Unlike traditional software development, embedded hardware design does not exist in isolation so an insular design abstraction system does not address the interaction needed across the traditional design boundaries.

The key to making such a system truly practical is creating an infrastructure where the raised level of design abstraction is manifested through the whole design process. Presenting a consistent interface across those design boundaries allows engineers from all disciplines to assemble predefined functional design blocks easily, by simply using or building on their existing skills.

Ultimately, these systems both unify and simplify electronic product design, while freeing designers from all disciplines to create more innovative and intelligent products within a highly-accessible design environment.