



Next Generation System Design – Platforms versus Tool-Chains

Consider the Benefits of a Platform-based Development Approach

**By Bob Potock
February, 2010**

Introduction: the Benefits of a Platform-based Development Approach

The original electronic design process built by linking tools together has remained largely unchanged for decades. Companies doing electronics design continue to design products by cobbling together a collection of tools to create what is commonly referred to as the “tool-chain”. Many of these tools are not even from the same vendor because building or acquiring so-called best-in-class tools by acquisition was the strategy of the day.

The argument behind this approach remains that this collection of “best-in-class” tools provides a competitive advantage. The suppliers of these tool-chains continually invest in feature upon feature even as the tools have matured. They find it more and more difficult to differentiate the tools’ capabilities.

And so the question becomes this: can tools alone add the productivity improvements necessary for product development companies to compete worldwide, especially today, following the most severe economic downturn for decades?

But some design companies are evolving, to build products faster, cheaper and with some competitive differentiation. And they’re doing this by not focusing on the tool. Instead, they are implementing an holistic way to manage the design data and processes, and it’s this holistic approach that supplies the dramatic improvements in cost and productivity.

A tool-chain centric approach does not address the needs of today’s companies which must build (and want to build) more complicated products that require multi-domain expertise spread among dispersed design teams. This new generation of product development is elevating the importance of data management and process over so-called best-in-class tools.

Companies must focus on developing product differentiation to be successful and not waste time managing the tool-chain. This paper examines the unified design platform as a new architecture that supports multiple domains (PCB, FPGA and embedded software) and offers significant productivity improvements over tool-chains.

The Rise of Tool-chains

The original printed circuit board (PCB) process did not require much more than a schematic editor, a layout tool, a manufacturing output generator and a library (Figure 1). The schematics were flat and the components simple through-hole devices. Given such a simplistic process, the focus and investment were naturally on the tools and the tool feature sets while the development process was secondary.

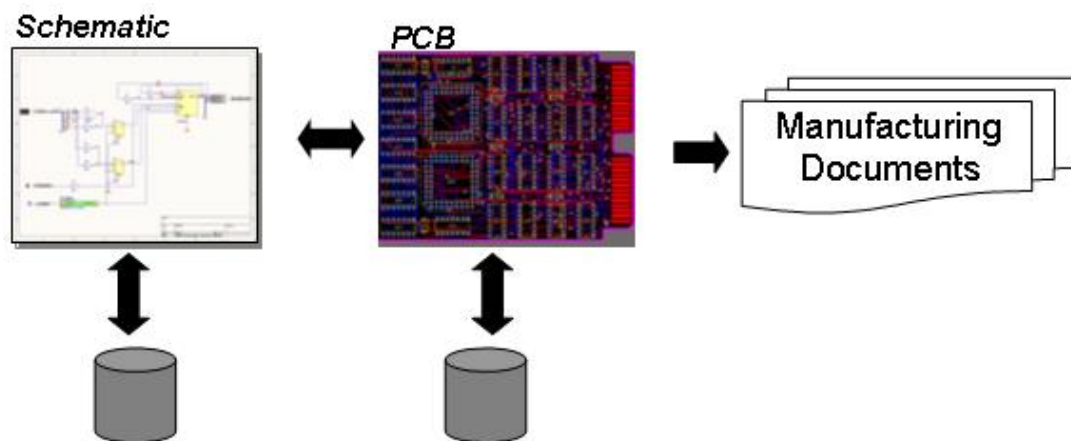
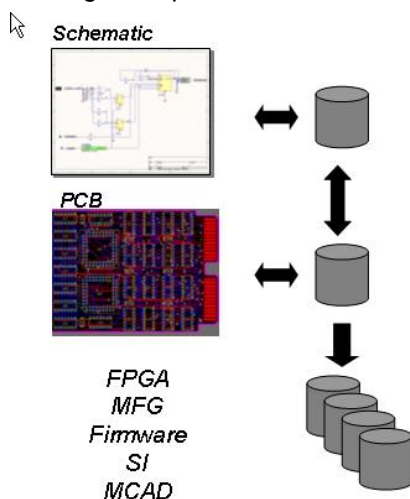


Figure 1: Basic Tool-Chain for PCB

As the basic process expanded to include router and simulation, EDA companies shopped around for the best-in-class tools and acquired them. This acquisition approach was a viable strategy when tools were maturing and the tools could significantly effect productivity. The risk was that the development process became a tool-chain comprised of applications from various vendors with different user interfaces and data models. To get the tools to work together they were “bolted” together with data import and export utilities. This created a different problem: each time the tool data model was expanded, there was a risk that the import/export functions would break.

Expanding the basic tool-chain to include signal integrity, simulation, manufacturing and mechanical integration, only complicated the situation (Figure 2). EDA vendors now built importers and exporters for each of the connectivity points in the chain. Tool vendors invested in building and maintaining the integration points in addition to enhancing the tools. And these tools (signal integrity and schematic) were never built with the intention of working in any particular tool-chain or with any specific tool. They may not have even been built by the same EDA company.



Today, as the number of tools grows and the versions of each tool change, so does the risk to users that something will break. For this reason, many companies fear upgrading a particular tool in the chain without extensive in-house testing before deployment. This is a costly overhead for any company, one that should not be necessary.

Today, as the number of tools grows and the versions of each tool change, so does the risk to users that something will break. For this reason, many companies fear upgrading a particular tool in the chain without extensive in-house testing before deployment. This is a costly overhead for any company, one that should not be necessary.

Figure 2: Today's More Complex Tool-Chain

And don't forget about the learning curve for users when dealing with multiple tools with different user interface protocols and capabilities. Tools may behave differently from simple zoom and pan commands to the supported scripting languages. And every difference has some type of cost associated with it.

What Happens when Multiple Tool-Chains Are Required?

So far, I've talked only on the PCB development process. Now consider a product with an FPGA and a soft processor inside the FPGA. The FPGA development process introduces a second tool-chain and the embedded software development a third. So now a company developing a product with a fairly standard set of technologies – PCB, FPGA and embedded software – has to contend with not one tool-chain but three. And all three need not only to move data between the tools within the chain, but externally to other chains as well. And remember that the three tool chains are most likely from three different EDA vendors or at a minimum, three different divisions within a company.

Let's examine a rather straightforward example of cross chain data movement. The product under development has a large high pin count FPGA, such as an 1153 pin package with 800 user I/Os. Let's assume that the FPGA is in the first stages of the design process, so all the pins have been initially assigned in the FPGA tool-chain. The device is routed and now those pin assignments must move to the schematic within the PCB tool-chain. The manual method is to wire up the FPGA symbol in the PCB schematic by hand. With hundreds of pins involved, this is time consuming, error prone and when over 1000 pin devices are involved, the symbol typically needs to be fractured.

The completed schematic is exported to the PCB tool where the design data are now in layout and the board is being routed. While the PCB is being routed, the designer discovers that the high pin count FPGA will not route with this pin assignment. As in decades past, the PCB designer, with the permission of the system/FPGA designer, swaps some pins. But now the data need to traverse back to the schematic and then to the FPGA tool-chain. It's easy to see how maintaining synchronization is a big problem where the pin assignments in the PCB tool-chain do not match the pin assignments in the FPGA tool-chain. This is a common example of how data must move through a complex error prone path for a product development process involving standard technology. Ironically, the promoted solution to this problem is to purchase yet another tool that helps with FPGA pin management and synchronization.

Tool-chains Struggle with Change Management and External Processes

With the maturation of the tools and the growing complexity of today's products, the focus is shifting from tools to data. Productivity is typically not increased significantly by improvement within a single tool when you consider the entire design process that includes PCB, FPGA and software development. But a data management misstep within the design process can cost significant time and money.

For instance, consider a situation where a change was made to some FPGA pin assignments to simplify the routing on the PCB. The design was completed and all the manufacturing outputs generated for release. Enter an engineer who decides that there may be some signal integrity issues due to the routing density of the FPGA and decides to change pin assignments. The manual data manipulation across tools and chains, and the lack of an effective change management system, allows the change to fall through the cracks and never make it into the product release. This results in a product with a signal integrity problem.

Today's complex and global design process require a robust change management capability. Change management is the ability to track changes and also be able to detect and visualize the differences between the different versions of the design. The problem described above could have been averted had the process included a change management system.

But implementing a simple version control system in a tool-chain is not a simple matter let alone a change management system. Each tool stores the design in a unique bundle of binary and files generated by the tool. This makes implementing a version control system across a heterogeneous tool chain a significant effort and requires modifications each time a tool is added or upgraded. Now consider a report generation capability that can identify the differences in different versions and then the ability to visualize those differences. Implementing a change management system like the one described is a significant challenge for a tool-chain.

Another consideration is the extensibility of the development process to include mechanical or business process integration. For instance, can the component data model and library management process be extended so commerce data can be loaded from an external component supplier? This would provide valuable component selection information like cost, availability and life cycle for the part under consideration. It is also becoming more important for mechanical integration given the miniaturization of so many products. Can the tool-chain accommodate a 3D library and PCB layout capability that can be easily integrated with the mechanical design process? Process and data model extensions as those described are areas with opportunity for productivity improvements. Tool-chains can be extended but because of the multiple data models the changes can have a tool to tool ripple effect that can add complexity and cost to the implementation process.

The Alternative to Tool-chains: Platform Based Product Development

An alternative to the common tool-chain approach is a platform approach built upon data and not the tool (Figure 3). That is not to say tools have secondary importance, but there is a consistent data model that is common among the tools. The tool layer is built with a single executable removing the need to import and export design data from tool to tool and chain to chain. The platform architecture also includes IP (intellectual property) and software source code as an intrinsic component. Having the IP layer work closely with the tool layer significantly increases productivity. Standard peripherals, such as USB, PS/2, TFT LCD panels and MIDI support, are built into the platform as part of the IP layers.

The blocks labelled Extended in Figure 3 demonstrate how the platform can extend to new technology or IP. IP specific to a vertical market can be easily added to the platform because the infrastructure to support it already exists. The platform can accommodate the needs of each vertical market in a simple, coherent manner.

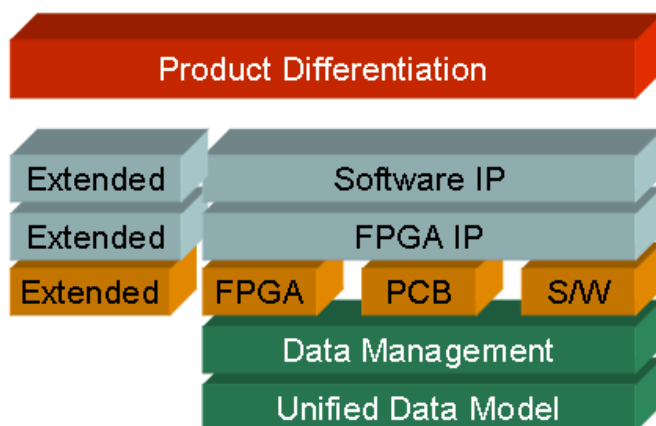


Figure 3: The structure of a System Design Platform consisting of FPGA, PCB and embedded software.

The foundation of platform based product development is the unified data model. The unified data model contains the entire design data set with each tool using the applicable section. The data management layer on the top of the unified data model is intrinsic to the platform architecture and is not an “add-on” as in the case of tool-chain. Providing these intrinsic data

management capabilities, that include change management, versioning, hierarchical projects, release

management and IP support, is where the platform approach begins to offer significant end-user value over the tool-chain.

Change Management and Release Management

The notion of data versioning has been around for many years in the software domain, but has never really become a standard in the hardware design world, primarily because of the tool-chain architecture. Schematic and PCB design data models have evolved over time and are comprised of binary and machine generated files. Versioning the design requires an intimate knowledge of what makes up a schematic or PCB design data set. Each tool would have a different requirement because each tool has evolved differently and independently of the other tools in the chain. In the case of the platform architecture, the data model and the data management function were developed in concert providing a solid foundation for data management capabilities, such as change management and ultimately configuration management.

An engineering change order (ECO) is an example of a common change management activity that a tool-chain has difficulty executing. Let's consider a situation where an engineer has made a design change to improve the performance of the design. This may involve adding a new component and some wires. Within the tool-chain design process, an import/export function called forward annotation is used to move the changes from the schematic to the layout tool. The tool-to-tool transaction completes, the PCB layout tool data are updated and, in most cases, a change history is not part of the forward/backward annotation process so the specifics of the change are lost. If a company wishes to maintain a design change history (i.e. change management), it would have to be done external to the normal operation of the tool-chain; typically it would be done manually.

Contrast this with a more desirable, change management process like a robust ECO process where a design change moves through a process that records all aspects of the request and creates a history of all document changes that occurred during the development process. The platform approach implements a data management layer that enables an ECO process. Design changes within a platform-based design process are kept as a history noting design document changes and dates.

Let's take change management a step farther – the ability to visualize changes between document versions.

Because of the data management layer, the platform architecture can offer change management benefits that go beyond typical design versioning. A platform implementation can provide the ability to visualize the differences between two versions of a schematic or PCB design. Consider a situation where the engineer wanted to compare version 1.6 and 1.9 of a schematic. The platform reports not only the differences found between the two versions but also allows the engineer to visualize the differences. The ability to view and verify differences between two versions of the design and associate those changes to an ECO is needed in today's global and complex design processes. This has tremendous value when considering the number of changes that occur during the development phase and the number of people involved. Figure 4 demonstrates how a removed INV is visualized when comparing two versions of the schematic.

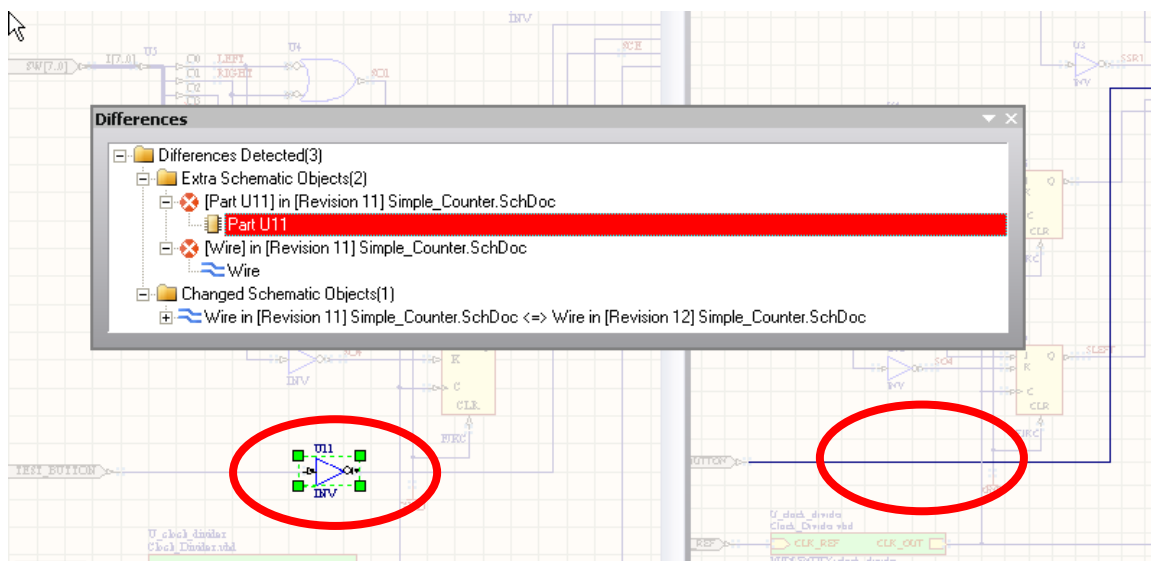


Figure 4: Note that U11 has been identified as a difference between the two schematic versions as well a wire change.

The unified data model offers significant advantages when one considers the all important release management function. Release management is the generation and packaging of a document set for manufacturing. The document set or package can contain schematic and layout PDF, BOM, drill files, Gerber files, ECO history, sign-off documents, etc. The platform architecture enables push-button generation because all the data are stored in one design database under one project. Contrast this to the tool-chain where the data are dispersed among multiple projects and tools. In many cases, the output files must be generated from multiple design databases and then collected for publication. The tool-chain generally relies on manual generation of the release package. Manual construction of the release package opens the door for the common mistake of selecting the wrong version of a design data set (e.g. Gerber version does not match drill file version) and the PCB is bad. Release management in the platform is part of the data management layer.

Hierarchical Projects

An even more powerful benefit of platform-based development is the hierarchical project. Today's products typically involve multiple technologies requiring multiple tool-chains. So a product development process based on a tool-chain architecture would typically involve multiple projects – one for each FPGA, PCB and embedded software design. In contrast, the platform architecture supports the notion of hierarchical projects with the PCB project at the top with the FPGA and embedded software design data as sub-projects (Figure 5). The product under development is now managed holistically and design teams can easily checkout the schematics, HDL or C code as documents, enabling worldwide design concurrency.

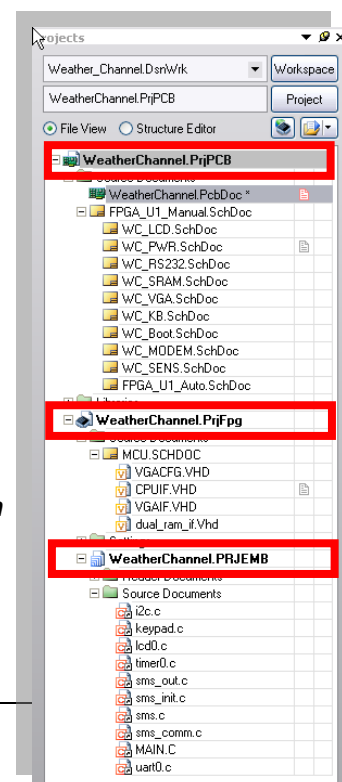


Figure 5: A hierarchical project consisting of a PCB, an FPGA and an embedded software design.

Because the data are managed holistically, data synchronization between the domains is automatic. The example described earlier, where FPGA pin swaps had to move from the PCB to the FPGA design process is now managed within the hierarchy automatically. The error prone process of manually synchronizing these design processes is gone. In addition, the project content can be easily extended to include documentation, release notes, test plans, test results, etc. The data management requirements are now being addressed systematically instead of as an external process.

IP Cores and Source Code Layer

The product development platform does not stop at the tool layer like most system design tool-chains. The platform architecture includes an IP core layer that includes IP common in today's products. For instance, common IP would include a USB, memory controllers, FPGA soft processors and UARTs (Universal Asynchronous Receiver/Transmitters). The IP layer would not only include the IP but have a design methodology that easily allows for usage of IP included in the platform and external IP that may be required for a particular market (telecommunications or industrial controls, for example). The IP implementation model may also include a layer of abstraction that removes the user from the details of constructing a processor-based system. An example would be the managing of the memory map as part of an embedded FPGA processor design where the hierarchical project enables the linkage from the FPGA and the embedded processor software development.

Another layer is the source code library for the embedded processor. Because the platform supports an embedded processor design, the user can write the code and include it as part of the project. The platform provides a library of standard device interfaces that dramatically shortens the design cycle. The platform manages the embedded software development as part of the hierarchical project. Each of the layers can be extended to allow companies to differentiate their product with special IP.

Altium Designer – A Platform-based Design Environment

Altium Designer is a system development platform engineered to free designers from the constraints of traditional limitations and the old "divide and conquer" approach. It provides a single unified solution for the entire design process and supports FPGA, PCB and embedded software development as well as a comprehensive library of IP and source code.

The ultimate measure of the effectiveness of a design environment, whether tool-chain or platform-based, is productivity. The system design landscape differs markedly today from when the tool-chain emerged for simple PCB design. First, the development demands now include complex products combining FPGA, PCB and embedded software development. Second, design teams are dispersed and the need for worldwide design concurrency is mainstream. Finally, business system integration is a growing necessity as engineers must make cost-effective component selections based on price, availability, their use in the design, and a release to the manufacturing process that integrates with PLM and ERP systems. So, if designers and engineers can build products faster and with better quality, then the choice becomes an easy one.

Conclusion

The tool-chain architecture has had a successful history but tomorrow's electronic product development demands can only be accommodated by shifting from being tool-centric to data- and process-centric. Today's companies require the ability to integrate different product technologies into a single hierarchical project or process. They can no longer afford the overhead of manually integrating tools into processes and adding the necessary data management support. While tool-chains cannot easily make the transition to support complex processes, including extending into business systems,

companies that have recently made the switch to a platform-based development process are seeing significant improvements in productivity.

Design companies who have moved to Altium Designer have said that this is as high as 400%.¹

Next generation system design is going to be based on platform architectures built on unified data models with a data management layer as the foundation for tool development and use. The platform will contain product IP and source code for standard functions so that companies can focus on product differentiation and not process management. The companies that can effectively differentiate their products will be more competitive and successful. Those companies bogged down with internal development process problems will not.

About the Author

Bob Potock (bob.potock@altium.com) is the Director of Technical Marketing for Altium North America and has been involved in board-based system and FPGA design throughout most of his career. He has held positions in software/hardware engineering, marketing and field operations at companies that include Mentor Graphics, AT&T Bell Labs, Intel, Burroughs, Cadnetix and NeoCAD. Bob holds a BSEE from Case Western Reserve University and an MBA from Regis University. He joined Altium in 2009.

¹ Altium company research, December 2009, 500 new customers surveyed. Survey results available from Altium on request.