S2C
FPGA PROTOTYPING SOLUTIONS

Getting the Most Out of FPGA Prototyping

S2C
FPGA PROTOTYPING SOLUTIONS

Table of Contents

EE Times

Introduction

Whether you are designing or verifying extremely complex cutting edge designs or more mainstream design, FPGA prototyping can help you achieve your goals with maximum benefit. The key to getting the most out of FPGA prototyping requires a good understanding of how this technology works and the FPGA prototyping solutions that match your design and verification requirements.

This eBook contains of series of articles published in EE Times that can help you navigate the world of FPGA prototyping technology – everything from overcoming FPGA prototyping hurdles to expanding the use of your FPGA prototype upstream of the design flow to using FPGA prototyping for even the largest designs. The eBook also gives you insight into how a complete prototyping platform can help for any design stage, any design size, and with enterprise-wide access, anytime, anywhere.

Five Challenges to FPGA-Based Prototyping

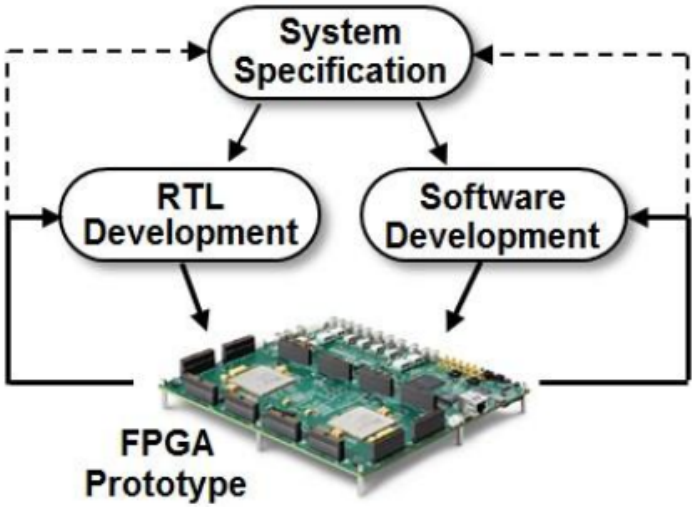
Ron Green, S2C Inc. 9/19/2014 01:05 PM EDT

The state of the art has progressed spectacularly since early forays into FPGA-based prototyping, but there are still challenges to be overcome.

What is FPGA prototyping, and why should I care? Not long after the introduction of FPGAs in the late 1980s, engineers seized upon these devices for building system prototypes of ASIC and SoC designs. Containing vast amounts of configurable logic, these versatile components were a natural choice for building and testing the latest designs. As designs grew in both size and complexity, FPGAs also grew to provide ever-increasing (equivalent) gate counts.

With earlier generations of FPGAs, it often took a large array of devices to fully accommodate a logic design. However, using today's devices with their mega-million gate counts, it may require only a handful of devices -- or even just one -- to implement a complete design.

The utility of a working FPGA prototype is undisputed. It allows hardware designers to develop and test their systems, and it provides software developers early access to a fully functioning hardware platform.



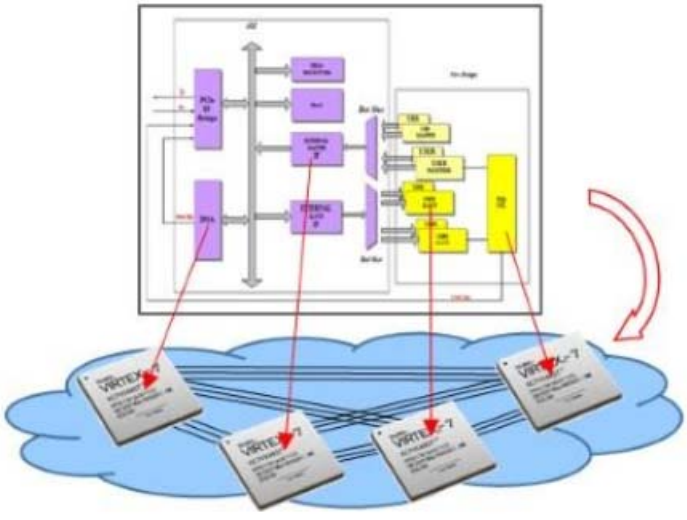
A prototype is used to develop both hardware and software iteratively. Exploring their interactions often has implications for the original system specification.

There are a number of key advantages that FPGA-based prototypes provide. These may be summarized as follows.

- **Beyond the limit:** At some point, software-only simulations hit the limit in terms of speed and capacity needed to run the latest designs effectively. If accurate software models are not available, prototyping may be the only option.
- **Ahead of the silicon:** An FPGA-based prototype can provide a functioning hardware platform long before silicon is available. This enables early software development tasks such as OS integration and application testing. Only a hardware prototype will run fast enough for practical software development.
- **Ideal test platform:** For designs that rely heavily on commercial IP, an FPGA-based prototype is an ideal test platform for ensuring all IP components perform together.
- **Seeing is believing:** FPGA prototypes can provide a demonstration vehicle for downstream customers, providing confidence the system is functioning as specified.

Five challenges Despite the advantages provided by FPGA-based prototyping, there are some significant hurdles to overcome. The five challenges presented below surfaced early on, and they haven't changed much over the years.

1. **Design partitioning:** Not many designs fit in a single FPGA; designs often must be partitioned across several devices. Initially, there were no tools to automate partitioning, so this task was tedious and error-prone. Worse still, designs that are split arbitrarily among several devices require a great deal of interconnect, which quickly surpasses the number of I/O pins available. Solving this problem requires a pin-multiplexing scheme.



Logical connections at the RTL level become a network of physical connections due to design partitioning.

2. **Long bringup:** Though engineering teams can design, build, and bring up a prototype, it can take a significant amount of effort. PCBs built to accommodate multiple FPGAs require numerous layers, and bringing up such a prototype typically requires verifying connectivity for more than 10,000 signals and pins. Mapping a design to FPGA prototype hardware can also be time-consuming and prone to error. When a design does not work in a prototype, it can be because of physical problems, design errors, or mapping issues. Without good techniques and the necessary tools, bringing up a prototype board can add months to your project schedule.

3. **Difficult debug:** It used to be that signals internal to an FPGA could not be probed unless they were brought out through the input/output (I/O). Fortunately, major FPGA vendors today have internal logic analyzers to address the visibility issue. However, many of these internal logic analyzers have several limitations, including support for only single FPGA debug, limited memory size using FPGA internal memory, and long place-and-route times to change probes.

4. **Performance:** After all this effort, the prototype may not perform as expected. Issues related to PCB physics -- such as signal routing, capacitive loading, and impedance matching -- will limit how fast the prototype can run. Considerations such as partition strategy, pin multiplexing, clock conversions, and FPGA timing constraints will also impact performance. Add to this the demands of high-speed transceiver interfaces -- such as PCIe, SATA, and 10G Ethernet -- that run in the GHz range. Without good design practices and reference designs, it is difficult to achieve the target performance in one pass.

5. **Reusability:** The ability to reuse a prototype (or even part of one) can save development time and lower implementation risk for future projects. But this is difficult to achieve with boards built for a specific project. As SoC designs grow in size, they may no longer fit in older FPGAs. If the interface to an external system is built directly on the prototyping board, it can't be reused for projects in which the interface is different.

Addressing the challenges With the advent of very large FPGAs, the problem of design partitioning diminishes to some degree. However, these devices also come with large pin-outs (e.g., some Xilinx Virtex-7 packages have nearly 2,000 pins), requiring boards with 18 signal layers (or more) coupled with multiple voltage planes. Large boards with a high number of interconnects still present many challenges.

To handle the latest designs mapped to the latest devices, solutions like the following are required.

- Quality PCBs that can accommodate high speeds (e.g., minimal clock skew, equal-length I/Os, stable power and ground, high signal integrity)
- Comprehensive self-test capabilities to detect and resolve hardware issues
- Integrated logic analyzer support to debug a design partitioned across multiple FPGAs
- Partitioning tools that can optimize results
- Systems with modular implementations that allow for scalability and reuse.

Due to these complexities -- and because development schedules don't allow time to develop a prototype from scratch -- commercial off-the-shelf prototyping and emulation systems have all but replaced internally developed solutions.

It's easy to see why this is the case. A robust off-the-shelf product provides guaranteed timing parameters, can be reused for multiple designs, and comes with dedicated technical support. The time spent on building a prototype from scratch is eliminated along with associated nonrecurring engineering (NRE) costs, which translates to realizing a working prototype faster with lower overall costs.

Selecting an FPGA-based prototyping system Ready-made prototyping solutions have become markedly more powerful in recent years, providing stable design environments and greater control through software and peripherals. There are a number of key parameters to consider if you're looking for an FPGA-based prototyping solution.

Capacity: Without enough gate-level capacity to accommodate your design, you can't build a prototype. Most systems need memory, too, so having sufficient memory available is critical.

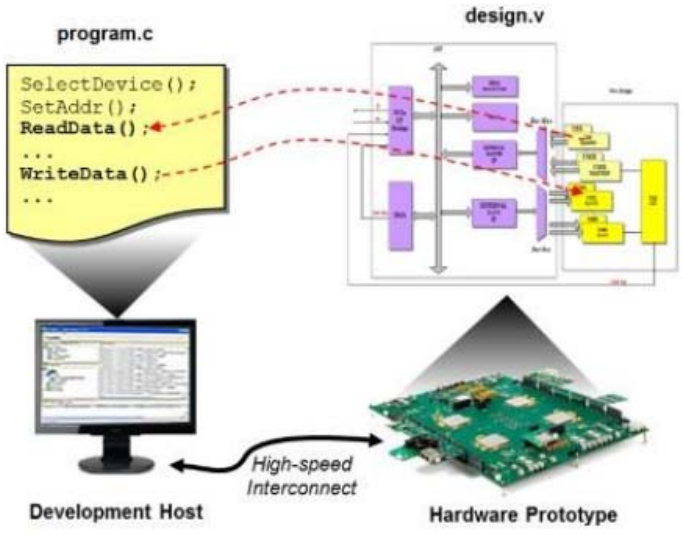
Scalability: This is the ability to add capacity as needed, including gate-level and memory capacity. Scalability also deals with extensibility -- adding components such as processors and communication interfaces to grow the system functionality. Another dimension of scalability is system replication, the ability to quickly (and cost-effectively) produce copies of a given design for multiple deployments or intersystem testing.

Performance: The whole purpose of building an FPGA-based prototype is to achieve sufficient performance for the next level of tasks: hardware verification, software development and optimization, and system test. If your prototype can't deliver the performance, it will fail at its original purpose. Achieving performance requires the PCBs to be specially designed with careful interconnect and I/O design, including load balancing, clock performance, and the use of high-speed communication channels such as gigabit transceivers.

Compile/partition software: The ability to partition a design reliably across multiple FPGAs is critical to handling large designs. It's very important to have a software tool to make this task easy and reliable. The partitioning software must also optimize I/Os, clocks, and pin multiplexing to ensure maximum performance. Since the partitioning algorithms and hardware architecture are closely related, they yield the best results if they are considered together, so finding an integrated solution for this is preferred.

Debug tools: Debugging a design partitioned across multiple FPGAs is all but impossible without a tool that helps set up probes and makes signals easy to track based on their RTL-level names. Debugging should use FPGA I/O efficiently and maintain a useful debug trace depth.

Transaction-based interface: A recent DeepChip report stressed the importance of "transactors" -- interfaces that bridge the gap between abstraction levels. This could be a well-known bus protocol such as AXI or an industry-standard transaction protocol such as SCE-MI that communicates between a design mapped to an FPGA and a behavioral simulation. Transactors extend the functionality of the system, allowing it to be used for algorithm/architectural exploration, system integration with virtual prototypes, and exhaustive testing through software-generated corner tests. Where transactors are used, they must be efficient to avoid becoming bottlenecks to overall system performance.



Transactors make early software development a reality.

Reliability: A reliable hardware platform is paramount. In order to ensure reliability, a system should have self-test capability and should include protection mechanisms to guard against problems such as overcurrent, overvoltage, and over temperature.

Reusability: The ability to reuse your prototype platform enhances its usefulness, speeds the process of developing new prototypes, and reduces overall costs.

Remote management: Managing a prototype remotely makes it easier to deploy for multiple users and projects. As a result, prototype resources can be located anywhere -- in a centralized facility for easy maintenance, for example -- yet used by groups distributed around the world. The ability to download the FPGAs remotely makes it easy to reconfigure a system for any purpose.

Whither the state of the art? The state of the art has progressed spectacularly since early forays into FPGA-based prototyping. Commercial systems abound in both size and approach -- from huge emulation boxes costing millions of dollars to modest providers with small, low-end boards.

In your search for a solution, when comparing offerings from different vendors, I would urge you to take a look at the systems from S2C Inc., the company where I work. We offer a wide range of rapid prototyping solutions providing tremendous gate-level and memory capacity, all built upon the latest Xilinx and Altera FPGAs. You'll find a family of systems that are scalable, extensible, and reusable; that support advanced debug tools that work across multiple FPGAs; and that include transactors such as an AXI bus interface and a C-API to ease integration with other systems. Offering many advanced features, S2C's FPGA-based prototyping systems are extremely cost-effective. They are among the most advanced in the world, and they are built to handle the most ambitious design challenges.

-- Ron Green *is the technical communications manager with S2C Inc. Prior to this, he was with Cadence Design Systems and Altos Design Automation. He has held various positions, including design engineer, AE, technical trainer, and technical writer. He appreciates clever designs, compelling materials, and the espressos at Mr. Toots Coffeehouse.*

FPGA-Based Prototyping: Big Design – Small Budget?

Mon-Ren Chene, S2C Inc. 11/19/2014 02:06 PM EST

Recent advancements in partitioning, debug, and scalability have made FPGA-based prototyping the ideal solution for even the largest ASIC/SoC designs.

A recent article, "Five Challenges to FPGA-Based Prototyping," explored some of the questions regarding FPGA-based prototyping and helped debunk the myths surrounding the issues of bring-up time, debug, performance, and reusability.

Over the next several articles, we'll look into the specifics of how FPGA-based prototyping can accelerate your design and verification.

There is a common misconception that FPGA-based prototyping is only suited to small designs and that the advantages of the technology diminish as designs grow. For verifying large designs, emulation is often the first technique that comes to mind. Now, emulation is fine for verifying large designs, but it comes at a cost. There is a penalty in both speed and price.

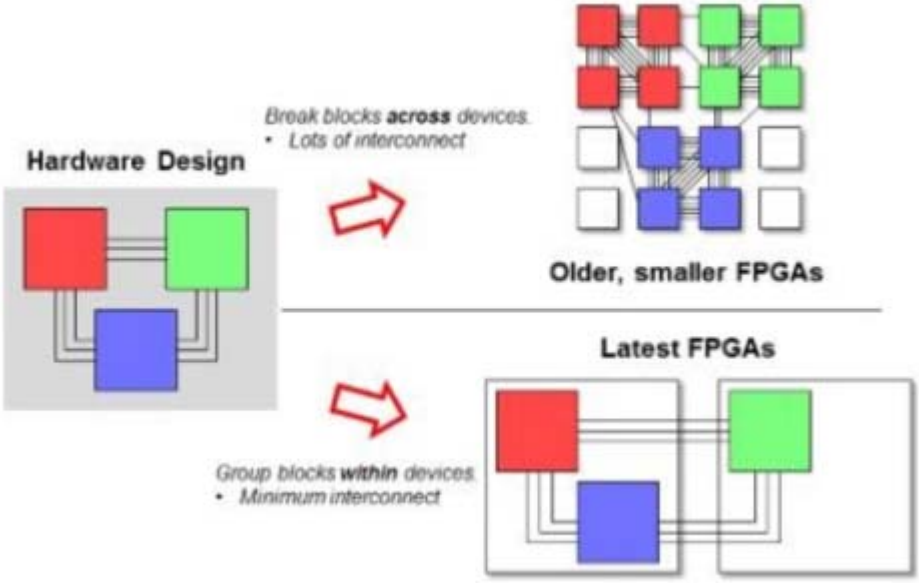
Emulation can be very slow at modeling your design, thereby impacting attempts at early software development. And emulation is expensive, putting it out of reach for companies on a budget. Every engineering manager is faced with considering time and cost tradeoffs to meet stringent time-to-market windows. To put these tradeoffs into perspective, let's first examine the advances in FPGA-based prototyping technology that help to close the gap between design size, speed, and cost.

FPGA technology advancements FPGA capacity has increased exponentially over the years. Today's largest V7 FPGAs from Xilinx based on a 28nm process have the capability to hold the equivalent of up to 20 million ASIC gates. By using an array of such FPGAs, we can leverage this further, with the potential to build a system that is close to half a billion ASIC gates. And the next generation of FPGAs, such as Virtex-UltraScale based on a 20nm process, holds even greater promise -- the prospect of building a practical and affordable system containing up to a billion gates.

FPGA-based prototyping can also run at much higher speeds than emulation. Internal FPGAs can run at hundreds of MHz, and I/Os can run at the multi-GHz range depending on the I/O standard. Even with very large FPGA prototyping systems, you can still expect anywhere from 5 to 20MHz system speed. This is an order of magnitude faster than emulation, which usually runs in the sub-MHz range.

To explore this idea of using FPGA prototyping for big designs further, let's look at the key technology considerations for adopting prototyping for these designs.

Partitioning Partitioning is key when prototyping a large design using FPGAs. In the past, with smaller FPGAs, partitioning required cutting through an IP block and distributing it across multiple devices. Today, most design blocks easily fit within one FPGA, so partitioning is primarily the process of grouping select IP blocks together onto a single device. Furthermore, the adoption of new FPGA I/O technology such as 1GHz LVDS enables the interconnection of more I/Os between FPGAs using a pin-multiplexing scher



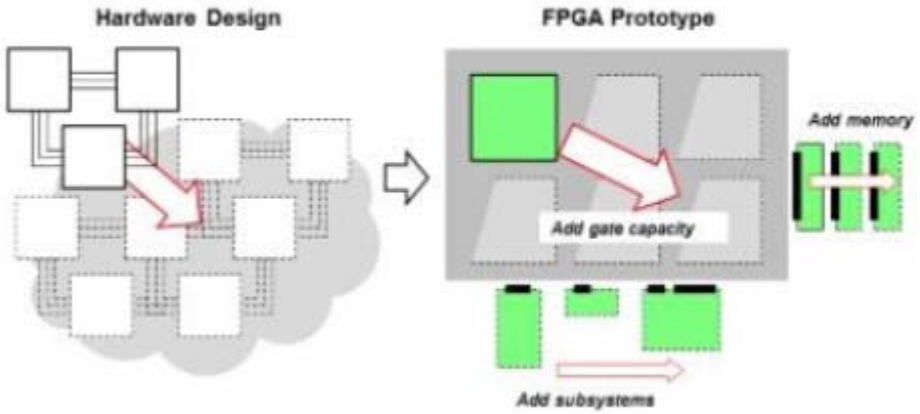
Partitioning software should also be considered. Along with other partitioning technology, the quality and usability of partitioning software such as Flexras and Mentor's Auspy have also significantly improved over the years. Similarly, the commercial partitioning tool from S2C has been used to partition a multi-core CPU design across 20 FPGAs.

Debug Partitioning and implementing the design in multiple FPGAs doesn't mean much unless you have the ability to debug the design effectively and efficiently. Many debug tools, including those provided by FPGA vendors, can only be used on a single FPGA. This is okay if you only have a single FPGA design or if you have the time to debug each FPGA on your system separately. But this can be both time-consuming and highly error-prone. In addition, most of these debug tools utilize the FPGAs' internal memories, which results in very limited trace depth. Debugging a multi-FPGA design works better if you can look at multiple FPGAs at the same time. This reduces both debug time and errors, while also using external memory for increased capacity.

S2C offers a unique multi-FPGA debug solution that uses an external FPGA to connect to multiple FPGAs at the same time via Gigabit Transceivers and stores the captured waveforms on 16GB of external DDR3 memory.

Due to its speed, FPGA-based prototyping is ideally suited for identifying and locating such bugs as critical corner cases, bugs related to subjective data, and bugs that require extremely long runs of actual system data. With larger designs, it is critical to adopt FPGA-based prototyping debug earlier in the process so as to help reduce late-stage ECOs.

Scalability FPGA-based Prototyping can be used at various design stages such as algorithm/architectural exploration, IP development, and full SoC in-circuit testing. Having a scalable or modular FPGA-based prototyping system will maximize your investment so various design projects at different design stages can all share the same platform.



Having said all this, you need to perform due diligence when evaluating potential FPGA-based prototyping systems. There are some such systems that are too specific in their use model to achieve this level of scalability. For example, FPGA boards utilizing PCIe are hard to scale beyond four FPGAs. Interconnections, global clocks, system control of multiple boards, power, and mechanical considerations become un-manageable. Most of these solutions also require too many cables making expansion unwieldy.

The following points should be considered in order to avoid any pitfalls:

- Adopt a platform that can be used by both smaller designs (requiring just one or two FPGAs) and bigger designs (requiring four, eight, sixteen, or more).
- Review the number of interconnections between each FPGA and how fast they run.
- Consider the reliability of the entire multi-board system, including mechanical, power supply, and self-test capabilities.
- Employ a global management solution that easily handles global clocks and resets everything via a software interface.

Budget tradeoffs As previously mentioned, engineering managers are consumed with overall budget tradeoffs. It is of utmost importance to strike the optimal balance between cost, resources, and time in order to achieve the necessary goals. Now, with the advances in FPGA-based prototyping technology to handle large designs, managers should take a second look at how to apply resources to their design.

Emulation, although effective, is expensive. Emulation also has debug limitations that FPGA-based prototyping can tackle much faster, thereby making a strong case for applying FPGA-based prototyping sooner so as to save time later in the flow. If a manager can apply a less expensive FPGA-based prototyping methodology sooner in the verification process, this could save hundreds of thousands of development dollars.

The cost and speed of FPGA-based prototyping has always been a good fit for smaller designs. The solution provided by S2C is a unique example of a system addressing these needs. Our offering allows one, two, and four FPGAs on a single prototyping board that can then be used in conjunction with other boards (using stacking/cabling or located in a chassis) to provide a system with up to 32 FPGAs. Using boards carrying a single FPGA module to form a big system offers maximum flexibility; using boards carrying four FPGA modules achieves minimum cabling.

The end result is that recent advancements in partitioning, debug, and scalability have made FPGA-based prototyping the ideal solution for even the largest ASIC/SoC designs.

— Mon-Ren Chene *is currently the Chairman and Chief Technology Officer for S2C. He has more than 30 years of engineering and management experience in EDA and FPGA software/application development. He co-founded Osprey Design Systems that later merged with Aptix, where he served as Software Architect and VP of Software Development. Chene also held engineering and management positions at Quickturn Design Systems, Xilinx, Cadence Design Systems, Silvar-Lisco Design Systems, and NCA. He holds five US Patents and three pending patents. He is a graduate of Stanford University with an MS in operations research.*

— Max Maxfield, Editor of All Things Fun & Interesting

Transactors -- Expanding the Role of FPGA-Based Prototypes

Ron Green, S2C 2/27/2015 05:30 PM EST

FPGA-based prototypes offer unbeatable flexibility, capacity, and speed. Extending their functionality through the use of a transactor interface opens up tremendous possibilities to designers.

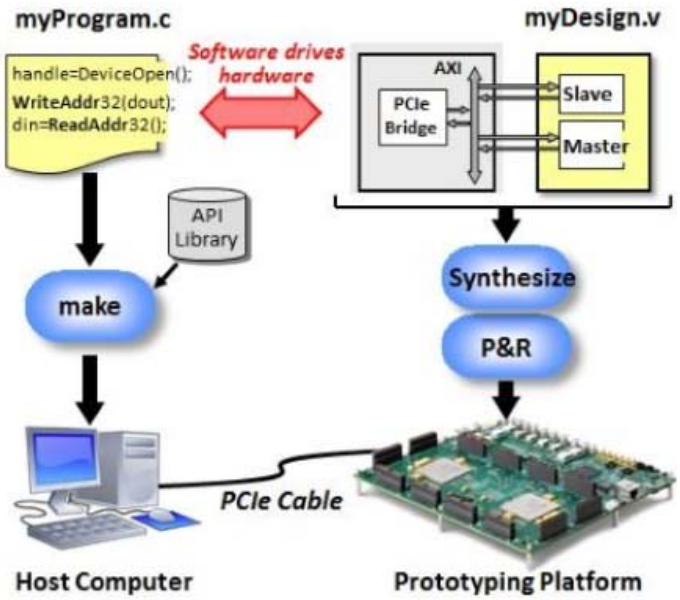
Continuing the discussions we began by considering the *Five Challenges to FPGA-Based Prototyping*, we'll now take a look at some recent functionality this is now available with the most sophisticated prototyping platforms.

Power to spare It wasn't all that long ago that FPGA-based prototypes were the sole province of hardware designers and lab technicians. Viewed as finicky boards covered with rows of devices and bristling with cables, they were relegated to back rooms where engineers would endlessly tinker with them in a desperate effort to bring up designs of limited size and complexity.

No more. Today's FPGA prototypes represent muscular platforms for developing ultra-large systems running at blistering speeds. With this kind of power, these systems are used for a wide range of tasks, including design integration, system verification, and software development (see also *Big Design -- Small Budget?*).

This solution is well-suited to designs fully rendered in RTL that can be mapped to an FPGA. But what about cases where portions of the design are still only available as behavioral models in descriptions such as C++ or SystemC?

FPGA-based prototypes to the rescue... again The latest systems now provide transaction-level interfaces -- often referred to as "transactors" -- that bridge the abstraction level between behavioral models and live hardware. Transactors offer a way to communicate between software running on a host machine and an FPGA-based prototyping platform that often includes memories, processors, and high-speed interfaces.

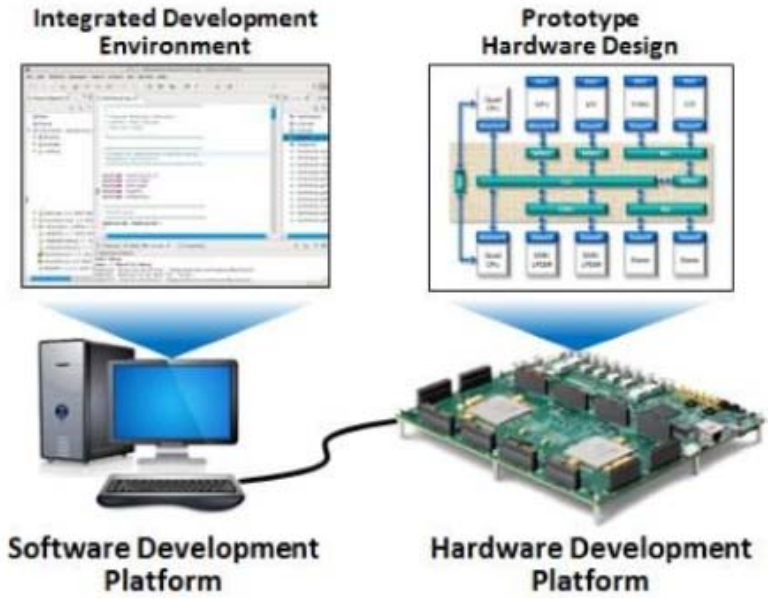


One example of this is the ProtoBridge system from S2C, which supplies a transactor interface between a software program and the world of AXI-compliant hardware. There are two key parts to this: an AXI-to-PCIe bridge that connects to a host computer, and a C-API that communicates to the design through the bridge. The software-to-AXI transactor offers new flexibility to designers building ARM-based systems. Also, coupling this to a PCIe interface supporting transfer speeds up to 500 megabytes/second provides a perfect development platform for data-intensive applications.

A Cornucopia of Applications A system like this allows designers to maximize the benefits of FPGA-based prototypes much earlier in the design project for algorithm validation, IP design, simulation acceleration, and corner case testing. A prototype combined with a transactor interface makes a range of interesting applications possible throughout the design flow:

- **Algorithm/architecture exploration:** When a new system is developed, behavioral models are created to explore different algorithms and architectures. But new systems are typically built upon the foundation of existing IP, often available in RTL. A transactor interface allows behavioral models to be co-simulated with RTL models, thereby exercising the full system regardless of the abstraction level or language used to define the different blocks.

- **Early software development:** The ability to run software at the earliest stage possible is becoming more important. It may be hard to access all the high-level models needed to implement a virtual platform capable of running software. Utilizing an FPGA with a transaction-level link to an ESL (electronic system level) design environment offers an effective solution. Tying together the programmer's development environment with the target platform from the hardware team allows the software to be developed earlier in the design cycle, and provides a way for each group to validate their changes against the work of the other team.



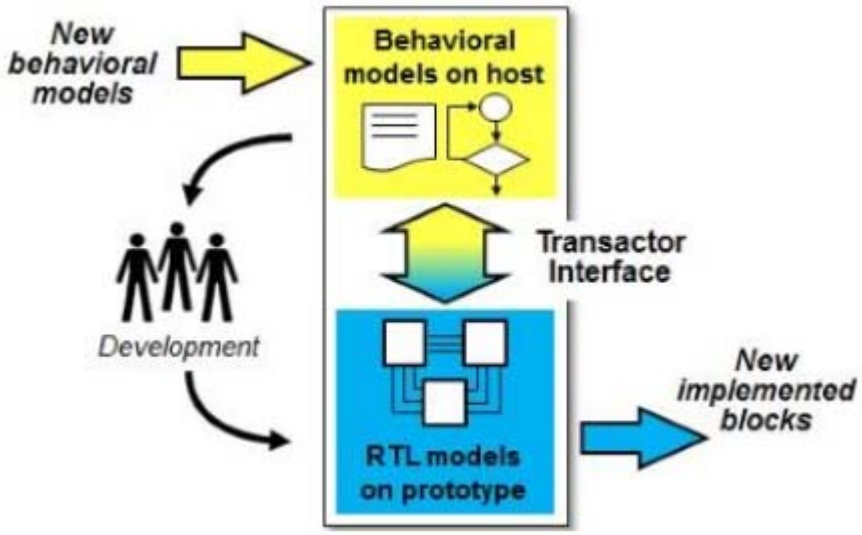
- **Block-level prototyping:** Mapping an entire design to an FPGA prototype can be challenging, especially when a large number of devices is needed. The use of transactors allows mapping designs block-by-block and verifying each block against its RTL-based simulation. This can be a very effective methodology, especially when separate teams are developing IP blocks independently. This approach can prevent issues from surfacing during integration.

- **Simulation acceleration:** RTL-level simulation alone can be prohibitively slow for verifying large designs. Mapping designs from a simulation environment to an FPGA prototype provides a high-performance, cycle-accurate test environment. Systems like this can run in the hundreds of KHz, typically out-pacing RTL-level simulation by three orders of magnitude.

- **Design debugging:** Debugging a complex design in an FPGA can be difficult, especially if access to a large amount of data in memory is needed. A transaction-level interface makes it both easy and fast to transfer large amounts of data in and out of memory. This can be used to read design conditions stored in memories and registers, or to write conditions back to memory, quickly getting to the design state required for debugging.

- **Corner case testing:** In-circuit testing is probably the most important reason for doing prototyping today. But in-circuit tests are usually based on un-constraint random tests, which don't always ensure complete test coverage, and which may not reflect real operating conditions at all. Using a transactor interface allows test cases developed in simulation to be run directly on the prototype, making these tests instantly available and insuring compliance. Moreover, these tests can be easily extended to large data sets, providing coverage for corner cases and hard-to-find bugs.

The great facilitator The addition of a transactor interface to an FPGA-based prototype facilitates development of new systems in interesting ways. As behavioral models are introduced, architectures become refined and block functionality determined. These blocks are eventually defined and implemented as part of the new system. But blocks that are defined and rendered in RTL become IP for the next generation of systems, allowing the cycle of development to repeat. In this way, the FPGA prototyping platform becomes a great facilitator -- the engine of system advancement.



This combination is too powerful to ignore. FPGA-based prototypes offer unbeatable flexibility, capacity, and speed. Extending their functionality through the use of a transactor interface opens up tremendous possibilities to designers. These are tools and techniques no team should be without.

-- Ron Green *is the technical communications manager with S2C Inc. His experience includes design and verification, IC layout, analog tools, cell characterization, emulation, and FPGA-based prototyping. A Silicon Valley veteran, Ron has held positions in engineering, applications, program management, and technical marketing. He appreciates clever designs, compelling materials, and the espressos at Mr. Toots Coffeehouse.*

FPGA Prototyping of System-on-Chip (SoC) Designs

The need for a complete prototyping platform for any design size at any design stage with enterprise-wide access, anytime, anywhere.

Mon-Ren Chene, S2C 4/8/2015 06:30 AM EDT

Today's off-the-shelf FPGA prototyping systems have established their value in every stage of the system-on-chip (SoC) design flow. Moving beyond traditional applications such as in-circuit testing and early software development, this technology has expanded to encompass functional design and verification (see also *Transactors -- Expanding the Role of FPGA Prototypes*).

FPGA-based prototypes work with electronic system level (ESL) design environments to refine, validate, and implement the chip's architecture, and with simulation tools to achieve an order of magnitude (or more) increase in verification speed.

There are several drivers of this technology: the need to quickly construct high-performance prototypes; the demands of growing design size and complexity (see also *FPGA-Based Prototyping: Big Design – Small Budget?*); and the need to utilize prototypes as an enterprise-wide resource. Globalization has replaced localized design teams with teams that are geographically-distributed. Consequently, FPGA prototyping solutions must now provide network access and remote management capabilities coupled with the ability to expand resources such as memory or add-on components. This allows realizing multiple hardware and software implementations for numerous, geographically-dispersed teams.

The FPGA prototyping system must offer enterprise-wide accessibility -- a complete prototyping platform is one that operates at any functional design stage, with any design size, and across multiple geographical locations. All of these capabilities must be available on-demand and be remotely-accessible at all times. Such an approach significantly increases engineering productivity and reduces the end-product's time to market, while increasing its return on investment (ROI), as well as increasing the lifetime ROI of the FPGA prototyping platform itself.

Growing SoC design challenges SoC size and complexity are increasing at an exponential rate. According to a keynote presentation by Gary Smith at the International Technology Roadmap for Semiconductors Conference in 2013, potentially available SoC gate counts will quadruple from 420 million in 2014 to 1.68 billion in 2020. International Business Strategies (IBS) reported that software development and hardware verification are the two leading factors in total SoC design cost (see Figure 1).

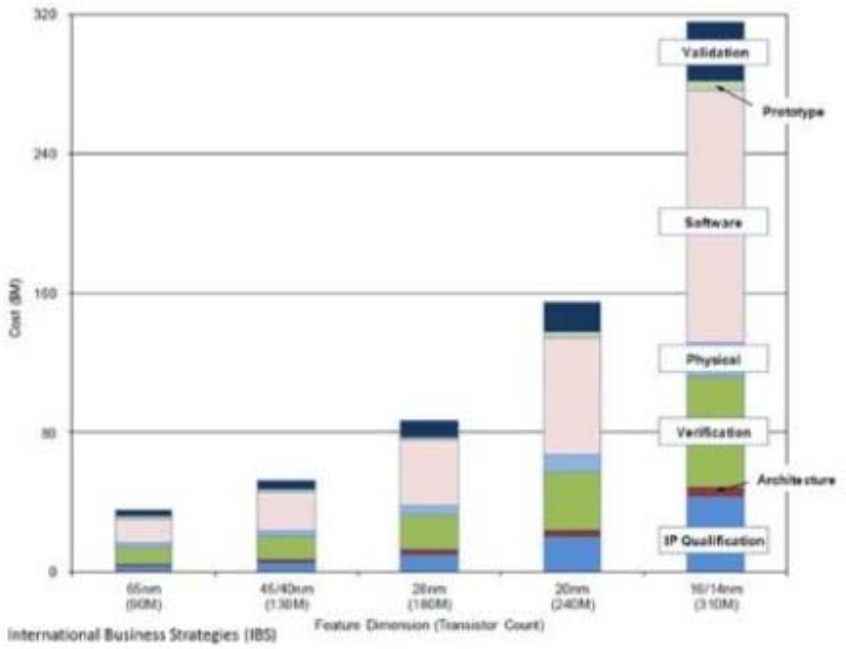


Figure 1. Software development and hardware verification are the predominant factors in SoC design cost (source: IBS). (Click here to see a larger image.)

These software- and complexity-driven cost and effort increases are accompanied by an elevated risk of late delivery, and even the possibility of outright failure. Cost and risk are generally mitigated by the extensive use and reuse of intellectual property (IP) -- both silicon and software -- but the complete silicon/software design must nonetheless be prototyped and tested as a whole.

FPGA-based prototyping solutions: Addressing today's needs For an FPGA prototype to meet the requirements of this "whole design", it must address the following criteria:

- User access
- Compile/partition efficiency
- System interface capability
- Scalability
- Extensibility
- Reusability
- Analysis and debug capability
- Application throughout the functional design flow

Utility of current FPGA-based prototyping systems

The key criteria for evaluating the utility of an FPGA-based prototyping system are as follows:

1. **Access** to FPGA prototyping systems must not be constrained by the use of localized systems that require local management and control. Limited access can present a significant hindrance to modern SoC design teams -- especially software development teams -- which are often globally distributed.

2. The **compile and build environment** must incorporate important features such as the ability to partition a design automatically and/or with user guidance; automatic pin-multiplexing insertion and clock analysis. Also important are a convenient user interface to FPGA-specific place-and-route (P&R) tools allowing for quick flow turnaround for changes and ECOs.

3. **Performance**, which is the key reason teams develop FPGA-based prototypes. FPGA-based prototypes can be expected to realize system speeds in the tens of megahertz -- some have been known to run at 100MHz and more. High-speed FPGA prototypes enable early software development.

4. **High-speed interfaces and add-ons**, such as PCIe, USB, 10GE, ARM Debugger, and DDR memory are important for building a complete development platform. Transaction-level interfaces such as a CAPI and AXI bus protocol support greatly expand the utility of the system.

5. The ability to **scale and extend** the system is also an important consideration. This involves adding gate-capacity and memory, as well as processors and communication interfaces to grow the system's functionality.

6. **Reusability** is inherent in off-the-shelf FPGA prototyping systems. A system's reusability in subsequent designs is determined by the quantity and diversity of its resources -- gates, memory, and processing power. These resources must grow to remain up to date with changing functionality requirements.

7. **Analysis and debug** must not be limited to one FPGA at a time, which makes whole-system debug slow and tedious. Signal probing should work easily with designs partitioned across multiple FPGAs; a deep debug trace capability must be provided with probing schemes that maximize the use of FPGA pin I/O.

8. Support for a **mixed-level prototype**. Often during development, not all blocks are available in RTL. A complete prototyping system should support behavioral blocks running on a host computer to interface and communicate with RTL blocks mapped to the FPGAs.

The complete solution

Given the attributes and shortcomings of existing FPGA-based prototyping systems, what should be the attributes of the next generations of systems? As noted, the coming generations of off-the-shelf FPGA prototyping solutions must offer greater choice and flexibility in the deployment of resources. Consequently, the coming generations must take a "complete prototyping platform" approach as follows:

1. Provide sufficient **performance** to operate as software development platforms. This requires PCBs with superior signal characteristics, clocking strategies, and connector structures.

2. Provide **capacity, scalability, extensibility, and reusability**. Modern FPGA-based prototypes can support designs from 20M to 500M gates. Add-on DDR2/3 modules can quickly create systems with multi-gigabyte memories. Peripherals and processors available on daughter cards make it easy to add various IP functionality.

3. Increase **scalability, extensibility and reusability** by enabling the deployment of an ever-increasing range of IP -- both silicon and software -- and pre-designed board-level subsystems. This requires the availability of copious IP and board-level reference design options.

4. Support the latest **high-speed interfaces** such as PCIe, HDMI, and 10Gig Ethernet.

5. Include an easy **compile and build** environment to accelerate the process of prototype bring up and ease the processing of design ECOs. Automatic partitioning tools should accept user input that can result in faster-running prototypes.

6. Offer global **remote access** to centralized prototyping resources, enabling access by multiple, geographically-distributed teams. This can be achieved by cloud-based control and storage.

7. Support **behavioral/transaction-level interface** and standard bus protocols such as AXI to ease system integration and provide a platform for software development.

8. Perform **analysis and debug** of the whole SoC design at full system speed with a focus on "deep" hard-to-find bugs. Simultaneously, it must achieve at least an order of magnitude increase in signal and cycle observability without consuming FPGA resources such as gates, memory, and I/O connectivity. This requires the implementation of structures that offer "pervasive observability" of design functionality. Fast probe-swapping is essential.

9. Continue to team with **functional design and verification tools** to perform tasks at every stage of the functional design flow, such as cross-leveraging the strengths of FPGA prototyping and simulation. This requires tight, well-integrated bridges between the diverse design environments.

Conclusion A modern FPGA-based prototyping system must meet a number of demanding criteria to help designers realize their latest system-on-chip designs. An extensible, scalable system must offer a variety of both hardware and software interfaces. High-performance and extensive debug capabilities are critical requirements. The ability to function as an enterprise-wide resource, with the easy access and configurability of a cloud service, multiplies the value of such a system. Meeting these criteria and combining features in a rich set of functionality qualifies a system as being a truly complete prototyping platform.

Mon-Ren Chene *is currently the Chairman and Chief Technology Officer for S2C. He has over thirty years of engineering and management experience in EDA and FPGA software/application development. He co-founded Osprey Design Systems, which later merged with Aptix, where he served as Software Architect and VP of Software Development. Chene also held engineering and management positions at Quickturn Design Systems, Xilinx, Cadence Design Systems, Silvar-Lisco Design Systems, and NCA. He holds five US Patents and three pending patents. Chene is a graduate of Stanford University with an MS in Operations Research.*