FPGA Prototyping Primer

S2C Inc.
1735 Technology Drive, Suite 620
San Jose, CA 95110, USA
Tel: +1 408 213 8818
Fax: +1 408 213 8821
www.s2cinc.com

What is FPGA prototyping?

FPGA prototyping is the methodology to prototype SoC and ASIC designs on FPGAs for hardware verification and for early software development. This methodology is sometimes referred also as ASIC prototyping or SoC prototyping.

Prototyping SoC and ASIC designs on FPGAs has become a mainstream verification methodology for hardware design as well as a method for early software and firmware co-design.

Why is it important to prototype?

1. With SoC designs becoming more complex than ever, designers are increasingly finding it difficult to rely only on software simulations to verify that their hardware design is correct -- due to both simulation speed and modeling accuracy limitations. Running your SoC design on a FPGA prototype is the most reliable way to ensure that your design is functionally correct.

2. But, hardware verification is no longer the number one reason why most designs today are prototyped on FPGAs. Early software and/or firmware development on FPGA prototypes, pre-silicon, has become more important as many unforeseen software bugs stem from the complexity of integrating operating system (OS), applications, and hardware. Most projects can't afford to wait until the silicon is back from the foundry to start software testing. An at-speed FPGA prototype allows for many extra months of rigorous software development and testing at the crucial software-hardware integration stage.

3. FPGA prototyping is also critical if your SoC design utilizes many commercial IPs. Prototyping on FPGAs will be your most reliable method to make sure all these IPs work well together.

4. FPGA prototypes can also be used as demo platforms to the SoC customers for getting them interested in the chip you build and allow you to work with them on improving features before chip tape-out.

Current Challenges in FPGA Prototyping

Long Bring-Up Time

Designers need a clear board-testing strategy prior to manufacturing because there will be thousands of pins to test. Without a good test plan, it will be hard to pinpoint problems if the board does not operate according to specifications.

The key question you should answer before you decide to build your own FPGA prototyping board is “How long it will take to bring up the board and test thousands of IOs?” In many cases, your

FPGA prototypes might require a re-spin in order to work, which may add up to 2 months to your project.

Solution: Source a prototyping tool with comprehensive self-test capabilities to detect and deal with hardware issues as soon as they happen.

Performance

Designers always prefer to run the FPGA prototype at real-time or near real time speed to more closely approximate performance of the end product.

FPGA prototyping performance might not be able to achieve the actual performance of the target SoC/ASIC. There are generally two reasons for this performance discrepancy: FPGA limitations and PCB board limitations.

Solution to FPGA Limitations: Use different synthesis tools, tighten timing constraints, or modify your design.

Solution to PCB board limitations: Work with a well-designed prototyping board with equal-length clock traces, equal length IOs, and stable power and ground to accommodate prototyping at high speeds. Impedance matching may also be needed for extreme high-performance IOs such as DDR memory interfaces.

Reusability

The ability to reuse your existing prototype or at least part of your prototype can save development time and lower implementation risk for future projects.

SoC design sizes continue to grow as new semiconductor processes become available and consumers desire new application features. Your FPGA prototype will probably require upgrading as well. Many designers like to build the interface to external systems directly on the FPGA board. This approach may serve for single projects but renders both the FPGA and the peripheral interface unfit for reuse in other projects if the design size is larger or the

peripheral interface is different.

Solution: Structure your FPGA prototyping system into independent FPGA and interface boards to create modularity that allows for a high degree of FPGA reusability.

Design Partitioning

Design partitioning is needed for designs that cannot fit into one FPGA.

Partitioning problems arise when the number of FPGA pins is limited, and is further magnified as the number of FPGAs increases. There are generally two main issues to deal with: - How do you interconnect the IOs among multiple FPGAs on your prototype? - How do you partition your design to fit the architecture of your FPGA prototype board? Hand partitioning a design to multiple FPGAs is error-prone and time-consuming. Examples of potential problems include: insufficient number of pins, clock synchronizations, failure to meet performance expectations, and external pin entry point.

Solution: Partition the design at the block-level, then utilize a user-guided partitioning solution to quickly to arrive at an optimal result in terms of performance, area balancing, and enabling incremental revisions. Employing this type of integrated HW/SW prototyping system upholds interconnect quantity and quality (via HW) and automatic partitioning (via SW), thereby saving development time.

Debug-ability

Taking steps to ensure a design is debug friendly minimizes the time spent on debugging later on in the process/schedule.

It's unlikely your design will work the first time after you download it to a FPGA. Your design might not be working because 1) the FPGA prototype itself has problems, 2) the design may have a problem and/or 3) an error has accrued during design compiling (e.g. wrong pin assignments). Ideally, you would first need a good testing method to identify if the hardware is running correctly and all the pins in the design are functioning normally. Then, either an External

Logic Analyzer and/or Internal Logic Analyze (e.g. Xilinx's Chipscope) would be needed to identify problems arising from design or mapping errors. This sometimes requires tedious work to get internal signals out to the boundary for external LA probing. Moreover, most internal logic analyzers today do not support debugging designs mapped to multiple FPGAs – making your debugging job even more difficult.

Solution: Select a prototyping system that includes self-test function capabilities and supports logic analyzers that can debug multiple FPGAs.

Build-Your-Own vs. Off-The-Shelf: Which Is Right for You?

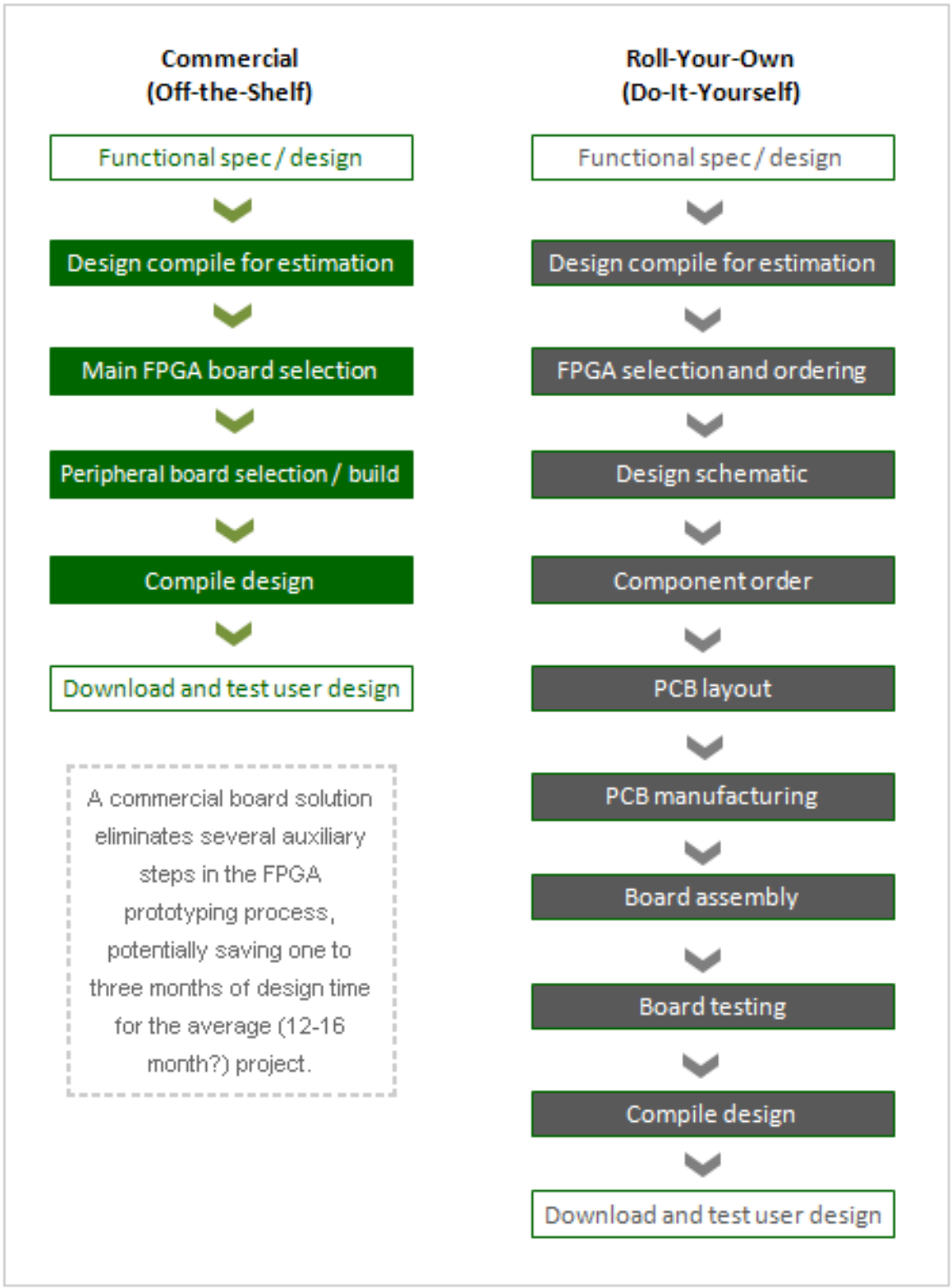
Do you view the decision to build-or-buy your prototyping system as a short-term cost-driven tactic or part of your functional verification strategy -- with implications on future product development?

For chip design companies that operate on huge economies of scale and adhere to a great degree of format-specific processes in their design flow, full custom capability may be important. However, given execution risk along with time and cost constraints - purchasing a proven prototyping system becomes hard to ignore and a much more compelling choice (especially for companies operating at less than huge economies of scale). Verification can consume from two-thirds to four-fifths of total development time. A mature off-the-shelf product provides guaranteed timing parameters, can be re-used for multiple designs, and comes with dedicated technical support – minimizing the risks of porting a prototype that doesn't work according to specifications. The time spent on building a prototype from scratch is eliminated, along with associated non-recurring engineering costs, which translate to faster time-to-prototype and in the end, lower overall costs.

FPGA Prototyping Design Flow - First Time

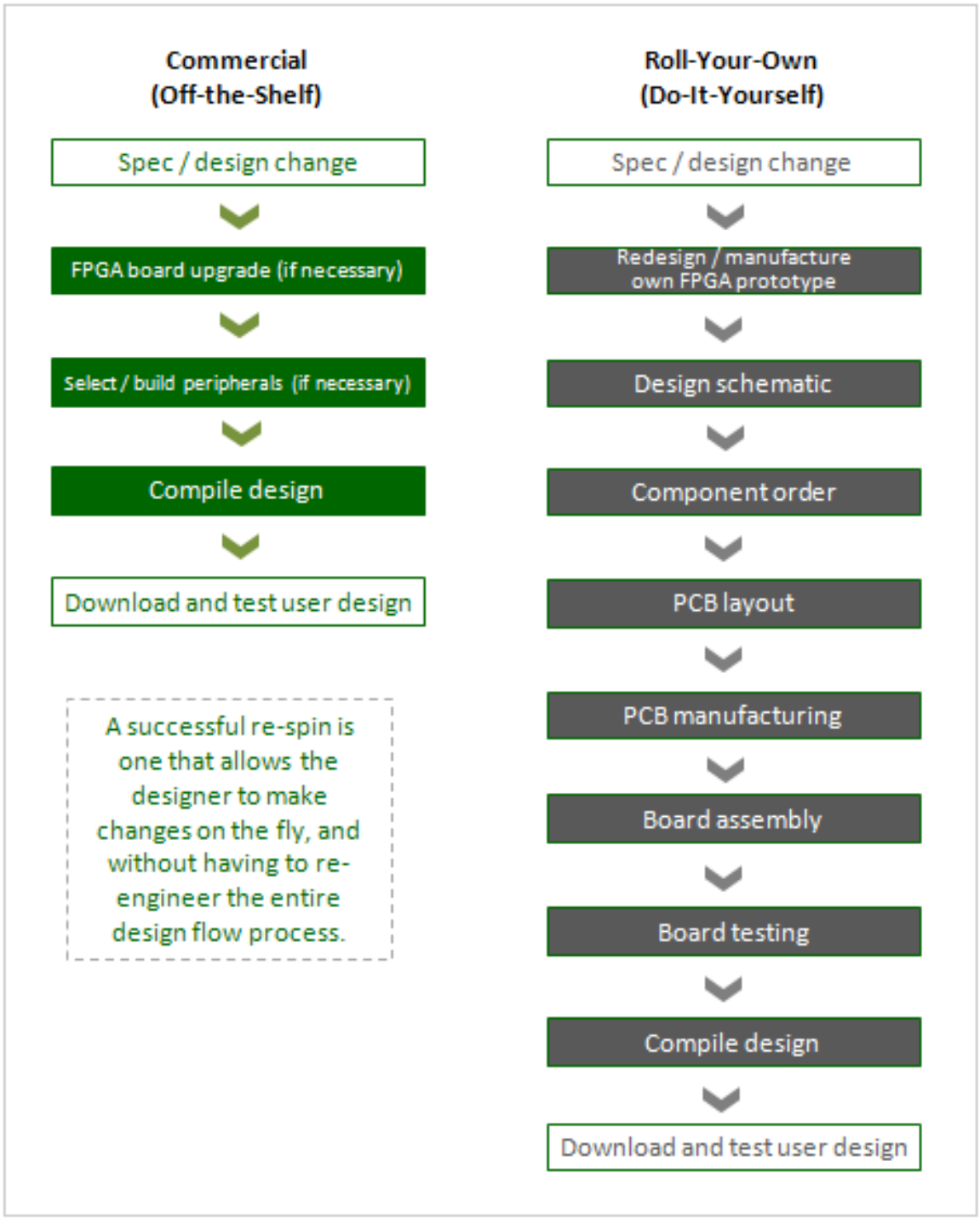
Off-the-Shelf vs. Build-Your-Own

Ready-made prototyping solutions have matured considerably in recent years, providing stable design environments and greater control through software and peripherals. The traditional build-your-own in-house modules may not only increase project time but also result in greater overall development costs.


Commercial
(Off-the-Shelf)
Functional spec / design
Design compile for estimation
Main FPGA board selection
Peripheral board selection / build
Compile design
Download and test user design
A commercial board solution eliminates several auxiliary steps in the FPGA prototyping process, potentially saving one to three months of design time for the average (12-16 month?) project.
Roll-Your-Own
(Do-It-Yourself)
Functional spec / design
Design compile for estimation
FPGA selection and ordering
Design schematic
Component order
PCB layout
PCB manufacturing
Board assembly
Board testing
Compile design
Download and test user design

FPGA Prototyping Design Flow - Reiteration

SoC design specifications must often be modified due to market or technical reasons. Therefore, it is important to keep the FPGA prototyping environment flexible, in case such events occur. The diagram below illustrates the flow difference when a design undergoes significant changes.



Summary

FPGA prototyping has many benefits including overcoming speed accuracy limitations from simulation, developing software and firmware much earlier, testing IP integration with high reliability, and creating demo platforms for early customer engagements. And although, FPGA has some perceived challenges, overcoming those challenges with the right prototyping solutions is easy and cost-effective leading to much faster time-to-market.