A Framework for FPGA Design Planning

by Jeffrey Lin
Senior Manager, Global Communications Services Group
Xilinx, Inc.
jeffrey.lin@xilinx.com

This time-tested FPGA development framework is your route to flawless project execution.

The capabilities and performance of modern FPGAs have long since reached the point where these devices now sit in a central location within systems and serve as a primary data-processing engine for many products.

Given the important role FPGAs play in so many applications, it is more important than ever to approach FPGA design with a formal and methodical development process. The purpose of such a process is to avoid costly and time-consuming design deficiencies discovered late in the development cycle that can have a potentially disastrous impact on schedules, cost and quality.

The Xilinx Global Communications Services Group has long been using a time-proven design framework to develop and deliver turnkey FPGA designs to Xilinx customers for products ranging from medical-imaging processing engines to self-learning network switch engines. It is a framework we have developed and evolved over the course of designing, developing and delivering hundreds of FPGA designs.

The framework we use covers everything from system architecture considerations to FPGA development and test planning. Let's take a closer look at this framework, with a focus on the FPGA hardware, in hopes that other engineering teams might find it a useful tool for tackling complex FPGA design projects of their own.

FRAMEWORK OVERVIEW

The framework is an iterative, top-down methodology for designing hardware in FPGAs. First, the planning starts at the system architecture level to determine FPGA functionality. Then we progressively refine the features implemented in the FPGA using the known capabilities and performance of the FPGA device.

In addition, the implementation of large FPGA designs necessitates well-defined development, simulation and verification plans. The framework aids us in crafting those plans as well. In short, the framework can be summarized as shown in the flow chart in Figure 1. The focus for this discussion is on the Planning and Documentation portion (top section).

SYSTEM ARCHITECTURE

In the context of this discussion, system architecture refers to the division of functionality among system software and hardware. In particular, the focus is on the subdivision of the hardware functionality into FPGA and other microchip components, with the assumption that product-level requirements are already defined. For example, a marketing or product-definition organization may have already weighed in with product requirements.

In the system architecture phase, the idea is to bring clarity to how these product requirements are to be realized in a physical product. For the FPGA, the primary decision is what features and functions should be implemented in the programmable device and furthermore, what is reasonably achievable in an FPGA.

By defining the high-level requirements of the FPGA up front, you can avoid costly design and requirements changes before you have proceeded too far down the development process. At this early stage, the definition of the system architecture will guide you to several important decisions that affect both development time and product cost.

At this level of discussion, only the general outline of the FPGA features is required. The detailed features and implementation requirements will be defined in a later stage during the FPGA requirements definition. Participants in this discussion should include someone familiar with the system-level requirements, someone with system-level architecture design knowledge and someone familiar with the features and capabilities of FPGAs.

In the context of the FPGA, there are 10 important questions to answer.

1. What is the feature list to be implemented in the FPGA?
2. What are the technical trade-offs for implementing features in the FPGA vs. using non-FPGA components?
3. What is the design effort/cost to implement in FPGA vs. non-FPGA components?
4. What custom features or processing are required?
5. What does the flexibility of the FPGA bring to the functions?
6. What future-proofing risk mitigation should you consider?
7. Can features be consolidated within the FPGA from multiple non-FPGA components?
8. Based on the design features to be implemented, what are the FPGA device choices?
9. Can the features be implemented in an FPGA?
10. What non-FPGA devices are required and how do these devices interface to the FPGAs?

FPGA ARCHITECTURE

The FPGA architecture is the microarchitecture and chip-level data-flow design at the physical level on the FPGA device. Your team should design this architecture concurrently with the system-level architecture for device sizing, selection and feasibility.

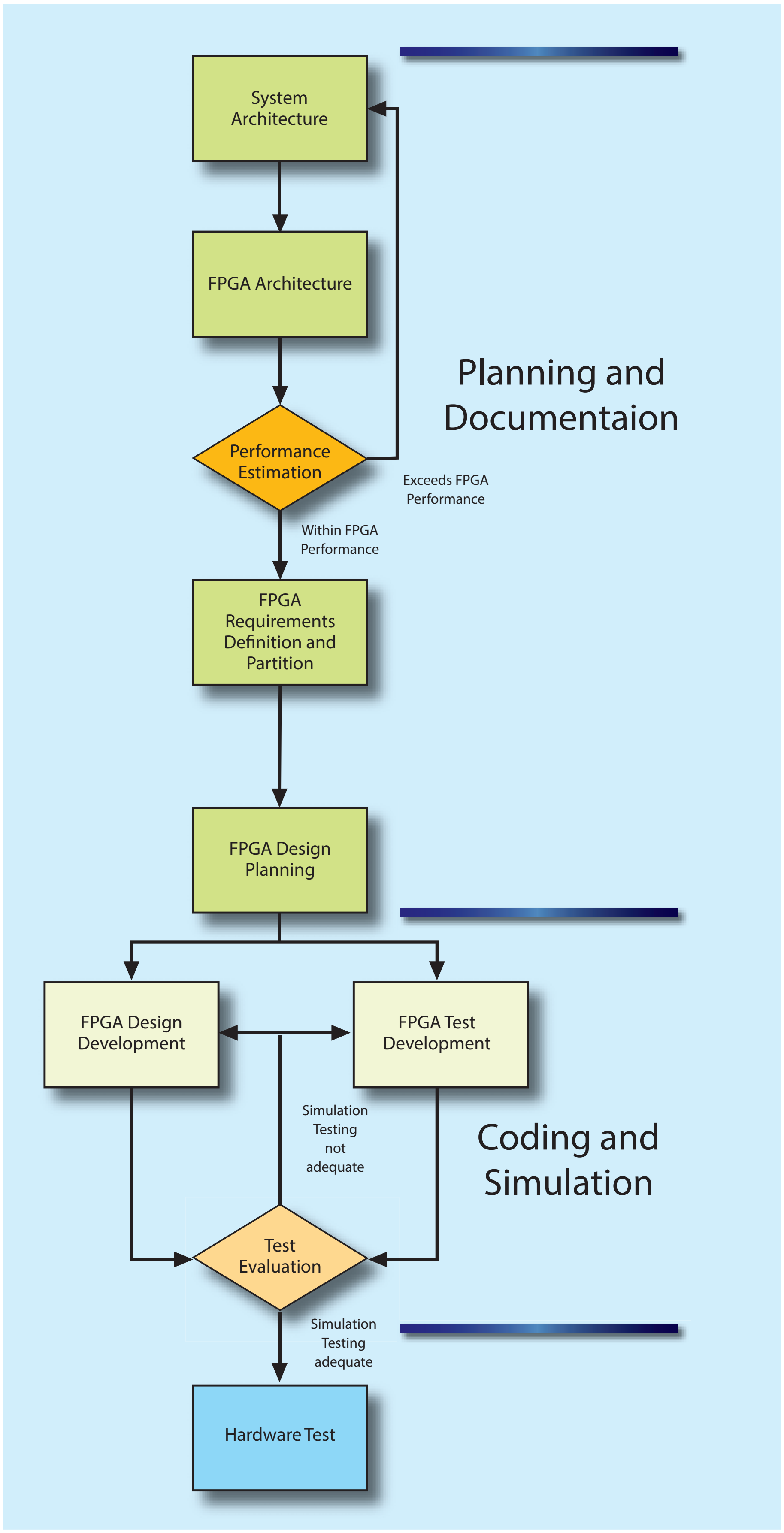

Figure 1 – The FPGA development framework

The purpose of defining the FPGA architecture is to ensure the system architecture requirements can be implemented with accurate, realistic and achievable design requirements in the FPGA.

This level of discussion requires intimate knowledge of the features and capabilities of the FPGA fabric and resources. It should therefore be undertaken with participants who are experienced FPGA designers. At this stage you must consider the decisions for performance targets of the FPGA, potential risk areas, as well as available FPGA fabric resource utilization.

During the FPGA architecture definition, it's possible you might find that the system-level requirements and architecture are unachievable or high-risk for implementation in the FPGA. In this case, you must reevaluate and update the system architecture to create a set of high-level requirements that are achievable in the FPGA.

Ask yourself what existing IP can be used and what must be created. You also need to examine I/O requirements and how to map the clock domains and features onto the FPGA clock resources. Other key questions include how to place the GT resources on the FPGA, whether cross-SLR data flow is accounted for in SSI devices and whether target clock frequencies are realistic for the design functionality. Finally, you must also assess whether your target performance goals are realistic for the selected FPGA.

FPGA REQUIREMENTS DEFINITION AND PARTITION

The FPGA requirements definition-and-partition phase is closely related to the system and FPGA architecture and is driven by the decisions made at those stages. The FPGA requirements definition refers to the detailed requirements to be implemented in the FPGA and will serve as the definitive feature list for the design and test-engineering teams to achieve, design and implement. This job differs from the system and FPGA architecture requirements in that the FPGA requirements are precise. The list defines the minute details of the FPGA rather than the division of functionality between the different components of the system, or the data flow through the FPGA.

The purpose of this stage is to clearly define exactly what the FPGA engineering team should implement and test. Here, you will translate the high-level system and FPGA architecture requirements into exact requirements for implementation. The benefits of doing this are twofold. First, discretely defining the FPGA requirements will highlight any limitations of the system and FPGA architecture and any previously unconsidered or unforeseen conditions. Second, this step will pave the way for smooth execution of the FPGA design development and test.

To properly outline FPGA requirements, you must have a clear definition that is concise and easily distilled into discrete requirements. We recommend labeling or numbering the requirements and defining them as a basic description easily determined to be achieved or not, rather than as a high-level, ambiguous requirement. You can use any industry-standard or proprietary format so long as it is clear and concise.

Avoid vague or loosely defined terns such as “fast” or “small.” Instead, stick with specific targets such as “400 MHz” or “4.2k flip-flops.” The goal of this definition is to ensure the document can be distributed to development engineering teams with no prior knowledge of the system or FPGA architecture to implement without the need for constant clarification. Ask yourself whether each requirement is clear, concise and unambiguous and whether it contains all the necessary information to avoid a need for constant clarification. Also, do the requirements include pin-out and I/O definition? Are all high-level requirements decomposed into fundamental design elements? Can a design team not involved with the early system architecture definition use the requirements to develop an FPGA? And finally, can a test and verification team use the document to develop test platforms and tests to validate each requirement with a definitive pass or fail?

FPGA DESIGN PLANNING

This stage of the framework is planning for the actual development of the FPGA hardware and to ensure features and development are completed in step with other parts of the overall product development.

The goal at this point is to properly map the current state of the system-level and FPGA-level requirements and architecture into a development plan. After going through the previously listed planning stages, there are typically two scenarios development teams will now encounter.

Scenario 1 is a case where the system and FPGA architecture and requirements are well understood and are finely detailed, resulting in an FPGA design development phase (that is, HDL coding) and test development phase (simulation testbench) that will proceed smoothly with minimal requirements changes.

Scenario 2 is a case where the system architecture and FGPA requirements are still in flux. Designs in this category will encounter multiple changes and revisions during the design and test development phases of the cycle.

While everyone shoots for Scenario 1, more often than not, people will find themselves in Scenario 2—clearly, a more difficult situation to manage.

The overall goal of design planning should be to reach Scenario 1 at this stage of the development cycle. In Scenario 1, the development of the FPGA is straightforward and is a matter of scheduling the implementation and test of design features.

In Scenario 2, the most important management task is to ensure you have a well-understood process in place to evaluate and determine what changes should be implemented, and the impact of each change to the overall development schedule. There are several program-management philosophies and techniques that you can employ here. The most important point is that such evaluation and impact assessments are made.

In terms of FPGA-specific planning and development, one of the benefits of FPGAs is the ability to revise and download a hardware platform to a prototype PCB multiple times. Design teams should take full advantage of this capability. Therefore, the recommended development plan is one that will incrementally add features to a working design. The idea is to start with a basic design that can enable the major communication interfaces to function without all of the requirements implemented.

The benefits of such an approach are twofold. First, you will ensure there is always a working design that you can use to debug a PCB and the larger system. And second, debugging the actual FPGA design will be easier, since you can check newly added features to ensure they do not interfere or break the currently operational design.

In parallel with the FPGA design development, it is vitally important to have a sound simulation environment plan for the resulting FPGA design. Investing in the development of a robust simulation environment will not only result in a reduction of design bugs, it can also significantly reduce lab debug time by allowing you to replicate real-life data flow so as to reproduce error conditions in simulation and quickly isolate and determine the root causes.

Development of a robust test and simulation environment is just as complex as development of the FPGA design itself and should be planned for and given consideration as such.

The Xilinx Global Communications Services Group has honed our FPGA development framework through consistent application over hundreds of FPGA designs. The result is a practical design approach that delivers results, is easily understood and is widely applicable to many different development tasks. The benefits of using this framework in developing your next FPGA design will be accurate overall development schedules, fast hardware bring-up and, ultimately, an on-time product launch.