White Paper

Choosing the ideal FPGA prototype for ASIC and SoC design.

Frédéric Leens – Exostiv Labs

November 2020

I. Introduction

A. FPGA Prototyping in 2020

In the 2020 edition of the Wilson Research Group Verification Survey, Mentor Graphics, a Siemens Business, shows that **at least 30% of all respondents designing ASIC or SoC declare using FPGA prototyping**, **no matter the size of the chip being designed.** Among the respondents, about 80% map their ASIC onto less than 4 FPGAs with the remaining 20% using more than 5 – and sometimes more than 40 FPGA chips in the largest cases! More than ever, FPGA prototyping is an important part of the ASIC and SoC design.

B. Why prototype?

Prototyping ASIC or SoC primarily requires gates. Today, leading FPGA vendors show that the biggest FPGAs reach more than **40 billion transistors!** So, 'gates' are *really* available.

How - and for what purpose - should these gates be used in a prototyping approach - are essential questions.

Over the last decade, EDA companies have devoted a lot of efforts to implement a 'shift left' strategy. 'Shift left' is roughly equivalent to an attempt to push most the validation and verification efforts early in the overall design flow. We know that 'prototyping' – and globally, any step in the ASIC design flow that involves using *hardware* – is more on the 'right' of it.

Consequently, the design flow has become heavier in terms of number of potentially usable design software and techniques – and this, thanks to an effort to add software tools early on for verification and validation.

It means concretely that some of the tasks previously performed with a hardware prototype are now potentially covered with a software technique earlier in the flow – and hence the value for using FPGA prototype must be re-evaluated as new tools are available.

A prototype can do three essential things:

1) Run at – or near – target operating speed.
2) Run in realistic environment and encounter random and unforeseen events.
3) Run during extended times like days, weeks and even more.

These are the 'unique and desirable' capabilities of FPGA prototypes. In other words, **this is what FPGA prototypes can do that 'left shifted' solutions and methodologies cannot do** – or at least not at a reasonable cost.

They translate into the main usages of FPGA prototypes:

- Design software for a SoC platform when the final hardware is not yet available.
- 'Debugging' and overcome the limitations of simulation-based techniques. Most of these limitations are related to long execution times and the difficulty to accurately reproduce the target environment.
- Regression and performance testing.

II. Choosing a FPGA prototyping platform

A. The key dimensions of a FPGA prototype.

Many questions arise when choosing a prototyping platform:

- How many gates are required?
- Is modularity a requirement?
- Will the system be used on multiple designs?
- Which software environment and tools are necessary with the prototype?
- Which environment and peripherals should be built into the prototype?
- …

We can use many dimensions to classify prototyping systems. Two of them are essential: **similarity to the target** and **visibility**.

B. What is 'similarity to the target'?

The 'similarity to the target' is a dimension that measures how much the FPGA prototype can be considered as a faithful model of the future chip. Although there is not one single way to measure it, we can say that a prototype has a 'perfect' similarity to the target chip if you can swap one for the other and see no difference.

There is at least one reason why a 'perfect similarity' cannot be achieved with a FPGA prototype, and that is the mechanical size of the prototype versus this of the final chip. Hence, for ASIC prototyped with FPGA, we should first consider 'functional equivalence' as a key indicator for a good similarity. To reach a good 'score' in 'similarity to target', we think that an ideal prototype should:

- Provide the target device functionalities,
- Be able to run in the future target environment, and:
- Run at the target operating speed.

We explore them below.

1. Reproducing the target device functionalities

Being able to reproduce the target device functionalities requires that the assembled 'FPGA fabrics' are able to behave like the silicon being designed. FPGAs are usually chosen as a prototyping technology for ASICs because of their silicon technologies similarities. However, technology mapping and especially partitioning are still challenging.

'Partitioning' consists in mapping ASIC gates onto a series of smaller FPGA chips. It involves cutting the logic into several pieces and adding interfaces at the boundaries of the FPGA devices. Partitioning can be more an art than a fully automated process. It deeply affects the similarity to target because it adds interfaces (serializers / deserializers, multiplexing, …) in the logic to manage the chip-to-chip connections. Consequently, the timings and the data flow of the initial logic are modified. It notably almost systematically results in having to reduce the system clock speed.

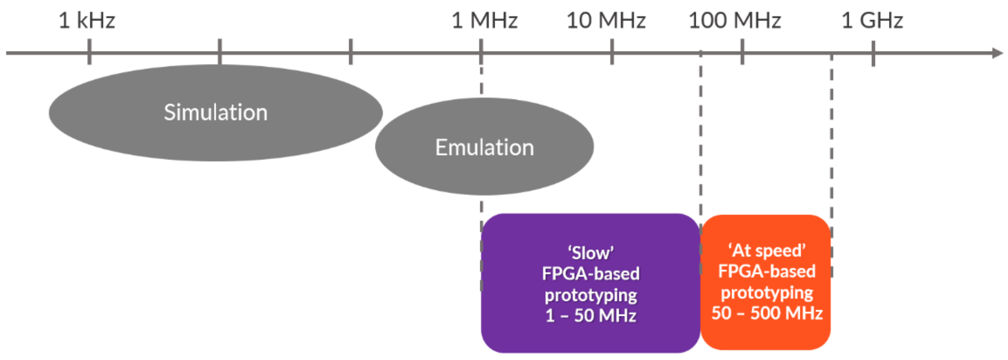

Figure 1: Relative speed of some design techniques

Due to the complexity of the partitioning process, it is desirable to minimize the number of FPGA chips required to map the ASIC when possible. For this reason, it is usually recommended to use the largest and fastest FPGA technologies.

2. Running in the target environment.

Being placed in the target environment should be one of the main goals of building or buying a FPGA prototype. This is one of the key qualities of prototypes, as it allows overcome modeling mistakes and test the design against real world inputs, with all its flaws and randomness.

It means that the FPGA prototype should not restrict to providing chips and gates for ASIC mapping. The prototype should therefore provide the adequate connectivity with the external world, in such a way that it is able to function in the future ASIC environment. Similarly, the prototype itself should feature the right set of system peripherals – DRAMS, flash, communication interfaces, processors, … With this environment, the prototype can execute its tasks as if it were the final ASIC, including running software in the case of SoCs.

3. Running at operating speed.

There are two main reasons that prevent a FPGA prototype from running at the target operating speed.

The first reason is the FPGA technology. Over the last decades, FPGA have become one of the most advanced chip technologies available, with operating speeds reaching 600 MHz range – and even more. That is remarkable, but the configurable structure of FPGA logic and routing potentially makes them systematically slower than the most advanced ASIC technologies. Running FPGA gates at Gigahertz is still currently nowhere in sight.

The second reason for lower operating speed is the need for partitioning, as the FPGA-to-FPGA interfaces of prototypes create data flow bottlenecks that force slowing down the whole system.

Despite the two above reasons, having a prototype running as fast as possible 'close' to the target operating speed is a tremendous advantage: it allows letting the prototype peripheral interfaces run at their speed of operation. For instance, if the future ASIC has an Ethernet interface, it is desirable to be able to connect the FPGA prototype to a standard network and exchange traffic with it. This is something that FPGAs are perfectly capable of. FPGAs are able to manage about all the usual types of peripherals at speed of operation.

So, in this context, 'at speed of operation' must be understood as 'at peripheral speed of operation', even if the logic gates of the FPGA run at a lower speed than this of the future ASIC. (see figure).

C. What is 'visibility?

Simply put, **'Visibility' is a measure of the ability to look inside the system in operation.**

Getting visibility is especially important for a prototype: it helps verify that it functions as expected. When it does not, it allows investigating what is wrong with the prototype by providing a direct access to its detailed internal behavior.

Visibility goes beyond what can be observed from the system interfaces, I/Os and software – even if these elements can be mobilized for visibility. The programmable nature of FPGAs eases the development of prototype visibility. Schematically, FPGAs are both the ASIC prototype and the resources to observe itself with a high degree of precision, down to bit-level with clock cycle accuracy.

For digital ASICs and SoCs, getting visibility involves recording the history of internal logic nodes switching over time – roughly encompassed in the term **'trace'**. Visibility can be measured by the **'reach over the system'** and the **'depth of the trace'**.

'Reach' measures what percentage of the prototyping system can be traced. Can you really record a trace of any of the bits composing the logic system and how easy is it?

'Depth' measures the trace 'length' over time. How many (micro-, milli-, -) seconds, minutes, hours of system trace can you record?

D. The usual FPGA prototyping options

At the risk of simplification, we think that there are three main types of FPGA prototyping platforms:

1) **'Standard FPGA boards':** these are commercial boards with usually one single FPGA chip (max. 2) surrounded by common peripherals such as DRAM, flash, PCIe and network interfaces and some

extension ports – usually FMC or FMC+. The FPGA vendors provide such boards as development kits. They usually do not come with any specific tool or software environment.

2) **Prototyping systems** (or 'Big EDA prototyping systems')**:** these systems are made by companies supplying in EDA tools for ASIC and SoC. Such systems are usually modular and composed of multiple interconnected FPGA boards. Each board typically contains 1 to 4 FPGAs placed as an array with interconnections and extension connectors used to add peripherals and interfaces with the outside world. They target ASIC and SoC prototyping and are ready for partitioning. They provide specialized resources like tunable clock circuitry used to synchronize the multi-FPGA array. They often come with partitioning and debug tools, as well as with software environments used to set up the prototype. These systems and environments are usually much more expensive than simple standard FPGA boards.

E. A 'Visibility-Similarity' chart

Figure 2 below shows how the usual FPGA prototyping options rank in terms of visibility and similarity to target. This is a 'radical view' – there can be nuances between these categories, of course.

Systems that run at target speed in the target system environment with its complete set of features and functionalities score high for the 'similarity to target'. As fully customizable systems, custom FPGA boards are naturally better positioned for this dimension, as they allow choosing the fastest and most complex FPGA technology to reach system speed on a minimal number of chips. Custom prototyping boards also provide total freedom of choice for the interfaces and peripherals. A prototype based on custom FPGA boards will potentially always be closer to the target ASIC than any generic commercial product.

On the vertical axis, prototyping systems, which usually come with a rich set of tools and designed for ASIC / SoC prototyping – which we call 'Big EDA' prototyping systems - have the best

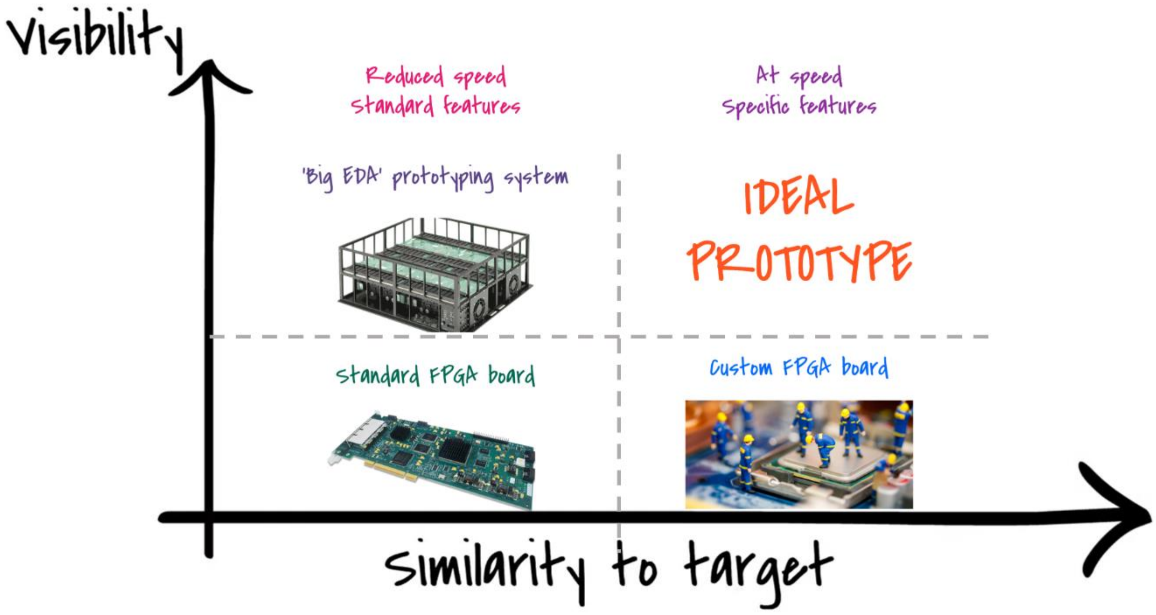
Visibility
Reduced speed standard features
At speed Specific features
'Big EDA' prototyping system
IDEAL PROTOTYPE
standard FPGA board
Custom FPGA board
similarity to target

Figure 2: Visibility - Similarity chart

3) **Custom FPGA boards:** as indicated in the name, these are full custom boards, designed for one or multiple purposes. Making a custom board provides the maximal flexibility for choosing the FPGA technology, the number of chips, the peripherals, the interfaces, and other extension features. Naturally, they tend to be more accurately closer to the target ASIC than their 'generic' commercial counterparts, designed for a larger market. Except for a software environment, and simple debug interfaces, they are usually not designed with specific EDA tools – due to the work effort it represents and the difficulty to efficiently maintain them.

visibility capabilities. Such capabilities can be built into the system or provided with an external their software tool environment. Their generic structure rarely allows them to run above 50 MHz at best – 5 to 10 MHz being already considered as a good performance.

Due to the high investment they require, they tend to be used during many years before being upgraded or replaced. The installed base at some companies can be outdated in terms of FPGA technology. For this reason, they score lower than custom boards for similarity because they potentially use a larger number of smaller and slower FPGA chips. The latest FPGA technology always has benefits.

Finally, due to their generic nature, standard FPGA boards score lower than custom boards for similarity to target. As they come with no specific tool, they also score low on visibility.

F. What is the best prototyping platform?

There is of course no single answer to that.

The complexity of your target ASIC radically defines your choices. In the **2020 Wilson Research Group Verification Survey**, Mentor Graphics, a Siemens Business shows that close to 40% of the respondents prototype ASIC on a single FPGA chip. Since they avoid partitioning, this category of users is able to run at close to system speed and might use small commercial boards or make their own boards.

At the other end of the spectrum, about 5% of the respondents need 20 to more than 40 FPGA chips to map their ASIC. In this case, a commercial EDA prototyping system seems to be desirable to benefit from the partitioning and built-in visibility tools. However, beyond 40-50 FPGA chips, a custom system might be the only option as this size comes close to the maximal gate capacities of current commercial systems.

The effort of building a custom system really increases with the number of FPGAs, which makes it a hardly viable solution for large designs, except when there is no other choice.

The figure below shows the numbers reported by the survey and displays our attempt to provide the ideal prototyping platform according to the prototype complexity in terms of number of FPGAs.

Choosing the best prototyping platform results of combining the constraints and characteristics of your design and – inevitably – the economics of it. While a Prototyping platform can seem out of reach for a prototype that requires no more than 2 or 3 FPGAs, the effort of building a board system for 20 FPGAs can be quite daunting. In addition, the cost of a modular system can be spread over multiple ASIC designs, while a full custom board might be difficult to re-use, even if it is the best fit in terms of performance for a specific project. Evidently, the choice of the platform is not just a matter of technology…

III. Towards an ideal FPGA prototype

An ideal prototype should score high for both the visibility and the similarity to target (Figure 2). In this section, we voluntary focus on technology only, not the cost of the solution(s).

A. Maximizing similarity to target

To reach maximal similarity to target, we should:

- **Customize standard boards and prototyping systems** so their features, interfaces and peripherals exactly match these of the target chip.
- **Make sure that the systems can run at speed of operation** in a realistic environment. This means that we do not have to artificially slow down the boards environment to adapt to a slow FPGA system speed.

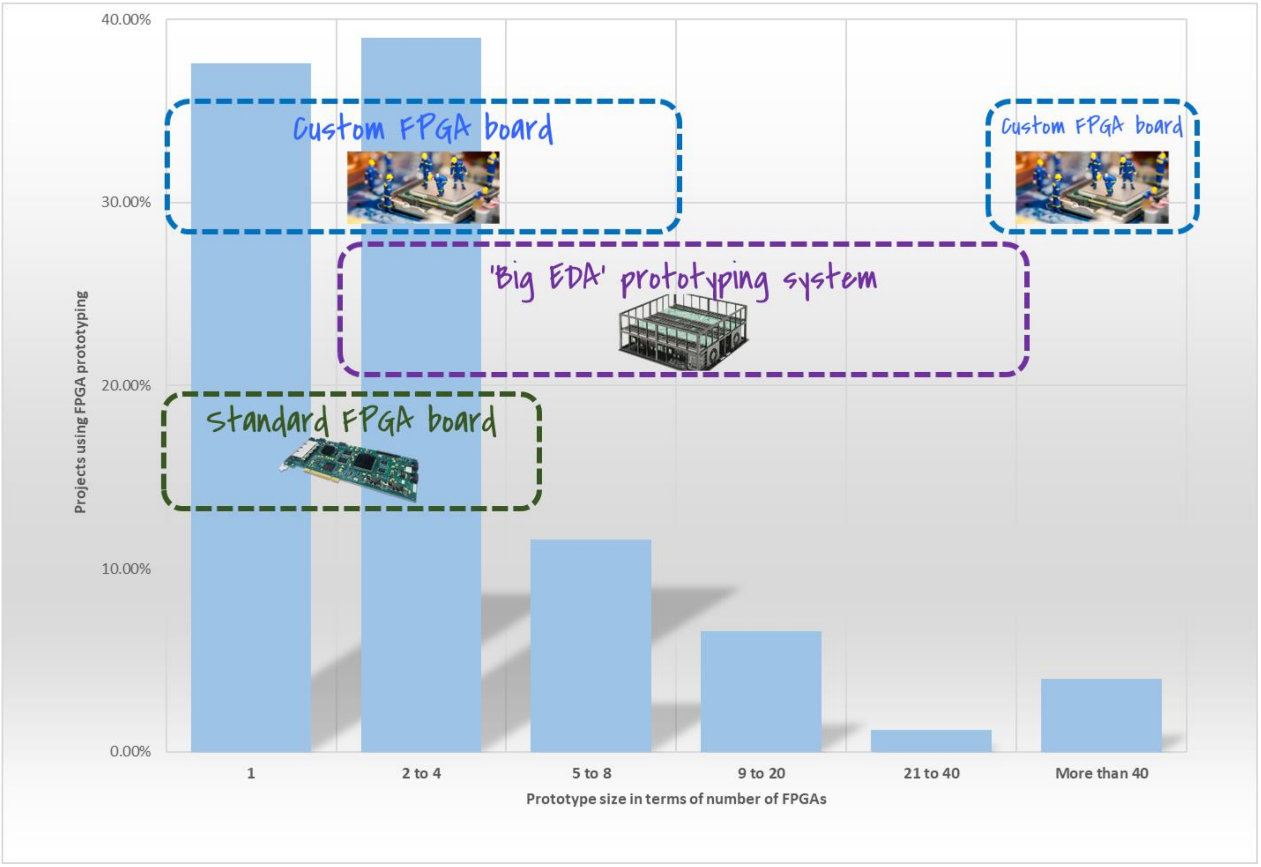

Figure 3 : Size of FPGA prototyping project (source: 2020 Wilson Research Group Verification Survey, Mentor Graphics, a Siemens Business) with typical prototyping platform usage.

Please note: running at speed of operation sometimes makes prototyping systems' tools unusable, since they frequently have a performance limitation that is stricter than the capabilities of the boards. For instance, they use a memory interface that has a bandwidth limitation that impacts the rate at which they are able to sample data in the FPGA, not allowing running the FPGAs above this frequency. Consequently, at higher speed, the benefit of these tools is lost.

Customization is the key to improve the similarity to target. It requires system and board design skills or outsourcing/buying the right system building pieces.

B. Maximizing visibility

Getting visibility into a FPGA prototyping requires having access to the internal nodes of the system and having the tools (software / hardware) to capture and analyze the data.

Standard and custom FPGA boards score low on visibility because only limited standard tools are available for them. They are generally limited to JTAG embedded logic analyzer such as Xilinx ILA and Intel Signal Tap - or even traditional instrumentation like logic analyzers and oscilloscope.

FPGA vendors' embedded logic analyzers have a good reach over FPGA but an extremely limited depth. They usually store traces in memories in the FPGAs, not bigger than a few kilobytes – much too limited to get a real insight into multi-million gates prototypes. Conversely, traditional instrumentation can only connect a few hundreds of I/O external FPGA I/Os, which is a too limited reach over the internal nodes and proves impossible to use when there is more than one FPGA in the system.

Prototyping systems are delivered with an integrated infrastructure and tools that provide a (very) good visibility into the system. These proprietary tools depend on the prototyping system structure and are generally not available for other FPGA boards. In a recent trend, large EDA vendors do not allow using their tools with 3rd party board vendors, locking up the users into using both their boards and their tools.

Improving the visibility into standard and custom FPGA boards requires a technology-independent tool. This tool would provide the required infrastructure to reach the prototype internal nodes and the functionalities to capture (very) deep traces. This technology-agnostic tool will have a lot of value for prototyping systems if it overcomes the limitations of the proprietary tools and provides even more visibility at speed of operation.

IV. Conclusions

In 2020, FPGA prototyping keeps on being an essential part of the ASIC and SoC design flow. Even if new validation techniques have been added to the early stages of the flow, FPGA prototyping keeps a unique and desirable capability: *running at speed in realistic environments during extended times.*

With the advent of extremely complex FPGA technologies, it seems that there has never been a better time to use FPGA prototyping. Logic gates and speedy components are available to build full custom FPGA prototypes that faithfully model the future ASIC or SoC.

One essential – and sometimes missing – piece for a successful prototyping approach is the visibility infrastructure.

Standard and custom FPGA boards display only poor visibility, as virtually no tracing tool is available – save maybe for extremely limited analyzers from the FPGA vendors. Such tools are not well fitted to provide the reach and trace depth required by complex multi-FPGA systems.

Prototyping systems supplied by large EDA companies usually include advanced visibility tools and particularly good visibility into the system. However, these tools often suffer from performance limitations, potentially preventing the prototype from running at speed. Such tools, however advanced, are not made compatible with custom boards, thereby limiting the ability to choose the best FPGA prototyping technology freely.

Therefore, a technology-independent tool that provides unprecedented visibility at speed into any system is the key to choosing the ideal FPGA prototype, no matter the ASIC or SoC complexity.