# Chapter 1
# Introduction

## 1.1 FPGAs in VLSI design and verification flow

Over the past few decades, digital VLSI design has played a very crucial role to realize a revolution in the fields of information technology, communication, infotainment, etc. The gadgets catering to these markets are constantly getting complex and therefore putting a greater challenge to VLSI design teams as these designs are at the heart of such systems. As an example, a mobile phone today performs functions of a telecommunication device, global positioning system, internet hotspot, messaging device, camera, display screen, radio, music player, and much more. Another example is the automotive industry wherein electronics have become a major component of the cost of a vehicle from approximately 1% in the 1950s to approximately 30%–40% today [1.1]. All of this has been possible with advancements in the VLSI design flows, smaller gate sizes and increased number of gates as governed by Moore's law. The increase in complexity for these designs is also complemented by rapid changes in features or enhancements which are putting a huge pressure on design teams to deliver functionally verified silicon chips. Moreover, the changes are typically requested by customers in a short span of time. As an example, a user typically upgrades his mobile handset once in one or two years nowadays compared to long usage of landline phones in the past. The challenge posed to the VLSI design and verification community to deliver system on chips in the shortest possible time was addressed by the FPGA industry in the past few decades. FPGAs evolved from programmable logic devices over the last 2-3 decades. FPGAs provide a lot of advantages to the VLSI engineers. Some of them are as mentioned below:

- Due to their symmetric (arrays of logic) and parallel architectures, FPGAs inherently support parallelism in hardware which means that if applications running on them support parallel processing, one can leverage the hardware in an optimal manner.
- With the increasing complexity of SoCs and microprocessor architectures, more and more embedded software and firmware elements are getting involved in the design process. So, FPGAs play a great role in helping to define efficient boundaries between hardware-software partitioning as well as co-verification which is not possible with conventional hardware description language (HDL) simulation platforms.

- Real world customer use-cases can be run in a pre-silicon environment with actual peripheral connectivity such as USB, PCIe, Ethernet, and external debuggers (e.g., JTAG), which are used as HDL models in simulations.

Due to the above mentioned points, FPGA systems are used in design and verification sign-offs for almost all application SoCs. This also means that for prototyping different application designs on FPGAs, designers need to ensure they are utilized in the best possible manner, i.e., minimal wall clock runtime, minimum resource utilization and minimum power dissipation. Speed, power, and area-optimal implementation of designs on FPGAs saves cost as well as time. Conventionally, hardware description languages like Verilog and Very high speed IC HDL (VHDL) have been used by designers to describe the behavior of the design (also known as register transfer language (RTL) code). FPGA synthesis tools like ISE and Vivado have been used to translate RTL code into logic gates and hence mapping them to FPGA technology primitives such as look-up tables and flip-flops [1.2, 1.3]. The verbose nature of these HDL languages and tight coupling with technology cells requires a deep knowledge of syntax and synthesis flows for designers and verification engineers. Moreover, HDLs are more prone to error and require a large number of design decisions to be taken by designers keeping synthesis targets in mind. High-level synthesis (HLS) tools started to get popular in the last 2-3 decades to boost the abstraction level of the FPGA design process [1.4, 1.5, 1.6, 1.7]. Figure 1.1 shows the levels of abstraction offered by different design methods for digital design. Section 1.2 discusses HLS in detail.

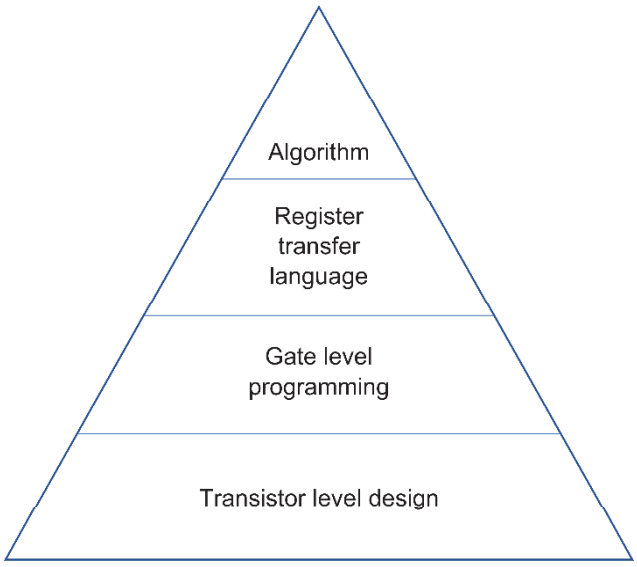


**Figure 1.1.** Abstraction levels in VLSI design flow.

## 1.2 High-level synthesis

HLS, also identified as behavioral synthesis, is an automated design process that follows an algorithmic specification of the required behavior and produces its digital hardware implementation. In HLS, synthesis starts with a design's high-level specification, such that the behavior is, in general, decoupled from clock-level timing. The behavioral code is analyzed and an RTL code is produced under architectural constraints. In turn and by employing a logic synthesis tool, it is usually synthesized to produce a gate level netlist.

This flow has gained considerable attention from the design community in the past 2–3 decades. It is getting the design community's recognition as a methodology for guaranteeing continued verification throughout the design cycle and enabling designers to express the design behavior at higher abstraction levels. HLS tools, including Vivado HLS, MATLAB HDL coder, and several other tools available in the open-source community, are typically employed by digital designers and architects [1.8, 1.9]. Such tools share designing and deploying algorithms that target diverse applications in aerospace, communications, image processing, deep learning, neural networks, among other fields. Furthermore, HLS tools reduce the code complexity by about 7–10 times, providing behavioral intellectual property (IP) reuse across projects and facilitating verification teams to apply high abstraction level modeling techniques, such as transaction-level modeling [1.10].

Additionally, a preponderant set of current system on chips have embedded processors. Therefore, they offer more software elements in the design process. That is because of the conjunction of microprocessors, digital signal processors (DSPs), memories and customized logic on a single chip. Consequently, an automated HLS process enables designers and architects to experiment with distinct hardware-software boundaries. The HLS process also allows exploring diverse area, power and performance tradeoffs from a single standard functional specification.

Accordingly, the industrial deployment of HLS tools has become more practical with advancements in RTL synthesis tools due to the broad adoption of RTL-based design flows since 1990s. Proprietary tools, in essence, were developed by important semiconductor design houses such as IBM, Motorola, Philips and Siemens [1.11, 1.12, 1.13, 1.14]. Major Electronic Design Automation tool vendors also introduced the commercialization of various HLS tools. In 1995, for example, Synopsys introduced the "Behavioral Compiler" tool [1.15]. It produces RTL implementations from behavioral HDL code and connects to downstream tools. Comparable tools include Mentor Graphics' "Catapult HLS" and Cadence's "Stratus high-level synthesis" [1.16, 1.17]. A typical flow for HLS in VLSI designs is presented in Figure 1.2. Provided a standard functional specification, one can therefore iterate between different possible

implementations, and depending on the requirements, employing HLS directives or architectural redesign. As Figure 1.2 shows, HLS flows allow designers to iterate between several implementation options to satisfy the area, speed, and power budget specified by architectural specifications. HLS flows also allow continued verification of the design during the development cycle (algorithm, RTL, and implementation) [1.9].

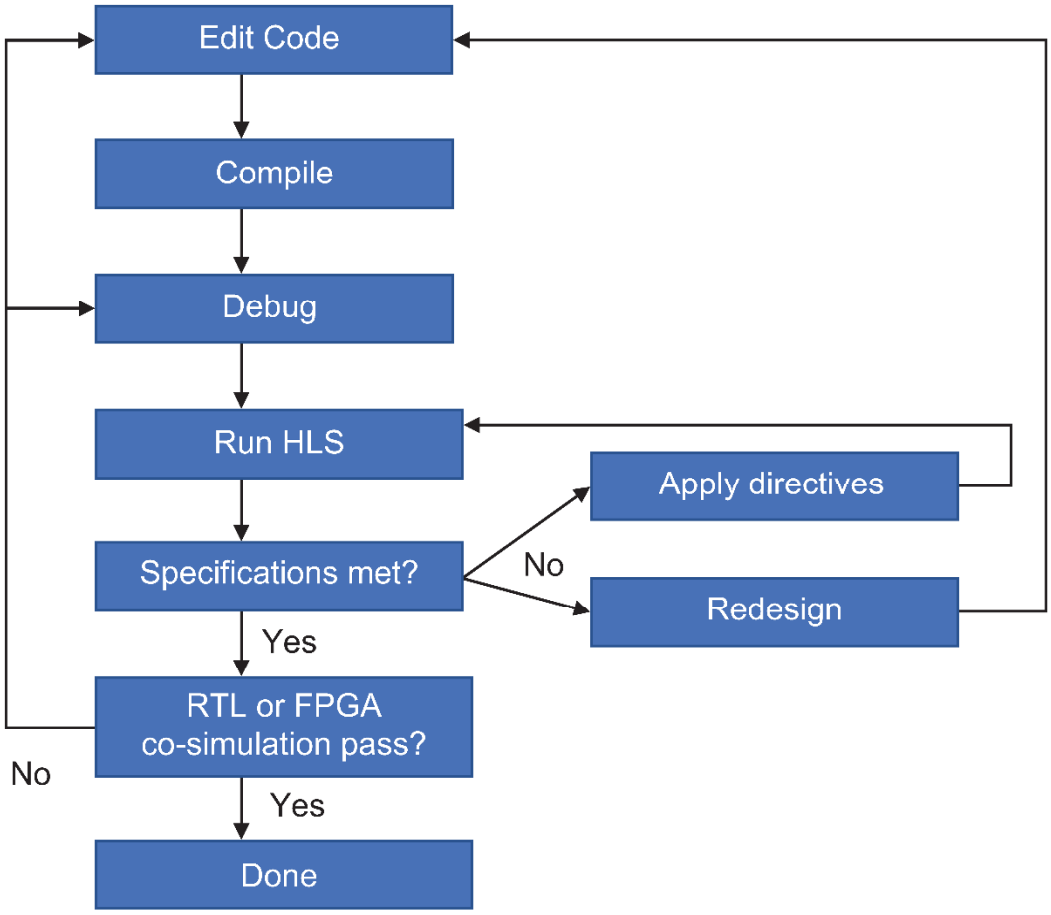


**Figure 1.2.** High-level synthesis flow of VLSI designs.

## 1.3 Motivation behind the work

Due to the advantages of the high-level synthesis we described in Section 1.2, multiple organizations are moving to HLS flows to stay competitive in the market so that last-minute changes in the architecture can be implemented and design cycle time can be reduced. Further, it is clear from the first two sections that achieving optimal FPGA implementations of RTL designs is the need of the hour for almost all applications in digital VLSI design.

There has been considerable research undertaken to improve the area, speed, and power dissipation of VLSI designs on FPGA platforms. But most of these optimization techniques apply to hand-coded RTL synthesis flows and do not apply to HLS flows. Some researchers in the recent past have also proposed optimization methods for HLS-based design flows, but they are

very much specific to the application or the FPGA synthesis platform being used, hence not generic.

Through this research, we wish to propose a generic methodology which can work across the tools and cater to most application designs. This work also aims to propose a set of guidelines which can be implemented to optimize different category designs (catering to different applications such as signal processing, image processing and mathematical algorithms and processor designs). Unlike other works, we also tested the proposed methodologies on an evaluation matrix with different designs and two HLS tools (MATLAB HDL coder or Vivado HLS due to their wide acceptance in research community) and compared results against hand-coded RTLs for the same set of designs.

To summarize, the following rationale was used to select the topic:

1) HLS helps cut the design cycle time by reducing code complexity and allowing behavioral IP reuse. Moreover, HLS allows continued verification throughout the design cycle. Hence, optimized implementation is important for HLS flows.
2) FPGAs are playing an increasingly important role in VLSI design flow for almost all application designs. Hence, having optimal FPGA implementations is the need of the hour for almost all application designs.
3) Platform and tool independent HLS optimization methods for design applications are a definite research gap. The methodologies which exist in literature are not independent of application and FPGA synthesis tool used.

## 1.4 Objectives of this work

The work's objectives have been laid out after the literature survey and identification of research gaps reported within Chapter 2. The principal objectives of this thesis are as follows:

1) To investigate design and verification of multiple application designs, namely signal processing, image processing, computation etc. using HLS tools.
2) To explore the area, speed, and power optimization in designs using HLS directives offered by MATLAB HDL coder and Vivado HLS.
3) Propose a generic methodology for optimization of HLS designs which is platform and application independent for FPGA prototyping.
4) Using HLS to design, verify and optimize (using a combination of generic methodology and directives) a test application and compare FPGA functional simulation results with post-silicon results obtained from a taped-out SoC model.

## 1.5 Organization of this thesis

The thesis is composed of six chapters. Chapter 1 introduces the research topic, motivation behind the work along with the exact objectives. The rest of the chapters are organized as follows: Chapter 2 presents a literature review about the HLS optimization methods already available. This chapter also covers some of the existing HLS directives offered by different tool vendors like MATLAB HDL coder and Vivado HLS.

Chapter 3 discusses the usage of HLS directives (the most common method of optimizing HLS designs) on multiple application designs. The implementation results of four different applications, namely, quadrature phase shift keying (QPSK) modulator, DSP filter, microprocessors without interlocked stages (MIPS) processor core and AES algorithm are discussed. The results attained by the usage of HLS directives for these applications are presented. Hence, this chapter presents the usage of methodologies that are already available in the literature or already offered by HLS tools.

Chapter 4 presents a novel HLS methodology based on application-specific bit widths for intermediate data nodes. This is independent in terms of platform, tool, and application. To prove the efficacy of this methodology, this chapter also presents the results obtained using this methodology on five different application designs, namely, bandpass DSP filter (signal processing), Sobel edge, Harris corner detector algorithm (image processing), digital down converter for software-defined radio applications (communications) and AES for automotive applications. A comparison is presented between the synthesis results obtained and the synthesis results achieved from hand-coded RTL implementations for the same designs presented in the literature.

In Chapter 5, a real industrial application-RADAR signal processor for automotive applications is designed based on the learnings from Chapter 3 and 4. This chapter covers all the aspects of design, verification, and synthesis optimization of the application for target FPGA. The functional simulation results obtained through verification of the generated RTL code against the results obtained from a real silicon taped-out test chip are also presented. It is shown that the achieved results were almost identical hence proving the efficacy of the novel design and verification method. Hence, this chapter proves the efficacy of HLS optimization techniques for targeting FPGA hardware without observing any adverse effects on the design functionality.

Chapter 6 presents a summary of this work's research contributions and also presents some of the perspectives of possible future work in the same area. Finally, Chapter 6 also presents a conclusion for this thesis.

## 1.6 Concluding remarks

This chapter discussed the importance of FPGAs in the VLSI design flow. In this chapter, High-level synthesis flows used in VLSI design flow to target different design implementations on FPGAs was discussed. The growing popularity of HLS based design flows has motivated us to carry out research in this area. This chapter also listed the objectives of the work done along with a brief description of different chapters in this thesis.