Low Cost, High Quality Core-based Design for Scientific Computation

Ian G. Harris, Wayne Burleson, Maciej Ciesielski, Israel Koren and Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
Amherst, MA 01003

1 Introduction

The profound impact that computer technology has had on society can be attributed in large part to the use of computers to enable research in diverse scientific fields. Many fields of research now rely on the computational capacity which can only be provided by special-purpose VLSI systems. Scientific and engineering computational tasks consistently press the performance limits of VLSI technology. To allow society to fully benefit from advances in VLSI technology, that technology must be made available to all branches of the scientific community. A significant limitation to the profusion of VLSI technology to scientific research is financial cost. Although the price of the manufacture of VLSI components went down with time, the cost to design and verify these devices has been increasing rapidly. The constant need for more advanced features and abilities increases the complexity of the design and verification process. High design costs are compensated by high volume manufacture, but low volume VLSI components can be financially infeasible. The special-purpose components needed for many scientific and engineering applications are often low volume, and their price becomes a barrier to the research and development process.

To accommodate increased design complexity, design reuse is quickly becoming the dominant approach for efficiently managing complexity. Although the concept of design reuse has been applied in the past, the complexity of reusable components has risen from gate-level (standard cells) to application-specific blocks (filters, controllers) and general-purpose processors (RISC, CISC, DSP). Clear evidence of the move towards design with reusable cores can be seen in the number of core designs currently on the market, and the magnitude of several core standardization efforts initiated by several industrial confederations including the Virtual Socket Interface Alliance. Use of a core-based system design methodology has the potential to greatly decrease design cost, thereby facilitating the more widespread use of advanced VLSI technology in science and engineering.

We propose to develop a family of core customizer software tools which facilitate user-assisted synthesis of core designs. In order to be suitable for reuse in a range of systems, hardware designs generated must be of near-custom quality (in terms of area, performance, power, and testability). To guarantee optimal design efficiency, the resulting hardware must have functionality which precisely matches the static design requirements imposed by the system. The core customizer will use domain-specific synthesis tool-boxes and template libraries to target a specific domain whose design options are well understood. The resulting hardware design will be what is termed a hard core which is implemented as either a partial or complete layout. We will develop core customizer tools targeted at both ASIC and FPGA technologies. By producing a hard core, as opposed to firm or soft, we guarantee more accurate and reliable design characterization, and also minimize the need for extra
design work on the part of the system designer using the core. The implementation of a hard core also enables more accurate and reliable parameter characterization.

2 Core-Based System Design

We propose to develop a core synthesis approach which will be integrated into a traditional system chip design flow. The currently accepted system chip design flow is composed of three parts as depicted in Figure 1(a), core design, user-defined logic (UDL) design, and system integration. User-defined logic is designed separately for each system. Core design occurs only once per core, which is assumed to be infrequent because a core is reused in many systems. To design a reusable core, its behavior must be generic, allowing it to be used in many systems, but specializing it to no single system. As a result the core behavior may not perfectly match the system requirements and additional logic will be required to integrate the core into the system. Both cores and UDL are designed either by hand, or with the assistance of standard synthesis tools. We propose the system chip design flow shown in Figure 1(b) which uses special-purpose core customizers (referred to as CCi in the figure) to define each core. In our design flow, each core is customized to the system in which it is used using the core customizer tools. The process of customizing a core must be performed for each system, but the customization process is much faster than the original core design process. The speed of the customization process arises because each customizer exploits the design space parameterization which is unique to each core. By performing high-speed customization, a core is generated to match the needs of the specific system, while still achieving the low design cost expected when pre-designed cores are used.

![Traditional System Design Flow vs. Core Customizer (CC) Flow](image)

**Figure 1:** System Chip Design Flow, (a) traditional, (b) proposed

We will develop a family of core customizer software tools which generate hardware designs that are suitable for use as cores. In order to be suitable for core use, hardware designs generated must be of near-custom quality (in terms of area, performance, power, and testability). To guarantee optimal design efficiency, the resulting hardware must have functionality which precisely matches
the requirements imposed by the user of the core. These high standards for the quality of the resulting designs make the use of standard function generation tools infeasible. A software tool is required which explores the design space as an expert human designer would do. In order to make this problem tractable, each core customizer tool will focus on the design of a single application class.

Each core customizer tool integrates synthesis at several levels of abstraction, and therefore involves the thorough exploration of an enormous design space. The sheer size of the design space motivates the application of structure to the solution space to enable efficient design space search. We propose a hierarchical design space framework which will be used to accurately capture the range of possible design decisions. Designer interaction is essential to produce near-custom quality core designs. The proposed design methodology provides support for designer interaction at all synthesis stages through a breakpointing mechanism, as well as several methods for manual synthesis controllability and observability. Synthesis algorithms at all stages will be designed to seamlessly accommodate manual designer direction.

We outline the following software components needed for the development of a core customizer:

- **Design Space Search Assistant** - An intelligent search engine is needed to identify a design solution which satisfies the user-imposed design constraints. The search assistant is composed of a set of synthesis tools at different abstraction levels which mutually interact through a set of interfaces. Synthesis directives specified by the user are used to greatly improve the efficiency of the design process.

- **Design Option Evaluation** - Fast estimation tools with varying precision are needed to quickly evaluate design points and sub-spaces within the entire space of design options. Support for design option evaluation includes novel methods of displaying quality metrics and tradeoffs across the entire design space.

### 2.1 Design Space Framework

A core is uniquely identified by a behavior which it implements at its associated abstraction level. The design space framework for an MPEG encoder core is shown in Figure 2. Several cores are defined at each abstraction level, except at the highest abstraction level where only a single core is described. Each core customizer tool is built for the design of a single core, and will make use of domain-specific information to steer the design process towards high quality solutions. Although the designer is not restricted from making any synthesis decision manually, direction is provided by the tool in the form of a set of proposed architectures, and a set of flexible parameters for each architecture. Each core may be associated with several architectures which have identical functional behavior but significantly different implementations at the lower levels of abstraction. For example, an adder core has several architectures (such as ripple-carry, carry-look-ahead, carry-select, and carry-skim) which all have the same RTL behavior but have significantly different implementations at the logic level. An MPEG encoder is an example of a high level core with several different architectures (such as RISC-based and DSP-based) which are equivalent at the system level. Each architecture is associated with a set of architectural parameters which collectively determine the details of the architectural structure. The architectural parameters describe relatively small refine-
ments in an architecture. Examples of architectural parameters for a ripple-carry adder might be bitwidth and saturation vs. non-saturation addition.

<table>
<thead>
<tr>
<th>Abstraction Levels</th>
<th>Components</th>
<th>Quality Metrics</th>
<th>Architectural Parameters</th>
</tr>
</thead>
</table>
| System             | Component: MPEG Encoder  
|                    | Architecture 1: RISC-based (C-Cube)  
|                    | Architecture 2: DSP-based (TI) | Compressed Data Rate  
|                    | Frame Rate  
|                    | Area | Resolution  
|                    | SNR | |
| Functional         | Component: Motion Estimator  
|                    | Architecture 1: Motion vector-based Linear Array  
|                    | Architecture 2: Source-pixel-based Linear Array | Clocks/source block  
|                    | | Area | Array size/ Pipelining  
|                    | | | Search window size  
|                    | | | Reference block size | |
| RTL                | Component: Adder  
|                    | Architecture 1: Ripple-Carry  
|                    | Architecture 2: Carry Lookahead | Critical Path Delay  
|                    | | Area | Bitwidth  
|                    | | | Saturating | |
| Logic              | Component: AND gate  
|                    | Architecture 1: Low fanout  
|                    | Architecture 2: High fanout | Critical Path Delay  
|                    | | Area | Fanin  
|                    | | | Fanout | |

Figure 2: Design Space Framework for an MPEG Encoder

Each abstraction level is associated with several quality metrics which measure various aspects of the quality of a core design. Common quality metrics include area, delay, power, and test application time. Although several quality metrics are common across many abstraction levels, the specificity of the metric information will differ across abstraction levels.

3 Design Space Search Assistant

To enable the efficient design of customized cores, we will develop core customizer synthesis tools, each targeted at a single application domain. The chief goal of our synthesis tools is to produce near-custom quality designs for a specific application. The interaction of an experienced designer has great potential to quickly direct the synthesis process towards high quality solutions. For this reason, designer interaction is allowed at all stages of synthesis.

3.1 Designer Interaction

Near-custom design quality is ensured by performing integrated design optimizations at all levels of synthesis, from system to circuit, and by relying on designer input at all stages of the synthesis process. The synthesis algorithms must have the flexibility to accept design modifications at any level. Several facilities must be integrated into a core customizer tool in order to allow the designer to interact effectively with the tool.

- **Synthesis Breakpoints** - The designer must have the ability to affect the synthesis process at any point in the process. The designer must identify the important decisions based on the changing state of the design process.

- **Control Mechanisms** - At any point in the synthesis process, the designer must have the ability to direct the process in a meaningful way.
• **Observability Mechanisms** - In order for the designer to make intelligent decisions, information about the state of the design must be made available.

### 3.2 Design Option Evaluation

Each architecture is associated with a set of *estimators* which estimate the quality metric values associated with an instance of an architecture. An estimation is a function of the defining parameters of the associated architecture, as well as the estimations of the subcomponents of the architecture. It is the task of the estimator to compose the estimations associated with the subcomponents in an appropriate way.

### 4 Core Testability

A hard core cannot be modified once designed, so it is essential that any testability enhancements be built in by the core customizer tool. During the core design process, the customizer will generate two deliverables which together encapsulate all testability information associated with a core. One testability deliverable is the hard core *test layout information* which includes layout information for all necessary testability structures and is a subset of the total core layout. The other testability deliverable is a *test plan* which describes the sequence of test patterns which must be applied to the core to perform testing. While both deliverables are needed for testing, the relative importance of the two deliverables is dependent on the testability technique used. For example, the use of a functional test strategy would involve almost no test layout information because functional testing is non-intrusive. By contrast, a built-in self-testing (BIST) approach would be described almost completely by the test layout information since minimal off-chip support is needed.

### 5 Summary

In this proposal a CAD tool suite is outlined that can synthesize flexible VLSI cores for use in computing systems. A key aspect of the proposed work is the flexible user environment in which cores may be customized. The core customizer software will allow the user to explore a largely parameterized core design space to enable low cost design. Unlike synthesis tools and module generators, the proposed tools are intended to be used in a semi-automated manner with tight human interaction. The objective of this core design approach is to support a custom design style that allows expert designers to use subtle design modifications to optimize a parameterized core template at all abstraction levels. By enabling the efficient design and customization of cores, the cost associated with the design of sophisticated VLSI components will be greatly reduced.