Reusable Internal Hardware Templates

Ka÷an Agun and Morris Chang
Illinois Institute of Technology
Computer Science Department
{agunsal | chang}@charlie.iit.edu

Abstract. This paper describes the framework of internal hardware templates. These reusable templates can be instantiated, inside the FPGA, to the required precision. Thus, the resource utilization of the target RCMs can be improved. Moreover, the configuration time can be eliminated after the first use of the template. The detail design is presented.

1. Introduction

After the recent development of Reconfigurable Computing Machines (RCMs), researchers have started to explore efficient algorithms and design methodologies to enhance the resource utilization. One of the software design techniques, design reuse, may help to achieve higher efficiency. Similar to the templates in C++, for example, hardware templates can be instantiated to the precision required by the applications. This way, only the necessary resources are used in the target RCMs. Moreover, this improves the portability among different RCMs. By using a library of templates, hardware system designs are independent from underlying technologies (e.g. FPGA architectures, HDLs). Reusability design technique makes these templates more powerful for system development.

In this paper we will introduce new design approaches to provide reusable hardware designs for RCMs. Currently, hardware templates are instantiated with parameters during the compilation time. In this paper, we propose to build *internal hardware templates* which will be instantiated internally by the hardware itself.

2. Reusability

Reuse, among the topics of software engineering, is one of the approaches to control complexity of software programs. Reusability is simply the re-application of source code and the most effective technique to achieve efficiency by reducing development time. HDLs can get the benefit of the same approach. Increasing the usage of hardware functions among the hardware tasks and among the different RCMs is crucial to the system design of RCMs. This can be done through building a set of parameterized templates. But reuse must be applied systematically to get its potential for improvement of RCMs. Ideally they are embedded in the design principles.

Generic model libraries based on hardware templates have been used in ASIC design. Researchers use application algorithms rather than the hardware design methodology for their hardware templates to increase reuse and reduce development time. Hardware templates are good ways to represent reusable hardware functions through generic and parameterized design methodology. Hardware templates can be classified into two categories. They are:

External Hardware Templates

These types of templates will be instantiated during compilation with parameters. External hardware templates are generic and parameterized hardware functions.

Internal Hardware Templates

The base hardware design will be compiled by HDL tools, but instantiation can be done in hardware independent from the design tools. Therefore it requires hardware support. This approach provides high performance with the cooperation of design tools and the FPGA architecture. The detail design is presented in next section.

3. A Design Alternative: Internal Hardware Templates

Considering a bit-slice design of *n*-bit adder, the smallest component is a one-bit adder which includes 3 inputs: *a*, *b*, *cin* and 2 outputs: *sum* and *cout*. Any size of adder can be instantiated from this design by passing parameter *n*. This adder can be implemented through modern Hardware Design Languages (e.g. VHDL). This design can be mapped into target FPGA with configuration data. The configuration of this adder consists of *n* one-bit adder logic functions, connections among these one-bit adders, and control information that shows how to place and route the configuration in the FPGA.

Configurable Logic Block (CLB) or Logic Element (LE) of FPGA implements these 4 input and 2 output logic functions. The architectures of these logic blocks are mostly based on Look Up Tables (LUT) which are small memories and multiplexers with SRAM. In our example, one CLB based on LUT can perform one-bit adder. Figure 1 shows the structure of the FPGA based on LUT and a logic function.

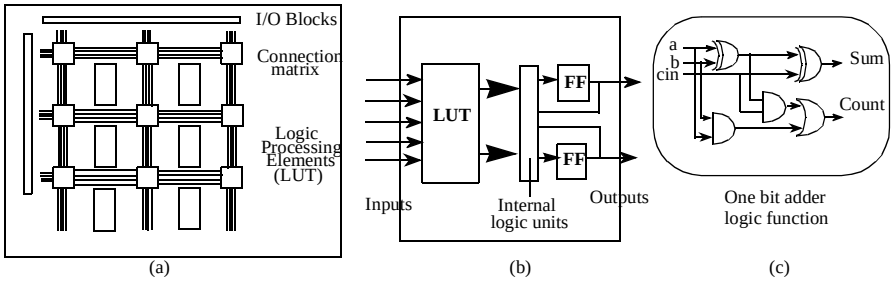

Figure. 1. Basic FPGA structure.

The idea is to divide a design into identical smallest components, so that several of these can form any size of the design. Configuration of the component and the rules of connections among them which is a kind of formula will form an *internal hardware template* to be downloaded into FPGA. The internal hardware template can be stored in the FPGA SRAM cache. Configuration controller will conduct all the operations to instantiate the templates internally in the FPGA. Without considering the technology of FPGAs, rolling of one-bit adder in the FPGA is much faster than downloading *n* bit adder configuration. Because the configuration controller allows to configure all selected cells at once (Figure 2). This design improvement will reduce the configura-

tion overhead as well.

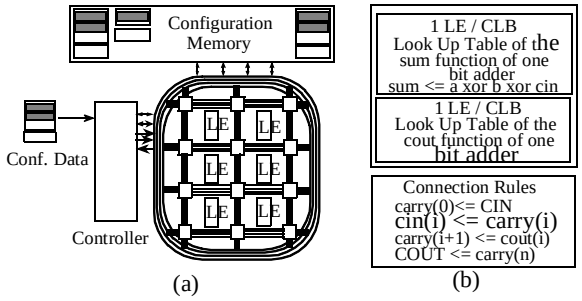

Figure. 2. A FPGA Architecture and Internal Hardware Template Example

Configuration time for a single FPGA can be a significant overhead in RCMs. In some recent FPGA architectures (e.g. XC6200 from Xilinx), it is possible to configure multiple CLBs at the same time. This reduces the configuration time drastically. However, the routing configuration is still required during the download. Moreover, for each different configuration (say 8-bit adder or 10-bit adder), different routing configuration data is required to be downloaded.

We propose to embed routing algorithms into the FPGA. This feature provides exact unrolling of internal templates inside the device as a part of controller. A routing configuration table will provide enough information to the controller to complete hardware design. Symmetrical architecture of the FPGA routing matrixes decreases the complexity of routing configuration.

This design decision minimizes the communication cost by eliminating configuration time. Having the routing algorithms inside the chip provides more portability to design tools. After download, instantiating the templates is completely platform independent. Sending a command with a parameter can instantiate any size of the hardware design.

4. Conclusion

Although there have been several development in the area of reconfigurable computing, the technology of the FPGAs and CPLDs are needed to be improved to provide better solutions for the wide-range applications. Developing hardware technology will reduce the complexity of designs by providing new features for synthesis tools. Internal hardware templates proposed in this paper is one of them. With this design approach, it will be easy to manage FPGA or CPLD devices. Operating Systems may use these programmable devices as a part of their resources, if the operating complexity is reduced and a global language is provided.