AUTOMATIC GENERATION OF RUN-TIME PARAMETERIZABLE CONFIGURATIONS

Karel Bruneel[*] *and Dirk Stroobandt*

Department of Electronics and Information Systems
Ghent University
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
{kbruneel, dstrooba}@elis.ugent.be

ABSTRACT

In many applications, subsequent data manipulations differ only in a small set of parameter values. Because of their reconfigurability, FPGAs (Field Programmable Gate Arrays) can be configured with an optimized configuration every time the parameter values change. These optimized configurations are smaller and faster than their generic counterparts. However, the overhead involved in generating the configurations at run-time with conventional tools is very large. This paper introduces an automatic method for generating run-time parameterizable configurations from arbitrary Boolean circuits. These configurations in which some of the configuration bits are expressed as a function of a set of parameters enable very fast run-time specialization since specialization only involves evaluating these functions. Our approach is validated on adaptive filtering. We show that the specialized filter configurations produced by our method are 2.3 times smaller and 36% faster than a generic filter configuration and that these configurations can be generated in on average 166 μs. Being a generic method, run-time hardware optimization suddenly becomes feasible for a large class of applications.

1. INTRODUCTION

The inherent reconfigurability of SRAM-based FPGAs enables the use of configurations optimized for the problem at hand. Optimized configurations are smaller and faster than their generic counterparts and therefore use the FPGA's resources more efficiently. However, at some point the problem at hand will change and valuable system resources will be used to generate/select a new configuration and reconfigure the system's FPGA. These two tasks are performed by a subsystem we call the configuration manager (CM). So, actually the designer trades FPGA resources for CM resources.

Conventional synthesis tools generate FPGA configurations from scratch. This involves using heuristics to solve NP-complete problems like placement and routing. Hence, generating a new configuration requires huge amounts of CM resources. This makes run-time generation of configurations with conventional tools inefficient for many applications.

However, in many applications, subsequent data manipulations only differ in a small set of parameter values. This property enabled us to do much of the generation process off-line. In our case the off-line generation step results in a parameterizable configuration. This is an FPGA configuration in which some of the configuration bits are expressed as closed form Boolean functions of the parameters, called the tuning functions. On-line specialization of a parameterizable configuration means evaluating these tuning functions. One can easily see that the number of CM resources needed to evaluate closed form Boolean functions is much smaller than the resources needed by conventional synthesis tools.

In this paper we present a generic method for automatically generating run-time parameterizable configurations for LUT-based FPGAs. Only the truth table configuration bits of some of the LUTs will be tunable. The method starts from an arbitrary digital circuit and a set of parameter inputs. These parameter inputs are a subset of the circuit's inputs.

The core of the method is an intermediate format we call a Tunable LUT (TLUT) circuit (section 2.1). An application can automatically be mapped to such a TLUT circuit with Tmap (section 2.3). The generation of a parameterizable configuration from a TLUT circuit is dealt with in section 2.2. We validate the method on an adaptive filtering example in section 3.

2. AUTOMATIC GENERATION OF PARAMETERIZABLE CONFIGURATIONS

In this section we show how to generate a parameterizable configuration from an arbitrary combinational circuit and the parameter inputs, which are a subset of the circuit inputs. Starting from a combinational circuit does not limit generality since it is easy to extract the combinational part of a sequential circuit.

[*]Karel Bruneel is supported by a BOF grant from Ghent University.

978-1-4244-1961-6/08/$25.00 ©2008 IEEE.