Loop Transformations to Reduce the Dynamic FPGA Reconfiguration Overhead

Tom Degryse, Karel Bruneel, Harald Devos and Dirk Stroobandt
Ghent University, ELIS Department
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
{Tom.Degryse, Karel.Bruneel, Harald.Devos, Dirk.Stroobandt}@UGent.be

Abstract

Dynamic hardware generation reduces the number of FPGA resources needed and speeds up an application by optimizing the FPGA configuration at run-time for the exact problem at hand. Because of the large overhead associated with dynamic hardware generation, it is important to minimize the number of reconfigurations. In this work, we present a technique to maximize the reuse of a configuration by means of loop transformations. Our approach builds on similar work on temporal data locality optimization. Our experiments on a matrix multiplication benchmark show that we can reduce the number of reconfigurations by an order of magnitude, making dynamic hardware generation techniques much more useful in practice. When we combine our approach with a dynamic hardware generation tool with a very low overhead, so called parameterizable configurations, we can obtain a significant speed up over generic counterparts.

1. Introduction

FPGAs (Field Programmable Gate Arrays) are becoming increasingly popular as an alternative platform for custom hardware acceleration. Although FPGAs are usually slower than Application Specific Integrated Circuits (ASICs), they are often preferred over ASICs thanks to the shorter time to market, the lower non-recurring engineering costs and their reconfigurability. However, the majority of today's FPGA-based systems do not exploit this reconfigurability or only exploit it on a very large time scale, i.e., to fix bugs or to upgrade the firmware.

Dynamic hardware generation uses this reconfigurability on a much shorter time scale by exploiting run-time knowledge of the exact problem at hand. At run-time, a specialized hardware circuit is generated, which is substantially smaller and faster than the generic counterpart. If the problem at hand changes, the dynamic hardware generation tool creates a new specialized FPGA configuration and reconfigures the FPGA. Conventional synthesis tools generate a circuit from scratch, involving computationally expensive tasks such as placement and routing. This results in a huge on-line hardware generation overhead, making dynamic hardware generation using these tools inefficient for many applications.

In this work, we make use of so-called parameterizable configurations [2]. These FPGA configurations make efficient use of the fact that some inputs, called the *parameters*, change values less often than other inputs. A specialized configuration can be generated off-line by expressing some of the configuration bits as closed form Boolean functions, the *tuning functions*, of the parameters. On-line specialization of a parameterizable configuration then reduces to evaluating these tuning functions. One can easily see that this step is much faster than synthesizing an FPGA configuration from scratch. This makes dynamic hardware generation using parameterizable configurations much more useful in practice than dynamic hardware generation using conventional synthesis tools (section 2.1).

Although the on-line hardware generation overhead is drastically reduced using parameterizable configurations [2], generating the new FPGA configuration and reconfiguring the device can still take a reasonable amount of time and resources. Hence it is important to minimize the number of reconfigurations during program execution. This can be done by maximizing the reuse of the parametric input values, which is very similar to the temporal data locality optimization in a system with a memory hierarchy, as discussed in section 2.2. The application of loop transformations is one well known technique to optimize both temporal and spatial data locality. In section 3, we will show how loop transformations can be used to reduce the number of reconfigurations. To do so, we have developed a novel cost function that reflects the dynamic hardware generation overhead.

We applied our proposed technique to the matrix multiplication (section 4). This experiment shows that we can drastically reduce the number of reconfigurations, resulting in a reduced dynamic hardware generation overhead. This