Loop Controller Area Estimation for Automatic Design Space Exploration

Tom Degryse [*,1,2], Harald Devos [*,1,3], Dirk Stroobandt [*,1]

[*] *ELIS, Ghent University, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium*

ABSTRACT

High-level synthesis systems overcome the high design effort required to program an FPGA by translating an algorithm at the behavioral level into a synthesizable hardware description. At this higher level, loop transformations are used to improve the characteristics of the program. These transformations have a great impact on the resulting hardware, only known after the time-consuming synthesis steps. This hinders a fast design space exploration.

In this paper, we tackle this issue by estimating the performance of the hardware loop controller, an often overlooked component in other approaches. We present an equation based model to estimate the area of the loop controller during high-level synthesis. The presented approach is accurate enough to be useful during design space exploration. Due to its simplicity, the overhead of the estimations is minimal. The proposed methodology can easily be adapted to new FPGA design flows and architectures.

KEYWORDS: FPGA; Area estimation; High-level synthesis; Loop controller

1 Introduction

Thanks to the high degree of parallelism that is available inside an FPGA, a large class of algorithms can be accelerated by implementing them in hardware instead of running them in software on a processor. However, the translation from an algorithmic or behavioral description in software into a synthesizable hardware description in a hardware description language such as VHDL, is a time-consuming and error-prone process. This issue is tackled in several high-level synthesis (HLS) tools. During HLS, the behavioral specification of a program is translated into a low-level, synthesizable hardware description.

Algorithms can be transformed to improve the characteristics of the program, like the

[1]E-mail: {Tom.Degryse, Harald.Devos, Dirk.Stroobandt}@elis.UGent.be
[2]Tom Degryse is supported by a BOF grant from Ghent University
[3]Harald Devos is supported by the IWT grant 060068

introduction of parallelism. These transformations have a great impact on the performance of the resulting hardware, but the effect of them is only known after the time consuming synthesis steps. The application of loop transformations is one well known optimization technique during hardware generation. There is a very large number of loop transformations possible. Since iterating through the design cycle is too slow, one can not implement every possible solution during design space exploration. So there is a need for early and fairly accurate estimations to guide the designer in selecting the right sequence of transformations. Most existing estimation tools, like [Kulk06], deal with the data path only. The loop controller is not or not sufficiently considered during performance estimations. However, it can have a big impact on the quality of the resulting hardware.

This paper presents a novel approach to estimate the performance of the loop controlling hardware directly from a polyhedral description of the program. Within this model, loop transformations can easily be executed by performing simple matrix operations. This allows the fast generation of lots of design variants. When this is coupled with performance estimations, a fast design space exploration is enabled. A simple equation based model is presented to estimate the area of the loop controller. Because of this, the estimation overhead is minimal, which is critical during design space exploration. Experiments have shown that despite its simplicity, the model is still accurate enough to be used to guide the design space exploration. This paper is thus a step towards an automated design space exploration framework. The presented technique can easily be adapted to new FPGA design flows or architectures.

2 The polyhedral model

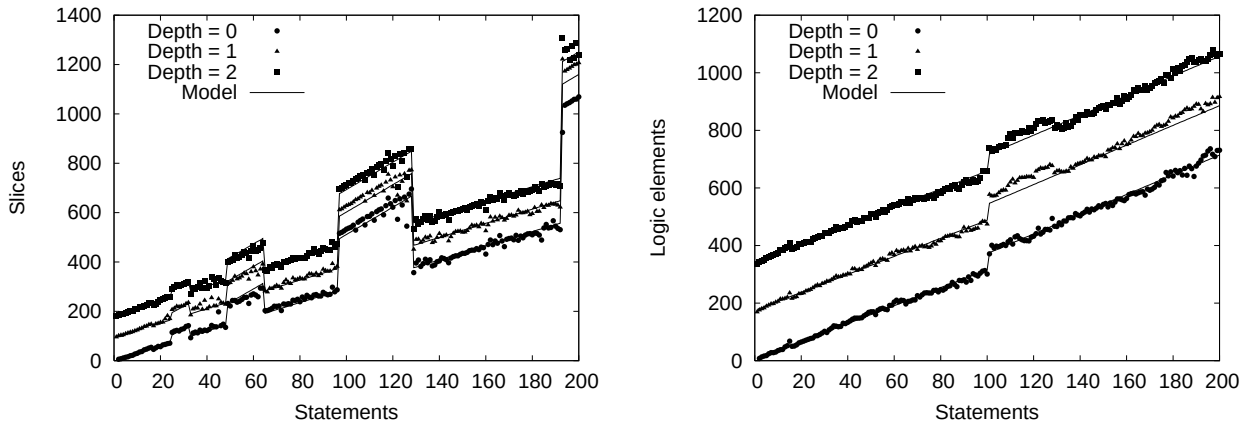
A program can be considered as a sequence of statements and control structures. A single execution of a statement inside a loop or a nested loop, a *statement invocation*, can be represented by its iteration vector:

$$\mathbf{x} = \left(i_1, i_2, \ldots, i_n\right)^{\mathbf{T}}$$

where i_k is the k^{th} loop index and n is the depth of the innermost loop. The set of values of the iteration vector for which a statement is executed defines the *iteration domain* of this statement. This is determined by the loop bounds and conditionals in the program. If all loop bounds and conditionals in a program are linear expressions (affine functions) of some parameters and the iterators of the surrounding loops, the iteration domain can be specified by a set of linear inequalities, defining a bounded polyhedron. Hence the name polyhedral model.

Loop transformations change the execution order of the statement invocations in a loop nest. All transformations in the polyhedral model are performed by transforming the matrices that describe the iteration domains. The polyhedral model makes thus an abstraction of the statement implementations. Loop transformations are easily combined by combining the corresponding matrix operations. During loop transformations the statement definitions remain untouched. The only things that change are the execution order of the statements and their arguments.

Generating code from a polyhedral representation of a program can be done with a tool called CLooG (Chunky Loop Generator) [Bast04]. CLooG generates efficient code for scanning all the integral points of one or more parametrized polyhedra. CLooGVHDL [Devo07] is an extension of CLooG translating the polyhedral representation of the program into a synthesizable VHDL description.



(a) Xilinx ISE (b) Altera Quartus II

Figure 1: Impact of the number of statements and loop nest depth on the controller area for Xilinx ISE (a) and Altera Quartus II (d). The points in the figure indicate the measured area, while the full line represents the estimation model.

3 Area estimation model

The proposed estimation strategy uses empirical equations drawn from detailed observations of the synthesis results of a set of synthetically generated controllers. A controller generator generates CLooG input files with varying parameters. CLooGVHDL is used to generate VHDL code for these controllers. This VHDL code is then further synthesized to a hardware implementation with Altera Quartus II 7.1 or Xilinx ISE 9.1. CLooGVHDL and the synthesis tools come with lots of possible settings. At this moment, only the default settings have been used. We have built a model to estimate the area usage, the number of registers needed and the clock frequency of the controller. Due to the page limitations for this paper, only the model for the area usage will be further discussed. The equations obtained from the synthetic controllers have been validated with a set of benchmarks provided with CLooG.

The two main parameters contributing to the resource usage of the loop controller are the number of statements and the maximum loop nest depth. Their influence on the area usage is shown in Fig. 1, for both Xilinx and Altera FPGAs. There is a clear difference between the two. As one can see in Fig. 1(a) several regions with different area utilization characteristics can be identified in the Xilinx graph. These regions can be divided into two categories with a different slope of the curve ($i = 1$ vs $i = 2$ in equation 1). Each category is characterized by an equation of the following form:

$$A_{xilinx} = a_i \times x + b_i \times log_2(x) + d_i \times y + c_i \ , \ i = 1, 2 \tag{1}$$

with x the number of statements in the program, y the maximum loop nest depth and a_i, b_i, c_i and d_i constants. All sets in the lowest region of the curve ($i = 1$), L_n, can be expressed as follows:

$$L_n = \{x \in \mathbb{N} \mid x \in]2^n, 2^n + 2^{n-1}]\}, n \in \mathbb{N}$$

For instance, all the points from 17 to 24 and from 33 to 48 satisfy this condition. The selection between both sets of coefficients (i.e. both regions in the curve) is thus determined by the fractional part of the base-2 logarithm of the number of statements. This is due to two different synthesis strategies used by the Xilinx synthesis tool.

The synthesis results obtained by Quartus II are much more regular than the results from ISE. The only irregularity in the area utilization curve is the gap at 100 statements. This leads to the following linear equation:

$$A_{altera} = a_i \times x + b_i \times y + c_i \ , \ i = 1, 2 \tag{2}$$

with x the number of statements in the program, y the maximum loop nest depth and a_i, b_i, and c_i constants. If the number of statements in the program is less then or equal to 100, then $i = 1$, otherwise $i = 2$.

We have also modelled other parameters influencing the loop controller area, like the presence of additional control instructions (IF) in the program.

We validated the proposed model on a set of loop controllers extracted from the SPECfp-2000 and PerfectClub benchmarks, provided with CLooG. These experiments have shown that the proposed model is fairly accurate, so it could be used during design space exploration. Because of its simplicity, the model has a very low overhead, which is critical when lots of designs have to be evaluated during design space exploration. The methodology followed is independent of the FPGA design flows and architectures used. Hence, the model can easily be adapted to future developments.

4 Conclusions

We presented a model to estimate the loop controller area at a high level. The model is accurate enough to be used during design space exploration. Due to its simplicity, the estimation overhead is minimal, which is critical when lots of design variants have to be evaluated. The presented methodology can easily be adapted to new FPGA design flows and architectures, like we showed with estimations for both Altera and Xilinx FPGAs.

References

[Bast04] C. BASTOUL. Code Generation in the Polyhedral Model Is Easier Than You Think. In *PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques*, pages 7–16, Juan-les-Pins, september 2004.

[Devo07] H. DEVOS, K. BEYLS, M. CHRISTIAENS, J. VAN CAMPENHOUT, E. D'HOLLANDER, AND D. STROOBANDT. Finding and Applying Loop Transformations for Generating Optimized FPGA Implementations. *Transactions on High Performance Embedded Architectures and Compilers I, LNCS*, 4050:159–178, 2007.

[Kulk06] D. KULKARNI, W. NAJJAR, R. RINKER, AND F. KURDAHI. Compile-time area estimation for LUT-based FPGAs. *ACM Trans. Des. Autom. Electron. Syst.*, 11(1):104–122, 2006.