A Design Exploration Environment

J. Wilberg, A. Kuth, H.-T. Vierhaus
GMD-SET
Schloß Birlinghoven
D-53754 Sankt Augustin, Germany
{wilberg, kuth, vierhaus}@borneo.gmd.de

R. Camposano
Synopsys Inc.
700 East Middlefield Road
Mountain View, CA 94043-4033, USA
raul@synopsys.com

W. Rosenstiel
University of Tübingen
Sand 13
D-72076 Tübingen, Germany
rosenstiel@peanuts.informatik.uni-tuebingen.de

Abstract

The paper describes the design exploration environment of the CASTLE system. The environment allows the exploration of hardware and software for complex processor designs. The exploration is subdivided into the measurement phase and the analysis phase. The measurement phase uses a retargetable compiler to determine the performance for a large number of different processors and different programs. These fine-grained data form the input of the analysis phase. It transforms the data into abstract representations that are visualized for the designer. The results are integrated into an HTML-based framework.

1 Introduction

The design of complex systems in short time is only feasible, if most of the design decisions are performed on a high level of abstraction. This means the main system parameters are selected without knowing all the implementation details. The basis for the selection is an automatic exploration of many different architectural alternatives [7] [16] [6]. A number of complex design tools are used in these explorations: fast instruction-set simulators are required to generate execution profiles [5] [15], retargetable-compilers are necessary for generating the scheduled code on many different architectures [10] [9] [11], and estimation tools for bounding processing times and hardware complexity [17] [3] [14]. Combining the results of these tools by hand is extremely time consuming and error prone. This paper describes the automatic generation of an exploration environment. The system creates makefiles and exploration scripts for the exploration. The results of the exploration are automatically integrated into a hypertext framework for distribution via the World-Wide Web [1].

In logic and datapath synthesis it is often sufficient to use cost functions for evaluating the exploration results [6]. In system-level design, there are many design constraints that are difficult to formulate as mathematical functions (e.g., market influences, costs of adapting existing components versus designing new parts, reliability gains from reusing existing design parts, additional costs when designing for reuse in the future). In most cases, the designer must decide on the benefits of the different possible solutions. However, the designer cannot directly use the fine-grained data produced by the design exploration. The data must be visualized in an abstract way. The paper describes the visualization of the multi-dimensional exploration data. Abstract 2-dimensional views are calculated which are connected via the hypertext framework. In this way the abstract data can be used for all the major decisions but detailed results are readily available by following the hypertext links.

The remainder of the paper is organized as follows. The next section describes the structure of the exploration environment and the automatic generation of the environment pages. Section 3 describes the visualization of the multi-dimensional design space for exploration results of the mpeg_play [13] program.

2 Exploration Environment

Fig. 1 shows the tool structure (boxes) and the result files (oval boxes) obtained by the different exploration tools. The exploration can be subdivided into two main parts: the measurement and the analysis. The measurement is the most time-consuming part of the exploration. It generates the fine-grained exploration data. Exploring 1200 different processor structures on a SPARCstation 20 takes about 5 days for the mpeg_play program. Three inputs are used for the measurement: (1) the program to be explored, (2) the exe-
Figure 1. Principal structure of the exploration environment.

cution profile from the sequential instruction-set simulation, and (3) the exploration specification. The exploration specification is a small script (approximately 10 lines of Tcl [12] code) that defines the setup of the exploration environment. It contains the exploration ranges of the different functional units, it contains the name of the program and the profiles as well as their location in the file system. The specification is read by a tool called Architec which creates an exploration script. The exploration script contains a set of nested loops. The loops iterate over all possible processor constellations in the design space that is defined by the exploration ranges and the type of exploration. For each processor constellation, a machine description generator is called. The generator produces a machine description which describes the particular processor structure for the retargetable compiler. The retargetable compiler schedules the program for that processor structure. A scheduled code is produced which is augmented with the basic block execution frequencies from the instruction-set simulation. The processing time calculation determines the execution time of the parallel code by multiplying the execution frequency of the basic block with the length of scheduled basic block. The resulting processing time is stored together with the processor constellation in the exploration data file. This file is the basic exchange format between the measurement and the analysis part. It contains all informations obtained during the measurement on a fine-grained level.

The main task of the analysis is the preparation of abstract views from the fine-grained measurement data, i.e., it can be considered as a kind of information filter which extracts the most important information for the designer. The main components for extracting the information are the exploration core and the exploration manager. The exploration core performs all calculations. It stores the results of the measurement as a multi-dimensional array. The array represents the design space that was explored in the measurement. The core calculates views on this design space as described in section 3. It allows to restrict the original design space and to calculate views on this restricted design space. This is important when selecting the functional units of the processor starting from the most important units and gradually restricting the design space until the most suitable unit constellation is obtained. The exploration manager provides an interface to the exploration core. It normalizes the exploration data calculated by the core (e.g., to make processing times independent from the length of the input data sequence). The results are stored as ASCII tables which is a convenient format for the final rendering tools and it is a suitable input format for many spreadsheet and text processing tools. In addition, the manager provides an interface to other tools, like the environment generator.

The environment generator provides the framework for
integrating the exploration tools and for storing the results in HTML pages which can be distributed via the World-Wide Web (WWW) [1] [2]. The generator creates three classes of pages: (1) HTML pages which form a kind of top level user interface and document the data at the same time, (2) the Makefiles which automate the exploration shown in Fig. 1, (3) and the analysis scripts which control the restriction of the design space and the calculation of the abstract views. The generator offers two ways for creating these pages. The first set of pages is created from the exploration specification. This bootstraps the environment. It creates default HTML pages which are often sufficient to document the exploration. The designer can modify all these pages. The second way of creating pages is during the interactive analysis of the exploration data. At this point the designer might like to create more detailed exploration data which were not provide by default (e.g., for a restricted design space). In this case the page generator is called directly from the manager. The new environment pages are created and linked to the existing pages. All subsequent runs of the exploration (e.g., if the program changes) will create the detailed data automatically.

![Figure 2. Result documentation in the HTML framework.](image)

The main interface for the designer are the HTML pages (Fig. 2) which can be accessed by a conventional HTML browser. The browser should be enabled for tool execution [2]. This allows to start the design tools directly from the HTML pages, i.e., the designer can write his/her own user interface by including the appropriate links into the HTML pages.

3 Results

We focus the description of the exploration results on the visualization of the combined exploration data [8]. The complete set of data for several different video compression programs is available via WWW [2]. This section gives the results for exploring the mpeg.play program and the flower.mpg sequence. The cycle times are normalized to cycles per macroblock. A macroblock consists of $16 \times 16$ pixel for the luminance and two blocks of $8 \times 8$ pixel for the chrominance. This normalization allows to compare data from image sequences with different length and different frame format. Two types of views are calculated: the total view and the unit view. The total view gives a summary of the performance results of all different unit types. The unit view depicts the performance obtained for a specific unit type. Fig. 3 shows the results for the adder. The x-axis shows the different number of adders that were explored in the current design space, i.e., from $1 \ldots 5$ adders. For each number of adders in the exploration range, a 3-tuple is calculated. It contains the minimum ($Min$), average ($avg$), and maximum ($Max$) cycle count over all processors in the design space with the specified number of adders, i.e., the left most tuple in Fig. 3 belongs to the subset of processors with just 1 adder. The variation between the $Min$ and $Max$ value indicates how much the performance depends on the other units that are explored in the combined exploration. If the variation is small, this means there is little influence of the remaining units. This is for example the case with just a single adder. Increasing the number of adders to five removes the adder bottleneck. The variation increases which means that the performance depends more and more on the constellation of the other unit types. For example, using five adders will require to increase the number of load/store ports as well, otherwise the large number of adders cannot be supported because the input operands are missing.

![Figure 3. The adder view from the exploration of Fig. 4.](image)

The unit view is adequate for estimating the performance of a specific unit type. The total view (Fig. 4 and Fig. 5) can be used to select the most important unit type. In this view, six values are created for each unit type. Adjacent pairs of bars in Fig. 5 give bounds on the elements of the 3-tuple in the unit view. For example $minMin$ and $maxMin$ of Add are bounds on the $Min$ value in the adder view.
If these adjacent bars differ very much, this indicates that the performance changes significantly when changing the number of units in the unit view. For example the variation between the min Min and max Min values in Fig. 4 is in the order of 67% for the adder while it is about 10% for the other units. This indicates that the performance changes significantly when changing the number of adders, as confirmed by the data in Fig. 3.

4 Conclusions

The design exploration environment of the CASTLE system was presented. Two exploration phases are used: the measurement phase and the analysis phase. The measurement performs a systematic variation of the processor structure, and determines the performance for each of the processors in the design space. These performance results are used as input to the analysis phase. Abstract views are prepared for the designer. The views provide an overview of the design space and allow to select the most appropriate unit constellation for the processor. The analysis results are automatically integrated into an HTML based design framework. The results can be distributed via the World-Wide Web.

References