High Performance Qualitative Simulation on a Multi–DSP Architecture[*]

Marco Platzner, Bernhard Rinner
{marco, rinner}@iti.tu-graz.ac.at
Institute for Technical Informatics
Graz University of Technology

Abstract

We present a special-purpose computer architecture for the qualitative simulator QSIM, which is mainly used in artificial intelligence (AI) applications. This architecture consists of DSPs TMS320C40 and specialized coprocessors (Xilinx FPGAs). We stress the distinct algorithm characteristic of qualitative simulation compared to DSP applications. We also demonstrate the suitability of the DSP TMS320C40 for this non–DSP application.

keywords:
special-purpose computer architecture,
multi-DSP TMS320C40,
qualitative simulator QSIM,
distributed operating system Virtuoso

1 Introduction

The goal of qualitative simulation is to derive a characterization of the behavior of a dynamic system given only weak information about it. This fundamental strength of qualitative simulation is exploited more and more in applications, like design, monitoring, and fault-diagnosis, nowadays.

QSIM, the widely-used algorithm for qualitative simulation has been developed by Kuipers [3]. A drawback of current QSIM implementations is poor execution speed. In our research project [6][5] a special-purpose computer architecture for QSIM is developed. Performance improvement and scalability are our most important objectives. Two approaches are considered to achieve these objectives. Complex functions are parallelized and mapped onto a multiprocessor system. Less complex but frequently used functions are directly implemented in hardware. These functions are executed on specialized coprocessors.

This paper presents the current state of our research project. Although QSIM has an algorithm characteristic which differs strongly from typical DSP algorithms, we are using the DSP TMS320C40 for our computer architecture. The suitability of the TMS320C40 for this high performance computing machine is demonstrated. Important results of QSIM analysis are presented in Section 2. Section 3 gives an overview of the computer architec-

[*]This project is partially supported by the Austrian National Science Foundation *Fonds zur Förderung der wissenschaftlichen Forschung* under grant number P10411-MAT.

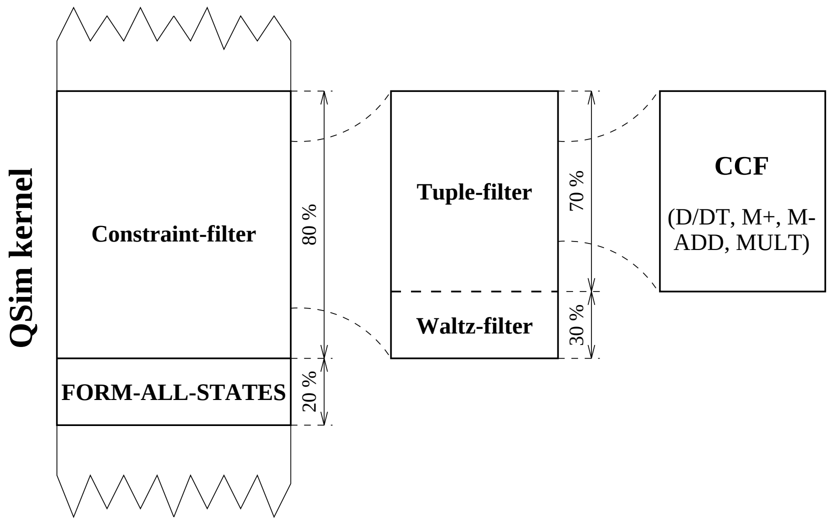

Figure 1: Runtime analysis of the kernel. Kernel functions are hierarchically structured and their runtimes are informally presented with regard to the runtime of the calling function.

ture design and some implementation details. Some remarks for further work conclude this paper.

2 Algorithm Characteristic and Analysis

The qualitative simulator QSim has a distinct algorithm characteristic which distinguishes QSim from typical DSP algorithms. Many algorithms in DSP-applications show a high inherent data parallelism. These algorithms are mainly based on extensive numerical computation, and their internal execution is hardly influenced by input data. QSim, however, can be characterized as follows.

- Symbolic computation
- Low to medium data parallelism
- High input sensitivity
- NP-completeness

Figure 1 presents an overview of the QSim kernel functions. The function hierarchy as well as runtime ratios are shown in this figure. Kernel functions are essential in calculation of one simulation step, and mostly they require more than 50 % of the overall runtime.

2.1 Constraint-Filter

The constraint-filter consists of mutually independent functions *(tuple-filter)* and the *Waltz-filter*. Figure 2 shows the pseudo-code of the constraint-filter. The number of tuple-filters depends on the input simulation model. The Waltz-filter is used for efficiency reasons to reduce the search space for the final kernel function FORM-ALL-STATES.

The constraint-filter can be easily parallelized. The independent tuple-filters are executed on individual processors. After all tuple-filter results have been received Waltz-filtering is performed.

2.2 FORM-ALL-STATES

FORM-ALL-STATES actually solves a *constraint satisfaction problem (CSP)* by a backtracking algorithm. A big search space has to be processed with a depth-

```
FOR all constraints $c_i$ DO
    tuple-filter($c_i$)
ENDFOR
waltz-filter()
```

Figure 2: Constraint-filter pseudocode

first search to find all solutions of the CSP. Although solving CSPs is *NP-complete* we did not experimentally observe an exponential behavior. The runtime of FORM-ALL-STATES did not exceed 20 % of the kernel runtime.

We use a *parallel-agent-based (PAB)* [4] strategy to parallelize FORM-ALL-STATES. The basic idea of PAB is to partition the overall search space into smaller independent sub-spaces which can be solved with any sequential CSP-algorithm.

2.3 Constraint-Check-Function (CCF)

There are many types of CCFs in current QSIM implementations (e.g. ADD, MULT, D/DT, M^+, M^-). Although the CCFs vary in their complexity these functions consist only of primitive operations, like evaluation of boolean functions, comparisons, and table-lookups. However, due to their frequent execution they dominate the overall kernel runtime. These functions are directly implemented in hardware *(CCF coprocessor).*

3 Multi-DSP Architecture

3.1 Requirements

To achieve a high performance computer architecture we defined a set of requirements for i) the processor type, ii) the interconnection network, and iii) the operating system. These requirements are derived from both the results of QSIM analysis and our objectives.

Processor type The processor requirements are high computation performance and excellent multiprocessing capabilities. This includes a large number of independent I/O channels and a high communication bandwith. In our architecture specialized coprocessors are embedded into the multiprocessor. Therefore, the protocol of the I/O channels should be convenient to implement in hardware.

Communication network The proposed computer architecture is a multiprocessor system with distributed memory. The processors are connected in a *wide tree* structure. The parallel algorithms for both kernel functions, constraint filter and FORM-ALL-STATES, have the same logical structure. The tasks are connected in a *star* structure. However, due to the limited number of I/O connections per processor, a star topology is not scalable. Thus a *wide tree* topology is a compromise between logical structure and scalability.

Operating system The development of this computer architecture has to be supported by a distributed multi-tasking operating system. This is due to two reasons: First, the use of an operating system eases a scalable and to some extend portable implementation. Second, the number of tasks of the parallelized QSIM kernel functions is not known at compile time. Hence, dynamic mapping and dynamic scheduling of all tasks is required. The operating system's overhead has to be minimal.

3.2 Multi–DSP Implementation

The digital signal processor TMS320C40 was chosen as processing element for

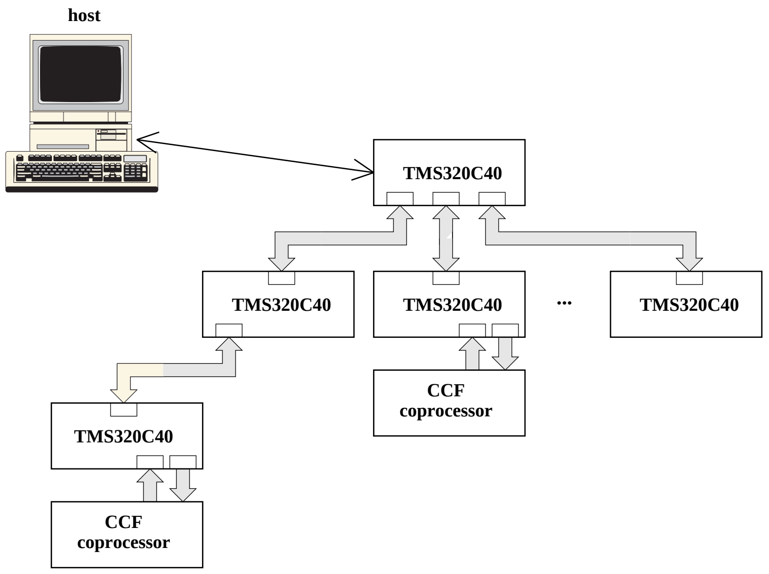

Figure 3: Example of the overall architecture. The processing elements are connected in a wide tree structure. Some processing elements are equipped with CCF coprocessors

our special-purpose computer architecture. This DSP fulfills above requirements [8]. The TMS320C40 has six communication ports, each with a data transfer rate of 20 MByte/s. All communication ports and the CPU can work in parallel. The TMS320C40 allows to implement multiprocessor trees with up to five children. Communication between TMS320C40 and the specialized CCF coprocessors is established via two independent communication ports. This reduces the hardware effort for a communication port interface to 8 data- and 2 control-lines. The distributed multitasking operating system Virtuoso [9] was chosen to support the design and implementation of scalable and portable multi-DSP software.

A prototype of the overall multi–DSP architecture is shown in Figure 3. Some processing elements of this multiprocessor architecture are equipped with CCF coprocessors. Our multi-DSP system is built using two Transtech ISA–bus motherboards each equipped with TMS320C40 TIM modules. The coprocessors are implemented using Xilinx FPGAs (XC4013). A 19” industrial PC rack serves as host and chassis for the multi–DSP architecture. Communication from the host to the multi-DSP system is established via ISA–bus and a FIFO buffer on the Transtech motherboard.

3.3 Current State

First experimental results of this project include speedup estimations for a parallelized FORM-ALL-STATES [7] and a prototype implementation for the CCF coprocessor [1]. Different partitioning heuristics for the CSP are implemented and compared using single-processor runtimes. The CCF runtime is improved using the coprocessor by up to a factor of 20 compared to a software implementation.

4 Conclusion, Future Work

We presented the design and implementation of a special-purpose computer architecture. Based on this computer architecture we demonstrated the suitability of the DSP TMS320C40 for this non–DSP application. The high I/O performance and the number of communication ports are major reasons for choosing TMS320C40 as processing element. High floating-point performance and special addressing modes of this processor are of minor interest for this project.

Ongoing and future work includes:

- Implementation of the parallel kernel functions
- Implementation of coprocessors for other often used constraint types
- Embedding of coprocessors into the multi-DSP system
- Evaluation of the overall computer architecture

References

[1] G. Friedl. Entwurf und FPGA-Implementierung eines Coprozessors für qualitative Simulation. Master's thesis, Institute for Technical Informatics, Graz University of Technology, 1995.

[2] G. Friedl, M. Platzner and B. Rinner. A Special-Purpose Coprocessor for Qualitative Simulation. In *EURO-PAR'95 International Conference on Parallel Processing*, Stockholm, Sweden, August 1995.

[3] B. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Artificial Intelligence. MIT Press, 1994.

[4] Q.P. Luo, P.G. Hendry, and J.T. Buchanan. Strategies for Distributed Constraint Satisfaction Problems. In *Proceedings 13th International DAI Workshop*, Seattle, WA, 1994. DAI.

[5] M. Platzner and B. Rinner. Improving Performance of the Qualitative Simulator QSIM — Design and Implementation of a Specialized Computer Architecture. In *Eighth International Conference on Parallel and Distributed Computing Systems*, Orlando, Florida, September 1995.

[6] M. Platzner, B. Rinner, and R. Weiss. A Distributed Computer Architecture for Qualitative Simulation Based on a Multi-DSP and FPGAs. In *3rd Euromicro Workshop on Parallel and Distributed Processing*, pages 311–318, San Remo, January 1995.

[7] J. Riedl. Parallele Algorithmen und Laufzeitmessungen für Constraint Satisfaction im qualitativen Simulator QSim. Master's thesis, Institute for Technical Informatics, Graz University of Technology, 1995.

[8] C. Steger, M. Platzner, and R. Weiss. Performance Measurements on a Multi-DSP Architecture with TMS320C40. In *International Conference on Signal Processing Applications & Technology*, Santa Clara, California, USA, September 1993.

[9] E. Verhulst. Virtuoso: A virtual single processor programming system for distributed real-time applications. *Microprocessing and Microprogramming*, 40:103–115, 1994.