A Special-purpose Coprocessor for Qualitative Simulation*

Gerald Friedl, Marco Platzner, Bernhard Rinner
E-mail: marco@iti.tu-graz.ac.at

Institute for Technical Informatics
Graz University of Technology

keywords: specialized coprocessor, FPGA, qualitative simulator QSIM

1 Introduction

Qualitative simulation is applied more and more in design, monitoring and fault-diagnosis. However, poor performance of current qualitative simulators complicates or even prevents its application in technical environments.

In our research project [3] a special-purpose computer architecture for the widely-used qualitative simulator QSIM [2] is developed. The design of this special-purpose computer architecture is mainly based on an extensive analysis [4] of current QSIM implementations. Figure 1 presents an overview of the runtime ratios of QSIM kernel functions and their hierarchical structure. An improved performance is achieved by mapping QSIM functions onto a multiprocessor system and executing runtime intensive functions on specialized coprocessors. In this paper, we present the current state of a part of this research project — i.e. the design and implementation of a specialized coprocessor for the *constraint check functions (CCFs)* of QSIM [1]. The constraint check functions are primitive kernel functions, but due to their frequent execution they dominate the overall kernel execution time. For most models, the CCFs require more than 50 % of the kernel runtime.

2 CCF Coprocessor

Current QSIM implementations include many types of CCFs (D/DT, ADD, MULT etc.). This section presents the design, implementation, and first experimental results of a coprocessor for the most complex CCF, the MULT-CCF.

2.1 Analysis of the MULT-CCF

Figure 2 shows the data dependency diagram for the MULT-CCF. This function is partitioned into 4 subfunctions SF1 to SF4. SF3 consists of n iterations. All

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

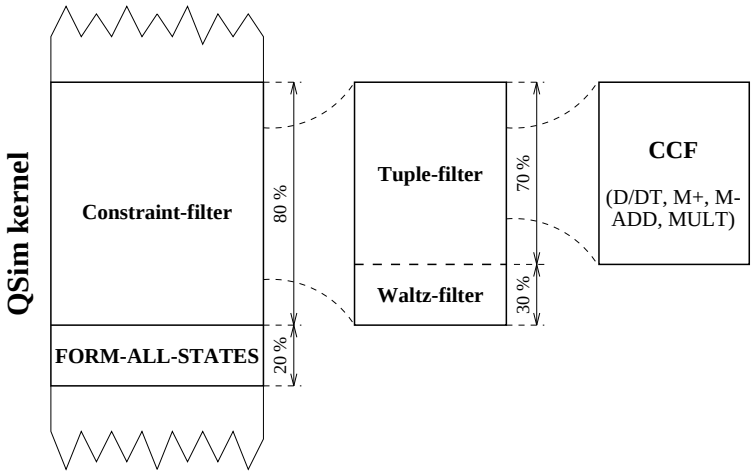

Fig. 1. Runtime analysis of the QSIM kernel. Kernel functions are hierarchically structured and their runtimes are informally presented with regard to the runtime of the calling function. The presented runtime ratios are extracted from various runtime measurements of a QSIM system implemented on a TI Explorer LISP workstation.

subfunctions produce a boolean result. Two facts can be taken from the data dependency diagram. First, the subfunctions SF1, SF2, and all iterations of SF3 are dataflow–independent. Therefore, they can be executed in parallel. Second, subfunction SF4 performs a logical AND-operation on the results of SF1 to SF3. QSIM uses *short circuit evaluation* for this logical AND-operation.

2.2 Design of the MULT-CCF Coprocessor

The MULT-CCF coprocessor was designed at the gate- and register level to obtain maximum execution speed. Main features of the design are:

- data structures are optimized for the application QSIM
- operations use maximum parallelism
- customized memory architectures allow parallel access

The block diagram of the MULT-CCF coprocessor is shown in Fig. 3. The blocks SF1, SF2, and SF3 correspond directly to the subfunctions in Fig. 2. Primitive operations of these subfunctions are comparisons and evaluation of boolean functions. These operations are supported by optimized comparators and lookup-tables. The input and output controller establish communication to the host processor (digital signal processor TMS320C40) via two separate communication channels. The operands and the instruction code are packed into a 32 bit word for communication from host to coprocessor. The function controller decodes the instruction, handles the data transfer to the functional blocks, and controls the execution of the instruction. The function controller also

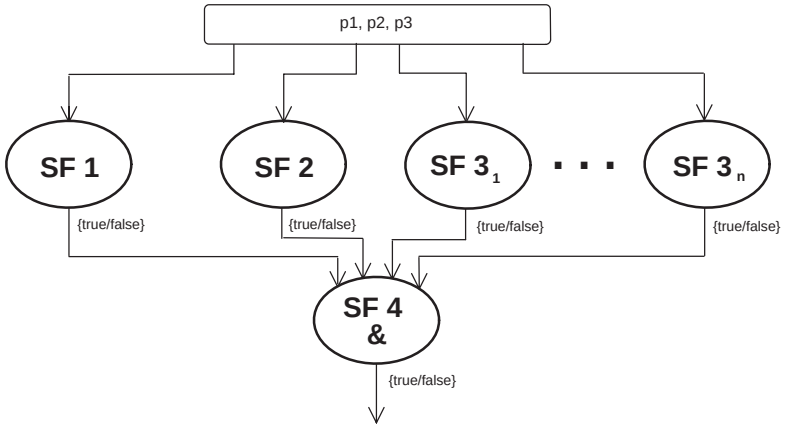

Fig. 2. Data dependency diagram for the MULT-CCF. p_1, p_2, and p_3 are input variables.

handles the short circuit evaluation mechanism. SF3 iterations are sequentially executed in the current design.

2.3 Experimental Results, Further Work

In order to compare the coprocessor to a SW reference system we consider two execution paths of the MULT-CCF. In the first case the SW implementation stops after the first executed subfunction due to short circuit evaluation. In the second case, all subfunctions have to be executed, including 4 iterations of SF3. The first execution path represents the worst case for the coprocessor.

The MULT-CCF coprocessor was implemented in an FPGA of type Xilinx XC4013 running at a clock frequency of 15 MHz. We compared experimentally the coprocessor to a SW reference running on a TMS320C40 at 32 MHz. For the first execution path the runtime improvement is given by a factor of 6, for the second execution path the gain is 20.7.

Further work includes:

- performance improvement due to routing optimization
- exploration of design alternatives (pipelining of SF3 iterations vs. simultaneous execution of SF3 iterations on several SF3 function blocks)

References

1. G. Friedl. Entwurf und FPGA-Implementierung eines Coprozessors für qualitative Simulation. Master's thesis, Graz University of Technology, 1995.
2. B. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Artificial Intelligence. MIT Press, 1994.

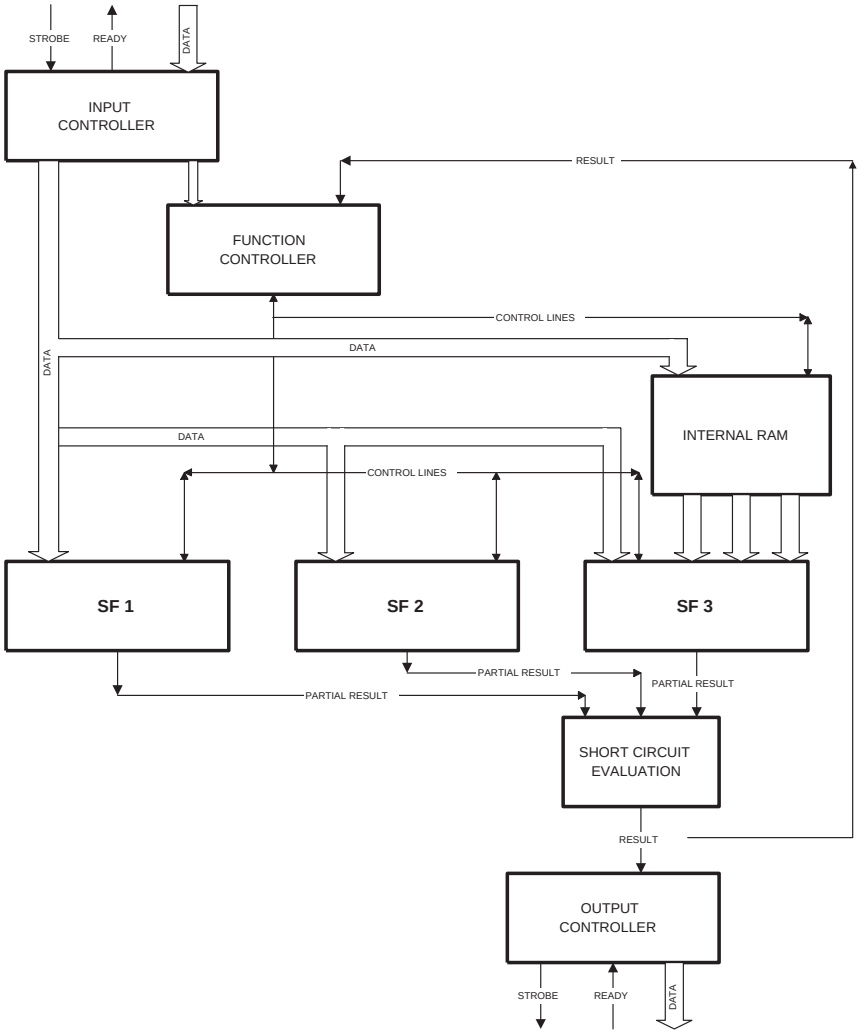

Fig. 3. Block diagram of the MULT-CCF coprocessor. The input and output controller connect the coprocessor to the host processor via an 8 bit data bus and two handshake lines (STROBE, READY). For simplification control lines between input/output controller and function controller are not shown.

3. 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. IEEE Computer Society Press.
4. B. Rinner. Konzepte zur Parallelisierung des qualitativen Simulators QSIM. Master's thesis, Graz University of Technology, October 1993.

This article was processed using the LaTeX macro package with LLNCS style