LOGAT

Mgr. Vít Strádal
Department of Software and Computer Science Education
Faculty of Mathematics and Physics
Charles University in Prague

Ing. Rudolf Matoušek
Department of Signal Processig
Institute of Information Theory and Automation
Czech Academy of Science

October 9, 2000

Abstract

A behaviour of atomic arithmetic operations (+, -, *, /) in a S-function cannot be changed in the Simulink. It disallows to model the behaviour of a complex algorithm on a special arithmetic hardware. One of possible ways to solve the problem is to use a S-function parser which converts arithmetic operators to functions. The parser is a part of a simulator of a Logarithmic Arithmetic Unit. Paper describes the development of the parser performed under the EU ESPRIT 33544 HSLA Long-term research project, coordinated by the University of Newcastle, UK.

1 Introduction

The main aim of the project is to develop the logarithmic arithmetic unit as the ASIC/FPGA hardware. For such development is very important simulation. It should verify the behaviour of the unit on cumulative operation sets. Another task of the simulator is the research of suitable algorithms for such type of arithmetic.

The simulator utilize a flexibility of the Field Programmable Gate Array technology (FPGA) as well as the user friendly interface of the Simulink. Our department has been involved in the algorithmic research area. We have quite large library of algorithms written in C-mex S-functions. The C-code parser converts existing algorithms to a code acceptable by the simulator of different arithmetics domain.

2 Simulator

The first version of the simulator has been written in PASCAL (see [1]). Such code was not portable to the Simulink, thus we rewrote some parts of it to a C library. It has provided full functionality but weak flexibility. All experiments was written in C. Results was displayed by a special data file parser written as a m-file in Matlab.

To perform tests more effectively, the atomic operators have been implemented as C-mex S-functions. It allowed to perform some automated experiments on cumulative operation sets in Simulink, without manual recoding of the original algorithm.

2.1 Structure

The contemporary version of the simulator is written as the IP core for Xilinx Virtex XCV1000 device in HANDEL-C. The hardware part of the simulator is based on a RC1000 prototyping board from Embeded Solutions (nowadays Celoxica) and it is connected by a special interface to Simulink. The Real-Time Workshop can be used to create a standalone application which improves the simulation performance.

Algorithms are created in Simulink by provided toolboxes. User-defined algorithms (written in C) are connected to Simulink by a template S-function. The S-function is an interface which allocates all the needed memory space and then it simply calls the included algorithm. The included C-code has

to be preprocessed by a C-code parser – LOGAT. The standard S-function compilation can be then performed and the model is prepared for the simulation operating on the 32bit logarithmic arithmetic format replacing the double format.

3 LOGAT

When we started to deal with the simulator, we thought about a modified C-code compliler, which would be able to work with a log type. Such development is not easy and it does not cover the project ideas. So we decided for the C-code parser LOGAT, which is much easier to implement but it needs a special approach to C-code writing.

3.1 Function description

The prarser is an utility which can be executed on all systems running gcc, including MS Windows platforms. It reads the original C-code and returns a processed C-code working in the 32bit logarithmic domain.

The first task of the parser is to find all variables and constants of the double type and convert it by a warp function to a logat type. The next thing is to convert all variables and constants, which are in a expression with the logat type, to the logat type. The next operation performed by the LOGAT is a operator replacing task. It changes all operators between logat types to warp functions. And finally, the last task of the parser is a conversion of the result to the demanded type.

The created code removes all FPU[1] operations and it can be compiled for simulation in the 32bit arithmetic domain by the C-mex compiler. The simulation-type can be choosen by the linking process. All the warp functions has the same names for the 32bit FPU, for the software emulated LAU[2] and for the hardware LAU version. Therefore, there are three libraries, each implementing one functionality.

The simulation execution is performed by the usual way. The choosen library forces the Simulink to call the demanded simulator each time, when a C operator is performed on the logat type or when a conversion is needed.

3.2 Mechanism

A C-code parser is build by bison and flex utilities (for details see [10]). Parser builds grammar tree of the input C-code and creates a table of new data types, functions and global or local variables in prallel.

The grammar tree is scaned backward and to each node is assigned a type. A special nodes with a explicitly defined conversion (type $\Leftrightarrow$ logat) are marked. The grammar tree is read recursively and the output is printed or saved in the outfile. Marked nodes are encapsulated by the right convert function. For example if the node is marked to convert int to logat, the 'lg_itol(' is printed then all subnodes are processed and then ')' is printed.

Binary and unary operations on logat types are solved in the same way. The operation consists of a left and right subnode. The LOGAT prints for example 'lg_mul' first then left subnode then it replaces operator ('*' in our case) by ',' then it prints right subnode and finally the close parentheses ')'. This process is applied on the tree recoursively until all subnodes are converted.

4 Conclusions

LOGAT helps to automatically generate code which can be simulated on the standard 32bit FPU, on the software emulated LAU and on the FPGA based LAU hardware by special i/o interface wrappers. It allows port to almost all available prototyping boards. The support is now being prepared for ADC RC1000 FPGA Reconfigurable Computing PCI Card (see [5]). We plan to support ADM-XRC PMC module (see [6]) and XSV800 board (see [7]).

Note, that some of the hardware support is not fully implemented yet, because of library linking problems. We are trying to fix it now. The contemporary FPGA LAU hardware interface for RC1000 is based on the Matlab m-file interface, matfiles and binary loaders. LAU hardware is described in a separate paper.

[1] Floating-Point Unit
[2] Logarithmic Arithmetic Unit

References

[1] J.N.Coleman: *A High Speed Logarithmic Arithmetic Unit*, ESPRIT Long-term research project No. 33544 "HSLA"

[2] J.N.Coleman, E.I.Chester: *A 32b Logarithmic Number System Processor and Its Performance Compared to Floating Point*, Proceedings 14th IEEE Symposium on Computer Arithmetic, Adelaide 1999

[3] HSLA 33544 Project: *LNS ALU home page*, http://napier.ncl.ac.uk/hsla/

[4] Vít Strádal: *LOGAT web page*, http://popelka.ms.mff.cuni.cz/~vitas/logat

[5] Celoxica: *Homepage*, http://www.celoxica.com/

[6] Alpha Data parallel systems: *Homepage*, http://www.alphadata.co.uk/

[7] XESS Corporation: *Homepage*, http://www.xess.com

[8] The MathWorks: *Real-Time Workshop*, User's Guide, version 3, 1999

[9] The MathWorks: *Simulink*, Writing S-Functions, version 3, 1999

[10] *Bison and Flex man pages*, UNIX man system

Contact

Mgr. Vít Strádal
Kabinet výuky software a informatiky, MFF, UK
Malostranské náměstí 25
118 00 Praha 1

email: vitas@popelka.ms.mff.cuni.cz
phone: +420-2-2191 4240
fax: +420-2-2191 4281
www: http://sun3.ms.mff.cuni.cz/ksvi/

Ing. Rudolf Matoušek
Oddělení zpracování signálů, ÚTIA, ČSAV
Pod vodárenskou věží 4
P.O. Box 18
182 08 Praha 8

email: matousek@utia.cas.cz
phone: +420-2-6605 2264
fax: +420-2-6605 2511
www: http://www.utia.cas.cz/ZS