Reconfigurable General-Purpose Processor
Presentation of idea

Igor Zarzycki, Lodz University of Technology

Abstract
In this paper the idea of the reconfigurable general-purpose processor (later in this document called REμP) implemented in dynamically reconfigurable FPGA is presented. Proposed solution is still in stage of development and because of that this paper presents the idea briefly not going into details of implementation. The novelty of the proposed solution lays in the lack of typical sequential processing. All operations are realized in parallel in the hardware. This solution should give large speed-up in comparison with existing ones. At the same time the new architecture does not impose any modification of the software development process. All necessary modifications can be done at the stage of compilation of currently portable code.

1. State of the art
Since few years the maximal frequency of processor clock is at the level of about 3GHz and is not increasing. The overall gain of computation power is achieved by small architectural changes and increase of cores number. Dual core and multi-core processors are constantly enlarging their market share. Similar trend can be observed even for mobile devices although for them power dissipation is one of crucial elements.

On the other hand reconfigurable devices are commonly used for scientific calculations. Reconfigurable computing is a new computing paradigm filling the gap between software and hardware. Reconfigurable hardware can be easily adapted to fulfil applications demands by changing the functionality.

Thanks to dynamic reconfiguration being supported in commercially available FPGAs (Field Programmable Gate Arrays) several different concepts of using them as computation acceleration have appeared. Initially the dynamically reconfigurable resources were used as stand-alone coprocessors. Later, thanks to the technology development, sequential processors where implemented together with dynamically reconfigurable resources. In [1] UNIX slightly modified to run on PowerPC of Virtex allows programmers to treat the tasks implemented in the reconfigurable hardware in the same way as software tasks running on processor core. However development of new applications or porting the existing ones requires a low-level programming approach that is far from modern approach to software creation making it difficult.

2. General description of idea
Although currently active research is done in the field of Application-Specific-Instruction set Processors (ASIPs) the one proposed in this paper is intended to be a general-purpose processor not application specific one and because of that it should be able to execute any kind of application not only particular problems. It also does not assume implementation of internal sequential processor core nor hardware simulation of such. The basic concept is creation of partitions in the run-time in dynamically reconfigurable FPGA. Such partitions should be responsible for realization of whole program or in most cases of its part and be loaded on demand at the moment when they should be performed or if this moment is approaching. Because of that program is no longer a list of instructions, but becomes a graph of partitions solving its fragments. This approach must also apply to elements such as operating system, drivers, shared libraries or interrupt handlers.

Such approach allows parallel execution of large amount of threads and high parallelisation of tasks within single thread. The execution of program can be divided to configuration and execution of partitions. Of course one partition can be configured when other is executed. This
allows to hide the configuration time if it is similar or shorter than execution time.

When the partition becomes no longer needed its resources are released and can be used for other partitions. Control unit is responsible for managing the partitions – configuration, release and freeze/restore. Partitions themselves are also not directly responsible for operations such as memory accessing.

![Diagram of concurrent thread execution](chart.png)

**Fig. 1. Concurrent thread execution**

Thanks to better adapting to executed task and higher parallelism (see fig. 1) than in standard, sequential processors achieved speed-up should significantly reduce time necessary to perform most of the tasks (even including initial configuration time, ones extending execution time or not “hidden”).

Additionally because of better resources usage overall power consumption should be lower than in traditional sequential processors. Modern researches shows that FPGAs have lower power consumption per task. This can be significant for mobile devices that are currently forced to use highly power consuming multi-core processors.

### 2. Hardware problems and solutions

Modern FPGAs assumes that bitwise operations are dominating. This is quite useful approach for building any kind of state machines that are important for control logic. However processors should mainly perform arithmetic operations on whole words rather than on single bits.

Additionally modern FPGAs try to give maximum flexibility which significantly extends their physical dimensions and makes their configuration far more complicated. Because of that their configuration is significantly longer than execution and makes configuration “hiding” during execution impossible, especially in case of larger number of offspring partitions. In such approach the size of configuration words of single cell (the smallest configurable element of partition) is of order of several kilobits. This makes the configuration time few hundreds times longer than execution time.

Because of this problem reduction of configuration words was necessary. For proposed architecture the size of words required to configure a single cell varies from 64 bits to 160 bits. This is from 2 to 5 configuration words, 32 bits each. The size of configuration data depends on usage of resources of particular cell. Unused resources are not being configured. This approach is strongly reducing total number of configuration words for whole partition and hence configuration time.

All those solutions are significantly reducing configuration time making it (on average) of the same order as execution time. Although flexibility is reduced, the number of cells in partition necessary to realise basic operations still stays similar or lower. This is minimising the overall time program is spending on configuration of its partitions without execution of any.

Additionally physical size of single cell was reduced. This allows to place more cells and to create either larger number of partitions or more complex ones. In current architecture it is planed to place few thousand cells controlled by few control units in single integrated circuit. Each configuration unit is responsible for about one thousand cells.

Proposed architecture is designed to operate internally on 32bit words. However thanks to possibility of adaptation it can also perform calculations on either longer or shorter words. Additionally some specific, complex applications that currently had to be performed in software, like coding/decoding speech, image, video data or...
3. Software problems and solutions

One of the crucial elements is simple creation of new programs or porting the already existing ones. Creation of processor without software or for which creation of software would be difficult is pointless. Because of that compilation of existing source codes written in high level software description language to any HDL (hardware description language) or directly to bitstream is crucial.

Currently there are many projects [2][3] able to convert software description languages like C++ to HDLs. However all of them are limited and are unable to perform the full conversion of most program source codes. The main source of limitation is assumption that such converters convert only algorithms that are fully implementable in FPGAs within single partition and are only described by means of software description language. This approach is designed for algorithms that can be performed by single, stand-alone device. Those converters do not assume the usage of FPGA as processor. Hence problems with such elements as standard functions, interrupts, memory access or pointers in general.

In this project it is assumed to use front-end of existing compiler, add elements operating on its intermediate code and finally create back-end producing bitstream directly. Since REμP architecture is adapted to operate as processor with precisely defined ways of communication with external elements (such as memory) and partitions changing mechanism, there should be no complex problems with compilation of most of the software description languages to bitstream. Because of that there is no need to create any specific programming language nor to program REμP in HDL directly (that would require to rewrite all programs making the processor useless).

Compiler for REμP should also analyse code and perform optimisations specific for proposed architecture. Each basic operation can be performed in many ways. The choice of particular cells layout should be based on such conditions like particular usage of operation (if it should be performed many times in small number of clock cycles or can use fewer resources and be performed slower with shorter configuration time) or border conditions (where from get the input data and where to place output one). Those elements (but not only them) are crucial in the stage of compilation.

Additionally optimisations should increase parallelism and reduce some data dependences. One of the main REμP advantage is massively parallel execution even at the level of single instructions. Lack of parallelism in program may seriously spoil performance. Hopefully some optimisations for parallel execution are already present in existing compilers. Even if they will produce output not enough efficient they may still useful and be a good base for further development.

Another important optimisation is reduction of partition switching. Quite often some partitions are frequently used (for example loop bodies). In such situation freezing the partition after end of its execution, slight modification of its parameters and rerunning it when needed may significantly reduce configuration time. Of course some of the partitions may be to big to be stored for longer time. Because of that surrounding of the particular partition is very important for optimisations and code generation. This element often may be hard to determine, for example in case of system functions, where only address of such function is known at the stage of compilation but not the function itself.

Solving the problems of partitions fitting and dealing with large ones require additional, empirical research. The analyse at the current stage of design can give only the general concept, but not the proper solution to them. Similarly with others optimisation problems mentioned before.

Another important software problem is the operating system. Although there is extremely large variety of currently existing ones, they may be not well suited to cooperate with REμP. The novelty of this architecture puts also high demands on the software. Satisfying them will require further work on both compiler and the operating system.

4. Summary

This paper briefly presents the concept of dynamically reconfigurable processor (REμP) realised by means of FPGA. The main novelty of this project are lack of sequential execution, typical for currently available processors and implementation of all operation in hardware. To fulfil those assumptions there is necessary to create new architecture different from both typical sequential processor and currently available FPGAs. Additionally to perform programs optimally (not to lose gain speed-up) there is required specific compiler and operating system.
fulfilling high demands put on the software. To create them necessary is additional, empirical research, that can not be performed at the current stage of the project.

5. Thanks

At the end of this document I would like to express my thanks for given support to team developing the REμP project. This is to:

Ph.D. Rafał Kielbik – head of the project,
Ph.D. Piotr Amrozik,
Ph.D. Grzegorz Jabłoński,
Ph.D. Konrad Przygoda.

And additionally to professor Andrzej Napieralski, M.Sc., Ph.D., D.Sc., SM IEEE, d.h.c. – head of department for enabling us to perform the research.

Bibliography


Authors:
MSc. Igor Zarzycki
Lodz University of Technology
ul. Włóczanska 221/223 budynek B18
90-924 Łódź
Poland
tel. +48 42 631 26 45
fax +48 42 636 03 27
email: izarzycki@dmcs.p.lodz.pl