5-th International Conference "High Performance Computing" HPC-UA 2018 (Ukraine, Kyiv, October 22-23, 2018). P. 127-130.

Reconfigurable Manycore System

Anatoliy Sergiyenko[1], Olexiy Molchanov[1], Pavlo Serhienko[1],

[1] *Computer Engineering Dept. of Sikorsky Kyiv Polytechnical Institute, 37 Peremogy Ave., Kyiv, Ukraine*

aser@comsys.kpi.ua

Abstract. *An approach for designing the reconfigurable computing systems in FPGA is proposed, which is based on mapping synchronous data flow graphs to a manycore system. The reconfiguration is performed by switching data flows and exchanging the instruction sets of the processor cores. To implement the processor elements of such a system, a 16-bit RISC-processor core is developed, which has small hardware costs and a configurable instruction set.*

Keywords

VHDL, FPGA, RISC, configurable computer.

1 Introduction

The expansion of the field programmable gate arrays (FPGAs) as the basis of the configurable computers is a constant trend. Usually, the application-specific processor is configured in FPGA once per operation period. Sometimes, FPGA is fully or partially reconfigured during this period providing an optimal computing structure for various fragments of the implemented algorithm. In the latter case, we speak about the dynamically reconfigured computer (DRC) [1,2].

As a rule, DRC implements the data flow algorithms for which it is possible to predict volumes of computing portions and interprocessor communications. When designing RC, the algorithm is modeled, computation portions are mapped in the processor units (PUs), the variants of the processor structure are determined, and a set of configuration files are generated, which are then loaded dynamically into FPGA [1].

New series of FPGAs of Altera and Xilinx firms support a partial reconfiguration. But, first of all, the minimum portion of the reconfiguration covers the rectangular area of the FPGA, which is quite large. For example, in the Xilinx FPGA, this area includes four DSP48 multiplication blocks. In addition, a part of the FPGA that has not changed the configuration and the island with a new configuration should have unchanged common data points. All this leads to difficulties in designing the DRC and restrictions to optimize hardware costs. Secondly, the minimum partial reconfiguration lasts several milliseconds, that is, it constitutes a large time overhead [2].

The new series of FPGAs have a number of features that infer the choice of architectures that are configured in. If the number of transistors in the FPGA increases in four times, then the tracing resources increase only twofold. The delays in the lines is in 1–3 times greater than ones in the look-up tables (LUTs) and in other logic elements [2]. Therefore, the effective structural solution is the use of compact configurable modules, so that their internal signals are not spread far.

Time and hardware overhead costs for reconfiguring the DRC can be reduced to the minimum if such reconfiguration does not change the FPGA configuration. Many works, for example [3], propose a multiprocessor architecture of DRC, which consists of a set of identical compact PUs, which are bound to a configuration system of commutation or a network on a chip. Such a system can be quickly and dynamically reprogrammed. The disadvantage of this architecture is that PUs are highly specialized, which imposes limitations on a plurality of implemented algorithms.

A program-driven processor core with a specialized instruction set is preferable to select as PU. When implemented in an FPGA, such a processor can dynamically modify the executable algorithm, due to the overlay structure of its programs. Besides, the usual configuration technology provides the exchange of the inner program RAMs of these PUs in the configuration file without recompiling the whole project. In addition, the instruction set of separate PUs can be dynamically changed by the partial reconfiguration of FPGA without affecting the common data transmission points.

The popular microprocessor cores, such as Nios, Microblaze, Mico32, OpenRISC, MIPS, do not fit the compactness requirements mentioned above. To achieve an acceptable clock frequency, these processors have a multi-staged pipeline

(up to 5 and more stages). System interrupts, context exchange, virtual memory, cache-RAM, and high-speed interface to external dynamic memory are implemented in such cores to execute several independent program threads in them. In order to implement the multiprocessor architecture, the high-bandwidth busses and hardware controllers are required for maintaining the of memory access coherence [4].

In these conditions, the architecture of the XMOS manycore system has a set of advantages. This architecture inherits the positive properties of the transputer architecture. Among them are the high-speed real time multithreaded operation without the use of the operational system, and the effective interprocessor communications through a set of the virtual links [5].

The purpose of the work is to construct a manycore DRC in FPGA, which implements the synchronous data flow model, thereby reducing hardware costs, increasing the speed of the system and the possibility of the dynamic reconfiguration. For this purpose a compact core of the RISC processor was developed, which plays the role of a PU in such a system.

2 Synchronous dataflow system

The synchronous data flow graph (SDF) is a convenient model for representing many algorithms with the cyclic nature, such as digital signal processing algorithms. SDF consists of nodes or actors that perform computational functions and edges, which interconnect the nodes in the system. Each edge can have a FIFO buffer to perform the dataflow delay for several cycles. The actor operates immediately, as soon as there is data at its inputs and outputs the results to its outputs. In contrast to the data flow graph of the general type, each actor in SDF consumes and issues the same amount of data during a single cycle of the algorithm execution.

The SDF model has the properties, such as deadlock absence, static scheduling, ability to execute a large set of algorithms. Due to this, SDF is used for both the programming of multiprocessor systems [6] and for the design of pipelined computing structures [7, 8, 9].

With a one-to-one mapping of the SDF algorithm, the structure of the system is derived, which is isomorphic to this graph. Each PU of such a structure performs a single process of calculating the corresponding function-actor. When the next data group arrives the input ports, the process starts, calculates the function, writes the results to the output ports, and stops.

In the case of *N*-to-one mapping of SDF, up to *N* actors are implemented in a single PU. In this case, PU is programmed according to the methods, which are proved in [6, 10]. Consequently, each PU of the system performs only a single program flow. They do not require the operating system, interrupt system, virtual memory, and the like. That is, such PUs have small hardware costs and they are compact in their FPGA implementation.

The cycle of SDF computing in such a parallel system is determined by the critical path which passes through PUs with the maximum loading. The rest of PUs, through which the critical path does not passes, are underloaded [6, 7]. Therefore, there is a problem with the PU loading balance.

3 Dynamic reconfiguration

A task, which requires the dynamic reconfiguration, should be represented as a set of different SDFs. Then, these SDFs are mapped in several manycore structures that are differentiated by the structures. These structures are combined into a single combined DRC structure using respective switching nodes. The tasks performed in PUs are described as programs in the corresponding programming language and are compiled in the PU executing codes. In this case, a set of used instructions is selected and allocated in each PU, where it is configured, as proposed in [10]. The resulting manycore system project is configured in FPGA along with the program codes according to the usual method, using the CAD tools of the FPGA manufacturer.

When solving the PU problem, the DRC executes its programs, and its reconfiguration consists only in the corresponding adjustment of the switching nodes and in the re-loading of programs in the PU from the external memory. Such a process of reconfiguration occurs much faster than with the full or partial change in the configuration of the FPGA.

4 PU architecture

A 16-bit processor core RISC-ST2 was developed as PU of the manycore system, which implements SDF. The RISC-ST core, which is described in [9], was taken as the basis. The instruction bit width is increased to 18, which made it

possible to optimize the instruction set and to use better the internal memory resources of FPGA. A set of specific instructions is added to the RISC-type instruction set. They process the selected bits, select bit fields in the word, merge fields, shift word, count the position of the most significant bit. To implement a quick access to the associative table, an instruction of the hash function calculating is added as well. Such instructions contribute to the effective implementation of the parsing and compression algorithms.

A set of used instructions is programmed during the formation of the configuration file, as proposed in [10]. As a result, the small footprint PUs and shortened program codes are derived, which are effectively configured in FPGA.

The registered memory of the processor has 32 registers, among them 16 registers are available in the instruction. In order to maximize the possibility of the registered memory, which is implemented on LUTs, a single instruction can read three operands from these registers and write a single result for a single clock cycle.

The following addressing modes are available: registered, indirect registered, base addressing, index addressing with the pre-increment. The data memory is divided into 4k byte pages and has a maximum capacity of 256 megabytes. To access the data memory, the standard Wishbone open interface is used.

Up to 256 peripheral registers are addressable. Such registers serve as input-output registers. The data exchange between PUs is implemented through these registers as well as the communication with the application specific processors is. These processors execute high-speed computing, for example, the calculation of elementary functions, digital signal processing, or encryption. Several PU cores are synchronized by the interrupt system. Due to the fact that the instruction pipeline has three stages, the most instructions are executed for a single clock cycle, and the branch, call and memory reading instructions are executed in two cycles.

The core of the microprocessor is described by the VHDL language and has no restrictions for the synthesis and configuring in any FPGA series. The processor model has a built-in disassembler, which simplifies the testing and debugging of programs. A cross-assembly program in Java has been developed, which outputs the VHDL files of the program memory.

5 Implementation of the processor core in the FPGA

Table 1 shows the results of the PU core synthesis for the Xilinx Kintex-7 FPGA and its closest known analogs. Compared to the analogs, the developed PU core has lower hardware costs and higher performance. It should be noted that the delay in the interconnection lines accounts for 75% of the delay of the critical path in this core. The core compactness makes it possible to place about three hundred cores in Xilinx xc7k480t FPGA, each with 12 kilobytes of program and data memory.

Tab. 1. Implementation of microprocessor cores in the FPGA Xilinx Kintex-7

Microprocessor core	Bit width	Hardware volume, LUTs	Maximum clock frequency, MHz
RISC-ST2	16	653	217
OpenMSP430	16	1387	150
OpenRISC1200	32	4945	107

The RISC-ST2 processor can be configured in FPGAs of different series. Table 2 shows the results of the processor configuration, which contatins the PU core and the minimum set of the peripherals.

Tab. 2. Implementation of the RISC-ST2 core in the FPGA

FPGA series	Hardware volume,		Maximum clock frequency, MHz
	LUTs	registers	
Xilinx Zynq	799	262	148
Xilinx Spartan6	824	264	133
Xilinx Virtex7	760	226	205
Altera StratixV	967	239	227
Altera Cyclone V	988	240	150
Altera Max10	2322	1309	76
Lattice ECP5U	742	245	112

6 Conclusion

An approach to the creation of manycore system on a FPGA chip is proposed, which is based on the mapping the SDF graph to the system structure. The approach provides to obtain a dynamically reconfigurable high-performance system for processing data flows with the reduced hardware costs.

The RISC processor core with an application specific instruction set is developed, which forms a basis of the manycore system. Due to the fact that this core is adapted to perform a single computing process, it has both small hardware cost and high speed. Up to several hundreds of such cores can be configured in a single FPGA chip forming the manycore system. It is possible to develop the manycore systems to process the data streams with the throughput of several tens of billions of operations per second. It is planned to create a system that executes fast compression - decompression, high-speed parsing of files. For this purposes, the ideas of performing of the GIF decompression in the proposed core [11], and of the high-speed reconfigurable parser [12], are helpful.

References

[1] A. DeHon, R. Tessier, K. Pocek, Reconfigurable Computing Architectures. *Proceedings of the IEEE.* V. 103, No. 3, – P. 332 –354, 2015.

[2] D. Koch, *Partial Reconfiguration on FPGAs. Architectures, Tools and Applications.* New York: Springer, 295 P., 2013.

[3] M. Lanuzza, S. Perri, P. Corsonello, MORA: A New Coarse-Grain Reconfigurable Array for High Throughput Multimedia Processing. *Proc. 7-th Int. Conf. on Embedded Computer Systems, SAMOS'07.* P. 159 – 168, 2007.

[4] Processor Design. System-on-Chip Computing for ASICs and FPGAs / Edited by J. Nurmi. *Springer,* 525 p., 2007.

[5] D. May, The XMOS Architecture and XS1 Chips. *IEEE Micro,* V.32, N.6. P. 28 – 37. 2012.

[6] E. A. Lee, S. Bhattacharyya, P. K. Murthy, *Software Synthesis from Dataflow Graphs*. Springer, 190 p., 1996.

[7] S. S. Bhattacharyya, H. Kee, C.-C. Shen, I. Wong, Y. Rao, and J. Kornerup, Mapping Parameterized Cyclo-Static Dataflow Graphs onto Configurable Hardware. *Journal of Signal Processing Systems*, V. 66, No 3, P. 285–301, 2012.

[8] А. М. Сергиенко, В. П. Симоненко, Алгоритмические модели обработки потоков данных. *Электронное моделирование.* Т.30, №6, С. 49–60, 2008.

[9] А. М. Сергиенко, *VHDL для проектирования вычислительных устройств*. Киев: ДиаСофт. 2003. 210 с.

[10] A. Sergyienko, O. Maslennicov, Ju. Shevtshenko, Configurable Microcontroller Array, *Proc. Int. Conf. on Parallel Computing in Electrical Engineering,* 2002. PARELEC'02, P.47–49, 2002.

[11] А. М. Сергиенко, А. А. Молчанов, П. А. Сергиенко, И. В. Мозговой, Система динамической визуализации на базе аппаратного GIF-декомпрессора, *6-та міжнародна наукова конференція «Моделювання-2018»,* С. 221–223, 2018.

[12] А. М. Сергиенко, А. А. Молчанов, Мікроконтролер з системою команд, що розширюється, *Summer InfoCom Advanced Solutions,* P. 58–60, 2017