Piotr DZIURZAŃSKI, Tomasz MĄKA, Paweł FORCZMAŃSKI
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE, WYDZIAŁ INFORMATYKI,
ul. Żołnierska 49, 71-210 Szczecin

Hardware acceleration of data classifiers for multimedia processing tasks

Dr inż. Piotr DZIURZAŃSKI

He received the MSc and PhD degrees in computer science from Szczecin University of Technology in 2000 and 2003, respectively. He is currently working as an assistant professor in Faculty of Computer Science & Information Technologies, West Pomeranian University of Technology, Szczecin. His scientific interests include hardware-software co-synthesis, high level synthesis and formal verification.

e-mail: pdziurzanski@wi.zut.edu.pl

Dr inż. Tomasz MĄKA

He received the MSc and PhD degrees in computer science from Szczecin University of Technology in 2000 and 2005, respectively. He is currently working as an assistant professor in Faculty of Computer Science & Information Technologies, West Pomeranian University of Technology, Szczecin. His scientific interests include hardware realization of digital signal processing systems and acoustic signal processing techniques.

e-mail: tmaka@wi.zut.edu.pl

Abstract

In this paper, experimental results of a proposed hardware acceleration of feature extraction and data classifiers for multimedia are presented. This hardware is based on multi-core architecture connected with a mesh Network on Chip (NoC). The cores in the system execute both data classifiers and feature extraction for audio and image data. Using various meta heuristics the system is optimized with regards to different data communication criteria. The system was implemented on an FPGA platform with use of ImpulseC hardware description language.

Keywords: feature extraction, multimedia data classification, Network on Chip, ImpulseC.

Sprzętowe przyspieszenie klasyfikacji danych multimedialnych

Streszczenie

W artykule zostały zeprezentowane wyniki eksperymentalne dotyczące sprzętowego przyspieszania ekstrakcji cech i klasyfikacji danych multimedialnych. Opracowane rozwiązanie sprzętowe bazuje na architekturze wielordzeniowej, w której każdy blok realizuje przypisaną mu statycznie funkcjonalność. Rdzenie połączone są ze sobą za pomocą sieci wewnątrzukładowej (ang. *Network on Chip*, NoC) o architekturze siatki. W artykule opisano pokrótce autorskie oprogramowanie służące do generowania kodu sieci wewnątrzukładowej. Graficzny interfejs użytkownika został zaprezentowany na rys. 1. Narzędzie ma za zadanie dokonywać odwzorowania wybranych funkcjonalności do poszczególnych rdzeni z wykorzystaniem takich meta-heurystyk jak algorytmy genetyczne, symulowane wyżarzanie, poszukiwanie losowe czy algorytmu gradientowego. Jako kryterium optymalizacji można wybrać minimalizację całkowitego przesyłu danych, minimalizację maksymalnej liczby danych transmitowanych przez pojedyncze łącze, a także minimalizację odchylenia standardowego rozmiaru strumieni transmitowanych przez poszczególne łącza. Przykładowe wyniki optymalizacji losowej dla sieci wewnątrzukładowej zostały przedstawione w tab. 1, natomiast wyniki optymalizacji dla sieci wewnątrzukładowej wykorzystującej inne podejścia - w tab. 2. Dla systemu zoptymalizowanego w ten sposób został wygnerowany opisujący go kod w języku ImpulseC, który następnie posłużył do syntezy sprzętowej na układzie FPGA z rodziny Xilinx Virtex 5. Zajętość układu XC5VSX50T dla trzech wykorzystanych klasyfikatorów została przedstawiona na rys. 3. Z kolei tab. 3 przedstawia liczbę zasobów wykorzystanych przez narzędzie syntezy wysokiego poziomu dla tych klasyfikatorów. Technika przedstawiona w publikacji umożliwia określenie warunków i ograniczeń implementacji sprzętowej systemu służącego klasyfikacji danych multimedialnych.

Słowa kluczowe: ekstrakcja cech, klasyfikacja danych multimedialnych, sieci wewnątrzukładowe, ImpulseC.

Dr inż. Paweł FORCZMAŃSKI

He received MSc title in 1998 and PhD title in 2002 from Faculty of Computer Science and Information Technology, Technical University of Szczecin (at present West Pomeranian University of Technology, Szczecin). His scientific interests include digital image processing, recognition and analysis, methods of dimensionality reduction and image classification in biometrical applications, content-based image retrieval and visual surveillance. He is the author of over 60 scientific articles, chapters in books, conference papers.

e-mail: pforczmanski@wi.zut.edu.pl

1. Introduction

One of important branches of today's IT technology is processing and analysis of multimedia data, including image, audio and video. Due to the increasing size of multimedia streams used in contemporary computer systems, their processing with traditional, software-based means is hindered by the limited performance of these systems. It is the main motivation of applying more effective models of computation, which are characterised with high parallelisation level and can be implemented in dedicated hardware structures [6].

Currently one of most important yet difficult problems associated with processing multimedia data is to recognise them (and, in particular, to classify them). A typical process of multimedia data recognition usually comprises two tasks: preparing the representative database and performing the classification process. Increase in the number of multimedia stream descriptors usually leads to improving the proper classification [7]. However, in software implementations of such multi-descriptor classification the unavoidable sequential computation and relatively large number of synchronization between tasks result in poor performance even using a multi-core CPUs and the parallel programing fork-and-join paradigm, as presented in [5]. Thus it looks more tempting to implement message-passing approach and to realise each descriptor calculating algorithm on a separate core, as it leads to decrease in the number of synchronization points. Then the main problem of obtaining computational effective systems moves to the communication domain, as relatively large streams are to be transmitted to each core [1]. It is the reason why we decided to utilize the mesh-based Network on Chip (NoC) architecture. In this architecture each processing core (except the boundary ones) is connected with its four neighbours with short links. A typical NoC architecture is a packet-switching network, where each packet is split into a number of even smaller packages, named flits (flow control units) [2]. The whole flit is transmitted between neighbouring IP cores at once.

Despite the growing popularity of NoCs, commercial EDA tools dedicated to this technique are rather rare. A commercial tool Arteris [8] is capable of linking IP blocks using wormhole switching with static routing. The academic tools Nostrum and Nirgam allow us to simulate a network for a given traffic, but are incapable of either generating synthesizable code, or using a given set of IP blocks [3]. Despite the fact that xpipescompiler is capable of generating a synthesizable code, it can use only the wormhole switching [4]. To the best of authors' knowledge, there is no available tool that offers a wide set of core mapping heuristic as

the one proposed in this paper. Also, there is presumably no other NoC-generation system dedicated to multimedia data feature extraction and classification, except for the one presented in this paper.

2. Generation of multi-core SoC code

In order to automatize the process of creation the multi-core SoC implementing multimedia data features extraction and classification, we decided to develop a software being able to generate a code (in a C-based hardware description language) for both cores and NoC-based connections using the parameters and the core set list provided by a user.

The main window of the tool is presented in Fig. 1, where 16 NoC nodes implementing the multimedia features extraction and classification are used. In the picture, the numbers near the links inform about the transfer (in bps), whereas the arrows indicate the direction of the transfer. The values shown in the picture are computed for a single-path XY routing algorithm without any optimization applied.

At the first step, the user is asked to provide the size of the SoC connected with mesh NoC. The SoC in its both dimensions is allowed to have from 2 to 6 rows/columns, so the system contains up to 36 cores. To each of these cores the user can assign various functionalities, including both audio and image data sources, audio and image features extraction elements and such classifiers as KNN, GMM and NBC. For audio feature extraction a user can choose one of the following core functionalities: frame statistics, time-based feature, chroma, spectrum calculation, spectrum statistics, SBP, frequency-based features, MFCC, autocorrelation, AC features, LPC, PARCOR, LPCC, LPC residual, RS features (see [5] for more details about these features), whereas for images it is possible to use RGB2HSV conversion block, quantization, convolution, EHD - Edge Histogram Descriptor, and DCD - Dominant Color Descriptor. After these assignments, the user can generate the appropriate HDL code for the system, or to perform some data communication optimization, described in the next section.

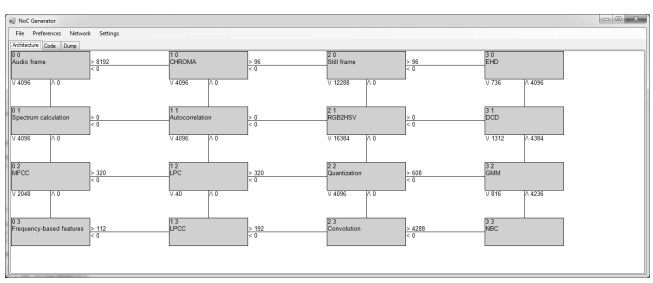

Fig. 1. The main window of the application
Rys. 1. Główne okno opracowanej aplikacji

3. Data communication optimization

In order to perform data communication optimization in a mesh NoC utilizing the wormhole switching with XY routing algorithm, we implemented four optimization techniques: random optimize (RO), simulated annealing (SA), genetic algorithm (GA) and hill climbing (HC). These techniques are used to change the position of the cores in order to minimize one of three criteria: minimization of the total transfer between all nodes, minimization of the standard deviation to eliminate hot spots, and minimization of the maximal data stream transferred by a single link. In the case of RO, the user is asked to provide the number of iterations. In each iteration, two cores are chosen randomly and their functionalities are exchanged.

If the value of the objective function is improved with respect to the selected criteria, the exchange is kept. Otherwise, it is cancelled. In case of SA, the user is asked to provide the maximal number of iterations together with initializing temperature and the value of temperature change in each iteration. Applying the GA requires providing the maximal number of epochs, population size and mutation probability (in pro miles). In HC there are no additional parameters.

In Tab. 1 we present the obtained results for the maximal amount of data transmitted by a single link using the random optimization strategy and NoCs of various sizes with a fixed set of nodes for each configuration. This strategy is of particular importance with respect to scalability assessment, as the number of iterations is deterministic, which is in sheer contrast with simulated annealing and genetic approaches. We executed each case 10 times and noted down the best result.

Tab. 1. Random optimization strategy results for NoCs
Tab. 1. Wyniki optymalizacji losowej dla sieci wewnątrzukładowej

NoC size	Iterations	Transfer, bps	Time, ms
2x2	100	4096	0.01
2x2	1000	4096	8
2x2	10000	4096	28
3x3	100	8192	1
3x3	1000	4136	46
3x3	10000	4096	455
4x4	100	8192	4
4x4	1000	8192	46
4x4	10000	8192	464
5x5	100	14336	10
5x5	1000	12288	116
5x5	10000	12288	1138

In Tab. 2 we studied one particular difficult case with other optimization strategies. In this particular 3x3 NoC we have an audio source node that transfers relatively large portion of data (4096 bps) to four different nodes (LPC residual, Autocorrelation, Frame statistics and Time-based features). Thus, after placing the source node in the centre of the mesh (i.e., in the best position for it) all its (four) outgoing links transmit the same amount of data (4096 bps). Despite a number of trials, only one heuristic (SA) produced a result close to this global minima regardless the parameters. Although the results obtained by SA for 100 and 10 iterations are the same, the difference is that in the majority of trials for 10 iterations SA returned the 8192 bps transfer, whereas for 100 iterations the result close to the optimum was more frequent. Relatively large number of the maximal number of epochs and population size is needed in the case of GA, since other parameters lead to unfavourable 8192 bps. Interestingly, despite this result is relatively easy to be found by SA and RO, GA with parameters 100/10/2 almost in every trial leads to 12288 bps. Also, it is worth noticing the computation time difference between SA and GA. HC analyses only a very limited subset of the search-space, thus the poor transfer but short time is typical while it is used. However, this algorithm is useful after obtaining a relatively promising result by RO, SA or GA, as in the close vicinity of that result can be found even a better one, which is easy checked by HC.

Tab. 2. Optimization results for NoCs; parameter order is consistent with the enumeration in the paper
Tab. 2. Wyniki optymalizacji dla sieci wewnątrzukładowej; kolejność argumentów jest zgodna z podaną w treści publikacji

NoC size	Strategy (Parameters)	Transfer, bps	Time, ms
3x3	SA(1000/300000/0.99)	4140	101
3x3	SA(100/300000/0.99)	4156	88
3x3	SA(10/300000/0.99)	4156	70
3x3	GA(1000/100/2)	4136	1731
3x3	GA(100/100/2)	4136	1150
3x3	GA(100/10/2)	8192	150
3x3	HC	8192	1

4. Implementation

Since the original algorithms described in the previous section were realized in C and SystemC languages, we chose one of the

existing C-based hardware description language to implement the hardware-targeted counterpart of the source code, namely ImpulseC [9]. It is an extension of ANSI C with new data types, aimed at hardware synthesis, and new functions and directives for steering the hardware implementation. The code is executed in the so-called processes, which communicate with other processes using streams, signals, shared memory and semaphores. As the processes are to be realized in hardware, they may benefit from various ImpulseC optimization techniques, such as loop unrolling or pipelining. The first of these techniques are quite important in our system, as numerous computations in data-dominated algorithms are independent of each other. However, there are usually no enough resources for generating hardware for each iteration, thus some kind of clustering is necessary. This trade-off between the computation time and target chip area can be established during a series of experiments. This is also the path followed by the authors. After some code modifications, aiming at improving its hardware realization, we obtained our final code to be implemented in hardware. These modifications included division of the code into coarse-grain partitions to be implemented in parallel. The transformation is always a trade-off between the computational time and the resource utilization, hence we analyzed the impact of each module onto the final realization in terms of particular functional blocks, such as adders, multipliers etc., estimated DSP blocks (present in our target FPGA chip) and the number of computational stages. To estimate the impact of these modifications into the target chip parameters, we used Stage Master Explorer tool from the ImpulseC CoDeveloper package. This tool computes two parameters, Rate and Max Unit Delay (MUD), which approximates the performance of future hardware implementation. It is worth stressing that these parameters are computed instantly, in contrast with long-lasting hardware implementation.

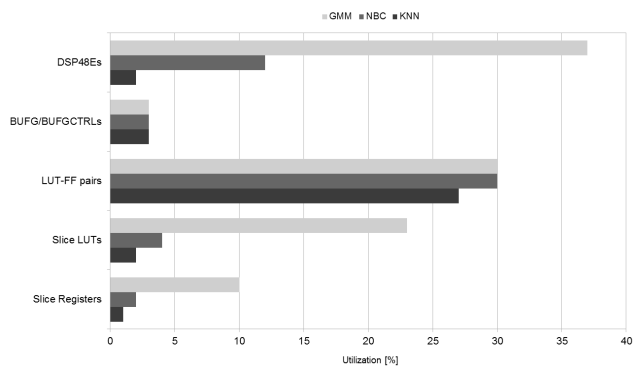

Fig. 3. XC5VSX50T utilization for three classifiers
Rys. 3. Zajętość układu XC5VSX50T dla trzech klasyfikatorów

Tab. 3. The number of resource used by a high-level synthesis tool
Tab. 3. Liczba zasobów wykorzystanych przez narzędzie syntezy wysokiego poziomu

Resource type	Number of used resource		
	KNN	NBC	GMM
1-bit Adder/ Subtractor	2	0	2
3-bit Adder/ Subtractor	0	0	5
4-bit Adder/ Subtractor	7	14	4
5-bit Adder/ Subtractor	0	0	6
6-bit Adder/ Subtractor	2	3	0
32-bit Adder/ Subtractor	12	28	37
32-bit Multiplier	1	6	10
32-bit Divider	1	6	4
2-bit Comparator	1	1	1
32-bit Comparator	10	14	31

After the resource assignment process for each classifier, we obtained the results presented in Tab. 3, where the elementary operators used for the implementation of the classifiers are given. In a similar way we found the implementation parameters for feature extraction cores and the NoC routing facilities.

At the last stage, we used Xilinx ISE to perform an implementation of the core in Virtex5 FPGA device (XC5VSX50T, Virtex 5 ML506 Evaluation Platform). We analysed a simple case when only three classifiers were to be implemented and got the following device utilization: 6035 Slice Registers, 14941 Slice LUTs, 4606 fully used LUT-FF pairs, 57 DSP48Es.

The above assignments mean that in the case of each classifier less than 45 percent of the device were used, leaving more than half for the NoC routers and the remaining cores. Fig. 3 presents FPGA utilization for these three classifiers.

5. Conclusions

The techniques, used in the described software can help in determining the conditions and limitations in implementing features extraction and classification functionality into a set of independent IP cores connected with a Network on Chip.

Having determined both NoC size and functionality mapping into a set of cores leading to satisfying transfer, one can generate synthesisable ImpulseC code of each core and the underlying transfer infrastructure. Also some further modification of transferring mechanisms (not described in the paper), such as duplication of the processing cores (usable in the case when hot-spots are determined) or allowing multi-path streaming transfers are possible to be firstly tested, and next to be applied in the obtained code generated by the developed tool.

The research work presented in this paper was supported by the Polish National Science Centre (grant no. N N516 475540).

6. References

[1] Bjerregaard T., Mahadevan S.: A survey of research and practices of Network-on-chip, ACM Computing Surveys (CSUR) archive, Vol. 38, Issue 1, 2006.
[2] Dally W. J., Towles B.: Route packets, not wires: on-chip interconnection networks, Proceedings of the 38th conference on Design automation, Las Vegas, Nevada, United States, June 2001, pp. 684-689.
[3] Jain L., Al-Hashimi B. M., Gaur M. S., Laxmi V., Narayanan A.: NIRGAM: A Simulator for NoC Interconnect Routing and Applications Modeling, Workshop on Diagnostic Services in Network-on-Chips, Design, Automation and Test in Europe Conference (DATE' 07), April, France, 2007, pp. 16-20.
[4] Jalabert A., Murali S., Benini L., De Micheli G.: xpipesCompiler: a tool for instantiating application specific networks on chip, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DAC'04), Vol. 2, pp. 884 - 889, 2004.
[5] Maka T., Dziurzanski P.: Parallel audio features extraction for sound indexing and retrieval systems, 55th International Symposium ELMAR, 2013, pp.185-189, 2013.
[6] Smit G., et al.: Efficient Architectures for Streaming DSP Applications, Dynamically Reconfigurable Architectures, Internationales Begegnungsund Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.
[7] Theodoridis S. and Koutroumbas K.: Pattern Recognition, Academic Press, 2009.
[8] Arteris(TM), The Network-on-Chip Company, A comparison of Network-on-Chip and Busses, 2005.
[9] Impulse Accelerated Technologies, Accelerating HPC and HPEC Applications Using Impulse C, Reconfigurable Systems Summer Institute (RSSI), Urbana, IL, July 17-20, 2007.

otrzymano / received: 08.03.2014
przyjęto do druku / accepted: 02.05.2014 *artykuł recenzowany / revised paper*