Bartosz CHOJNACKI, **Piotr DZIURZAŃSKI**, Tomasz MĄKA
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE,
ul. Żołnierska 49, 71-210 Szczecin

Multi-Stream Routing in Network on Chip for Data-dominated Algorithms

MsC Eng. Bartosz CHOJNACKI

He graduated from the Faculty of Computer Science & Information Technology, West Pomeranian University of Technology in Szczecin in 2010. Currently he is working in an IT company in Szczecin. His research interests include hardware implementations in SystemC.

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

PhD Eng. Piotr DZIURZAŃSKI

He graduated from the Faculty of Computer Science & Information Technology, Szczecin University of Technology in 2000. He obtained the PhD degree at the same Faculty in 2003. Currently he is working at the Chair of Computer Architecture and Telecommunication, West Pomeranian University of Technology. His research interests include hardware/software co-design, high-level synthesis, and formal verification.

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

Abstract

In this paper a multi-path routing algorithm dedicated to Network on Chip (NoC) together with its implementation is presented. The proposed algorithm is based on the Ford-Fulkerson method and is aimed at data-dominated multimedia applications realized in Multi Processor Systems on Chip. The efficiency of the proposed technique is compared with the state-of-the-art NoC routing. Our implementation utilizing virtual channels allows us to obtain promising results in some popular multimedia codecs.

Keywords: Network on Chip, multi-path routing, Ford-Fulkerson method.

Routing wielościeżkowy w sieciach wewnątrzukładowych dla algorytmów zdominowanych przez dane

Streszczenie

W artykule został przedstawiony wielościeżkowy routing przeznaczony do sieci wewnątrzukładowych (ang. *Network on Chip*, NoC) wraz z jego implementacją. Proponowany algorytm został oparty na metodzie Forda-Fulkersona i jest przeznaczony do multimedialnych aplikacji strumieniowych zdominowanych przez dane, realizowanych w wielo-procesorowych systemach jednoukładowych (ang. *Multi Processor Systems on Chip*, MPSoC). Efektywność prezentowanej techniki została porównana z najpopularniejszym algorytmem routingu używanym w NoC - XY. Badania eksperymentalne wykazały, że w niektórych przypadkach uzyskano znaczącą poprawę czasu transmisji. Przedstawiona implementacja algorytmu wykorzystuje kanały wirtualne i, chociaż wymaga wykonania dodatkowych obliczeń, umożliwiła otrzymanie obiecujących wyników dla niektórych popularnych kodeków Multi-medialnych, natomiast dla innych uzyskano nieco gorsze wyniki. Stąd trudno jednoznacznie wnioskować o wyższości wielościeżkowych mechanizmów routingu nad tradycyjnymi jednościeżkowymi. Routing typu tapeworm należy zatem postrzegać jako alternatywną propozycję routingu przeznaczoną dla strumieniowych algorytmów realizowanych w NoC, która poszerza przestrzeń poszukiwań korzystnej realizacji układowej. W niektórych przypadkach jej stosowanie znacznie polepsza wyniki, czasami zaś lepiej zastosować tradycyjne podejście. W chwili obecnej autorzy nie są w stanie zidentyfikować cech wspólnych algorytmów, które są korzystnie realizowalne z wykorzystaniem proponowanej techniki.

Słowa kluczowe: Sieci wewnątrzukładowe, trasowanie wielościeżkowe, metoda Forda-Fulkersona.

PhD Eng. Tomasz MĄKA

He graduated from the Faculty of Computer Science & Information Technology, Szczecin University of Technology in 2000. He obtained the PhD degree at the same faculty in 2005. Currently he is working at the Chair of Computer Architecture and Telecommunication, West Pomeranian University of Technology. His research interests include audio and speech signal processing and hardware signal processing algorithms' implementations.

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

1. Introduction

Contemporary multimedia algorithms, such as MPEG-4, DAB, DVB, MP3 and many others, are typically computational-intensive and data-dominated but they can be split into stages to be implemented in separate computational units. Thanks to this property, they can benefit from parallel and distributed processing working in a pipeline-like way and transmitting streams of relatively large, but usually fixed, amount of data. In these applications it is usually required to keep an assumed quality level of service and meet real-time constraints [12]. Multi Processor Systems on Chips (MPSoCs) are often considered as suitable hardware implementations of these applications [9]. As each processing unit of a MPSoC can realize a single stage of streaming application processing, it is still problematic to connect these units together.

The most popular routing algorithm used in NoCs, named XY, can be also viewed as inappropriate for switching large streams of information. According to this algorithm, a flit is firstly routed according to the X axis as long as the X coordinate is not equal to the X coordinate of the destination core, and then it is routed vertically. Despite being deadlock-free [10], this algorithm is not adaptive and thus is not equipped with mechanism for decreasing the contention level.

Moreover, as it was shown in [6], in a mesh-based NoC realizing a typical streaming multimedia algorithm, few links are used significantly while the remaining ones are utilized in a small degree and relatively large part of links are not utilized at all.

Taking into consideration the above mentioned facts, it follows that in order to design a NoC-based MPSoC for multimedia streaming applications, it is necessary to propose a routing algorithm that is more suitable to this task that the traditional XY algorithm and to propose a mapping scheme of IP cores into mesh nodes that decreases the contention level. In this paper, we focus on the first of these issues.

2. Tapeworm Routing

In order to avoid the majority of problems found in a usage of the XY algorithm for streaming multimedia applications, we introduced a multi-path routing scheme that we named Tapeworm routing. We propose this name due to the similarity with the anatomy of a tapeworm - both its body and a package body are split into segments; segments are comprised of a number of flits.

The Tapeworm algorithm uses the well-known Ford-Fulkerson [7] method to compute maximal throughput of the network between a set of cores and for each core permutation. The Ford-Fulkerson method for an arbitrary flow network $G = (V,E)$; where V is a set of vertices and E is a set of edges with source s and sink t. Each edge has capacity $c(u,v)$ and flow $f(u,v)$, denotes the notion of the residual network and augmenting path (u, v). The residual network for ow network G is network $G_f = (V, E_f)$; where E_f is defined as follows: $E_f = \{f(u,v) \in V \times V : c_f(u, v) > 0\}$, where $c_f(u, v)$

denotes the residual capacity for path (u, v) which is defined with $c_f(u,v) = c(u,v) - f(u,v)$: The augmenting path for a network is any path from s to t in a residual network for G. The residual capacity for any augmenting path for network is determined with the following formula: $c_f(p) = \min\{c_f(u,v):(u,v)\in p\}$.

With the notions defined as above, we can present the Ford-Fulkerson method in a pseudo-code given in Fig. 1. In case of the Tapeworm algorithm, the input data is a list of data transfers. A transfer T_i consists of three elements (S_i, D_i, A_i), where S_i and D_i denote the source and the target router, respectively, and A_i is a number of bits transmitted between these routers. The pseudo code of the whole Tapeworm algorithm is presented in Fig. 2.

```
1. while any augmenting path exists $p \in G_f$
2.   for each $(u,v) \in p$ do
3.     $f(u,v) := f(u,v) + c_f(p)$
4.     $f(v,u) := f(v,u) - c_f(p)$
```

Fig. 1. Ford-Fulkerson algorithm
Rys. 1. Algorytm Forda-Fulkersona

```
1. for a given permutation $p$
2. Max = $\Sigma_i A_i$
3. Min = 0
4. while (Max > Min)
5.   create flow network $G$ for the appropriate NoC
6.     for each transfer $i$
7.       find all paths $P_j$ between $S_i$ and $D_i$
8.       sort $P_j$ with respect to the paths length
9.       sort the paths in $P_j$ of equal lengths
         according to XY rule
10.      determine minimal flow $c_{min}$ utilizing
         the Ford-Fulkerson method
11.        $A_i = A_i - c_{min}$
12.        if ($A_i$ > 0)
13.          Max = Average(Min, Max) - 1
14.        else
15.          Min = Average(Min, Max) + 1
```

Fig. 2. Pseudo-code of the proposed algorithm for transfer balancing
Rys. 2. Pseudo-kod proponowanego algorytmu do wyrównywania transferów

Each router owns a routing table that stores all the paths to the target router sorted according to their lengths; paths of the same length are sorted with the XY rule. Following this rule, the first path is obtained based on the XY routing algorithm. In the second path, the flit is routed horizontally as long as the X coordinate is lower (or higher, according to the direction in X between the source and the target nodes) by 1 from the target router. Then the flit is routed vertically by one router, and then according to the XY algorithm. In the next path, a flit is routed horizontally as long as the X coordinate is lower (or higher) by 2 of the target router, etc. This approach guarantees receiving the flits in the same order as they were sent [5].

The Tapeworm and the XY algorithms include some common properties as, for example, dealing with deadlocks by limiting the possibility of flit turning [2] and by utilizing the wormhole type of switching. The difference between these algorithms is clearly visible in the number of paths used to transmit data between two routers. It consists of 4 functional blocks. Buffers N, S, E, W receive data from their neighbouring routers. Buffer L receives data from the directly connected core. In the scheme there is no output buffers as the Tapeworm algorithm operates according to the Wormhole switching technique that permits to buffer flits of a single package in a few routers at once, such that the routed flits can be immediately transferred into input buffers of the next router. Data from buffers is then transferred to a switch which implements the Tapeworm algorithm. Moreover, the switch controls the flit flow in the router, reserving inputs for the corresponding outputs and works also as an arbiter.

In the description there are no included, for simplification, virtualrec, virtualsend and input-enabled ports. These ports are a part of the virtual channel mechanism. The virtual channel is a pair of buffers in one physical channel which is shared with other such pairs. An access to a physical channel is controlled by a dedicated arbiter. Its introduction is aimed at getting rid of the deadlocks. This mechanism can be also used for decreasing the delay in a network and for increasing its capacity [4]. Using virtual channels in the presented router is indispensable for implementing the Tapeworm algorithm. Tapeworm utilizes the Ford-Fulkerson method [7] to determine paths that can have common edges for various messages and thus can send flits originated from various messages using the same channels at the same time. In the described router architecture it is possible to realize up to 3 virtual channels for each physical channel, which was enough for the analysed algorithm. However, this approach is scalable and the maximum number of virtual channels can be easily increased. The switch block is the most complex router block considering its implementation that consists of executing Tapeworm, which computes paths for transferring messages between the source and the destination router. Based on these paths, it creates the routing table for sending messages.

The first column denotes the physical channel from which the data has been obtained. The second column denotes the number of the target router. (The routers are numbered from left to right and from bottom to top of a network, i.e., in case of 3x3 network, the first router is placed in bottom-left corner. The 2nd router is its right neighbour, whereas the 4th router is placed above the 1st one.) The third column informs about the virtual channel of a given physical channel to be used for input data routing. The fourth column contains the maximum number of bits that can be sent using the virtual channel, whereas the last column contains the number of bits sent by this channel. It is worth commenting on the situation when the data from an input channel and a target router has to be routed to more than one virtual channel. Then the arbiter iterates through virtual channels and assigns the first free one. If there is no free router, then it assigns the channel whose queue is the shortest and which is not in the unused state. After creating the routing table, a switch iterates through each virtual channel for every physical channel checking if a header flit, denoting the beginning of a new message, has appeared. If this happens, based on the routing table an output is assigned. Then the switch controls flit flow.

3. Experimental results

The router architecture has been developed and implemented in CoCentric System Studio - a design and simulation environment allowing us to use the SystemC language. In order to compare both the Tapeworm and XY routing algorithms,we performed one default mapping of functionalities into cores. Only in case of the MP3 codec we checked 5 different mappings of functionalities into cores for the Tapeworm routing. In our research we have analysed 3 different codecs: iLBC [1], MP3 [3], and H.264 [11] for generating traffic in a network. From each of these codecs we have singled out functional blocks to be implemented in separate cores. For example, the flow graph for MP3 is shown in Fig. 3. (Notice that it contains 7 functional blocks so 2 nodes in a 3x3 mesh NoC remain unassigned). Each core (C1-C7) corresponding to particular functionalities has generated randomly messages of the constant size as long as the number of sent bits has not reached the number of bits defined in the transfer of a particular core of the codec. The maximal time needed for sending and receiving data in the network generating traffic for the iLBC codec, in the network routed according to the Tapeworm routing algorithm, is more than 38% lower comparing with the XY-based routing (Fig. 4). The standard deviation is over 10% and 25% lower in case of the Tapeworm algorithm, i.e., the traffic in the network routed according to this algorithm is more balanced, what should result also in lower power dissipation [2]. However, the maximal time needed for sending and receiving data in network generating traffic for the MP3 codec and using the Tapeworm algorithm is higher than its is XY-routing-based counterpart, which implies longer time of the codec execution by about 29%. The standard deviation for sending and receiving data is over 7% higher, which shows worse traffic balance in the network.

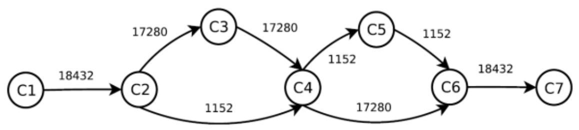

Fig. 3. Flow graph for MP3 audio decoder (flows given in bps)
Rys. 3. Diagram przepływów dekodera MP3 (przepływy podano w bps)

The maximal time needed for sending and receiving data through the network generating traffic for the H.264 codec and utilizing the Tapeworm routing scheme is lower than in the XY routing algorithm case by over 19%. In case of the Tapeworm algorithm, the standard deviation of sending and receiving flits is lower by 12% and 13%, respectively.

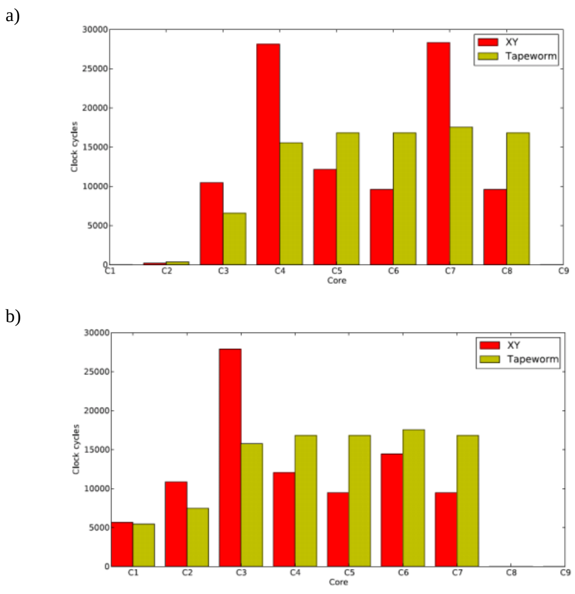

Fig. 4. Receiving (a) and sending (b) times for iLBC codec
Rys. 4. Czasy otrzymywania (a) i wysyłania w kodeku iLBC

Tab. 1. Examples of MP3 codec mappings and their communication time
Tab. 1. Przykłady odwzorowania kodeku MP3 i ich czasy transmisji

mapping communication	time [cycles]
(C1, C2, C3, C4, C5, C6, C7, C8, C9)	30316
(C9, C1, C2, C3, C4, C5, C6, C7, C8)	29965
(C3, C2, C1, C6, C5, C4, C7, C8, C9)	29608
(C1, C2, C3, C5, C4, C6, C7, C8, C9)	26419
(C4, C2, C3, C1, C5, C6, C7, C8, C9)	30334

In Tab. 1 the results of the experiments for the network generating traffic for the MP3 codec and various arbitrary chosen core mappings are given. We denote the mapping as a string of numbers, where position of a number denotes the router number and the number at this position means the core number that is connected to this router. For example, using mapping (C1, C2, C3, C5, C4, C6, C7, C8, C9) we can obtain almost 13% improvement in terms of the network speed than in the default mapping (C1, C2, C3, C4, C5, C6, C7, C8, C9), but still the network with the XY protocol works faster for the MP3 codec.

4. Conclusion

In this paper we have presented an architecture of a router implementing the Tapeworm routing algorithm realized in the SystemC language. Then, a survey on the efficiency of a Network on Chip using the implemented router has been given. The obtained results have been compared with the network realizing the same applications using the XY algorithm. The Tapeworm algorithm has been meant to result in faster networks with more balanced transfers [5, 6], but according to the experimental results it is not always better than the traditional approaches. In case of the iLBC encoder, the whole communication time is lower by more than 38% in the network with the Tapeworm routing.

On the other hand, a Tapeworm-based implementation of the MP3 encoder is more than 29% slower in comparison with the XY-based NoC. For the H.264 video encoder, the Tapeworm approach is again faster by more than 19%. Thus, it is difficult to conclude firmly that an application of the Tapeworm algorithm improves the network balance, as the values of the standard deviations have always been lower for the faster network. In case of the MP3 codec there have additionally been carried out more experiments for various core mappings; thanks to then we have got almost 13% improvement comparing with the initial mapping. But even for this best mapping we have not obtained better communication time than for the XY-based network. From the experimental results we also conclude that the Tapeworm-routing-based network works faster if the FFC mechanism, described earlier in this paper, is disabled. Switching off this mechanism resulted in improving the results in case of iLBC by 4%, in case of MP3 by 32%. Only in case of the H.264 codec this mechanism has resulted in tiny improvement by 0.12%. However, for each network this mechanism has decreased the standard deviation, which indicates better transfer balancing in this case. We consider the time needed for implementing the virtual channels and contention as the reason for worse operation of the Tapeworm algorithm in comparison with XY.

5. References

[1] Andersen S.V. et al.: iLBC - a linear predictive coder, IEEE Workshop on Speech Coding, 2002.
[2] Bjerregaard T., Mahadevan S.: A Survey of Research and Practices of Network-on-Chip, ACM Computing Surveys (CSUR), vol. 38, 2006, Article 1.
[3] Brandenburg K.: MP3 and AAC Explained, 17th International Conference: High-Quality Audio Coding, 1999.
[4] Duato J., Yalamanchili S., Ni L.: Interconnection Networks. An Engineering Approach, Morgan Kaufmann Publishers, 2003.
[5] Dziurzanski P., Maka T.: Stream-based Multi-path Routing Scheme in On-chip Networks, 16th EUROMICRO Conference on Parallel, Distributed and Network-based Processing PDP, 13-15 February, Toulouse, France, ss. 15-16, 2008.
[6] Dziurzanski P., Maka T.: Stream Transfer Balancing Scheme Utilizing Multi-Path Routing in Networks on Chip, 4th International Workshop ARC 2008, 26-28 March, London, UK, ss. 294-299, 2008.
[7] Ford L.R., Jr. and Fulkerson D.R.: Flows in Networks. Princeton University Press, Princeton, NJ, 1962.
[8] Greenberg H. J.: Ford-Fulkerson Max Flow Labeling Algorithm, University of Colorado, at Denver, 1998.
[9] Kavaldjiev N. et al.: Routing of guaranteed throughput traffic in a network-on-chip, Technical Report TR CTIT-05-42 Centre for Telematics and Information Technology, University of Twente, Enschede, 2005.
[10] Li M., Zeng Q.A., Jone W. B.: DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. 43rd ACM IEEE Design Automation Conference (DAC), 2006, pp. 849-852.
[11] Richardson I. E. G.: H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, Wiley, 2003.
[12] Smit G. J. M., et al.: Efficient Architectures for Streaming DSP Applications, Dynamically Reconfigurable Architectures, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*