Jacek ZBYLUT, Tomasz MĄKA, Piotr DZIURZAŃSKI
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, SZCZECIN,
FACULTY OF COMPUTER SCIENCE & IT

NoC-based Realization of Multi-core Speech Encoders

Mgr inż. Jacek ZBYLUT

He received the MSc degree in computer science from Szczecin University of Technology in 2008. He is currently working as a programmer. His scientific interests include high-level programming and hardware realization of digital signal processing systems.

e-mail: jzbylt@wi.ps.pl

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 Systems, West Pomeranian University of Technology. His scientific interests include hardware-software co-synthesis, high level synthesis and formal verification.

e-mail: pdziurzanski@wi.ps.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 Systems, West Pomeranian University of Technology. His scientific interests include hardware realization of digital signal processing systems and acoustic signal processing techniques.

e-mail: tmaka@wi.ps.pl

Abstract

In this paper, we demonstrate a technique for mapping a multimedia streaming application into a mesh NoC using an example of speech encoder named SPEEX. To decrease the size of the target mesh, we use an algorithm for merging functional blocks using various metrics, such as core code size or execution time. We propose and test three algorithms for core mapping. According to the presented experimental results, the process of assigning the functional block into the NoC mesh is strongly influenced by the selected strategy.

Keywords: Network on Chip, core mapping, speech encoder.

Wielordzeniowa realizacja koderów mowy wykorzystująca sieć NoC

Streszczenie

W artykule zaprezentowano technikę odwzorowywania bloków realizujących algorytmy strumieniowe na strukturę mesh sieci NoC z wykorzystaniem przykładu – kodera mowy SPEEX. Aby zmniejszyć rozmiar docelowej sieci NoC, wykorzystano algorytm łączenie funkcjonalnych bloków wykorzystujących wybrane miary, takie jak rozmiar kodu lub czas wykonania. Dla optymalizacji sieci pod względem obciążeń czasowych oraz liczby instrukcji zawartych w poszczególnych blokach IP rozpatrywana jest sieci NoC o rozmiarach 6x6. Rozmiar omawianej struktury wynika z zestawienia kodera Speex o 4 różnych przepływnościach. Zaproponowano i przetestowano trzy algorytmy odwzorowujące rdzenie. Zaprezentowane algorytmy generują lokalnie najlepsze rozwiązania, dzięki wprowadzeniu funkcji heurystyki. Z punktu widzenia czasu realizacji zadań przez niezależne rdzenie, najmniejszy całkowity transfer uzyskano przy użyciu algorytmu drugiego. Z wykorzystaniem dodatkowego algorytmu balansującego uzyskano zmniejszenie standardowego odchylenia transferów na poziomie 20%. Otrzymane podczas badań wyniki dowodzą, że proces ustalenia odwzorowania bloków IP podczas projektowania sieci NoC jest niezwykle istotny. Efektywność i wydajność otrzymanego układu SoC może w dużej mierze zależeć od obranej strategii przydziału elementów funkcyjnych algorytmu DSP.

Słowa kluczowe: sieci wewnątrzukładowe, odwzorowanie rdzeni, koder mowy.

1. Introduction

Network on Chip (NoC) is a communication technique connecting cores inside a Multi Processor System on Chip (MPSoC) that offers high bandwidth and good concurrent communication capability [1]. A mesh is one of the most often used on-chip network topologies owing to its regularity and reliability caused with a large number of redundant interconnections between nodes [1]. In this architecture, each mesh node is comprised of the IP core realizing a particular stage of the algorithm and a router which is typically connected to four neighboring nodes. In mesh-based NoCs the most popular routing algorithm is XY where a flit (i.e. the smallest portion of data that can be sent atomically) 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 the flit is routed vertically. Since the mesh architecture and the XY routing algorithm are rather inherent to the popular NoC solutions, one of the most important problems for NoC-based chip designers is to propose a mapping scheme of IP cores into mesh nodes that decreases the contention level [2]. This issue is especially crucial in the case of multimedia streaming algorithm. In this paper, we focus on this issue and introduce a few mapping algorithms. They are then verified with a popular speech encoder [4].

2. A Speech Encoder Example

In this paper, the proposed mapping techniques are illustrated with the example of the SPEEX speech encoder. Below, some details on this encoder are provided [3, 4].

After analysis of the code of the SPEEX encoder, eight main modules can be singled out. These modules are: c_0 – Initialization, c_1 – Linear Prediction Coefficients (LPC) calculation, c_2 – Line Spectral Pair (LSP) calculation, c_3 – Analysis/Synthesis filters, c_4 – Long-Term Prediction calculation, c_5 – Overlapped codebook search, c_6 – Vector quantization, c_7 – LSP vector quantization, as presented in Fig. 1. The literal denotations introduced in this figure (i.e., labels c_0-c_7) are used in the sequel of this paper.

In the SPEEX encoder, each signal frame is composed of 160 samples and is divided into 4 subframes – 40 samples each. The flows underlined in the figure denote the stages that are executed for every subframe. As the algorithms presented latter in this paper require information about computation time of each module, we have measured these time for the encoder software realization run on a PC computer (Pentium IV 1.6 GHz). The obtained results are presented in Tab. 1; the table shows the time (in µs) needed for processing a single frame, i.e., 160 samples. Due to the encoder algorithm, these results vary for different subframes and the measurements presented in the table have been taken for the worst cases.

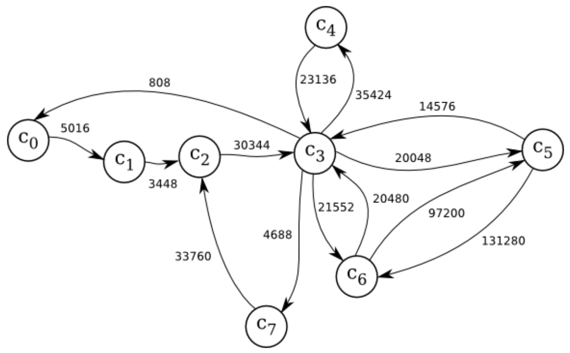

Fig. 1. Data flow graph of the SPEEX
Rys. 1. Diagram przepływu danych kodera SPEEX

Tab. 1. Computational time [μs] and code size [bytes] of NoC cores for various encoder configurations
Tab. 1. Czas obliczeń [μs] oraz rozmiar kodu [bajty] rdzeni NoC dla różnych konfiguracji kodera

Encoder flow	**NoC cores**							
	c_1	c_0	c_5	c_2	c_3	c_6	c_7	c_4
8 kbps	65	13	440	370	1116	160	150	720
11 kbps	65	13	1200	370	1116	560	150	720
15 kbps	65	13	440	370	1116	360	150	720
18.2 kbps	65	13	1164	370	1116	1280	150	720
Code size	5016	808	9872	11888	25720	11648	4688	8776

From the table it follows that the only differences for various encoder flows are observed in modules c_5 and c_6. This difference is caused by the various size of the code books that have to be browsed to find the appropriate vector representing the given voice sample. Consequently, between the sender and receiver the longer index (in bits) is to be sent.

3. The Encoder Mapping into a Mesh Structure

In order to illustrate the proposed mapping technique, we utilize an encoder with four different flows (enumerated in Tab. 1). The elements labeled with the numbers 0 to 35 in Fig. 2 denote the separate functional blocks of 4 subnetworks, each realizing the SPEEX encoding for a single data flow [4] (in every subnetwork eight cores realize the functionality of the c_0-c_7 blocks, the remaining block is a buffer). We decided to exclude from the encoder a number of additional features, such as silence detection, dynamic dataflow selection, echo cancellation and an additional signal filtering stage. Since our secondary goal was to synthesize the encoder in an FPGA chip, we translated the C-code into the SystemC code and removed from the SPEEX code all the SystemC non-synthesizable operations, such as dynamic pointers, variable and function references, dynamic memory allocation etc.

We analyze an initial mesh NoC of the 6x6 size, where the cores realizing four various transfers are to be placed (Fig. 2). However, the goal is to place all the 36 functional blocks into a NoC of size 3x3. The criteria for the functional block placement are: its computation time and the core code size. Following these assumptions, the number of possible core mappings can be denoted as a Stirling number of a second kind $\{n/k\}$. This number informs about the number of possibilities of splitting a n-element set into k non-empty sets, which in our case can be described as $\{36/9\}$. This is a set of all possible solutions, unable to be analyzed in a reasonable time exactly.

In our research we have made the following assumption: the initial NoC is comprised of four subnetworks labeled with number 0-3. Each of these subnetworks realizes the encoder with different bit flow.

The parameters that we have taken into account in the core mapping problem is the code size of the separate modules and their computational time on a PC computer. The modules from the 1-3 subnetworks are to be matched with the modules of the 0th subnetwork. The choice of the element to be matched with the selected module form the 0th subnetwork is performed based on the specific metric, denoting its feasibility with respect to the given target module in the 0th subnetwork. The basis of the metric computation for the separate cores is the calculation of the average value from the particular parameter of all the blocks. This average is treated as the value that is to be reached while the new elements are merged with the 0th subnetwork. The aim of the algorithm is to obtain such the block placement that the standard deviation of the selected parameter is minimized. Thus, for each block from the 0th subnetwork the algorithm tries to realize the formula A=min($|S(c_i)$-$avg|$), where $S(c_i)$ is the cumulated value of the analyzed parameter for block c_i, and avg is the average value of this parameter for the whole NoC. Each of the 9 values of every element from subnetworks 1-3 is computed using the formula $h_{ij}=S(c_i)-avg+c_j$, where h_{ij} denotes the fitness of the j-th node after being merged with the i-th one. This formula indicates how node c_j complements $S(c_i)$ to the average value, avg. The algorithm iterates through all the blocks, computes the value of the h metric and selects the most promising solution. The minimal value of h_{xi} denotes a better fitness of the c_x element.

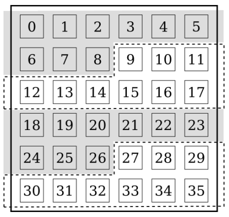

Fig. 2. Example of a NoC realizing SPEEX encoding with four various data flows
Rys. 2. Przykład sieci NoC realizującej kodowanie SPEEX z czterema różnymi przepływnościami danych

The next stage is to choose the algorithm for the selection of the block that is to be placed in the 0th network. After this selection, the process of searching the whole 6x6 NoC network is performed for the selection of the node that is to be removed from this network and moved to the appropriate place in the new 3x3 network. Having selected the node, it is moved to the target position and the actualization of the block parameters is performed. Then, the next iteration of the node choice takes place. When all the nodes are assigned to the new places, the next stage of our approach is carried out.

At this moment it is possible to perform an optional step of the proposed approach: to find minimal elements included in the individual blocks and moving them into the remaining blocks of the 0th subnetwork in order to additionally balance the parameter distribution inside a network. Elements are added in the places where the value of the considered parameter is below the average value, avg. Then, the process of the transfer mapping into the block structure is to be carried out. After determining all the paths for the transmitted data, the total amount of the transferred data is summarized for each router, its ports and the links connecting the routers. There is a parameter of the balancing algorithm, α, which is the percentage of the average code size. The algorithm is not allowed to move blocks from the j-th node if the code size to be removed exceeds α per cent of the whole code size and it cannot add any additional elements to the i-th block if the total code size, after adding the last element of the j-th block, exceeds by α percent the total code size.

The first mapping algorithm of functional blocks into the NoC mesh nodes is based on the analysis of each block from the 0th subnetwork and to select for this block the most feasible (using the h metric) element from the remaining subnetworks. After a given block is selected, it is removed from the pool of the block to be assigned. The algorithm is run in a loop for all the blocks until the block pool is empty. In the second mapping algorithm, the searching problem is performed for the subsequent block from the

0th subnetwork. Consequently, the last blocks from this subnetwork are associated with the elements with worse values of the *h* metric. In order to avoid this situation, every second iteration the elements are analyzed in the reverse order. The third mapping algorithm is based on a global searching of the 0th subnetwork (of the 3x3 size) and selection of the element that is characterized with the best *h* value from all 9 blocks. The block selection is not influenced with the currently analyzed element from the 0th network as the globally most suitable element is chosen for the arbitrary 0th subnetwork node.

4. Experimental results

In the first experiment, we determined the computation time for each block for the four analyzed variants of the SPEEX encoder and then used the described earlier algorithms for core mapping. In Tab. 2, the total amount of transferred data and the standard deviation of the transferred data by all the nodes are presented.

Tab. 2. Total amount (T) and standard deviation (SD) of transferred data (the computation time parameter)
Tab. 2. Łączna liczba (T) i odchylenie standardowe (SD) transmitowanych danych (parametr - czas obliczeń)

	Algorithm I	Algorithm II	Algorithm III
T	2712320 B	2379056 B	2684176 B
SD	51.2091 µs	59.6395 µs	80.448 µs

The lowest amount of transferred data are obtained with algorithm II. According to the standard deviation, however, algorithm I leads to the most balanced transfers on all the network nodes. Algorithm III is the least favorable in both criteria of the total amount and the standard deviation of the transferred data.

The next analyzed parameter is the even distribution of the amount of code (in bytes) inside each of the 9 blocks. The obtained results are presented in Tab. 3.

Tab. 3. Total amount (T) and standard deviation (SD) of transferred data (the amount of code parameter)
Tab. 3. Łączna liczba (T) i odchylenie standardowe (SD) transmitowanych danych (parametr - rozmiar kodu)

	Algorithm I	Algorithm II	Algorithm III
T	3210160 B	2603488 B	3854784 B
SD	1978.12 B	1421.11 B	2049.91 B

The lowest value of the standard deviation is obtained with algorithm II. (The average amount of code to be placed in each node is equal to 36858 bytes.) This algorithm leads to the best results also with respect to the second criteria, i.e., the total amount of transferred data. The relatively high value of the standard deviation for algorithm I means that the differences of code size implemented in the cores are rather large and vary from 8 to 5177 bytes.

Taking into account the results presented in Tab. 2 and 3 one may conclude that the more even balance of code amount implemented in the cores leads to the increase of the transfers in the network.

In Tab. 4, the results of the influence of execution time balancing on parameters of the NoC are provided. Having set the parameter α to 5% improves the obtained parameters. The highest improvement is observed in the case of Algorithm I, where the standard deviation decreases by 563 bytes (27%). For the remaining algorithm (II and III), the standard deviation decreases by 9%. The amount of transferred data decreases slightly (1%). If parameter α is set to 25%, Algorithm III leads to the decrease of the standard deviation by 23%, whereas for the remaining algorithm the obtained values are similar to the ones obtained with the previous parameter set.

Tab. 4. The influence of balancing algorithm (in bytes) on standard deviation (SD) and Total transfer (T)
Tab. 4. Wpływ algorytmu balansowania (w bajtach) na odchylenie standardowe (SD) i łączny transfer (T)

Algorithm	Parameter	Without balancing	Balancing	
			α=5%	α=25%
I	SD	1978.12	1454.85	1454.85
	T	3210160	3204336	3204336
II	SD	1421.11	1272.93	1272.93
	T	2603488	2599280	2599280
III	SD	2049.91	1869	1588.95
	T	3874784	3854784	3873232

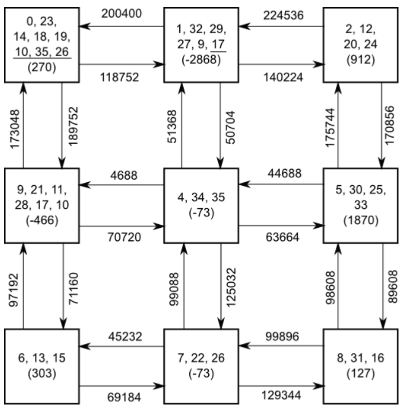

Fig. 3. Decreasing of standard deviation for Algorithm I
Rys. 3. Zmniejszenie odchylenia standardowego dla Algorytmu I

An example of core mapping with the additional process of balancing the code size in nodes is presented in Fig. 3. The numbers in gray denotes the blocks that have been moved to another block; the target place is underlined. The standard deviation of the code size in nodes has been decreased from 1978,12 to 1454,85. The relative deviation has substantially decreased in block 5 (from -139 to 73), 7 (from -983 to -134), and 8 (from 5117 to 3661) (in bytes).

5. Conclusion

In this paper, we presented a technique for mapping SPEEX encoder into a mesh NoC. To decrease the size of the target mesh, we used an algorithm for merging functional blocks using various metrics, such as core size or execution time. We proposed and tested three algorithms for core mapping. For the execution time criteria of separate cores the lowest total transfer value were obtained with the Algorithm II. Similarly, the same algorithm leaded to the lowest transfer when the amount of code of each core was taken into account. Additionally, we used a balancing scheme which decreased the standard deviation by more than 20 per cent.

6. References

[1] Bjerregaard T., Mahadevan S.: A Survey of Research and Practices of Network-on-Chip, ACM Computing Surveys (CSUR), vol. 38, 2006, Article 1.
[2] Hansson A., Goossens K., Rădulescu A.I.: A Unified Approach to Mapping and Routing on a Network-on-Chip for Both Best-Effort and Guaranteed Service Traffic, VLSI Design, vol. 2007.
[3] Herlein G., Valin J.-M., Morlat S., Hardiman R., and Kerr P.: RTP Payload Format for the Speex Codec, Internet Engineering Task Force, Internet-Draft draft-ietf-avt-rtp-speex-05, February 2008.
[4] Valin J.-M.: The Speex Codec Manual Version 1.2 Beta 2, 2007.

Artykuł recenzowany