An ILP-based DMA Data Transmission Optimization Algorithm for MPSoC

Yingbiao Yao
Hangzhou Dianzi University/School of Communication Engineering, Hangzhou, China
Email: yaoyb@hdu.edu.cn

Guangpei Zhao, Xuan Wang
Hangzhou Dianzi University/School of Communication Engineering, Hangzhou, China
Email: {664395483, 738915687}@qq.com

Abstract—**With the rapid development of integrated circuit design technology and the processed tasks and data volumes growing, MPSoC is becoming increasingly popular in a variety of applications. In MPSoC design, parallelism is a very important issue, for example, how to realize task parallelism and data parallelism. Focusing on this issue, this paper analyzes the role of DMA and presents an ILP-Based DMA data transmission optimization algorithm to reduce the pipeline time when employing multi-stage pipeline scheduling method to solve task parallelism and data parallelism. The proposed ILP model integrates task allocation/schedule and data transmission and thus realizes the optimal parallelism of data transmission and data processing. In addition, we divide data transmission of ILP model into four cases: (1) DMA0, do not use DMA to optimize data transmission; (2) DMA1, use DMA to transmit data between SPM and off-chip memory; (3) DMA2, use DMA to transmit data between SPM and SPM, SPM and off-chip memory; (4) DMA3, use DMA to transmit and prefetch all data. Simulation results show that the ILP model with DMA3 can reduce the pipeline time 17.8% compared with that of the ILP model with DMA0.**

Index Terms—**MPSoC, ILP, DMA, Data Processing and Transmission, Parallelism**

I. Introduction

With the rapid development of integrated circuit design technology and the processed tasks and data volumes growing, MPSoC (Multi-Processor System on Chip) [1] is becoming increasingly popular in a variety of applications. In MPSoC design, parallelism is a very important issue, for example, how to realize task parallelism and data parallelism [2, 12].

The task parallelism is mainly to solve the task allocation and schedule problem in MPSoC [19]. To this problem, scholars have carried out extensive researches in the past several decades, and their algorithms can be divided into two classes: heuristics and deterministic analysis algorithm. Heuristic algorithm can gradually find near-optimal solution according to heuristic information dynamically adjusting the allocation and schedule plan, which is suitable for the large-scale task schedule problem [3, 4]. For example, the genetic algorithm is the typical representative of heuristics algorithm [5]. In contrast, deterministic analysis algorithm can find the optimal solution according to the problem model at the cost of complexity, which is suitable for small tasks scheduling problem. The ILP-based (Integer Linear Programming) algorithm is the typical representative of deterministic analysis algorithm [6, 8].

For data parallelism, due to its data transmission without CPU resources, DMA (Direct Memory Access) is widely used in various kinds of data parallel problem [7]. DMA can realize the concurrent execution of data transmission and data processing and is becoming more and more important for performance optimization in MPSoC [8, 9]. In [10], the DMA is used for the data transmission between SPM (Scratch pad Memory) and off-chip memory. In [11], a DMA-based SPDP (Scratch Pad Data Pipelining) technique is proposed and DMA is also used only for the data transmission between SPM and off-chip memory.

In order to further improve system performance, MPSoC designers begins to consider task parallelism and data parallelism at the same time. In [6], an ILP-based data parallelism and multi-task mapping/scheduling technique are proposed for heterogeneous MPSoC with the known task graph input. Yi Wang, *et al*, propose a technique for removing inter-core communication overhead of streaming applications in MPSoC [13, 14]. Task parallelism and data parallelism are considered at the same time during multi-task assignment and schedule in [15]. Software pipelining parallelization method is proposed for media data processing on MPSoC in [16].

In this paper, focusing on the parallelism problem of data transmission, data processing and task allocation/ schedule in MPSoC design for multimedia applications, we propose an ILP-Based DMA data transmission optimization algorithm to reduce the cycle time when employing multi-stage pipeline schedule method to solve task parallelism and data parallelism. The proposed ILP model integrates task allocation/schedule and data transmission and thus realizes the optimal parallelism of data transmission and data processing. In addition, we divide data transmission of ILP model into four cases: (1) DMA0, do not use DMA to optimize data transmission;

(2) DMA1, use DMA to transmit data between SPM and off-chip memory; (3) DMA2, use DMA to transmit data between SPM and SPM, SPM and off-chip memory; (4) DMA3, use DMA to transmit and prefetch all data. Simulation results show that the ILP model with DMA3 can reduce the pipeline time 17.8% compared with that of the ILP model with DMA0.

The remainder of this paper is organized as follows. In Section II, we present our system models, including the MPSoC structure and task flow graph. In Section III, we firstly introduce the ideas of our method by an example and then give the ILP models for DMA data transmission optimization in MPSoC design. The experiment results are shown in Section IV and we conclude the paper in Section V.

II. System Model

A. The Structure of MPSoC

In this paper, the structure of embedded MPSoC is shown in fig.1. It includes processor cores, local memory and DMA&EMI modules. Local memory module is composed of SPM, which is a kind of high-speed on-chip memory with SRAM structure [17]. The core module is consisted of multiple homogeneous RISC processor cores and each core has a private SPM. Each core can access SPM of other cores through DMA. DMA&EMI module is responsible for the communication between the cores and off-chip memory.

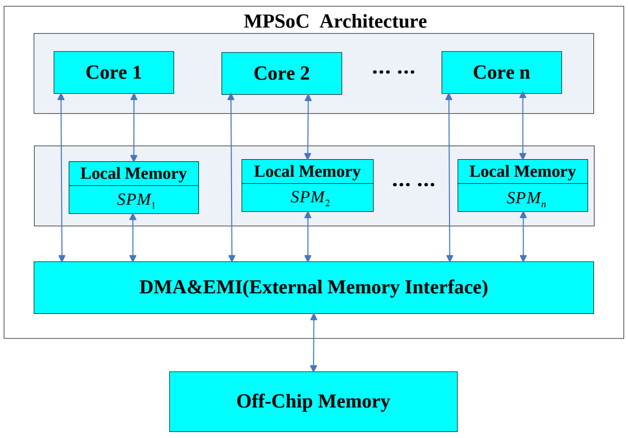

Figure 1. Embedded MPSoC structure.

For MPSoC shown in fig.1, the DMA can optimize the system performance from three aspects: 1) using DMA to move data between on-chip SPM and off-chip memory; 2) using DMA to move data from SPM of one core to SPM of another core; 3) according to the characteristic of streaming data processing in media application, using DMA to prefetch data to SPM which requires these data further processing. The first aspect is studied more in existing literature and the second and third aspects are few studied. Therefore, we carried on detailed and comprehensive study on DMA role and proposed an ILP-based DMA transmission optimization algorithm for MPSoC design, which integrates the above three DMA optimization methods and task allocation & schedule.

B. The Model of Task Flow Graph

We assume that the coarse-grained TFG (Task Flow Graph) of media applications is known and shown as fig.2. Fig.2 is a directed acyclic TFG and can be expressed as *TFG*=<*T*, *TE*>. *T* is the node sets and represents the task, for example, T_1 to T_5 in fig.2; *TE* is the edge sets and represents the data flow relationship between tasks, for example, d_{13}, d_{23}, and *etc* in fig.2. We use the compiler and simulator to collect its specific information, which mainly include: ① execution time of each task; ② the data transmission time between tasks. In this paper, we only consider the on-chip data memory, and assume that all instructions are assigned into on-chip memory.

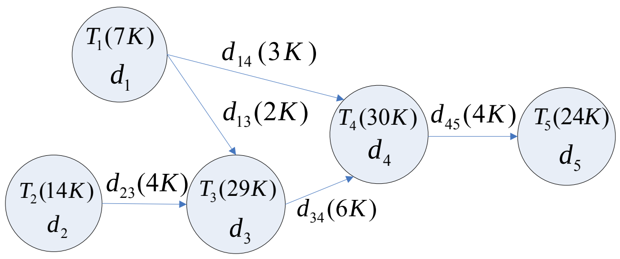

Figure 2. Directed acyclic TFG.

III. ILP-Based DMA Datatransmission Optimization Algorithm

Because the core number of embedded MPSoC is not too much in current actual embedded application, we do not need to divide coarse-grained TFG into fine-grained TFG. Therefore, we use ILP-based deterministic analysis method to solve task allocation & schedule and DMA data transmission optimization.

A. An Example

We use task flow graph shown in fig.2 and MPSoC with 4 homogeneous cores as an example to introduce the ideas of our method. For coarse-grained task flow graph in fig.2, using ILP modeling method whose objective function is to minimize initiation internal time and constraint conditions are control and data dependencies between tasks, we can get a task allocation and schedule results as shown in fig.3 under the condition of without DMA transfer optimization. In fig.3, T_i is executing time of task *i*, C_{ij} is data transmission time between task *i* and *j*, C_{ij}^k means that data transmission between task *i* and *j* is performed by the core P_k, and the initiation internal time is 35k.

In fig.3, the total time for data transmission of inter-core is 3+2+4+6+4=19k and the longest time for data processing is 30k. However, the initiation internal time of pipeline is 35k. The reason is that the data transmission time of core P_2 and P_3 (or C_{34}) is 6k and P_3 starts its data processing until finishing C_{34}. But in the actual media processing, the data is flow processing. We can start DMA after partial data are calculated. By this way, we can realize data processing and data transmission of inter-core in parallel. For example, C_{34} in fig.3 can be divided into two transmissions $C_{34\text{-}1}$ and $C_{34\text{-}2}$ as shown in fig.4. After this optimization, all data transmission of inter-core is hidden by DMA, thus the initiation internal time of pipeline is 30k which is the longest time for data processing. Therefore, the performance of fig.4 is upgraded 16.7% than that of fig.3.

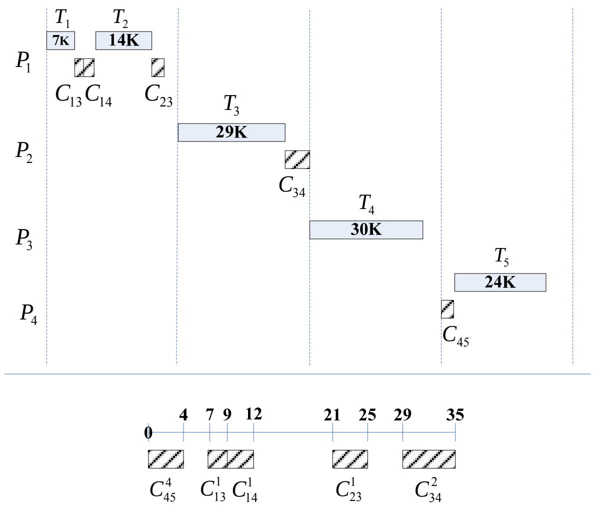

Figure 3. Task allocation and scheduling before DMA optimization.

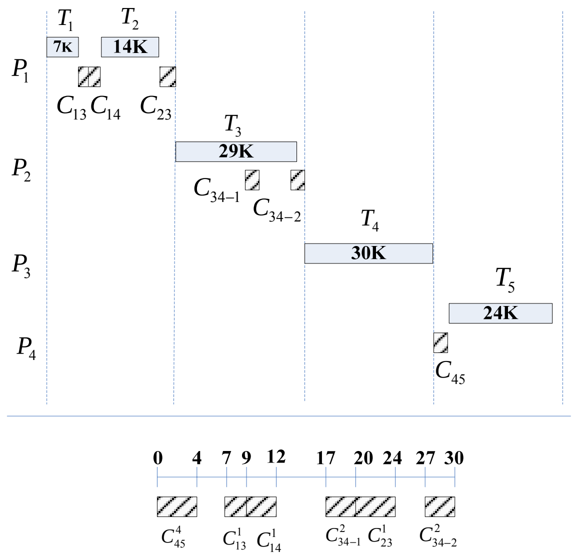

Figure 4. Task allocation and scheduling after DMA optimization.

B. The Objective Function

Assuming that the media application is made up of N tasks (T_1, T_2, ⋯, T_N) and is executed on MPSoC with M processor cores (P_1, P_2, ⋯, P_M). We employ multi-stage pipeline scheduling method to solve task parallelism and data parallelism. The cycle time of pipeline, which also is called as initiation internal (II) time, is decided by the maximum running time in each pipeline. In addition, we assume that the pipeline length is equal to the number of processors. The objective function of our ILP model is as follows:

$$Min\{II\} = Min(Max\{PT_K\}) \quad k = 1,2,\cdots,M \qquad (1)$$

In (1), PT_k represents the time produced by processor P_k to finish the tasks and data transmission, and it can be expressed by the following expression:

$$PT_k = \sum_{T_j \in P_k} Time_j + \sum_{i \neq k} CommTime_{i,k} \qquad (2)$$
$$i, k = 1,2,\ldots,M;\ j = 1,2,\ldots,N$$

$Time_j$ is the processing time of task T_j which is assigned into P_k and $CommTime_{i,k}$ is the data communication time between P_i and P_k.

C. Modeling Data Communication of Inter-Cores

$CommTime_{i,k}$ in (2) can be expressed as the sum of data communication time of tasks:

$$CommTime_{i,k} = \sum_{T_m \in P_i, T_n \in P_k} CTime_{m,n} \qquad (3)$$
$$i, k = 1,2,\ldots,M;\ m, n = 1,2,\ldots,N$$

In (3), $CTime_{m,n}$ is data communication time between task T_m and T_n. If task T_m and T_n have no data dependency, or they are assigned into the same processor, $CTime_{m,n}$ is 0. In this paper, $CTime_{m,n}$ can be expressed as following four models:

(1) Do not use DMA to optimize data transmission (Briefly as DMA0)

In this case, we assume that MPSoC has no DMA and each core cannot directly access the private SPM of other cores. $CTime_{m,n}$ can be expressed as following:

$$CTime0_{m,n} = \sum_{DS \in T_{m \to n}} \{(2 \times DS_{Spm} + DS_{OM}) \times Delay_{OM}\} \qquad (4)$$

In (4), $Dealy_{OM}$ is access delay time of off-chip memory and DS represents the communication data of task T_m and T_n which includes DS_{SPM} and DS_{OM}. DS_{SPM} represents the data in SPM, DS_{OM} represents the data in off-chip memory and $DS = DS_{SPM} + DS_{OM}$. Since each core cannot directly access the private SPM of other cores, DS_{SPM} must firstly be saved to off-memory in T_m and then be loaded into the private SPM in T_n. Therefore, DS_{SPM} need to access twice off-chip memory and DS_{OM} need to access once off-chip memory. Here, we assume that $Dealy_{OM}$ =50.

(2) Using DMA to transmit data between SPM and off-chip memory (Briefly as DMA1)

In this case, we assume that MPSoC has DMA and DMA is only used for data transmission between SPM and off-chip memory. $CTime_{m,n}$ can be expressed as following:

$$CTime1_{m,n} = \sum_{DS \in T_{m \to n}} \{2 \times DMA_{init} + (2 \times DS_{spm} + DS_{OM}) \times Delay_{dma}\} \quad (5)$$

DMA_{init} represents the time of initializing DMA and $Delay_{dma}$ represents the time of transmitting a word by DMA. Since DMA is only used to transmit data between SPM and off-chip memory, we need initialize DMA twice: one is to transmit DS_{SPM} to off-chip memory, and the other is to transmit all data, DS, to private SPM of another core. Here, we assume that DMA_{init} =100 and $Delay_{dma}$=10.

(3) Using DMA to transmit all data (Briefly as DMA2)

In this case, DMA is used for all data transmission including on-chip & on-chip data transmission and on-chip & off-chip data transmission. Therefore, DS_{SPM} can be directly transmitted to another SPM by DMA and $CTime_{m,n}$ can be expressed as following:

$$CTime2_{m,n} = \sum_{DS \in T_{m \to n}} \{2 \times DMA_{init} + DS \times Delay_{dma}\} \quad (6)$$

(4) Using DMA to transmit and prefetch data (Briefly as DMA3)

In this case, DMA is not only used for all data transmission but also used for data prefetching. Data prefetching technique is a very useful technique for shortening the data communication time between cores, for example, $C_{34\text{-}1}$ and $C_{34\text{-}2}$ in fig.4 are data prefetching. In order to model the situation of data prefetching, we introduce a variable P to represent the ratio of DS whose transmission time can be hidden by DMA prefetching. Because data prefetching can realize the perfect parallelism between data transfer and data processing, the remaining data, which may affect pipeline time, are $(1-P)*DS$. $CTime_{m,n}$ can be expressed as following:

$$CTime3_{m,n} = \sum_{DS \in T_{m \to n}} \{2 \times DMA_{init} + (1-P) \times DS \times Delay_{dma}\} \quad (7)$$

Obviously, P=0 means that there has no data prefetching, and P=1 means that all data can be prefetched. In practical applications, P values should be in [0, 1].

D. Constraints

In order to correctly perform a task and transmit data, we also must ensure the dependency of tasks.

Firstly, for any pairs of task T_m and T_n, which have data dependency, we must ensure the following constraint condition:

$$StartComm_{m,n} \geq EndTask_m \quad (8)$$

$$EndComm_{m,n} \geq StartComm_{m,n} + X_{m,n} * CTime_{m,n} \quad (9)$$

$$StartTask_n \geq EndComm_{m,n} \quad (10)$$

$StartComm_{m,n}$ and $EndComm_{m,n}$ is the start time and end time of data transmission between task T_m and T_n; $StartTask_n$ is the start time of task T_n and $EndTask_m$ is the end time of task T_m; $X_{m,n}$ is an indicative variable which equals 1 when task T_m and T_n are assigned into different processors and equals 0 when task T_m and T_n are assigned into the same processor.

Secondly, we need to ensure that any pairs of tasks which are assigned to the same processor cannot overlap. We can use the following constraint condition:

$$(StartTask_n - EndTask_m) \times (StartTask_m - EndTask_n) \leq 0 \quad (11)$$

Thirdly, we need to ensure that any pairs of data transmission cannot overlap. We can use the following constraint condition:

$$(StartComm_{m,n} - EndComm_{m',n'}) \times (StartComm_{m',n'} - EndComm_{m,n}) \leq 0 \quad (12)$$

At last, for any task T_m, its end time and its start time must satisfy the following constraint condition (13) and all variables must be non-negative.

$$EndTask_m \geq StartTask_m + Time_m \quad (13)$$

$$StartTask_m, EndTask_m \geq 0 \quad (14)$$

$$EndComm_{m,n}, StartComm_{m,n} \geq 0 \quad (15)$$

IV. Experimental Results

A. Test Programs

We have chose three test programs, which are lame (Version 3.70), cjpeg (Version 6a), mpeg2enc (Version 1.2), to evaluate our algorithm. Program lame is chose from an embedded benchmark set called MiBench [18] and it is a MP3 encoding program. Program cjpeg and mpeg2enc are chose from multimedia benchmark set called MediaBench [19]. Cjpeg is JPEG encoding program and mpeg2enc is MPEG-2 video coding program. These three test programs are practical multimedia applications and their characteristic information are shown in table 1. In addition, we have modified SimpleScalar3.0 simulator to support a homogeneous dual-core MPSoC which is our objective MPSoC.

Table
The Characteristic Information Of Multimedia Applications

Multimedia applications	lame	cjpeg	mpeg2enc
Coarse-grained task number	5	4	6
Original input data size (Bytes)	26504	101970	380160

B. Experimental Results

The initiation internals of pipeline with different DMA modes are shown in fig.5. In fig.5, DMA0 indicates that DMA is not used in MPSoC, DMA1 represents that DMA is only used for data transmission between on-chip and off-chip memory, DMA2 adds inter-core DMA data transmission (on-chip and on-chip memory) on the basis of DMA1, DMA3 adds DMA data prefetching on the basis of DMA2 and we assume that $P = 0.8$.

From fig.5 we can conclude that, after using the DMA optimization technique, the initiation internals of lame, cjpeg and mpeg2enc can be decreased for all SPM size. In fig.5, SPM size is the total SPM capacity of MPSoC and the private SPM of each core has the same size. DMA1 has an average 9% performance improvement compared with DMA0, DMA2 has an average 2% performance improvement compared with DMA1, DMA3 has an average 6% performance improvement compared with DMA2, and the total performance improvement of DMA3 is about 17.8% compared with DMA0. Therefore, it is very suitable for multimedia applications to employ DMA to optimize inter-core data transmission in MPSoC.

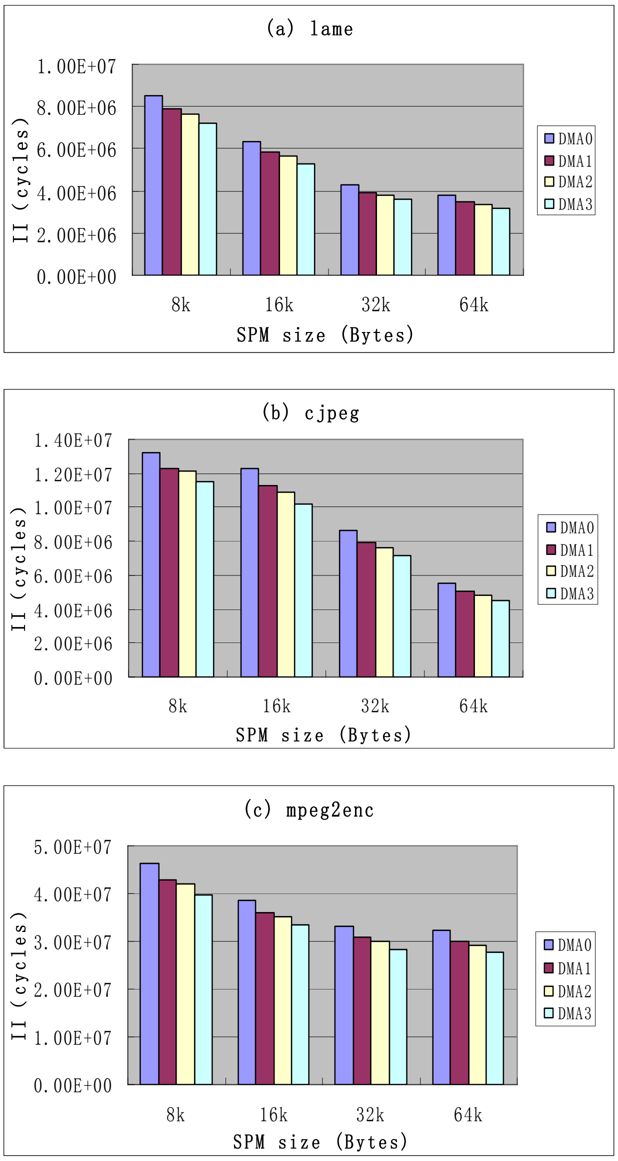

Figure 5. The simulation results with different DMA mode

In addition, we use cjpeg program as an example to test the performance of our algorithm under different input data size. After using DMA to optimize inter-core data transmission, the performance improvement ratio of cjpeg benchmark under different input data size is shown in fig.6 compared with DMA0 which does not use DMA. From fig.6, we can find the performance of all DMA optimization modes for cjpeg benchmark can improve with the increase of input data size, and with the increase of input data, the performance improvement ratio is slow down. Therefore, in a fixed application, by optimizing data transmission to optimize the performance of the system, there will be an upper limit values. Similarly, for the lame and mpeg2enc test bench, the results are similar to cjpeg.

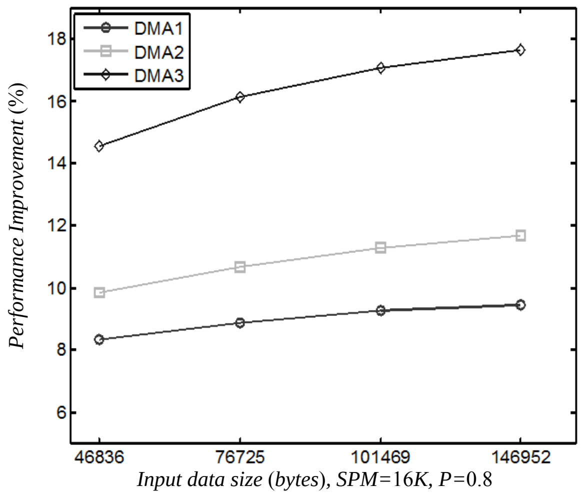

Figure 6. The performance improvement with different input data size.

V. CONCLUSION

For the parallelism problem of data processing and data transmission in MPSoC, we propose an ILP-based optimization algorithm, which integrates task assignment & schedule and DMA data transmission optimization. The experimental results show that DMA1 has an average 9% performance improvement compared with DMA0, DMA2 has an average 2% performance improvement compared with DMA1, DMA3 has an average 6% performance improvement compared with DMA2, and the total performance improvement of DMA3 is about 17.8% compared with DMA0. Therefore, the algorithm is very suitable for MPSoC with multimedia application. The future work is combination of DMA data prefetching technique and on-chip and off-chip data allocation algorithm, to further optimize the performance of our proposed algorithm.

ACKNOWLEDGMENT

This work was supported in part by a grant from the National Natural Science Fund No. 61100044 (China).

REFERENCES

[1] Wolf W, Atlanta GA, Jerraya A.A, Martin G, “Multi-processor System-on-Chip (MPSoC) Technology,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 10, pp. 1701-1713, 2008.

[2] A. Jerraya, H. Tenhunen, and W. Wolf, “Introduction: Multiprocessor Systems-on-Chips,” *Computer*, vol. 38, no. 7, pp. 36-40, July 2005.

[3] G. Wang et al., “Design space exploration using time and resource duality with the ant colony optimization,” *in Proceedings of the 43rd Annual ACM/IEEE Design Automation Conference*, New York: ACM Press, pp. 451-454, 2006.

[4] M. Damavandpeyma et al., “Hybrid Code-Data Prefetch-Aware Multiprocessor Task Graph Scheduling,” *in Proceedings of 14th Euro micro Conference on Digital System Design, Washington D C: IEEE Computer Society Press*, pp. 583 -590, 2011.

[5] Reakook Hwang, Mitsuo Genb, Hiroshi Katayama, “A comparison of multiprocessor task scheduling algorithms with communication costs,” *Computers & Operations Research*, Vol.35, No. 2, pp. 976 – 993, 2008.

[6] Hoeseok Yang, Soonhoi Ha, “ILP based data parallel multi-task mapping/scheduling technique for MPSoC,” *in Proceedings of International SoC Design Conference, Washington D C: IEEE Computer Society Press*, pp. 134-137, 2008.

[7] Selma Saidi, Pranav Tendulkar, Thierry Lepley, Oded Maler, “Optimal 2D Data Partitioning for DMA Transfers on MPSoCs,” *2012 15th Euromicro Conference on Digital System Design*, pp. 584-591, 2012.

[8] Suhendra V, Raghavan C, Mitra T, “Integrated scratchpad memory optimization and task scheduling for MPSoC architectures,” *in Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems, New York: ACM Press,* pp. 401-409, 2006.

[9] S. Udayakumaran, A. Dominguez, R. Barua, “Dynamic allocation for scratch-pad memory using compile-time decisions,” *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 5, No. 2, pp. 472–511, 2006.

[10] L. Li, L. Gao, J. Xue, “Memory coloring: A compiler approach for scratchpad memory management,” *In Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT), Washington D C: IEEE Computer Society Press*, pp. 329–338, 2005.

[11] Yanqin Y, Meng W, Zili S, and Minyi G, “Dynamic scratch-pad memory management with data pipelining for embedded systems,” *2009 International Conference on Computational Science and Engineering. Washington D C: IEEE Computer Society Press*, pp. 358-365, 2009.

[12] Peng Zhao, Dawei Wang, Ming Yan, Sikun Li, “Parallel Processing of Sequential Media Algorithms on Heterogeneous Multi-Processor System-on-Chip”, *Journal of Computers*, Vol. 4, No. 6, pp. 477-484, June, 2013.

[13] Yi Wang, Duo Liu, Meng Wang, Zhiwei Qin, Zili Shao, “Optimal Task Scheduling by Removing Inter-Core Communication Overhead for Streaming Applications on MPSoC,” *16th IEEE Real-time and Embedded Technology and Applications Symposium*, pp. 195-204, 2010.

[14] Yi Wang, Zili Shao, Henry C. B. Chan, et al, “Memory-aware task scheduling with communication overhead minimization for streaming applications on bus-based multiprocessor system-on-chips,” *IEEE Transactions On Parallel And Distributed Systems*, No. 99, pp. 1-11, 2013.

[15] C. Q. Xu, C. J. Xue, B. C. Hu, and E. H. M. Sha, “Computation and data transfer co-scheduling for interconnection bus minimization,” *in Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC ’09)* , pp. 311–316, 2009.

[16] Hoeseok Yang, Soonhoi Ha, “Pipelined Data Parallel Task Mapping/Scheduling Technique for MPSoC,” *Design, Automation & Test in Europe Conference & Exhibition (DATE). Washington D C: IEEE Computer Society Press*, pp.69-74, 2009.

[17] R. Banakar et al., “Scratchpad memory: Design alternative for Cache On-chip memory in embedded systems,” *in Proceedings of the 10th International Symposium on Hardware/Software Codesign (CODES), New York: ACM Press*, pp. 73–78, 2002.

[18] Guthaus M.R, Michigan Univ, Ann Arbor, et al, “MiBench: A free, commercially representative embedded benchmark suite,” *2001 IEEE International Workshop on Workload Characterization*, pp. 3-14, 2001.

[19] Chunho Lee, Miodrag Potkonjak, William H. Mangione-Smith, “MediaBench: a tool for evaluating and synthesizing multimedia and communication systems,” *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture, Washington, DC, USA*, pp. 330-335, 1997.

[20] Honglei Han, Wenju Liu, Wu Jigang, Guiyuan Jiang, “Efficient Algorithm for Hardware/Software Partitioning and Scheduling on MPSoC”, *Journal of Computers*, Vol. 8, No. 1, pp. 61-68, January, 2013.