논문 2007-44SD-10-2

프로세스 네트워크 모델의 정적 분석에 기반을 둔 다중 프로세서 시스템 온 칩 설계 공간 탐색

(MPSoC Design Space Exploration Based on Static Analysis of Process Network Model)

안 용 진*, 최 기 영*

(Yongjin Ahn and Kiyoung Choi)

요 약

본 논문에서는 다중프로세서 시스템 온 칩 설계를 효율적으로 하기 위한 한 설계 방법론 및 환경을 제시한다. 본 논문에서 제시하는 설계 환경은 SystemC로 작성된 프로세스 네트워크 모델을 입력으로 한다. 프로세스 네트워크 모델은 뛰어난 모델링 파워를 가지고 있지만 정적 분석이 불가능하기 때문에 시스템의 성능을 미리 예측하기가 힘들다는 단점이 있어서 실시간 시스템을 설계할 때 심각한 문제를 발생할 수도 있다. 따라서 본 논문에서는 이를 보완하기 위해서 주어진 프로세스 네트워크 모델을 자동으로 정적 분석이 가능한 모델로 바꾸는 방법을 제시한다. 또한, 설계 과정에서 초기에 효율적인 설계 공간 탐색을 위해서는 애플리케이션을 어떻게 타켓 아키텍처에 잘 매핑할 지 결정하는 문제가 아주 중요하다고 할 수 있다. 따라서 본 논문에서는 효율적인 매핑을 할 수 있도록 하는 알고리즘을 제시한다. 매핑 과정에서 정적 스케줄링 방법을 사용하여 시스템의 성능을 예측하게 되는데 본 논문에서 제시하는 알고리즘은 단일 버스 구조뿐만 아니라 다중 버스 구조에서도 성능 예측이 가능하도록 한다. 실험에서는 본 논문에서 제시한 방법으로 여러 멀티미디어 예제를 가지고 그들의 프로세스 네트워크 모델들이 성공적으로 정적 분석이 가능한 모델로 자동 변환됨을 보이고 이전 연구들과 비교하여 매핑 알고리즘의 효율성을 보인다.

Abstract

In this paper, we introduce a new design environment for efficient multiprocessor system-on-chip design space exploration. The design environment takes a process network model as input system specification. The process network model has been widely used for modeling signal processing applications because of its excellent modeling power. However, it has limitation in predictability, which could cause severe problem for real time systems. This paper proposes a new approach that enables static analysis of a process network model by converting it to a hierarchical synchronous dataflow model. For efficient design space exploration in the early design step, mapping application to target architectures has been a crucial part for finding better solution. In this paper, we propose an efficient mapping algorithm. Our mapping algorithm supports both single bus architecture and multiple bus architecture. In the experiments, we show that the automatic conversion approach of the process network model for static analysis is performed successfully for several signal processing applications, and show the effectiveness of our mapping algorithm by comparing it with previous approaches.

Keywords: Process network, multiprocessor system-on-chip, design space exploration, static analysis, mapping algorithm

I. Introduction

In embedded electronic systems design, Multiprocessor System-on-Chip (MPSoC) has become a necessity for efficiency in several design aspects. Especially, technology innovation has made it possible to realize MPSoC and has drawn researchers' interest

* 정회원, 서울대학교 전기컴퓨터공학부
(School of Electrical Engineering ad Computer Sciences/ Electrical Engineering, Seoul National University)

※ 본 논문은 MIC/IITA/ETRI, SoC산업진흥센터에서 수행한 IT SoC 핵심설계인력양성사업의 연구결과임.

접수일자: 2007년5월26일, 수정완료일: 2007년8월27일

in developing efficient MPSoC design environments. However, since the environment usually depends on the intuition of designers, it is a time-consuming job even with the skills of experienced experts. Therefore, an efficient MPSoC design methodology is becoming more and more critical. In this paper, we introduce a new design environment for efficient MPSoC design space exploration of signal processing applications. Figure 1 shows the overall design flow. In this paper, we focus on research in system specification and application-to-architecture mapping in the flow.

From a process network model[1] in SystemC, our conversion tool (SC2SDF and SC2C in Figure 1) parses the input SystemC files using Lex/Yacc to extract the connection information of ports between processes and it extracts the behavior from the SystemC files and generates the corresponding C files. Then, from these results, the conversion tool generates a model composed hierarchically of Finite State Machine (FSM) graphs and Synchronous DataFlow (SDF)[2] graphs through control/data flow analysis using SUIF1 compiler[3]. The hierarchical model graph and C codes are the inputs for static software/hardware (SW/HW) estimation. The estimated results are also annotated into the hierarchical model and used to map the application to the target architecture. To estimate the performance of the whole system, we schedule it statically. Our mapping algorithm takes the hierarchical model and target architecture information which contains bus architecture, communication overhead, OS information, and so on. With the input, the algorithm maps actors to the target architecture using an extended evolutionary algorithm based on Quantum-Inspired Evolutionary Algorithm (QEA)[4]. The algorithm can perform single objective optimization and multiobjective optimization as well. Finally, we validate the system by simulation. It is also possible to feed some information in the result back to the mapping step to find a better solution.

In Section II, we present related work. In Section III, we introduce the automatic conversion method for

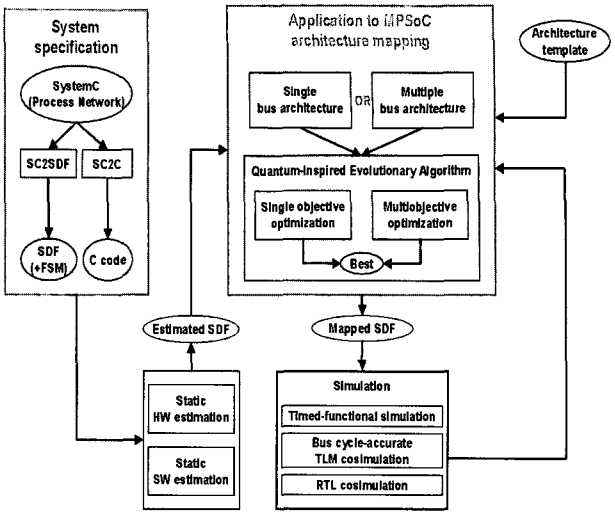


그림 1. 전체적인 설계 흐름도
Fig. 1. The overall design flow.

doing static analysis of process network model. Then, Section IV explains our mapping algorithm. In Section V, we show some experimental results with several signal processing applications. Finally, we conclude this paper in Section VI.

II. Related Work

Accurate and fast estimation of the performance of a system is very important for an efficient design space exploration in early design stages. Simulation-based approaches[5~6] have been proposed for the performance estimation of a process network model. Although they can easily accommodate dynamic behaviors in their modeling and simulation, they cannot guarantee worst case execution time analysis for real time systems. Quasi-static scheduling based on runtime statistics[7] is a good complement of the above mentioned approaches to work around their weak points for real time systems, but it requires more modeling effort to the designers compared to the process network model. In the quasi-static approach using Petri-net[8], they find all possible static paths and schedule them statically to reduce runtime scheduling overhead. However, it has a problem that the number of static paths can increase exponentially. In this paper, we present a method to analyze a process network model statically

by automatically converting the model to a hierarchical SDF model.

HW-SW codesign methodology has been investigated to enhance design quality during the last decade. Many heuristics[9~11] have been proposed for mapping tasks to target architecture with an objective. However, these approaches are hard to observe various aspects of system since many different design objectives in complex systems leads to a very large design space. Recently, there has been an increasing interest in the evolutionary multiobjective optimization algorithm, since the designer can find several members of the approximated Pareto-optimal set in a single run of the algorithm. Examples are SPEA2[12], NSGA2[13], and IBEA[14]. In this paper, we newly introduce a mapping system based on the improved QEA which finds best known results even faster than any other heuristic algorithms and simultaneously finds the approximated Pareto-optimal set for multiobjective optimization problem in a single system.

III. System Specification and Model Conversion

This section presents key ideas for the automatic conversion and we mention current restrictions of our approach.

1. Key Ideas for Automatic Conversion

First, we define several terms before we describe the generation process of the hierarchical model.

Definition 1 If there is a data flow path between two points in an actor/process and all points on the path belongs to the same actor/process then we call it a convex data flow path. Otherwise, we call it a non-convex data flow path. If an actor/process has a non-convex data flow path, then it can be blocked in the middle of its execution.

Definition 2 For a proper static analysis of an SDF model, an actor should have fixed input/output rates and should not have a non-convex data flow path. Such an actor is called a well-formed actor. An actor with variable input/output rates (for example, by dynamic constructs) and/or a non-convex data flow path is called an ill-formed actor.

The process of generating the hierarchical model is performed step by step as follows.

1. It first transforms the process network model to an intermediate SDF model where each actor corresponds to a process in the process network model. Note, however, that a process may not be well transformed to an actor since process is a more general concept than actor. More specifically, if a process has a non-convex data flow and/or variable input/output rates, then the actor generated from the process will be an ill-formed actor. So the intermediate SDF may contain one or more ill-formed actors.
2. Ill-formed actors having non-convex data flow are decomposed into more than one actor by finding non-blocking code segments.
3. Finally, if there are ill-formed actors with variable input/output rates, FSMs are added to model the rate control.

2. Restrictions

Our tool extracts input and output data rates for each process. It is done by counting the number of calls to read/write functions in the process network model. If such a call is made from inside a loop, we add an FSM implementing counters, one for each input condition. We determine the loop count by data flow analysis to compute the total number of calls made in the loop. In the current version, we assume that all loop counts and the number of calls can be determined statically, provided that the input condition is fixed.

Even though FSM can capture input-data-dependent behavior, it is not always possible to analyze the process network model statically. Therefore, we need to restrict the input process network model to a limited class. Such a problem in heterogeneous modeling has been investigated in several previous researches. Especially, A. Girault et al.[15] has proposed heterochronous dataflow (HDF),

where an actor has a finite number of type signatures each of which specifies fixed numbers of tokens consumed and produced. If all the type signatures have a fixed rate of execution, then static scheduling of the HDF is possible. In this paper, we restrict the input process network to satisfy the conditions posed by HDF, which limits the application of our approach. However, from our experience, we see that most digital signal processing applications are well modeled with such restrictions.

IV. Application-to-architecture Mapping

As the input, our mapping algorithm takes a task graph represented by a directed acyclic graph and target architecture information which contains processor allocation, bus architecture, communication overhead, OS information, and so on. The task graph consists of nodes and edges. Each node is annotated with one or more execution times, one for each type of processor that can run the task, deadline, and period. With the input, the system maps tasks to the target architecture using improved QEA for single objective optimization, for example, performance maximization or power consumption minimization, and multiobjective QEA to find approximated Pareto-optimal set simultaneously. Then we can compare the results and find the best solution satisfying design constraints.

1. Target Architecture

Our mapping algorithm can support two types of bus architecture that is, single bus architecture and multiple bus architecture with a bus matrix.

In the single bus architecture, processors (GPPs or DSPs), DMAs, a shared memory, and HWs are connected to a shared bus such as AMBA. This architecture is relatively simple but can produce degradation of performance because of communication overhead caused by using a shared bus.

In the multiple bus architecture, processors, HWs, DMAs, and shared memories are connected to a bus matrix. and a scheduler is in charge in executing slaves in the order of a static schedule. Using the bus matrix, we can find better parallelism of the system.

2. Improved QEA

QEA is based on the concept and principles of quantum computing such as quantum bit and superposition of states. Like other evolutionary algorithms, QEA is also characterized by the representation of the individual, the evaluation function, and the population dynamics. Instead of binary, numeric, or symbolic representation, QEA uses a probabilistic representation, which has the advantage of being able to represent a linear superposition of states in the search space. Thus, it has better characteristics of population diversity than other representations.

However, the original QEA can only represent two states for a Q-bit. If we want to apply it to multiprocessor mapping problem, we need multiple bits to represent multiple processors. To solve the multiprocessor mapping problem, we adopt one-hot encoding of processors.

The original QEA uses a rotation Q-gate as a variation operator. A rotation angle for each Q-bit is determined to drive it toward better solutions based on polar plot of the rotation gate. However, once a Q-bit becomes "0" or "1" state, it cannot be changed by the rotation Q-gate, thus inhibiting possible further improvement. To solve this problem, we apply a meta rule to the QEA. Initially, we set the possible minimum probability to a value larger than 0 and the possible maximum probability to a value smaller than 1. And then we increase or decrease these values after a meta-period which is given by the user or determined empirically. Thus, it has more chances to explore the search space and gradually settles down toward a globally optimal solution as we go through generations.

3. Multiobjective Optimization

For multiobjective optimization with QEA, we use the concept of a known evolutionary multiobjective

algorithm called IBEA (Indicator-Based Evolutionary Algorithm) which is currently the most efficient one, which is one of the algorithms that can be used via PISA (Platform and Programming Language Independent Interface for Search Algorithms)[16].

4. Evaluation by Static Analysis

We use basically a list scheduling technique to evaluate the performance of each mapping solution obtained by the improved QEA. To support hard real time constraints and multi-rate systems, we schedule all task instances within the hyper-period (least common multiple of the task periods). We give higher priority to tasks with earlier deadline. If a task does not specify deadline, then we assign priority based on the path-lengths to tasks with deadline specification. Currently, our scheduler supports only non-preemptive scheduling.

가. Evaluation for Single Bus Architecture

Our scheduling algorithm considers dynamic OS behavior (e.g. context switch overhead, interrupt service routine overhead and device driver overhead), bus communication overhead, bus conflict and bus arbiter overhead.

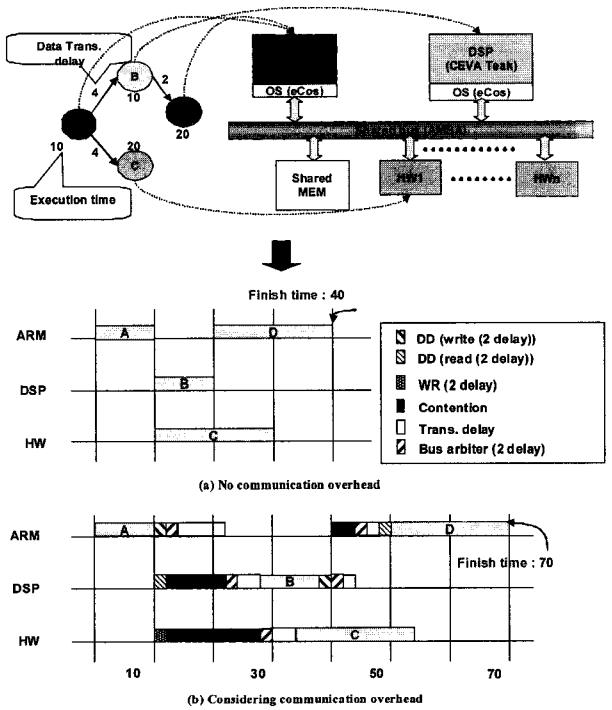


그림 2. 단일 버스 구조에서의 스케줄링
Fig. 2. A scheduling example for single bus architecture.

Figure 2 shows an example of the scheduling for single bus architecture. In Figure 2, (a) describes a scheduling behavior without communication overhead. On the other hand, (b) considers the communication overhead including device driver, bus conflict, bus arbiter, and etc. As shown Figure 2, we should note that communication quite affects the overall performance of a system.

나. Evaluation for Multiple Bus Architecture

We initially assume architecture with a fully connected bus matrix. To evaluate the performance of the architecture, we define some assumptions as follows.

1. If a memory between more than two processors (both are masters or slaves) is mapped to a local bus, the processors can share the local bus. And if they are masters, they can also use local DMAs depending on mapping results.
2. All masters always do not use global DMAs.
3. Communication between slaves is always done via global DMAs.

Under these assumptions, after the evaluation of the system, we can optimize the architecture with the following guidelines.

1. If there is no communication overlap between two memories, we can cluster them into one memory to reduce the number of connections in the bus matrix.
2. If there is conflict due to using only one DMA, we add a new DMA to make the performance better.

Figure 3 shows an example of initial architecture. Initially, all tasks (P1~P7) are assumed to be master, and all memories (M1~M6) between tasks are assumed to be slave.

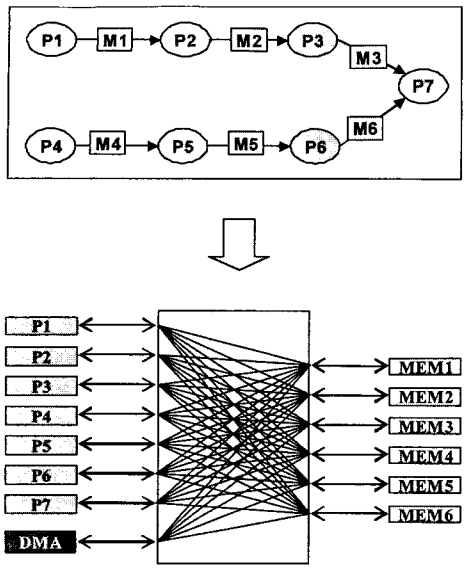


그림 3. 초기의 버스 매트릭스 구조
Fig. 3. Initial fully connected bus matrix architecture.

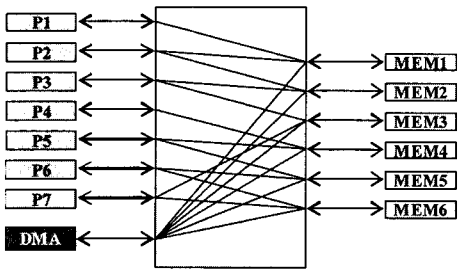


그림 4. 최대로 연결된 버스 매트릭스 구조
Fig. 4. Maximally connected bus matrix architecture.

They are connected to a fully connected bus matrix. And it is also assumed that a global DMA manages communication of slaves. However, in the initial architecture, there are unnecessary connections via which there is no actual communication. Thus, we refine the initial architecture by removing the unnecessary connections. This architecture is called maximally connected bus matrix architecture. Figure 4 shows the maximally connected bus matrix architecture refined from the initial fully connected bus matrix architecture in Figure 3.

Figure 5 shows an architecture obtained from a mapping example. The architecture includes 2 ARM processors (masters), 1 DSP (master), and 4 HWs (one master and three slaves). Assuming this architecture, we first estimate overall performance of the system by static scheduling.

Figure 6 shows a scheduling result. From the

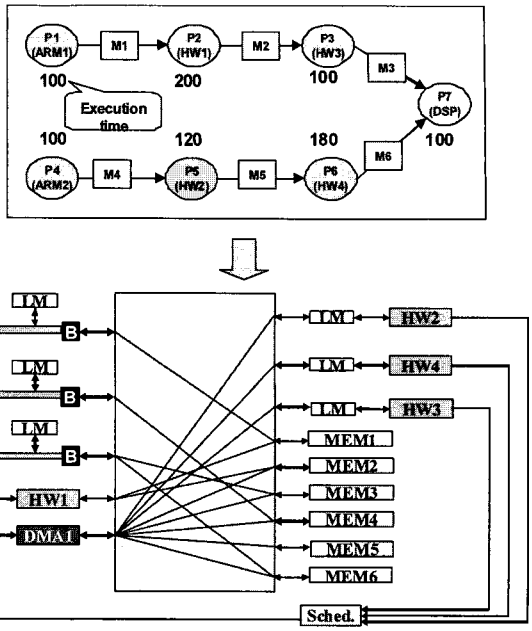


그림 5. 매핑에 의해 얻어진 한 다중 버스 구조
Fig. 5. An architecture obtained from a mapping example.

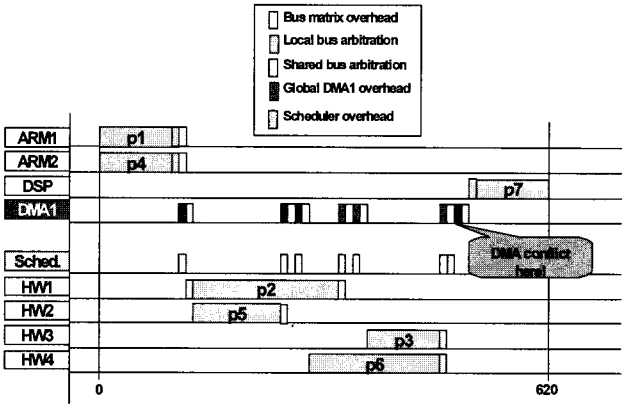


그림 6. 다중 버스 구조에서의 스케줄링
Fig. 6. A scheduling for multiple bus architecture.

schedule, we analyze if there is communication overlap between two memories, and then we cluster them into one memory if there is no communication overlap. It reduces the number of connections in the bus matrix. In the Figure 6, we can also see that DMA conflict exists. If we add another DMA to the architecture, we can avoid the conflict. The optimized architecture is shown in Figure 7. Again, the optimized architecture is evaluated to get its performance by static scheduling. A final schedule result is illustrated in Figure 8.

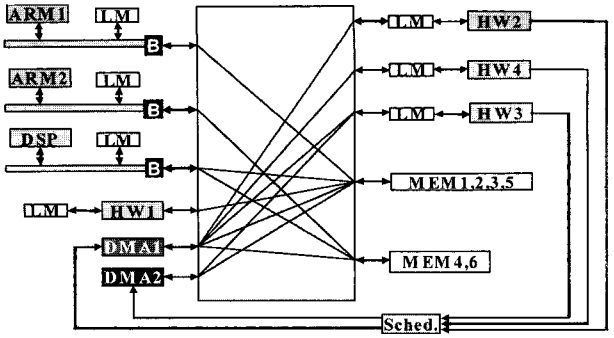


그림 7. 최적화된 최종 구조
Fig. 7. A final optimized architecture.

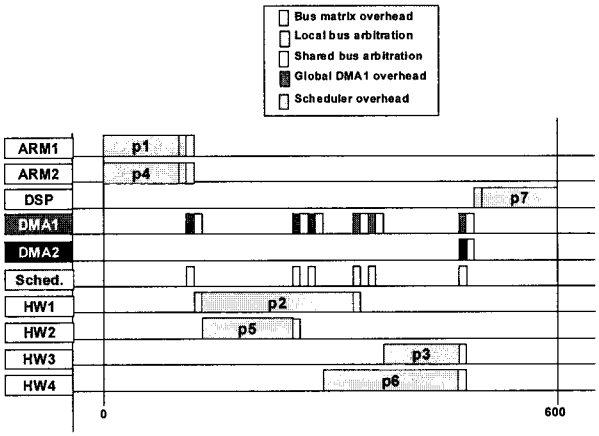


그림 8. 최종 스케줄링 결과
Fig. 8. A final scheduling result.

V. Experimental Results

The design environment proposed in this paper has been implemented on Pentium IV 2.4 GHz Linux machine with a 4 GB memory. We have also developed several GUIs.

1. Automatic Model Conversion

We have successfully performed the automatic conversion for four examples, JPEG, H.263, and H.264 encoder, and H.264 decoder. However, we show only one example due to space limit. Figure 9 shows the generated SDF model for H.264 decoder. There are two actors each of which is decomposed into three actors and six actors for which FSMs are added since they have conditional constructs.

2. Application-to-architecture Mapping

We compare our results with those of Yen[9],

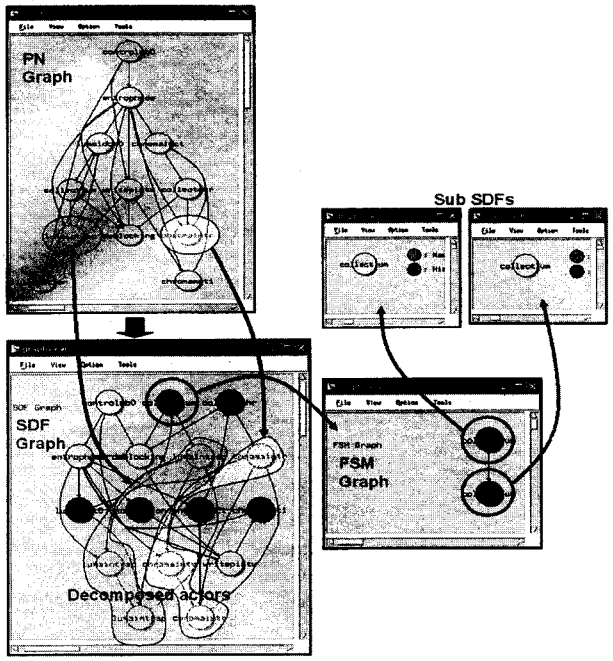


그림 9. H.264 decoder의 생성된 SDF 모델
Fig. 9. The generated SDF model for H.264 decoder.

표 1. MOGAC의 아주 큰 랜덤 예제들
Table 1. MOGAC's very large random examples.

Examples	No. of nodes	MOGAC		Oh & Ha		SHaPES		Modified QEA	
		cost	time (s)	cost	time (s)	cost	time (s)	cost	time (s)
MOGAC random1	510	39	2454	39	17.6	39	1.302	39	0.2727
MOGAC random2	990	35	12210	13	299.8	13	2.486	13	2.7389

MOGAC[17], Oh & Ha[10], and SHaPES[11]. Table 1 shows the performance of improved QEA with MOGAC's very large random task graphs[17]. It is worth to note that improved QEA finds the best known results faster than other heuristics even though it is an evolutionary algorithm, which can be flexibly extended to combine various other features.

Simulated annealing (SA)[18] is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global optimum of a given function in a large search space. The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions and

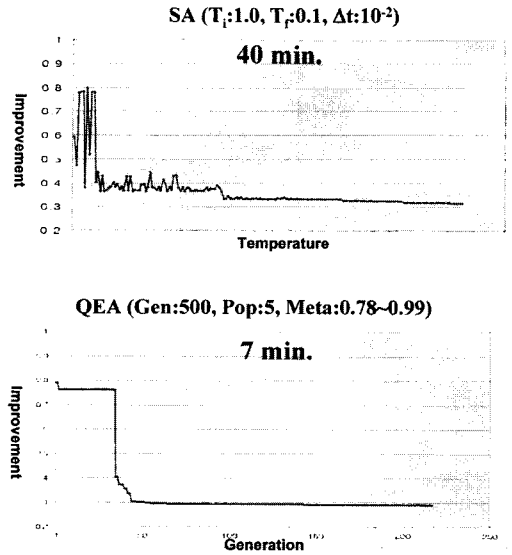


그림 10. H.264 decoder에 대한 다중 버스 구조에서의 QEA와 SA의 비교

Fig. 10. QEA vs. SA with H.264 decoder for multiple bus architecture.

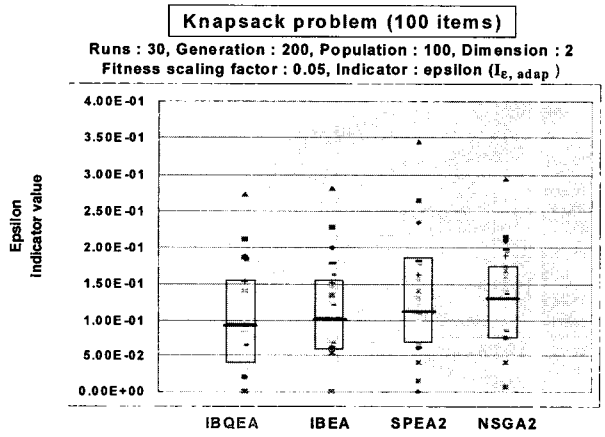


그림 11. 다중목적 최적화에 대한 비교 결과

Fig. 11. The comparison results for multiobjective optimization.

wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. SA is slow but finds a good solution. We have also compared the improved QEA with SA using H.264 encoder example in the multiple bus architecture. Figure 10 shows the comparison results. As shown in the figure, the improved QEA finds the same solution as SA but faster than SA.

Next, we show the results comparing our multiobjective QEA with other multiobjective evolutionary algorithms. Figure 11 shows the results for the knapsack problem using I+(A,R) (epsilon indicator) which gives the minimum distance between two Pareto set approximations[14]. In Figure 11, IBQEA is our multiobjective QEA. For each algorithm, 30 runs with different random seeds have been carried out. Here, for the quality measure we use box plots. A box plot consists of a box summarizing 50% of the data. A thick line within the box encodes the median. Smaller indicator value means better quality. From the results, we see that our algorithm is better than any other algorithms.

Through these experimental results, we can know the effectiveness of the modified QEA. Therefore, it enables fast and efficient design space exploration by finding good solutions quickly.

3. Design Space Exploration with H.264 decoder

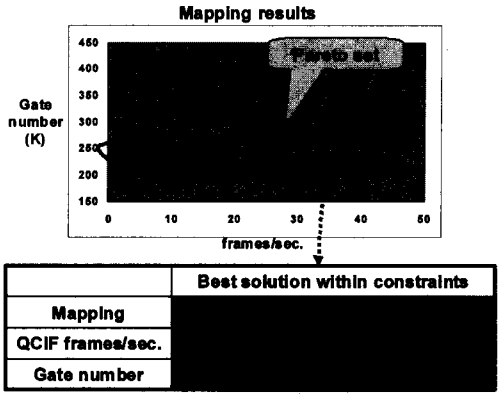
Figure 12 shows an example of the profiles for H.264 decoder. In this example, we assume the profiles are given by designer. In the table, the information of HW IPs comes from[19]. Given the table, we show the process for exploring design space using our mapping algorithm. Figure 12 also shows the mapping results of H.264 decoder. First, when we perform the mapping algorithm to maximize the performance of the system, we find a mapping solution that 2 actors are mapped to 1 GPP and others are mapped to HWs. On the other hand, when we perform the algorithm for multiobjective optimization, we find Pareto approximation set like the graph shown in Figure 12. Then, we can select the best solution which satisfies given performance constraint and minimizes area.

VI. Conclusions

In this paper, we propose a new method that automatically generates a hierarchical SDF model from a process network model to enable static analysis of the system. And we propose a mapping system to support single objective and multiobjective optimization as well. The mapping algorithm supports

	Delay (ns)		Area (K gates)
	SW Profile (200MHz)	HW IP (50MHz)	HW IP
ControlMb	590,530	-	-
VLD	63,875,673	879,120	4.7
Deblock	610,214,445	995,940	76.4
Chromaidct	5,716	180	7.9
Lumaidct	2,982	100	8.0
Chromaintra	40,760	720	12.3
Lumaintra	2,982	80	12.3
ChromaMC	106,295	9,740	23.5
LumaMC	4,008	400	23.6
CollectChroma	994	160	1.0
CollectLuma	1,864	280	1.0
WritePic	196,843	-	-

	All SW	Maximize performance
Mapping	1 GPP	1 GPP (2 actors), 10 HWs
QCIF frames/sec.	1.3	46.3
Gate number	150 K	170 K (HWs) + 150 K (GPP)



	Best solution within constraints
Mapping	
QCIF frames/sec.	
Gate number	

그림 12. H.264 decoder의 프로파일 및 설계 공간 탐색 결과
Fig. 12. The profiles and design space exploration results for H.264 decoder.

not only single bus architecture but also multiple bus architecture with a bus matrix. Our automated flow makes designers verify the system fast, thus reduces overall design time.

Currently, our mapping algorithm supports two kinds of architectures. However, in order to find more efficient solutions and increase the applicability, we must be able to support other system architecture with various design constraints such as the number of memories, many types of buses, and many communication structures. We are trying to extend our method considering those features. And scheduling overhead can be large with the increase in the size of the problem. We are going to develop a scheduling algorithm considering pipelining in scheduling and maximizing the performance of the system.

Acknowledgement

The authors thank the CAE Team at SAMSUNG Electronics for their support in various ways.

References

[1] G. Kahn, "The semantics of a simple language for parallel programming," in *Proc. of the IFIP Congress*, pp. 471-475, Amsterdam, Netherlands, Aug. 1974.

[2] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Trans. on Computer*, Vol. C-36, no. 1, pp. 24-35. Jan. 1987.

[3] The SUIF 1.x compiler system. http://suif.stanford.edu/suif/suif1/index.html.

[4] K. Han and J. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. on Evolutionary Computation*, Vol. 6, no. 6, pp. 580-593, Dec. 2002.

[5] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *Computer*, Vol 36, no. 4, pp. 45-52, Apr. 2003.

[6] A. D. Pimentel and C. Erbas, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Trans. on Computers*, Vol. 55, no. 2, pp. 99-112, Feb. 2006.

[7] S. Ha and E. A. Lee, "Compile-time scheduling of dynamic constructs in dataflow program graphs," *IEEE Trans. on Computers*, Vol. 46, no. 7, pp. 768-778, Jul. 1997.

[8] M. Sgroi, L. Lavagno, Y. Watanabe and A. Sangiovanni-Vincentelli, "Synthesis of embedded software using free-choice Petri nets," in *Proc. of DAC*, pp. 805-810, New Orleans, USA, Jun. 1999.

[9] T.-Y. Yen, *Hardware-Software Cosynthesis of*

Distributed Embedded Systems. Ph.D. dissertation, Dept. Electrical Engineering, Princeton Univ., Princeton, NJ, Jun. 1996.
[10] H. Oh and S. Ha, "A hardware-software cosynthesis technique based on heterogeneous multiprocessor scheduling," In *Proc. of CODES*, pp. 183-187, San Diego, USA, May 1999.
[11] D. Engels and S. Devadas, "A new approach to solving the hardware-software partitioning problem in embedded system design," In *Proc. of SBCCI*, pp. 275-280, Manaus, Brazil, Sep. 2000.
[12] E. Zitzler, M. Laumanns, and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switerland, 2001.
[13] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA-II," In *PPSN*, pp 849-858, Paris, France, Sep. 2000.
[14] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," In *PPSN*, pp. 832-842, Birmingham, UK, Dec. 2004.
[15] A. Girault, B. Lee, and E. A. Lee, "Hierarchical finite state machines with multiple concurrency models," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, no. 6, pp. 742-760, Jun. 1999.
[16] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA - A platform and programming language independent interface for search algorithms," In *Proc. of Conference on EMO*, pp. 494-508, Faro, Portugal, Apr. 2003.
[17] R. P. Dick and N. K. Jha, "MOGAC: A multiobjective genetic algorithm for hardware-software co-synthesis of distributed embedded systems," *IEEE Trans. on Computer- Aided Design of Integrated Circuits and Systems*, Vol. 17, no. 10, pp. 920-935, Oct. 1998.
[18] S. Kirkpatrick, C. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, Vol. 220, no. 4598, pp. 671-680, May 1983.
[19] S. Yoon, S. Park, and S. Chae, "Implementation of an h.264 decoder with template-based communication refinement," in *Proc. of APCCAS*, pp. 570-573, Singapore, Singapore, Dec. 2006.

저 자 소 개

안 용 진(정회원)
1999년 한양대학교 전기공학부 학사 졸업.
2002년 서울대학교 전기공학부 석사 졸업.
2007년 서울대학교 전기컴퓨터 공학부 박사 졸업.
2007년 현재 서울대학교 전기컴퓨터공학부 Post Doctor.
<주관심분야 : 컴퓨터이용설계, SoC설계>

최 기 영(정회원)
1978년 서울대학교 전자공학과 학사 졸업.
1980년 KAIST 전기및전자공학과 석사 졸업.
1989년 Stanford대학교 전기공학과 박사 졸업.
1978년~1983년 금성사 기정보 근무.
1989년~1991년 케이던스사 SMTS 근무.
2007년 현재 서울대학교 전기컴퓨터공학부 교수.
<주관심분야 : 컴퓨터이용설계, SoC설계, 마이크로 프로세서 구조>