On-Chip Communication Architecture Exploration for Processor-Pool-based MPSoC

Young-Pyo Joo
School of EECS
Seoul National University
Seoul, Korea
youngpyo@iris.snu.ac.kr

Sungchan Kim
School of EECS
Seoul National University
Seoul, Korea
sungchan.kim@iris.snu.ac.kr

Soonhoi Ha
School of EECS
Seoul National University
Seoul, Korea
sha@iris.snu.ac.kr

Abstract— **MPSoC is evolving towards processor-pool (*PP*)-based architectures, which employ hierarchical on-chip network for inter- and intra-PP communication. Since the design space of PP-based MPSoC is extremely wide, application-specific optimization of on-chip communication is a nontrivial task. This paper presents a systematic methodology for on-chip network design of PP-based MPSoC. The proposed approach allows independent configurations of PPs, which leads to efficient solutions than previous work. Since time-consuming simulation is inevitable to evaluate complicated on-chip network during exploration, we do early pruning of design space by a bandwidth analysis technique that considers task execution dependencies. Our approach yields the Pareto-optimal solutions between clock frequency and area requirements. The experiments show that the proposed technique finds more efficient architectures compared with the previous approaches.**

I. INTRODUCTION

With the continuous evolution of semiconductor process technology, it is possible to integrate more than 160 million gates into a 100 mm^2 chip with 45 nm CMOS technology [1]. Since a typical embedded processor uses a few million gates only, tens of processors can be integrated in a single chip to make an MPSoC (Multi-Processor System-on-Chip); it brings forth a design challenge of software as well as communication architecture. A traditional MPSoC where all processing components are tightly coupled is often unstructured as well as over-constrained. As a result, bringing a set of processing elements into a subsystem and, then, constructing a whole system in a well-structured form with multiple subsystems are considered as a promising solution. We call such a subsystem a processor-pool (*PP*), which is normally composed of processing elements, memories, and I/O components.

Decoupled restructuring with PP-based design has many benefits such as good scalability, design reuse of subsystem, modularity, and so on. Examples of PP-based architecture can be found in the SHAPES project [2], GPUs from NVIDIA, AMD, and Intel, and Cell BE from IBM. They have usually two levels of communication architecture: inter- and intra-PP communication. For instance, a PP that is called 'tile' in the SHAPES architecture consists of a set of heterogeneous processing component such as RISC, DSP, on-chip memory, and inter-tile communication interface. Intra-tile interconnection is multi-layer bus matrix, while packet-based switching network is used for inter-tile communication.

Local memory access goes through intra-PP communication architecture that should support relatively short and frequent memory accesses with small latency. In contrast, an inter-PP communication architecture is likely to deal with relatively infrequent massive data stream. It implies that the preferred configurations of the two communication architecture types quite differ from each other. For intra-PP communication, short access latency can be obtained by using standard bus interface with relatively narrow bus widths [3][4][5]. Inter-PP communication architectures often come with wide data bus widths more than 512 or 1024 bits to achieve higher throughput. Therefore, the design of communication architecture for PP-based MPSoC requires exploration of explosively large search space to consider different wish lists of inter-/intra-PP communications. On the other hand, performance evaluation of communication architecture is commonly resorted to simulation due to its dynamic behavior; it prohibits designers from exploring sufficient design space to choose desirable solution(s) in a limited time budget. This makes application-specific optimization of PP-based MPSoC a challenging problem.

This paper proposes a systematic methodology to explore communication architecture of PP-based MPSoC to tackle those difficulties. Recently, bus matrix is considered as a promising solution to achieve high performance and parallelism for rapidly increasing on-chip communications in chip multiprocessors [6] as well as MPSoCs [3][4][5]. Thus, in the current implementation of the proposed methodology, cascaded bus matrix architecture is used for both inter- and intra-PP communication. Considering other types of on-chip networks, such as ring-bus or NoC (Network-on-Chip), remains as a future work.

The design parameters considered for both inter- and intra-PP communication are clock frequency, arbitration priority, cascaded bus matrix topology, and on-chip memory allocation. There is an additional parameter, off-chip memory allocation, for inter-PP communication. The proposed technique configures each of communication architectures differently. To our best knowledge, none of previous work has addressed hierarchical communication architecture exploration. Instead, they have focused on a flat communication architecture based on a bus matrix [11]-[14].

Since simulation is typically resorted to evaluate on-chip communication of the underlying architecture, early pruning of design space is required to shorten the exploration time and explore wider design space. To this end, we compute the lower bound of memory bandwidth requirement for a given application considering task scheduling information. Any architecture candidates that fail to meet this requirement are excluded from simulation.

The paper is organized as follows. In the next section, we overview the related work and state our novel contributions. Section III provides background on the proposed approach and the problem definition, which are followed by the explanation of the overall design flow in Section IV. Section V and VI explain the details of the exploration algorithm. Then, we show the experimental results in

This work was supported by BK21 project, System IC 2010 project of Korea Ministry of Knowledge Economy, and Acceleration Research sponsored by KOSEF research program (R17-2007-086-01001-0). The ICT and ISRC at Seoul National University and IDEC provided research facilities for this study.

978-3-9810801-5-5/DATE09 © 2009 EDAA

Section VII to validate the proposed technique. Lastly, Section VIII concludes the paper and addresses future extensions.

II. Related Work

Target communication architectures in previous work on exploring on-chip communication architecture of MPSoC span hierarchical shared bus, crossbar-based bus matrix, and NoC. For bus-based architectures, main design decisions include mapping of processing components to buses, memory allocation, and bus topology generation [7][8][9]. The effects of floorplan to hardware area and achievable clock frequency of buses are also investigated in [10].

To provide more communication bandwidth, crossbar-based bus matrix architectures have been actively researched recently [11]-[14]. For a single bus matrix, main objective is to reduce the complexity of bus matrix by reducing the number of buses and merge logical memory segments into a physical memory through life-time analysis. In [11], a static bandwidth analysis is proposed to prevent exploring invalid design space and full system simulation for the qualified architectures is used to see if they satisfy given bandwidth constraints. The idea of pruning the design space by bandwidth analysis is similar to ours. The difference is that they compute the average bandwidth with respect to the whole execution time of application while we account for varying bandwidth requirements due to burst communication requests. An approach in [12] considers floorplanning to estimate wire delay for a global clock of a bus matrix. And the final topology is determined by ILP (Integer Linear Programming) formulas. Cascaded bus matrix topologies are searched to satisfy the given bandwidth constraints with reduced cost; simulated annealing [13] and MILP (Mixed ILP) [14] approaches have been proposed for this purpose.

There have been extensive researches on NoC architecture exploration. They mainly focus on the selection and mapping of tiles to network topology [15], buffer allocation in router design [16], and QoS (Quality-of-Service) support [17]. Since these works consider inter-PP communication only, interaction between inter- and intra-PP communications is not accounted for.

The contributions of our work compared with related approaches can be summarized as follow:

First, we propose a systematic method of communication architecture exploration, considering both inter- and intra-PP communications for a PP-based MPSoC. We seek for individual configurations of inter-/intra-PP communications considering clock frequency, arbitration priority, cascaded bus matrix topology, and on-/off-chip memory allocations. As a result, each of communication architectures can be designed efficiently.

Second, to cope with the extremely huge design space, we perform static analysis for the lower bound of bandwidth requirement, considering the varying bandwidth requirements according to the task schedule. The bandwidth requirement analysis eliminates invalid design space, reducing the number of simulations during exploration.

Lastly, the proposed method is flexible enough to easily consider different types of on-chip network architectures. Although we consider bus matrix architectures only in this paper, other architectures, such as hierarchical shared buses, packet-based NoCs, or ring-buses, can be easily brought into the proposed framework.

III. Background and Problem Definition

A. Terminology and Problem Definition

This section explains a task model of application and a target communication architecture to define the problem to be solved in this paper. An application $g=(V,E)$ is specified as a graph where V is a set of vertices to represent coarse grain tasks and E is a set of edges for communicating channels between tasks. A task is a primitive unit of mapping onto a processing element (*PE*). We distinguish two types of memory segments: one is dedicated for each PE and the other is shared for communications between tasks. We define such memory segments as logical memory blocks (*LMB*s). A LMB is associated to a PE for the tasks mapped on the PE to use their own local memory access. On the other hand, each channel of a task graph is associated with an LMB that represents a shared memory for inter-processor communication. Multi-application is modeled by a set of graphs $G=\{g\}$. To each application $g \in G$, there is a deadline by which its execution is required to be completed once enabled.

An architecture used in this work is modeled as shown in Figure 1; it consists of several PPs and one global communication architecture (*GCA*) for inter-PP communication. Each of PPs includes PEs, on-chip SRAMs, and an interface to GCA. GCA contains on-chip SRAMs and an off-chip DRAM interface. We denote a memory component in architecture by a physical memory block (*PMB*): LMBs defined in the task model will be mapped to PMBs. Interconnect architectures for both inter- and intra-PP communications are cascaded bus matrices. A bus matrix consists of multiple arbitration points (*AP*s) as depicted on the right side of Figure 1. We assume that bus matrices in a PP use a single clock while each PP may have a different clock frequency.

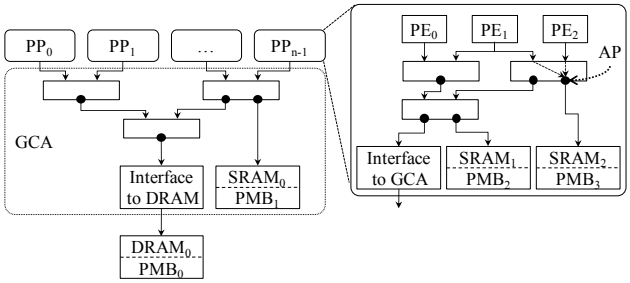


Figure 1. Processor-pool-based MPSoC architecture.

Here is a list of abbreviations for summary:

- PP: processor-pool
- PE: processing element
- LMB: logical memory block
- PMB: physical memory block
- GCA: global communication architecture
- AP: arbitration point

The problem addressed in the paper is to find an on-chip communication architecture for a given set of applications $G=\{g\}$ satisfying the associated deadline requirements D: The following design axes are explored: 1) on-chip memory synthesis by configuring the number of PMBs and the mapping of LMBs onto PMBs, 2) finding a topology of cascaded bus matrix, 3) finding the clock frequency of each cascaded bus matrices, and 4) deciding arbitration priorities in each bus matrix.

B. Quantum-inspired Evolutionary Algorithm

Evolutionary Algorithms (EAs) maintain a population of solutions for a given problem, selecting the best in each generation to make a better solution survive to final as in natural adaptation. Like all other EAs, QEA also consists of representation of individuals (solutions), fitness evaluation function, and population dynamics [18]. The only difference is that it uses quantum bits (Q-bits) as probabilistic representation for individuals instead of binary representation of genes. The probabilistic representation of Q-bit is described with two values, α and β, where $|\alpha|^2$ and $|\beta|^2$ mean the probabilities that the corresponding Q-bit becomes 0 or 1 respectively, as shown in the lower part of Figure 2. QEA does not require a crossover operation

between multiple solutions, which makes the implementation of QEA simpler than genetic algorithm (GA).

For the diversity of generated solutions, we can deploy multiple groups at the same time, each of which has its own probabilistic representation of Q-bit stream. This redundancy results in better solution by preventing from being stuck in local optima, even though too excessive duplication may slow down the proposed technique. In the figure, the length of individual Q-bit stream is *m*, while the number of groups and the number of generated individuals in a group are *n* and *p* respectively. Overall procedure of QEA can be summarized as follows:

1) For each group, *p* individuals (solutions) are generated based on the probability that Q-bits of the group imply. Then, a set of individuals from all groups at generation *t*, *P*(*t*), is generated. Each of individuals in *P*(*t*) is repaired by application dependent rules to make valid solutions.

2) Then, we evaluate generated individuals *P*(*t*) and save a set of the best solutions of groups, *B*(*t*), from *P*(*t*).

3) Based on *B*(*t*), the Q-bit stream of each group is adjusted so as to make newly generated individuals close to the best solution in probability: α and β values are adjusted, which is called "Q-bit rotation".

4) Among the best solutions for all groups, the best one is saved for the highest solution.

5) If the termination condition is met, we stop the evolution process and output the best solution obtained so far. A termination condition is usually defined by the maximum number of generations or by the case that all individuals are converged probabilistically to the best solution. For the case that neither of above conditions are met, we go back to the step 1 for the next generation.

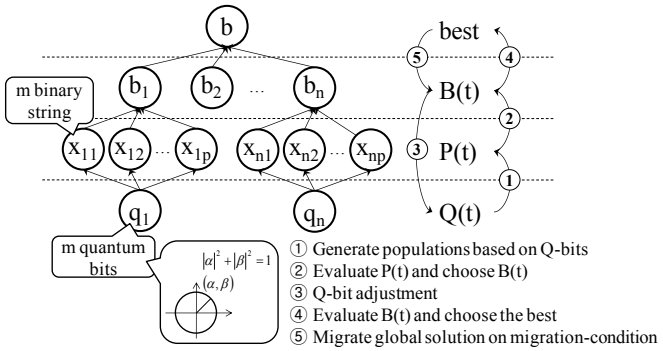


Figure 2. Overall QEA procedure.

IV. PROPOSED DESIGN FLOW

Figure 3 shows the overall procedure of the proposed exploration framework. There are three inputs: 1) the task model of applications, 2) memory traces for each task, and 3) the minimum bandwidth requirements for the tasks to access LMBs. The proposed exploration consists of two nested loops. The outer loop corresponds to the generation of a new architecture template by adding an on-chip SRAM to each of PPs and GCA, which increases the maximum bandwidth of communication architecture. Since not all PMBs will be used, this increase does not mean the cost increase of all PPs. The other parameters remain undetermined.

The grey box in the figure is associated with the inner loop where we use QEA to synthesize architecture candidates, x_{ij} in Figure 2; it allocates LMBs to PMBs, determines bus matrix topologies, and configures the operating conditions, such as clock frequency and arbitration priority. An initial architecture template has a single off-chip DRAM on GCA.

Generated architectures, x_{ij}, are evaluated first through a static analysis that examines whether all PMBs and APs in the generated bus matrix topology satisfy their associated minimum bandwidth requirements. The sustainable bandwidth of a PMB should be greater than the total bandwidths of the LMBs allocated to the PMB. Likewise, any AP in an architecture candidate should support all of the bandwidth requirements associated with the tasks passing through the AP at the same time. The algorithm to extract the lower bound of minimum bandwidth requirement from traces is explained in Section V. Based on the bandwidth requirement analysis, all candidates in which any PMBs or APs fail to meet the associated bandwidth requirements are discarded. In case no candidates remains, a candidate which can provide highest bandwidth is selected as a member of *B(t)*. In such a way, time-consuming simulation of invalid candidates is avoided without missing potential solutions. To obtain *B*(*t*), we compute the score (or fitness) of each candidate considering all of sustainable bandwidth, on-chip memory and bus matrix area, and clock frequency which will be explained in the next section in detail.

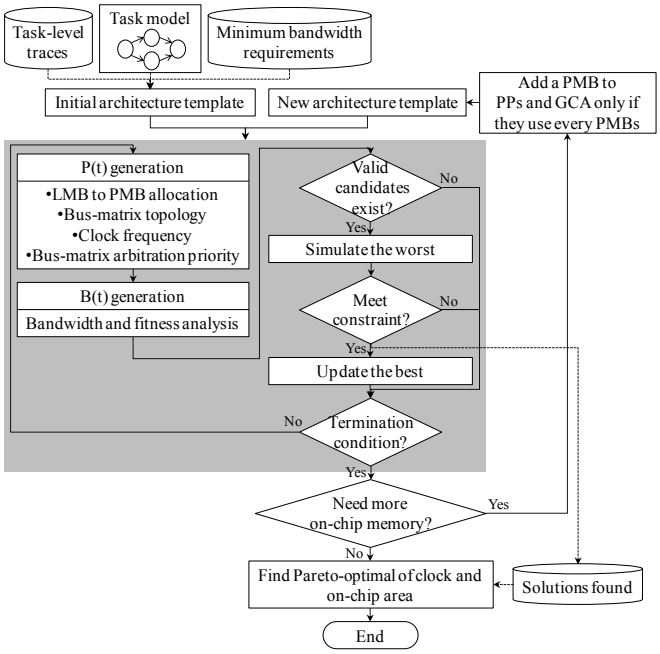


Figure 3. Overall procedure of the proposed approach.

After *B*(*t*) is constructed, the fitness of architecture candidates in *B*(*t*) should be evaluated to choose the best one at generation *t*. Simulation promises more accurate evaluation, but pays too long evaluation time to simulate all candidates in *B(t)*. Therefore we make a compromise between the quality of solutions and exploration time. At each generation, we take the worst solution (or the most expensive candidate) in *B(t)*, and perform simulation since it is mostly likely to satisfy the design constraints. If it meets all the constraints, we update the global best solution. And we move to the next generation. Although more efficient solutions may exist in *B(t)*, they are not simulated and discarded. But the evolutionary algorithm converges to a better solution anyway as generation is iterated. When an iteration of the inner loop is finished, QEA checks the termination condition to see if the current loop should be continued or not. The termination condition can be the convergence of solution qualities or its maximum iteration count given by the designer.

When QEA reaches any termination condition in the inner loop, the numbers of PMBs are increased by one for each of PPs and GCA to generate a new architecture template for exploring higher performance solutions. Then the inner loop starts again based on the

new architecture template. The maximum number of on-chip memories on each PP or GCA is increased up to as many as the number of LMBs. After the outer loop is finished, we obtain a set of architectures that satisfy the given deadline constraints. Based on on-chip area and clock frequency of the networks, we establish Pareto-optimal solution sets.

V. Trace Generation and Bandwidth Requirement Calculation

As explained in the previous section, the minimum bandwidth analysis allows us to avoid simulating invalid architecture candidates that have insufficient sustainable bandwidth of interconnect components, APs and PMBs. Memory access traces are obtained from the initial run of application execution on an in-house execution-driven multiprocessor simulator for this purpose. In this simulation, we assume an ideal communication network so that the traces contain the ordering of traces as well as processor execution time between traces excluding any communication architecture effects. Execution dependencies between communicating tasks are annotated as special symbols in the traces. Thus, the trace-driven simulator in the exploration process is aware of task execution dependencies of a given application.

Based on traces with execution dependencies annotated, we calculate the minimum bandwidth requirement for each task to access an LMB. Figure 4(a) illustrates an example of simple traces for three PEs. A trace for task is a sequence of blocks and the arrows between blocks indicate execution dependencies. For instance, a block MA_0 of P_A cannot start until a block MB_0 of P_B is finished. Note that there can be some delay to trigger successive blocks due to overhead such as synchronization or polling.

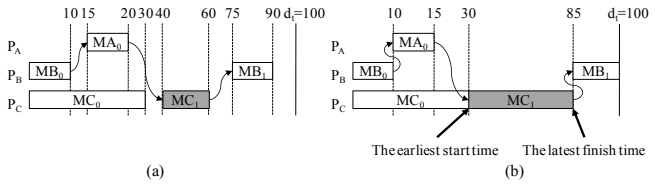


Figure 4. (a) Traces with dependencies annotated and (b) the minimum bandwidth calculation.

Let $m_{i,j}$ be a block of $task_i$ that contains accesses to lmb_j. Let $EST(m_{i,j})$ and $LFT(m_{i,j})$ be the earliest start and the latest finish times of a block $m_{i,j}$. $EST(m_{i,j})$ can be obtained by scheduling all of its preceding blocks in ASAP (as soon as possible) manner and $LFT(m_{i,j})$ by scheduling all successive blocks in ALAP (as late as possible) manner until the deadline. In the computation of $EST(m_{i,j})$ and $LFT(m_{i,j})$, we ignore all communication overhead to achieve the lower bound of the bandwidth requirement. $\{prev_{i,j}\}$ and $\{next_{i,j}\}$ are sets of preceding and successive blocks of $m_{i,j}$. $EXE_{min}(m_{i,j})$ is the minimum execution time of a block $m_{i,j}$. d_i means a deadline of an application where $task_i$ belongs to. Then $EST(m_{i,j})$ and $LFT(m_{i,j})$ become

$$EST(m_{i,j}) = \begin{cases} \max(\{EFT(prev_{i,j})\}) \text{ if } |\{prev_{i,j}\}| > 0 \\ 0 \quad \text{otherwise} \end{cases},$$

$$LFT(m_{i,j}) = \begin{cases} \min(\{LST(next_{i,j})\}) \text{ if } |\{next_{i,j}\}| > 0 \\ d_i \quad \text{otherwise} \end{cases}, \text{ where} \quad (1)$$

$$EFT(m_{i,j}) = EST(m_{i,j}) + EXE_{min}(m_{i,j}) \text{ and}$$

$$LST(m_{i,j}) = LFT(m_{i,j}) - EXE_{min}(m_{i,j}).$$

Let $M_{i,j}$ be a set of blocks of $task_i$ that contains access to lmb_j. Let $ACC_{i,j}$ be the total memory access counts to lmb_j by $task_i$. Then, we compute the minimum bandwidth $BWmin_{i,j}$ of $task_i$ to lmb_j as

$$BWmin_{i,j} = \frac{ACC_{i,j}}{\max(\{LFT(M_{i,j}\}) - \min(\{EST(M_{i,j})\})} \quad (2)$$

Figure 4(b) shows the case of a block MC_1 on a PE P_C. Note that blocks on the same PE have implicit execution dependencies since they cannot be overlapped.

VI. QEA Description of the Proposed Methodology

In the proposed technique, the inner loop of architecture exploration is performed by a QEA-based heuristic, where it is crucial to generate valid candidate solutions and properly evaluate them. In this section, we describe the QEA used in the proposed methodology focusing on the Q-bit representation of architecture and the fitness evaluation of a solution.

A. Q-bit Representation

Figure 5 shows a Q-bit representation for an architecture that consists of the following four parts.

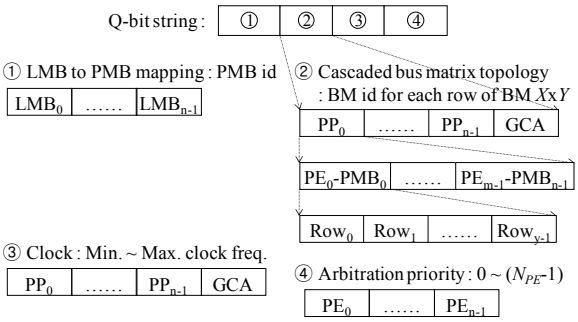


Figure 5. Q-bit representation of an architecture.

1) Mapping LMBs onto PMBs

This part indicates where each LMB is allocated by dividing the part into as many sections as the number of associated LMBs. For PP_i, it considers the LMBs confined in the pool, which include all local LMBs as well as the shared LMBs between processing elements in the same pool. Then an LMB can be mapped to a PMB in the pool or in GCA. Therefore, each section is given an integer that is associated with a PMB or the interface to GCA of the pool. The maximum integer is the number of PMBs in the pool or in GCA. For GCA, this part represents the mapping of shared LMBs between pools to PMBs in GCA.

2) Cascaded bus matrix topology generation

The Q-bit representation of cascaded bus matrix topology describes the sequence of bus matrices for paths from PEs to PMBs as shown in Figure 6(a). To construct a cascaded bus matrix topology for each of PPs, we consider a pool of bus matrices in $N_{PE} \times (N_{PMB}+1)$ grid form where N_{PE} is the number of PEs and N_{PMB} the number of PMBs in a PP of interest. For GCA, N_{PE} is replaced with N_{PP} where N_{PP} is the number of PPs. Then, any path for a PE to access a PMB is made by choosing an arbitrary bus matrix at each row of grid. During constructing a path, the following two restrictions are applied: First, a PE is connected to a single bus matrix. For instance, in Figure 6(b), PE_0 accesses PMBs only through bus matrix (0,0) in the first row of the grid. Second, a PMB too is connected to a single bus matrix. Thus, PMB_1 is connected to bus matrix (3,2) in the fourth row of the grid. As shown in Figure 6, a path for a PE-PMB pair is described as a set of integers indicating bus matrices chosen at the rows. Suppose that a path for PE_0 and PMB_0 in Figure 6 is constructed to visit bus matrices (0,0), (1,1), (2,1), and (3,0). Then, the corresponding Q-bit representation is {0,1,1,0} as in Figure 6(a). Once the entire paths are built, there might be redundant bus matrices that have a single master and a single slave interface; they are removed in the repair procedure.

The bus matrices (0,1), (0,2), (2,2), (3,0), (3,2), and (3,3) belong to this case. As a result, we obtain a cascaded bus matrix topology as shown in Figure 6(c).

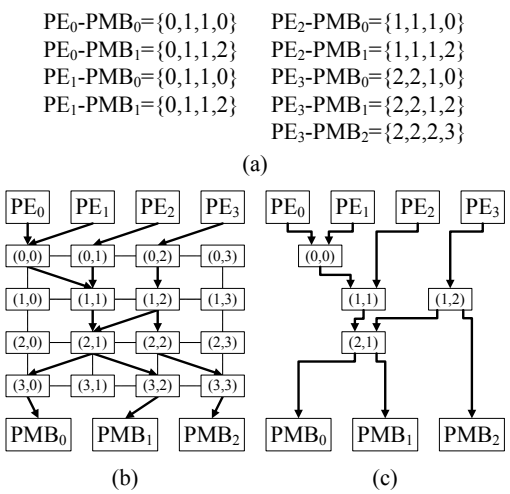


Figure 6. (a) Q-bit representation of cascaded bus matrix for each PE-PMB path, (b) initially generated cascaded bus matrix topology, and (c) final cascaded bus matrix topology.

3) Clock frequency of bus matrix

The Q-bit representation of clock frequency is straightforward. The selected clock frequency is randomly generated with an integer value, then repaired to meet the bandwidth constraints. It helps QEA to generate proper architecture candidate more quickly.

4) Arbitration policy of bus matrix

This part sets the priority of a PE to access PMBs. For every PE, a random priority is given.

B. Cost Evaluation Function

Another important part in a QEA-based heuristic is a cost function to evaluate architecture candidates generated by the inner loop of the proposed design flow. The cost function considers the bandwidth of an architecture candidate under consideration as well as implementation related parameters such as clock frequency and on-chip area. The cost function *Cost* is formulated as

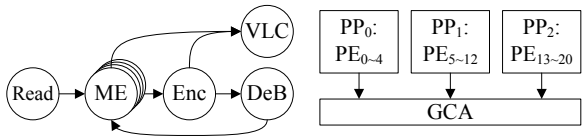
$$Cost = \begin{cases} BWcost & \text{if } BWCost < 0 \\ SYSscore & \text{otherwise} \end{cases}, \text{ where}$$
$$BWcost = \sum_{i=0}^{N_{AP}-1} \begin{cases} BW_i - BWreq_i & \text{if } BW_i - BWreq_i < 0 \\ 0 & \text{otherwise} \end{cases} \text{ and} \quad (3)$$
$$SYSscore = (area_{max} / area_{sum})^{\alpha} \cdot (clk_{max} / clk_{avg})^{\beta} + L.$$

BW_i is the bandwidth of AP_i that is defined by the product of the bus widths and the clock frequency. In this paper, bus widths are set to 32 bits, while a clock frequency is considered as a design parameter. $BWreq_i$ is the minimum bandwidth requirement of an AP_i. N_{AP} is the number of APs that target architecture has. Thus *BWcost* represents the amount of deficient bandwidth. *SYSscore* is the gain by using small on-chip memory and bus matrix area and low clock frequency. α and β are the coefficients for the gain factor of area and clock. These terms control the search direction of QEA to bias toward a specific design objective. By those two terms, the cost depends more on the deficiency of bandwidth or implementation cost once the bandwidth requirement is satisfied. L is a term to represent a reward for architecture that satisfies the given time constraints. L is set to a value larger than the maximum score of the candidates that fail to the constraints. Otherwise L is set as zero.

VII. EXPERIMENTS

We have implemented the proposed design flow in C++. A trace-driven simulator was implemented as a separate SystemC program operating at cycle-accurate level. As mentioned in Section V, we used an in-house execution-driven multiprocessor simulator to extract traces. To estimate the on-chip memory area of architectures, we used CACTI 4.2 [19]. To obtain on-chip bus matrix area, we synthesize and route bus matrices varying the dimension from 1 master and 1 slave to 16 masters and 16 slaves using Design compiler and Astro of Synopsys. Memory and bus matrix area information is obtained under 0.13 um CMOS technology. All experiments were conducted on a workstation with 3.0-GHz Xeon processor and 4.0-GB main memory running Linux.

As an application, we used a Picture-in-Picture (PiP) application that consists of one H.264 encoder for 4CIF-sized frame and two for CIF-sized frames as depicted in Figure 7. The application has 39 tasks with 75 LMBs and 54 inter-task communication channels. The target PP-based architecture is composed of 3 PPs and GCA. PPs have 5, 8 and 8 PEs respectively.



	Read/Enc/DeB/VLC	ME(4CIF:17/CIF:5 tasks)
4CIF	PE_3	$PE_{0,5\sim20}$
CIF1	PE_4	$PE_{1,13\sim16}$
CIF2	PE_4	$PE_{2,17\sim20}$

(c)

Figure 7. A PiP application: (a) H.264 encoder task model, (b) target architecture, and (c) task-to-PE mapping

In the first experiment, we compare the proposed approach with the exploration on flat communication architectures as in [8]-[11], which use a global clock without considering different communication requirements of PPs. For both approaches, we assumed that maximum clock frequency of generated architectures is 300MHz. The parameters of QEA used in the experiments are shown in Table I.

TABLE I. THE PARAMETERS OF QEA.

Para-meter	Coeff.		# of groups (*n*)	# of individuals in a group (*p*)	Prob. threshold for convergence	Max. generation count	Reward *L*
	α	β					
Value	1	1	50	20	0.999	2000	1000

Figure 8 shows the Pareto-optimal solution space obtained from the explorations based on the proposed approach and the flat architecture-based approach. In the graph, the X-axis corresponds to normalize on-chip areas while the Y-axis to average clock frequencies of all PPs and GCA in each of Pareto-optimal solutions. The values of the X-axis are normalized by the maximum on-chip area obtained from the exploration, which is 1.0 in the graph.

As shown in Figure 8, the proposed approach results in much efficient Pareto-optimal solutions than the flat architecture-based approach. Since the required clock frequency of flat communication architecture is probably dominated by the most bandwidth-hungry PP, the average clock frequency is larger than the proposed solution at the same on-chip area. The clock frequency difference between the approaches grows larger as the communication requirements of PPs become more diverse.

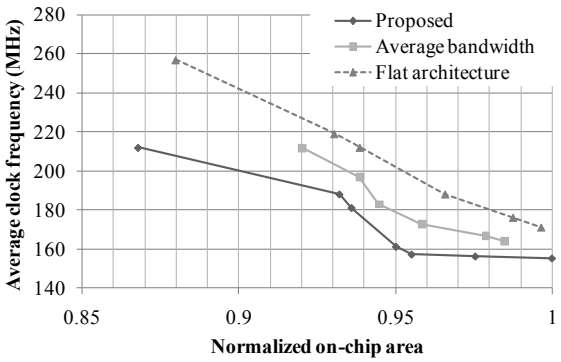


Figure 8. The Pareto-optimal solutions of on-chip area and average clock frequency.

In the next experiment, we verify the proposed bandwidth analysis technique compared with the average bandwidth analysis that is the most popular method to evaluate the performance of communication architecture as used in the previous work [11]. Considering (2), the average bandwidth $BWavg_{i,j}$ of a task t_i to access an LMB lmb_j can be calculated as

$$BWavg_{i,j} = \frac{ACC_{i,j}}{d_i} \tag{4}$$

We observe the followings from the graph: first, the Pareto-optimal curves for both approaches decrease gradually according to the increase of on-chip area as expected. Second, the proposed approach yields more efficient Pareto-optimal solutions; lower clock frequencies for every on-chip area are obtained compared with the average bandwidth approach. Moreover, whereas the average bandwidth approach is not able to produce a solution of smaller on-chip area than 0.92, it is possible to explore more diverse design space with our approach. From these observations, we verify that the proposed bandwidth analysis technique provides more scalability.

Table II shows the performance of the exploration techniques. The number of simulations performed by the average bandwidth approach is greater than the proposed as shown in the column '*# of simulated architectures*'. This is because the average bandwidth is usually less than the expected bandwidth from our analysis so that more architecture candidates satisfy the average bandwidth requirement. However, finding more candidates does not always mean more solutions satisfying the constraints. As we see in the last column of the table, the proposed approach finds more valid solutions even with the smaller number of simulations.

TABLE II. PERFORMANCE COMPARISON OF THE PROPOSED APPROACH AND THE AVERAGE BANDWIDTH-BASED APPROACH.

	Execution time	**# of pruned architectures**	**# of simulated architectures**	**# of selected architectures**
Proposed	7 hr 36 min	264,800	90	56
Average bandwidth	9 hr 28 min	408,800	158	29

VIII. CONCLUSION

In this paper, we have presented a systematic exploration methodology of on-chip communication architecture for processor-pool-based MPSoCs. In the current implementation, we consider bus matrix architectures for both local communication network inside each processor-pool and global communication architecture. Unlike the previous work, the proposed approach allows different configurations for each of PPs' communication architectures, which leads to efficient solutions in terms of lower clock frequency and less on-chip area. To avoid excessive usage of time-consuming simulation in the exploration, we use an efficient bandwidth analysis technique to prune inferior solutions failing to meet minimum bandwidth requirements. The experimental results show that the application of the proposed approach is able to draw much efficient Pareto-optimal solutions compared with the previous works: always lower clock frequency at the same on-chip area. As future work, we plan to support ring-bus or NoC for global communication architecture. Another extension is the consideration of power consumption estimation during the evaluation of generated architectures.

ACKNOWLEDGMENT

The authors thank Hoeseok Yang and Kyunghyun Kim for their help on the design of exploration algorithm and the experiments.

REFERENCES

[1] D. Wingard, "Tiles: the heterogeneous processing abstraction for MPSoC," in MPSoC '04, July 2004.
[2] P.S. Paolucci, A.A. Jerraya, R. Leupers, L. Thiele, and P. Vicini, "SHAPES: a tiled scalable software hardware architecture platform for embedded systems," in Proc. CODES+ISSS, pp.176-172, October 2006.
[3] AXI, ARM, http://www.arm.com/products/solutions/AMBA3AXI.html
[4] STBus Interconnect, STMicroelectronics, http://www.st.com/stonline/products/technologies/soc/stbus.htm
[5] SonicMX, Sonics Inc., http://www.sonicsinc.com
[6] Niagara 2 opens the floodgates, Microprocessor Report, November 2006.
[7] K.K. Ryu and V.J. Mooney III, "Automated bus generation for multiprocessor SoC design," IEEE TCAD, vol.23, no.11, pp.1531-1549, November 2004.
[8] K. Lahiri, A. Raghunathan, and S. Dey, "Design space exploration for optimizing on-chip communication architectures," IEEE TCAD, vol.23, no.6, June 2004.
[9] S. Kim and S. Ha, "Efficient exploration of bus-based System-on-Chip architectures," IEEE TVLSI, vol.14, no.7, pp.681-692, July 2006.
[10] S. Pasricha, N.D. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane, "FABSYN: floorplan-aware bus architecture synthesis," IEEE TVLSI, vol.14, no.3, pp.241-253, March 2006.
[11] S. Pasricha, N.D. Dutt, and M. Ben-Romdhane, "BMSYN: bus matrix communication architecture synthesis for MPSoC," IEEE TCAD, vol.26, no.8, pp.1454-1464, August 2007.
[12] S. Murali, L. Benini, and G. De Micheli, "An application-specific design methodology for on-chip crossbar generation," IEEE TCAD, vol.26, no.7, pp.1283-1296, July 2007.
[13] J. Yoo, S. Yoo, and K. Choi, "Communication architecture synthesis of cascaded bus matrix," in Proc. ASPDAC, pp.171-177, January 2007.
[14] M. Jun, S. Yoo, and E.-Y. Chung, "Mixed integer linear programming-based optimal topology synthesis of cascaded crossbar Switches," in Proc. ASPDAC, pp.583-588, January 2008.
[15] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor Systems-on-Chip," IEEE TPDS, vol.16, no.2, pp.113-129, February 2005.
[16] J. Hu, U.Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific Networks-on-Chip router design," IEEE TCAD, vol.25, no.12, pp.2919-2933, November 2006.
[17] A. Radulescu, J. Dielissen, S.G. Pestana, O.P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," IEEE TCAD, vol.24, no.1, pp.4-17, January 2005.
[18] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," IEEE TEC, vol.6, no.6, pp.580-593, December 2002.
[19] Cacti, HP, http://www.hpl.hp.com/research/cacti