CPNoC: On Using Constraint Programming in Design of Network-on-Chip Architecture

Ayhan Demiriz
Sakarya University
54187, Sakarya, Turkey
Email: ademiriz@gmail.com

Nader Bagherzadeh
University of California, Irvine
Irvine, CA, 92697, USA
Email: nader@uci.edu

Abdulaziz Alhussein
University of California, Irvine
Irvine, CA, 92697, USA
Email: azizsa@gmail.com

***Abstract*—NoC technology is composed of switched-based interconnections, where the communication resources are shared. Therefore, the optimal resource utilization is a crucial consideration for the efficient architecture designs. Application mapping and scheduling are important optimization problems. This paper studies the practicality of the Constraint Programming (CP) models on NoC architecture designs that effectively use a regular mesh with wormhole switching and the XY routing. The complexity of the CP models is compared to the earlier Mixed Integer Programming (MIP) models. Practical CP-based mapping and scheduling models are developed and the results are reported on the benchmark datasets. The results indicate that mapping and scheduling problems can be solved at near optimality even under relatively shorter run-time limits compared to those required by the MIP models.**

I. Introduction

High-performance System-on-Chip (SoC) and Chip Multi-Processors entail an increase in the number of required Processor Elements (PEs). However, the communication performance between these PEs is degraded due to the bus-system limitations. Shared buses become bottlenecks for system communications as the number of PEs increases. NoC is a new technology developed to overcome the limitations of the classical bus and point-to-point architectures and features more scalability [1]. NoC technology is based on the switched-based interconnections, where the communication resources are shared. PEs communicate and exchange data via packets using these shared resources (links, buffers, switches, etc.). These packets are transferred from a router to another until reaching the final destination. The NoC topology and the routing algorithms can greatly influence the resource utilization and the performance of the NoC systems. The NoC topology describes how these PEs are connected. Routing algorithms determine how packets are routed on the network and control how the traffic flows. Routing algorithms can be classified into adaptive and non-adaptive routing. In non-adaptive routing, the path that a given packet passes through is fixed and deterministic. Since packets follow the same path, this results in packets arriving in-order of initiation. Whereas in an adaptive routing, the route is not fixed and can be changed according to some conditions such as the network traffic status. Adaptive routing is flexible in terms of choosing alternative routes to route packets. Alternative routes can be selected according to traffic congestion which leads to the reduction in the delay and the power metrics. While the routing algorithms and the network topology influence the performance of NoCs, mapping applications to the NoC architecture is the key for the overall performance. Mapping tasks to the PEs and optimizing NoC traffic among them determine the actual performance metrics in terms of speed-up, power consumption, and the degree of parallelization.

Mapping facilitates assigning and placing the IP cores onto PEs such that an objective function (metric of interest) is minimized where a particular topology and corresponding task assignments are given beforehand [2]. Mapping can be considered as a sub-problem of floor-planning where finding the optimal topology is also considered as a part of the problem. However, floor-planning and mapping are practically equivalent in a regular mesh topology with the identical PEs. In the rest of the paper, floor-planning and mapping are used interchangeably. Mapping is related to the Quadratic Assignment Problem (QAP) [3], [4] which can be posed as Mixed-Integer Programming (MIP) model and was shown to be *NP*-hard [5], [6]. The QAP can generally be defined as assigning facilities to the locations in order to minimize the total cost of flowing goods among these locations. Combinatorial nature of the QAP makes it challenging by increasing the number of locations, n. Provided that the mapping problem has been solved at some optimality level and the IP cores are assigned to the corresponding PEs, application scheduling then minimizes the task completion time for the designated resource constraints such as the PEs and the communication bandwidth [2]. Together with the PE allocation layout, routing algorithms define the boundary for finding the optimal or near optimal solutions for application scheduling problems.

This paper explores the applicability of using the Constraint Programming (CP) models in finding solutions to the mapping and application scheduling problems. CP plays an important modeling role as being at the intersection of traditional mathematical programming and Artificial Intelligence (AI) [7]. The ability of utilizing intuitive and powerful optimization models that borrow modeling ideas from mathematical programming and powerful search strategies from AI makes the CP paradigm a good candidate to design efficient NoC architectures. Instead of complicated MIP models used in NoC synthesis earlier in [5], [6], [8], [9] that do not necessarily return a solution within the run-time limits, we propose very intuitive and efficient CP models which can achieve optimal or near-optimal solutions

within relatively very short run-time limits. Contributions of this paper are the following:

- Application mapping and scheduling problems are successfully formulated and modeled by the CP approach.
- Wormhole switching is incorporated to the CP models to find the optimal application scheduling solutions based on deterministic XY routing.
- Fifteen different benchmark datasets are generated by varying the number of tasks, the number of links, and the communication volume to study the behavior of the CP models under random realizations of the task assignments.
- Packet latency is studied under deterministic conditions to observe its behavior by changing the task complexities.

The remainder of this paper is organized as follows. Section II introduces the preliminaries of NoC platform and the related literature on the mapping problems. In Section III, the problem formulation and the CP-based model are given to present the advantages of our approach. Section IV contains information on the experimental setup, further details of the CP models, and discussions of the results. The paper is concluded and future directions are highlighted in Section V.

II. Preliminaries and Previous Work

A. Network-on-Chip Platform

Each PE is connected to a router through a Network Interface (NI). These routers are connected together into a mesh based NoC architecture. A router has five input/output ports, one port is connected to the Network Interface while the remaining ports are connected to other routers at each side (North, West, South, and East). Message Passing Interface (MPI) architecture is adopted to transfer the packets between the cores. Fig. 1 shows the interconnections between routers and the PEs. Wormhole switching strategy is used to transmit flits between the routers and the PEs where a packet is divided into a header flit and body flits. Flits are 64-bits wide and the header flits contain control information such as source/destination addresses, packet size in term of flits and packet sequence number for higher network layers usage. Packet sizes may vary from 1 flit to 64 flits. Dimension Order Routing (DOR) is employed to route packets among NoC architecture where it features deadlock and livelock freedom and design simplicity.

The header flits require extra cycles to be transmitted. One clock cycle is consumed by the Header Processing Unit (HPU) to process the destination address. Another clock cycle is consumed by the arbiter to arbitrate between different inputs. Moreover, flit transmission to the next router takes one cycle. Body flits require only one clock cycle to be transmitted to the next router.

B. Power Model

Power consumption in the semiconductor technologies consists of static and dynamic power. Static power consumption becomes a critical factor in the total power as the transistors become smaller and faster. The leakage current is a significant

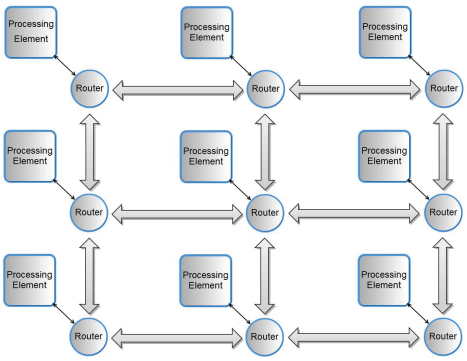

Fig. 1: A 3 x 3 NoC Mesh Network Example

factor when the technology is scaled down. On the other hand, dynamic power consumption is produced by the switching activities in the semiconductor circuits. In CMOS technology, it is caused by the capacitive load charge of the transistor gates as well as the capacitance and resistance of wires. NoC systems consist of routers and links that connect the routers. Flits travel from a hop to another through routers until they reach their final destinations. As they are traveling, they are processed by the routers' logic and passed through the links. Therefore, the flit energy can be expressed as follows:

$$E_{flit} = H \times E_{router} + (H-1) \times E_{link} \quad (1)$$

Where E represents the energy and H represents the number of hops. The header flit consumes higher energy since it takes extra clock cycles to be evaluated. The packet energy is expressed by the following equation [10]:

$$E_{packet} = H \times (E_{Hflit} + t \times E_{Bflit}) + (H-1) \times E_{link} \quad (2)$$

where t is the number of data flits. E_{Hflit} and E_{Bflit} represent the energy consumption of header and data (body) flits respectively and furthermore can be expressed more to the following:

$$E_{Hflit} = E_{buffer} + E_{crossbar} + E_{arbiter} \quad (3)$$

$$E_{Bflit} = E_{buffer} + E_{crossbar} \quad (4)$$

Dynamic power consumption can be reduced when the energy of the packet is minimized. This can be achieved by minimizing the packets communication energy between PEs. Mapping of the tasks plays an essential role of determining the packet traveling paths which reduces the number of hops. On the other hand, tasks scheduling affects the buffering requirements of packets along with travel path.

C. Previous Work

An efficient branch-and-bound algorithm was proposed in [8] for mapping problems in order to yield energy and reliability oriented solutions. The results of the proposed algorithm in [8] were compared to the results from simulated annealing based mapping. A heuristic random greedy algorithm was compared to the optimal MIP formulation of mapping and

voltage islanding problem in [11]. It was shown in [11] that the MIP formulation was not tractable. It is evident from [8], [11] that the MIP formulations of mapping problem may result in impractical and inferior solutions.

Another branch-and-bound algorithm is proposed in [5] to find near optimal mapping solutions that are energy aware and satisfy the bandwidth constraints under some performance constraints. The proposed mapping algorithm in [5] optimizes the energy requirements based on the random assignment of the tasks to the IP cores. So the mapping problem is to place the IP cores in the appropriate locations on the regular grid in order to minimize the energy based on the network traffic [5]. Authors in [12] propose a comprehensive two-stage NoC synthesis model by utilizing the MIP. In the first stage, an energy efficient system-level floor-planning is achieved through MIP. The second stage is conducted for a detailed routing functionality. At stage two, placement of routers is optimized to enable the traffic flow. The MIP model is very complicated in [12] and it often does not return a solution within the run-time limits. So a clustering-based heuristic is proposed to address the complexity issue of the second stage. It should be noted that if a certain level of the problem abstraction is not applied appropriately in the MIP models, it is very likely that the MIP models will not be able to return a solution within the run-time limits due to complexity issues. In a similar way, the MIP models can be applied to the dynamic voltage scaling problem on the heterogeneous platforms [6]. From the earlier work, it is evident that the MIP implementations may suffer gravely from two shortcomings: misjudging the level of problem abstraction and the difficulty in accurate modeling.

The CP paradigm is used in [13], [14] on scheduling problems of multi-task application on Multi-Processor System-on-Chip (MPSoC) after using the MIP models for finding allocation solutions. Thus, mapping and scheduling problems are effectively decomposed [13], [14] into two sub-problems and solved in tandem like in our implementation. Computational efficiency of the decomposition method was shown in [14] in comparison to the full optimization models. Therefore, floor-planning and application scheduling tasks are conducted in two different stages as a result of decomposition. However, we utilize the CP approach for both mapping and scheduling problems rather than using the MIP for the mapping stage and the CP for the scheduling stage in contrast with [14].

III. Motivation and Problem Formulation

A. Basic Assumptions and Overview

Assume a set of PEs, organized as a 2-D mesh of dimensions $n = m \times m$. Since all of the PEs are identical, the architecture is homogeneous. Each PE can be labeled according to its position in the mesh (as an x and y coordinate pair) and has the capability of executing several tasks of the application in tandem. Because all the PEs on the mesh are identical, there is no differences in the task execution time. Nevertheless, the heterogeneous architecture can be implemented by introducing PE dependent execution times and some additional constraints to relate each task type to the PE families. The implementation of the heterogeneous PEs is out of scope of this paper and is planned for a future publication.

The task set is represented by an annotated task graph (TG), e.g. the one shown in Fig. 2 which depicts a sample task graph generated by TGFF [15]. Each node in the graph represents a task of the application with its execution time given in the parentheses. Communication requirements (flits) between tasks are shown on the edges (links). TGFF can generate individual task deadlines. However we omit this type of constraints in this study. TGFF was used in generating several benchmark task graphs that have been used in the literature before (see [11], [8], [12]).

The cost of transferring data from one PE to another is a function of the routing algorithm. Deterministic routing algorithms can easily be incorporated to the optimization problem on hand and can help in finding the optimal application task schedule. The transfer cost based on the XY routing algorithm is proportional to the Manhattan distance which can be calculated between points a and b on a grid as: $\mathrm{TC}_{ab} = |x_a - x_b| + |y_a - y_b|$.

The buffer size is assumed to be unconstrained (i.e. infinite buffer size). Any communication link can only be occupied by a single packet at any given time without any constraint on the bandwidth size. The communication links are bi-directional. In other words, any particular link between two routers can be considered as a separate link (resource) in each direction.

B. Problem Formulation

The basic mapping problem is an instance of the QAP and can be formulated as a mathematical program model given in Eq. 5 where the decision variable $x_{ij},\ i,j = 1,\ldots,n$ is a Boolean variable, f and d are flow and distance matrices (parameters) respectively. In general, QAP instances that have size of $n > 30$ cannot be solved in a reasonable time [4].

$$\begin{aligned} \underset{x\in\mathcal{X}}{\text{minimize}} \quad & \sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl} \\ & \sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ldots,n, \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ldots,n, \\ & x_{ij} \in \{0,1\},\ i,j = 1,\ldots,n. \end{aligned} \tag{5}$$

where the decision variable, x_{ij} simply determines whether goods (or information in our context) should be sent from node i (PE_i) to node j (PE_j). The parameter d is the Manhattan distance between all the possible pair combinations of the PEs on the mesh. Parameter f represents the information to be sent between each PE.

The objective function in Eq. 5 is a representation of the dynamic power and it can be simplified by introducing a permutation variable, π, as below [4]

$$\underset{\pi}{\text{minimize}} \quad \sum_{i,j=1}^{n} f_{ij} d_{\pi_i \pi_j}$$

where $\pi_i = j$, if $x_{ij} = 1$ for $i, j = 1, \ldots, n$. Introducing the permutation variable enables us to use the CP by invoking `alldifferent` constraint which specifies the values assigned to the decision variables that must be pairwise distinct [16] on the permutation variable,π, as in Eq. 6. In other words, each of the n variables should be assigned a unique value between 1 and n.

$$\begin{aligned} \underset{\pi}{\text{minimize}} \quad & \sum_{i,j=1}^{n} f_{ij} d_{\pi_i \pi_j} \\ \text{s.t.} \quad & \text{alldifferent}(\pi_1, \ldots, \pi_n), \\ & \pi_i \in \{1, \ldots, n\}, \ i = 1, \ldots, n. \end{aligned} \tag{6}$$

As given in Eq. 6, the QAP can be modeled in a very intuitive way by utilizing the CP approach. Instead of n^2 Boolean decision variables and $2n$ constraints besides 0-1 integrality constraints in Eq. 5, there are only n decision (permutation) variables and a single `alldifferent` constraint besides the integrality constraints. Aside from the simplicity of the model representation, the central strength of the CP approach is to construct smart propagation search techniques to detect the dead-ends on the search tree as early as possible and prune them [16]. The efficient search depends on the efficient filtering of the domain which is defined as a finite set of elements that can be assigned to a decision variable. Efficient arc consistency algorithms exist to find the solutions that satisfy the `alldifferent` constraint. In general, `alldifferent` can be checked for consistency, i.e. to be determined to have a feasible solution, in $O(z\sqrt{n})$ time [16] where n is the number of decision variables and z is the sum of cardinalities of each domain that belongs to a decision variable. Considering that $z = n^2$ in our problem, the complexity of consistency checking of `alldifferent` constraint is then $O(\sqrt{n^5})$.

Once the solution to the floor-planning problem determines the position of the processing elements on the mesh, the application task scheduling problem can be posed as a separate mathematical programming model. Recall that all the PEs are identical and the processing times for a given task are equal for all the PEs. Hence, the optimal solution found in Stage I can be considered as the best floor-planning assignment that does not consider the bandwidth constraints. In other words, it is a relaxed solution without the bandwidth constraints. The application scheduling problem in Stage II becomes challenging when the bandwidth constraints are introduced. The new scheduling problem can be posed as a constraint programming model by letting appropriate decision variables to represent start time, end time, and processing time of tasks. The advantages of using CP in scheduling are two-fold [17]:

- Natural and flexible way of modeling the scheduling problems by the CP.
- Efficient temporal and resource constraints.

For example, precedence constraints can be represented by `endBeforeStart`, `endBeforeEnd`, `endAtStart`, and `endAtEnd` constraints easily. An `endBeforeStart` constraint requires a job to end before the other job starts. Similarly, an `endAtStart` constraint requires a job to end at the start of the other job. A `noOverlap` constraint can be used to schedule the tasks that utilize certain resources. Thus, precedence constraints can be constructed from the task graph. Communication links between routers can be considered as resources to model bandwidth requirements. We can then easily use `noOverlap` constraint to find a schedule of transmission tasks through a particular communication link without violating precedence and resource availability constraints.

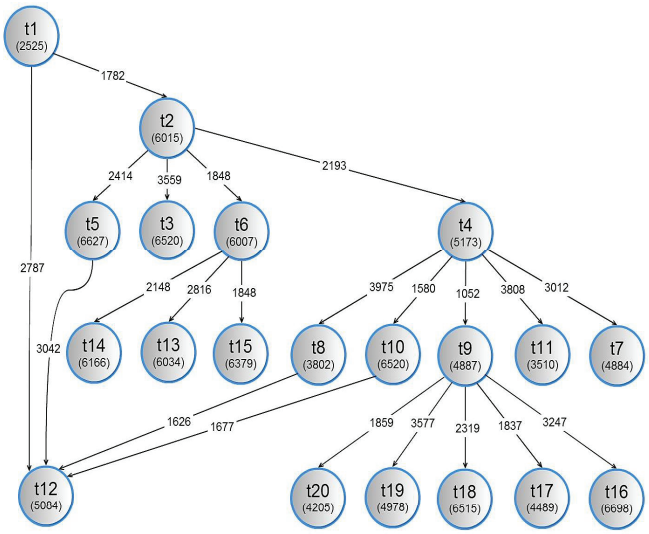

Fig. 2: A Representative CTG: D20A

IV. Experimental Study

This section illustrates the applicability of our constraint programming based approach on randomly generated benchmark datasets.[1] Throughout the experiments, we assume that channel can hold 64-bit flits, the packet size is 64 flits, and a 1000MHz architecture is utilized. Tasks execution time and packets transmission delay are measured in clock cycles and therefore clock operating frequency does not influence the mapping and scheduling performance. The header flits require extra time of three clock cycles. 3×3 and 4×4 regular mesh architectures with identical PEs are utilized in our experiments. Notice that it is impractical to solve the QAP for larger meshes by using MIP, since the problem size becomes very large. However, the larger mesh architectures can be easily applied in our implementation due to the parametric nature of the CP models. All experiments are run on a dedicated dual Xeon quad-core processor server running on Windows Server 2008 with 14GB memory for bookkeeping purposes. However, individual models can be run on development laptop that has a dual-core processor running Windows 7 with 4GB memory. The CP models are implemented by using IBM ILOG OPL Studio, which is available free of charge to the academicians at IBM Academic Initiative web site.[2]

[1] All the datasets and optimization models can be downloaded at http://tinyurl.com/cdq5l9n .

[2] http://tinyurl.com/cu5txlg

A. Benchmark Set Generator

For the sake of providing a set of benchmarks for the mapping and scheduling algorithm, a random problem generator, called TGFF, was used [15]. TGFF generates annotated communication task graphs (CTG) in a pseudo random way. Each task graph represents an executable application, which can be created starting from a set of configuration parameters, such as the number of tasks and maximum input and output task degrees. Fig. 2 depicts a sample communication task graph with task execution times (clock cycle) are depicted within each task, communication requirements are given on the arcs (links). Precedence constraints can easily be derived from Fig. 2

A wide and representative set of task graphs was synthesized. The number of tasks (i.e. the problem's size) was randomly generated to be in the range from twenty to seventy tasks. Several task degrees were also set, in order to model different complexities of the fifteen different task graphs generated. A summary table is given in Table I which lists the number of tasks, the number of communication links, the sum of computational time in clock cycles, and the sum of communication volume in flits of each CTG. Table I also reports representative scheduling problem sizes for both 3×3 and 4×4 regular mesh architectures.

B. Implementation Details of the Proposed Model

There are practically two separate CP models, one for the each stage. The stage I is primarily composed of the CP model given in Eq. 6. Since 3×3 and 4×4 regular meshes are used in our implementation, there are only nine and sixteen decision variables respectively for the CP model in the stage I and the domain (D) of these variables are $\{1, \ldots, 9\}$ and $\{1, \ldots, 16\}$ respectively. There are two main input parameters for this stage besides the random assignment of the tasks to PEs: namely flow (f) and distance (d) matrices. The distance matrix, d, represents the transfer cost TC for the XY-routing, i.e. Manhattan distances between the nodes on the meshes which are 3×3 and 4×4 in our implementation (see Fig. 1). The flow matrix, f, expresses the total communication requirements (data flits) between each task given on the edge of the task graphs (see Fig. 2) with the additional header flit overhead for each data packet transferred.

The mapping problem is poorly-defined without randomly assigning tasks to the PEs first. Otherwise, it is always feasible to assign all of the tasks to a single PE which will result in '0' energy consumption i.e. null solution for all the data communications of the tasks. The random assignment is devised in such a way to guarantee the assignment of at least one task to the each PE. However, each PE is expected to serve for the multiple tasks as equally likely.

After producing an assignment solution at Stage I by solving CP model given in Eq. 6, we can now schedule all the tasks and their corresponding communication tasks as well. The second stage is more complicated than the first one in terms of the modeling as the detailed implementation of the routing algorithm is required. The objective of Stage II can be set to minimizing the maximum completion time (makespan) among all the tasks. To enable an easier scheduling implementation, one can consider communication tasks among PEs at a very detailed level to generate precedence constraints appropriately. The reason for this is primarily the implementation of wormhole switching algorithm with XY routing.

ID	Task	Packet ID	Seq	Link	Header	Clock Cycle

Fig. 3: Data Structure for Communication Tasks

We propose the creation of a data structure at flit type level given in Fig. 3 to manage this implementation. This particular data structure is created after Stage I to initiate the scheduling model in Stage II of the CP model. The `ID` variable is a unique ID to identify each communication task. The `Task` variable represents the communication task between two particular PEs. For example, in Fig. 2, the communication task between `t6` and `t15` can be considered as the pair "<6,15>". Notice that we do not necessarily consider the physical location of PEs that process tasks `t6` and `t15` for the `Task` variable. The data transfer between any two routers can be divided into several packets. The variable `Packet ID` is used to identify the corresponding packet. If the number of flits carried between two routers is t, then the number of packets can be calculated by $\lceil \frac{t}{p-1} \rceil$ where p is the packet size. The sequence ID is represented by the variable `Seq`, which is incremented accordingly depending on the particular link the packet visits and, whether a header or body flit is being transmitted. The variable `Header` is used to identify the type of flit being transmitted. For a header flit, the `Clock Cycle` variable is set to 3. Otherwise, `Clock Cycle` is set to $p-1$ (the remaining size of the last packet) for the body flit. Representative scheduling problem sizes for the data structure created after Stage I are given in Table I for both 3×3 and 4×4 meshes.

By carefully crafting the data structure given in Fig. 3, the precedence constraints which are used to implement the wormhole switching and the XY routing can be easily devised in order to find a solution to the application task scheduling. As mentioned in Section III `endBeforeStart`, `endBeforeEnd`, `endAtStart` and `endAtEnd` constraints can accordingly be used in Stage II CP model. Notice that the physical links between routers should be treated as resources in the application task scheduling problem. When we consider these links as resources, it is pretty straight forward to implement the bandwidth constraints by `noOverlap` constraints. Notice that we do not have a bandwidth size limitation in our implementation, however our formulation implicitly constraints the bandwidth size as equal to the packet size. Therefore only one packet can be served at any given time for a particular link (physical link) on the network.

The run-time limits for the Stage I and the Stage II CP Models are both set to 1000 seconds total CPU time in our experiments. Considering that four processors are set to be used by IBM ILOG solver, CP models return optimal or the

TABLE I: Summary Data of Communication Task Graphs and Corresponding Problems

Task Graph Instances	D20A	D20B	D20C	D30A	D30B	D30C	D40A	D40B
Number of Tasks	20	20	24	30	30	38	42	40
Number of Links	22	19	23	41	37	37	53	52
Tot. Comp. Time	108469	89687	108240	146841	153730	181745	199638	195887
Tot. Comm. Vol.	54526	46771	59463	103420	94909	94909	131847	129208
Sched. Prob. Size (3×3)	3016	2741	3304	5798	5578	4649	7063	7902
Sched. Prob. Size (4×4)	5208	4442	5172	8290	8301	8014	9932	10409
Task Graph Instances	D40C	D50A	D50B	D50C	D70A	D70B	D70C	
Number of Tasks	41	54	53	57	72	70	77	
Number of Links	58	68	70	71	87	110	91	
Tot. Comp. Time	220619	267207	246648	288135	349326	357449	399142	
Tot. Comm. Vol.	150717	164801	170996	172843	211611	266535	219866	
Sched. Prob. Size (3×3)	8328	9517	10648	10390	12158	14140	12842	
Sched. Prob. Size (4×4)	13335	13573	14440	13254	16563	23401	19454	

best solution within 250 seconds at each stage. This is the way CPU time is controlled by solver, therefore there is no need to report the actual run times.

C. Results

We conducted the experiments for fifty random realizations of the instances that are summarized in Table I. A random realization is essentially a set of random assignments of the tasks to the PEs. All of the experiments ran successfully and returned solutions without any failure. Compared to very limited the number of tasks and the number of links used in the experimental studies previously (e.g. around 30 in [13], at most 24 in [11]), the task graphs generated for our experiments include graphs with 77 tasks and 110 links at most. It should be noted that even the benchmark datasets used in the previous work such as [11], [8], [12] were generated by TGFF [15]. The floor-planning model i.e. the Stage I model always returned the optimal results. But the application scheduling model i.e. the Stage II model which is a more complex model than the Stage I returned the best solution within the run-time limit. As a representative result, the Stage II models picked the best solution among 2.1, 9, 3.9, and 2.24 solutions on the average for D30C, D40C, D50C, and D70C instances respectively. Since CP is an AI-based search technique that will return the current best solution if the none solution is proven to be the optimal one.

We observed the objective values of the Stage I and the Stage II CP models, these are summarized by the box-plots depicted in Fig. 4 and Fig. 5 respectively for the 3×3 mesh architecture. A box-plot is a common plotting tool to summarize and depict the distribution of a continuous random variable. The bottom side of the rectangular box represents the first quartile. The middle horizontal line in the rectangular area is the median. Finally, upper side of the rectangular box shows the third quartile of the distribution. The range of the distribution can be easily visualized by a box-plot. Outliers are presented at the tail or the head of the box-plot as circles. As the complexity of the underlying problems increases, the objective values of the CP models increase as well, since the underlying problems are the minimization problems.

As power models given in Section II indicate relationship between energy consumption and the number of hops packets that traveled between the PEs. Hence it is assumed that the actual power usage will be a function of the current objective function of the Stage I CP model which is the sum of information flow multiplied by the distance between the PEs. Therefore, our formulation successfully addresses the issue of energy-aware design of NoC architecture. The completion time of all of the tasks in a task graph is used as the objective function in the Stage II CP model. Some other objectives such as task (job) tardiness or lateness can be used as well. But it is still safe to use universally the latest task completion time as the objective function to minimize in scheduling problems. Notice that, none of the scheduling problem was found to be infeasible, all of them returned the best solution found.

The distributions of latency at packet level for each instance are depicted in Fig. 6 for 3×3 mesh architecture. Due to the network congestions by increasing complexity of CTGs, it is expected to have higher latencies for the packet transmissions. We can conclude from Fig. 6 that the variation of the packet latency increases as well by increasing the complexity of the CTG with respect to the congestion of the network.

Fig. 7, Fig. 8 and Fig. 9 summarize the results from the 4×4 mesh architecture. When we compare Fig. 5 and Fig. 8, we can see the benefits of the extra resources (i.e. PEs) effects on the clock time. As expected the 4×4 mesh architecture completes the tasks earlier than the 3×3 mesh architecture. On the other hand, the application scheduling task becomes more complex by increasing the mesh size due to the extra decision variables and the resources in the model as seen in Table I. However, the CP models were able to return the results within the same run-time limits as the experiments for the 3×3 mesh architecture.

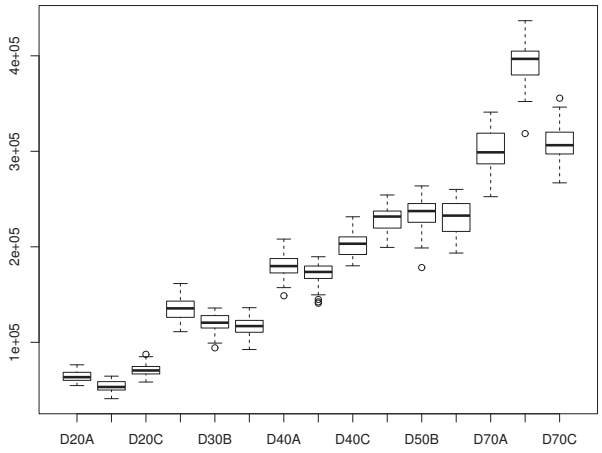

Fig. 4: The Distribution of Stage I 3×3 Mesh Objective Values for Each Instance

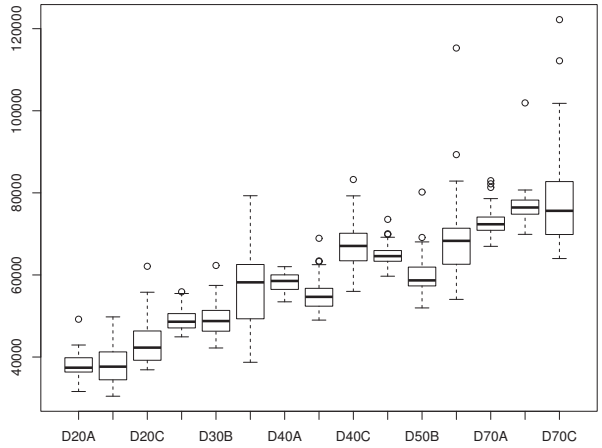

Fig. 5: The Distribution of 3×3 Mesh Completion Times (Clock Cycle Time) for Each Instance

V. Conclusion and Future Work

A CP-based two-stage model is proposed to solve the mapping and the application scheduling problems listed in [2] as outstanding research problems in application modeling and optimization for NoC communication. The major benefit of using CP is the clarity and understandability of the models. The CP modeling is more flexible than the MIP on many challenging optimization problems including mapping and scheduling.

We successfully experimented our model on various benchmark datasets generated by TGFF. Deterministic XY-routing with wormhole switching is successfully used for the task and the data communication scheduling. We were able to use flexibility provided by the CP modeling tools. The usage of a deterministic routing algorithm in our implementation can be justified by the adverse effects of the uncertainties of the routes taken and the cost of modeling to prevent the dead-lock and the live-lock situations in the case of adaptive routing. However, adaptive routing algorithms should be incorporated to the CP models for a realistic application.

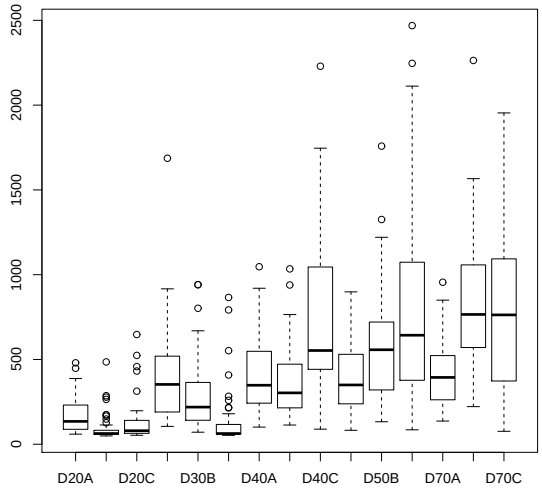

Fig. 6: The Distribution of 3×3 Mesh Latencies (Clock Cycle Time) for Each Instance

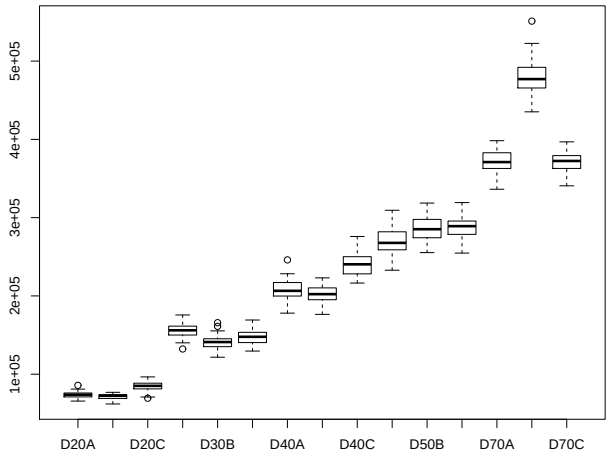

Fig. 7: The Distribution of Stage I 4×4 Mesh Objective Values for Each Instance

Certainly, the models that handle the router buffers are more plausible in terms of being closer to the realistic NoC designs. However, the flexible data structure proposed in Fig 3 and the mathematical models of the routers [18] can greatly help in achieving so in future studies. In addition, we can join both the mapping and the application scheduling problems as previously done in [9] instead of using a two-stage model, but one can achieve it by the CP modeling in a more elegant and simpler way without having excessive complexity.

A logical extension to the homogeneous platform is to study the case of having the heterogeneous PEs. The challenge of

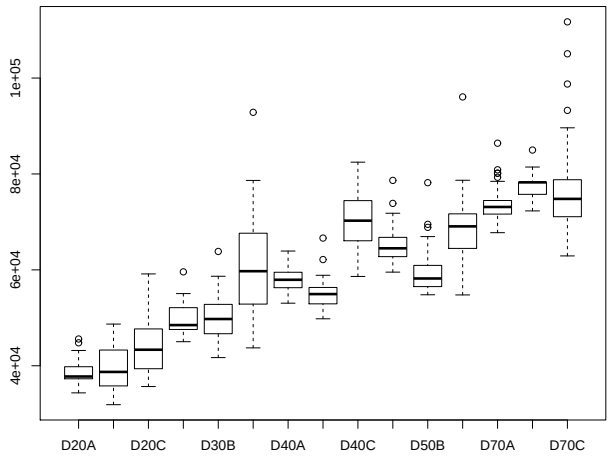

Fig. 8: The Distribution of 4×4 Mesh Completion Times (Clock Cycle Time) for Each Instance

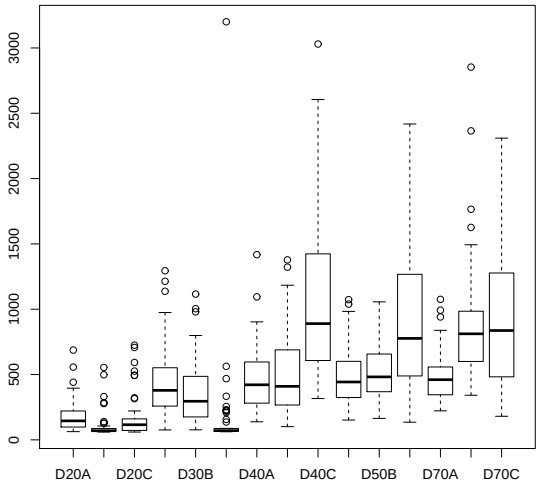

Fig. 9: The Distribution of 4×4 Mesh Latencies (Clock Cycle Time) for Each Instance

the dynamic voltage frequency island (VFI) is open in terms of providing the robust models [19], [11]. Therefore, an extension of the CP model should be in the direction of studying VFI on heterogeneous platforms and adaptive routing. The work in this paper can be extended to the real applications on various processing platforms in the future.

ACKNOWLEDGMENT

Authors would like to thank to Freddy Bolanos for generating TGFF files. First author's visit to UCI was supported by Sakarya University, Turkey and The Council of Higher Education, Turkey.

REFERENCES

[1] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, and A. Pullini, "Networks on chips: from research to products," in *DAC*, S. S. Sapatnekar, Ed. ACM, 2010, pp. 300–305.

[2] R. Marculescu, Ü. Y. Ogras, L.-S. Peh, N. D. E. Jerger, and Y. V. Hoskote, "Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[3] H. Zhang, C. Beltran-Royo, and M. Constantino, "Effective formulation reductions for the quadratic assignment problem," *Computers & OR*, vol. 37, no. 11, pp. 2007–2016, 2010.

[4] E. M. Loiola, N. M. M. de Abreu, P. O. B. Netto, P. Hahn, and T. M. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657–690, 2007.

[5] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular noc architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, 2005.

[6] K. Srinivasan and K. S. Chatha, "Integer linear programming and heuristic techniques for system-level low power scheduling on multiprocessor architectures under throughput constraints," *Integration*, vol. 40, no. 3, pp. 326–354, 2007.

[7] L. Michel and P. Van Hentenryck, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010, ch. Basic CP Theory: Search. [Online]. Available: http://dx.doi.org/10.1002/9780470400531.eorms0088

[8] C. Ababei, H. S. Kia, O. P. Yadav, and J. Hu, "Energy and reliability oriented mapping for regular networks-on-chip," in *NOCS*. IEEE Computer Society, 2011, pp. 121–128.

[9] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "Unism: Unified scheduling and mapping for general networks on chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1–14, 2011.

[10] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel, "Exploring noc mapping strategies: An energy and timing aware technique," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, ser. DATE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 502–507. [Online]. Available: http://dx.doi.org/10.1109/DATE.2005.149

[11] P. Ghosh and A. Sen, "Efficient mapping and voltage islanding technique for energy minimization in noc under design constraints," in *SAC*, S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2010, pp. 535–541.

[12] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Trans. VLSI Syst.*, vol. 14, no. 4, pp. 407–420, 2006.

[13] L. Benini, D. Bertozzi, A. Guerri, and M. Milano, "Allocation and scheduling for mpsocs via decomposition and no-good generation," in *Principles and Practice of Constraint Programming - CP 2005*, ser. Lecture Notes in Computer Science, P. van Beek, Ed. Springer Berlin / Heidelberg, 2005, vol. 3709, pp. 107–121, 10.1007/11564751_11. [Online]. Available: http://dx.doi.org/10.1007/11564751_11

[14] M. Ruggiero, A. Guerri, D. Bertozzi, M. Milano, and L. Benini, "A fast and accurate technique for mapping parallel applications on stream-oriented mpsoc platforms with communication awareness," *International Journal of Parallel Programming*, vol. 36, pp. 3–36, 2008, 10.1007/s10766-007-0032-7. [Online]. Available: http://dx.doi.org/10.1007/s10766-007-0032-7

[15] R. P. Dick, D. L. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *CODES*, G. Borriello, A. A. Jerraya, and L. Lavagno, Eds. IEEE Computer Society, 1998, pp. 97–101.

[16] W.-J. van Hoeve and I. Katriel, *Handbook of Constraint Programming*. Elsevier, 2006, ch. 6 Global Constraints, pp. 169–208.

[17] P. Baptiste, P. Laborie, C. L. Pape, and W. Nuijten, *Handbook of Constraint Programming*. Elsevier, 2006, ch. 22 Constraint-Based Scheduling and Planning, pp. 761–799.

[18] Ü. Y. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 2001–2013, 2010.

[19] Ü. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. VLSI Syst.*, vol. 17, no. 3, pp. 330–341, 2009.