Multi-Objective Tabu Search Based Topology Generation Technique For Application-Specific Network-on-Chip Architectures

Anita Tino and Gul N. Khan
Department of Electrical and Computer Engineering
Ryerson University, Toronto, Canada

Abstract – *This paper presents a power and performance multi-objective Tabu Search based technique for designing application-specific Network-on-Chip architectures. The topology generation approach uses an automated technique to incorporate floorplan information and attain accurate values for wirelength and area. The method also takes dynamic effects such as contention into account, allowing performance constraints to be incorporated during topology synthesis. A new method for contention analysis is presented in this work which makes use of power and performance objectives using a Layered Queuing Network (LQN) contention model. The contention model is able to analyze rendezvous interactions between NoC components and alleviate potential bottleneck points within the system. Several experiments are conducted on various SoC benchmark applications and compared to previous works.*

Keywords – Network-on-Chip, Topology Generation, Tabu Search, Layered Queuing Networks, Contention

I. INTRODUCTION

The Network-on-Chip (NoC) concept has emerged as a result of the limitations posed by future bus-based systems found in Multiprocessor System-on-Chip (MPSoC) architectures. As forthcoming MPSoC designs continue to grow in size and complexity, hundreds of memory and processing elements are expected to communicate at a gigascale speed, consuming low power and minimal on-chip area. NoC systems consist of cores and on-chip routers to execute packet-switched communication. Cores access the network by means of network interfaces, and have their packets forwarded to their destination through a multi-hop routing path [1]. NoCs replace the busses in MPSoC systems with routers and links, where packets communicate simultaneously between the cores, improving overall system performance. Furthermore, the power consumption in the MPSoC is reduced by using shorter links as opposed to long shared busses. The NoC approach has thus shown improvements in issues such as performance, power and area on a chip [2]. As a result, NoC research and design have gained significant attention in both industry and academia.

Previous works in NoC design have dealt with generating topologies with minimal power dissipation and/or maximum performance. However, when taking actual SoC applications into consideration, focusing strictly on power could imply that the performance constraints in a system will not be met. Similarly, concentrating solely on maximizing system performance can lead to an unwarranted amount of power dissipation. Therefore, a tradeoff between power and performance is needed to design NoC topologies. Regular topologies such as mesh, torus, butterfly fat tree etc., result in poor performance due to increases in power usage and hardware area overhead. MPSoCs often comprise of different system requirements given an application. Therefore scalability is not as significant as power, performance, and area factors that a system must maintain. It has become evident through research that application-specific NoC architectures are superior in comparison to regular topologies in terms of power consumption and NoC resources [3]. As a result, irregular NoC topologies have become the preferred solution when minimizing area and power consumption on a chip.

This paper presents a multi-objective approach to topology design, employing a Tabu Search (TS) based technique to meet the power and performance requirements of an application. The proposed NoC synthesis method takes into account constraints such as network latencies, power consumption, and dynamic effects to generate application-specific topologies which cater to system requirements. A new contention analysis method is also presented in this work which uses a Layered Queuing Network (LQN) model to observe rendezvous interactions among the NoC components and identify possible contention points. The analyzer aims to relieve contention in an optimal manner by temporarily inserting virtual channels (VCs) to evaluate performance improvements and additional power consumption incurred by the network prior to actual VC insertion. By using this technique, VC resources are inserted carefully to provide the system with low resource costs, low power consumption, and high performance.

II. PREVIOUS WORK

A number of researchers have utilized optimization methods to design NoC topologies. Ahonen et al [4] apply a simulated annealing (SA) method to optimize NoC synthesis using the objectives of power and latency with the aid of its OIDIPUS design automation tool. The cores in the overall topology are partitioned into two major parts to form two connected ring topologies, where swaps between the partitions are made until a stopping criterion is met. The method can only invoke one objective at a time, and when optimizing for power only considers wirelength while disregarding other factors such as router consumption. The overall technique displays that SA and system partitioning degrade the optimized level of achievable block placement within a topology. By aiming to lower the cost of the partition, the method increases the traffic within the resulting partitions and limits the boundaries of possible solutions in the search space.

The work employed by Beraha et al [5] searches for a way to optimize both power and performance, also using a SA optimization technique. The general method aims at attaining a factor trade-off by generating a topology with minimal power consumption which meets the required latency demands of the application. The SA randomly assigns the initial mesh topology, where swaps are made in a random fashion between two cores to generate another mesh topology with a lower cost. Contention analysis is taken care of subsequent to designing the on-chip network. Therefore, the method does not account for the dynamic effects such as contention when analyzing throughput and latency. As a result, the minimum performance requirements may not end up meeting the constraints needed by the application. Furthermore, the generator does not maintain accurate values for the power dissipation of the system using a floorplanner or power models, but simply relies on the cost function which employs router area and bandwidth of links to account for NoC power.

Ascia et al [6] use power and performance as objectives in order to optimize mesh based NoC architectures using a state-machine and NoC simulator to evaluate mapping alternatives. The exploration engine evaluates core mapping schemes using a Genetic Algorithm (GA) to determine the next move until the stopping criteria is met. The final GA outputs two Pareto curve solutions of high performance and low energy mappings. Although the work by Ascia et al aims at designing NoCs with the objectives of power and performance in mind, the experimental outcomes represent a performance or power mapping. Therefore, the work does not combine both objectives when aiming to create an on-chip network, but instead provides the user with a mono-objective topology. In terms of power optimization, the work does not assess the effect of link power consumption within the on-chip network.

```
        GENERAL TABU SEARCH ALGORITHM
1.   Generate initial solution $s = N(s)$
2.   Evaluate current solution conditions
3.   $TL(s)$ = {}  //initial empty Tabu List
4.   WHILE stopping criteria NOT met
     DO
5.       Identify $s'$ = neighbourhoodSet()
6.       Move to the temporary solution $s'$
7.       Evaluate $s'$ solution
8.       OptimalityCheck($s'$, $AL(s)$, $TL(s)$)
9.       IF optimal solution
10.          Place solution as last optimal $TL(s)$ entry
11.          Update current solution, $N(s) = s'$
12.          IF Constraints satisfied
13.              EXIT
             END
         ELSE
14.          Place as a non-optimal $TL(s)$ entry
15.          Refer to $AL(s)$ to revert back to last optimal solution
         END
     END
```

Figure 1: General Tabu Search Algorithm

Leary et al [7] implement a 3-level GA to represent topology mappings as a set of binary and integer arrays. The algorithm employs genetic operator techniques, a fitness function, and a floorplanner to create an application-specific architecture until a stopping criterion is met. A Pareto curve technique is implemented based on the factors of power consumption, or the amount of routers used in the topology. The designer is given the choice of selecting the best solution of the two mappings. The GA technique in this work randomly generates solutions. However, given that information of communication and system requirements are a priori, there is no logic in generating random solutions. In addition to these facts, due to the random nature of the GA, the topology generation scheme invoked in this work can lead to longer wires and cause invalid solutions within the algorithm. As a result, the GA method can lead to high execution times and not guarantee a global optimal solution. Furthermore, making use of the number of routers as a factor within the GA technique does not directly address the performance of an on-chip network.

III. TABU SEARCH AND NOC TOPOLOGY DESIGN

The TS topology generator assesses all NoC characteristics given in the system core graph. Each step in the NoC generation method analyzes system performance and observes changes in the frequency of operation and power, while optimizing for other constraints within the NoC topology arrangement. The Parquet floorplanner [8] and Orion 2.0 power models [9] are integrated in order to assess various power consumption factors. The topology is run through the system-level floorplanner to assess the wirelengths and area at significant stages of the solution. In cases where the objectives and constraints do not meet the criteria, the TS method aims at discovering a new best solution that will satisfy the objectives. When the system requirements have been met, contention analysis is performed. The on-chip network is modeled as a LQN to analyze possible contention points within the system. The potential bottlenecks are identified through contention analysis using the Layered Queuing Network Solver (LQNS) tool and relieved using virtual channel (VC) insertion.

A. Tabu Search Based Optimization

Tabu search is a meta-heuristic algorithm that employs an aggressive search procedure. The procedure progresses iteratively from one solution to another by moving in a neighbourhood space with the assistance of adaptive memory [10]. TS reaches a better solution by considering the influence of a move within the neighbourhood and incorporating factors of search history and the problem context [11]. The TS method escapes the trap of local optimality by using its ability to retrieve prior optimal and non-optimal solutions from memory. By keeping track of solutions within the search, it is possible to locate the global optimal solution with less computational effort and time as compared to other optimization methods.

As seen in Figure 1, the algorithm commences with an initial feasible solution *N(s)* and explores other possible neighbourhood space solutions in *neighbourhoodSet()*. TS inquires with its short term memory lists in order to prevent the reversal of recent moves as performed in *OptimalityCheck*(*s*', *AL(s)*, *TL(s)*). The function ensures that the new temporary solution *s*' coincides with the Aspiration List *AL(s)* and Tabu list *TL(s)* to find a new local or global optimal solution. In order to escape a local minimum, *AL(s)* refers to *TL(s)* to understand whether the current solution is inferior to a previous solution. If the latter is true, the algorithm reverts back to the state it was previously in and places the current solution into the *TL(s)* to escape a local minimum. The algorithm continues until the stopping criteria are met.

The memory structures within the TS method play a fundamental role in strategizing for optimal solutions and provide a quality solution by supporting multiple objectives during the search. Memory in the TS method relies on the four principles of recency, frequency, quality and influence [10]. The TS method makes use of explicit and attributive memory types to guide the search in finding a solution which employs the four principles.

Explicit Short Term Memory: Explicit memory directs the search towards an influential and quality-based solution, keeping record of past solutions within the TS to avoid cyclic behavior. Explicit memory services are provided by the Aspiration List, *AL(s)*, and Tabu List, *TL(s)*. The *TL(s)* is responsible for keeping track of the non-optimal solutions to prevent the algorithm from revisiting previous solutions. During each TS iteration, the algorithm checks the *TL(s)* entries to verify that solution *s'* is an optimal solution, and that its current evaluated criteria does not match other non-optimal entries. The Aspiration criterion is satisfied if a move yields a solution better than the best obtained so far. The *AL(s)* is therefore responsible for overriding restricted *TL(s)* entries if the outcome of the move under consideration is sufficiently desirable [10]. The final type of explicit memory is known as the Candidate List *CL(s)*. *CL(s)* is responsible for generating a list of possible moves within the neighbourhood. More advanced TS methods use candidate list strategies to help narrow the examination of solutions in order to achieve a high quality solution within a shorter period of time and a reasonable amount of effort.

Attributive Long Term Memory: Attributive memory acts as long term memory known as Frequency-Recency based memory (FR-Memory). This memory is made use of during neighbourhood exploration for a new possible solution. FR-Memory encourages the search to explore different regions within the neighbourhood and allow for the diversification amongst the different feasible solutions. FR-Memory keeps track of the frequency of moves within each area of the neighbourhood, and the recency of the vertices which have been previously moved.

B. *NoC Design Flow*

1) Input Model

It is assumed that the target application selected by the user has been mapped to processing cores, and that the corresponding communication volume requirements between the cores have been determined statically. The application can be specified as a directed graph *G(V,E)*, where:

- → Each vertex $v_i \in V$ represents a core within the graph.
- → The communication between vertex *i* and *j* represent a directed edge *(v_i, v_j)*, expressed as $e_{i,j} \in E$.
- → The weight found on an edge $e_{i,j}$ denoted by $b(e_{i,j})$ characterizes the bandwidth.
- → A destination vertex (core) d_x, where $d_x \in V$ may have 1 to many sources cores s_x.
- → The source vertex $s_x \in V$, and $\forall x \in 1...N$.
- → N represents the number of cores in the core graph.

The core graph is a model based on the cores and their communication, with details about the source and destinations used within the system.

Switch and link power models are used in the generator and are based on 65nm technology established on Orion 2.0 values. The floorplanner performs wirelength minimization, compaction on the final solution to minimize the area within the design, in addition to detecting timing violation within the system. Characteristics such as average arbitration delay, packetization and de-packetization delays, maximum number of ports, flits per packet, and maximum operating frequency can also be specified. Although the TS technique is successful in finding optimal solutions, its downfall occurs in memory allocation. Given a problem space with a predetermined amount of cores and edges, the user can specify the required amount of *TL(s)* entries needed for a successful optimal solution to be reached.

2) Tabu Search Based Topology Generator

Initial Tabu Search Solution: Given the constraints specified by the user, the topology generator creates an initial NoC topology, referred to as *N(s, f, P)*. Each vertex V_i in the core graph is assigned to a Network Interface (NI). The initial NoC topology generation is referred to as the crossbar approach, where all the NIs/vertex cores are connected to one central router [12]. An initial NoC frequency is determined based on all the connections within the on-chip network. The iterations within the TS method divide the initial large crossbar router, where preference is given to grouping frequently communicating cores within the same routers. Router connections are based on the communication requirements between source cores s_x, and their respective destination cores d_x.

Problem Formulation: Let *N(s, f, P)* represent the current feasible NoC topology solution *s* consuming power *P* at a frequency *f*, and *N(s)* express a new possible solution *s'* within the neighbourhood set. We define a TS based *TL(s)* that contains non-optimal solutions, and *AL(s)* that is responsible for consulting the Tabu list to ensure that *s'* is optimal and more desirable than the previous encountered solutions. Thus in order for the new solution *N(s) = s'* to be an optimal solution φ(s), and a possible current feasible solution *N(s, f, P)*, the following must be satisfied:

$$\varphi(s) = \{N(s) \cap TL(s)\} = \emptyset \qquad (1)$$

If the $\varphi(s)$ condition in (1) holds true, *N(s)* is disjoint with the Tabu list set and is an optimal entry with respect to *TL(s)*. *N(s)* is then consulted *AL(s)* to verify that it is an element which is optimal

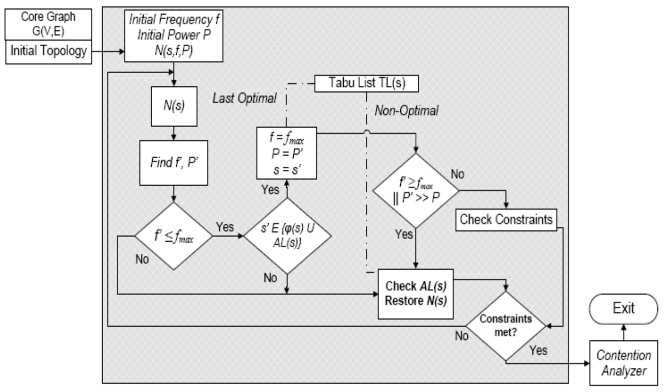

Figure 2: Tabu Search Topology Generation Algorithm

with respect to the previous encountered solutions. This can be expressed as:

$$N(s) \in \{\varphi(s) \cup AL(s)\} \qquad (2)$$

If expression (2) holds true, the old solution *s* can be updated to the new solution s', and frequency *f* can be changed to reflect the new frequency *f'*. The TS based NoC design algorithm of Figure 2 iterates through feasible solutions, each time identifying a new possible topology configuration *N(s)*. The method rearranges the topology to the new solution in order to assess the factors within the system. Latency, NoC frequency of operation, and router port constraints for the possible new NoC topology solution are evaluated during each iteration in '*Check Constraints'* to verify whether the stopping criteria have been met. The power is also determined for each new solution, where the NoC design method ensures that the power consumption has not significantly increased from the last *N(s, f, P)* move. If the operating frequency exceeds the maximum possible frequency, or the power is significantly increased within the current possible solution being evaluated, *AL(s)* is referred to and the last optimal solution is restored. The undesired solution is placed into *TL(s)* and the algorithm attempts to obtain another neighbourhood solution.

Frequency is determined by evaluating the bandwidth and datawidth values, but also considering factors related to a pipelined wormhole network. In order for a flit to traverse through a network, latencies must be considered as it can greatly affect the frequency of operation between communicating cores. As a result, the user must specify the intended arbitration delay within a router, flits per packet, and the intended packing and unpacking delays of the NIs. As discussed by Dimitriu and Khan [12], the transfer latency in a pipelined wormhole network can be expressed as:

$$\mathrm{D_{lat}} = \mathrm{D_{pk}} + (\mathrm{N_{fl}} - 1) + \sum_{\mathrm{i=0}}^{\mathrm{n}} \mathrm{D_{arb}} + \mathrm{D_{dpk}} \qquad (3)$$

Here, D_{lat} represents the latency of a packet consisting of N_{fl} flits, traversing to its destination with *n* hops. D_{pk} and D_{dpk} represent the packetization and de-packetization delays within the network interfaces respectively. Finally D_{arb} represents the arbitration delay within each switch throughout the *n* hops incurred in the network. These latency values are then converted into time units and incorporated to assess the frequency of operation.

Neighbourhood Selection Structure: The initial topological solution in this work implements a crossbar and can be considered to be a poor solution as a large router consumes a considerable amount of power. Furthermore, contention within this central router is also at the highest since all connected cores contend for the same crossbar. Finding a new solution within the neighbourhood of possible solutions is therefore greatly desired. The *CL(s)* and the candidate list strategy employed in this work, known as the Successive Filter Strategy (SFS), assist the TS in

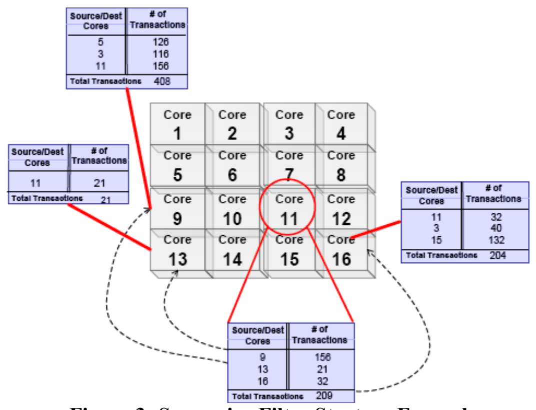

Figure 3: Successive Filter Strategy Example

finding a new solution N(*s)* within the neighbourhood. A high quality solution in this work results in a low power, high performance topological arrangement which does not exceed the maximum frequency of operation. Hence, the SFS initially aims at first filtering the cores with low transaction rates since their moves have less effect on the frequency and performance of the topology as compared to the highly utilized cores. Power consumption is also decreased in comparison to the initial crossbar approach, as the large central router is divided into smaller components.

The head of the *CL(s)* is the core chosen by the SFS based on either the initial low transactional cores, or the neighbourhood cores being kept track of by the FR-Memory. Given the initial head candidate list condition, low transactional cores are chosen by employing the following expression:

$$MinTrans = min_n \left\lceil V_n \left(\sum N_{tr}(s_x) + \sum N_{tr}(d_x) \right) \right\rceil \qquad (4)$$

Given a vertex/core n, N is the total number of vertices/cores in the core graph, where $n = \{1,2,\ldots,N\}$. N_{tr} is the number of source, s_x, and/or destination, d_x, transactions that the vertex V_n is expected to incur. X is the total amount of sources or destinations for the respective core n, where $x = \{1,2,\ldots,X\}$. $V_n(f)$ represents vertex n and its expected total number of transactions f.

The candidate list is formed initially based on the minimum ceiling function of the vertex which exemplifies the lowest amount of transactions with other cores. In addition to this, the candidate list and SFS also verify with the FR memory whether the core has been previously selected, which allow for other low transactional cores to also take the head candidate position. The SFS then filters through the cores which communicate with the head of the candidate list. Subset arrangements of these cores are formed in different combinations, where the topology generator selects the subsets and configures the topology such that the cores are positioned according to the SFS combinations. The subset core combinations are then evaluated for solution quality. Let:

→ $\pi \in \Pi$, where Π is a set of positions in the search space.
→ $\pi(j)$ represent core j attaining the head candidate position.
→ $\Omega(s)$ denote the set of possible moves that core j can have, when core j has occupied the position $\pi(j)$.
→ m signify all possible combination of moves formed by the SFS.
→ $\Omega(s)$ be divided into subsets $\Omega(1,s)$, $\Omega(2,s)$, ... $\Omega(m,s)$, where $\Omega(1,s)$ denotes the 1st subset move in the possible set of total moves generated by the topology generator etc.

An example of the SFS applied to a 16 core application is given in Figure 3. The head of the candidate list is core 11. Core 11 performs transactions with cores 9, 13 and 16. When finding a new *N(s)*, core 11 has occupied the position $\pi(11)$ in the possible set of moves $\Omega(s)$. The SFS then generates a set of m possible

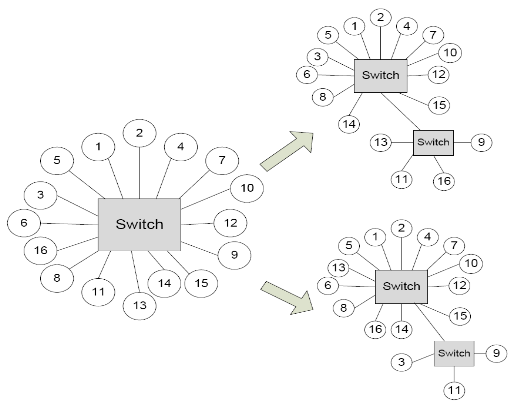

Figure 4: Possible Subset Configurations of Successive Filter Strategy

combination of moves given core 11, analyzing cores 9, 13 and 16 and their respective transactions with other cores as well. The resulting topological arrangements are then evaluated for quality given the factors of operational frequency, power, and port constraints. Two possible subset combinations $\Omega(1,s)$ and $\Omega(2,s)$ for the SFS example is shown in Figure 4.

IV. Contention Analysis

LQN and Performance: LQN was developed as an extension to the concept of queuing networks using a layered structure approach [13]. Employing LQNs can determine factors of system performance. A system can have multiple functionalities, where each function is carried out by a subcomponent. A subcomponent can also perform various tasks. Contrary to queuing networks, LQN is modeled as subcomponents, which are interconnected to simulate the functionality of the entire system. These subcomponents are then divided into layered tasks which communicate amongst each other. The role of an LQN is to model the wait-and-reply rendezvous interactions that the multiple task layers can experience. An LQN task can also consist of multiple entries that characterize its different operations.

An example of an LQN representing a real benchmark is shown in Figure 5. The LQN of Figure 5 is an example of a subcomponent of the MPEG-4 Decoder, where each element is clearly labeled. The first layer in this example consists of three tasks, where each of these tasks contain a thinking time Z. These tasks are referred to as reference tasks (represented by a square) which await other tasks to complete. Task 4 and 5 are non-reference tasks (represented by parallelograms), where the number in brackets within the task characterizes the execution time to perform its functionality. Task 4 contains two entries, E1 and E2, which perform different functions. The values on the arcs represent the number of transactions that a task *i* makes to a task *j*.

Contention can occur in a network when a task is heavily utilized, in turn creating bottleneck points and limiting the throughput of a system. Utilization can therefore be defined as the fraction of time when the task is busy or blocked by a rendezvous [13]. As an example, the heavy utilization of Task 4, E2, in Figure 5 is noted. By dividing the network into subcomponents and observing rendezvous, it is possible to pinpoint and relieve various contention points to achieve higher throughput and increase system performance. In the proposed methodology for contention analysis, the NoC network is divided into subcomponents/sub-networks, where the router-to-router connections and respective interconnected cores are modeled as LQNs to pinpoint potential bottlenecks during the topology synthesis stage. LQNS is used to evaluate factors of utilization and throughput in the topology.

Table 1: LQN/Contention Model Conversion

LQN Element	Contention Model Element
Reference Task	Source Core
Non-Reference Task	Router/ Destination Core
Thinking Time (*Z*)	Packetization Delay
Execution Time of Task	Reference Task Packetization Delay Non-Reference Task De-packetization Delay Router Arbitration Delays
Number of Transactions	Number of packets sent from component *i* to *j*

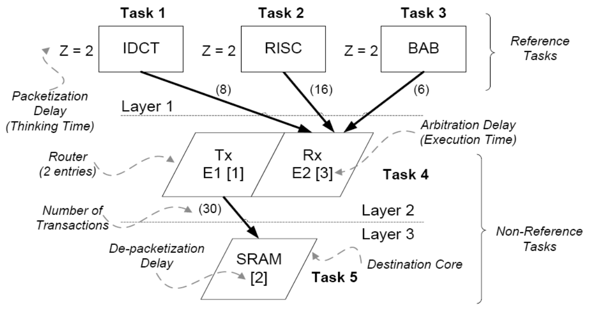

Figure 5: LQN Subcomponent for MPEG-4 Decoder

LQN/Contention Model: The LQN model conversion is presented in Table 1, where each LQN element models an element in the contention analysis method. Each source core in the NoC is considered as a reference task which awaits a response from the lower layers. The lower layers are considered to be the router's ports and/or destination cores. The source core and its respective NI have a packetization delay, and are modeled as the reference task's thinking time *Z*. Each router port task is modeled as a non-reference task with two entries representing the router operations of receiving (Rx) and transmitting (Tx) data. The execution time within the router entries represent the arbitration delay.

VC Insertion Technique: The contention analyzer is given a finite amount of VC resources imposed on by the user, and aims at equally distributing the VCs according to the hot spots within the system. The generator models the NoC topology subcomponents using the LQN/contention model. Throughput and utilization factors are noted, and the bottleneck points are verified. Thereafter, temporary VCs are inserted at the bottleneck points to model the task once again and evaluate the improvement in performance and extra power dissipation. The following criteria must be satisfied in order for the temporary VC insertion to take permanent effect:

→ The performance improvement is greater than the extra power dissipation that the on-chip network will experience.
→ There are enough VC resources for the insertion to take place.
→ The new frequency of operation will not exceed the maximum allowable frequency.

If all criteria are satisfied, the system permits the temporary VC to be inserted into the system at the designated spot to relieve contention. The algorithm inserts multiple VCs if all criteria are satisfied and there are enough resources. Given the scenario that all the contention points have not been analyzed, or one of the criteria has not been satisfied, the analyzer proceeds to search for and relieve other potential bottleneck points within the topology.

V. COMPLEXITY ANALYSIS

Given a solution space of *N* cores, determining a move within the TS given the constraints imposed by the SFS yields $N(N-1)/2$ moves, expressed as $O(N^2)$. The swaps needed to place the cores in the new topological arrangement results in a complexity of $O(1)$, where $O(N)$ time is needed to evaluate the *N* cores. Given *k* total iterations within the search, and an average *TL(s)* search time of *i*, the overall complexity of the proposed method can be expressed as $O(k(N^2+N+ i))$, further simplified as $O(N^2 + N)$, assuming *k* and *i* as negligible.

VI. EXPERIMENTAL RESULTS

The TS topology generator is applied to five different multimedia and networking benchmark applications: *Video Object Plane Decoder (VOPD – 12 cores), Multi-Window Display (MWD – 15 cores), Network Communication System (NCS – 15 cores), MPEG-4 Decoder (12 cores), and the Set Top Box (25 cores)*. We refer readers to the work of various researchers for the application core graphs [12,14,15,16]. The topology generator implementation for the proposed method was tested using a computer with a 1.66 GHz atom processor and 1 GB of RAM, running a Linux operating system. Network Interfaces are taken into account within the core area, where each NI is estimated to be approximately 0.2 mm^2 [17]. The routers are modeled as individual components within the floorplanner. Power is calculated with virtual channel consumption being considered, where router port buffers are sized for 4 flits. Topologies in this work were generated using a restriction of 6 ports per router and a maximum operating frequency of 1 GHz.

Table 2: Topology Technique and Area Comparisons

Application	Method	Topology	Area (mm^2)
MPEG4 Decoder	Tabu	App Specific	4.13
	1	App Specific 1 [12]	11.89
	2	App Specific 2 [12]	4.43
VOPD	Tabu	App Specific	2.55
	1	App Specific [18]	3.11
	2	App Specific [3]	1.93
Set Top Box	Tabu	App Specific	6.41
	1	App Specific[7]	6.19
	2	App Specific[19]	6.35
NCS	Tabu	App Specific	4.58
	1	ANOC [7]	No Solution
	2	App Specific [7]	No Solution
MWD	Tabu	App Specific	4.42
	1	App Specific 1 [12]	9.00
	2	App Specific 2 [12]	10.87

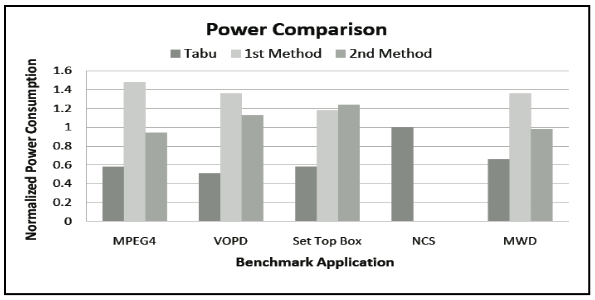

Figure 7: Power Dissipation Benchmark Comparison

The benchmarks have been compared to previous NoC topology design techniques [3,7,12,18,19]. Due to inconsistencies between power libraries used in the various works, the topology layouts were redesigned with the 65nm library used in this work to accurately compare power dissipation and area values. All topologies required less than one minute to be generated and analyzed. The NoC area distributions for the techniques are presented in Table 2 and take network components such as NIs and routers into account. The ANOC and GA technique employed in the work of Leary et al [7] was unable to generate solutions for the NCS benchmark. Analysis of the results display a reduction of 2.03x in power dissipation compared to other topology generation techniques, with comparable values of on-chip area distribution. Figure 7 compares the normalized power consumption of the various methods to their respective applied benchmarks.

The throughput oriented topology generation technique of Dumitriu et al [12] was used to compare performance results. The MPEG4 and MWD used in [12], and NCS benchmark which required VC insertion were assessed for system performance. The NoC routers invoked in both [12] and in this work use round-robin arbitration amongst all the ports to prevent the system from starvation. Livelock is prevented by incorporating minimum length paths from source to destination cores. The topologies were tested using an in-house cycle accurate NoC SystemC simulator.

The throughput values for the different techniques are presented in Figure 8. The VC insertion technique for the MPEG4 Decoder displayed a 17% improvement in throughput due to a 5% increase in power, where the NCS networking benchmark demonstrated a 67% performance improvement as a result of a 13% dissipation increase. The comparison of this work to the work of Dumitriu et al showed a slight throughput improvement of 0.57x and 0.84x for the MPEG4 Decoder and MWD application respectively. This signifies that the multi-objective power and performance technique presented in this work was capable of reaching and exceeding the performance levels that a mono-objective throughput oriented topology generator was able to achieve.

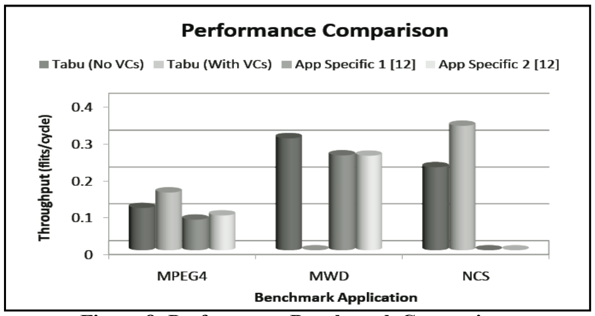

Figure 8: Performance Benchmark Comparison

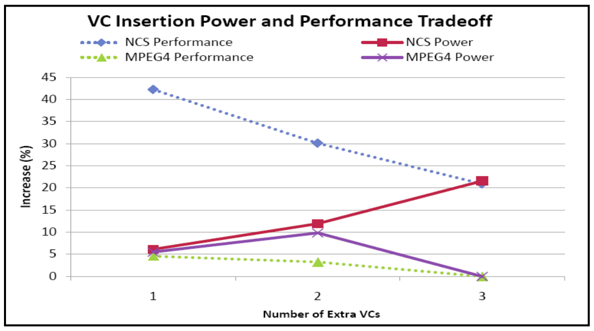

Figure 9: Power/ Performance Tradeoff for MPEG4 and NCS

An example of the power and performance tradeoffs for the MPEG4 and NCS benchmarks applied to the VC insertion technique is depicted in Figure 9. VC insertion for a NCS subcomponent/sub-network initially shows significant increases in throughput as compared to extra power dissipation incurred by the system. The tradeoff is substantially high until 3 VCs are inserted. The additional performance improvement and power consumption levels then intersect, where performance is no longer at an advantage. Therefore, 2 VCs were inserted at the designated contention point, in addition to considering the effects of frequency and available resources. The MPEG4 tradeoff depicted in Figure 9 illustrates the contention at the SDRAM core sub-network, where inserting 1 VC gives a fair trade-off between power and performance. The insertion of 2 VCs leads to a higher power consumption than performance increase. Hence 1 VC is inserted into the system. During topology synthesis, the contention analyzer had estimated a 6.6% and 66.67% increase in performance due to a 5.7% and 21% increase in power consumption for the MPEG4 and NCS benchmarks respectively. The contention analyzer was therefore able to predict the power and performance values within a 96.5% accuracy rate when compared to the actual simulation results.

V. Conclusion

This paper presented an efficient methodology for a multi-objective power and performance NoC design using an automated technique. The work also introduced a LQN/contention model to analyze and relieve various contention points within an on-chip network during topology generation. Results demonstrated a 2.03x reduction in power dissipation in comparison to previous works, with performance values which were comparable to a mono-objective throughput oriented topology generation technique. The contention analyzer was able to predict the power and performance tradeoff for VC insertion with a 96.5% accuracy rate during topology synthesis. The contention analyzer was thus successful in increasing performance and eliminating contention points while considering power dissipation and system requirements.

References

[1] Bertozzi, D., Benini, L.," Xpipes: a Network-on-Chip Architecture For Gigascale Systems-on-Chip," *IEEE Circuits and Systems Magazine*, Vol. 4, Issue 2, pp. 18 – 31, 2004.

[2] Goosens, K., Dielissen, J., Radulescu, A., “AEthereal Network-on-Chip: Concepts, Architectures, and Implementations,” *IEEE Design and Test of Computers*, pp. 414-421, 2005.

[3] Murali, S., Meloni, P., Angiolini, F., Atienza, D., Carta, S., Benini, L., De Micheli, G., Raffo, L., "Designing Application Specific Network on Chips with Floorplan Information," *ICCAD*, pp. 355-362, 2006.

[4] Ahonen, T., Siquenza-Tortosa, D., Bin, H., Nurmi, J., “Topology Optimization for Application-Specific Networks-on-Chip,” *Proc. SLIP*, pp. 53 – 60, 2004.

[5] Beraha, R., Walter, I., Cidon, I., Kolodny, A., “Leveraging Application-Level Requirements in the Design of a NoC for a 4G SoC – a Case Study,” *Proc. DATE*, pp. 1408 – 1413, 2010.

[6] Ascia, G., Catania, V., Palesi, M., “Multi-Objective Mapping for Mesh-Based Networks-on-Chip Architectures,” *Proc. 2nd IEEE IFIP*, pp. 182-187, 2004.

[7] Leary, G., Chatha, K., Srinivasan, Mehta, K., “Design of Network-on-Chip Architectures with a Genetic Algorithm-Based Technique,” *IEEE Trans. VLSI Systems,* vol 17, no. 5, pp. 674-687, 2009.

[8] Adya, S. N., Markov, I. L., “Fixed Outline Floorplanning: Enabling Hierarchical Design,” *IEEE Trans. Very Large Scale Integr (VLSI) Syst.,* vol. 11, no. 6, pp. 1120-1135, 2003.

[9] Samadi, K., Kahng, A., Li, B., Peh, L., "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," *GSRC Annual Symposium*, September 2008.

[10] Glover, F., M. Laguna, "*Tabu Search*," Kluwer, Norwell, MA. 1997.

[11] Xu J., Chiu S., Glover F., “Optimizing a Ring-Based Private Line Telecommunication Network Using Tabu Search,” *Management Science,* Vol. 45, No. 3, pp. 330-345, 1998.

[12] Dimitriu, V., and Khan G. N., "Throughput-Oriented NoC Topology Generation and Analysis for High Performance SoCs," *IEEE Trans. VLSI Sys.*, vol. 17, no. 10, pp. 1433-1446, 2009.

[13] Woodside, C., Nelson, J., Petriu, D., Majumdar, S., “The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server Like Distributed Software,” *IEEE Trans. On Computers,* vol. 44, pp. 20-34, January 1995.

[14] Hu, J., Marculescu, R., “Energy-Aware Mapping for Tile-Based NoC Architectures Under Performance Constraints,” Proc. ASPDAC, pp. 233-239, 2003.

[15] Bertozzi, D., “NoC Synthesis Flow for Customized Domain Specific Multi-Processor Systems-on-Chip,” *IEEE Trans. Parallel Distributed Syst.*, vol. 16, no. 2, pp. 113-129, 2005.

[16] Pasricha, S., Dutt, N., Bozorgzadeh, E., Ben-Romdhane, M., “FABSYN: Floorplan-aware Bus Architecture Synthesis,” *IEEE Trans. VLSI Sys.*, vol. 14, no. 3, pp. 241-253, 2006.

[17] Alho, M., Nurmi, J., “Implementation of Interface IP for Proteo Network-on-Chip,” *International Workshop Design Diagnostic for Electronic Circuit Systems*, 2003.

[18] Srinivasan, K., Chatha, K., Konjevod, G., “An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks,” *Proc. IEEE/ACM ICCAD*, pp. 231-237, 2005.

[19] Chatha, K. S., Srinivasan, K., and Konjevod, G., "Automated Techniques for Synthesis of Application-Specific Network-on-Chip Architectures," *IEEE Trans. CAD of Integrated Circuits and Systems*, Vol. 27., No. 8, pp. 1425-1438, 2008.