Improving GA-based mapping algorithm of NoC using a formal model

Vinitha A Palaniveloo[1] Arcot Sowmya[1]

[1] University of New South Wales, Australia
{vinithaap,sowmya}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201335
December 2013

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Network on Chip (NoC) is a sophisticated communication infrastructure designed to interconnect components in a complex system on chip (SoC). NoC provides quality of service (QoS) guarantee to the applications mapped on it. QoS depends on NoC router architecture, NoC communication scheme, application traffic characteristics as well as application mapping strategy.

Applications are mapped to NoC using mapping algorithms that satisfy power/ latency/ bandwidth constraints. The effect of different mapping algorithms on QoS parameters such as average latency, throughput, power and area is evaluated using NoC simulators. The suitable QoS parameter for evaluating mapping algorithms that satisfy bandwidth constraint and minimizes average communication delay is worst-case communication latency. Worst-case latency is a measure of latency upper bound, it provides insight on the latency guaranteed by NoC to the application. However, it is not possible to measure worst-case latency using NoC simulator so analytical models are used. The formal model previously proposed for measuring worst-case latency formally is used here to improve mapping algorithms constraint to bandwidth and latency.

1 Introduction

Applications may be mapped to NoCs using static or dynamic mapping techniques [1]. In static mapping, the mapping is decided off-line before the application executes. In dynamic mapping, mapping is performed on-line during the execution of the application dynamically during run time. Static mapping is generally used in NoCs to avoid the excess communication overhead in dynamic mapping.

The static application mapping approaches can be broadly classified into two: (a) Exact mapping and (b)Search based mapping. Exact mappings are developed either by using greedy mapping that places cores by considering the cost of placement [2] or iterative mapping that run iteratively until the design goal is met [2–4]. Search algorithms however search the mapping space systematically or heuristically for a solution [5]. Genetic algorithms (GA) based mapping is a heuristic search algorithm, which maps applications by transforming existing mapping solution(s) to arrive at better ones [6–10].

Genetic algorithms based mapping techniques converge at a near optimal solution by using a cost function to select off springs for replacing parents in the next generation [9]. Generally analytical NoC models are used in the cost function, however, they are not robust as they abstract architecture specific details in the model [5]. To overcome this drawback, we propose using a formal NoC model [11] in the cost function. A model checker estimates worst-case latency in the formal model by exhaustive state space search traversing all possible scenarios, considering the behaviour of applications during system execution. The formal model is used in the cost function of a simple generational GA here, to check if it enables the algorithm to search for a near optimal solution in fewer generations. A generational GA uses replacement strategy where offspring replaces the parents.

2 Related Work

Applications may be mapped to NoCs using static or dynamic mapping techniques [1]. In static mapping, the mapping is decided off-line before the application executes. In dynamic mapping, mapping is performed on-line during the execution of the application dynamically during run time. Static mapping is generally used in NoCs to avoid the excess communication overhead in dynamic mapping.

The static application mapping approaches can be broadly classified into two: (a) Exact mapping and (b)Search based mapping. Exact mappings are developed either by using greedy mapping that places cores by considering the cost of placement [2] or iterative mapping that run iteratively until the design goal is met [2–4]. Search algorithms however search the mapping space systematically or heuristically for a solution [5]. Genetic algorithms (GA) based mapping is a heuristic search algorithm, which maps applications by transforming existing mapping solution(s) to arrive at better ones [6–10].

Genetic algorithms based mapping techniques converge at a near optimal solution by using a cost function to select off springs for replacing parents in the next generation [9]. Generally analytical NoC models are used in the cost function, however, they are not robust as they abstract architecture specific details in the model [5]. To overcome this drawback, we propose using a formal NoC model [11] in the cost function. A model checker estimates worst-case latency in the formal model by exhaustive state space search traversing all possible scenarios, considering the behaviour of applications

during system execution. The formal model is used in the cost function of a simple generational GA here, to check if it enables the algorithm to search for a near optimal solution in fewer generations. A generational GA uses replacement strategy where offspring replaces the parents.

3 Formal NoC model for worst-case end-to-end latency estimation

A formal model for worst-case latency estimation has been published [11], however a brief outlined is presented here as a prelude. A formal NoC model was developed using PROMELA (PROcess MEta LAnguage), the input modeling language for SPIN (Simple Promela INterpreter) [12] model checking tool. Worst case latency estimation was posed as the problem of checking the reachability of packet destination states in the NoC model. A model checker performs verification by exhaustive state space search traversing all possible scenarios. Hence, the verifier will pass through the path with worst latency at least once during exhaustive search.

In the formal model, each router is modeled as a one state FSM and the behaviours of the functional blocks are implemented as events or functions on the state transitions as shown in Fig 3.1. $[P_0, ..., P_8]$ are the router processes at routers $[R_0, ..., R_8]$. At each clock tick, the router process reads the status of all the input ports $receive\_R0$, injects packet $inject\_pkt\_R0$, performs packet processing such as routing $switch\_R0$ and contention resolution $arbitrate\_R0$, receives packet $sink\_pkt\_R0$ and updates the output ports $transmit\_R0$ before transitioning to the next state.

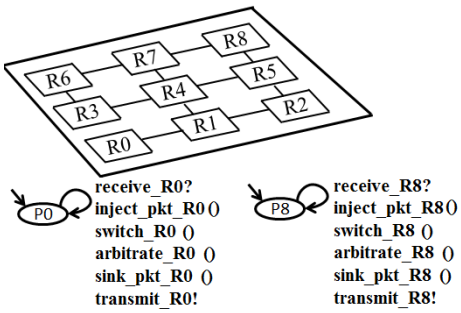

Figure 3.1: On-chip router processes

By using the synchronous modeling already proposed [11], the concurrent operation of router processes at every global clock tick is replaced by a sequence of processes at every clock tick as shown below:

$$P_{t0} = P_0.P_1.P_2.P_3.P_4.P_5.P_6.P_7.P_8 \tag{3.1}$$

Hence, the model of a complete synchronous NoC can be simplified to a single state process with all the functional blocks of all the routers as events on the state transition, shown in Fig 3.2.

As a result, the number of processes required to model an NoC is 4 SPIN processes irrespective of the number of routers in the NoC, overcoming the state explosion problem in model checking large NoC designs. The model was used to estimate worst-case

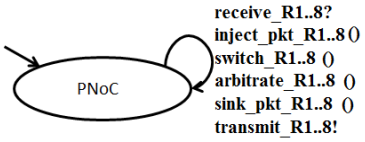

Figure 3.2: Synchronous model of NoC

latencies for large NoCs using SPIN model checker for various traffic rates [11]. The model checker performs verification by unfolding the state machine to specified depths and exploring all the possible paths of execution in the expanded state machine. Since all the paths from the initial state are traversed, in each run it is possible to obtain the worst-case latency. The communication latency of the packets for each run is logged in a file by the verifier, which is used to obtain the worst-case end-to-end latency of all the execution runs.

4 Motivation

Consider the DSP Filter application shown in Fig 4.1, where each block corresponds to a functional core and the edges connecting the cores specify the bandwidth demands of communication between them. The bandwidth demands are in the order of hundreds of MBytes/s.

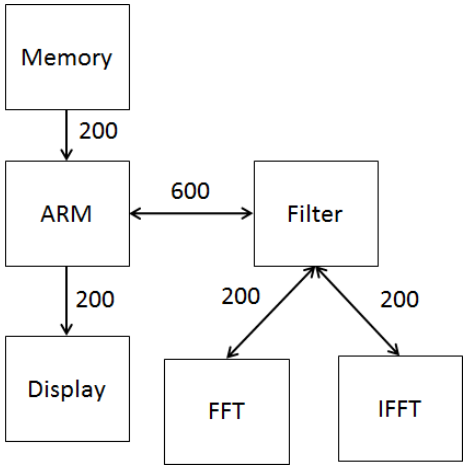

Figure 4.1: DSP Filter Application

Mapping generated by NMAP algorithm satisfies bandwidth constraint and minimizes the communication delay. The algorithm supports both single-minimum path routing and split-traffic routing. The mapping of cores is done based on average traffic between the cores. The mapping obtained for the DSP application using NMAP algorithm [13] is shown in Fig 4.2.

The size of the search space for mapping algorithms explodes as the size of the application increases. For example, the search space size for mapping an application

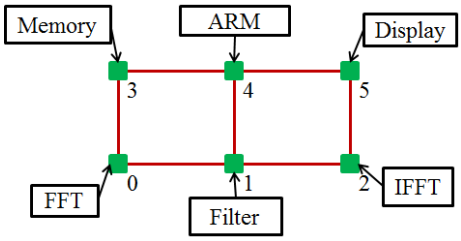

Figure 4.2: Optimal Mapping using NMAP algorithm

with 6 cores on 2x3 mesh NoC can be calculated using the formula for permutation:

$$nPr = \frac{n!}{(n-k)!} \tag{4.1}$$

where,'n' is the total number of items in the sample and 'r' is the number of items selected from the sample. For n,r = 6 the search space size is nPr = 720. Similarly, the search space size is 362880 for an application with 9 cores and it increases exponentially to 2.09227x10^{13} for an application with 16 cores.

The formal NoC model proposed in [14] is used to determine worst-case latency when DSP application mapped on NoC as shown in Fig 4.2. The application traffic rate is specified as input to the application traffic model. In the formal model the peak link bandwidth is assumed 1000 packet/s with one filt in each packet. The DSP application bandwidths are specified as 200MB/s and 600MB/s for peak link bandwidth of 1000MB/s in Fig.5, hence they translate to 200packets/cycle and 600packets/cycles in the formal model, i.e., 20% and 60% packet injection rate for 1000packet/cycle peak bandwidth. 20% and 60% packet injection rates under uniform traffic scenario is specified as one packet every 5 clock cycles and one packet every 2 clock cycles respectively [14].

The worst case latency estimated for mapping solution generated by NMAP mapping algorithm is 3.375 clock cycles. The formal model took 0.016 sec to determine worst-case latency for one mapping solution of the DSP application. As the number of permutations is 720 it is possible to estimate worst-case latency for all the permutation to ensure if the solution produced by NMAP algorithm is right.

he average worst-case end-to-end communication latency was obtained for all the 720 mapping permutations in 17 min 42 sec. The lowest average worst-case latency estimated was 3.375 clock cycles and there were 8 optimal mappings with this lowest average worst-case latency, the list is shown in Table 4.1.

In the table, the mappings are identified by referring to permutation file name and R0, R1, R2, R3, R4 and R5 are router numbers to which the functional modules of the application are mapped in the NoC topology. The mapping solution obtained using NMAP and the simple mapping heuristic are MAP 391 and MAP 701 respectively. As it is possible to design different mapping algorithms for the same constraint [13, 15, 16] and the solution produced by different mapping algorithms may not be the same, it is necessary to compare them quantitatively.

As several optimal solutions are available in the design space hence it is necessary to explore the possibility of finding as many optimal solutions as possible. Hence a mapping exploration algorithm was developed.

Table 4.1: Mappings with lowest worst case latency in 3X2 NoC

Modules	**MAP 93**	**MAP 95**	**MAP 271**	**MAP 276**	**MAP 391**	**MAP 396**	**MAP 699**	**MAP 701**
Memory	R0	R0	R3	R5	R3	R5	R2	R2
Filter	R5	R4	R1	R1	R1	R1	R4	R4
IFFT	R3	R5	R0	R0	R2	R2	R3	R5
FFT	R4	R3	R2	R2	R0	R0	R5	R3
ARM	R1	R1	R4	R4	R4	R4	R1	R1
Display	R2	R2	R5	R3	R5	R3	R0	R0

Mapping evaluation tool A simple mapping evaluation tool was developed using PYTHON, consisting of two key modules (a) mapping generation module and (b) latency analysis module. The mapping generation module uses the SPIN model as the back-end tool to estimate the worst-case latency of a mapping. The architecture of the mapping tool is shown in Fig 4.3 .

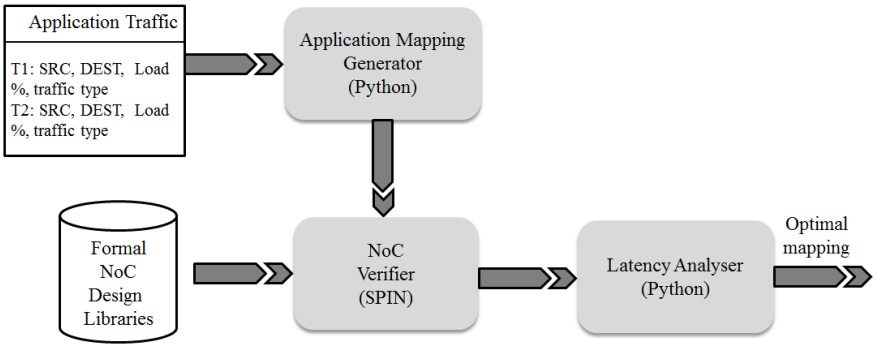

Figure 4.3: NoC verifier architecture

The mapping tool contains a configuration file to specify the size of the application and bandwidth constraint for every functional core in the application. For example, for functional core $T1$ the source (SRC), destination ($DEST$), bandwidth in terms of $\%$ load ($LOAD$) and type of traffic as uniform ($UNIFORM$) is specified in the file. Similarly, the configuration may be specified for all the functional cores $T2, ...Tn$ in the application.

The mapping generation module in the mapping evaluation tool generates the mapping permutations, for each mapping permutation application traffic specification was generated using the configuration file which is used by SPIN NoC model to calculate worst-case latency for the mapping configuration. The worst-case latency was estimated by the SPIN model for each mapping is logged in the file $logging.txt$. The latency analyser module sorts and picks the mapping with lowest worst-case latency from the log file. The log also maintains record of worst case latency estimated on the individual packet paths for every mapping.

The algorithm underlying the mapping evaluation tool for the DSP filter application

is shown in Algorithm 1. The application traffic is specified in the algorithm as $P1$, $P2$,..,Pn, the injection rate of each packet. The list with mapping permutations is generated in A. For each permutation the mapping configuration files was generated with traffic details. SPIN model was used to estimate latency and for each mapping the latency was logged into $latencylog.txt$ file. The suitable mapping was selected from the log file by the latency analyser.

Algorithm 1 Pseudocode of Mapping generation tool in PYTHON

```
Input: Application traffic specification
Output: Optimal mappings
P1: N1, N2, 20%, Uniform traffic
P2: N1, N5, 60%, Uniform traffic
A = list(itertools.permutations(['N1','N2','N3','N4','N5','N6'], 6))
for items ∈ A do
   Map applications to routers in NoC SPIN
   Generate application traffic file with source and destination address of router
   Estimate worst case latency for mapping and log it latencylog.txt
end for
for items ∈ latencylog.txt do
   Sort worst case latencies
end for
Output optimal mapping
```

There are 8 different packet paths in the application. Using the SPIN formal model, it was possible to estimate the worst case latencies on specific latency paths. The worst case latencies on specific packet paths for all the solutions in the optimal solution space are shown in Table 4.2. It was observed that mappings MAP 93, MAP 271, MAP 276, MAP 699 have similar latencies on the individual paths. Similarly, the mappings MAP 95, MAP 391, MAP 396 and MAP 701 have similar latencies on the individual paths.

Table 4.2: Worst case latency on packet paths for mappings in optimal solution space

Path No	**MAP 93**	**MAP 95**	**MAP 271**	**MAP 276**	**MAP 391**	**MAP 396**	**MAP 699**	**MAP 701**
P1: *ARM to Filter*	5	5	5	5	5	5	5	5
P2: *ARM to Display*	3	3	3	3	3	3	3	3
P3: *FFT to Filter*	3	4	3	3	4	4	3	4
P4: *Filter to FFT*	3	3	3	3	3	3	3	3
P5: *Filter to IFFT*	3	3	3	3	3	3	3	3
P6: *Filter to ARM*	3	3	3	3	3	3	3	3
P7: *IFFT to Filter*	4	3	4	4	3	3	4	3
P8: *Memory to ARM*	3	3	3	3	3	3	3	3

Knowing the latencies on individual packet paths in addition to the average worst-case latency of NoC facilitates finding optimal mapping solutions with respect to real-

time constraints. DSP applications have real-time latency constraint to complete communication within a given time [16], for example, consider a real-time constraint that packets from FFT must reach Filter in less than 4 clock cycles in addition to the general requirement that the average worst case of NoC must be the lowest. Then, from the table it can be found that MAP 93, MAP 271, MAP 276 and MAP 699 satisfy the real-time constraint.

An extension of NMAP was proposed to map application traffic can have real-time constraint [16], for example, packets from ARM must reach Filter as quickly as possible with maximum throughput. Then the suitable architecture may be selected by sorting the worst case latencies. The worst case latencies sorted with ARM to Filter latencies to be the lowest is shown in the Table 4.3. The lowest worst-case latency that is possible between ARM and Filter is 3 clock cycles. Since, latency is inversely proportional to throughput, i.e., if the latency is lower then the throughput of the network is high. NoC mappings with high throughput are MAP 45, MAP 69, AND MAP 402 as they have lowest worst case latency of 3.875 clock cycles. Hence, the possible solutions are MAP45, MAP69, MAP402.

Table 4.3: Worst case latencies on packet paths of NoCs mappings

Mapping	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**Average**
MAP45	3	3	4	3	3	3	5	7	3.875
MAP74	3	3	4	3	7	3	7	3	4.125
MAP69	3	3	5	3	3	3	4	7	3.875
MAP130	3	3	5	5	7	4	8	5	5.000
MAP76	3	3	7	7	3	3	4	3	4.125
MAP124	3	3	8	7	5	4	5	5	5.000
MAP233	3	5	5	5	7	3	8	3	4.875
MAP213	3	5	7	7	3	3	4	3	4.375
MAP402	3	7	4	3	3	3	5	3	3.875
MAP118	4	3	3	3	5	3	6	5	4.000
MAP7	4	3	3	3	5	4	6	5	4.125
MAP50	4	3	3	3	7	3	8	8	4.875
...	..	..	..	..	..	..	..	..	..
MAP93	5	3	3	3	3	3	4	3	3.375
MAP117	5	3	3	3	3	5	4	5	3.875
...	..	..	..	..	..	..	..	..	..
MAP541	7	8	5	5	3	5	3	3	4.875
...	..	..	..	..	..	..	..	..	..
MAP693	8	5	6	5	5	7	5	5	5.750

Inference The case study shows that for any given application, NoC specification and QoS requirement constraints, there is more than one correct solution. There exists

an optimal solution space from which the designer can select a solution based on constraints such as power, area and cost. The possible solution space is shown is Fig 4.4. However, as the application size increases it is not possible to explore the complete mapping space using SPIN model, as the mapping permutation increases exponentially with the size of the application.

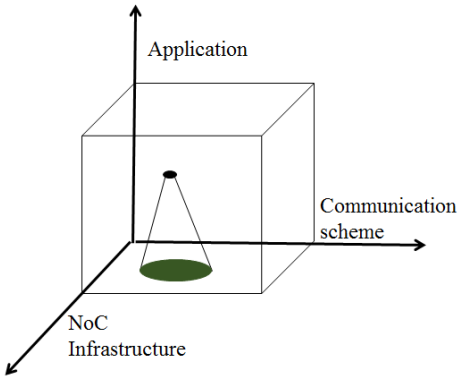

Figure 4.4: Possible solution space

In order to understand the capability of the proposed mapping evaluation technique, complete mapping space exploration was done for mapping an application with 9 functional cores onto a 3x3 NOC. For a 3x3 NoC there are 362880 possible mapping permutations. The complete mapping space was explored and it was found that there were 30 optimal mapping solutions with lowest worst-case latency. 6 and half days (148.62 hours) were taken to explore the complete mapping space. However, exploration of entire mapping space is not possible for large application due to exponential increase in the number of mapping permutations.

But, result confirms two important things (a) it satisfactorily confirms the fidelity of SPIN model for estimating worst-case latency and (b) SPIN model is suitable for reason about NoC mapping problem.

As, there is no mapping heuristics in the literature to more than optimal mapping in the literature, initially the possibility of developing a mapping heuristic to find more than one solution was carried out. In NMAP algorithm, an exact placement is generated and then iterated to obtain exact optimal mapping solution. In order to find multiple optimal solutions, the solutions are generated iteratively by systematically placing the cores according to bandwidth constraint, retaining all the placements that satisfied the bandwidth constraint. The proposed mapping algorithm proposed does not terminate for large applications and the algorithm does not find the complete set of optimal mappings. However, it was a partial success so the algorithm and the result are recorded for reference in the Appendix 8.

Genetic algorithm based mapping mapping algorithms are capable of finding multiple optimal solution [7, 8]. The possibility of improving GA based mapping using formal model is explained in the next section.

5 Integrating formal model into GA-based mapping algorithm

Genetic Algorithms (GAs) belong to the class of evolutionary algorithms, with heuristics inspired by the process of natural evolution. It is an iterative process for finding near-optimal solutions for very large and/or multidimensional solution spaces. The heuristic starts with a population of random solutions called seeds and uses a fitness function to keep or discard a part of the population. Then mutation and crossover operators are used to generate seeds for the next generation from the first population. We improve the GA based mapping algorithm is improved by using the formal NoC model in the fitness function to estimate worst-case latency and select part of the population with lowest worst-case latencies.

A simple GA based algorithm for finding an optimal mapping is shown below:

A Generate a random population of solutions. The individual member of the population is a chromosome, which represents a plan of mapping. Every chromosome contains a series of genes. Every gene represents an application node and its position represents the vertex in a mesh topology.

B The fitness function is used to estimate latency for every member in the population. Using fitness function for estimating worst case latency, the part of the population with lowest worst case latency is selected from the old population.

C A new population is generated from the selected population using chromosome crossover and mutation operators. Both crossover and mutation are used to converge to the results. Mutation introduces diversity in the population and crossover operator combines parents to create offspring. Single point crossover is generated by combining at a single cut.

D Go to Step B and repeat for fixed number of loop iterations called generations.

E The best individual of the last population is the optimal mapping obtained using genetic algorithm.

Initially in step [A], the GA algorithm generates seeds or initial population with a plan for mapping randomly. The number of seeds can be initialized, this is called offspring size. In the next step [B], a fitness function is used to check if the seeds of the first population are suitable for breeding in order to create the next generation. The fitness function uses the formal NoC model to estimate worst case latency for all the seeds, and the seeds with lowest worst-case latency are selected for breeding. Breeding is done using mutation operator or crossover operator. The number of rounds of mutation/scheduled flow can be set in the code, this is called the generation size.

The novelty of our approach is the integration of NoC formal model in the fitness function of the genetic algorithm. The formal model is called as an inline function to estimate worst case latency. The fitness function selects the offspring with the lowest worst case latency from the seeds. It is necessary to set the right population size and generation size to find the optimal solutions. By executing the mapping algorithms with different population sizes and generation sizes for a simple problem, it was found that population size = 100 and generation size = 10 are appropriate to find many optimal / near optimal solutions with formal model in fitness function.

6 Experiment

The mapping solution generated by GA based mapping algorithm integrated with formal model is compared with the solution of GA based mapping algorithms with analytical model, to find if integrating formal model has improved the mapping algorithm.

Sample test application: Mapping algorithms are compared using 7 sample applications [APP1...APP7], each containing 6, 9, 16, 25, 100, 400, 900 cores with 5, 15, 25, 38, 150, 600, 1349 communication traces respectively. These applications are mapped on NoCs in 3x2, 3x3, 4x4, 5x5, 10x10, 20x20 and 30x30 mesh topologies respectively. Routers of NoC are typical HERMES routers [17] with XY routing algorithm, priority based round robin arbitration logic and unit buffers at input port.

The formal NoC model is used to estimate worst-case latency for the mapping solutions generated by different mapping algorithms, and they are compared to check if integrating formal model produced better mapping solution than the mapping algorithm without formal model.

GA with analytical model: Latency refers to the length of time elapsed for packets from source node to reach the destination node. Average latency in NoC is depends on hop count and delay in contention resolution in the network [18]. Latency associated with hop count depends on interconnect topology and routing algorithm. Contention latency is impacted by traffic loads and buffers in the router, it is calculated using either probabilistic models or queuing theory [19]. The basic method of delay estimation by hop-count does not consider the contention latency. The development of good analytical model encompassing several design factors such as buffer size, traffic load, routing algorithm is an open problem.

Generally, mapping algorithms use hop count as cost hence, in this paper the analytical model for GA based mapped algorithm is developed for basic delay estimation, i.e., end-to-end latency of packets in NoC is estimated in terms of hop count which depends on the routing algorithm in the routers. In this case, hop count is calculated deterministically for any given mapping configurations as the routers are implemented with XY routing algorithm.

6.1 Compare GA based mapping algorithms with respect to worst-case latency

The population size is set to 100 and generation size is set to 10, and mapping solutions are generated by GA based mapping algorithms using formal model in the fitness function and analytical model in the fitness function. The mapping solutions generated by the algorithms are compared with respect to worst case latencies.

In the genetic algorithms, the offspring is generated using two types of genetic operators (i) random mutation operator and (ii) single point crossover operator. The worst-case latency estimated for the mapping solutions generated by using random mutation and single point crossover in the GA with formal model are shown in the columns WCL (R_FM), WCL (S_FM) and for GA with analytical model in the columns WCL (R_AM), WCL (S_AM) of Table 6.1.

From the table, it can be seen that all the GA-based algorithms perform equally well for smaller applications [APP1...APP4]. However, the GA algorithm integrated with formal model is better than GA with analytical model for larger applications [APP5...APP7], as the worst-case latency estimated for the mapping solution generated by GA algorithm with formal model is lower that GA algorithm with analytical model.

Table 6.1: Comparative results for Genetic algorithm: Worst case latencies (cycles)

Application	WCL (R_FM)	WCL (S_FM)	WCL (R_AM)	WCL (S_AM)
APP1	3.375	3.375	3.375	3.375
APP2	3.84	3.84	4.384	4.3
APP3	5.32	5.44	5.56	5.56
APP4	6.6	7.4	7.89	8.31
APP5	20.2	20.7	25.14	29.55
APP6	86.45	80.4	121.515	112.47
APP7	203.8	222.1	395.47	271.38

Inference: The worst case latency estimated for mapping solution generated by GA with formal model using random mutation is on an average 20 $\%$ lower than GA with analytical model using random mutation.

The worst case latency estimated for mapping solution generated by GA with formal model using single point crossover operator is on an average 13 $\%$ lower than GA with analytical model using single point crossover operator.

6.2 Compare results with a simple mapping heuristic

A simple mapping heuristic was developed by simplifying the mapping algorithm optimized over several parameters [15] to bandwidth constraint only as shown in Algorithm 2. It uses the application *Communication Graph (CG*) and the *Architecture Graph (AG)* of the NoC as inputs to generate an optimal mapping using bandwidth constraints.

A **Communication Graph** $CG = (P, E)$ is an undirected graph, where a vertex/node $p_k \in P$. Where P represents a node in the application (e.g., P may be an IP core such as a processor or a memory unit), and an edge $e_i \in E$, represents the communication trace between vertices p_k and p_j . Each edge e_i is associated with a communication bandwidth request between vertices p_i and p_j given in bits per second (bps), represented by $b(e_i)$, which is representative of the rate of packet injection by the nodes.

An **Architecture Graph** $AG = (T, L)$ is an undirected graph, where each vertex $t_i \in T$, where T represents a tile and each edge $l_i \in L$ represents the communication link between tiles t_k and t_j.

The mapping algorithm $M : P \rightarrow T$ maps each vertex in the Communication Graph onto an available tile in the Architecture Graph. $M(p_i)$ represents the mapped tile in AG, where $p_i \in P$ and $M(pi) \in T$.

The mapping problem is formulated as: given a CG(P, E) representing the communication traces of an application and an AG(T, L) representing the topology of target NoC architecture, where $|P| \geq |T|$, find a mapping $M : P \rightarrow T$ which maps all the vertices in CG onto available routers in AG, such that the nodes have minimal routing for XY routing algorithm.

Initially, the application nodes p_k in the communication trace graph $CG(P, E)$ with maximum outgoing links is found and appended to the to_be_routed list and routers

t_i in the topology graph $AG(T, L)$ with 4 outgoing links are found and appended to $available\_routers$ list. Mapping is done by placing nodes in to_be_routed list to routers in $available\_routers$ list. The mapped routers are appended to $routed\_node$ list. If the length of to_be_routed list is greater than $available\_routers$, it will be placed in the next round. The nodes attached to nodes in 'routed-node' list are found and the one with the highest bandwidth is placed to router on 'x' co-ordinates of the placement. This is done to satisfy the bandwidth constraint as well as minimize communication latency. Similarly all the nodes attached to the nodes in $routed\_node$ list are placed. The process is repeated all for application nodes with outgoing links greater than or equal to 3, then 2 and 1. Finally any unplaced node is mapped.

Algorithm 2 Map(CG(P,E),AG(T, L))

Input: CG(P,E), AG(T,L)
Output: Mapping $M : P \rightarrow T$
$NUM\_LINKS = 4$ /*Max. fanout in mesh topology is 4*/
$to\_be\_routed, available\_routers, routed\_node = \emptyset$
while len(routednode) $\leq$ len(CG) **do**
  $\forall\ p_k \in P$ of $CG = (P, E)$ with maximum outgoing links $\Rightarrow to\_be\_routed \cup \{p_k\}$
  $\forall\ t_k \in T$ of $AG = (T, L)$ with fanout $\geq NUM\_LINKS \Rightarrow available\_routers \cup \{t_k\}$
  $\forall\ p_k \in to\_be\_routed$ and $t_k \in available\_routers$ map p_k to $t_k \in$ AG and $routed\_node \cup \{p_k\}$, $available\_routers \setminus \{t_k\}$, $to\_be\_routed \setminus \{p_k\}$
  for all $p_k \in routed\_node$ **do**
    $\forall\ p_k \rightarrow e_i p_n$ for $p_k \in$ CG $\Rightarrow to\_be\_routed \cup \{p_n\}$
    for all $p_n \in to\_be\_routed$ **do**
      pos = position($p_n \in$ AG) with highest $b(e_n)$
      $pos \rightarrow l_i t_m$ for $t_m \in AG$ is not mapped $\Rightarrow available\_routers \cup \{t_m\}$
      $\forall\ p_k \in to\_be\_routed$ and $t_k \in available\_routers$ map p_k to $t_k \in$ AG and $routed\_node \cup \{p_k\}$, $available\_routers \setminus \{t_k\}$, $to\_be\_routed \setminus \{p_k\}$
    end for
    $to\_be\_routed, available\_routers = \emptyset$
  end for
  $NUM\_LINKS = NUM\_LINKS - 1$
end while

First, the mapping heuristic is used to map applications on NoCs optimally. The time taken to obtain mapping solution using simple mapping heuristics is 16ms for 3x2, 3x3, 4x4, 5x5 NoC, 0.5 sec for 10x10 NoC, 1.8 min for 20x20 NoC and 43 minutes for 30x30 NoC. The worst-case latency estimated using the formal model for the mapping solutions generated mapping heuristic is shown in Table 6.2. The worst-case latency for solutions generated mapping heuristic is shown in the column WCL(Heuristic), it is measured in number of clock cycles.

The processing time, memory and size of state vector used to estimate worst case latency using formal model for large NoCs is also shown in Table 6.2. State vector denotes the number of bytes of memory (per state) required for complete description of a global system. It is seen that the memory usage increases with the state vector size, which in turn increases with NoC size and number of communication traces in the application. As the size of physical memory available for computing has become vast with the new VLSI technologies, any large NoC can be modeled and verified using

Table 6.2: Worst case latency (WCL) for large applications using mapping heuristic

Expt No.	NoC Size	Total traces	WCL (Heuristic)	Time (sec)	State Vector	Memory (MB)
APP1	3x2	8	3.375	0.016	940	6.3
APP2	3x3	13	4.0	2.73	1408	43.9
APP3	4x4	25	7.91	4.52	2376	45.39
APP4	5x5	38	27.1	6.74	3620	46.34
APP5	10x10	150	35.54	19.1	14076	241.28
APP6	20x20	600	210.72	189	55972	4942.68
APP7	30x30	1349	295.65	896	125716	24926.9

formal model depending on available memory.

Compare algorithm execution time: The time taken to generate the mapping solutions by the GA algorithm for large applications are shown in Fig 6.1 for comparison.

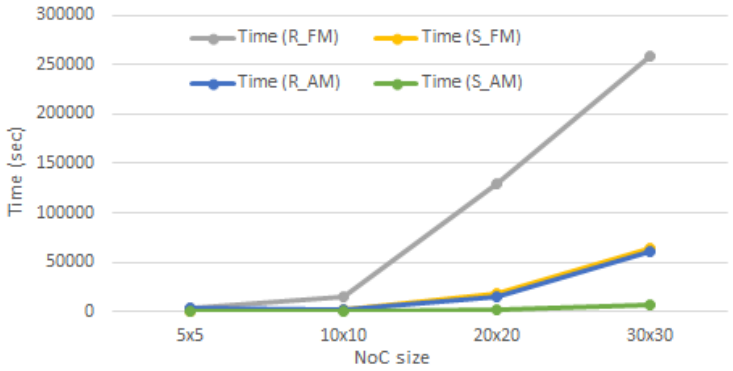

Figure 6.1: Comparison of time taken for different mapping algorithms

GA with formal model using random mutation operator generates better mapping solution than GA with formal model using single point crossover as mutation operator introduces diversity in the population, as a result the solution converges closer to optimal solution quickly. However, the mapping algorithms with single-point crossover operator are faster than random mutation operator.

Hence, it is can be concluded that GA algorithm with formal model is more effective than GA algorithm with analytical models in this case. However, better analytical model might enable GA algorithm with analytical surpass GA algorithm with formal model.

7 Conclusion and Future Work

This technical report shows that formal methods based models are more than just tools for functional verification. A formal NoC model was used to improve mapping algorithms that are constrained to latency. As part of future work, formal models may be stretched to estimate other QoS parameters, thereby improving NoC design decisions.

8 Appendix : Optimal mapping space generation heuristic

The mapping algorithm uses communication graph of the application and topology graph of NoC architecture as input to generate optimal mapping based on application bandwidth specification and minimizing latency. The communication pattern between the nodes in an application is described by a *Communication Graph (CG*). The topology of the NoC architecture is described by a *Architecture Graph (AG)*.

A **Communication Graph** $CG = (P, E)$ is an undirected graph, where a vertex/node $p_k \in P$, where P represents a node in the application (IP core such as a processor or a memory unit, etc.), and an edge $e_i \in E$, represents the communication trace between vertices p_k and p_j . For each edge e_i, is associated with a communication bandwidth request between vertices p_i and p_j given in bits per second (bps), represented by $b(e_i)$. $b(e_i)$ is representative of the rate of packet injection by the nodes.

An **Architecture Graph** $AG = (T, L)$ is an undirected graph, where each vertex $t_i \in T$, where T represents a tile and each edge $l_i \in L$ represents the communication link between tiles t_k and t_j.

A mapping algorithm $M : P \rightarrow T$ was developed to map each vertex in the Communication Graph onto an available tile in the Architecture Graph. $M(p_i)$ represents the mapped tile in AG, where $p_i \in P$ and $M(pi) \in T$.

The mapping problem is formulated as: Given a CG(P, E) representing the communication traces of an application and an AG(T, L) representing the topology of target NoC architecture, where $|P| \geq |T|$, find a mapping $M : P \rightarrow T$ which maps all the vertices in CG onto available routers in AG, such that the nodes have minimal routing for XY routing algorithm.

8.1 Mapping algorithm to find all the optimal solution

The mapping algorithm to find all the optimal solution is shown in Fig. 3. The inputs to mapping algorithms is communication graph CG(P, E) and architecture graph AG(T, L). The output of mapping algorithm is list of optimal mapping solutions.

The mapping algorithm is implemented in 3 steps: firstly, mapping nodes with high bandwidth requirements, secondly, mapping nodes connected to previously mapped nodes and finally mapping rest of the nodes. Every time mapping is generated by generating permutations for the available routers. The available routers are selected based on the routers that have maximum outgoing links.

Step 1: From the communication graph (CG) the functional cores with maximum outgoing edges is found and appended to the list 'keynodeap' . The nodes in 'keynodeap' are mapped first on the nodes in architecture graph (AG) with maximum outgoing links. The nodes with maximum outgoing links in AG are found and appended to the list 'keyrouter'. If there are more functional core with maximum outgoing edges

Figure 3 Map_all_opt_sol(CG(P,E),AG(T, L))

```
Input: CG(P,E) graph of communication trace of application, AG(T,L) graph of
NoC topology
Output: Route packet to an appropriate output port
find initial key nodes of the application and generate mapping permutation
initialize 'mapping_graph' to 'AG(T, L)'
for all p ∈ 'CG(P,E)' do
   Find nodes with maximum number of edges and append to 'keynodeap'
end for
Find routers with maximum number of edges and append to 'keyrouter'
for all q ∈ 'AG(T, L)' do
   Find routers with maximum number of edges and append to 'keyrouter'
   if len(keynodeap) > len(keyrouter) then
      Append additional routers on x axis of 'keyrouters' as XY routing is used
   end if
end for
Map nodes in CG to routers in AG
for all nds ∈ 'keynodeap' and rts ∈ 'keyrouter' ∈ mps in 'mapping_graph' do
   Generate mapping permutations and append to 'mapping_graph'
   Append to nds 'routednode'
end for
for all x ∈ mapping_graph do
   Remove items from 'mapping_graph' that contain previously mapped router as
   available
   if x contains rts ∈ 'keyrouter' then
      Remove items from 'mapping_graph'
   end if
end for
find next level of keynodes and generate mapping permutation
while len(keynodeap) ≠ 0 do
   re-initialize 'keynodeap' to null
   find nodes with high bandwidth requirement attached to nodes in 'routednode'
   and append to nds 'keynodeap'
   find available routers to map node in 'keynodeap' to 'mapping_graph'
   for all p ∈ 'routednode' do
      Find nodes with with high bandwidth requirement and append to 'keynodeap'
   end for
   Mapnode(mapping_graph,routednode,keynodeap,keyrouter)
end while
find the remaining unmapped nodes and map it
re-initialize 'keynodeap' to null
find nodes of CG not in 'routednode' and append to 'keynodeap'
find available routers to map node in 'keynodeap' to 'mapping_graph'
Mapnode(mapping_graph,routednode,keynodeap,keyrouter)
```

than nodes available in AG in i.e., length if 'keyrouter' is less that 'keynodeap' additional keynodes are added to 'keyrouter' list. XY routing algorithm is used in the NoC where the X axis are touted first. Hence the nodes adjacent to nodes with maximum outgoing links on x-axis are found and appended to 'keyrouter' list. Since, there may be more than one node in AG with maximum outgoing links, permutation of mapping is made and appended to mapping_graph list for every nodes with maximum outgoing links in architecture graph (AG). The nodes in 'keynodeap' that are having been routed are finally appended to 'routednode' list. The combination list in mapping_graph is cleaned up by removing redundant items and removing items containing router address that have been already mapped. The routers that are already mapped are in 'keyrouter' list.

Step 2: After the initial mapping of nodes with maximum outgoing edges. The nodes connected to initially mapped nodes are mapped next. The nodes to be mapped next are identified based on the bandwidth. The nodes with higher bandwidth is mapped first. The nodes to be mapped are appended to 'keynodeap' list The mappings on AG in the mapping_graph list is performed using mapping function in Fig 4. The mapping is done till all the nodes attached to keynodes are recursively mapped. The inputs to mapping function are the mapping_graph, the available routers for each item in mapping_graph and appended to 'keyrouter' based on number of outgoing links. More routers are added if the number of nodes to be router is more. The combination of mappings are generated are appended to mapping_graph and the routednode list is updated.

Figure 4 Mapnode(mapping_graph,routednode,keynodeap,keyrouter)

```
    for all q ∈ 'mapping_graph' do
      For each item in 'mapping_graph' find the available router
      For the available router find routers with maximum outgoing links and append
      to 'keyrouters'
 4:   if len(keynodeap) > len(keyrouter) then
        Append additional routers on x axis of 'keyrouters' as XY routing is used
      end if
      generate mapping permutation
 8:   for all nds ∈ 'keynodeap' and rts ∈ 'keyrouter' do
        if nds ∉ 'mapping_graph' then
          Generate mapping permutations and append to 'mapping_graph'
          Append to nds 'routednode'
12:     end if
      end for
    end for
    for all x ∈ mapping_graph do
16:   Remove items from 'mapping_graph' that contain previously mapped router as
      available
      if x contains rts ∈ 'keyrouter' then
        Remove items from 'mapping_graph'
      end if
20: end for
```

Step 3: The last step of mapping is to place the remaining nodes at the available routers. The mappings on AG in the mapping_graph list is performed using mapping function in Fig 4 as explained above.

The output of this algorithm produces a list of optimal mapping. The worst-case latencies of all the mappings is estimated and mappings with lowest worst-case latency is obtained.

Results Using the above algorithm all the mapping solutions for applications with 6 nodes and 9 nodes were found. However, the mapping algorithm does not terminate for 16 node application. This may be due to the number of permutations. For 6 node application mapped to 2x3 NoC 48 solutions were found. The worst case latencies of the mappings were found using SPIN. The worst case latencies found were 3.375 cycles, 4.0 cycles, 4.125 cycles and 5.25 cycles. All the 8 optimal mappings with worst-case latency of 3.375 clock cycles were found.

For 9 node application mapped to 3x3 NoC 1152 solutions were found. The worst case latencies were estimated using SPIN model The worst-case latencies range from 3.9 clock cycles to 7.4 clock cycles. 6 optimal mappings with worst-case latency of 3.9 clock cycles were found. However, 30 optimal mapping were identified during complete mapping space exploration.

Bibliography

[1] P. K. Sahu and S. Chattopadhyay. A survey on application mapping strategies for network-on-chip design. *Journal of Systems Architecture*, 2013.

[2] Chen-Ling Chou and R Marculescu. Contention-aware application mapping for network-on-chip communication architectures. In *ICCD'08*.

[3] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular noc architectures. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2005.

[4] C. Marcon, A. Borin, A. Susin, L. Carro, and F. Wagner. Time and energy efficient mapping of embedded applications onto nocs. ASP-DAC '05.

[5] S. Mohalik, A. C. Rajeev, M. G. Dixit, S. Ramesh, P. V. Suman, P. K. Pandya, and S. Jiang. Model checking based analysis of end-to-end latency in embedded, real-time systems with clock drifts. DAC '08.

[6] A. Giuseppe, C. Vincenzo, and P. Maurizio. A multi-objective genetic approach to mapping problem on network-on-chip. *j-jucs, 2006*.

[7] S. Tabandeh, C. Clark, and W. Melek. A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. In *IEEE Evolutionary Computation, CEC 2006.*

[8] P. A. Turner. Genetic algorithms and multiple distinct solutions. Technical report, University of Edinburgh, 1993.

[9] P. Mesidis and L. S. Indrusiak. Genetic mapping of hard real-time applications onto noc-based mpsocs - a first approach. In *ReCoSoC'11*.

[10] Tang Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *DSD'03*.

[11] D. Bui, A. Pinto, and E. A. Lee. On-time network on-chip: Analysis and architecture. Technical Report UCB/EECS-2009-59, 2009.

[12] G. Holzmann. *Spin model checker, the: primer and reference manual*. Addison-Wesley Professional, 2003.

[13] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto noc architectures. In *DATE 2004*, volume 2, pages 896 – 901 Vol.2.

[14] xxxx. xxxx. *XXXX*.

[15] X. Wang, M. Yang, Y. Jiang, and P. Liu. A power-aware mapping approach to map ip cores onto nocs under bandwidth and latency constraints. *ACM Trans. Archit. Code Optim.*, 2010.

[16] S. Murali, L. Benini, and G. de Micheli. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In *ASP-DAC 2005*, volume 1, pages 27 – 32 Vol. 1.

[17] F. Moraes A. Mello, N. Calazans, L. Moller, and L. Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *VLSI Journal*, 38(1), 2004.

[18] N. Nikitin, J. de San Pedro, J. Carmona, and J. Cortadella. Analytical performance modeling of hierarchical interconnect fabrics. NOCS '12.

[19] A. E. Kiasari, Z. Lu, and A. Jantsch. An analytical latency model for networks-on-chip. *IEEE Trans. VLSI Syst.*, 2013.