Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A framework for low power synthesis of interconnection networks-on-chip with multiple voltage islands

Nishit Kapadia*, Sudeep Pasricha

Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523-1373, USA

ARTICLE INFO

Available online 1 December 2011

Keywords:
Networks-on-chip
Voltage islands
Synthesis algorithms
Chip multiprocessor
Core-to-tile mapping

ABSTRACT

The problem of VI-aware Network-on-Chip (NoC) design is extremely challenging, especially with the increasing core counts in today's power-hungry Chip Multiprocessors (CMPs). In this paper, we propose a novel framework for automating the synthesis of regular NoCs with VIs, to satisfy application performance constraints while minimizing chip power dissipation. Our proposed framework uses a set of novel algorithms and heuristics to generate solutions that reduce network traffic by up to 62%, communication power by up to 32%, and total chip power dissipation by up to 13%, compared to the best known prior work that also solves the same problem.

1. Introduction

Emerging chip multiprocessors (CMPs) today are severely constrained because of their high power dissipation in compute cores due to high levels of core integration and nanoscale CMOS process technology characteristics. High power dissipation on a chip not only reduces overall performance, but also negatively impacts reliability and cooling costs. As the number of cores integrated in CMPs continues to increase into the hundreds [1,2], the complexity of network-on-chip (NoC) fabrics [3] required to interconnect communicating cores on a chip has also increased. NoCs have been shown to dissipate significant power (e.g., ~30% of chip power in Intel's 80-core teraflop [1] and ~40% of chip power in MIT RAW [2] chips). As a result, reducing both computation and communication power has become a high priority for chip designers.

Among the several circuit and (micro)-architectural techniques proposed by designers in recent years to reduce power dissipation, dynamic voltage and frequency scaling (DVFS) is widely used to reduce dynamic power [4], while clock/power gating techniques [5] are commonly used to reduce leakage power in cores. However, implementing these techniques at a per-core granularity requires separate V_{DD} and ground lines, as well as a prohibitively large number of VLCs (voltage level converters) and MCFIFO (multiple clock first-in-first-out) queue based frequency level converters [6], [7], especially as the number of cores on a die increase. The use of voltage islands (VIs) limits the overhead of implementing these power management schemes by combining cores into islands that use the same V_{DD} and ground lines, and thus also minimizing the number of VLCs and MCFIFOs required [8]. Different islands can then operate at different voltage levels and perform runtime optimizations independent of each other to more aggressively reduce chip power. In the presence of VIs, not only can cores run on optimal voltages with minimal overhead to reduce chip power dissipation, but the on-chip interconnection network can also exploit VIs to reduce communication power. However, VIs complicate the problem of NoC synthesis, requiring designers to revisit the problems of VI partitioning, core to die mapping, and routing path allocation so that both computation and communication power can be minimized.

In this paper we address the problem of synthesizing NoCs for regular CMPs with VIs to reduce overall power dissipation. We focus on CMPs with homogenous and regular sized cores because of the prevalence of regular topologies in almost all recently released commercial CMPs [1], [2]. Our proposed framework for VI-aware Synthesis of IntercOnnection Networks on chip (VISION) combines novel algorithms and heuristics to perform VI partitioning, core to die mapping, and routing path allocation to minimize power dissipation while satisfying application bandwidth requirements. We propose three variants of our framework based on different core-to-tile mapping heuristics: *(i)* incremental swapping (*VISION*), *(ii)* Probabilistic Incremental Swapping (*VISION-P*), and *(iii)* Incremental Swapping using Branch & Bound (*VISION-B*). Experimental studies presented in Section 5 indicate that the proposed VISION framework outperforms the best known prior work [9] that focuses on the same problem of VI-aware regular NoC synthesis. In particular, our framework generates solutions that have lower network traffic by up to 62%, lower

* Corresponding author. Tel.: +1 970 492 9509.
E-mail addresses: nkapadia@colostate.edu (N. Kapadia), sudeep@colostate.edu (S. Pasricha).

0167-9260/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2011.11.010

communication power by up to 32%, and lower total power dissipation by up to 13% compared to solutions generated by the approach in [9].

2. Related work

Many researchers [10–14] have proposed custom topologies for NoC architectures that improve overall performance at the cost of sacrificing the regularity of mesh-based topological structures. Although these custom architectures are expected to achieve better latency and area utilization, their design process is significantly more complex and faces several challenges, such as greater crosstalk and uncertainty in link delays due to irregular interconnect structures. Thus, a conservative enough custom design may actually offset the advantages of better performance [15]; especially for medium to large sized NoC architectures (in terms of total number of cores).

Lackey et al. [8] give an overview of the preliminary considerations for power and performance for NoC designs with VIs. The problem of NoC synthesis on regular structures with multiple supply VIs has since been addressed in several works [9,16–21]. Ghosh et al. [17] solve a linear programming formulation of the same problem, but without considering the effects of multiple frequencies and the required MCFIFO-based converters. Leung et al. [16] propose a genetic algorithm to combine the different steps in a VI-aware NoC synthesis flow. Ogras et al. [18] perform the VI partitioning and static voltage–frequency assignment on a pre-mapped NoC to optimize communication and energy consumption while meeting task deadline constraints. By performing voltage-partitioning before core mapping, [9] demonstrated improvements over prior work (e.g., [18]) to achieve less power overhead by reducing total number of MCFIFOs and VLCs needed for inter-island communication.

The problem of mapping cores on to regular NoCs has been handled differently by the works above. Heuristics implementing incremental mapping on NoCs with *VIs* that have been proposed to date [9,20] map the cores in order of their communication bandwidths. As the order of mapping is fixed, every new mapping decision is based on optimizing some communication metric with just the previously placed cores, thereby restricting the search space to a possible local minimum and failing to capture a holistic view for optimizing the communication over the entire network. In this paper, in addition to proposing novel techniques for voltage partitioning and routing path allocation, we present three new mapping techniques that use the foundational concept of incremental swaps with a distributed decision making process, as opposed to a central one used in prior work.

3. Problem formulation

We are given the following inputs to our problem:

(i) A core graph *G (V, E)*; with the set of *N* vertices $\{V_1, V_2, V_3,\ldots,V_N\}$ representing the homogeneous cores on which tasks have already been mapped and the set of *M* edges $\{e_1,e_2,e_3,\ldots,e_M\}$ that represent communication dependencies between cores;
(ii) A regular mesh-based NoC with *T* tiles such that $T \geq N$, and $T=(d^2)$, where *d* is the dimension of the mesh, and each tile consists of a compute core, a NoC router, and a network interface (*NI*) between them;
(iii) A set of minimum operating voltage levels required for meeting task performance requirements on each of the *N* cores $\{min\_v(V_1), min\_v(V_2), \ldots, min\_v(V_N)\}$;
(iv) A set of *n* candidate voltage levels $\{v_1, v_2,\ldots,v_n\}$ and the corresponding candidate frequency values $\{f_1 f_2,\ldots f_n\}$ that the cores can take; and
(v) The maximum allowable number of voltage islands on the die, Ω; where $\Omega \leq n$.

Objective: Given the above inputs, the goal of our synthesis framework is to obtain a core to die mapping and synthesize a regular 2D mesh NoC architecture for a specific application, such that all application performance requirements are satisfied, while minimizing the total power dissipation in the compute cores and the communication resources (routers, links, VLCs, MCFIFOs).

Definition 1. Voltage Island Integrity of any island is said to be respected when every core within it has at least one neighbor (out of the maximum of four neighbors possible in a mesh) of the same voltage level as itself. Mathematically:

$$\forall v(t_{xy}) = v_i : \exists\{v(t_{x-1,y}) = v_i | v(t_{x,y-1}) = v_i | v(t_{x+1,y}) = v_i | v(t_{x,y+1}) = v_i\}$$

where t_{xy} represents the tile at the respective co-ordinates of the mesh, with integral values in the range *1–d*.

Definition 2. Pre-Routing Traffic is an early estimation of the total traffic assuming that all routing paths are minimal. It is equal to the sum of products of Manhattan distances and bandwidths of all individual communication flows. Mathematically, this can be expressed as: $\sum_j MD_j{}^*BW_j$; where *j* is a uni-directional communication flow between any two cores on the die.

Definition 3. Link Tension is defined as the product of the communication bandwidth and post-mapping Manhattan distance for any edge on the core graph *G (V, E)*. Thus, for two cores adjacent to each other (i.e., Manhattan distance = 1), the tension on the link connecting them is equal to the bandwidth of communication between them.

Definition 4. Total Traffic is the overall bandwidth a routed-network is required to suport. It is the sum total of bandwidths of all communication flows over their respective actual (minimal or non-minimal) path-lengths, expressed as: $\sum_j Path\text{-}length_j{}^*BW_j$; where *j* is a uni-directional data communication flow between any two cores.

Not unlike previous works [9,18,19], our proposed NoC synthesis framework is subdivided into three major steps: voltage partitioning, core-to-tile mapping, and routing path allocation. Where our framework differs from previous works is in the algorithms used and more rigorous implementation assumptions that we consider. As the computation power (dissipated in compute cores) still dominates the communication power (dissipated in routers, links, MCFIFOs, VLCs) for CMPs with up to several tens of cores, optimizing the former should take precedence over the latter for minimal total power. Therefore, in this work (unlike in [18]), we perform the voltage partitioning step before core-to-tile mapping. The problems addressed in the three major steps of our framework are summarized below:

A. Communication-power aware voltage partitioning
The objective in this first step is to assign voltages to cores in the core graph to minimize the computation power and the communication power at the same time, such that the constraints on the minimum operating voltage levels of all *N* cores $\{min\_v(V_1), min\_v(V_2), \ldots, min\_v(V_N)\}$ and the maximum number of VIs allowed (Ω), are satisfied.
B. Core to tile mapping
The objective in this second step is to perform a one to one mapping of voltage assigned cores onto NoC tiles such that the integrity of each VI is respected (Definition 1), hikes in

compute power of cores are avoided, and total estimated traffic (pre-routing traffic in Definition 2) is minimized, to optimize overall communication power dissipation.

C. Routing path allocation:
The objective in this third and final step is to allocate routing paths for all application-specific communication flows in the NoC, such that the communication power overhead associated with MCFIFOs and VLCs needed for inter-island communication is minimized.

4. NoC synthesis flow

In this section, we present details of our framework for the synthesis of NoCs with VIs. Fig. 1 shows the high level flow of our synthesis framework that improves upon the state of the art VI-aware NoC synthesis frameworks, such as that proposed in [9]. The *partitioning* stage supports trade-offs between computation power and the estimated communication power, while assigning voltages to cores and partitioning them into islands. The *mapping* stage attempts to meet physical proximity constraints for cores in islands and performs swapping heuristics to reduce pre-routing traffic. Finally, the routing path allocation stage integrates link insertion and routing path selection for communication flows, to minimize traffic and reduce communication latency.

In the following subsections (Sections 4.1–4.3), we present a detailed explanation of the algorithms used in the three major stages of our synthesis framework.

4.1. Communication power-aware voltage partitioning

Voltage partitioning attempts to assign core voltages, in order to meet constraints imposed by application performance and the Ω levels of allowable voltages. The voltage partitioning approach in [9] performs an exhaustive search on all n voltage level candidates using a maximum of Ω levels of allowable voltages (for Ω VIs). Thus, ${}^{n}C_{\Omega}$ combinations of different voltages are generated. Then, each core is assigned the least voltage level out of the available voltages for each of the ${}^{n}C_{\Omega}$ combinations, to ensure that application performance constraints are satisfied. Out of the ${}^{n}C_{\Omega}$ combinations of voltage assignments, the least expensive assignment in terms of power is then finally chosen. The shortcoming of this approach is that the voltage partitioning only optimizes the computation power, ignoring the effects of the communication power on the resultant total power. We propose a new *repartitioning* stage that can be appended to the previous approach to overcome the above mentioned drawback.

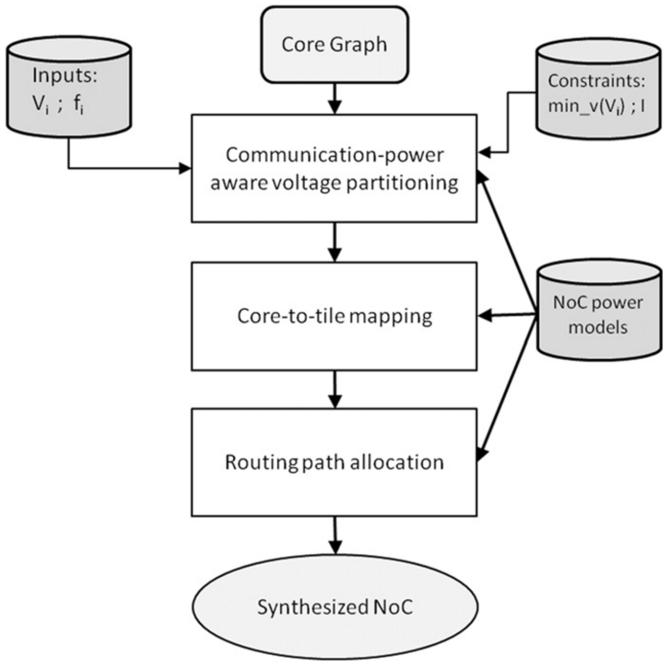

Fig. 1. NoC synthesis design flow.

After the voltage partitioning of [9] is completed based on optimal computation power, in our repartitioning stage we allow certain cores to be moved to a neighboring voltage island with the next higher allowable voltage level, in order to reduce the communication power associated with the core. Such a move is meant to reduce communication traffic overhead by merging the core with an island it communicates heavily with; and is referred to as an 'allowable move' as long as the increase in resultant power is within a certain threshold ψ. Note that the allowable move is restricted to only the voltage island with an immediately higher voltage level because in low to medium sized NoCs (where the maximum distance between any two cores is restricted by the dimension of the mesh), the computation power dominates the communication power. Therefore, the reduction in communication power by performing a move that would increase the voltage of any core to a much higher level would seldom justify the penalty incurred on the computation power.

With repartitioning, cores that heavily communicate with each other end up residing in the same island and thus are likelier to be mapped in each other's vicinity. The probable reduction in the communication power as well as the total traffic is primarily due to lesser inter-island communication that leads to: (i) shorter average communication path lengths, and (ii) a reduction in the number of MCFIFOs and VLCs, with lesser inter-island communication. The cost function for any candidate core move from island i to island j can be expressed as:

$$C(i,j)=(P_{Rj}-P_{Ri})+(O_j-O_i)+\mathscr{H}$$

where P_{Rj} and P_{Ri} are the router powers if the core were in island j or island i respectively, O_j and O_i are the power overhead of MCFIFOs and VLCs associated with the respective islands, and $\mathscr{H}$ is the compute power hike for the core because of the move. The mapping threshold ψ can be expressed as:

$$\psi=(comm_j/(comm_i))\times(1/(v_j-v_i))^{*}\omega$$

where $comm_j$ and $comm_i$ represent the total communication bandwidths of the core when it is mapped to island j and island i, respectively, v_j and v_i are the corresponding island voltages, and ω is designer specified parameter that regulates the aggressiveness of the moves. We use a conservative value of $\omega=0.0001$ to balance computation and communication power, and avoid dramatically increasing total power during a move. Ultimately, the mapping threshold ψ quantifies the projected reduction in post-mapping traffic (and thus the communication power) resulting from the move under consideration. Algorithm 1 below summarizes the repartitioning approach.

Algorithm 1. Repartitioning.

```
input: voltage-partitioned core graph G(V, E)
1: Compute the cost C and threshold $\psi$ for all allowable moves
2: Make the move with the lowest cost (below the
   corresponding move threshold $\psi$) and lock this core in the
   new island.
3: Re-compute a new set of allowable moves for all the
   unlocked cores and the associated costs and thresholds.
4: Repeat steps (2) and (3) until all costs of the remaining
   unlocked cores are above their respective thresholds
output : voltage-repartitioned core graph G'(V, E)
```

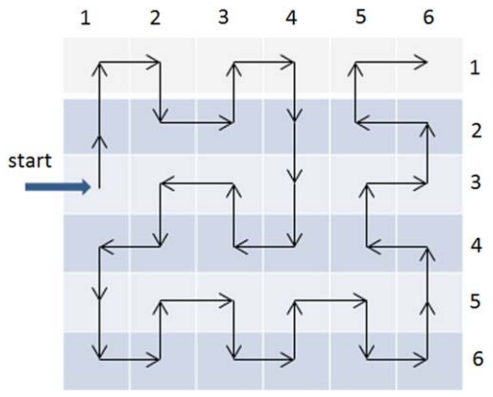

Fig. 2. Sequence of tile co-ordinates for the initial mapping on a 36 core (6 × 6) regular mesh NoC.

4.2. Core to tile mapping

In the next stage, we map cores to tiles on the die. Our proposed approach consists of two major steps: initial mapping generation, and incremental swapping. The following subsections describe the initial mapping generation, and three variants of incremental swapping.

4.2.1. Initial mapping generation

In this step, we generate an initial core to tile mapping by traversing an Inter-Island Communication Graph *IICG*(V_{isl}, E_{isl}), where the vertices constitute the entire islands and edges constitute aggregate communication bandwidth between the respective islands. A breadth-first search (BFS) starting with each of the Ω islands as the root node, would produce Ω distinct sequences of islands, each of length Ω. The order of islands in each sequence is based on decreasing communication bandwidths with the island selected as the root node. Subsequently, the cores are mapped onto the tiles of the NoC in order of the island $sequence_j$ to generate a $mapping_j$.

We follow a pre-defined sequence of tile co-ordinates for the mapping process as shown in Fig. 2. Such an ordering of the core to tile mapping grows in both the x and y directions of the mesh in a symmetrical way, thereby keeping the Manhattan distances between the currently placed core and recently placed cores shorter; as compared to, say, growing the mapping in just the x or y directions (i.e., row-wise or column-wise mapping). Furthermore, our ordering ensures that the placement of the current core is adjacent to the previously placed core in order to guarantee VI integrity. For all Ω mapping configurations generated, we perform Incremental Swapping (Sections 4.2.2–4.2.4 describe three variants of the swapping step), Routing Path Allocation (Section 4.3) and then compute the resultant total power of the network. Algorithm 2 below summarizes the key steps in the initial mapping generation procedure. It generates Ω initial mappings to constitute a *mapping-set*.

Algorithm 2. Initial Mapping Generation.

```
input: core graph G′(V, E)
1: Create an inter-island communication graph IICG($V_{isl}$, $E_{isl}$)
2: for $j=1$ to $\Omega$ do
3:     With $V_j$ as the root node; perform BFS on IICG to get a
       $sequence_j$ of islands
4:     Map the cores within each island in the order of the
       $sequence_j$ to obtain $mapping_j$
5: end for
output : mapping-set
```

4.2.2. Incremental swapping

The incremental swapping step is intended to reduce link tension (Definition 3) in the NoC. Cores connected by communication links with greater tension have a higher force of attraction between them. The tensions in the mesh network force cores with high communication bandwidths to come closer during the incremental swapping step. After the initial mapping, all communicating cores have some tensions associated with them, in one or more directions. The incremental swap algorithm attempts to decrease the Manhattan distance of the core with the highest tension in the entire mesh by swapping it with another core to reduce the tension. The swaps are allowed to occur between neighbors either in the x-direction, y-direction or diagonally on the mesh structure. A swap is considered valid if it can pass three checks:

1. *Total tension decrease check:* the swap is valid only if it will result in a decreased total tension (total pre-routing traffic).
2. *VI integrity check:* the swap is valid only if it will not disintegrate any islands.
3. *Tabu-direction check:* the swap is valid only if the direction of the move is not in the Tabu list.

If a swap does not pass the checks, the swap is aborted, and the core is marked-off to restrict it from swapping for the next d (dimension of the mesh) number of swap attempts (iterations). If a swap is valid, then the x, y tensions and x, y co-ordinates of the cores are updated after the swap, and the complementary move direction for the core is added to a Tabu list (e.g., -y if the core moved in the $+y$ direction) so that the core will never move back in the direction it came from. If a diagonal swap takes place, both the x and y directions are added to the Tabu list. Also, the total pre-routing traffic is updated after every swap. The incremental swapping continues until d consecutive aborts have been encountered. This state of the NoC, where valid swaps, which reduce total NoC tension are no longer readily available, is defined as equilibrium.

Algorithm 3 below describes the key steps in the incremental swap algorithm, which eventually decreases the Manhattan distances of high bandwidth communication flows in the mesh NoC. The input is the mapping set for which each of the Ω constituent mappings is processed, to create a final mapping-set. Note that cores are never swapped back in the direction of their previous locations; this gives us a theoretical upper bound on the total number of swaps that the N cores in the mesh can undergo: $O(N*2*(N^{1/2}-1))$.

Algorithm 3. Incremental Swapping.

```
input: core graph G′(V, E) and mapping-set
1: for each mapping-solution $\in$ mapping-set do
2:     do until $d$ consecutive aborts are encountered
3:         Choose the core with the maximum total tension
4:         Choose the direction with the most tension
5:         Perform the three pre-swap checks
6:         If the swap is invalid, consider other directions
7:         If no directions valid, abort; mark-off the core for next $d$
           iterations
8:         If some direction is found to be valid, perform the swap
9: end for
output : final-mapping-set
```

4.2.3. Probabilistic incremental swapping

On an initial mapping solution, instead of just considering the core under most tension for the next swap (as discussed in Section 4.2.2); probabilistic incremental swapping selects a core for the next swap out of up to p cores which are under most tensions. The p cores under highest cumulative tensions in the NoC having at least one 'valid swap' (i.e., satisfying the 3 pre-swap checks) in the direction of their net x-y tension are considered for the next swap. Here, p is a parameterizable value defined as an upper bound on the number of cores in the mesh that become candidates for the next swap. The pseudo code (Algorithm 4) is given below, which is run on each Initial Mapping (IM).

Algorithm 4. Probabilistic Incremental Swapping.

```
input: core graph G'(V, E) and mapping-set
1: for each mapping-solution (IM) ∈ mapping-set do
2:   do K times
3:     do until d consecutive aborts are encountered
4:       Find p cores under most cumulative tensions
5:       For the p cores, find q cores which have at least one
         valid swap
6:       If no valid swaps found (q=0), abort; mark-off all the
         p cores for next d iterations and goto next iteration
7:       Probabilistically, choose one of the q cores for the
         next swap
8:       Perform a valid swap in the direction of most tension
9: end for
output : K final mapping candidate solutions for each IM
```

Each of the candidate cores are assigned swap probabilities for the next swap, proportional to the value of cumulative tensions they are under. The swap probabilities are computed as follows. Let q be the number of candidate cores for the next swap, where $q \le p$. Let T_1, T_2, …, T_q be the cumulative or net tensions associated with the q swapping candidates and $T = T_1 + T_2 + \ldots + T_q$ be the sum of q tensions. Then, the swap probabilities for the q candidate cores P_1, P_2, …, P_q become:

$$P = T_1/T,$$
$$P = T_2/T,$$
$$\vdots$$
$$P = T_q/T.$$

For every swap, the q probabilities are computed and one of the candidates is selected based on the respective probabilities for the next swap; a valid swap is performed in the direction (x, y or diagonal) of most tension. The procedure of probabilistic incremental swapping continues until equilibrium is reached. This procedure is performed K times (as shown in Fig. 3; K is a designer specified parameter) for each initial mapping solution to produce $K * \Omega$ final mapping solutions, out of which the one corresponding to least communication power (obtained after performing routing path allocation on each of the final mapping solutions) is selected as the final NoC synthesis solution for the particular value of p.

4.2.4. Incremental swapping with branch & bound (BB)

The probabilistic swapping approach (Section 4.2.3) selects a single core (out of p probable candidates) for every swap and performs this procedure iteratively until equilibrium is reached. Therefore, it is a sequential process where just one mapping solution (intermediate or final solution) exists at any given time during execution. We also propose a branch and bound based incremental swapping approach where multiple mapping solutions co-exist during execution. In the BB approach, up to n mapping solutions are branched out from an initial or an intermediate mapping solution in the search tree (as shown in Fig. 4).

The BB technique combines random search (constituting of random swaps) with directed search (constituting of directed swaps) to generate multiple final mapping candidate solutions. In the 'directed search', the set of next swaps are determined by the current link-tension map of the NoC, while a set of random swaps is selected in the 'random search'. The directed swaps are geared to reduce the highest tensions in the NoC in order to reduce communication power; where a 'best swap' swaps the core under most tension (on the NoC link-tension map) in a direction of most tension. We combine the directed swaps with random swaps for effective exploration of the solution search space. Note that random swaps are performed within the same island (to satisfy *VI* integrity requirements). Also, total tension/Tabu-direction checks are not considered and Tabus for the cores participating in the swap are reset during random swaps.

Let n be the maximum branching degree and K an upper bound on the total number of final candidate solutions which is a multiple of n-1; α be a positive fraction which governs the weight of the random component in BB, and C be the number of current mapping solutions. The pseudo code for the BB procedure (Algorithm 5) is given below, which is run on each Initial Mapping (IM).

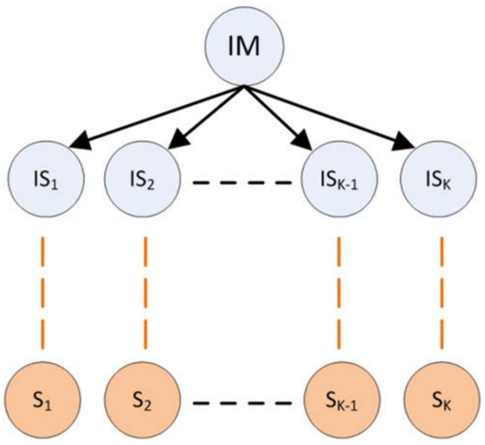

Fig. 3. Probabilistic incremental swapping technique for a single Initial mapping (IM). IS_x are the intermediate mapping solutions and S_1-S_K are the K final mapping solutions.

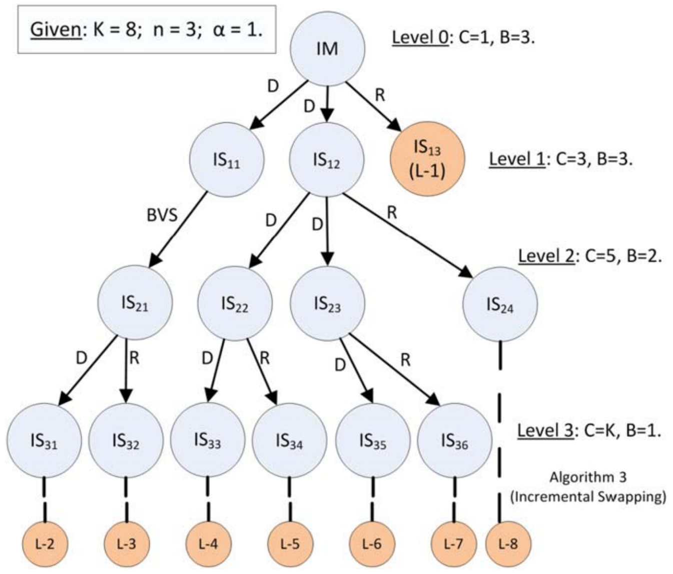

Fig. 4. BB search tree for incremental swapping from a single IM. IS_{ij} are intermediate mapping solutions at level i, L-x are leaf nodes; R (or D) represents a random (or directed) swap branching-out a new mapping solution.

Algorithm 5. Incremental Swapping using BB.

```
input: core graph G'(V, E) and mapping-set
1: for each mapping-solution (IM) ∈ mapping-set do
2:   while ((C < K) && (at least one non-leaf node exists in C));
  do ∀ non-leaf nodes on the current BB level {3: Compute B, R
  and D
4:     Find out the D best swaps (directed search) and check
  their validity
5:     If one or more valid swaps found, proceed to step 8
6:     Find the best valid swap while considering all cores
7:     If no swap is valid, mark this candidate (node) as a leaf;
  else execute swap (branch out child) and delete current
  node; then goto next iteration
8:     Compute the R random valid swaps
9:     Execute computed random and directed swaps,
  branching out a new child for each swap and delete current
  node; then, goto next iteration }
10:  ∀ non-leaf nodes, run Algorithm 3
11: end for
output :Up to K final mapping candidate solutions for each
  IM
```

At any level of a B-way search tree (B is variable representing current degree of branching) of intermediate mapping solutions, D best swaps and R random swaps are considered for each node. The branching degree for any swap, B is computed according to the following equation:

$$B=(n+1)-(n-1)*(C+1)/K$$

With K as an upper bound on the total number of final candidate solutions, the branching degree proportionally decreases with increasing number of intermediate solutions. The number of directed swaps and random swaps that could be branched out are computed from the value of B as follows:

$$R=\alpha*B/2$$

$$D=B-R$$

At each BB node, only D cores (with D highest tensions) are considered for swapping. If no valid directed swaps are available for the D cores under consideration, no random swaps are branched out either and a ‘Best Valid Swap’ (BVS) is attempted on the current solution (IS_{11} & IS_{13} in Fig. 4). The BVS swaps the core under most tension (on the NoC link-tension map) for which a valid swap exists, in a valid direction of most tension. The intermediate mapping solution (IS_{21} in Fig. 4) obtained from the BVS could potentially participate in the BB search once again, as one of the D best swaps might now be valid on the updated solution. On the other hand, when no more directed swaps are possible, i.e., no BVS is available; the intermediate solution node (IS_{13}) becomes a leaf node (*L-1*), signifying a final candidate solution, and is never again considered for further swaps.

When the existing number of solutions in the BB search reach the upper bound of K, only the best swaps are made on all the non-leaf solutions (B is reduced to 1) until they converge to equilibrium, i.e. Algorithm 3 (Incremental Swapping) is run on each of the non-leaf solutions ($IS_{24}, IS_{31\text{-}36}$). Note that, running Algorithm 3 on a solution which has no BVS available is redundant and therefore such a node is marked as a leaf (*L-1*). Alternatively, if no non-leaf solutions remain, BB terminates as no random swaps are allowed on leaf-nodes. Finally, a set of up to K final mapping candidates are obtained from a single initial mapping. Out of the $K*\Omega$ mapping candidate solutions, the one corresponding to least communication power (obtained after performing the routing path allocation on each of these solutions) is selected as the final NoC synthesis solution for the particular value of n.

4.3. Routing path allocation

The mapped NoC obtained after the previous stage consists of multiple VIs that not only run on different voltage levels, but also different frequencies. Therefore, whenever an inter-island link is inserted, voltage level converters (VLCs) and frequency level converter resources (MCFIFOs) are required in the corresponding routers. Whenever a low voltage core transmits to a higher voltage core, a VLC is needed on the outgoing port of the source router. Also, for any inter-island link, an MCFIFO is needed for the higher frequency/voltage core as the connecting link works at the lower frequency. These frequency and voltage conversion components incur an overhead in terms of power dissipation and delay. Thus, the main objective of routing is to find a path for each communication flow such that communication path lengths and the number of inter-island links are reduced.

The routing algorithm in [9] minimizes the number of inter-island links by calculating the inter-island bandwidths between different neighbor-islands and inserting a proportional number of links between them. Then, the inter-island routing is carried out by using just the inserted links. One of the drawbacks of this approach is increased traffic because of less flexible link insertion that is performed once and never changed again during the disjoint routing step, leading to long communication path-lengths. Because the link-insertion and routing steps are disjoint, communication between non-neighbor islands is also not considered during the link-insertion step, giving rise to a possibility of creating isolated islands that do not communicate with any of their neighbor-islands and therefore are unable to communicate with non-neighboring islands.

Our proposed routing algorithm integrates the link-insertion and the routing steps thereby alleviating the above mentioned drawbacks from [9]. We employ minimal routing in order to minimize total traffic. The inter-island links are inserted only when minimal paths cannot be found within the residual bandwidth capacities of the existing inter-island links. Fig. 5 shows how the disjoint link-insertion and routing approach from [9] has longer path lengths because the routing is constrained to the available inter-island links (shown with the thick black arrows); whereas, with an integrated approach, path lengths would be optimal. As the inter-island link does not lie in the direct path between the source (2, 2) and the destination (4, 2) in (a); path lengths are longer when compared to (b).

For each communication flow, we consider all candidate minimal paths. Out of these candidate minimal paths, we choose

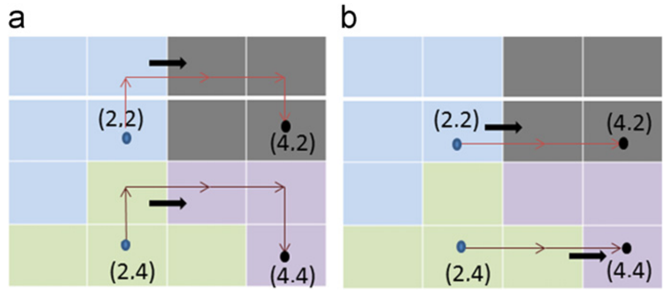

Fig. 5. Cores of the same color belong to the same VI. Shown are two communication flows (a) with the algorithm in [9], (b) by integrating link insertion and minimal-path routing.

a routing path based on the following optimization objectives (in that order):

1) Minimize total number of inter-island link insertions needed on the path
2) Minimize total number of intra-island link insertions needed on the path

The communication flows with longer minimal paths have more choices for routing and thus have a larger scope for optimization. Also, flows with smaller bandwidths, require lesser residual capacities to be accommodated within existing links. Therefore, communication flows are sorted in the increasing order of their path lengths, in decreasing order of their communication bandwidths for the same path length; and routed in that order. While routing any communication flow over a given path, links insertions are performed whenever the existing link(s) cannot support the bandwidth of the current flow or if no links are available. In summary, this routing scheme optimizes both the number of inter-island links and total traffic in the NoC, thereby resulting in significant savings in communication power. Finally, a voltage-assigned, mapped and routed NoC is obtained. Algorithm 6 below summarizes the routing path allocation approach.

Algorithm 6. Routing Path Allocation.

```
input: core graph G′(V, E), final-mapping-set
1: Sort all communication flows in the increasing order of path
   lengths and in decreasing order of bandwidths for the same
   path length
2: for each communication flow in sorted list do
3:     Out of all the candidate minimal paths, choose the set of
       paths that would require the least number of inter-island
       link insertions
4:     Out of the chosen paths, choose the one path that would
       result in the minimum number of intra-island link insertions
5:     Insert the necessary links and allocate the current flow
       bandwidth over the chosen routing path.
6: end for
output :Synthesized NoC with all communications routed
```

5. Experiments

5.1. Experimental setup

We performed several experimental studies and generated NoCs for different applications to evaluate the quality of solutions generated by our VI-aware NoC synthesis frameworks: *VISION* (uses the basic incremental swapping approach for tile-to-core mapping) and its variants *VISION-P* (uses probabilistic incremental swapping) and VISION-B (uses branch and bound based incremental swapping). All three of the proposed frameworks use the same VI-partitioning and routing path allocation schemes, which are discussed in this paper.

We used the ARM11 MPCore multi-core processors [22] as the base compute cores in our experiments, which support six operating voltage levels as shown in Table 1. Correspondingly, we constrained the maximum number of VIs to be synthesized by our framework to six. Our experiments were conducted on three applications based on pseudo-random core graphs derived using TGFF [23], and consisting of 16 (4×4), 36 (6×6), 64 (8×8) and 100 (10×10) cores, with edge weights annotated with bandwidths representing the inter-core communication requirements. The diverse core counts allow us to ascertain the scalability and

Table 1
Core voltages and the corresponding frequencies and average power values used in the experiments.

V (volts)	1.26	1.2	1.15	1.1	1.0	0.9
Freq. (MHz)	483	437	401	368	304	246
Power (mW)	126	101	85	72	49	32

applicability of our approach to low and high complexity systems. We conservatively assume that the square of the voltage scales linearly with the frequency, as in previous works [24]. The compute core power values account for both dynamic and leakage power, and vary on average as shown in Table 1. The router and link powers for different voltages, frequencies and router complexities; and with varying communication loads are obtained from a modified version of ORION 2.0 [25]. The voltage/frequency assignment for a router is the same as the voltage/frequency of the core it is connected to in its corresponding tile. The voltage/frequency pair value for a link connecting two routers is the lower voltage/frequency pair value of the two routers connected by the link. As the VLCs and MCFIFOs required to interact between VIs incur a power overhead that is proportional to their voltage supply, we assume that every MCFIFO or VLC introduced in the router consumes 10% of the base router power, based on reported overheads from existing literature [26]. For deadlock avoidance, we use two virtual escape channels per router [27], with appropriate power overhead in our experiments.

For probabilistic incremental swapping in *VISION-P* and the branch and bound in *VISION-B*, a value of $K=400$ is used. Therefore, for same number of islands, the upper bounds on the total number of candidate mapping solutions are the same for both the procedures.

In *VISION-B*, our goal with the BB search tree is to decrease branching degree B in proportion to the increasing number of existing mapping solutions C. Therefore, in order for B (range: n down to 1) to span approximately uniformly over the range of C (1 to K), in our experiments n takes values where (n-1) is a factor of K. $(n-1)$ thus takes values of 2, 4, 8, 10, 16, 20, 25, 40 and 50 (all of which are factors of 400). Also, a value of $\alpha=1$ is used in the BB procedure.

In *VISION-P*, with increasing values of p during probabilistic incremental swapping, more cores are considered as potential candidates for the next swap; therefore, cores under relatively low tensions can be selected for the next swap with higher likelihood (even though with proportionately low probabilities). At the same time, with higher values of p, the search space is not restricted to the swaps constituting of just the cores with highest tensions. In our experiments, we have found that relatively low values of p (less than 20% of N) yield the best results in terms of communication power in the NoC. The results shown for *VISION-P* later in this section utilize the following empirically derived optimal values of p: 3, 7, 12 and 20 for NoC sizes of 16, 36, 64 and 100, respectively.

5.2. Results

We compare the results of our synthesis frameworks with the results obtained by using VI-aware NoC synthesis approaches presented in two prior works: a heuristic based recent work on VI-aware regular NoC synthesis [9] that claims to improve upon other previous works such as [18]; and a synthesis approach based on a genetic algorithm (GA) in [16]. We implemented the frameworks presented in [9,16] to the best of our understanding and used the same performance and power models for all implemented approaches to ensure a fair comparison of the algorithms.

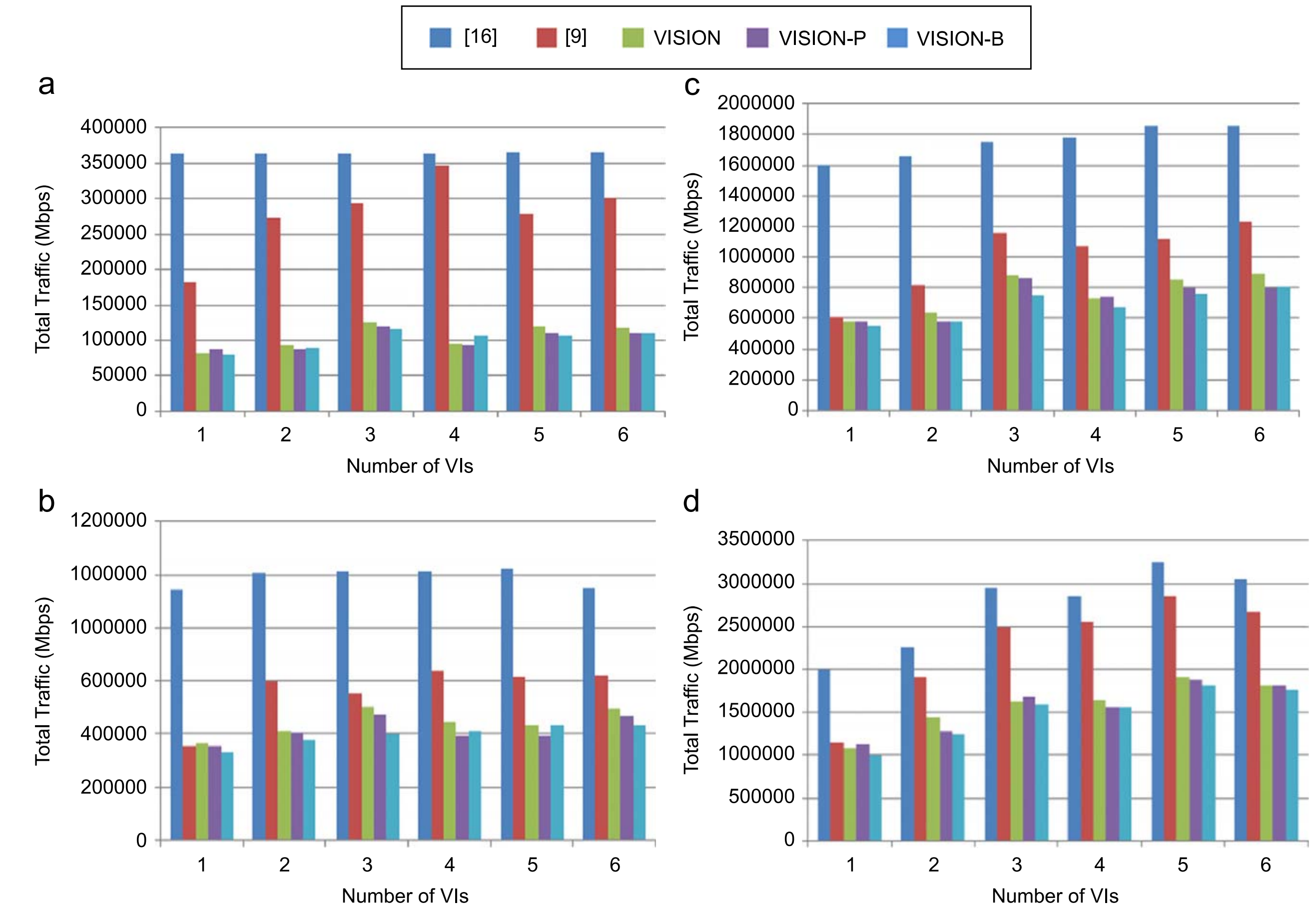

Fig. 6. Total traffic in (a) 16 core mesh, (b) 36 core mesh, (c) 64 core mesh, (d) 100 core mesh.

Fig. 6(a)–(d) shows the total traffic that exists in the results generated (for the four applications considered) by the five approaches: GA-based [16], heuristic based [9], and 3 of our proposed frameworks: VISION, VISION-P and *VISION-B*. Results are generated for a range of input Ω values that vary the number of VIs to be generated in the solution. The x-axis shows the solutions for the different input values of Ω. It can be seen that our frameworks have significantly lower traffic compared to approaches proposed in prior work. In particular, VISION-B generates solutions with the lowest traffic compared to [9] (by up to 62%), as well as [16] (by up to 78%). This is because the core-to-tile mapping approach in *VISION-B* not only utilizes the link-tension methodology for directed search, but unlike *VISION-P* it also permits random swaps, which are not necessarily between neighbors. Therefore, *VISION-B* provides for a more systematic and efficient exploration of solution search space resulting in lower traffic. The repartitioning approach used in all three of our approaches clusters the heavily communicating cores into the same VI, with a constraint on the hike in the communication power. This not only results in a reduction in the number of inter-island links, but also reduces the average Manhattan distance over all communication paths; as any two cores in separate islands are more likely to be farther apart than if they are within the same island. Our incremental swap mapping always attempts to reduce the communication Manhattan distances for the heaviest communication flows; leading to much lower traffic. Lastly, our routing path allocation algorithm also produces significant reduction in total traffic by prioritizing reduced path lengths. In comparison, the mapping approaches in [9,16] do not consider reducing the Manhattan distances over the entire network in a holistic manner. Also, unlike our approach, [9] does not employ minimal routing to reduce the total number of inter-island links.

Table 2 shows the total number of inter-island links (accompanied by corresponding VLCs and MCFIFOs) generated by the five approaches. For most cases, the GA based approach [16] has the lowest number of inter-island links compared to [9] and our approaches. However, [16] accomplishes this by having many more cores being assigned higher voltages thereby reducing the variability of the voltage distribution of the cores. Thus the number of inter-island links is much lower but with highest penalties on computation power. On average, our synthesis frameworks have fewer inter-island links compared to [9], particularly because our routing path allocation considers all candidate minimal paths and prioritizes the paths with the least number of inter-island links. Also, as the core-to-tile mapping in *VISION-P* and *VISION-B* further optimize for communication power (over VISION), they produce even fewer inter-island links.

Fig. 7 summarizes the total power dissipation of the solutions generated by the five approaches, for the four applications considered. A decomposition of the total power into computation and communication power is also presented. As the number of islands in the NoC increases, it can be seen that the total power decreases. This decrease in total power is due to the decreasing computation power, because with more voltage levels available, the cores in general can run at lower voltages. With reductions in total traffic as well as inter-island communication, the communication power for our proposed frameworks show quite significant improvements; for instance, our best performing *VISION-B*

Table 2
Total number of inter-island links for the configurations with least communication power.

n/w sizes	Core_16					Core_36				
# of VIs	[16]	[9]	*VISION*	*VISION-P*	*VISION-B*	[16]	[9]	*VISION*	*VISION-P*	*VISION-B*
1	0	0	0	0	0	0	0	0	0	0
2	16	12	9	6	6	22	18	19	10	11
3	20	20	12	12	12	37	33	25	22	30
4	23	35	16	14	12	40	33	24	28	21
5	32	33	18	17	16	42	50	41	45	30
6	25	38	22	22	22	76	64	33	34	34
n/w sizes	Core_64					Core_100				
# of VIs	[16]	[9]	*VISION*	*VISION-P*	*VISION-B*	[16]	[9]	*VISION*	*VISION-P*	*VISION-B*
1	0	0	0	0	0	0	0	0	0	0
2	25	43	36	28	34	35	46	34	52	37
3	34	48	45	32	40	68	69	93	82	80
4	33	50	44	48	40	74	90	94	77	85
5	56	67	54	52	51	90	99	118	111	103
6	58	69	61	63	54	92	112	120	110	105

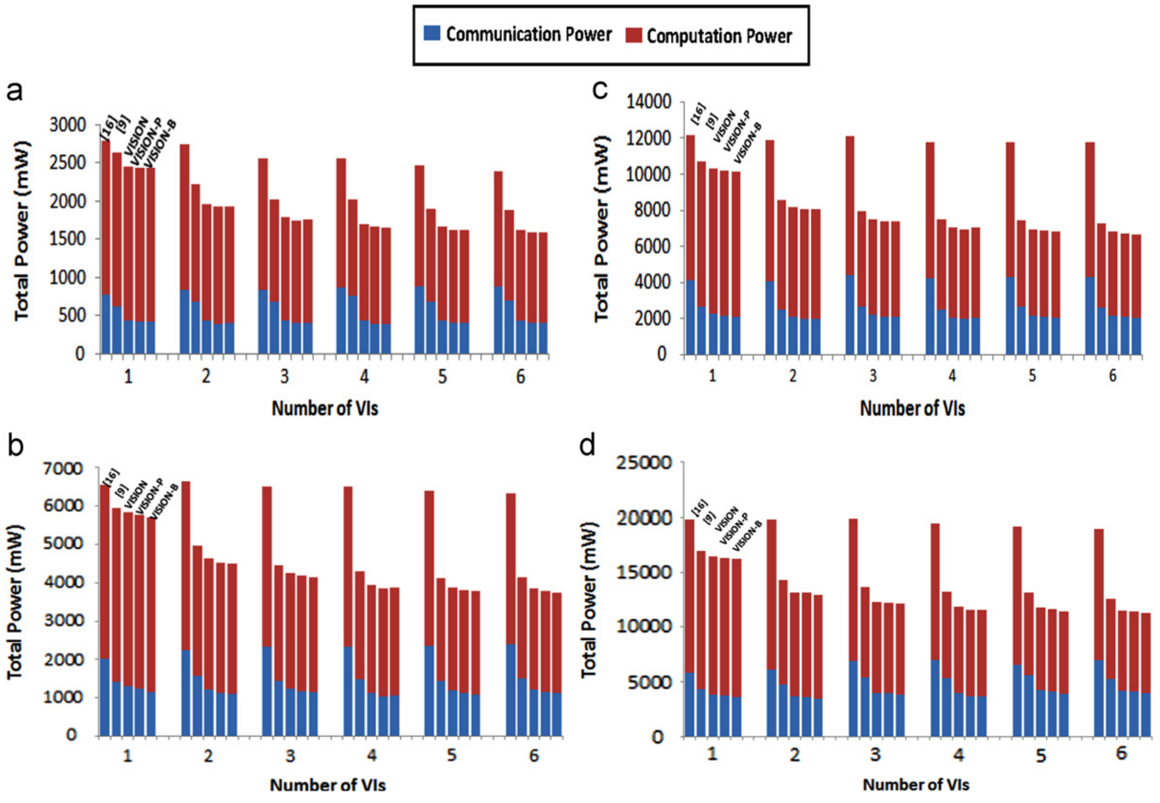

Fig. 7. Total power (compute+communication) (a) 16 core mesh, (b) 36 core mesh, (c) 64 core mesh and (d) 100 core mesh.

framework improves by up to 32% over [9] and by up to 54% over [16]. Even though most of these gains are obtained due to our more communication-centric partitioning and mapping techniques, for larger applications (where the total communication power is a more significant portion of the total power, due to longer communication paths), the routing technique used provides for significant improvements as well. In terms of total power dissipation, the *VISION-B* framework outperforms both [9] (by up to 13%) and [16] (by up to 41%). These results clearly indicate that our proposed approach can provide superior solutions when compared to the best known prior works, and is thus a promising set of tools for automated exploration and synthesis of emerging VI enabled homogenous multi-core NoC systems.

Finally, we present a sensitivity analysis for the value of maximum branching degree n in *VISION-B* in Fig. 8(a)–(d). The impact on total communication power dissipation of different values of n across different number of VIs is presented. Note that compute power does not change when the value of n is varied; therefore we only show changes in communication power. For the branch and bound approach, the total number of final candidate solutions is ($K*\Omega$); therefore the solution search space increases linearly with the maximum allowable number of VIs. With

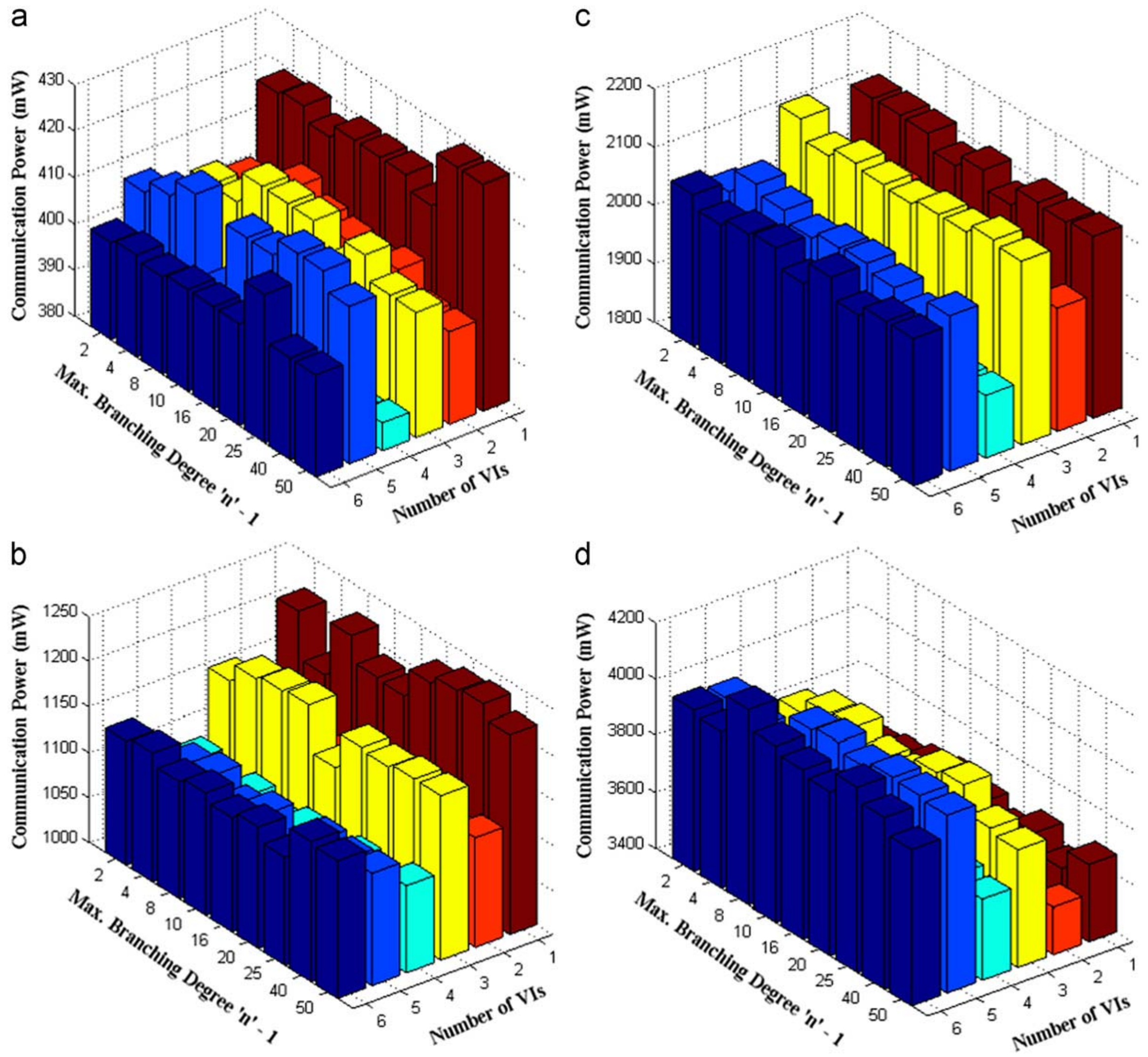

Fig. 8. Sensitivity analysis of total communication power with VISION-B over a range of values of maximum branching degree n, across different number of VIs for (a) 16 core mesh, (b) 36 core mesh, (c) 64 core mesh and (d) 100 core mesh.

increasing number of VIs, routers and links can in general run at lower voltage/frequency levels, thus lowering the communication power; at the same time, additional MCFIFOs and VLCs needed for inter-island communication will add-up to the total communication power in the NoC. The additional components needed for inter-island communication is the reason why communication power does not monotonically decrease with increasing number of VIs, but rather shows an uneven trend. As the maximum degree of branching n increases, the BB search spawns more children out early in the search, but ends sooner with relatively less number of total swaps. Note that irrespective of the value of n, the incremental swapping is performed at each non-leaf node at the end of the BB procedure.

Even though from Fig. 8(a)–(d), no obvious pattern of communication power in relation to varying n may seem to emerge; it can be observed that one of the best (if not the best) communication power values are almost certainly obtained by using values of 2, 4 or 8 for $(n-1)$, even for the 100-core scenario. Therefore, if simulation time is constrained (which is the case for most real world design flows rushing to meet time-to-market deadlines), the authors recommend simulating for a few relatively small values of n to achieve an optimized solution.

6. Conclusion

In this paper, we have proposed a set of novel synthesis frameworks (*VISION*, *VISION-P* and *VISION-B*) for VI-aware synthesis of mesh-based NoCs. Our partitioning techniques consider not only the optimization of computation power, but communication power as well. Our mapping approach uses a distributed decision making over the whole mesh; instead of a centralized approach used in previous works. Our routing path allocation algorithm integrates link-insertion and routing in contrast to previous work. Our best performing framework *VISION-B* produces up to 32% savings in communication power and up to 13% savings in total power; compared to the best known prior work on VI-aware NoC synthesis.

References

[1] S. Vangal et al., An 80-Tile 1.28 TFLOPS Network-on-Chip in 65 nm CMOS, Proceedings of the IEEE International Solid-State Circuits Conference., pp. 98–589, Feb. 2007.
[2] Tilera Corporation. TILE64^{TM} Processor. Product Brief. 2007.
[3] L. Benini, G. De-Micheli, Networks on Chip: A New SoC Paradigm, Proc. Computer 49 (1) (2002).

[4] J.M. Rabaey, Digital Integrated Circuits, Prentice Hall, 1996.
[5] F. Fallah, M. Pedram, Standby and active leakage current control and minimization in CMOS VLSI circuits, IEICE Transactions on Electronics 88 (4) (2005) 509–519.
[6] T. Chelcea, S. Nowick, A low latency FIFO for mixed-clock system, Proc. IEEE Workshop on VLSI, pp. 119–126, April 2000.
[7] P. Choudhary, D. Marculescu, Hardware based frequency/ voltage control of voltage frequency island systems Proc. CODES+ISSS, pp. 34–39, 2006.
[8] D. Lackey et al., Managing Power and Performance for System-on-Chip Designs using Voltage Islands, in: Proc. ICCAD., pp. 195–202, Nov. 2002.
[9] W. Jang, D. Ding, D. Pan, A Voltage-Frequency Island Aware Energy Optimization Framework for Networks-on-Chip Proc. ICCAD, pp. 264–269, Nov. 2008.
[10] S. Murali et al., Designing Application-Specific Networks on Chips with Floorplan Information, Proc. ICCAD., pp. 355–362, Nov. 2006.
[11] C. Seiculescu, S. Murali, L. Benini, G. De-Micheli, NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs, Proc. DAC, pp. 822–825, July, 2009.
[12] P. Zhou, P. Yuh, S. Sapatnekar, Application-Specific 3D Network-on-Chip Design Using Simulated Allocation, Proc. ASPDAC., pp. 517–522, Jan., 2010.
[13] K. Srinivasan, K. Chatha, A low complexity heuristic for design of custom network-on-chip architectures Proc. DATE, vol. 1, pp. 1–6, March 2006.
[14] K. Srinivasan, K. Chatha, G. Konjevod, Linear-Programming-Based Techniques for Synthesis of Network-on_Chip Architectures, IEEE Transactions on VLSI systems (TVLSI) 14 (4) (2006).
[15] U. Ogras, R. Marculescu, It's a Small World After All: NoC Performance Optimization Via Long-Range Link Insertion, IEEE Transactions on VLSI systems (TVLSI) 14 (7) (2006).
[16] L. Leung, C. Tsui, Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands, Proc. DAC, pp. 128–131, June 2007.
[17] P. Ghosh, A. Sen, Efficient Mapping and Voltage Islanding Technique for Energy Minimization in NoC under Design Constraints Proc. SAC, pp. 535–541, 2010.
[18] U. Ogras, R. Marculescu, P. Choudhary, D. Marculescu, Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip, Proc. DAC, pp. 110–115, 2007.
[19] J. Hu, R. Marculescu, Communication and task scheduling of application-specific networks-on-chip, IEE CDT 152 (5) (2005) 643–651.
[20] C. Chou, U. Ogras, R. Marculescu, Energy and Performance-Aware Incremental Mapping for Networks on Chip with Multiple Voltage Levels, IEEE Transactions on CAD (TCAD) 27 (10) (2008) 1866–1879.
[21] J. Hu, R. Marculescu, Energy and Performance-Aware Mapping for Regular NoC Architectures, IEEE TCAD 24 (4) (2005) 551–562.
[22] ⟨http://www.arm.com/products/processors/selector.php⟩ last accessed: August 11, 2011.
[23] ⟨http://ziyang.eecs.umich.edu/~dickrp/tgff/⟩ last accessed: August 11, 2011.
[24] S. Murali, et al., Mapping and configuration methods for multi-use-case networks on chips Proc. ASP-DAC, pp. 146–151, 2006.
[25] A. Kahng, B. Li, L.-S. Peh, K. Samadi, ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration, Proc. DATE, pp. 423–428, Apr. 2009.
[26] T. Chelcea, S. Nowick, A Low-Latency for Mixed-Clock Systems Proc. CSW, pp. 119–126, Apr. 2002.
[27] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kauffman, 2005.

Nishit Kapadia received a B.E. in Electronics Engineering from University of Mumbai in 2001 and his Master's Degree in Electrical and Computer Engineering at California State University, Northridge in 2005. From 2006 to 2008, he worked as a VLSI faculty at the Department of Electronics, Sardar Vallabhbhai National Institute of Technology at Surat, India. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at Colorado State University. His research interests are CAD for Multi-Core 2D and 3D Systems, Networks-on-chip, Power, Thermal and Variation-Aware Design Methodologies and VLSI Automation.

Sudeep Pasricha received the B.E. degree in Electronics and Communication Engineering from Delhi Institute of Technology, Delhi, India, in 2000, and the M.S. and Ph.D. degrees in Computer Science from the University of California, Irvine, in 2005 and 2008, respectively. He is currently an Assistant Professor at Colorado State University, Fort Collins with a joint appointment in the Department of Electrical and Computer Engineering, and Department of Computer Science. His research interests are in the areas of embedded systems and chip multiprocessors, mobile computing, system level CAD algorithms, networks-on-chip (NoC) and emerging interconnect technologies (photonic, carbon nanotube, 3D), scheduling theory, and secure, low-power, thermal-aware, and fault-tolerant computing. Dr. Pasricha is currently an Advisory Board member of ACM SIGDA, Technical Program Committee member of the DATE, CODES+ISSS, ISQED, NOCS, NoCArc, VLSI Design, and GLSVLSI conferences and the SIGDA DAC PhD Forum, and Organizing Committee Member of the AICCSA and ICECS conferences and ICCAD CADathalon. He was the recipient of the Best Paper Award at the IEEE ISQED 2010 and ACM/IEEE ASPDAC 2006 conferences, and a Best Paper Award nomination at the ACM/IEEE DAC 2005 conference.