Application Specific Networks-on-Chip Synthesis: An Energy Efficient Approach

Somayeh Kashi*, Ahmad Patooghy‡, Dara Rahmati†, Mahdi Fazeli*, Michel A. Kinsy‡
*Department of Computer Engineering, Iran University of Science and Technology University, Tehran, Iran
†Institute for Research in Fundamental Sciences, Tehran, Iran
‡Department of Electrical and Computer Engineering, Boston University, Boston, USA

***Abstract*—Multiple Voltage Supply (MSV) chip fabrication is considered a viable technique for addressing the power and thermal challenges of modern many-core systems. Efficiency of this technique has been demonstrated in application specific Network-on-Chips (NoCs) which have lots of cores and various operating voltages/frequencies. In this paper, a four-phase synthesis toolchain is proposed and evaluated for the design of multi-voltage application specific NoCs. The proposed synthesis toolchain performs (i) core to router allocation, (ii) voltage islanding to match voltages of cores connected to the same router, (iii) hierarchical floorplanning to reduce the complexity of power delivery network, and (iv) path allocation to connect routers based on the application requirements. The distinguishing feature of the proposed toolchain is that, for the first time, the router allocation phase is performed prior to voltage islanding. This approach offers more flexibility and more efficiency in the multi-voltage NoC synthesis process. Experimental results on real benchmarks show that the toolchain (a) provides 63% less energy consumption and (b) produces twice as much alternative designs satisfying the benchmarks requirements when compared to existing approaches.**

***Index Terms*—application-specific chip, custom NoC synthesis, partitioning, islanding, floorplanning.**

I. Introduction

With the recent advances in VLSI fabrication technology, complex Multi-Processor System on Chips (MPSoCs) are now widely available in the marketplace. MPSoCs contain several general processing, DSP, I/O controller, and memory cores working with different voltages and frequencies. Such heterogeneous multi-core systems need an efficient on-chip communication architecture to ease data transmission between the cores. In such chips, general purpose Network-on-Chip (NoC) which is normally used in homogeneous chips [1], [5] is often not a viable solution, because MPSoCs (1) have cores with different working voltages/frequencies, (2) generally use multiple voltage islands [4], (3) may have information about the target application at the design time [2], [3], and finally (4) commonly use the router sharing technique, in which multiple cores are connected to a single router [7]–[9]. As a result, application specific NoC design and synthesis for multi-voltage MPSoCs has been investigated by researchers [7]–[9]. These application specific NoC designs directly affect many aspects of the MPSoC's performance including the area, temporal and spatial temperature, and power consumption of the chip.

The main steps of a custom NoC synthesis are core to router allocation, router to router connection and path allocation for the communication flows [7]. Seiculescu et al. proposed a synthesis flow for voltage-driven custom NoC design [7]. In their work, cores are assigned to voltage islands based on their nominal operating voltages at the first step. Then, cores are allocated to the routers, using a min-cut based partitioning algorithm in each island. Finally, the Dijkstra algorithm finds the shortest deadlock free path for each flow when establishing the router to router connections and physical links. Todorov et al. proposed a synthesis methodology which uses spectral partitioning and spanning trees to construct a custom NoC [8]. The partitioning step is done according to the communication graph, the floorplan information, and the voltage islands. In [9], Wang et al. have presented voltage-driven frameworks to do voltage island generation, voltage-driven floorplanning and post-floorplan processing. The objective of the voltage assignment is to select the least possible voltage value for cores to minimize the overhead of voltage converters under performance critical constraints. After this step, cores are assigned to the routers by a min-cut based partitioning algorithm in each island. A simulated-annealing based floorplanning is used simultaneously to determine the position of the cores and the routers during voltage-driven floorplanning.

In almost all of the previously proposed methods, the core to router assignment step is done after the voltage islanding. As a result, cores with high communication demand operating on different voltages have to be placed in different voltage islands. This increases the number of hop counts between the communicating cores and imposes a voltage conversion overhead between the islands. The former degrades the chip performance and the latter results in more power consumption and heat generation. To address the problem, our proposal is to do the partitioning step prior to voltage islanding in order to have more flexibility in the partitioning step. To prevent potential voltage fragmentations, we consider the voltage related issues during all steps of the synthesis, including partitioning, islanding and floorplanning. To the best of our knowledge, this is the first work to consider the voltage values in all the steps of the NoC synthesis flow with the concurrent goals of reducing the power consumption, delay and the complexity of the power delivery network.

The rest of the paper is organized as follows. Section II defines the NoC synthesis problem. Section III introduces the key observations motivating our proposal. Our multi-voltage synthesis toolchain is described in detail in Section IV. Experimental results are presented in Section V and finally Section VI concludes the paper.

II. Synthesis Problem Definition

The custom NoC synthesis process starts with a given communication core graph and tries to find a design satisfying application requirements. The core graph $Gcomm(C, E)$ is a directed and weighted graph where each vertex $c_i \in C$ is a core and each directed edge $e_{ij} = (c_i, c_j)$ denotes needed communication flow from core c_i to core c_j. The required bandwidth of the communication flow from core c_i to core c_j is represented by bw_{ij}; the latency constraint for the flow is represented by lat_{ij}. Each core c_i has its own minimum operating voltage v_i subject to performance constraints. The voltage difference between corresponding cores c_i and c_j is represented by $diff_{volt_{ij}} = |v_i - v_j|$.

Weight function: The weight of each edge e_{ij} ($Wcomm_{ij}$) is a combination of bw_{ij}, lat_{ij}, and $diff_{volt_{ij}}$ according to the proposed cost function of equation (1). It is used during the synthesis process.

$$Wcomm_{ij} = \begin{cases} \alpha . \frac{bw_{ij}}{max_{bw}} + \gamma . (1 - \frac{diff_{volt_{ij}}}{max_{diffvolt}}) & \\ + \beta . \frac{min_{lat}}{lat_{ij}} & lat_{ij} \neq 0 \\ \alpha . \frac{bw_{ij}}{max_{bw}} + \gamma . (1 - \frac{diff_{volt_{ij}}}{max_{diffvolt}}) & lat_{ij} = 0. \end{cases} \quad (1)$$

Parameters max_{bw}, min_{lat}, and $max_{diffvolt}$ are the maximum required bandwidth, the worst latency constraint, and the maximum voltage difference between cores, respectively. Parameters α, β, and γ, which are the weight coefficients, should satisfy the $\alpha + \beta + \gamma = 10$. According to this cost function, the communication flow between two cores with high bandwidth requirement, low latency constraint and low voltage difference would have a large $Wcomm$ value.

Partitioning Problem: Partition the $Gcomm(C, E)$ into l graph-partitions $p_1, p_2, ..., p_l$ such that 1) $p_1 \bigcup p_2 \bigcup ... \bigcup p_l = Gcomm(C, E)$, 2) $p_i \bigcap p_j = \phi, 1 \leqslant i, j \leqslant l, i \neq j$, and 3) for each partition $p_n, 1 \leqslant n \leqslant l$ sum of inter partition weights be lower than from sum of intra partition weights i.e., equation (2) be satisfied.

$$\sum_{p_m, m \neq n} \sum_{c_i \in p_n, c_j \in p_m} Wcomm_{ij} < \sum_{c_i \& c_j \in p_n} Wcomm_{ij}. \quad (2)$$

We assume that the minimum operating voltage of a core is determined based on the performance constraints of the core. Then, we build an island graph $Gvolt(V, F)$ which is a directed and weighted graph. In the island graph each vertex $v_i \in V$ is a voltage island in which the operating voltage of all cores is v_i. Therefore, the directed edge $f_{ij} = (v_i, v_j)$ connects a lower voltage island (v_i) to a higher voltage island (v_j). Each edge has a weight of $Wvolt_{ij} = (v_j - v_i) \times n_i$ (n_i is the number of cores in island v_i) and denotes the cost of merging islands v_i and v_j with respect to their power consumption. It is clear that in the case of merging two voltage islands, the lower voltage island should work with the voltage of the other island to meet the performance constraints of both islands.

As described, extending the number of on-chip voltage converters increases the power consumption and degrades the performance of the chip. To limit the number of on-chip voltage converters, we need a way to merge some the adjacent voltage islands. This problem is defined as follows.

Voltage Island Merging Problem: Find the smallest set U in the $Gvolt(V, F)$ graph, $U \subset V$, such that 1) $\sum_{i \& j \in U} Wvolt_{ij}$ is minimized, and 2) by merging the islands of set U to their higher voltage islands, the power constraint of the application is satisfied. When such a subset is found, the merging continues till at most two islands remain in the set U.

III. Observations & Motivations

Considering the existence of several heterogeneous cores on most MPSoCs, multiple-supply voltage (MSV) technique [7] can be effectively used to reduce the power consumption of MPSoCs. In an MSV enabled chip, the performance-critical cores normally have to work with higher supply voltages to meet their performance constraints, while other cores have the chance to work with lower supply voltages. To make data communication feasible between the cores working with different voltages, the use of voltage converters have been proposed [9]. However, even if there are no physical limitations, voltage converters imply significant overheads on both communication power and packet delivery delay. These overheads are obviously in contrast with the main objective of MSV enabled MPSoC design.

A communication core graph with 26 IP cores, D-26-media, [6] is shown in Figure 1. The color of each vertex here represents the working voltage of the core and the numbers next to edges represent the required bandwidth between communicating cores. Some pairs of the cores with different voltages, e.g., (DMA, MEM5), (MEM3, PE1), (ARM, SDRAM1), demand a high communication bandwidth. However, if the voltage islanding takes place prior to the partitioning, cores with different operating voltages and at the same time high communication demands will be placed in different voltage islands. To support this claim, we synthesized 3 real communication core graphs of D-35-bot, D-36-4, and D-38-VOPD under the two policies of 1) single voltage MPSoC, and 2) MSV enabled MPSoC with no limitation on the number of voltage converters. Results of power delay product which simultaneously shows power consumption and delay are depicted in figure Fig. 2. One can conclude from this figure - across all the synthesized benchmarks - that the MSV enabled chips have significantly more power delay product than the single voltage chips. This confirms that in application specific NoC synthesis it is needed to 1) limit the number of voltage converters and 2) place communicating cores as close as possible regardless of their working voltages. Driven by these requirements, our proposed toolchain performs the core partitioning phase prior to the voltage islanding. The proposed synthesis toolchain is described in the Section IV.

IV. Proposed Synthesis Toolchain

In this section, we describe the proposed toolchain for application specific NoC synthesis. The toolchain flow follows four steps to reach designs that satisfied the power and performance constraints of the given application. Unlike other methods, we

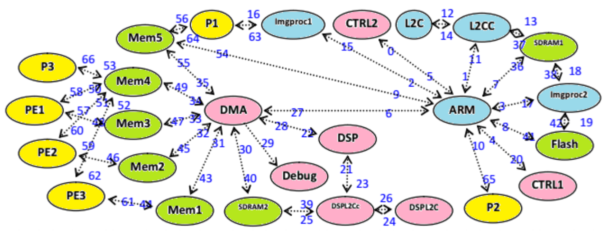

Fig. 1: A sample application core graph with multiple voltages.

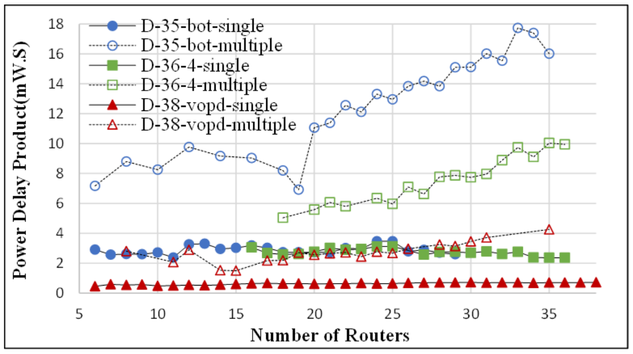

Fig. 2: PDP comparison between single and multiple voltage syntheses without limitation on the number of voltage converters.

start with core partitioning, without any assumption on the floorplan of the cores.

A. Core to Router Allocation

The first step of the proposed toolchain assigns cores to on-chip routers based on the bandwidth requirement of cores. When designing an NoC based system for a given class of multi-core applications, the number of routers could theoretically vary from a single global router to one per core. While use of several-port routers will consume more power due to the complex switching structure, using many small routers in turn increases the hop count as well as the switching activity, which inevitably increases the power needed to transfer a packet. Therefore, for a given class of applications, it is necessary to elaborate different scenarios based on the constraints to find the best switching structure. For this purpose, the core to router allocation step tries to assign the cores with high communication bandwidth and/or tight latency demands to the same partition, using a unique router for each partition [7].

Equation (1) calculates $Wcomm_{ij}$, the weight of each communication flow e_{ij}. The parameters α, β, and γ are either set by the designer based on the application characteristics or needed iterations over a range of values. In this case, the objective is to meet the specified constraints.

The number of routers (e.g. i) iterates from one to the number of cores. For every value of i, the input communication graph $Gcomm(C, E)$ is partitioned into the number of i min-cut partitions, namely $Gcomm(C_i, E_i), 1 \leq i \leq n$. To describe intuitively, the partitioning is done in such a way that, the edges that are cut among the partitions will have lower weights than the remaining edges. The number of vertices assigned to each partition is the same as defined in Equation (2). As a result, the communication flows with higher

Algorithm 1 Merging Island

```
function MERGEISLANDS()
    for $i \leftarrow 0$ to $|partitions|$ do
        Islanding Partition(i);
        $IslandNum \leftarrow$ MakeIslandGraph();
        while $IslandNum < 2$ do
            integrate vertices belong to the least weighted edge
            $IslandNum \leftarrow$ UpdateIslandGraph();
        end while
        if $EdgeWeight > Threshold$ then
            break to two partitions
        else
            integrate to one partition
        end if
    end for
end function
```

Fig. 3: Pseudo-code of merging voltage islands.

bandwidth requirements or tighter latencies or closer voltages are connected to the same router in the same partition. This assignment reduces the hop count and also dissipated power. We have done the partitioning using an efficient partitioning tool called Chaco [10].

B. Merging Voltage Islands

The proposed method performs core partitioning according to $Wcomm$ weight values and without taking into account the core voltages at the first step. This approach may lead to having cores with different voltages in the same partition i.e., the island fragmentation. The fragmentation imposes a large number of voltage converters and a complicated power delivery network. To alleviate this problem, we propose an island merging algorithm to re-adjust the voltages of cores in every partition.

A simplified pseudo code of the island merging algorithm is shown in Fig. 3. For this purpose, cores in each partition are grouped in voltage islands according to their operating voltages (**Islanding()** function in Fig. 3). New edges are added to the graph and appropriate weights ($Wvolt$) compute the merging cost for them. The weight value is defined as $Wvolt_{ij} = (v_j - v_i) \times n_i$, in which v_i and v_j are the supply voltages of the islands. In order to minimize the merging cost, $Wvolt$ values are listed in ascending order. The top of the list denotes the minimum cost and corresponding islands are merged. During this process, cores with supply voltage vi are readjusted to operate on v_j ($v_i < v_j$). As a result, $Gvolt(V, F)$ is updated by removing the lower voltage island and changing the weight values (**UpdateIslandGraph()**).

The island merging (within each partition) continues until a single island emerges. The weight of the edges between the islands is compared with an input threshold (*Integration-threshold*). In case of a lower value the islands are separated into two partitions. Otherwise, all cores of the island are set to work under the same voltage. This process is applied to all the partitions, and thus, the supply voltages of all the cores inside a partition will be the same.

Lemma 1) The proposed method merges the islands with minimal cost.

Proof) Let us consider m islands $V = \{v_0, v_1, ..., v_m\}$ as well as the number of cores in each island to be defined as $n_0, n_1, ..., n_m$. We define the current abstract computation power as $cur = n_0 \times v_0 + ... + n_i \times v_i + .. + n_j \times v_j + .. + n_m \times v_m$. Assume that the voltage of an island v_i is lower than that of an island v_j and these islands are merged. The next abstract power after merging is defined as $next = n_0 \times v_0 + ... + n_i \times v_j + .. + n_j \times v_j + .. + n_m \times v_m$. The difference between the current abstract power and next one is $n_i \times (v_j - v_i)$, which is equal to $Wvolt_{ij}$ on $Gvolt(V, F)$ graph. Therefore, as the order of $Wvolt$ is ascending, the lower values are selected first. As a result, the difference between the current power and next power i.e., cost of merging, is minimized. ■

C. Hierarchical Voltage-Aware Floorplanning

The aim of the partitioning step is to group tightly connected cores (in terms of low delay and high bandwidth constraints) at the same partition to make them connected through the same router. This effectively reduces the communication costs; otherwise, if such cores are placed far away without provisioning the communication requirements, the link power consumption and latency may grow drastically [7], [9]. Although, we have merged the voltage islands within the same partition to solve the voltage fragmentation problem and to reduce the complexity of the power delivery network, islands with similar supply voltages may still remain far away among different partitions. This will potentially increase the complexity of the power delivery network. To tackle this issue, we introduce a hierarchical floorplanning method, which includes three steps; i) core-floorplanning, ii) partition floorplanning and iii) core placement update. The first step places the cores within each partition close to each other to meet their communication requirements. In this step the cores on each partition are floorplanned separately. In the second step, each partition is considered as a fixed block and the blocks are floorplanned to determine the position of the partitions on the chip. This step is done with the aim of placing the blocks with the same voltage and also communication requirements close to each other.

For our hierarchical floorplanning, we use the Parquet floorplanner tool [11] in a two-step approach. The method does *close-proximity-aware* placements of (1) cores within the same partition and (2) islands with the same voltage and high communication constraints. As a result we will have shorter links and a simpler power delivery network.

D. Routing and Path Allocation

This phase tries to establish the required physical links between each pair of routers. A set of links are used to establish the best routing path between the source and destination of a communication flow. We have used the algorithm proposed in [7], [12], [13] to find a deadlock free routing path for each communication data flow. The process starts with sorting the data flows based on their required bandwidth. The flow with the highest requirement is processed first. A fully connected graph is generated where the vertices represent routers. The graph is pre-processed to find the prohibited turns and to break any cyclic dependencies between flows and thus avoid deadlocks [13], [14]. The next step is to assign a weight (cost) to each edge of graph based on its corresponding physical link. Once the weights are assigned, choosing the shortest path is equivalent to finding a path with the least cost to route a data flow. This is done by applying Dijkstra's shortest path algorithm [15]. It should be noted that as the voltage converters between the islands induce overheads in terms of dissipated power and delay, the routing path allocation algorithm finds a path for each data flow such that the number of inter-island links are reduced.

V. Experimental Results

Our proposed toolchain utilizes Chaco [10] and Parquet floorplanner [11] tools to do the partitioning and floorplanning phases respectively. We also implemented a version of the Sunfloor synthesis flow [7], [12] that we call *Sunfloor Inspired* (SFI) flow and use it to perform the power/performance comparisons with our proposed design flow. The SFI toolchain needs a given floorplan and assigned voltages of cores to start its synthesis process. This toolchain groups the cores into different voltage islands according to their assigned voltages, and then cores within each voltage island are partitioned to do router allocation; finally, routers are connected to find routing paths for communication flows.

In the experiments, standard core graphs of D-36-4, D-35-bot, D-65-pipe, D-38-tvopd [12] are synthesized under 45nm technology size, and are compared in terms of power delay product. We used Orion2.0/Orion3.0 [16], [17] power models to estimate static/dynamic power consumption of routers and links respectively for our proposed method and our SFI implementation.

In our proposed synthesis toolchain, we treated the lowest allowed voltage as an input parameter i.e., we repeated the synthesis with the lowest allowed voltages of 0.8, 0.9, 1, 1.2 volts. To have a better insight, we implemented the proposed synthesis toolchain in two scenarios. In the first scenario the synthesis is done assuming a given initial core floorplanning, while in the second scenario there is no initial floorplanning given, i.e., the toolchain does a voltage-aware floorplanning.

A. Power, Delay and PDP Results

For each of the mentioned core-graph benchmarks, we let the tools find their synthesis solutions under the specified constraint of the core-graph. The experiments are repeated for different upper bounds on the number of on-chip routers from one to the number of cores in the core-graphs. In cases that tools found a synthesis solution, the power delay product is represented and shown in Fig. 4. As an example, in Fig. 4.a, for the maximum allowed on-chip routers of 10, the proposed toolchain *with* initial floorplanning provides the best synthesis solution in terms of power delay product. The SFI toolchain has not been able to find any synthesis solution that satisfies the application constraints. The key outcome highlighted in Fig. 4 is that in almost all the cases the proposed toolchain offers better designs against the SFI tool in terms of PDP. Our average PDP improvement over the SFI design flow is 59% and 63% for *with* and *without* initial floorplanning scenarios

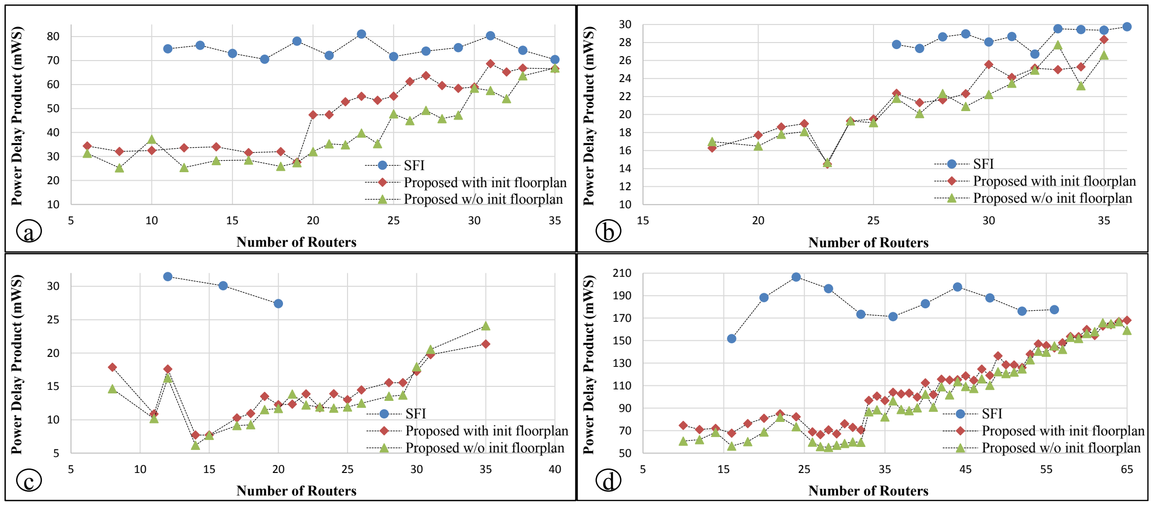

Fig. 4: PDP Comparison of four benchmarks (a) D-35-bot, (b) D-36-4, (c) D-38-tvopd, and (d) D-65-pipe.

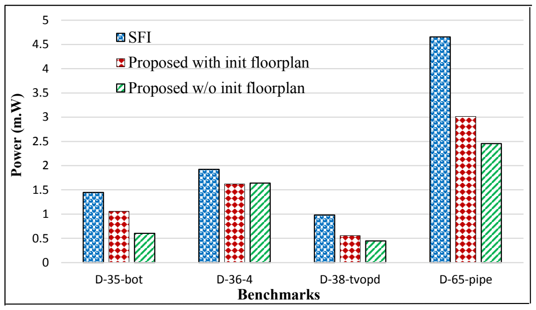

Fig. 5: Design solutions of the proposed and SFI tools based on best-power criteria.

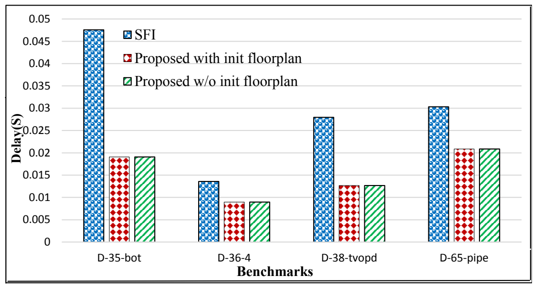

Fig. 6: Design solutions of the proposed and SFI tools based on best-delay criteria.

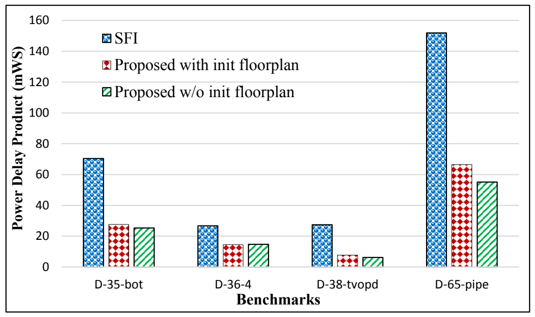

Fig. 7: Design solutions selected based on best-PDP criteria.

respectively. It is clear that when the maximum number of allowed on-chip routers changes, both of our proposed and SFI toolchains have to redo the synthesis flow from start. Due to this reason, solutions for the maximum router limit of *K* are independent of those for *K+1* and/or *K-1*.

The general pattern that can be seen in all four core graphs of Fig. 4 is that increasing the maximum allowed number of on-chip routers increases the PDP of solutions found by the proposed synthesis toolchain under both scenarios. The other important point is that, based on the found solutions, our proposed toolchain finds better synthesis solutions when it is not forced to obey an initial floorplanning. In fact, the average PDP improvement of 9% can be seen in all 109 different synthesis cases when we let the proposed toolchain do the floorplanning by itself. The best solutions found by our proposed toolchain and the SFI tool, in terms of power consumption and average network delay, are shown in figures Fig. 5 and Fig. 6. The proposed tool outperforms the SFI design flow in all benchmarks.

Another important observation is that the SFI tool is able to find synthesis solutions satisfying the application constraints for only 37% of the design candidates. Whereas, our proposed synthesis toolchain shows a great efficiency with 73%. In fact, our toolchain gives designers a greater flexibility by providing twice as much valid design alternatives. Using PDP as the metric, Fig. 7 shows the comparative evaluations of four core-graphs. Based on these results, to use of the SFI tool leads to at least 84% - and on average 155% - worse PDP than our proposed toolchain. The improvements seen with our design flow is even larger under the *without* initial floorplanning.

To further validate the PDP reductions seen in the proposed methodology, we count the number of used voltage converters in the output designs for the two toolchains. The results for the examined four benchmarks are presented in Figure Fig. 8. The proposed synthesis toolchain needs at least 40% fewer voltage converters. This positive outcome is due to the fact that our algorithm places communicating cores as close as possible. A higher number of voltage converters leads to more power consumption and delay.

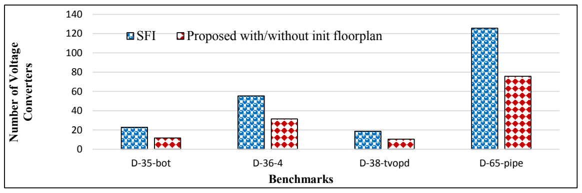

Fig. 8: Average number of required voltage converters.

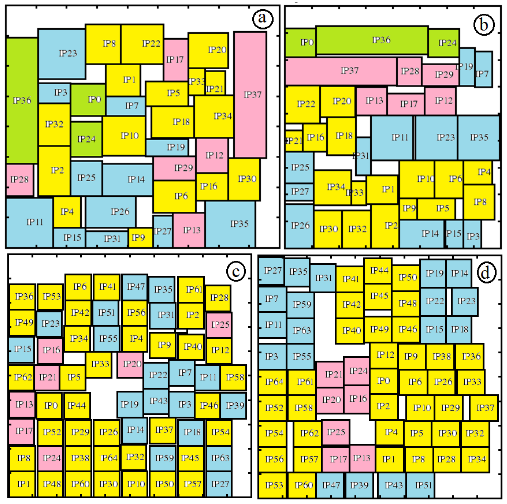

Fig. 9: Final floorplanning of the proposed synthesis toolchain. (a) and (c) benchmarks D-38-tvopd and D-65-pipe *with* initial floorplanning; (b) and (d) the same benchmarks *without* initial floorplanning.

B. Complexity of the Power Delivery Network

As we discussed earlier in the paper, the main objective of voltage-aware floorplanning is to reduce the cost of the power delivery network. In this regard, we compared the final floorplanning of our proposed toolchain under *with* and *without* initial floorplanning scenarios in Fig. 9. Comparing the generated floorplans for D-38-tvopd and D-65-pipe (Fig. 9), one can gather that the hierarchical floorplanning algorithm of the proposed toolchain gives a better core layout when no initial placement is given. Most of the cores with the same voltage are automatically placed in the same groups under the *without* initial floorplanning scheme. This means that synthesized *without* initial floorplans have fewer voltage converters and lower PDN costs. Although our main focus is power and performance, the output designs of the proposed toolchain also show on average 3.56% and 4.64% lower on-chip areas - for the four examined benchmarks - under the *with* and *without* initial floorplanning scenarios, respectively.

The time complexity of the proposed algorithm is $O(|C|^3 ln(|C|)|E|)$, where $|C|$ and $|E|$ are the number of vertices and edges in the core graph $Gcomm(C, E)$ respectively. These two values respectively show the number of on-chip cores and communication flows between cores. The number of routers is varied from 1 to $|C|$ and the tool tries to build a topology under each router count. Also, for each topology, $|C|^2 ln(|C|)$ corresponds to Dijkstra algorithm which is used to find the shortest path for flows. The time complexity of the proposed method is similar to time complexity of the SFI method [7].

VI. Conclusions

In order to benefit from the multi-voltage NoC design, core to router allocation, core to voltage island assignment, voltage-aware floorplanning, and routing paths should be done judiciously. In this paper, we propose heuristic methods to reduce communication power consumption and complexity of the power delivery network. By performing partitioning before islanding, the power consumption and latency are significantly reduced. In addition, with voltage island merging and voltage-aware floorplanning methods, voltage fragmentation and power delivery network complexity can be further minimized. The multi-voltage custom NoCs synthesized from our design flow show 63% *power and delay product* improvement (on average) over the ones synthesized with the SFI tool.

References

[1] G. Chen, F. Li, and S.W. Son, M. Kandemir, *Application Mapping for Chip Multiprocessors*, Design Automation Conference (DAC), 2008.

[2] S. Tosun Y. Ar and S. Ozdemir, *Application-specific topology generation algorithms for network-on-chip design*, IET Computers & Digital Techniques,Vol.6, Iss.5,pp.318-333, 2012.

[3] K.S.-M. Li, *CusNoC: Fast Full-Chip Custom NoC Generation*, IEEE Transactions on VLSI Systems, vol. 21, no. 4, pp. 692-705, 2013.

[4] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn, *Managing power and performance for System-on-Chip designs using Voltage Islands*, IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2002.

[5] N. Kapadia and S. Pasricha, *A System-Level Cosynthesis Framework for Power Delivery and On-Chip Data Networks in Application-Specific 3-D ICs*, IEEE Transactions on VLSI Systems, 2016.

[6] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli and H. Sarbazi-Azad, *A Method for Calculating Hard QoS Guarantees for Networks-on-Chip*, ICCAD, 2009.

[7] C. Seiculescu, S. Murali, L. Benini, and G.De. Micheli, *NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs*, Design Automation Conference (DAC), 2009.

[8] V. Todorov, D. Mueller-Gritschneder, H. Reinig, and U. Schlichtmann, *Deterministic Synthesis of Hybrid Application-Specific Network-on-Chip Topologies*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 33, No. 10, 2014.

[9] K. Wang, S. Dong, and F. Jiao, *TSF3D: MSV-driven Power Optimization for Application-Specific 3D Network-on-Chip*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 36, No. 7, 2017.

[10] B. Hendrickson, R. Leland, *The Chaco User's Guide: Version 2.0*, Sandia Tech Report SAND942692, 1994.

[11] S.N. Adya and I.L. Markov, *Fixed-outline Floorplanning: Enabling Hierarchical Design*, IEEE Transactions on VLSI Systems, Vol. 11, Iss. 6, 2003.

[12] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, *SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3-D Systems on Chips*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 29, No. 12, 2010.

[13] M. A. Kinsy, M. H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas, *Application-aware deadlock-free oblivious routing*, ACM International Symposium on Computer architecture, ISCA '09, pages 208–219, 2009.

[14] D. Starobinski, M. Karpovsky, and L.A. Zakrevski, *Application of network calculus to general topologies using turn-prohibition*, IEEE/ACM Transactions on Networking (TON), 2003.

[15] M. A. Kinsy, M. H. Cho, K. S. Shim, M. Lis, G. E. Suh, and S. Devadas, *Optimal and heuristic application-aware oblivious routing*, IEEE Transactions on Computers, 62(1):59–73, 2013.

[16] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, *Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration*, in Design, Automation Test in Europe Conference Exhibition (DATE), 2009, pp. 423 428.

[17] A.B. Kahng, B. Lin, and S. Nath, *ORION3.0: A Comprehensive NoC Router Estimation Tool*, IEEE Embedded Systems Letters, Vol. 7, No. 2, 2015.