Physical Design Aware NoC design for DNN Inference Accelerator

Ashwin Bhat
Georgia Institute of Technology
ashwinbhat@gatech.edu

Vima Gupta
Georgia Institute of Technology
vima.gupta@gatech.edu

ABSTRACT

Deep Neural Networks have been adopted widely in diverse domains. With the end of Dennard scaling, custom architectures to accelerate specific workloads are gaining traction. Accelerators for DNN inference have multiple processing engines which are usually connected in a mesh like NoC that is layout friendly. MAERI proposed a Augmented Reduction Tree topology which provides flexibility to the accelerator in terms of mapping. In this work, we evaluate the feasibility of the physical design of such a topology. The goal is to automate the placement and floorplanning of MAERI for different sizes in terms of number of PEs. We adopt H-trees used for clock networks to develop a floorplanning algorithm. Finally, based on the analysis of the generated floorplan, we propose modifications to the MAERI architecture to trade off performance for physical design friendliness.

ACM Reference Format:
Ashwin Bhat and Vima Gupta. 2021. Physical Design Aware NoC design for DNN Inference Accelerator. In *Proceedings of ECE 6115: Interconnection Networks (ICN '21).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Deep Neural Networks (DNNs) are being deployed at an increasing scale in widespread domains right from edge devices to the cloud [4]. As the performance benefit of scaling slows down with the end of Dennard scaling, where power density remains constant as the transistor size reduces, there has been a rise in custom design to accelerate specific workloads. There have been multitude of DNN accelerators proposed in literature.

A common design aspect across these DNN accelerators is the use of several Processing Engines (PEs) which operate in parallel and communicate with each other. This inter-PE communication is supported by a Network-on-Chip (NoC). Traditional choices for such network topologies involve PEs laid out in a systolic array connected by a mesh. Although a mesh is layout-friendly, it does not allow for flexibility in PE connectivity based on the architectural data-flow. This can result in underutilization of the hardware resources.

ICN '21, Georgia Tech,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

In order to support flexibility, MAERI [1] proposes adding switches to the MAC units and connecting them in a fat binary tree like structure. The tree is further appended with internal links at specific levels to support local forwarding. While this design allows for a high degree of programmability in the mapping, the floorplanning and placement of this irregular topology comprised of a distribution tree and an augmented reducing tree like structure is still a challenge. The challenge here is that it is not easy to visualize this topology in a two dimensional layout like it is in case of a mesh. A two dimensional visualization implies that it is easy to translate the logical view onto physical silicon.

In this work, we explore the floorplanning of the Augmented Reduction Tree (ART) topology used in MAERI. The goal is to design a framework that generates layout-friendly placement and routing methodology for the nodes of the ART topology. We propose a scalable framework which can generate the MAERI layout for variable number of PEs based on the requirement. Further analysis of the non-locality of these links leads to possible modifications of the MAERI architecture in order to compromise performance and make it more physical design friendly.

2 BACKGROUND

The design of an NoC begins with determining the topology. The topology is key in deciding the primary bounds on low load latency and throughput. There have been several network topologies proposed in literature for different applications. However, several of them are not practically viable since they are not layout friendly on a 2-D chip. The challenge is the presence of non-local links or lack of regular structure in such topologies. Irregular topologies rely on links of unequal lengths spanning the entire floorplan which can be costly to route in terms of timing and layer resources. Recently, there has been foundational work aimed at the physical design of topologies which have non-local connections. Specifically, the physical design of Ruche networks has been studied[2].

Ruche networks [3] build upon the 2-D mesh topology. Non-local links are added between routers that have a hop distance equal to the Ruche factor in the mesh. The main idea is to have dedicated tracks within each router for the Ruche links so that the layout can be done hierarchically. Thus, the idea here is to start with a regular topology which has an established physical design methodology and append it with the additional links.

3 MOTIVATION

Physical design on digital chips comprises of three primary stages, a) Synthesis, b)Place and Route and c) Physical Verification. The RTL level views are fed as input to synthesis which delivers the gate level simulation. This gate level view is purely a logical view at this stage. Once the timing constraints and pin-level restrictions

are fixed, the placer tool spreads the gates and memories to ensure the setup frequency target is met while honoring area and power limitations. Once the digital gates and memories have been placed, the clock tree is synthesized. Clock tree synthesis is carried out by EDA tools that ensure the clock signal is delivered to the input clock pin of every flip-flop with minimal skew and clock jitter. Certain custom structures are used for high performance designs. The Fig. **??** shows H-tree and X-tree methodologies that use a fractal based approach to cover the entire floorplan area.

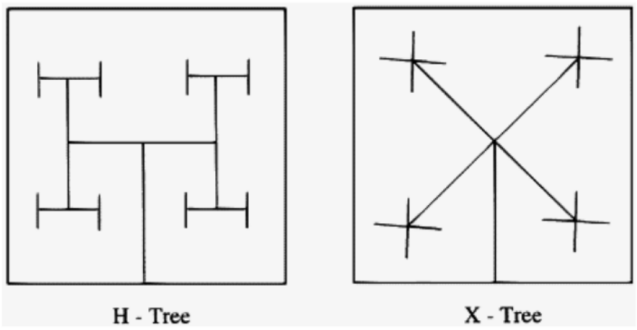

Figure 1: Custom clock tree structures for high performance designs scalable due to their fractal like structure.

An H-tree is a symmetric tree in which wire length from any sink to the root is the same. Fig. 2 is an illustration of an H-tree topology. This figure shows a topology in which the clock signal is driven from a central location to multiple clock sinks. Since the clock pin may not be located in the centre of the chip it is necessary to route the clock from the clock pin to the centre of the H-tree.

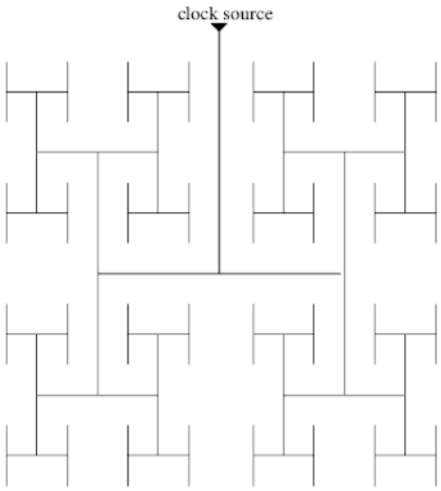

Figure 2: H-tree root and tap-points

An H-tree necessarily presents a homogeneous sink distribution in the X and Y axis. H-tree can be used to drive the clock signal directly to the flip-flops or to the inputs of a mesh. Although wire lengths are equalized by the H-tree structure, buffers must be carefully inserted and sized in order to keep skew small. Wire widths can also be changed either to compensate different loads driven by each branch or to satisfy electron-migration rules. In both cases the larger the load driven is larger the wire width should be.

The binary structure of an H-tree serves as the inspiration for designing the fat tree topology for MAERI architecture. The primary difference is that, in an H-tree the same clock signal is propagated from the root to all its tap-points, whereas for our fat tree structure, the links carry unique signals from one node to another. The nodes are placed at all those points in the H-tree where we switch between the X and Y dimension. Thus, similar to Ruche networks, we have a base regular design to which we add the additional non-local links. The base design in this case is the binary H-tree and the placement of the nodes is done so that forwarding links are as short as possible. The algorithm to do the same is described later in the report.

4 PRELIMINARY DESIGN

MAERI contains a distribution network comprised of virtual neurons (1 : 2 switches) which are arranged in a simple binary tree that computes weights/input activations from a prefetch buffer. This feeds the weights to a reduction network that computes output activation using the augmented reduction tree. Adopting the idea from the physical design of Ruche networks, we can consider the baseline regular topology as the binary tree. Augmented reduction tree can be considered as the baseline with additional links appended between certain nodes.

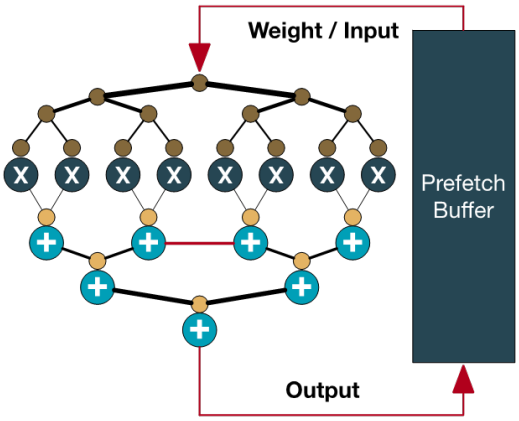

Figure 3: MAERI architecture outlines in terms of reconfigurable virtual neurons with adders and multipliers

In this report, we propose unravelling the augmented tree structure in the shape of classic H-tree. In the case of a clock tree, the same clock signal is delivered to all the leaf nodes of the H-tree. In the case of the distribution network, we will have routers at the points of the H-tree where it switches between the X and Y dimension. The routers will comprise of the MAERI building blocks such as the adder switch, multiplier switch, multiplexer, and look-up-tables. This serves as the baseline design for the distribution network.

For the augmented reduction tree, forwarding links (for adder switches) are added between nodes that are adjacent to each other

but have different parent nodes. In our preliminary design, we ensure that the node placement along the H-tree is such that these adjacent nodes remain adjacent to each other and are not diagonally opposite for an optimal route length. We observe that it is possible to keep all such nodes adjacent to each other i.e. the arrangement does not lead to contradictory placement location at any level of the tree. In addition to these, there are forwarding links (for multiplier switches) which connect all the nodes in a contiguous manner at the last level of the tree. For these connections, we can simply consider the leaf nodes of the tree to connect in a mesh like structure. This is because the H-tree like layout would cause nodes at a particular depth of the tree to overlap in the placement position with a 2-D mesh of a similar size. The complete mesh is not required and we only retain the links that are necessary.

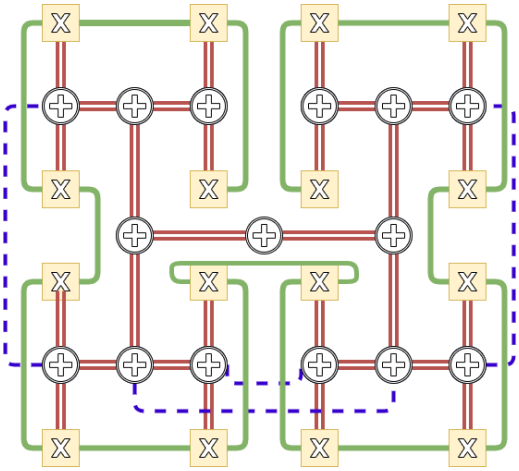

Figure 4: Physical design floorplan of Augmented Reduction Tree for MAERI architecture

5 FRAMEWORK

We provide a scalable framework that generates the floorplan and the corresponding link connections based on the number of nodes in our tree. In addition to creating the skeletal H-tree layout friendly map, we also provide functions that generate a string of nodes that are to be connected to enable forwarding. Electronic Design Automation (EDA) tools such as Innovus (Cadence suite) and ICC Compiler 2 (Synopsys suite) are capable of creating custom routes when provided with the endpoint regular expressions.

5.1 Framework modelling

- The routers are labelled in a level order binary tree traversal.
- The leaf node layer is comprised of multipliers (adders) for the augmented reduction tree (distribution tree) with forwarding links.
- Total number of nodes (which is necessarily one less than an exponent of two) in the reduced tree is given as an input to the framework.
- The model generates the co-ordinate pair for every router and multiplier(adder) that can be used for placement on a 2-D floor-plan.
- The aforementioned points are first connected through an underlying H-tree (indicated by green links in Fig. 4.
- The model also outputs string of points which constitute of additional forwarding links (indicated in blue in Fig. 4) placed as close to each other as possible.

5.2 Walk-through for an example

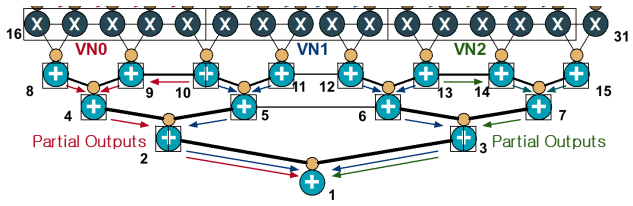

Figure 5: Logical view of the Augmented Reduction Tree with routers numbered

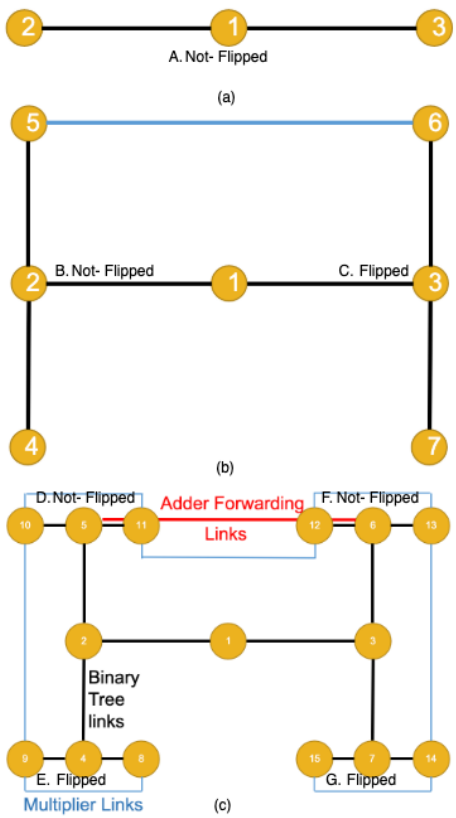

Figure 6: Physical view of the Augmented Reduction tree with routers numbered

In this walk-through, we will describe the steps to unfurl the logical view (Fig. 5) of the augmented reduction tree into a physical view (Fig. 6). We consider 15 nodes here. We conventionally place child nodes in left→right or bottom→ top manner in increasing

order of their numbering. This approach is referred to as 'Not Flipped'. For certain parent nodes, the child nodes are placed in right→left or top→bottom manner which is henceforth called the 'Flipped' convention.

(1) We place the router 1 in the middle of the floor-plan as the source of our H-tree.
(2) The child nodes, 2 and 3 in the augmented reduction tree are placed in a usual 'Not Flipped' manner as indicated in Fig. 6(a).
(3) For the parent node 2, we layout the child nodes 4 and 5 in a 'Not Flipped' manner as referred in Fig. 6(b).
(4) For the parent node 3, we layout the child nodes 6 and 7 in a 'Flipped' manner, instead of 'Not Flipped' manner. This is done so that the forwarding link between the nodes 5 and 6 is local. If we do not flip, the link would be diagonal.
(5) In Fig. 6(c), the children for parent nodes 4 and 7 are 'Flipped' whereas those for nodes 5 and 6 are 'Not Flipped'.

5.3 Algorithm to determine the node with flipped child nodes

The walk through in the previous section gives an overview of the process to be followed for generating the floorplan. The key decision is whether to flip or not to flip when placing the child nodes. This decision determines whether the forwarding links are local or non-local. We abstract out the pattern from the walk through example into the floorplanning algorithm in the following manner.

(1) Input: The total number of nodes in the tree which can be expressed as $2^N - 1$ since it is a binary tree.
(2) The nodes are numbered from 1 to $2^N - 1$. This ensures that for a node k, its child nodes will be 2k and 2k+1.
(3) In the logical view, the left child is numbered 2k while the right child is numbered 2k+1. In the physical view, the 2k child can either be on the left(bottom) or the right(top) if the link from the parent node k is in the horizontal (vertical) direction.
(4) Each node will have a binary variable to indicate whether it is in the flipped state or not
    - T : the 2k child is on the left (bottom)
    - F : the 2k child is on the right (top)
(5) The root node 1 is considered to be at depth 0. For a node at even depth level, the 2k child retains its flipping state while the 2k+1 child has the opposite state (Fig. 8).

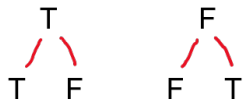

Figure 7: Even depth level

For a node at odd depth level, the 2k child has the opposite flipping state while the 2k+1 child retains the flipping state of the parent node k (Fig. **??**).

(6) Based on the above rule, the pattern for the entire tree is shown in Fig. 9.

Figure 8: Odd depth level

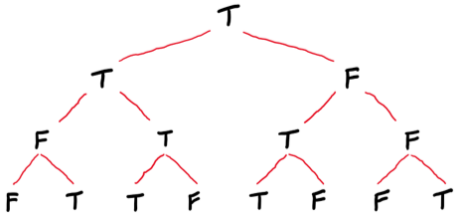

Figure 9: Pattern for entire tree

6 SCALABLE FLOORPLANS

The floorplans generated using the above algorithm are shown below for different sizes of MAERI. The black links are the base structure of the binary tree. The red links are the forwarding links between the adder switches. The blue links represent the forwarding links between the multiplier switches.

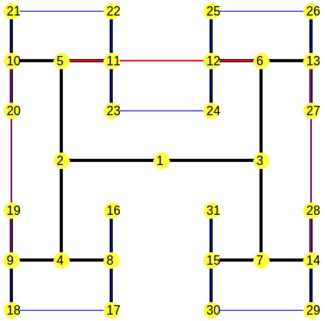

Figure 10: Floorplan for MAERI with 32 processing elements

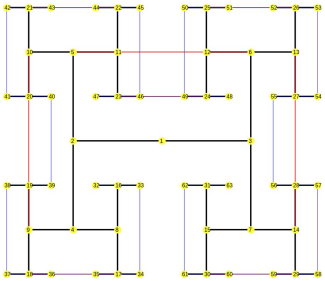

Figure 11: Floorplan for MAERI with 64 processing elements

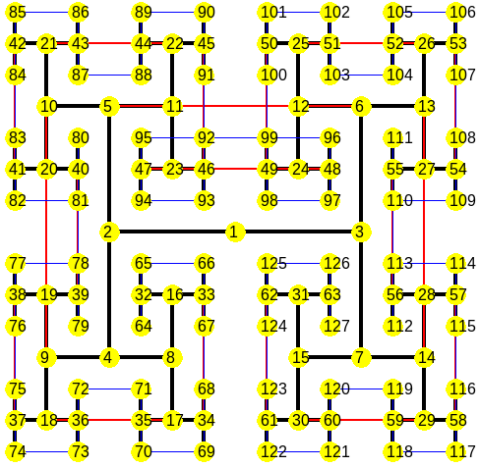

Figure 12: Floorplan for MAERI with 127 processing elements

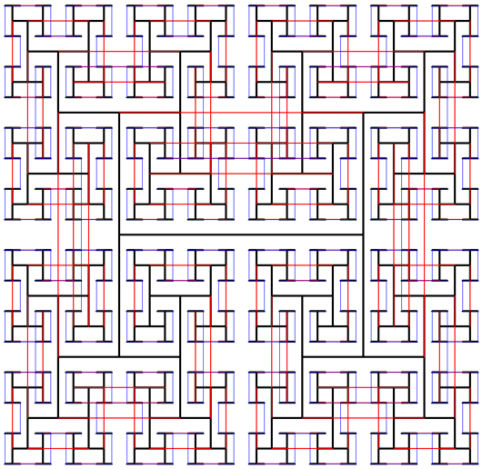

Figure 13: Floorplan for MAERI with 1023 processing elements

7 ANALYSIS

The walk through was provided for a small example comprising of 15 nodes. The floorplans generated from the algorithm reveal that this methodology is able to ensure that the node placement is such that no forwarding link is non-local upto 31 PEs. When we hit 63 PEs, the forwarding link between nodes 47 and 48 (Fig. 14) turns out to be non-local.

The notion of non-locality in a hierarchical floorplan can be understood in the following manner. First, we observe that each depth level of the tree forms a mesh like structure with a certain pitch in the x and y dimension. If the number of nodes at a particular depth is a perfect square (even depth levels of the tree), the pitch is equal in the x and y dimension. Non-local links are links which are not minimum pitch in length. In Fig. 14, the link between nodes 47 and 48 is 3 times the minimum pitch and hence is non local.

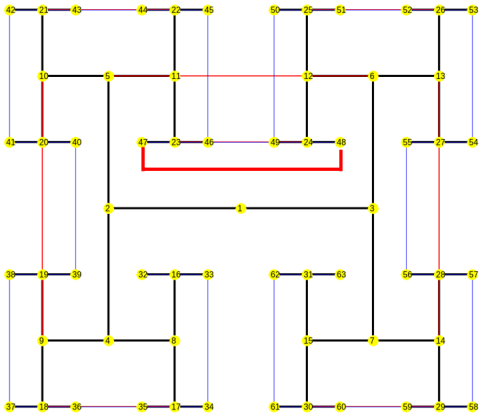

Figure 14: Floorplan for MAERI with 63 processing elements showing the non-local forwarding link

The above idea is extended to extract the number of links that are non-local as the design scales. The forwarding links can be considered to be the one connecting the adder switches while the multiplier switches are connected in a contiguous manner. The variation of the number of these links with increasing PEs is shown in the figure below.

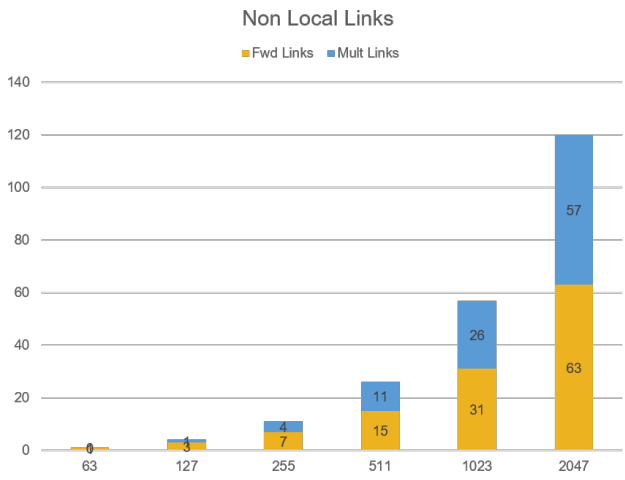

Figure 15: Number of non local links as a function of PEs

It is clear that the number of non-local forwarding links increases with increase in PE count. The hierarchical or fractal like nature of the H-tree design creates interesting artifacts. As we increase the size (say going from 511 to 1023 PEs), the total number of non-local links becomes the total number of non-local adder forwarding links at the next size. This is because some of the multiplier links for a particular size become adder forwarding links at the next size. However, the non-local links are never between nodes of consecutive number and hence non-local links will monotonically increase. Another important factor to remember is that not all non-local links are of same size. In the PE sizes shown, the non-local

links vary from 3 to 5 times of the minimum pitch at that tree depth level.

8 SUGGESTIONS

8.1 Architecture Modification

In MAERI, the basic binary tree is sufficient to provide the flexibility required for mapping different dataflows by combining multiplier switches into virtual neurons. The forwarding links are added in order to imporve performance and to make the topology non-blocking in nature. Thus, these forwarding links are not necessary for correctness. Since only a small fraction of these forwarding links are non-local, one possibility is to drop these links altogether. This would have a performance penalty but make the architecture more physical design friendly.

8.2 Folded Distribution and Reduction Trees

The pre-fetch buffer stores the inputs and weights that are distributed amongst the virtual neurons and then reduced via the augmented reduction tree. An H-tree based approach places the first router in the centre of the floorplan. Transferring all the weights and input to the centre can be inefficient in terms of routing resources. In order to mitigate this communication challenge, we further propose a folded H-tree structure where the existing implementation is folded along the Y-axis along an axis passing through the first route. This allows for placement of the pre-fetch buffer adjacent to the input router allowing for minimal wire-length for the transferring bus.

8.3 Hierarchical Topology

The H-tree design methodology implies that the floorplans are fractal in nature. This means that when we increase the size of the design, the new design build upon the design of the lower size. The key difference is that the multiplier links will become adder forwarding links. Thus, only some of the multiplier links will have to be retained when we increase the size. The multiplier links between nodes having the same parent node will have to be removed when we increase the number of PEs. However, the larger designs still have blocks which look like smaller designs. For example, Fig. 12 with 127 PEs looks like it has four sub-blocks that are connected with some links that go across them. The sub-block has 31 nodes and looks like 10 which is a 31 PE instance of MAERI. Similarly, the MAERI instance with 1023 PEs (Fig. 13) looks like it has 16 sub-blocks of MAERI with 63 PEs (Fig. 11) or 4 sub-blocks of MAERI with 255 PEs.

Thus, we can modify the topology for larger sizes and break it into two hierarchical pieces. The sub-block comprises of either the 31 or 63 PE instance of MAERI depending on the overall size. These sub-blocks can then be connected to each other in a mesh at the top level or with a bus depending on the number of sub-blocks. This way, the total number of non-local links will be reduced. However, the performance might be affected since the top level links are shared by the nodes that had the non-local forwarding links connecting them.

9 CONCLUSION

In this report, we analyze the physical design friendliness of a flexible DNN Inference accelerator called MAERI. It comprises of a binary tree with several forwarding links that make it difficult to visualize a two dimensional scalable physical view. The clock based H-tree design is adopted for the binary tree NoC topology of MAERI. An algorithm is developed for the placement, floorplanning and global routing of the nodes in MAERI such that maximal number of forwarding links are kept local. The algorithm is used to generate the floorplans for different size instantiations of MAERI. Analysis of these floorplans revealed important aspects about the behavior of number of non-local links with scaling the number of PEs. Finally, certain modifications are suggested to the core MAERI architecture in order to trade off performance for higher ease in its physical design.

REFERENCES

[1] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. 2018. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects. *SIGPLAN Not.* 53, 2 (March 2018), 461–475. https://doi.org/10.1145/3296957.3173176

[2] Y. Ou, S. Agwa, and C. Batten. 2020. Implementing Low-Diameter On-Chip Networks for Manycore Processors Using a Tiled Physical Design Methodology. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 1–8. https://doi.org/10.1109/NOCS50636.2020.9241710

[3] D. Petrisko, C. Zhao, S. Davidson, P. Gao, D. Richmond, and M. B. Taylor. 2020. NoC Symbiosis (Special Session Paper). In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 1–8. https://doi.org/10.1109/NOCS50636.2020.9241584

[4] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna. 2020. SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 58–70. https://doi.org/10.1109/HPCA47549.2020.00015