AN SRAM COMPILER FOR MONOLITHIC 3D INTEGRATED CIRCUIT

A Dissertation
Presented to
The Academic Faculty

By

Daehyun Kim

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

Dec 2022

AN SRAM COMPILER FOR MONOLITHIC 3D INTEGRATED CIRCUIT

Date approved: November 18, 2022

ACKNOWLEDGMENTS

I would like to thank my parents. I would not where I am without their support. I also want to thank Dr. Saibal Mukhopadhyay who has been a great advisor and mentor. I would like to thank Dr. Edward Lee for his help on this work. Last but not least, I would like to thank all the members of our lab for helping me with my work and life.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This thesis presents a monolithic-3D (M3D) SRAM arrays using multiple tiers of Carbon Nanotube (CNT) transistors. The compiler automatically generates single-tier 2D SRAM subarrays and multi-tier 3D SRAM subarrays with different tiers for cells and peripheral logic. Moreover, the compiler can integrate multiple subarrays of different dimensions to generate larger capacity SRAM arrays. The compiler is demonstrated in a commercial-grade M3D process design kit (PDK) with 2 tiers of carbon nanotube transistors (CNFETs). Simulations show that the M3D CNT SRAM design can improve the properties of memory compared to the 2D CNT SRAM design. In a 32KB memory implementation, the M3D design can reduce footprint, latency, and energy by 33%, 10% and 19% respectively. The compiler is used to show the feasibility of fine-grain logic and SRAM stacking in M3D technology.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Monolithic 3D (M3D) integrated circuits (ICs) using fine-grain nano-scale inter layer vias (ILVs) promises significant energy-efficiency improvements over 2D ICs [1]. However, M3D requires sequential fabrication of multiple layers of transistors in one substrate, where the circuit components in different tiers are interconnected via high-density 3D vias. The need for high-temperature processing of silicon-based MOSFET (1000°C) during sequential fabrication of one tier can degrade the reliability and performance of devices in previously fabricated tiers [1, 2, 3, 4, 5]. The recent works on silicon-based M3D processes are exploring techniques that address the high-temperature processing challenges [6, 7].

Carbon nanotube FET (CNFET) based M3D process has emerged as an attractive alternative to the silicon-based M3D process [8, 9, 2, 10, 11, 4]. This is because CNFET can be fabricated at the temperature below 425°C which eliminates potential defects of devices and interconnections on previously fabricated tiers [2]. Hence, in a CNT based M3D IC, on/off currents of CNFET transistors in different tiers are close to each other. CNFET also promises high energy-efficiency in designing logic and memory circuits [8, 9]. Shulaker et. al. have demonstrated applications of CNFET based M3D processes [12, 13, 14]. Srimani et. al. have demonstrated commercial-grade M3D process design kits (PDK) and the operation of logic and SRAM [2].

This thesis present an SRAM compiler for CNFET-M3D using the PDK developed by Srimani et. al. [2]. Although there are few prior works on SRAM cell design in M3D silicon [15] or CNFET [2], there has been no SRAM compiler for CNFET based M3D. Figure 1.1 shows a schematic of the M3D stack that includes two CNFETs layers and six metal layers. Proposed compiler leverages similar CNFET performance in different tiers in two ways. The compiler first exploit this observation to generate a large SRAM array

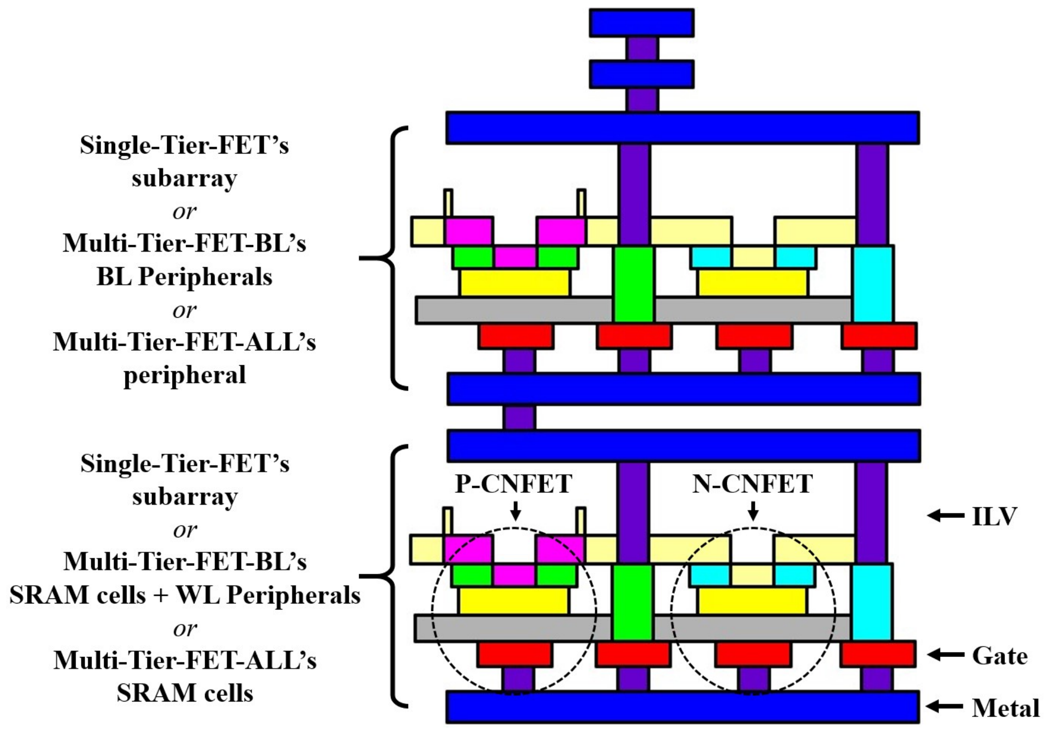


Figure 1.1: Schematic layer organization of M3D PDK used in the compiler (re-drawn after [2]). The 2 tiers of CNFETs are included in this process.

by combining *stackable single-tier* subarrays designed in individual tiers. These *stackable single-tier* subarrays are referred to as the **STF** i.e. **S**ingle **t**ier of **F**ETs contain bit-cells and peripherals, all in a single tier. The *stackable single tier* design is achieved by separating metal layers used in each SRAM tiers as shown in Figure 1.1. As subarrays in different tiers have similar performance, the compiler can efficiently integrate them to generate a large array.

Second, this thesis presents 3D SRAM subarrays composed of multiple tiers of the transistor where bit-cells in one tier and peripheral circuits in other tiers are connected using fine-grain ILVs. As transistors in different tiers have similar performance, the folding of the peripheral circuits into multiple tiers allows reducing footprint while maintaining (or improving) performance. The compiler presents two different types of multi-tier SRAM subarrays. An **MTF-BL** subarray contains **M**ultiple **t**iers of **F**ETs with bit-line (**BL**) peripherals in a second layer of transistors. An **MTF-ALL** subarray contains **M**ultiple **t**iers of **F**ETs with **all** peripherals in word-line (WL) and BL peripherals in one tier of transistors and bit-cells in a different tier.

The memory compiler flow generates the layout files (library exchange format (LEF) and graphical data system (GDS)) and the timing file (liberty timing file (LIB)) for SRAM memory with different types of subarrays with varying dimensions. The compiler supports the scaling capacity of the generated SRAM by integrating multiple subarrays. First, the compiler can generate the physical design of an **SRAM block** by integrating smaller subarrays using an H-tree architecture. Second, the further scaling of memory capacity is achieved by connecting multiple **SRAM blocks** using Network-on-Chip (NoC) to generate an **SRAM array**. The compiler demonstrates the application of CNFET M3D SRAM compiler for generating SRAM subarrays, blocks, and arrays of varying capacities. The thesis shows the compiler to generate physical design of a system architecture with a multi-core processor in one tier integrated with an SRAM array in the other tier. Each core locally connects to a smaller capacity SRAM block but all SRAM blocks are connected via an NoC to create a large capacity array but with multiple distributed access ports.

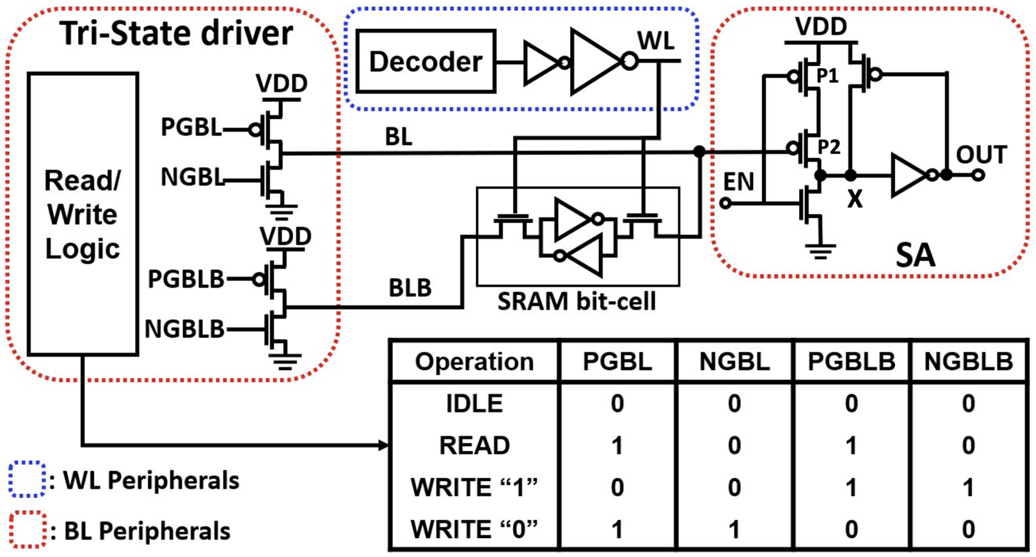
The compiler shows that M3D subarray designs can improve the properties of the memory compared to 2D subarray designs. The combination of WL and BL, used for the comparison, are 64WLx64BL, 64WLx128BL, 128WLx64BL and 128WLx128BL. The footprint, read energy, write energy and read latency can be reduced 27.8%-39.8%, 1.7%-2.8%, 4.7%-8% and 9.7%-11.3%, respectively. In addition, the properties of SRAM block also can be improved by using MTF designs instead of STF design. 32KB of SRAM block can achieve 24.8% lower footprint and 9.5% lower energy consumption.

CHAPTER 2
M3D SRAM ARCHITECTURES AND COMPILER

2.1 Subarray Architectures

The SRAM subarrays include bit-cells, address decoders and drivers, sense-amplifiers (SA), and write-driver circuits (Fig. 2). The compiler uses the single-ended dynamic SA (Fig. 2). Before WL is raised high, the signal EN is high and pre-discharges the node X, thereby pre-charging OUT to high. During reading, the signal EN is made ‘low’ which turns the PMOS (P1) ON. If the SRAM bit-cell is storing a ‘0’, the BL discharges, thereby turning on the PMOS P2 which charges the node X, and discharges OUT to low. If the SRAM cell is storing an ‘1’, the BL remains high, which ensures the node X and OUT remains low, and high, respectively. All of the logic is custom-designed and automatically placed by the SKILL code. The compiler assumes all BLs are read/written in parallel i.e. no BL interleaving or column multiplexing inside the subarray. All the columns are read out in parallel and multiplexed outside of the subarray to create the designed data width (32-bit).

Figure 2.2 shows the STF, MTF-BL, and MTF-ALL subarray micro-architectures. All of these micro-architectures follow the same schematic design (Figure 2.1), but different placement. For both STF and MTF designs, the bit-cell arrays remain in a 2D arrangement. The key difference resides in the placement and arrangement of peripheral circuitry. The physical implementation of bit-cells and all peripherals limit layer usage to one of the two tiers (layers) of CNFETs and the two immediate layers of metal (one above and below the FET, each) (Fig. Figure 1.1) to prevent metal usage overlap between top tier and bottom tier designs. This allows M3D stacking of bit-cells and peripherals without need for re-implementing physical layouts.



Operation	PGBL	NGBL	PGBLB	NGBLB
IDLE	0	0	0	0
READ	1	0	1	0
WRITE “1”	0	0	1	1
WRITE “0”	1	1	0	0

Figure 2.1: Schematic of SRAM subarray

In the STF subarray, peripheral circuits are placed in the same tier as bit cells following conventional 2D subarray arrangements. The compiler supports the design of STF subarrays in both tiers of transistors provided by the PDK. From post-PEX simulations, Tier-2 STF subarrays demonstrate only 4% and 6% reduction in read energy and latency, respectively, compared to Tier-1 STF designs. The detailed analysis shows that the difference is mainly contributed by different parasitics involved when changing tiers. However, for comparison with MTF designs, Tier-1 STF designs are used as the baseline due to the fact that Tier-1 STF and MTF designs all use the same tier (Tier-1) for implementing bit-cells.

Figure 2.2 (b) shows the MTF-BL subarray architecture. The WL drivers are placed in Tier-1 along with bit cells, but WL address decoders and BL peripherals are in Tier-2. As BL peripherals and WL drivers reside in different tiers, additional functionality is added to scale/distribute the BL driver output stage as subarray dimensions change. This allows trimming the BL driver for small subarrays to reduce footprint/performance overhead. Figure 2.2 (c) shows MTF-ALL architecture where all peripherals are in the Tier-2. The arrangement is such that, with peripherals and bit cells rotated/flipped, it can ensure that output edges of peripherals are aligned along the center axes of the bottom bit-cell arrays. This structure prevents BL peripherals from blocking WL access and vice versa by

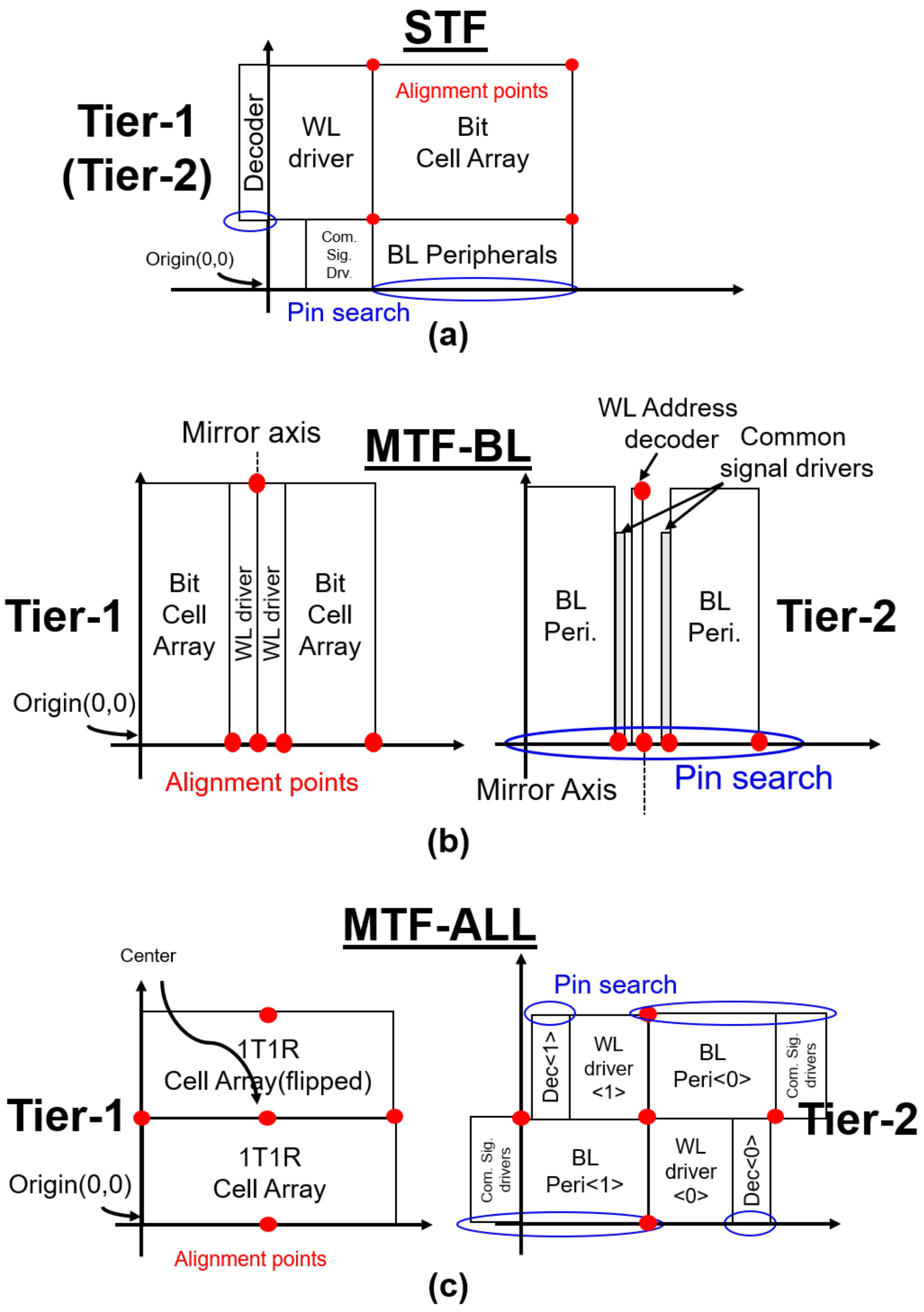


Figure 2.2: Subarray peripheral/bit-cell arrangement for (a) 2D STF, (b) MTF-BL, and (c) MTF-ALL subarrays.

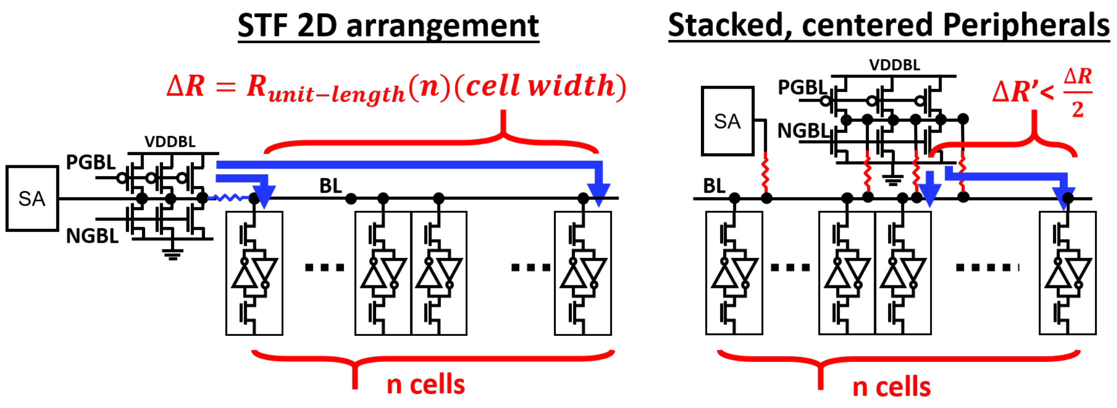

STF 2D arrangement
$\Delta R = R_{unit-length}(n)(cell\ width)$
VDDBL
PGBL
SA
NGBL
BL
n cells
Stacked, centered Peripherals
VDDBL
PGBL
NGBL
SA
BL
$\Delta R' < \frac{\Delta R}{2}$
n cells

Figure 2.3: Centered placement of peripherals reduces worst-case mismatch.

segmenting all peripheral instantiating to twice. However, the central location of peripherals also separates the access pins to different edges, which needs to be considered during automated generation for multiple inter-connected subarrays.

A unique property of the CNFET M3D process is the use of back-gated CNFET (Figure 1.1). A key advantage of back-gate FET geometries is the reduction in gate-to-plug capacitance [16], which reduces overall WL capacitance and BL capacitance in M3D SRAM. The compiler utilizes the back-gate structure by placing bit-cells in the bottom tier and BL peripherals in the top tier where BLs occupy the metal layers between tiers. This arrangement allows multiple ILVs to be constructed for resistance reduction and minimizes parasitic capacitance applied from surrounding devices/metals to WLs that occupy the bottom-most metal layer.

MTF subarrays created by stacking peripherals directly on top of the bit-cell array allow several advantages. The first advantage is reduced mismatch seen by peripherals (Figure 2.3). MTF designs allow flexible placement of peripherals compared to traditional 2D structures. By centering the stacked peripherals, up to 2$\times$ reduction in worst-case mismatch can be achieved. As worst-case conditions have been cut short, improvement in read/write performance can also be observed. The second advantage is a benefit enabled by the ability to manufacture dense ILVs in M3D technologies. Instead of utilizing ILVs

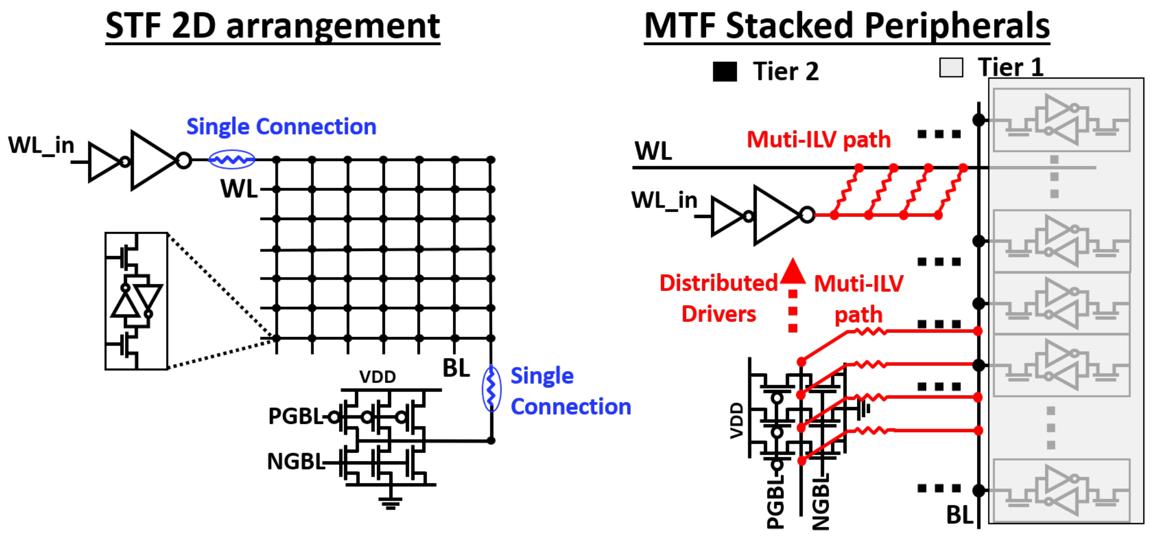


Figure 2.4: Multiple ILV path for writing with MTF stacked peripherals.

for digital signals, MTF subarray structures use multi-ILV paths to create low-resistance connections between WLs/BLs and their corresponding drivers (Figure 2.4). Reduced resistance not only allows faster BL and WL switching, but also reduces I-R drop, which can affect write stability for far-end bit-cells.

2.2 SRAM Block and Array Architecture

The compiler generates a large capacity SRAM array by integrating multiple smaller capacity subarrays. First, the compiler uses an H-tree architecture to integrate the smaller subarrays to compile an **SRAM block** (Figure 2.5). Next, the compiler combines multiple memory blocks using an NoC to design an **SRAM array** (Figure 2.6).

2.2.1 Architecture of the SRAM Block

An H-tree is a hierarchical design where each node of the tree accumulates data from lower level nodes. The compiler refers the logic necessary to combine the subarrays within the H-tree as the **top module**. The compiler can integrate any sizes and types of subarrays as the leaf nodes of an H-tree. The compiler places the H-tree router in the empty space

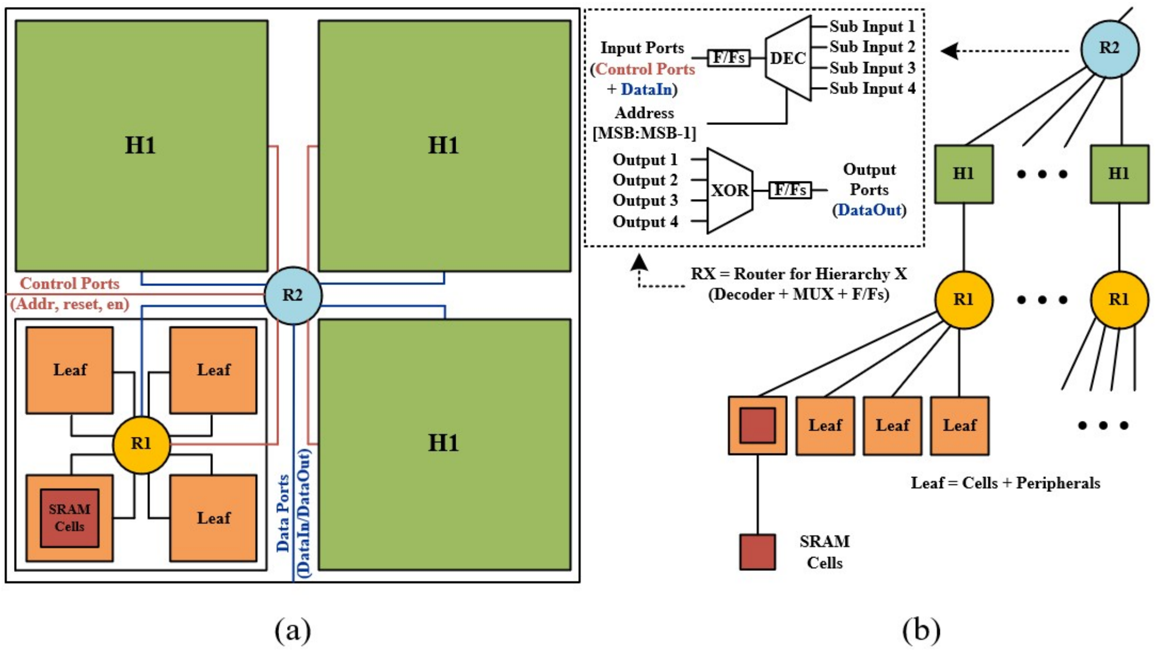


Figure 2.5: SRAM block design: (a) high-level placement and (b) H-tree architecture.

among the submodules. During reading, the H-tree router multiplexes multiple incoming data ports from a lower level in the hierarchy to a single output port to the higher level in the hierarchy. During writing the H-tree router de-multiplexes the incoming data port from the higher level in the hierarchy to one of the output ports to the lower level in the hierarchy.

2.2.2 Architecture of the SRAM Array

An SRAM array is implemented by connecting multiple SRAM blocks using a memory NoC (Figure 2.6). The current compiler generates a mesh NoC where each SRAM block is connected to a router. The compiler uses the open-source NoC router which has a virtual channel (VC) for the deadlock-free algorithm [17]. The router controls the read/write access to individual SRAM blocks and manages the data movement within the memory NoC.

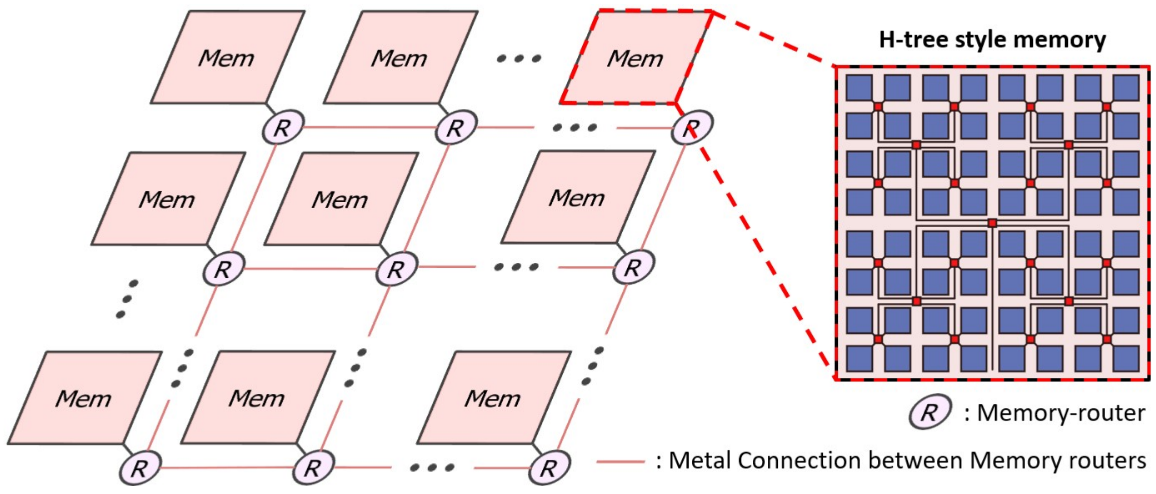


Figure 2.6: Multi-port memory network architecture.

2.2.3 Dataflow of SRAM Access

The compiler creates a distributed multi-port memory where each router acts as an I/O port. Individual logic blocks can be connected to one router and access the entire memory. When there is a memory access request from the logic blocks, the memory network automatically calculates the index of the target SRAM block from the input address. Depending on the physical distance between the target SRAM block and the requesting logic block, the access request can traverse zero, one, or multiple hops in the network.

2.3 Evaluation and Design Space Exploration

The latency, area, and energy of the sub-arrays are evaluated using SPICE simulations of the extracted netlists. The SRAM block including the H-tree network and the routers of the memory NoC in the entire array is synthesized to meet the target memory frequency while ensuring single cycle communication between successive levels in H-tree and nearest routers in the NoC.

To evaluate the energy consumption of the array, the compiler synthesizes and perform PNR of each level of the hierarchy independently. To calculate the dynamic energy of an SRAM block, the compiler recognizes that only a single subarray and only one node in each level of the H-tree is active every cycle. The static energy for the entire SRAM block is computed. The energy for accessing the entire array includes the energy of one SRAM block and the number of active routers (i.e. hops in NoC). As the number of hops for an access can vary and only available from a detailed architectural simulation, this paper considers all the routers are active (the worst-case scenario).

The compiler allows exploring various dimensions of subarrays, capacity of SRAM blocks (i.e. number of levels in the H-tree hierarchy), and the number of SRAM blocks (i.e. number of nodes in the mesh NoC) for a given memory capacity to meet a design goal such as minimum footprint, or minimum energy. The subarray options include the type of the subarray (STF, MTF-BL, and MTF-ALL), and different dimensions of a given subarray type, all of which determine the access latency/energy of individual subarrays. Generating a memory array with smaller capacity SRAM blocks will lead to more hops while accessing a distant address. However, as a smaller capacity of SRAM block have lower read/write latency/energy, the cost of local accesses will be reduced.

CHAPTER 3
SIMULATION RESULTS

3.1 Run time of Compiler

The run time of the compiler is measured on a desktop with i7-9700 core and 16GB memory. The subarray layout generation takes less than one minute. The analysis takes 30 and 120 minutes for a 64WL$\times$64BL and a 128WL$\times$128BL subarray, respectively. The run-time to compile an SRAM block depends on the number of levels in the H-tree, where each level requires 20 to 30 minutes. The generation of a 2MB SRAM array with 4x4 SRAM blocks (128KB) and mesh NoC requires 120 minutes.

3.2 Subarray Compilation Results

Figure 3.1 visualizes the layout of different subarray types.

3.2.1 Footprint Analysis

Figure 3.2 (a) shows the 2D footprint area for different subarray dimensions normalized to 64WL$\times$64BL STF subarray footprint. MTF designs show a lower footprint than the STF design. However, the benefit reduces for larger subarrays where peripheral circuits have relatively lower contributions. MTF-ALL structures suffer from peripheral overhead at small subarray dimensions. This is because peripherals placed in Tier-2 extend beyond the bit-cell array boundaries in Tier-1. This overhead is addressed by scaled drivers in MTF-BL, which provide the lowest footprint for 64WL$\times$64BL subarray.

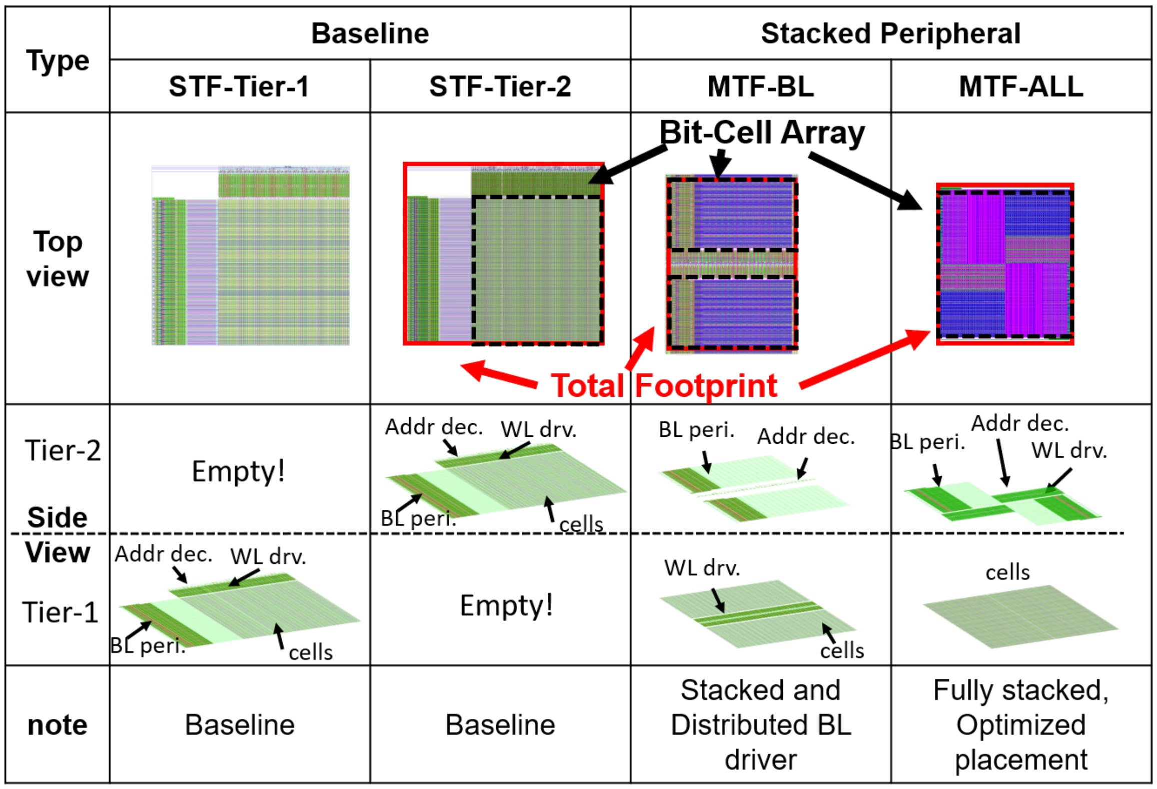
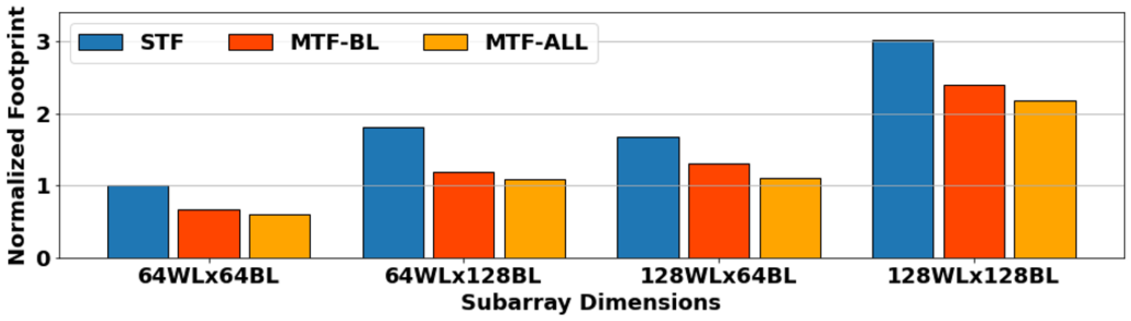
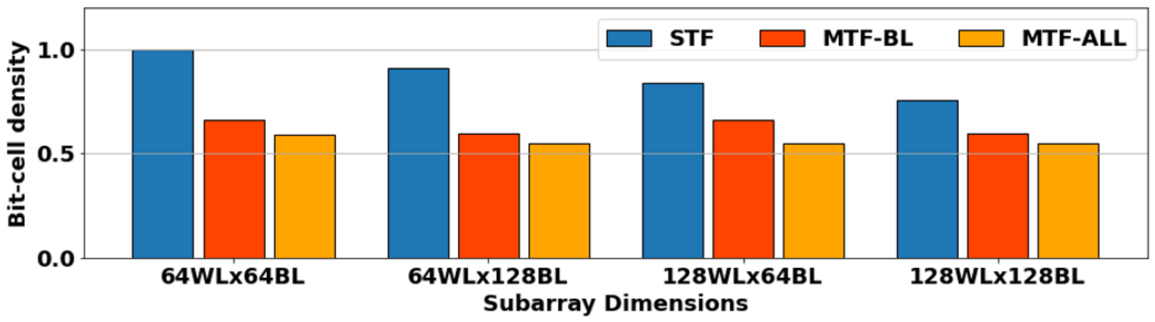


Figure 3.1: Visualization of layouts of different subarray (64WL×64BL) structures.

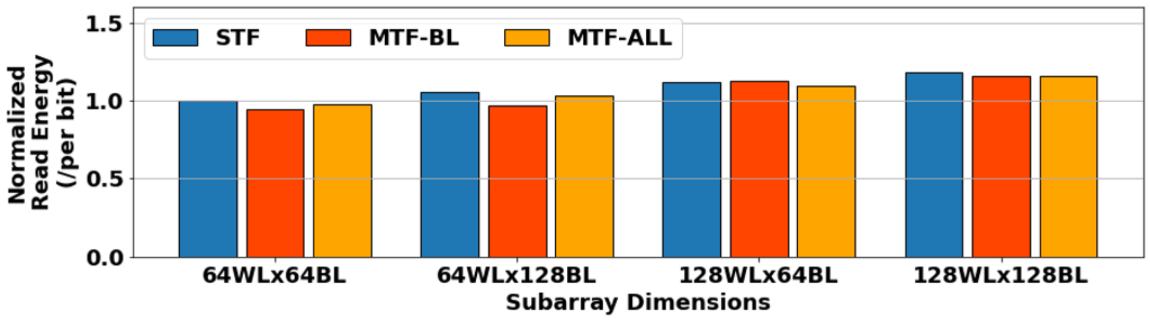


(a)

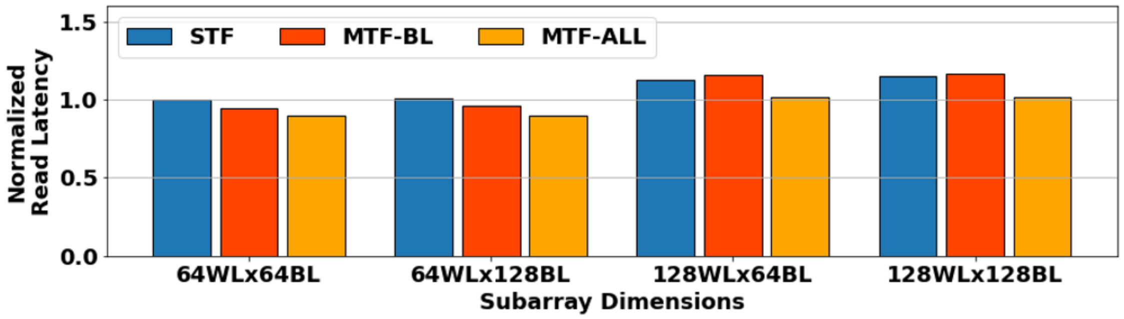


(b)

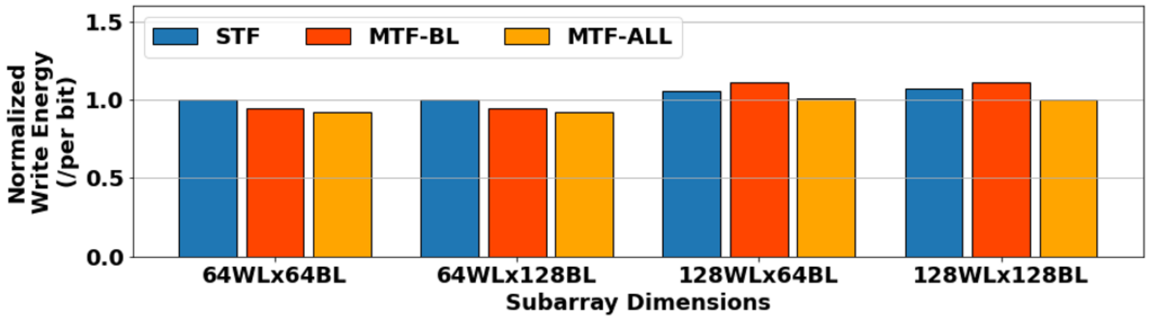
Figure 3.2: Subarray area analysis: (a) footprint, (b) bit-cell density.



(a)



(b)



(c)

Figure 3.3: Normalized subarray's comparison for different subarrays on (a) Read energy per bit, (b) read latency, and (c) write energy per bit.

3.2.2 Cell-density Analysis

Figure 3.2 (b) shows the bit-cell density for different subarray architectures. In traditional 2-D structures, the reduced cell density limits the usage of small subarrays. The cell density can be increased by $1.32\times$ by changing STF subarray dimensions from 64WL$\times$64BL to 128WL$\times$128BL. MTF subarrays with 64WL$\times$64BL can achieve superior bit-cell density than 128WL$\times$128BL STF.

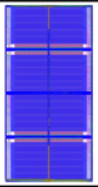
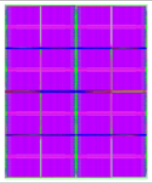
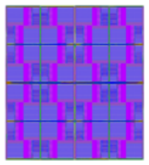
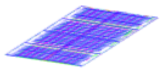
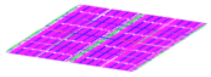
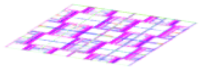
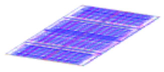
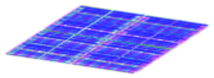
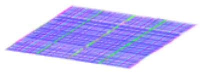
Type	Single Tier	Stacked Peripheral	
	STF-Stack	MTF-BL	MTF-ALL
Top view			
Top **Side**			
View Bot			

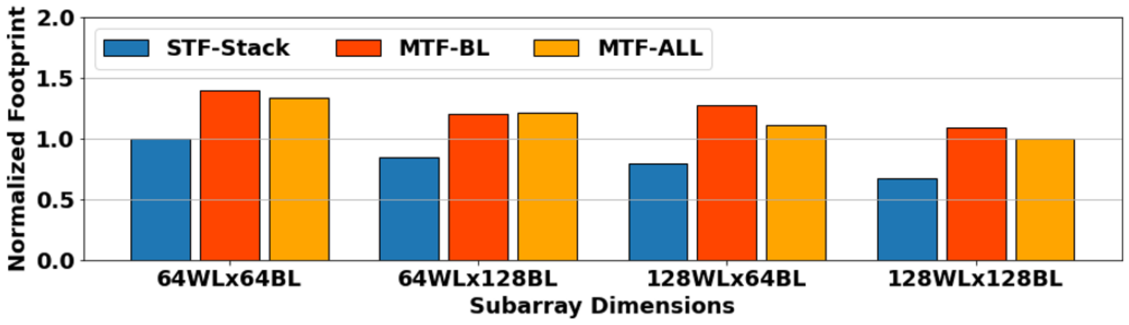
Figure 3.4: Layout of compiled arrays (32KB with 128WL×128BL subarrays).

3.2.3 Read Latency and Energy Analysis

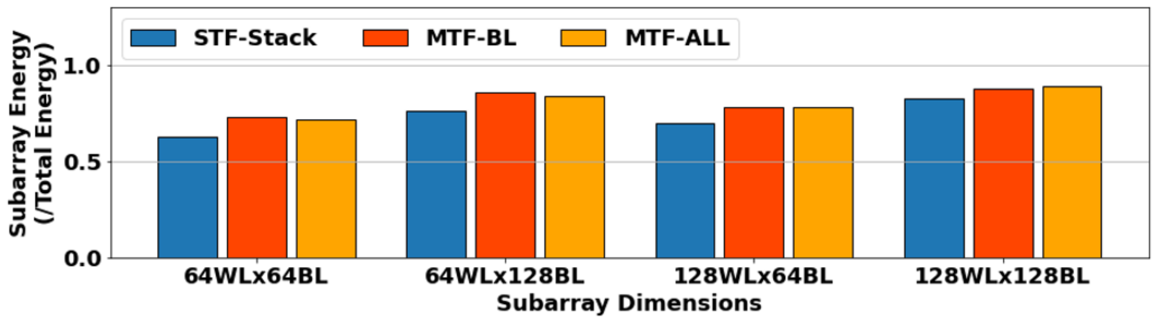
Figure 3.3 compares the read performance and read/write energy per bit of different types of subarrays. The data is normalized to the energy and latency of STF subarrays at the **same dimensions**. The results shows that the MTF-ALL designs show lower read latency and read energy compared to the STF subarrays for all subarray dimensions. MTF-BL structures show lower read latency for small subarrays but higher read latency for large subarrays due to the additional parasitics introduced when the driver output stage is distributed across longer BLs for large subarrays.

3.2.4 Write Energy Analysis

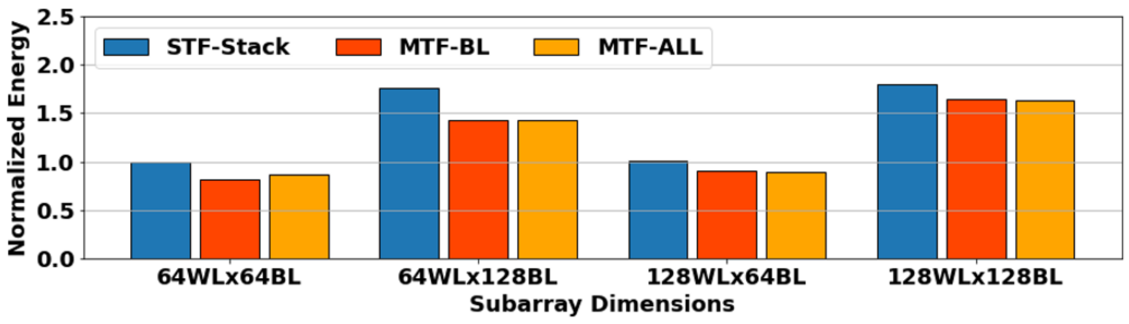
As the write energy is dominated by the parasitic resistance and capacitance along the BL. The result shows a reduction in write energy for both MTF designs at smaller subarray dimensions. However, similar to read access metrics, a marginal increase in write energy for MTF-BL due to the higher parasitics introduced to/by distributed drivers is also observed for larger subarray dimensions. MTF-ALL subarrays, on the other hand, demonstrate a reduction in write energy for all subarray dimensions.



(a)



(b)



(c)

Figure 3.5: SRAM block (32KB) analysis (a) Normalized footprint (b) Subarray's energy per total energy (c) Normalized total energy.

3.3 SRAM Block Compilation Results

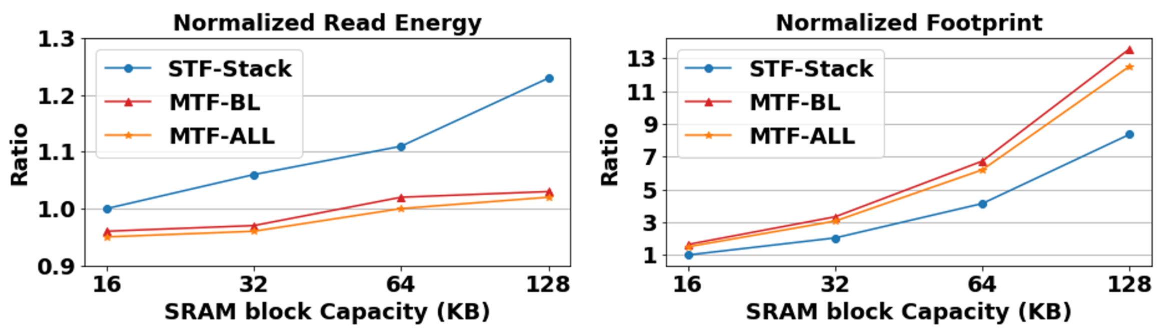
Figure 3.4 shows the top view and side view of an SRAM block. The STF-Stack design represents the stacking of Tier-1 and Tier-2 STF arrays to create the entire capacity. In this example, the 32KB of SRAM block is designed with 128WL$\times$128BL subarrays. Sixteen subarrays are used in MTF designs while STF designs in each tier use eight subarrays. 32-bit bus width is assumed for simulation. The frequency target for the PNR follows the subarrays' maximum frequency (Figure 3.3).

3.3.1 Footprint Analysis

Figure 3.5 (a) shows the footprint comparison results (normalized to footprint generated by 64WL$\times$64BL STF subarray). As expected, the subarray with a larger dimension reduces the total footprint. The STF-Stack shows a lower footprint as the STF subarrays are stacked using both Tier-1 and Tier-2. The MTF-ALL shows a smaller footprint than the MTF-BL as all peripherals are in 3D.

3.3.2 Energy Analysis

The minimum energy design of an SRAM block depends on the trade-off between subarray and top module energy. A larger dimension increases subarray energy but reduces top module's energy as fewer subarrays are used. Also, using large subarrays reduces footprint and parasitic capacitance in the top module. Hence, the results show that larger subarrays increase the ratio of the subarray's energy to the SRAM block's energy as shown in Figure 3.5 (b). As the majority of the energy consumption is from the subarrays, using larger sub-array dimensions increases the energy of an SRAM block (Figure 3.5 (c)).



(a) (b)

Figure 3.6: Scalability of SRAM block capacity: (a) read energy (b) footprint.

3.3.3 Scalability of the Compiler

The developed compiler is used to generate SRAM blocks of varying capacities (Figure 3.6). All of the designs use the 128WL×128BL subarray to implement the SRAM block. As only one subarray consumes read/write energy irrespective of the total capacity, normalized read energy only increases marginally for larger capacity, mainly due to the increased complexity of the top module (Figure 3.6 (a)). On the other hand, the footprint advantage of STF-Stack over MTF designs increases with the larger capacity of the SRAM block (Figure 3.6 (b)).

3.3.4 Block Optimization

Figure 3.7 shows the memory compiler output for 32KB SRAM block but varying design targets. The compiler explores the 3 types of subarrays (STF-Stack, MTF-BL and MTF-ALL) and 4 dimensions (64WL×64BL, 64WL×128BL, 128WL×64BL and 128WL×128BL) while generating final designs. The compiler observes that designs with the largest possible STF-Stack result in the minimum footprint. On the other hand, the MTF-ALL subarray with the smallest dimension shows the minimum latency. Among the 12 cases that are compared, the 32KB SRAM block designed with 64WL×64BL MTF-BL subarray shows the minimum energy. However, when the compiler generates the 128KB SRAM block, MTF-ALL with 128WL×64BL subarray-based design shows minimum energy.

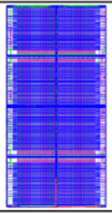
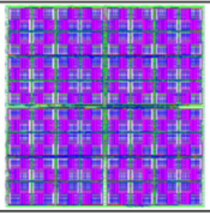
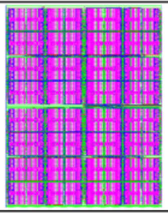
	Minimum Footprint	Minimum Latency	Minimum Energy
Top view			
Type	STF-Stack	MTF-ALL	MTF-BL
Dimension	128WLx128BL	64WLx64BL	64WLx64BL
Normalized Footprint	0.67	1.34	1.40
Normalized Latency	1.15	0.9	0.95
Normalized Energy	1.8	0.86	0.81

Figure 3.7: Memory compiler generates the best design for different objectives. All the data is normalized by STF-Stack 64WL×64BL subarray-based design.

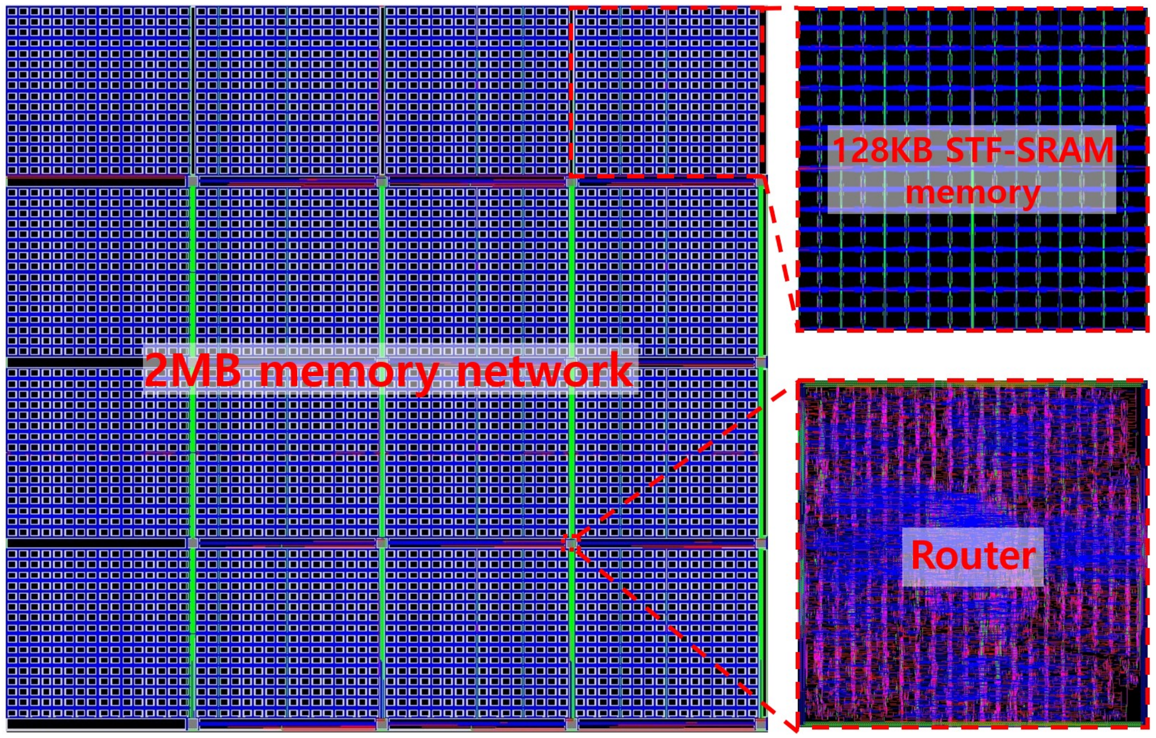
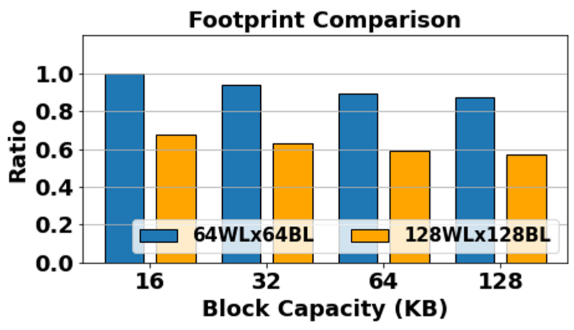
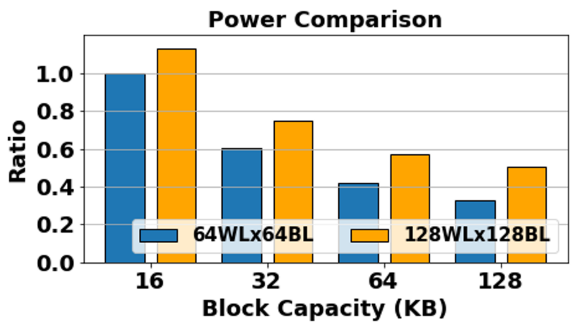


Figure 3.8: The 2MB memory network consisting of 4x4 128KB SRAM blocks.



(a)



(b)

Figure 3.9: Analyze the (a) footprint and the (b) power ratio about 2MB memory network which consists of different SRAM block and subarray capacity.

3.4 Multi-port SRAM Array Analysis

Figure 3.8 shows the layout of a 2MB SRAM array designed with 4×4 SRAM blocks and routers. Each 128KB SRAM block is designed with STF-64WLx64BL subarrays. Figure 3.9 shows the footprint and power of the design considering various SRAM block and subarray capacities. All the results are normalized to the SRAM array generated by 16KB SRAM block and 64WLx64BL subarray. As expected, using a higher capacity SRAM block shows a smaller footprint and lower worst-case power (assuming all routers are active) because the number of SRAM blocks and memory-routers is decreased. Although the power of individual SRAM blocks is higher with a larger capacity, the power reduction due to fewer routers dominates. Note, if only a few routers are active (i.e. local memory access) and/or router power is more optimized, the smaller capacity blocks will be more power efficient. Among different subarray choices, as expected from Figure 3.5, the design with 128WLx128BL subarray shows a lower footprint but higher power than the design with 64WLx64BL subarray.

3.5 Logic and Memory Stacking

The physical design of a 3D multi-core design is implemented using the proposed compiler (Figure 3.10). The compiler uses the OpenPiton as multi-core architecture [18] connected

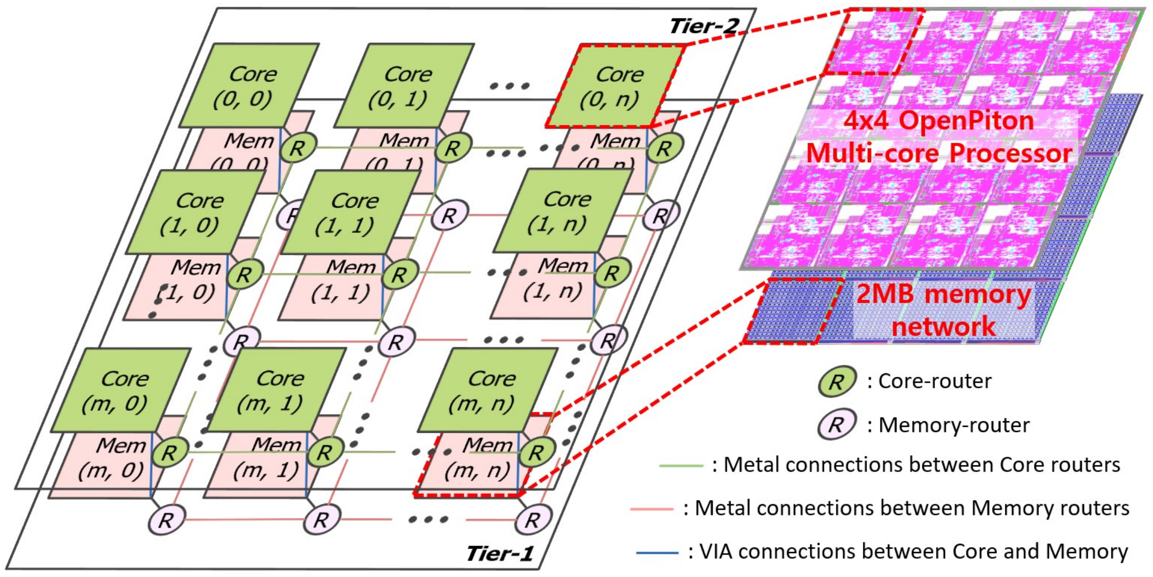
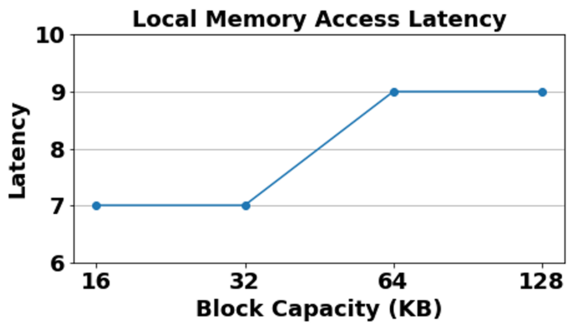
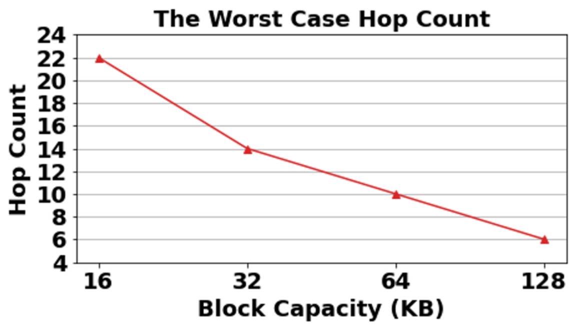


Figure 3.10: 3D multi-core processor and multi-port memory network stacked architecture.



(a)



(b)

Figure 3.11: Analyze the (a) local memory access latency and the (b) worst-case hop count about 2MB memory network which consists of different SRAM block capacity.

to a shared on-chip cache organized as multiple networked SRAM blocks. Figure 3.10 show layout of a design with 4×4 OpenPiton cores connected to 2MB of shared cache organized as 4×4 128KB SRAM blocks (same as in Figure 3.8).

The multiple cores in M3D tier-2 are connected via a core NoC, and each core vertically connects to a memory router in tier-1. The core-to-core communication occurs via the NoC in the core-tier, while the NoC in the memory tier supports the core-to-memory communication. The distributed memory access enables fast and low-latency access between a core and its local memory, thanks to direct vertical access (reduced wire-length) using ILVs. The multiple access ports also avoids the memory congestion associated with using a single port. On the other hand, the memory network enables any core to access any memory location enabling a large shared cache. The absence of the memory network will limit the available memory capacity of each core, and lead to data duplication and cache coherence challenges.

The impact of SRAM block capacity on the memory access latency is analyzed (Figure 3.11). The local read or write access from a core to its local SRAM block is performed directly without using a memory-router. Hence, latency of the 'local' memory access depends on the access latency of the SRAM block i.e. number of levels in the H-tree hierarchy. Increasing capacity of SRAM block, increases the local access latency (Figure 3.11 (a)). Thanks to the H-tree organization, latency of the local access is constant for any address. The access to a distant memory block, referred to as the 'global memory access' is performed by using the memory network. Hence, the global access latency is variable depending on the target address. The worst-case latency of the global access depends on the maximum number of hops in the mesh network. Using larger SRAM block capacity reduces worst-case number of hops (Figure 3.11 (b)). Note that, the global access latency of a specific request depends on the memory access patterns of *all* cores which determines the congestion in the network.

CHAPTER 4
DISCUSSION

This paper has studied an M3D memory compiler that can exploit the multiple layers of transistors on a single substrate and the high-density (3D via) interconnection between tiers. The design concepts and compiler methodologies developed in this paper can be adopted for alternative M3D IC technologies. In particular, as the array generation and the multi-port memory network generation follows logic design automation methods, they can be easily ported to other M3D processes. The circuit design concepts within the subarray generation are also technology agnostic. However, due to the inherently analog nature of the circuits the exact topologies, sizing, and layout are technology specific.

The 3D SRAM compiler needs additional considerations compared to a traditional 2D SRAM compiler. The goal of the M3D compiler is to maximize the benefits of the M3D PDK by allowing (1) logic to be placed above (below) STF subarrays in bottom (top) tiers and (2) external routing to cross MTF subarrays when needed. The compiler also needs to consider the availability of only two metal layers between the top and bottom tiers of CNFETs. Hence, the M3D SRAM compiler must restrict the use of metal layers (to only one layer above and one layer below the transistors) while implementing bit-cells and peripherals. The above restriction creates several challenges. For example, additional spacing is needed between transistors to allow bypass vias to transit between metal layers above/below the transistor. This spacing relieves the routing congestion, but results in lower area efficiency. Likewise, the M3D compiler can use only a single metal layer to design power delivery network (PDN) of SRAM sub-arrays. Therefore, compared to traditional 2D SRAMs that use multiple layers of metals for PDN, the M3D SRAMs can experience higher IR drop. This IR drop reduces the robustness of the memory operation.

The orientation and organization of BLs/WLs metal wires and VDD/VSS grids while compiling MTF sub-arrays must consider the constraints on metal layer usage. For example, multiple routing vacancies are required to allow ILVs for cross-tier access and intra-tier routing. These routing vacancies are aligned to avoid possible connectivity hazards during the automated generation when subarray dimensions scale. The cross-tier access is the connections for signal/power nets from M3D peripherals to nets in the bottom bit-cells. The intra-tier routing is the transistor-bypass path to allow access between the horizontal-routing layer and the vertical-routing layers that are below and above transistors. Moreover, the routing connections passing through a transistor layer need to be carefully distributed to place ILV stacks in MTF sub-arrays. The compiler places routing connections for WL and BL in the top and bottom tier metal layers, respectively. The ILVs required for BL connection use the vacancies in the top-tier, and the WL connection uses the vacancies in the bottom-tier to improve area efficiency.

The parasitics of individual tiers are important factors while optimizing M3D designs. For example, although CNFETs in different tiers have similar on current, there is still 4%-6% difference in the performance of STF SRAMs in different tiers which can be important for high frequency designs. The differences in parasitics also play the key role in optimizing the design/placement of peripheral circuits for MTF subarrays.

The compiler uses H-tree based connection of multiple sub-arrays of small dimension to create a single-port memory block of higher capacity. The H-tree based connectivity simplifies the router design and wiring density within a memory block; but can only support one memory access per read/write cycle. On the other hand, the compiler connects multiple memory blocks using a mesh-style NoC to create a high-capacity memory array. As each memory block is vertically connected to a logic core, the NoC based design allows multi-port access to the overall large capacity memory allowing high degree of access concurrency (and hence, bandwidth) for local core-to-cache communications, like a distributed cache architecture. However, this approach also preserves the logically shared structure of the cache as all memory blocks can be accessed from any core thereby avoiding the complex cache coherency protocol required in traditional distributed caches. In other words, the proposed approach uses the M3D integration to create a physically distributed but logically shared cache that harness the bandwidth advantage of M3D for local access while maintaining globally shared models of the memory to reduce the burdens on memory managements of distributed caches. Further, connecting memory blocks with a NoC in the memory tier reduces the communication burden on the core routers and separate the core-to-core and core-to-memory communication.

M3D technology can improve the latency and the energy of the memory access on the multi-core system with the additional level of cache and routing resources. In the 2D system design, the memory is placed next to the edge of the processor. Therefore, to read/write function, data is transmitted across all the NoC routers from/to the edge of the processor to/from the target core. This transmission increases the memory access latency, energy and NoC congestion. However, in M3D designs, it is possible to place the multi-port memory above/below the processor. In this design, the compiler can directly connect the memory block to core and does not require the data transmission through NoC routers in the multi-core processor.

CHAPTER 5
CONCLUSION

This thesis demonstrates an SRAM memory compiler for CNFET-based M3D technologies with multiple tiers of transistors. The compiler exploits the ability to have identical device performance in all tiers to generate scalable subarray using single and multiple tiers of peripherals. The compiler efficiently integrates the subarrays in a single or multiple tiers to generate an SRAM block and integrates the blocks using an NoC to generate an SRAM array. Ultimately, the developed compiler automatically generates the layout (LEF/GDS) and timing (LIB) files of SRAM blocks that can be used in the full-chip design. The simulation results show that multi-tier SRAMs with peripheral and bit-cells in different tiers show lower energy and latency while 3D stacking single-tier SRAMs show a better footprint. This thesis also show the feasibility of using the memory compiler to generate an M3D stack of logic and memory. The future work will be improving the compiler as well as exploring new system architectures enabled by efficient M3D memory compilers.

REFERENCES

[1] S. Wong, A. El-Gamal, P. Griffin, Y. Nishi, F. Pease, and J. Plummer, "Monolithic 3d integrated circuits," in *2007 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, 2007, pp. 1–4.

[2] T. Srimani *et al.*, "Heterogeneous integration of beol logic and memory in a commercial foundry: Multi-tier complementary carbon nanotube logic and resistive ram at a 130 nm node," in *2020 IEEE Symposium on VLSI Technology*, 2020, pp. 1–2.

[3] D. Akinwande, S. Yasuda, B. Paul, S. Fujita, G. Close, and H. .-. Philip Wong, "Monolithic integration of cmos vlsi and carbon nanotubes for hybrid nanotechnology applications," *IEEE Transactions on Nanotechnology*, vol. 7, no. 5, pp. 636–639, 2008.

[4] M. M. Shulaker, K. Saraswat, H. .-. P. Wong, and S. Mitra, "Monolithic three-dimensional integration of carbon nanotube fets with silicon cmos," in *2014 Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*, 2014, pp. 1–2.

[5] P. Batude, T. Ernst, J. Arcamone, G. Arndt, P. Coudrain, and P. Gaillardon, "3-d sequential integration: A key enabling technology for heterogeneous co-integration of new function with cmos," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 4, pp. 714–722, 2012.

[6] C. Fenouillet-Beranger *et al.*, "New insights on bottom layer thermal stability and laser annealing promises for high performance 3d vlsi," in *2014 IEEE International Electron Devices Meeting*, 2014, pp. 27.5.1–27.5.4.

[7] L. Pasini *et al.*, "Nfet fdsoi activated by low temperature solid phase epitaxial regrowth: Optimization guidelines," in *2014 SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2014, pp. 1–2.

[8] M. D. Bishop *et al.*, "Fabrication of carbon nanotube field-effect transistors in commercial silicon manufacturing facilities," *Nature Electronics*, vol. 3, no. 8, pp. 492–501, Aug. 2020.

[9] G. Hills *et al.*, "Understanding energy efficiency benefits of carbon nanotube field-effect transistors for digital vlsi," *IEEE Transactions on Nanotechnology*, vol. 17, no. 6, pp. 1259–1269, 2018.

[10] M. M. Shulaker, T. F. Wu, M. M. Sabry, H. Wei, H. .-. P. Wong, and S. Mitra, "Monolithic 3d integration: A path from concept to reality," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 1197–1202.

[11] T. F. Wu *et al.*, “Hyperdimensional computing exploiting carbon nanotube fets, resistive ram, and their monolithic 3d integration,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3183–3196, 2018.

[12] M. Shulaker *et al.*, “Experimental demonstration of a fully digital capacitive sensor interface built entirely using carbon-nanotube fets,” in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2013, pp. 112–113.

[13] G. Hills *et al.*, “Modern microprocessor built from complementary carbon nanotube transistors,” *Nature*, vol. 572, no. 7771, pp. 595–602, Aug. 2019.

[14] M. M. Shulaker *et al.*, “Monolithic 3d integration of logic and memory: Carbon nanotube fets, resistive ram, and silicon fets,” in *2014 IEEE International Electron Devices Meeting*, 2014, pp. 27.4.1–27.4.4.

[15] S. Srinivasa, X. Li, M. Chang, J. Sampson, S. K. Gupta, and V. Narayanan, “Compact 3-d-sram memory with concurrent row and column data access capability using sequential monolithic 3-d integration,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 671–683, 2018.

[16] T. Srimani, G. Hills, M. D. Bishop, and M. M. Shulaker, “30-nm contacted gate pitch back-gate carbon nanotube fets for sub-3-nm nodes,” *IEEE Transactions on Nanotechnology*, vol. 18, pp. 132–138, 2019.

[17] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, “Prediction router: Yet another low latency on-chip router architecture,” *Proceedings - International Symposium on High-Performance Computer Architecture*, pp. 367–378, 2009.

[18] J. Balkind *et al.*, “OpenPiton: An Open Source Manycore Research Framework,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 2, pp. 217–232, 2016.