Canopy: A CNFET-based Process Variation Aware Systolic DNN Accelerator

Cheng Chu
chu6@iu.edu
Indiana University Bloomington
USA

Dawen Xu
xudawen@seehi.com
Seehi Microelectronics Co., Ltd
China

Ying Wang
wangying2009@ict.ac.cn
Chinese Academy of Sciences
China

Fan Chen
fc7@iu.edu
Indiana University Bloomington
USA

ABSTRACT

Although systolic accelerators have become the dominant method for executing Deep Neural Networks (DNNs), their performance efficiency (quantified as Energy-Delay Product or EDP) is limited by the capabilities of silicon Field-Effect Transistors (FETs). FETs constructed from Carbon Nanotubes (CNTs) have demonstrated >10× EDP benefits, however, the processing variations inherent in carbon nanotube FETs (CNFETs) fabrication compromise the EDP benefits, resulting >40% performance degradation. In this work, we study the impact of CNT process variations and present Canopy, a process variation aware systolic DNN accelerator by leveraging the spatial correlation in CNT variations. Canopy co-optimizes the architecture and dataflow to allow computing engines in a systolic array run at their best performance with non-uniform latency, minimizing the performance degradation incurred by CNT variations. Furthermore, we devise Canopy with dynamic reconfigurability such that the microarchitectural capability and its associated flexibility achieves an extra degree of adaptability with regard to the DNN topology and processing hyper-parameters (e.g., batch size). Experimental results show that Canopy improves the performance by 5.85× (4.66×) and reduces the energy by 34% (90%) when inferencing a single (a batch of) input compared to the baseline design under an iso-area comparison across seven DNN workloads.

CCS CONCEPTS

• **Hardware → Emerging technologies**; • **Computer systems organization → Architectures**.

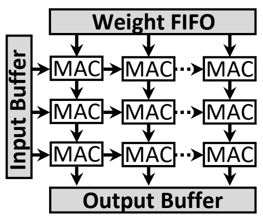
ACM Reference Format:
Cheng Chu, Dawen Xu, Ying Wang, and Fan Chen. 2022. Canopy: A CNFET-based Process Variation Aware Systolic DNN Accelerator. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '22), August 1–3, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3531437.3539703

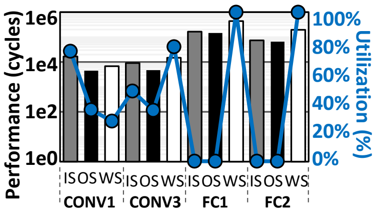
1 INTRODUCTION

The excessive computational intensity of Deep Neural Networks (DNNs) motivated the wide adoption of DNN accelerators which provide two to three orders of magnitude performance improvement compared to CPUs/GPUs through intensive data reuse and specifically designed memory hierarchy. Among previous designs, the systolic architecture has been extensively explored [13]. Continued progress in system performance and efficiency of such designs, however, is prevented by the capability of silicon digital systems [1] due to the end of Moore's law and Dennard scaling. Consequently, multiple potential technology candidates [18, 21] are being explored for constructing next-generation electronic systems. In particular, Field-effect transistors (FETs) fabricated with Carbon Nanotubes (CNTs) has demonstrated 10× improved Energy-Delay Product (EDP) compared with silicon CMOS [6] based memory [18] and processors.

Despite the EDP benefits, CNFETs are subject to substantial intrinsic defects and process variations (PVs) [3] introduced during the current synthesis processes used to produce CNTs. Previous works [12] simulated the latency distribution in ALU (SRAM) and reported a worst-case delay of 2× (3×) the nominal value, indicating that PVs have a significant impact on the operational reliability of CNFET circuits. In this work, we study the impact of CNT process variations on systolic architectures and present Canopy, a process variation aware DNN accelerator that enable efficient DNN processing while retaining the major EDP benefits of CNFETs. The following highlights the challenges and outlines our contributions.

(1) **Previous approaches sacrificed the EDP benefits of CNFETs.** We model the CNT process variations and characterize their impact on systolic DNN accelerators. Experimental results indicate that current techniques (i.e., *worst-case design* and *CNFET upsizing*) both sacrificed the EDP benefits of CNFETs by forcing all processing engines (PEs) to work at the same frequency, resulting >40% performance degradation. Fortunately, the process variation in CNFETs exhibits a strong direction-dependant correlation [24]. Therefore, we implement Canopy with a systolic PE layout that allows PEs to operate at their best performance with non-uniform latency, thereby minimizing performance degradation incurred by CNT variation.
(2) **Non-uniform PE processing is promising to preserve EDP benefits, but it causes problems in data synchronization.** To address this challenge, we propose a computing dataflow that





(a) The TPU architecture. (b) Exploration results on TPU.
Figure 1: Google TPU.

breaks the long systolic pipelines in conventional designs and exploits data parallelism when inputs are processed in batches, enabling high-throughput parallel execution across heterogeneous computing units.

(3) **The new dataflow is effective only for batch processing.** For single-input scenarios, the proposed dataflow will fail as there is no data parallelism opportunity. To this end, we device Canopy with dynamic reconfigurability, providing adaptation for latency-oriented single-input processing.

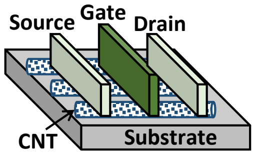
While Canopy addresses the negtive impact of CNT process variations to enable efficient DNN execution in CNFETs, it does not impose extra overhead, but instead provides improved performance and efficiency for both single-input and batch DNN processing. To establish the effectiveness of our design innovations, we evaluate Canopy on various applications across seven DNN workloads. On average, Canopy delivers 5.85× (4.66×) speedup and 34% (90%) energy saving over a baseline CNFET design when inferencing a single input (a batch of inputs) under an iso-area comparison.

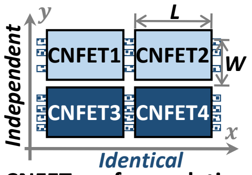
2 BACKGROUND

2.1 Systolic Architecture

Systolic DNN accelerator and its dataflow. The Google TPU [13] is used as the baseline systolic DNN accelerator architecture in this work. We show the high-level TPU architecture overview in Figure 1 (a), which mainly consists of (1) 256×256 8-bit Multiply-and-Accumulate (MAC) units; (2) local buffers including weight FIFO, input buffer, output buffer that are used to store weights, input feature maps (IFPs), and output feature maps (OFPs), respectively; and (3) a 24 MB on-chip SRAM for data storage (not shown in Figure 1 (a)). Depending on different data mapping strategies, the computing dataflow can be classified into *Input Stationary* (IS), *Weight Stationary* (WS), and *Output Stationary* (OS). The dataflow has direct implications on the performance of the systolic computing system. The microarchitecture of a systolic accelerator is usually optimized for an immutable dataflow.

Exploration on TPU. For preliminary studies, we implemented a TPU-like systolic array and simulated it using SCALE-Sim [17]. Detailed experimental setup is described in Section 5. We use convolutional (CONV) and fully-connected (FC) layers in AlexNet as our study cases. The bar chart in Figure 1 (b) compares the performance of different dataflow when inferencing a batch (e.g., a batch size of 64) of inputs. It shows that the OS dataflow outperforms the IS/WS dataflow in most aspects. We also report the the hardware utilization when inferencing a single input with different layer topologies. As showing in the blue line in Figure 1 (b), the hardware utilization ratio can be <5% on an FC layer, indicating that computing resources need to be flexibly reconfigured to adapt to different networks or even different layers of a network.





(a) The CNFET 3D view. (b) CNFET perf. correlation.
Figure 2: CNFET basics.

2.2 CNFET Technology

CNFET-based circuits and systems. As shown in Figure 2 (a), a CNFET device utilizes CNTs arrays instead of bulk silicon as channel material and are implemented in the same structure as conventional MOSFETs [18]. CNTs are hollow cylinders of carbon atoms with a 1∼2 nm diameter and its conductivity can be adjusted by the gate voltage. CNFETs have shown 10× improved EDP [6], low leakage power [20], scalability down to 3 nm and beyond [8], and superior intrinsic thermal properties [22]. High-performance and energy-efficient CNFET-based digital processors [18] and analog circuits [2] have already been experimentally demonstrated.

CNT process variations. CNFETs performance are quantized in terms of the CNT-count in each device. Limited by the current CNT growth processes [24], it is difficult to ensure uniform density (i.e., *CNT density variation*) and precise positioning of CNTs (i.e., *mis-aligned CNTs*). Moreover, roughly 33% of the CNTs are metallic (i.e., *m*-CNT) [19, 25]. Current *m*-CNT removal techniques [23] may inadvertently remove some semiconducting CNTs (*s*-CNTs), aggravating the CNT density variations. Previous work [24] presented a parameterized model for CNT-count by incorporating all the aforementioned process variations. As validated by experimentally results, the CNT-count in a practical sized CNFET follows the Gaussian distribution and can cause significant latency variations in CNFETs-based computing units and memory elements [12].

Existing solutions. The misaligned-CNT-immune layout [16] has addressed the *mis-aligned CNTs* problem with minimal overhead. For other CNT process variations, there are mainly two main schemes. (1) *Worst-case design*: For a CNFET circuit, a nominal cycle time is determined by the critical path delay when there are no variations. Such *worst-case design* ensures reliable compute but sacrifices system performance; (2) *CNFET upsizing*: The CNFETs circuit failure probability, p_F, is defined by the following equation [26], where *CNT density variation*, $Prob\{N(W) = N_i\}$, is a function of the CNFET chanel width W, and $p_f^{N_i}$ represents the *m*-CNT induced variation. Since p_F decreases exponentially with W. Therefore, a minimal W can be determined to meet the delay requirements of a target system.

$$\mathrm{p}_F = \sum_{N_i} p_f^{N_i} Prob\{N(W) = N_i\} \tag{1}$$

Spatial correlation. A strong spatial correlation in CNT process variations have been identified and confirmed by prototypes [24]. Figure 2 (b) conceptually illustrates this unique property with a simple four CNFET layout. The CNT-count distribution along the CNT growth direction is highly correlated, therefore, for FETs implemented along the CNT growth direction, i.e., *CNFET1* and *CNFET2*, have identical performance. While for FETs implemented perpendicular to the CNT growth direction, i.e., *CNFET1* and *CNFET3*, are independent. Note that when the *x-distance* increase beyond the length of CNT stripes, a change in correlation should be expected

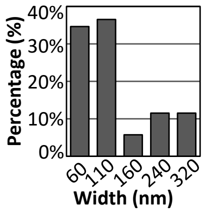


Figure 3: CNFET size distribution.

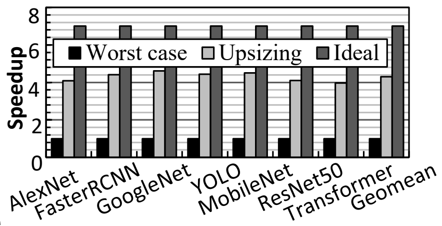


Figure 4: Speedup (norm. to TPU).

even for devices along the *x-axis*. Current fabrication processes can achieve stable and consistent millimeter-scale CNTs [4], which can accommodate large functional units with tens of thousands of CNFETs.

3 RELATED WORK AND MOTIVATION

Baseline modeling. For *ideal* CNFET-based designs, we model the circuit in Verilog and synthesize it with Synopsys Design Compiler using 28 nm Open Cell Library (modified for CNFET technology). We achieved a clock frequency of 2.4 GHz. The transistors number per 8-bit MAC is 2817. We implement each MAC in a square-sized area. We adopt the misaligned-CNT-immune layout [16] in the backend flow using Cadence Innovus. Based on the practical device nonidealities calibrated with experimental CNT data [5], a conservative 13.2% (12.5%), 11.6% (13.4%), and 8.1% (6.4%) overhead is imposed respectively on circuit area, energy, and delay for an 8-bit full adder (multiplier). We assume a 200 *um* [4] CNT length and validated that 256 MACs can be realized within the length. For *non-ideal* designs, we apply the CNT count variation by using parameters in Table 1. Detailed explanation of each parameter can be found in Section 5.

Performance comparison. We show the extracted transistor sizing distribution in Figure 3. According to Equation 1, a minimum 160 nm of transistors width is desired to ensure a >90% circuit yield, resulting a 2× enlarged area and 1.5% increased delay (due to gate capacitance penalty associated with transistor upsizing) in the *CNFET upsizing* implementation. For fair comparison, we increase the computing resource for *worst-case design* by 2× and report the iso-area performance comparison in Figure 4. All data are normalized to *worst-case design*. The results show that *worst-case design* and *CNFET upsizing* respectively lead to 85% and 40% performance degradation compared to the *Ideal* CNFET-based design.

Limitation of prior techniques. Previous CNFET-based designs sacrifice the CNFETs EDP benefits to reduce the impact of process variation. The spatial correlation of CNT variations is exploited in a SIMD processor [12], but the the impact of such correlations on a systolic array is not studied. The recent work [7] proposed a CNFET-based DNN accelerator in monolithic 3D, but its fabrication cost increases significantly due to expensive monolithic 3D interconnects. In this work, we set out to explore process variation aware DNN accelerator designs and expect to retain the benefits of CNFETs despite the CNT process variations that exsist in today's CNT fabrication.

4 CANOPY

In this section, we propose a process variation aware systolic DNN accelerator architecture, Canopy, which allows each MAC row to run at its best performance, and hence retain the EDP benefits of CNFETs. To address the computing synchronization challenge in non-uniform MAC arrays, we present an optimized execution flow for high-throughput DNN batch processing and a dynamic reconfigurability to adapt for latency-oriented single-input processing.

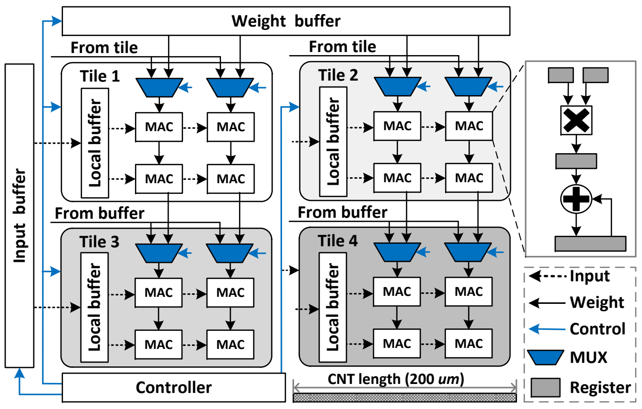


Figure 5: The Canopy architecture overview.

4.1 The Canopy Architecture

Figure 5 shows the Canopy architecture. Canopy consists of four tiles, each of which has a 256×256 MAC array, a 6KB *local buffer* and a two-input *multiplexer* (MUX) for each computing column. Each MAC has two one-byte registers to store IFPs and weights, one two-byte register to store temporal product, and one four-byte register to store the partial sum. Each register is connected to their neighboring MAC units for local operands transforming. Each row has an individual MAC on/off control. The four tiles shares an 8 MB *input buffer*, an 8 MB *weight buffer*, and an 8MB *weight buffer*. A controller manages the dataflow and orchestrate the computing.

Based on our preliminary study in Section 3, all the 256 MACs in a row within the same tile can be implemented along the growth direction of a monolithic 200 μm CNT. We assume they have the same latency due to the CNT spatial correlation. It is safe to assume that the chip can be fully tested after fabrication to obtain the 256 distinct MAC row delay in each tile. In this case, each tile can be taken as 256 1×256 MAC rows with non-uniform computing latency. We quantize the delay in four discrete values and apply a working frequency such as f Hz, $2f$ Hz, $4f$ Hz, and $8f$ Hz (e.g., f=300 *MHz*) to the corresponding MAC row.

A monolithic systolic array architecture inherently utilizes data reuse along its rows and columns in a two-dimensional pipeline [13, 17]. The synchronization of such parallel execution depends on the identity of processing units. However, Canopy contains 256 1× 256 MAC rows cannot support parallelism and ensure synchronization among different rows with non-uniform delays. Therefore, it is desirable that we design a new dataflow that can break the long systolic pipeline in the original design and parallelize DNN execution across multiple (e.g., 256) MAC rows, such that it would exploit finer-grain parallellism and yield better resource utilization.

4.2 The Canopy Computing Dataflow

Data mapping and computing flow. To accommodate the non-uniform MAC delay, the key is to avoid systolic pipeline which requires data synchronization among different rows. Therefore, we

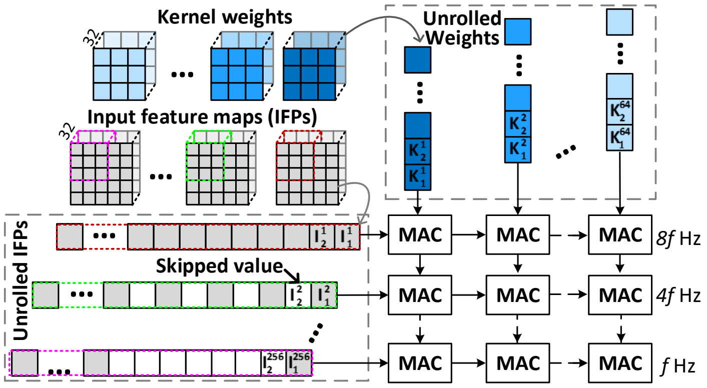


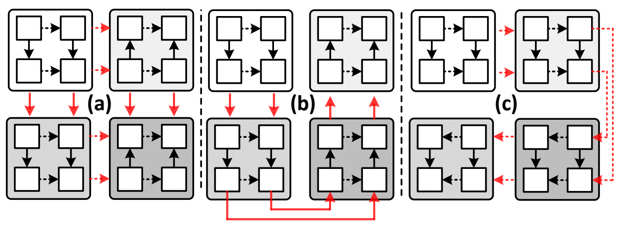
Figure 6: Data mapping and computation on Canopy.

present an OS-like data mapping and computation flow, where the compute required for each element in OFPs are mapped to a given MAC such that the partial sum reduction operation is performed in place with no further communication between MAC rows.

We explain the steps by walking through a simple example of a convolutional layer as illustrated in Figure 6. The example layer has 256 5×5×32 IFPs, 64 3×3×32 weight kernels. We first prepare kernel weights onto different column of the systolic array. We then fill MAC rows with unrolled inputs from *different* IFPs. The whole array is clocked at the highest working frequency, i.e., $8f$ Hz.

Now consider the following computation order. In the first cycle, the first element in the first IFP, i.e., I_1^1, and the first weight value in the first weight kernel, i.e., K_1^1, are read into the leftmost MAC in the first row. Accordingly, a partial sum in the first channel of the first OFP is calculated and stored in place. In the second cycle, the leftmost two MACs in the first row and the leftmost MAC in the second row both obtained their operands to generate an output value in the corresponding OFPs. Different MAC rows have different compute delay and hence their operands (e.g., IFPs and weights) are consumed at different frequency. For the specific configuration in Figure 6, it takes respectively 1, 2, and 8 cycles for the 1^{st}, 2^{nd}, and last systolic row to complete one MAC. In this case, when MAC rows with a working frequency of $8f$ Hz finish calculation and are expecting a new set of IFPs/weights to be loaded, MAC rows with a frequency of $4f$, $2f$, and f Hz only complete calculation on 1/2, 1/4, and 1/8 of the IFPs/weights values.

Operands alignment on non-uniform systolic array. To ensure functional correctness in such a systolic array with non-uniform row-wise latency, IFPs and weights need to be carefully aligned with each other. To address this problem, a control signal is sent to each row every cycle by using the discrete on/off logic reserved for each MAC row. As shown in Figure 6, no particular dataflow control is applied in the first round of computation. While in the second round, a *row_off* signal is applied on the first cycle to all the MAC rows except those with an $8f$ Hz frequency. In this case, MAC rows with a frequency of $4f$, $2f$, and f Hz can finish compute on the unused IFPs in the buffer while MAC rows with a frequency of $8f$ is executing on a new set of loaded IPF values. Similarly, a *row_off* signal is applied on the first two cycles for MAC rows with a frequency of $2f$ and f in the next round. With such configuration, operands for non-uniform MAC rows are respectively aligned to ensure the correctness of function.



	(a)	(b)	(c)
Application	Batch input	Single CNN	Single FC
Whole array size	512x512	1024x256	256x1024
Optimal array size	128x512	256x256	64x1024

Figure 7: Illustration of possible configurations.

4.3 Dynamic Reconfigurability

The Canopy dataflow exploits reuse of weights among *different* IFPs, which is optimized for the high-throughput execution of DNN batch (e.g., 64, 128, etc.) processing. For single-input DNN processing, there is no such data parallelism, and more importantly, the most critical metric becomes latency instead of throughput. To this end, we devise Canopy with reconfigurability among tiles such that the fast MAC rows (i.e., MAC rows working at $8f$ Hz) from each tile can be resembled to construct a uniform systolic array for latency-oriented DNN processing. The reconfigurability is ensured by (1) *dynamic weights selection:* the weights fed to the top row of each tile come either from neighboring tile or directly from the weight buffer; and (2) *direct inputs fetch:* inputs streamed to the left edge of each tile are directly fetched from the local buffer to avoid the non-uniform row delay among different tiles.

Canopy is composed of four tiles that are physically implemented in a 2×2 layout. Figure 7 illustrates the three possible logical configurations. Figure 7 (a) shows a assembling array with square aspect-ratio, while Figure 7 (b) and (c) illustrate instances of vertical long-narrow and horizontal short-wide configurations, respectively. In general, Figure 7 (a) yields the best performance for batch processing. For single-input processing, Figure 7 (b) and (c) will be the best choice respectively for a model that dominated by CONV or FC layers. The table in Figure 7 lists the applicable tasks and the overall array size for each configuration. For single input inference, Canopy provides an high-performance MAC array where only the MAC rows with a frequency of $8f$ Hz are selected. In this case, a CNFET-based monolithic systolic array with uniform MAC delays can be constructed.

4.4 Design Overhead

One MUX is added to each MAC column, which is negligible. One 6 KB local buffer is added to each tile, resulting in a total of 24 KB SRAM. Compute alignment requires a minimal circuit revision for adding a control signal to each MAC row.

5 EXPERIMENTAL METHODOLOGY

CNT process variation modeling. We use the variation-aware CNFET process design kits (PDKs) [9] to evaluate the circuit-level impact of CNT count variations. The CNT count variations are quantified by the parameters shown in Table 1. Back-end-of-line (BEOL) wire parasitics are extracted with Synopsys Raphael using wire dimensions in [13] and then modified for CNFET technology.

Table 1: CNT processing parameters.

Definition	Ideal Val.	Experimental Val.
Mean μ and variance σ	$\sigma^2/\mu^2 = 0$	$\sigma^2/\mu^2 = 0.5$
Probability of $m-CNT$	p_m = 0%	p_m = 10%~33%
C.P.* of removed s-CNT	p_{Rs} = 0%	p_{Rs} = 1%
C.P. of removed m-CNT	p_{Rm} = 100%	p_{Rm} = 99.99%

* *C.P.: Conditional Probability.*

We also leverage the SPICE-compatible CNFET compact model [6], which is calibrated with experimental data from CNFETs down to 9 nm gate length. We leverage RTL-based simulations to model the impact of timing violations. Variations in standard cell power/timing are characterized using the methodology described in [10]. We quantify the following circuit-level metrics: (1) nominal systolic array (256×256) delay/energy: the critical path delay and associated energy when there are no timing variations; (2) MAC row (1× 256) delay/energy: the compute delay and associated energy of MACs implemented with correlated CNTs. We adopt the CNFET-based SRAM in [14]. We upsize all non-MAC logic (e.g., control, nonlinear units, etc.) to ensure system reliability, similar to [11].

Simulation. We adopt SCALE-Sim [17] to capture the MAC compute cycles, MAC resource utilization, memory bandwidth and access counts for various accelerator configurations and DNN architecture. The dynamic energy consumption are calculated based on the compute cycles and the energy of different components obtained from circuit-level simulation. The chip static energy consumption are estimated based on the chip size and validated against previous CNFET-based digital prototype [11]. We assumed a low-power DRAM interface with 4 pJ/bit, similar to baseline HBM [15].

Benchmarks. We evaluate the Canopy architecture on various deep learning applications across seven state-of-the-art DNN models. The detailed DNN topology and compute density are summarized in Table 2. Note that we use the same DNN configurations and workloads hyper-parameters as in [17]. We quantized both the activations and weights of all CNNs with 8-bit of precision, similar to Google TPU [13].

Schemes. We implement the following schemes:

- *Worst-case:* The chip contains 4 256×256 systolic arrays. The working frequency is set as the slowest f Hz to ensure there are no variations;
- *Upsizing:* All CNFETs are upsized to meet the $8f$ Hz working frequency. The chip contains 1 256×256 systolic array. A 30% delay overhead is applied due to the increased gate capacitance penalty at 28 nm technology node [26].
- *Ideal:* The chip has 4 256×256 systolic arrays. No process variation is considered. The frequency is set at $8f$ Hz.

Table 2: The DNN benchmarks (C: conv. layer; F: FC layer).

Name	Database	Topology	Ops
AlexNet*	ImageNet	5C	724.4M
ResNet-50	ImageNet	49C,1F	4.1G
MobileNet	ImageNet	10C,1F	569.4M
GoogleNet	ImageNet	57C,1F	1.5G
YOLO	ImageNet	9C	2.24G
FasterRCNN	ImageNet	46C	3.9G
Transformer	ImageNet	288C,603F	868.7M

* *We omit the FC layers in AlexNet similar to [17].*

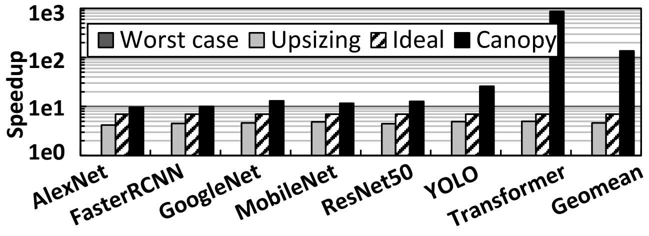


Figure 8: The batch speedup (norm. to TPU).

- *Canopy:* The chip contains 4 256×256 systolic arrays with non-uniform MAC row delays.

The MAC rows in *Worst case*, *Upsizing*, and *Ideal* all runs at identical speed. Their data mapping and computing flow follows the conventional scheme presented in the TPU work [13]. For Canopy, we implement all data mapping and computing flow schemes proposed in this work.

6 RESULTS AND ANALYSIS

6.1 Inferencing A Batch of Inputs

Performance. Figure 8 compares the DNN batch processing performance. Among the four schemes, the available on-chip computing resource in *Upsizing* has reduced by 4× compared to others due to the exponentially increased CNFET device sizes, while its working frequency is 8× higher than *Worst case*. On average, *Upsizing*, *Ideal*, and *Canopy* achieves respectively 4.66×, 7×, and 135.2× speed up compared to *Worst case*. It is remarkable that *Canopy* achieves ~20× speedup compared with *Ideal*, which can be attributed to the efficient Canopy computing dataflow. More specifically, the utilization of systolic array in OS dataflow is greatly limited by the hyper-parameters of DNN layers, i.e., <5% when processing an FC layer. In our design, the problem of low utilization is addressed by parallel processing different inputs in each row. And of course, the effectiveness of such dataflow only holds when a batch of IFPs are processed at the same time. For batch inferencing of FC-dominated Transformer, an order-of-magnitude improvement can be observed.

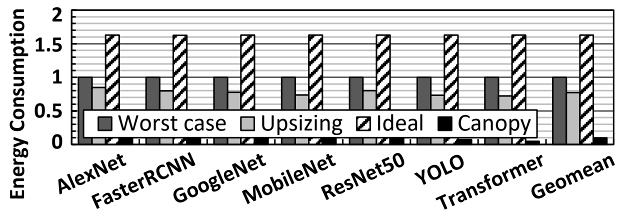


Figure 9: The batch energy consumption (norm. to TPU).

Energy. Figure 9 compares energy consumption of different schemes when inferencing a batch of IFPs. Although *Worst case* achieves a lower power consumption due to its very low working frequency (i.e., 300 MHz), its prolonged processing latency leads to a high energy consumption. *Upsizing* and *Ideal* both run at the highest frequency (i.e., 2.4 GHz), but the computing resource of *Ideal* is 4× that of *Upsizing*. Therefore, *Upsizing* consumes less energy than *Ideal*. In contrast to *Upsizing*, *Canopy* allows each MAC rows run at its best performance without extra hardware overhead. The performance improvement from the dataflow also leads to significant reduced computing latency when compared to *Ideal*. Overall, the energy consumption of *Upsizing*, *Ideal*, and *Canopy* are reported as 0.77×, 1.63×, and 0.096× compared to the baseline *Worst case* scheme.

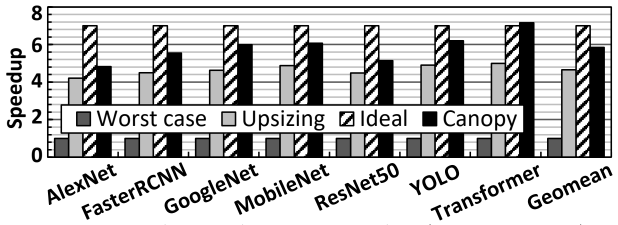


Figure 10: The single input speedup (norm. to TPU).

6.2 Inferencing A Single Input

Performance. Figure 10 compares the single-input performance of different designs. Canopy provides reconfigurability for different workloads and the optimized 65536 (i.e., 256×256) MACs are guaranteed to run at $8f$ Hz. Averagely, *Canopy* achieves 1.25× speedup compared to *Upsizing*. The performance comparison between *Ideal* and *Canopy* depend on the topology of DNN workloads. In general, *Canopy* with optimized array size also runs at $8f$ Hz, but its computing resource is only 1/4 of *Ideal*. For all the seven workloads, *Ideal* and *Canopy* show respectively 7× and 5.85× improvement compared to *Worst case*. For the FC-dominated *Transformer* model, *Canopy* achieves a 2% speedup than *Ideal*. This is mainly due to the flexibility of array reconfigurability, which provides improved hardware utilization.

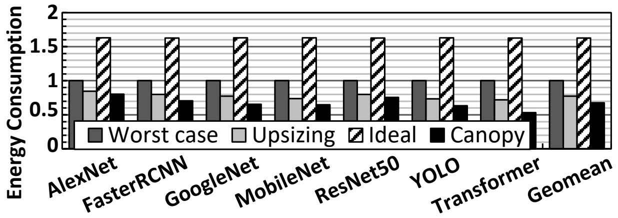


Figure 11: The single input energy consum. (norm. to TPU).

Energy. Figure 11 compares the energy consumpation of different designs when processing a single input. *Upsizing*, *Ideal*, and *Canopy* respectively consume 0.77×, 1.63× and 0.66× energy compared to *Worst case*. The energy saving of *Canopy* when compared to *Ideal* mainly comes from (1) *Canopy* only assembles MAC rows working at $8f$ Hz, while the rest of the array is all powered off, resulting in improved utilization and energy saving; (2) The performance speedup demonstrated in Figure 10 leads to short processing latency, thereby reducing leakage energy consumption.

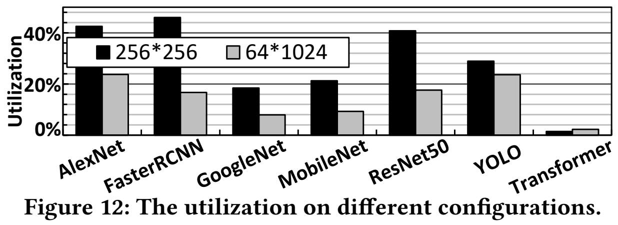


Figure 12: The utilization on different configurations.

Resource utilization. Figure 12 compares the hardware utilization among different Canopy configuration (corresponding to Figure 7 (b) and (c)) when inferencing a single input. In general, CONV-dominated models favor a systolic array with square aspect-ratio, i.e., a 256 × 256 MAC array achieves averagely 2× speedup compared to a 64 × 1024 MAC array. FC-dominated models (e.g., *Transformer*), however, demonstrate a better performance on a short-wide systolic array due to the significantly improved hardware utilization. This can be attributed to the small number of convolution windows and the large number of kernels in such models. Similar observation is also reported in [17].

7 CONCLUSION

In this work, we study the impact of process variations on the performance of a carbon nanotube field-effect transistor based systolic array. We then present Canopy, a process variation aware systolic accelerator for deep neural networks. We co-optimize architecture and dataflow to allow the compute units in a systolic array run at their best performance with non-uniform processing latency. We further device Canopy with dynamic reconfigurability to adapt to different compute requirements of various deep learning models. Experimental results show that Canopy achieves significant performance improvement and energy reduction compared to the baseline design across a wide range of workloads.

REFERENCES

[1] Mohamed M Sabry Aly et al. 2018. The N3XT Approach to Energy-Efficient Abundant-Data Computing. In *Proceedings of the IEEE*.
[2] A. G. Amer, R. Ho, G. Hills, A. P. Chandrakasan, and M. M. Shulaker. 2019. SHARC: Self-Healing Analog with RRAM and CNFETs. In *ISSCC*.
[3] S. Banerjee et al. 2020. Analysis of the Impact of Process Variations and Manufacturing Defects on the Performance of Carbon-Nanotube FETs. *IEEE TVLSIS* (2020).
[4] Qing Cao et al. 2013. Arrays of Sngle-Walled Carbon Nanotubes with Full Surface Coverage for High-Performance Electronics. In *Nature Nanotechnology*.
[5] Jie Deng et al. 2006. A Circuit-Compatible SPICE Model for Enhancement Mode Carbon Nanotube Field Effect Transistors. In *SISPAD*.
[6] Jie Deng and H-S Philip Wong. 2007. A compact SPICE model for carbon-nanotube field-effect transistors including nonidealities and its application—Part II: Full device model and circuit performance benchmarking. *IEEE T-ED* (2007).
[7] Samuel J. Engers et al. 2022. MOCCA: A Process Variation Tolerant Systolic DNN Accelerator Using CNFETs in Monolithic 3D. In *GLSVLSI*.
[8] Aaron D Franklin et al. 2012. Sub-10 nm carbon nanotube transistor. *Nano letters* (2012).
[9] Gage Hills. 2015. Variation-Aware Nanosystem Design Kit (NDK). https://nanohub.org/resources/22582
[10] Gage Hills et al. 2015. Rapid Co-Optimization of Processing and Circuit Design to Overcome Carbon Nanotube Variations. *IEEE TCAD* (2015).
[11] Gage Hills et al. 2018. TRIG: Hardware Accelerator for Inference-based Applications and Experimental Demonstration using Carbon Nanotube FETs. In *DAC*.
[12] Li Jiang et al. 2018. CNFET-Based High Throughput SIMD Architecture. *TCAD* (2018).
[13] Norman P. Jouppi et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *ISCA*.
[14] Tianjian Li et al. 2016. CNFET-based high throughput register file architecture. In *ICCD*.
[15] O'Connor et al. 2017. Fine-Grained DRAM: Energy-Efficient DRAM for Extreme Bandwidth Systems. In *MICRO*.
[16] Nishant Patil et al. 2007. Automated Design of Misaligned-Carbon-Nanotube-Immune Circuits. In *DAC*.
[17] Ananda Samajdar et al. 2018. SCALE-Sim: Systolic CNN Accelerator Simulator. *arXiv e-prints* (2018).
[18] M. M. Shulaker et al. 2014. Monolithic 3D Integration of Logic and Memory: Carbon Nanotube FETs, Resistive RAM, and Silicon FETs. In *IEDM*.
[19] Fabian Teichert et al. 2017. Electronic Transport in Metallic Carbon Nanotubes with Mixed Defects within the Strong Localization Regime. *Computational Materials Science* (2017).
[20] H-SP Wong et al. 2003. Carbon Nanotube Field Effect Transistors-Fabrication, Device Physics, and Circuit Implications. In *ISSCC*.
[21] Damien Woods. 2012. Photonic neural networks. In *Nature Physics*.
[22] Choongho Yu et al. 2005. Thermal Conductance and Thermopower of an Individual Single-Wall Carbon Nanotube. *Nano Letters* (2005).
[23] Guangyu Zhang et al. 2006. Selective Etching of Metallic Carbon Nanotubes by Gas-Phase Reaction. *Science* (2006).
[24] Jie Zhang et al. 2009. Carbon Nanotube Circuits in the Presence of Carbon Nanotube Density Variations. In *DAC*.
[25] Jie Zhang et al. 2009. Probabilistic Analysis and Design of Metallic-Carbon-Nanotube-Tolerant Digital Logic Circuits. *TCAD* (2009).
[26] Jie Zhang et al. 2010. Carbon Nanotube Correlation: Promising Opportunity for CNFET Circuit Yield Enhancement. In *DAC*.