UPTPU: Improving Energy Efficiency of a Tensor Processing Unit through Underutilization Based Power-Gating

Pramesh Pandey, Noel Daniel Gundi, Koushik Chakraborty, Sanghamitra Roy
USU BRIDGE LAB, Electrical and Computer Engineering, Utah State University
{pandey.pramesh1, noeldaniel}@aggiemail.usu.edu, {koushik.chakraborty, sanghamitra.roy}@usu.edu

***Abstract*—The AI boom is bringing a plethora of domain-specific architectures for Neural Network computations. Google's Tensor Processing Unit (TPU), a Deep Neural Network (DNN) accelerator, has replaced the CPUs/GPUs in its data centers, claiming more than $15\times$ rate of inference. However, the unprecedented growth in DNN workloads with the widespread use of AI services projects an increasing energy consumption of TPU based data centers. In this work, we parametrize the extreme hardware underutilization in TPU systolic array and propose UPTPU: an intelligent, dataflow adaptive power-gating paradigm to provide a staggering $3.5\times - 6.5\times$ energy efficiency to TPU for different input batch sizes.**

***Index Terms*—TPU, DNN Accelerator, Power-Gating, Energy Efficiency, Leakage**

I. Introduction

Artificial intelligence (AI) is predicted to contribute up to $15.7 trillion to the global economy by 2030 [1]. In line with the AI evolution, the computing industry has already embraced specialized AI accelerators, as conventional CPUs and GPUs are no longer able to match up the required throughput [2]–[5]. Google's Tensor Processing Unit (TPU) [6], a representative Systolic Array (SA) based architecture, has been widely employed throughout Google data centers to meet the excessive performance demand in its Deep Neural Network (DNN) inference computations. The unprecedented growth in DNN workloads demands huge energy efficiency in these architectures. Additionally, the scarce energy resources coupled with limited hardware cost in battery powered edge-AI applications necessitates an extremely energy efficient inference architecture.

Researchers have demonstrated impressive energy savings from power-gating in CPU/GPU architectures. However, the savings are being limited due to the relatively unpredictable idleness pattern (from general purpose applications) and performance loss considerations due to the sleep and wakeup cycles of sleep transistors [7]–[9].We observe and parametrize a well structured and massive hardware underutilization problem in TPU SA, and uncover a much larger opportunity of improving energy efficiency through power-gating the idle hardware resources.

DNN inference through TPU SA is carried out by performing a matrix multiplication between input matrix and weight matrix, in the array of 256×256 Multiply-And-Accumulate (MAC) units. Weight matrix is preloaded into the SA and the input vectors can be grouped and sent as batches of different sizes. Major chunk of the SA energy consumption comes from the dynamic energy consumed by the computationally active MAC units and leakage energy consumed by idle MAC units. We observe that the number of computationally active MACs on an average for a batch computation period decreases with the batch size. We find that, the MAC units are computationally active only for less than 40% of the time for practical batch sizes, incurring a substantial leakage loss. By parameterizing the activity and idleness pattern of the MAC units, we formulate UPTPU: an intelligent and adaptive power-gating paradigm for SAs. Moreover, UPTPU prevents any performance or accuracy loss by a smart control around circuit level tolerances of the sleep transistors.

Furthermore, as the weights remain static for a batch computation lifecycle and any computation with a zero weight is just an energy overkill, we reuse the same sleep transistor resources to powergate the zero weight holding MACs, further inflating the energy efficiency. More importantly, as the share of leakage energy in the total energy becomes more prominent for lower technology nodes [8], the effectiveness of UPTPU magnifies with the future technology scaling.

Prior works have enhanced energy efficiency from optimizations in different granularities of DNN accelerator spanning around dataflow, algorithm, memory, undervolting and so on [2], [4], [5], [10]–[14], which either reduce the number of computations, or skip unwanted computations or reduce the energy per computation. *However, our work is the first one that improves the energy efficiency of TPUs by saving leakage loss in the under-utilized SA resources which are idly waiting for computations.* Following are the key contributions of our work:

- We parametrize the cycle accurate activity and idleness profile of MAC units inside a TPU systolic array on different batch sizes of inputs (Section II-B).
- We establish that the MACs in TPU systolic array remain computationally idle for 40-90% of the time, depending on server or edge Inference applications (Section II-C).
- We propose UPTPU - a low overhead power-gating paradigm, adaptive of batch size, sleep transistor's tolerances and zero-weight computations (Section III).
- Combined with Zero-Skip [2], UPTPU offers a staggering $3.5\times - 6.5\times$ energy efficiency for eight DNN applications, with zero sacrifice on the performance and inference accuracy (Section V).

II. Motivation

In this section, we demonstrate the opportunities of drastically reducing the energy consumption in TPU systolic arrays. Section II-A provides a background and architectural overview of the TPU systolic array. Section II-B presents a rigorous mathematical analysis to parametrize the computation and dataflow pattern. Finally, Section II-C presents new insights in energy saving opportunities.

A. TPU Systolic Array

Matrix Multiplication is the most crucial part of computation in the inference phase of DNN applications. TPU hosts a weight stationary 256×256 systolic array of MAC units to perform Matrix multiplication between 8-bit-quantized activation inputs and pre-trained weights. Weights are fetched from weight FIFO and pre-loaded to each MAC unit. Unified Buffer stores the activation inputs, which are streamed into the corresponding row of the systolic array, to be multiplied by all the weights in the row. Partial sums move downstream, adding themselves to the multiplication outputs at each row. Activation matrix is transposed and sent to the the systolic array as a (systolic) diagonal wavefront, creating a predictable dataflow.

Figurative representation of the scaled down TPU systolic array dataflow can be seen in Figure 1. A 3×3 Activation matrix

978-1-6654-3274-0/21/$31.00

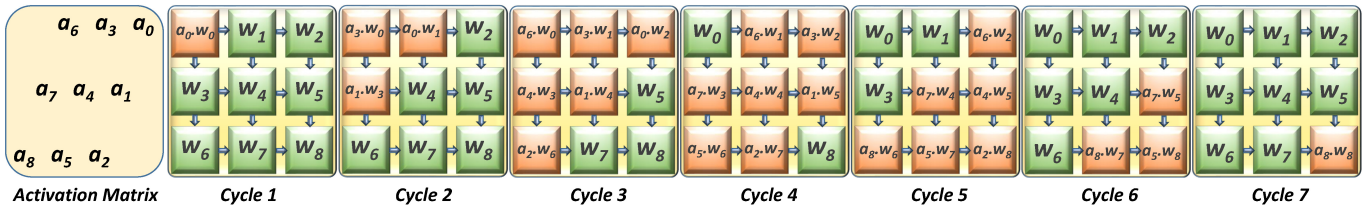

Fig. 1: *Cycle accurate representation of matrix multiplication between Activation and Weight matrices. Orange represents computationally active MAC unit (only multiplication shown for space constraints), whereas, green represents idle MAC unit, waiting for activation stream.*

($[a_0, .., a_8]$) is streaming into 3×3 systolic array, with Weight matrix ($[W_0, .., W_8]$) preloaded into it. We can see a distinct systolic pattern in the computation activity and idleness of a MAC with green and orange colors, respectively. Seeking a general and exhaustive outlook on the pattern, we present and analyze our accurate analytical model of the usage of hardware resources in Section II-B.

B. Mathematical Parametrization

In this section, we accurately parametrize the usage of hardware resources for a general $B \times N$ Activation matrix multiplied by $N \times N$ Weight matrix in an N-dimension systolic array. We define different metrics in Table I and illustrate them in Equations 1-3.

$\boldsymbol{T_C}$	Architectural lifetime (clock cycles) of the matrix multiplication of $B \times N$ activation and $N \times N$ weight matrix.
$\boldsymbol{U(n)}$	No. of computationally active MACs in n^{th} clock cycle.
$\boldsymbol{TRU}$	$\boldsymbol{True\ Resource\ Usage}$: Number of computationally active MACs over matrix multiplication lifecycle, T_C.
$\boldsymbol{MAR}$	$\boldsymbol{Maximum\ Available\ Resource}$: Maximum number of MACs ideally available for computation over T_C.
$\boldsymbol{RUR}$	$\boldsymbol{Resource\ Usage\ Ratio}$: Percentage of the MAC resources used for the matrix multiplication over T_C.

TABLE I: *Definitions of Metrics used for parametrization.*

Through our rigorous mathematical analysis, we find that $U(n)$ (Equation 2) can be described accurately with eight distinct quadratic and linear arithmetic sequences of n, as an artifact of varied dependence on B and N. Six regions (viz. $R_1 - R_3$ and $R_6 - R_8$ in Equation 2) annotate the rise and fall in the number of actively used MACs at the beginning and towards the end of T_c and two regions (viz. R_4 and R_5) describe the connection between the rising and falling regions. R_1 (and R_8) includes quadratic rise (fall) from (to) the minimum of a single active MAC . They are followed (and preceded) by linear regions R_2 (and R_7) for $B < N$ or quadratic regions R_3 (and R_6) for $B > N$. Finally, R_4 connects R_3 and R_6 for large batch sized Activation matrix ($B > 2N - 2$) with capping constant value (N^2) representing the usage of all possible MACs. And quadratic region R_5 connects regions $R1(R8)$ or $R2(R7)$ or $R3(R6)$ for medium B's ($2 < B < 2N - 2$).

TRU aggregates $U(n)$ over T_c. MAR amounts to all N^2 MAC units available for T_C clock cycles. Finally, RUR, reflecting the actual resource usage ratio manifests itself as a function of N and B (Equation 3). Next, we use this parametrization to better understand the hardware utilization scenario in TPU systolic array with N=256.

$$Total\ Computation\ Clock\ Cycles\ (T_C) = 2N + B - 2 \tag{1}$$

$$U(n) = \begin{cases} \frac{n^2}{2} + \frac{n}{2}, & R_1\ [1 \le n \le min(N,B)] \\ Bn - \frac{B(B-1)}{2}, & R_2\ [min(N,B) < n \le N] \\ -\frac{n^2}{2} + \frac{(4N-1)n}{2} - N(N-1), & R_3\ [N < n \le max(N, min(B, 2N-2))] \\ N^2, & R_4\ [max(N, min(B, 2N-2)) < n <= max(N,B)] \\ -\frac{n^2}{2} + (B + 2N - 1)n - \frac{B(B-1) + 2N(N-1)}{2}, & R_5\ [max(N,B) < n <= max(B, (N + min(N,B) - 2))] \\ -\frac{n^2}{2} + \frac{(2B-1)n}{2} - \frac{B(B-1) - 2N^2}{2}, & R_6\ [max(B, (N + min(N,B) - 2)) < n <= (T_C - N)] \\ -Bn + \frac{B^2}{2} + \frac{(4N-1)B}{2}, & R_7\ [(T_C - N) < n <= (T_C - min(N,B))] \\ \frac{n^2}{2} - \frac{(2B + 4N - 1)n}{2} + \frac{B^2 + B(4N-1) + 2N(2N-1)}{2}, & R_8\ [(T_C - min(N,B)) < n <= T_C] \end{cases} \tag{2}$$

$$TRU = \sum_{n=1}^{T_C} U(n) = N^2 \times B, \qquad MAR = N^2 \times T_C, \qquad RUR(\%) = \frac{TRU}{MAR} \times 100\% = \frac{100B}{(2N + B - 2)} \tag{3}$$

C. TPU Hardware Resource Utilization

Using Equation 2, we plot the distribution of computationally active MAC units, $U(n)$, in the TPU SA ($N = 256$), for different batch sized (B) activation matrices in Figure 2. We experimentally verify the correctness of this distribution on our systolic array simulator (Section IV). Following are the important observations from the figure. Number of active MACs reach the peak, only during the mid of the multiplication lifecycle. During the start and end

of the multiplication, maximum MACs are unused. We can only spatially use all the available 65536 MACs simultaneously (region R_4 in Equation 2), if we supply a batch (B) with more than 510 inputs, and we can then sustain that peak utilization period for the difference of $(510 - B)$ cycles. For any batch size, the number of active MACs at the beginning and the end of multiplication lifecycle follows a rise and fall leading to a massive underutilized hardware components. Furthermore, with smaller batch sizes, the distance from the maximum usage keeps on increasing to result an almost entirely unused SA.

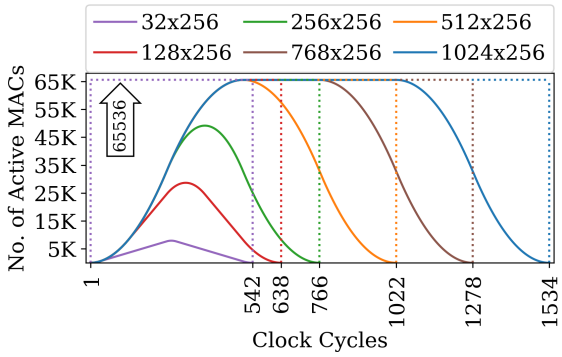

Fig. 2: *Distribution of computationally active MACs over all the clock cycles for different $B \times 256$ input matrices multiplied to 256×256 weight matrix. X-axis labels show the respective ends of T_c.*

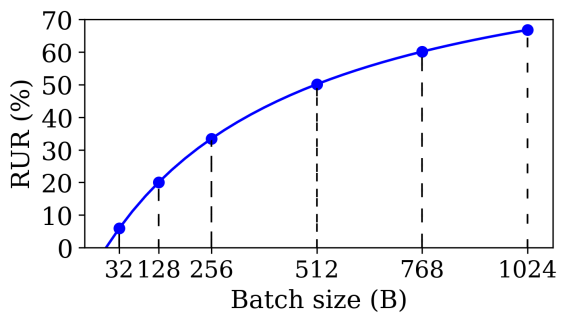

Fig. 3: *Resource Usage Ratio (%) for different batch sized input in TPU Matrix Multiplier Unit.*

Figure 3 plots the Resource Usage Ratio (RUR) (Equation 3 and Table I). It exhibits the stark dependence of the RUR on the batch size. We see that even for a batch size of $4\times$ the SA dimension, the SA resource usage is under 65%, while smaller batch sizes result in very poor resource utilization. There are practical limits to the batch size and thus the RUR, as outlined below.

- Very expensive high speed (typically on-chip SRAM) and higher size unified buffer is required to stream the large batch sizes of inputs and handle the consequent large output matrix.
- The inference decision from a batch of inputs can only be completed after matrix multiplications with different weights from all the layers of DNN. Hence, larger batch sizes introduce real time inference response latency [6]. *Consequently, although TPU can accommodate batch sizes upto 2048 from its 24 MiB SRAM, Google workloads' latency requirements limit the batch sizes to only around 30-200 ($RUR = 5 - 30\%$) [6].*
- Edge Inference applications are limited to handful of batches (as low as one) due to real time need of low response latencies, and are bound by energy and cost budget for streaming inputs and holding the outputs of larger batch sizes [15], [16].

This analysis points that any RUR less than 100% denotes that 100-RUR% MACs are consuming wasteful leakage energy while waiting for computation and holding the weight value. This scenario provides us with the unique opportunity to make the TPU highly energy efficient by capping the wastage energy, while not interfering with the computation throughput and accuracy. Section III details UPTPU, an intelligent power gating paradigm to carefully exploit this opportunity.

III. UPTPU Design

In this section, we present ***UPTPU**: **U**nderutilization based **P**ower-gating Paradigm for **TPU***, a low overhead design paradigm which curbs almost entire wasteful leakage energy coming from severely underutilized MAC units, through intelligent power gating. Section III-A presents the batch size aware gating control strategy. Section III-B delves into replacing the volatile storage units of MAC unit with NVMs. Finally, Section III-C discusses the circuit level specifics on the power gating design choices.

A. Power-Gating Control Strategy

1) Systolic Power Gating (SPG): Section II exposes the stark dependence of the underutilization of the TPU systolic array on batch size. The batch of activation inputs is supplied to the systolic array in a diagonal fashion so that the wave of the *activity* in MACs advance diagonally to use up the idle MACs one diagonal each clock cycle. After the clock cycles equal to the batch size, the wave of *idleness* advances one diagonal each cycle. The phenomenon can be seen in Figure 1 with orange (*activity*) and green (*idleness*) colors. With the advance knowledge of batch size, we can have an accurate assessment of which diagonal of MAC units are active and idle in any computation clock cycle. We assume that the software can provide batch size along with the data.

Figure 4 shows the design overview of UPTPU. Following the systolic-diagonal trend of activity and idleness, We have a $2N - 1$ bit Serial-In-Parallel-Out (SIPO) Right Shift Register (SR) whose bits are physically mapped onto $2N - 1$ diagonals of MACs. The parallel right-shifting bit values serve as systolic wake/sleep signals for each diagonal. We conceive a Batch Counter which will be loaded with the binary representation of $Batch\_Size - 1$, the value of which will stream $sleep\_bit$ and $wake\_bit$. Algorithm 1 and its respective description explain how we use the batch size information to map bits in the SR, while ensuring a *zero overhead in performance*.

2) Zero Weight Power Gating (ZWPG): We observe that significant amount of the computations in the MAC units for the practical DNN datasets see *'zero computations'* involving either zero activation or zero weight. Figure 6 shows the percentage distribution of *Zero Activation or Weight Computations (ZAWC)* (average 75%) and *Zero Weight Computations (ZWC)* (average 26%) for different DNN datasets. These zero computations can occur either naturally from the activation and trained weights or from zero padding of some activation and weight matrices to fit into the 256×256 TPU Systolic Array. However, only the MACs with zero weights are practical to be powergated as the weights remain static over the batch computation lifecycle, T_c (Equation 1) and the sleep transistors won't have to be woken up until T_c.

We propose *Zero Weight Power Gating (ZWPG)* to extend the applicability of the SPG sleep transistor to curb the energy consumptions from zero weight MACs. As seen in figure 4, the zero weight (zw) signal coming from weights stored in NVM (Section III-B) puts

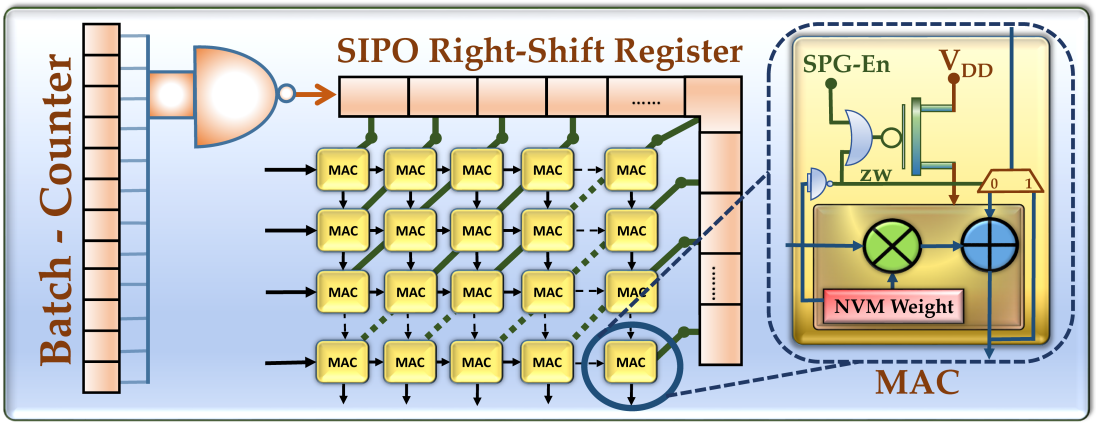

Fig. 4: *UPTPU design overview*

Algorithm 1 SPG Control Algorithm

1: $Batch\ Counter(BC) \leftarrow Batch\_Size - 1$
2: $Systolic\ Array\ Dimension(N) \leftarrow 256$
3: $T_w \leftarrow wake\_up\_tolerance$
4: $sleep\_bit \leftarrow 1,\ wake\_bit \leftarrow 0$
5: $SR[(2N-2).....(2N-2-T_w)] \leftarrow wake\_bit$
6: $SR[(2N-2-T_w).....0] \leftarrow sleep\_bit$
7: **while** $BC > 0$ **do**
8: $\quad SR \leftarrow SR \gg 1$
9: $\quad SR[2N-2] \leftarrow wake\_bit$
10: $\quad BC \leftarrow BC - 1$
11: **end while**
12: **while** $current\_batch$ **do**
13: $\quad SR \leftarrow SR \gg 1$
14: $\quad SR[2N-2] \leftarrow sleep\_bit$
15: **end while**

Line 3: T_w refers to the number of clock cycles alloted for the MAC diagonals to completely power up from sleep-state and be ready for error free computation. This enables a **zero performance (TOPS) overhead.**

Lines 5-6: SR is initialized to T_w $wake\_bits$ followed entirely by $sleep\_bits$. Right Shifting of SR along with these T_w $wake\_bits$ ensures that **all the idle MAC diagonals will start switching on T_w clock cycles in advance of the computation scheduled in that diagonal.**

Lines 7-11: SR shifts right by injecting $wake\_bit$ to the MSB, **waking-up only one additional diagonal at a time**, until BC counts to zero.

Lines 12-15: After BC cycles, SR starts shifting right by injecting $sleep\_bit$ at the MSB, **sleeping only one additional diagonal at a time.**

the MAC unit to sleep irrespective of SPG-En signal. Unlike a MAC gated with SPG, a MAC gated with ZWPG should be able to route the upstream data through itself. A demux is used to route the upstream MAC's data by bypassing the MAC powergated with ZWPG.

B. Usage of NVMs

As each MAC unit stores a weight value in its associated volatile SRAM cells, we need to preserve the weight values during power-gating for seamless computation on wake-up. Classic powergating employs retention cells, which call for further leakage [7], given that there is a SRAM register in each of the 256×256 MACs. We envision the replacement of leaky weight holding SRAM cells with non-leaky STT-MRAM Non-Volatile Memory (NVM) to solve both the hurdles of volatility and memory leakage. STT-MRAMs also provide other compelling advantages for the niche of weight stationary systolic arrays. STT-MRAMs boast about $20\times$ reduction in leakage energy and about $4\times$ increment in packing density with respect to SRAMs and they have comparable read characteristics to SRAMs [17].

The unique write pattern in TPU systolic array facilitates the adoption of STT-MRAMs although they suffer from a $3\times$ write speed and $20\times$ energy penalty for writes [17]. The weights in the MACs are only written once for an entire batch of computation, which gives the delay overhead of less than 0.5%, even for the smallest of batch sizes. More importantly, the spread of $20\times$ savings in leakage power over the batch computation lifecycle, diminishes the once-per-batch $20\times$ write-energy-increase. Thus, usage of NVM overturns the otherwise overkill in both energy and area consumption coming from SRAMs, and also facilitates the sleep of entire MAC unit without retention leakage. In attempt to replace the SRAMs with STTMRAMs in CPU/GPU caches, researchers have compensated the write penalties by packing manyfold MRAMs in the same area footprint of SRAM cache [17], [18]. The area savings with STT-MRAM contributes to amortizing the area overhead due to sleep transistors.

C. Circuit Level Considerations for Power-Gating

Sleep transistor design is a challenging VLSI domain because of the difficulty in optimization around its various effects on design performance, area, routability, overall power dissipation, and signal/power integrity [9]. We conceive one sleep transistor per MAC, which receives a per-diagonal power gating control signal. Although it might seem intuitive to house one sleep transistor per diagonal, per-MAC strategy eliminates several circuit-level complications. As the diagonals represent diverse switching loads pertaining to the varying number of connected MACs (1-256), design of graded sleep transistors adds huge design complexity. In addition, the usage of sleep transistor at the granularity of a MAC facilitates ZWPG (Section III-A2) and eliminates the need of having bulkier power lines running through each diagonal, ultimately improving routability.

As the sleep and wake-up happens at a granularity of just one diagonal at a time during computation (Algorithm 1), noise and current crowding issues are minimized and the average sleep time is maximized. Moreover, we favor the decrease in area penalty, noise, and the high power-on rush current by compromising the switching speed of the sleep transistor. The system wide performance is not

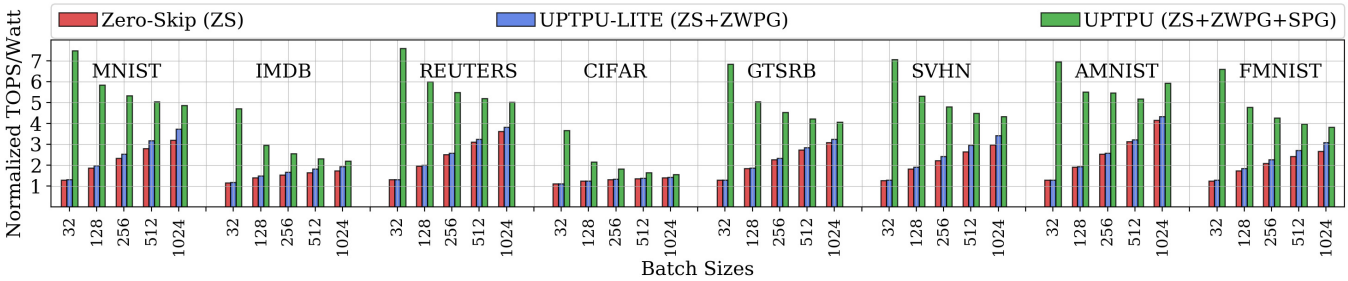

Fig. 5: *Normalized TOPS/Watt of eight DNN datasets computed on a TPU systolic array with different batch sizes brought about by the comparative schemes.*

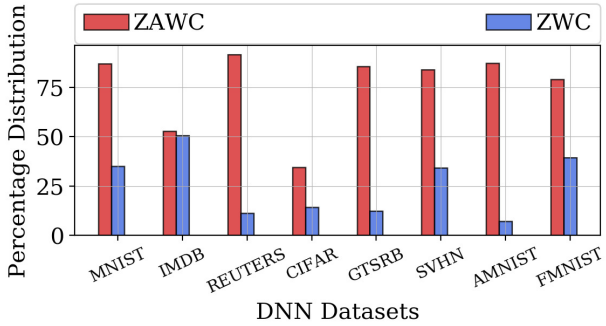

Fig. 6: *Zero Activation or Weight Computations (ZAWC) and Zero Weight Computations (ZWC) expressed as percentage of total computations for different DNN datasets.*

affected by slower sleep transistors because of the wake-up tolerance included in the gating control strategy (T_w in Algorithm 1). The 6% area overhead of PMOS sleep transistors [8], combined with the overheads from control hardware, dilutes to only around 3.4% area overhead with respect to the entire TPU die.

IV. Methodology

We use our cycle accurate TPU systolic array simulator built upon [19], with architectural details from [6], as an architectural simulator for the cycle accurate assessment of computation data and resource utilization pattern. First, we train eight DNN applications (viz., MNIST [20] , Reuters [21] , CIFAR-10 [22] , IMDB [23] , SVHN [24] , GTSRB [25] , FMNIST [26] , FSDD (Audio-MNIST) [27]) using Keras with TensorFlow back-end and extract the weights from the trained model. We stream the 8-bit quantized activation input from the datasets in several batch sizes to the simulator to be multiplied with the weight matrices stored in SA. The output matrices from the simulator are combined to evaluate the inference accuracy.

We develop the energy efficiency model by conjoining the architectural outcomes of the datasets with estimations of dynamic and leakage energy from CAD tools. We synthesize the RTL description of SA MAC units with different design augmentations, through Synopsys Design Compiler followed by place and route through Cadence SoC Encounter using 45nm standard cell library, to estimate the area and energy (dynamic and leakage) consumption and associated overheads. We find and use the leakage energy to be 20% of the dynamic energy. The wake-up tolerance (T_w in Algorithm 1) is set to three clock cycles, inline with the prior power gate implementations [8], [9], [28] . The switching energy overhead is embedded in the model with break even clock cycles, as suggested by [28].

V. Experimental Results

In this section, we evaluate the efficacy of different schemes on increasing the energy efficiency of a 256×256 TPU systolic array. Section V-A presents the comparative schemes. Section V-B compares and describes the energy efficiency coming from different schemes.

A. Comparative Schemes

- **Zero-Skip (ZS):** This is a widely used technique for drastically improving the energy efficiency of DNN Accelerators [2], [29], where the computation in MAC is entirely skipped if activation input or weight is equal to zero. Zero skipping gets rid of the dynamic energy for those MAC units which hold zero weight or receive zero activation.
- **UPTPU-LITE:** This is an extension to ZS, with application of *Z*ero *W*eight *P*ower *G*ating (ZWPG). All the MAC units holding the weight value of zero are power gated for the computation lifecycle of a batch of activation inputs. In addition to the dynamic energy savings from ZS, this scheme prevents the leakage power from the zero weight holding MACs.
- **UPTPU:** UPTPU includes the Systolic Power Gating (SPG) of unutilized MAC units, in addition to the benefits provided by UPTPU-LITE. It intelligently powergates almost all the idle MAC units arising from TPU underutilization on different batch sizes.

B. Interpretation of Energy Efficiency

We simulate the gains in energy efficiency for eight DNN datasets, when the computation is performed in different batch sizes. Figure 5 presents the gain in Tera Operations Per Second per Watt (TOPS/Watt) normalized with base TPU SA for eight DNN datasets, for different comparative schemes. Figure 6 presents the batch-size independent *Z*ero *A*ctivation or *W*eight *C*omputations (ZAWC) and *Z*ero *W*eight *C*omputations (ZWC) among the total MAC computations pertinent to the ZS and ZWPG schemes respectively. We see various trends in energy efficiency gains for different datasets and schemes. In general, the maximum average gain for any dataset (Figure 5) is dictated by the percentage of ZAWC (Figure 6). Higher ZAWC gives many opportunities for ZS embedded in all comparative schemes. The datasets with relatively lower ZAWC (viz. IMDB and CIFAR) have relatively lower energy efficiency gains.

We see a minimal benefit in UPTPU-LITE (ZS+ZWPG) in comparison to ZS, as the extra ZWPG scheme adds the small additional leakage savings coming from the small subset (ZWC-Figure 6) of dynamically skipped MACs. The relatively smaller subsets (viz. REUTERS, AMNIST, GTSRB) result in minimal benefit addition to gains. However, more importantly, the gains from UPTPU-LITE (ZS+ZWPG) decrease for lower batch sizes. As the RUR decreases

with lower batch sizes (Section II), the constant benefits coming from ZS and ZWPG are progressively diluted by the increasing leakage energy consumption in unutilized MACs.

Finally, UPTPU (ZS+ZWPG+SPG) is able to achieve much higher gains, because of the addition of Systolic Power Gating (SPG) which intelligently power gates the unutilized MACs. In addition to higher average gain, we also get a complementing effect to ZS and ZWPG, pronounced by the increase of the energy efficiency with the decrease in the batch size. As the batch sizes decrease, SPG gets increasing opportunities from decreasing RUR to give massive gain in TOPS/Watt. UPTPU achieves, on a average of $3.5 \times -6.5\times$ gain in TOPS/Watt for batch sizes $1024 - 32$. This shows that UPTPU can achieve staggering energy efficiency gains throughout the range of both highest and lowest ends of the batch sizes. The performance and inference accuracy is not compromised at all, because of our dataflow adaptive intelligent power gating (Algorithm 1).

VI. Related Work

Improvement of energy efficiency in DNN accelerators has been explored in several ways. Chen et al. proposes an energy efficient Row-Stationary dataflow to optimize the data movement inside a deep neural network (DNN) [3]. Reagen et al. outlines automated methods to extract collective energy efficiency from each of the algorithm, architecture, and circuit layers [2]. Researches have also explored voltage underscaling to increase energy efficiency. Zhang et al. underscaled the supply voltage and tackled the timing errors by skipping the errant computations [10]. Power-gating the unutilized sections of computing systems have been extensively explored. Tschanz et al. have demonstrated the efficacy of a PMOS sleep transistor in reducing the power consumption by 8% in 130-nm node microprocessor [8]. Shi et al. outline the challenges and opportunities in optimal sleep transistor design in different configurations [9]. Hu et al. have provided extensive analytical model of the idleness in CPU [28]. *However, our work is the first one in the DNN accelerator domain to explore the severe, yet predictable resource underutilization and propose power-gating strategies to extract a staggering gain in energy efficiency.*

VII. Conclusion

The upsurge in the AI based economies around the globe and the consequent growth of DNN workloads are demanding highly energy efficient DNN accelerators both at the server and the edge. This paper parametrizes a huge hardware underutilization problem in the weight stationary systolic array, and provides a staggering energy efficiency to TPU by solving the problem through intelligent power-gating. Our scheme can be superimposed on top of other existing architectural or circuit level techniques to inflate the energy efficiency, without any compromise in the inference accuracy or performance. More generally, due to a predictable data-flow pattern in the AI workload, this work opens up newer avenues for exploration of power-gating based energy efficient solutions for all forms of AI accelerators.

References

[1] "Ai will add 15 trillion to the world economy by 2030," https://www.forbes.com/sites/greatspeculations/2019/02/25/ai-will-add-15-trillion-to-the-world-economy-by-2030/, 2019.

[2] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 267–278.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[4] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. S. Sathe, "Energy-efficient neural network acceleration in the presence of bit-level memory errors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, no. 99, pp. 1–14, 2018.

[5] V. Gokhale, A. Zaidy, A. X. M. Chang, and E. Culurciello, "Snowflake: An efficient hardware accelerator for convolutional neural networks," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[6] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, 2017, pp. 1–12.

[7] Y.Wang, S.Roy, and N.Ranganathan, "Run-time power-gating in caches of gpus for leakage energy savings," in *Proc. of DATE*, March 2012.

[8] J. Tschanz, S. Narendra, Y. Ye, B. Bloechel, S. Borkar, and V. De, "Dynamic-sleep transistor and body bias for active leakage power control of microprocessors," in *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, Feb 2003, pp. 102–481 vol.1.

[9] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Proc. of DAC*, 2006, pp. 113–116.

[10] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep neural network accelerators," *arXiv preprint arXiv:1802.03806*, 2018.

[11] P. Pandey, P. Basu, K. Chakraborty, and S. Roy, "Greentpu: Improving timing error resilience of a near-threshold tensor processing unit," in *Proc. of DAC*, 2019, pp. 173:1–173:6.

[12] N. D. Gundi, T. Shabanian, P. Basu, P. Pandey, S. Roy, K. Chakraborty, and Z. Zhang, "Effort: Enhancing energy efficiency and error resilience of a near-threshold tensor processing unit," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 241–246.

[13] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, p. 696–701.

[14] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 92–104.

[15] J. Hanhirova, T. Kamaainen, S. Seppaa, M. Siekkinen, V. Hirvisalo, and A. Yla-Jaaski, "Latency and throughput characterization of convolutional neural networks for mobile computer vision," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18, 2018, p. 204–215.

[16] Z. Jiang, "Efficient deep learning inference on edge devices," in *SysML COnference*, 2018.

[17] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3d stacking magnetic ram (mram) as a universal memory replacement," in *2008 45th ACM/IEEE Design Automation Conference*, June 2008, pp. 554–559.

[18] J. Zhang, M. Jung, and M. Kandemir, "Fuse: Fusing stt-mram into gpus to alleviate off-chip memory access overheads," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2019, pp. 426–439.

[19] "Ucsb archlab opentpu project," https://github.com/UCSBarchlab/OpenTPU.

[20] Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010.

[21] "Reuters-21578 dataset," http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html, 2021.

[22] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis." Association for Computational Linguistics, 2011, pp. 142–150.

[24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[25] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[26] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[27] "Free spoken digit dataset (fsdd)," https://github.com/Jakobovski/free-spoken-digit-dataset, 2021.

[28] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *ISLPED*, 2004, pp. 32–37.

[29] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016.