The Pennsylvania State University
The Graduate School

A SCHEDULING FRAMEWORK FOR DECOMPOSABLE KERNELS ON ENERGY HARVESTING IOT EDGE NODES

A Thesis in
Computer Science and Engineering
by
Sethu Jose

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2022

The thesis of Sethu Jose was reviewed and approved by the following:

Mahmut Taylan Kandemir
Distinguished Professor of Computer Science and Engineering
Thesis Advisor

John Sampson
Associate Professor of Computer Science and Engineering

Vijaykrishnan Narayanan
A. Robert Noll Chair Professor of Computer Science and Engineering and Electrical Engineering

Chita R. Das
Distinguished Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

Abstract

With the growing popularity of the Internet of Things (IoTs), emerging applications demand that edge nodes provide higher computational capabilities and long operation times while requiring minimal maintenance. Ambient energy harvesting is a promising alternative to batteries, but only if the hardware and software are optimized for the intermittent nature of the power source. At the same time, many compute tasks in IoT workloads involve executing decomposable kernels that may have application-dependent accuracy requirements. In this work, we introduce a hardware-software optimization framework for such kernels that aims to achieve maximum forward progress while running on energy harvesting Non-Volatile Processors (NVP). Using this framework, we develop an FFT and a convolution accelerator that computes 3.7x faster, while consuming 5.4x less energy, compared to a baseline energy-harvesting system. With our accuracy-aware scheduling strategy, approximate computing in this framework delivers on average 6.6x energy reduction and 4x speedup by sacrificing minimal average accuracy of 6.9%.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. 1822923.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

This work has been accepted in the proceedings of the 2022 on Great Lakes Symposium on VLSI (GLSVLSI 2022).

Chapter 1 Introduction

To match battery life with deployment longevity needs, the processing demands of many IoT devices are limited to data sampling and transmission to a centralized server where most of the computations are performed [1]. Augmenting IoT devices with ambient energy harvesting does not only reduce servicing requirements, but can also enable greater local computation [28]. Ultra low power processors with energy harvesting are appealing for applications in long-term environmental monitoring [16], medical devices [12], smart accessories [10], computer vision [22], wildlife tracking [27], and machine health monitoring (MHM) [4, 15].

Ambient energy sources, however, are fickle. Moreover, the efficiency of energy conversion from these sources is relatively low. Non-Volatile Processors (NVPs), with their state retention characteristics, are designed to work competently in these scenarios. Depending on the depth of integration of the nonvolatile elements with the datapath logic, NVPs can achieve 1,000× improved backup and restore speeds compared to non-volatility enabled traditional processors [29]. However, an energy-optimized NVP system alone is *not* capable of meeting the energy constrained workload demands of today [20]. The lack of workload-optimized hardware and intelligent scheduling policies limit their efficiency on compute-heavy tasks.

To address these challenges, our work proposes an edge node accelerator architecture and its scheduling policies that dynamically tune workloads to the intermittent nature of the energy source. Our scalable system is designed as an auxiliary co-processor, powered by energy harvesting, that augments a baseline NVP. This work makes the following contributions:

- **Decomposition of IoT workloads**: Our accelerator targets commonly used IoT workloads like FFT, convolution and matrix multiplication which can be *decomposed* into smaller compute and merge tasks. Using a general hardware-software

optimization framework, we show that availability of many such decomposed tasks presents many candidates for scheduling, there by tuning the overall workload to the instantaneous energy harvested.

- **Scheduling strategies for decomposable workloads**: We introduce *greedy* scheduling strategies (ONLY_ONE and N_OF_EACH) which selects from decomposed tasks the ones that best match the instantaneous system energy. Greedy even discards already computed partial results to achieve overall improved performance. *Predictive* strategy schedules on speculated future energy availability. Compared to a generic NVP system which sets capacitor-constrained task boundaries and executes tasks in sequential program order (conservative strategy), our system achieves upto 3.7x speedup while consuming 5.4x less energy.

- **Approximate Computing**: We augment the system with approximate computing where trading accuracy for efficiency is viable. We discuss *Fully Approximated Scheduling* (FAS) which is agnostic to application accuracy requirements, and *Selective Approximate Scheduling* (SAS) which statically disables lower precisions to minimize output errors. We improve on this by introducing *Accuracy-Aware Scheduling strategy* (AAS) which dynamically identifies accuracy-sensitive sensor data phases for high precision computation. With real-world applications (ECG [6], sound monitoring [24], noise filtering, sleep-apnea detection [23], MHM [4] and saliency detection [14]), AAS achieves on average 6.6x energy reduction and 4x speedup with average accuracy loss of 6.9%.

Chapter 2
Background

2.0.1 NVP System

Fig. 2.1 shows the basic architecture of an energy harvesting Non-Volatile IoT device. Typical ambient energy sources include solar, RF, piezoelectric, and thermal-electric [17–19]. Transducers, like photo-voltaic cells, RF antennas, thermocouples, or piezoelectric sensors, generate electricity from these sources. The harvested energy is stored in a capacitor that meets the energy requirement of the single largest uninterruptible action that needs to be powered. A DC-DC converter steps up the capacitor voltage and powers the system. NVP with its non-volatile architecture retains system state across power outages.

NVPs switch through ON/OFF states depending on the intermittent input power. The workloads are therefore divided into tasks such that the amount of energy available from the capacitor is sufficient for one uninterruptible burst of task execution. These tasks are then executed in the program order relative to each other depending on energy availability.

The intermittent and varying nature of harvested energy dictates that these systems

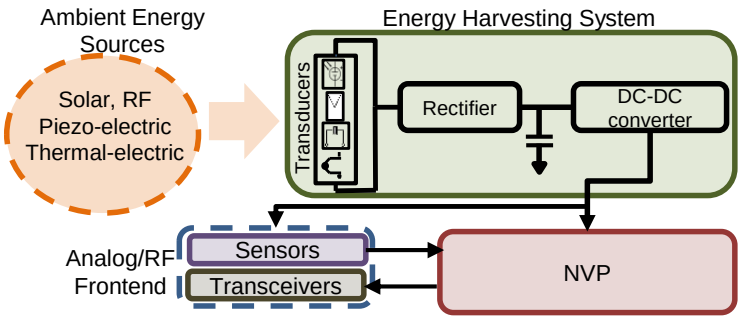

Figure 2.1. IoT edge node architecture.

adopt execution strategies favoring *partial completion of results*. In some scenarios, partial results can produce well-defined intermediate results allowing subsequent computations to easily consume them. Divide-and-conquer algorithms fit this pattern. These algorithms recursively break down the input problem into sub-problems of different sizes but of similar type until they become simple enough to be solved directly. The partial results are then merged to obtain the final result. Decomposition of workloads produce many sub-tasks of varying energy requirements, many of which are not dependent on other sub-tasks. Hence, they favor parallel execution and are more tuned to the intermittent nature of the energy source.

2.0.2 Decomposable Compute Kernels

A compute kernel is a unit of mapping of work to an accelerator that performs a pre-defined computation. FFT [4, 6, 23], convolution [24], and matrix multiplication [22] are among the most frequently used kernels in IoT devices. A common characteristic of these kernels is that they are *decomposable*. Decomposable kernels have computation that can be broken down to simpler sub-problems using the divide and conquer approach. For example, if a problem of size n is decomposable into a sub-problems each of size n/b, solving these a sub-problems and merging their results give the final solution to the original problem of size n more efficiently.

Below, we model the kernels FFT, convolution and matrix multiplication using the divide-and-conquer strategy to tune them for our architecture.

2.0.2.1 Loop Tiling for Matrix Multiplication

In divide-and-conquer based matrix multiplication, $C = A \times B$, A, B, C are $n \times n$ matrices and n is a power of 2. Each divide step generates four $\frac{n}{2} \times \frac{n}{2}$ sub-matrices. The algorithm iteratively divides till simpler sub-problem sizes are reached. The partial results are then merged to get the result matrix C.

2.0.2.2 Convolution/Polynomial Multiplication

To calculate convolution (CONV) sum, signals A and B are partitioned into two halves $\{A_0, A_1\}$ and $\{B_0, B_1\}$, respectively. The sum C is obtained by merging results of four $\frac{N}{2}$ sized sub-problems.

$$C = A * B = (A_0 + A_1 * x^{n/2}) * (B_0 + B_1 * x^{n/2}). \tag{2.1}$$

If $X = (A_0 + A_1)(B_0 + B_1), Y = A_0 * B_0, Z = A_1 * B_1$, then Equation 2.2 reduces the number of multiplications:

$$C = A * B = Y + (X - Y - Z) * x^{n/2} + Z * x^n. \tag{2.2}$$

2.0.2.3 Fast Fourier Transform (FFT)

Radix-2 Cooley-Tukey FFT is a widely used DFT algorithm. The algorithm splits the input sequence, $x(0), x(1) \ldots x(N-1)$, into two separate sub-sequences, x_{even} and x_{odd}, which contain only the even and odd indexed inputs, respectively. An $\frac{N}{2}$-FFT algorithm generates Fourier transforms X_{even} and X_{odd} as follows:

$$x_{\text{even}} \xrightarrow{\mathscr{F}} X_{\text{even}} = [X_0, X_2, X_4, \ldots, X_{\text{N-2}}] \tag{2.3}$$

$$x_{\text{odd}} \xrightarrow{\mathscr{F}} X_{\text{odd}} = [X_1, X_3, X_5, \ldots, X_{\text{N-1}}] \tag{2.4}$$

X_{even} and X_{odd} are then *merged* to obtain the N-FFT result:

$$X_{\text{k}} = X_{\text{even}} + W_N^k \times X_{\text{odd}} \tag{2.5}$$

where $k = 0, 1, \ldots, N-1$. $W_N^k = e^{-j\frac{2\pi n}{N}}$ is twiddle factor.

In this paper, we focus on the FFT and CONV workloads. Our system can be easily extended to support any decomposable kernel as discussed in Section 5.

2.0.3 Approximate Computing

Applications running on energy harvesting devices are often amenable to approximation [9] as a means to achieve forward progress even in energy-insufficient environments. However, the levels of approximations feasible are entirely application dependent. Consider the following case studies:

Machine health monitoring (MHM): Fig. 2.2 shows results from our implementation of Boudiaf et al's [4] MHM system. The FFT1024 spectrum shows that fault detection (peak at 3.6KHz) deteriorates as precision reduces to 4 bits.

Saliency detection: Saliency Detection is used in computer vision to identify conspicuous regions in an image. Fig. 2.3 shows the effect of approximation on saliency system by Itti et al. [14]. Clearly, the 1-bit saliency detection fails.

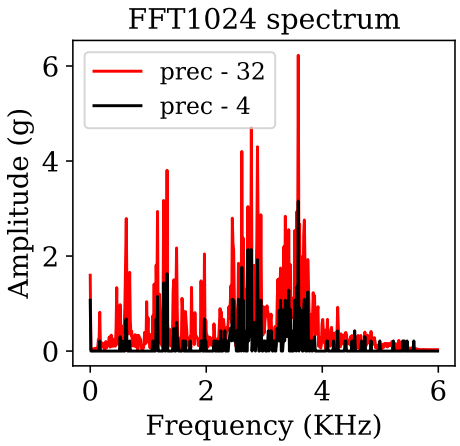

Figure 2.2. Impact of approximation on vibration detection spectrum - MHM.

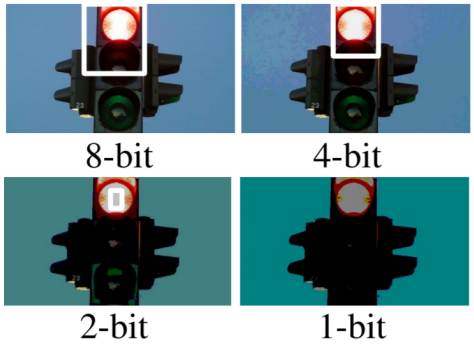

Figure 2.3. Impact of approximation on saliency detection.

2.0.4 Gaussian Mixture Error Estimation (GMEE) [11]

The propagation of errors through our system follows Gaussian Mixture Error Estimation (GMEE) proposed by Ghasemazar et al. [11]. Approximation errors, though not confined to small range, follow Gaussian distribution [11]. GMEE propagates statistical input distributions represented as probability mass functions (PMFs) through the approximated circuit to estimate final error metrics. PMF of an input of N instances is as follows:

$$PMF = \sum_{i=1}^{N} w_i \mathcal{N}(\mu_i, \sigma_i^2) \tag{2.6}$$

w_i, μ_i, σ_i represents the weight, mean and standard deviation of the input instance i. GMEE propagates expected input distributions through the approximated module keeping track of both correct and incorrect value distributions.

Consider a module with inputs A and B. In the non-approximated version of this

module, the correct output PMF, $cPMF_{out}$ is obtained as follows:

$$cPMF_{non\_approx\_out} = cPMF_A * cPMF_B \tag{2.7}$$

Here, $cPMF_A$ and $cPMF_B$ are the correct PMFs of inputs A and B respectively. If these inputs are erroneous, error output PMF of this non-approximated module, $ePMF_{out}$ is obtained as follows:

$$ePMF_{non\_approx\_out} = ePMF_A * cPMF_B + cPMF_A * ePMF_B + ePMF_A * ePMF_B \tag{2.8}$$

In addition to propagating errors, an approximated module introduces approximation errors into the output. The approximated module can be modeled as a combination of correct filter function, *cFF*, and an error filter function, *eFF*. These filters when convolved with output of non-approximated module generates the cPMF and ePMF as shown below:

$$cPMF_{approx\_out} = cFF * cPMF_{non\_approx\_out} \tag{2.9}$$

$$ePMF_{approx\_out} = cFF * ePMF_{non\_approx\_out} + eFF * ePMF_{non\_approx\_out} \tag{2.10}$$

Chapter 3
System Architecture

A divide-and-conquer based NVP acclerator should support different computational granularities. In certain computations, like convolution and matrix multiplication, the fundamental operation and its associated acceleration hardware, the Multiply-Accumulate block, can scale via simple replication for different input dimensions. In more complex kernels, like FFT, supporting different input sizes requires size-specialized hardware modules to support piece-wise execution.

Fig. 3.1 shows the proposed accelerator architecture interfaced with the energy harvesting system and a baseline NVP for scheduling workload. MCU initializes accelerator's Config NVM with energy and timing configurations of different kernel dimensions that are needed during run-time. Once the sensed input data is loaded into source-NVM, MCU can enter deep sleep state. The accelerator consists of dedicated *compute* and *merge* modules for each kernel dimension. For an input size N, an N dimensional (N-dim) compute module performs direct computation, while the N-dim merge module merges two $\frac{N}{2}$-dim results. Each module has a simple pre-processor that identifies sensor data phases for accuracy-aware scheduling. The modules themselves are volatile. To achieve non-volatility by retaining partial results across power outages, each compute/merge module is preceded and succeeded by a *source-NVM* and a *destination-NVM*, respectively. *source-NVM* retains the input to the module. *destination-NVM* stores the results. Since computation proceeds in a recursive fashion, a *destination-NVM* becomes the *source-NVM* for the next module.

If energy is insufficient to compute N-dim results directly, a lower granularity $\frac{N}{2^k}$, within the power budget, is invoked. On a module invocation, the compute block loads input from *source-NVM*, runs the computation and then writes the results to *destination-NVM*. Each block is invoked only when system energy is sufficient to complete one *read-compute-write* sequence. This ensures that partial results are retained across power

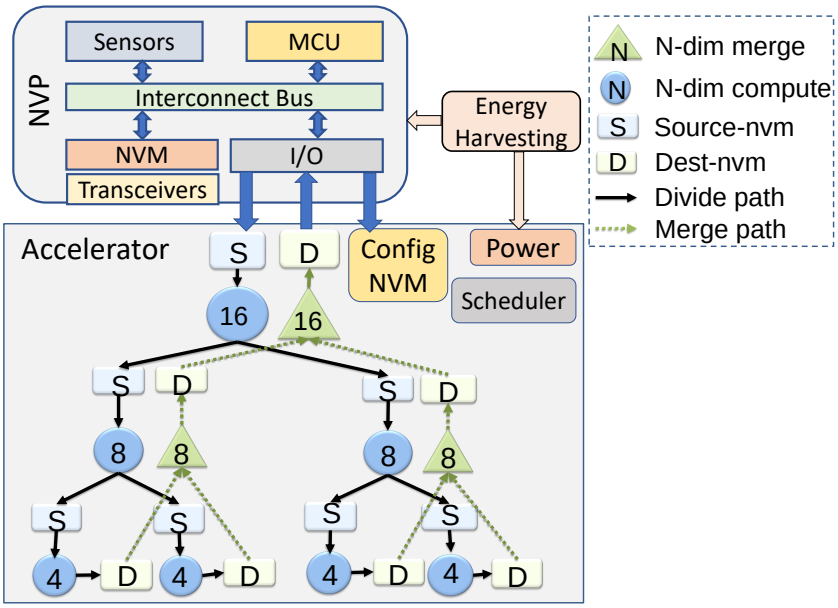

Figure 3.1. High-level view of decomposable architecture.

outages.

Fig. 3.1 shows an example execution flow of a 16-dim FFT computation. The input is initially stored in the 16-dim source-NVM by the MCU. If system energy allows, workload is directly executed on the 16-dim compute module. Else, the workload is decomposed into two 8-dim kernels and assigned to the two 8-dim compute modules. If the system energy is insufficient to execute 8-dim kernels as well, they are further decomposed to four 4-dim kernels. In the system, 4-dim modules are chosen as the smallest unit of work. Hence, the system waits for the 4-dim kernels to finish their execution. Based on energy availability, 8-dim merge modules merge their results. Further, the two 8-dim results are merged using 16-dim merge module to generate the FFT16 results which can then be read by the MCU.

Compute modules of FFT/CONV for power-of-2 dimensions 4–1024 are adequate for most practical systems. Larger dimension results can be computed by merging the results from smaller dimensions as discussed in Section 2.

Chapter 4
Scheduling Strategies

In this section, we discuss performance- and energy-efficient scheduling strategies for our decomposable architecture.

4.0.1 Conservative Strategy (CNSV) [7, 21]

In the *CNSV* strategy, the system always computes N-dimensional (N-dim) results directly, without decomposing and exploiting parallelism. Fig 4.1 and Fig 4.2 shows the CNSV execution of FFT32 and CONV16 kernels. The workload may also be broken down into sequential sub-tasks, such that each sub-task can be powered from the the energy buffering capacitor. Since the subtasks are dependent on each other, the system proceeds by executing these sub-tasks one after the other waiting in between for enough energy to accumulate [7, 21]. The idle wait time of CNSV varies according to the input dimensionality. Moreover, these systems are agnostic to the possibility of parallelism through kernel decomposition as they are restricted to program order of sub-task execution.

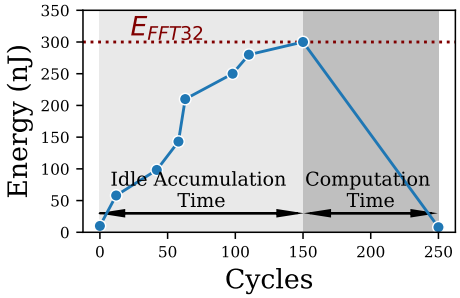

Figure 4.1. CNSV Strategy - FFT.

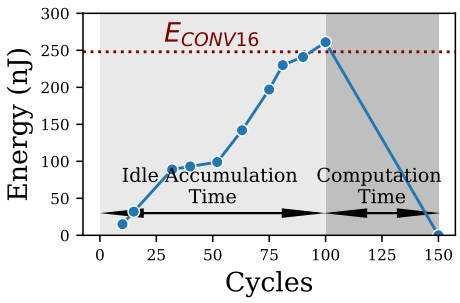

Figure 4.2. CNSV Strategy - CONV.

4.0.2 Greedy Strategies

The greedy strategy strives for *maximum forward progress possible with the instantaneous system energy.* Here, the workload is decomposed into smaller units of work, called kernels, that require less energy. These compute kernels are independent of each other and can be executed whenever their energy constraint is met. Hence, the hardware remains idle only till sufficient energy has accumulated to run any of these kernels. In our FFT/CONV system, kernel dimension 4 is chosen as the smallest unit of work. Consequently, the system can make forward progress if system energy, E_{sys}, crosses at least the E_{FFT4}/E_{CONV4} energy threshold. If E_{sys} is large enough, the system executes the largest kernel that is still within the E_{sys} limit. The greedy strategy is further classified based on the number of kernels allowed to execute *in parallel.*

4.0.2.1 ONLY_ONE

Here, only one kernel is runnable at a time. Fig. 4.3 illustrates an FFT32 computation using ONLY_ONE. Kernels are represented as nodes in the graph. At time ❶, system energy, E_{sys}, is just over the energy requirement of an FFT8 computation. The runnable nodes are 8_{LL}, 8_{LR}, 8_{RL}, 8_{RR}, and all the eight FFT4 nodes. The scheduler picks any of the highest energy requiring nodes, say 8_{LL}, and runs it to completion. At time ❷, $E_{sys} > E_{FFT4}$. Hence, node 4_{LRL} is executed. The same is again true at time ❸. Hence, node 4_{LRR} is executed.

At ❹, where $E_{sys} > E_{FFT16}$, 16_L and 16_R are the two runnable nodes. Of these, 16_L is identified as a partially completed node. Hence, 16_R is selected for execution as it has more pending sub-nodes than 16_L. At ❺, we have $E_{sys} > E_{16}$. At this point, the system discards the completed works of nodes 4_{LRL}, 4_{LRR} and 8_{LL} for the greater good by executing 16_L. At ❻, results of nodes 16_L and 16_R are merged. Finally, at ❻, FFT32

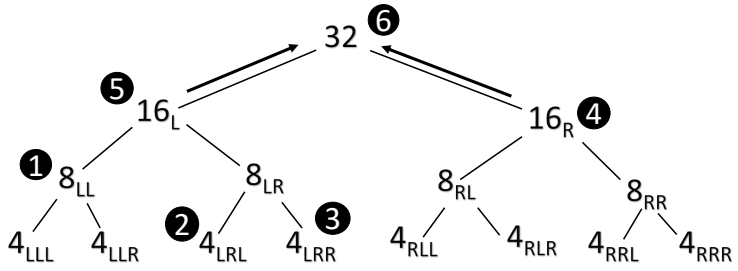

Figure 4.3. ONLY_ONE Greedy strategy on FFT32.

finishes its execution.

Greedy strategy prioritizes work based on instantaneous energy. It mitigates the limitations of sequential execution in CNSV strategy by providing more kernel options to choose from at every schedule point. The possibility that the sub-node execution can be marked redundant in the future means that the system is more *adaptive to variations in the ambient energy*. For example, at ❺, if the system did not have enough energy to complete 16_{L}, it would still make incremental progress by executing 8_{LR} and then merging the results.

Algorithm 1 explains the ONLY_ONE greedy strategy. At every scheduling point, the system picks the most energy requiring nodes each of which is executable using the current system energy $\mathrm{E_{sys}}$. The set of maximum energy nodes is stored in *maxNodes*. If there are multiple such nodes (each with same energy requirement), *maxNodes* will have all of them listed in it. In this scenario, the node with the highest pending sub-node energy is assigned to *nextNode* and then scheduled. On the other hand, if there is only one node in *maxNodes*, it is scheduled. With this strategy, the system tries to achieve the maximum forward progress within the least amount of time.

Algorithm 1 Greedy - only one

```
function SCHEDULER(E_sys)
    if nodes are pending then
        runnables ← nodes with energy < E_sys
        maxNodes ← get max energy node from runnables
        if maxNodes.count >1 then
            //multiple nodes with same max energy.
            nextNode ← least complete from maxNodes
        else
            //only one max energy node
            nextNode ← maxNodes
        schedule(nextNode)
```

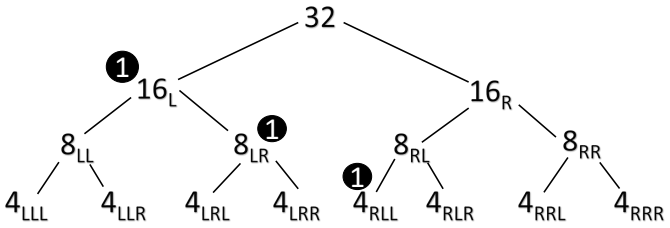

Figure 4.4. Running 1_OF_EACH greedy strategy on FFT32.

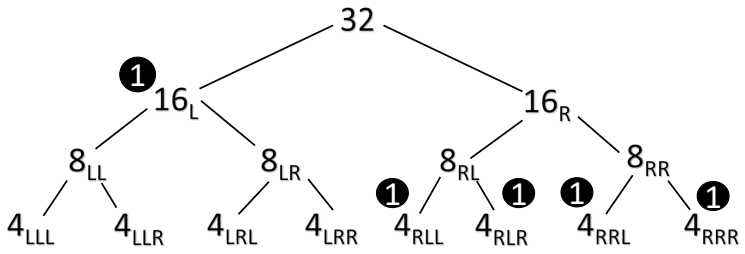

Figure 4.5. N_OF_EACH Greedy strategy on FFT32.

4.0.2.2 1_OF_EACH

Even though the ONLY_ONE strategy is an improvement over the conservative one, it only allows one unit of work to run at a time. By increasing the number of hardware modules allocated per dimension, a faster and more energy efficient system design is possible. 1_OF_EACH strategy allow one execution module invocation per FFT dimension. This means that, one instance each of FFT4, FFT8, ..., FFT1024 can be executed simultaneously.

Figure 4.4 shows a sample execution sequence under the 1_OF_EACH strategy. At time ❶, $E_{FFT16} < E_{sys} < E_{FFT32}$. If $E_{sys} > E_{FFT16} + E_{FFT8} + E_{FFT4}$, 1_OF_EACH strategy allows execution of 16_L, 8_{RL} and 4_{RRL} *in parallel*.

4.0.2.3 N_OF_EACH

In this strategy, each dimension has N hardware copies. Hence, N instances each of FFT4, ..., FFT1024 can be executed simultaneously. More hardware modules allow the energy harvesting system to achieve forward progress faster. A possible execution sequence of FFT32 in an N_OF_EACH system is shown in Fig 4.5. At time ❶, assume that $E_{FFT16} < E_{sys} < E_{FFT32}$. With $N = 4$, the 4_OF_EACH system can choose to execute 16_L *in parallel* with 4_{RLL}, 4_{RLR}, 4_{RRL} and 4_{RRR}, provided $E_{sys} > E_{FFT16} + 4 \times E_{FFT4}$.

Algorithm 2 explains the N_OF_EACH strategy. In each schedule point, upto N pending nodes can be scheduled per kernel dimension, restricting the total workload to be within E_{sys}. Runnable nodes with high energy requirements are updated into

nextNodes[*dim*] for every supported dimension *dim*. For each *dim*, upto N of these nodes are scheduled. Arbitration is done based on pending sub-node energies if there are more than N nodes with same energy requirement per dimension.

Algorithm 2 Greedy - N_OF_EACH

```
function SCHEDULER(E_sys)
    if nodes are pending then
        runnables ← nodes with energy < E_sys
        for each dim in dimension do
            maxNodes[dim] ← get max energy dim-nodes from runnables
            if maxNodes[dim].count >N then
                nextNodes[dim] ← N least complete dim-nodes from maxNodes[dim]
            else
                nextNodes[dim] ← maxNodes[dim]
        for each dim in dimension do
            schedule(nextNodes[dim])
```

4.0.3 Predictive Strategy

The work-set scheduled by greedy strategy is guaranteed to complete because system energy at the schedule point exceeds the greedy work-set's energy requirement. However, greedy work-set is limited by the capacitor's energy capacity. Predictive strategy expands the greedy work-set to include additional work beyond that in greedy *in anticipation of future energy income*. As energy is expended on greedy work-set execution during run-time, capacitor becomes available for further energy harvesting. Predictive strategy leverages this surplus energy to execute the anticipatory work-set without waiting for the next schedule point. The greedy work-set is guaranteed to finish. However, completion of anticipatory work-set depends on availability of "surplus energy" during run-time. Assume the workload of Fig. 4.5 where 16_{L} is completed. At ❷, if $\mathrm{E_{FFT4}}<\mathrm{E_{sys}}<\mathrm{E_{FFT8}}$, only 4_{RLL} is scheduled by greedy strategy. Predictive strategy adds all pending nodes to anticipatory work-set. If the surplus energy harvested during 4_{RLL}'s run-time makes $\mathrm{E_{sys}}>\mathrm{E_{FFT4}} + \mathrm{E_{FFT4\_pending}}$, 4_{RLR} which requires $\mathrm{E_{FFT4}}$ starts execution along with pending work of 4_{RLL} which requires $\mathrm{E_{FFT4\_pending}}$, thereby improving parallelism. If surplus energy makes $\mathrm{E_{sys}}>16_{\mathrm{L}}$, 4_{RLL} is made redundant and system computes 16_{R}, thereby improving performance through redundancy. If surplus energy is unavailable, 4_{RLL} is still completed as scheduled by greedy.

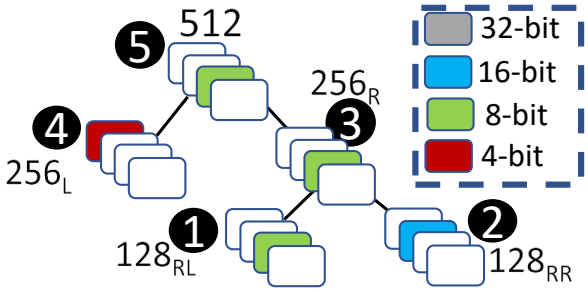

Figure 4.6. ECG - SAS scheduling with 32-bit precision disabled.

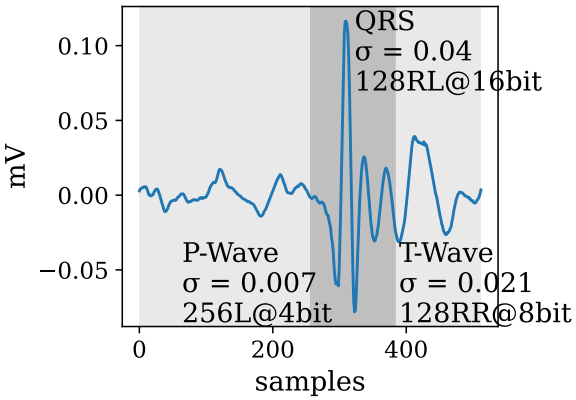

Figure 4.7. ECG data partition for FFT with SAS scheduling and 32-bit partition disabled.

4.0.4 Selective and Fully Approximated Scheduling (SAS / FAS)

Approximation offers a knob of configurability to fine-tune computations to the energy intermittency. To a first order, the lower the precision, the lower the energy requirement for a computation. However, *Fully approximated scheduling* (*FAS*), where all precisions are available and activated based on energy availability, may prove infeasible due to error incurred from insufficient precision.

In *selective approximated scheduling* (*SAS*), our system is statically configured to use only a selected subset of precisions. Note that most IoT applications do not require results to be computed at the highest precision. Fig. 4.6 shows an execution paths followed by our FFT system in processing the ECG dataset [6] of Fig. 4.7. Here, 32-bit precision is statically disabled. The feasibility of lower precisions is decided by the application's accuracy constraints. In MHM [4] and saliency [14], 4-bit and 1-bit precisions should be eschewed as shown in Fig. 2.2 and Fig. 2.3 respectively.

In FAS based FFT64 system, MHM incurs an error of up to 20.8% if 16-, 8- and 4-bit precisions are enabled. A SAS-derivative can be obtained by statically disabling the 4-bit precision, thereby reducing the error to 0.49%.

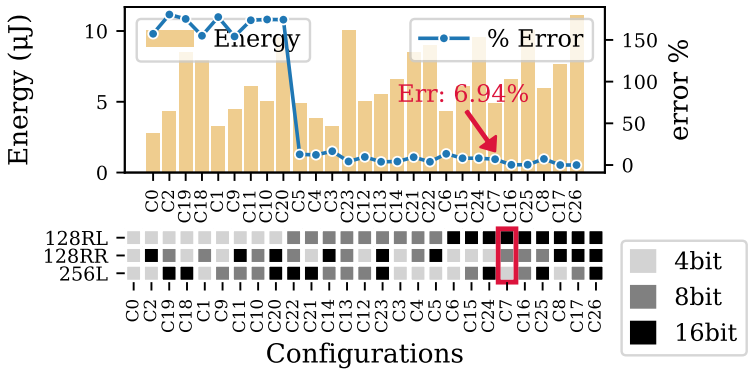

Figure 4.8. AAS based FFT Design space exploration for ECG.

4.0.5 Accuracy Aware Scheduling (AAS)

Precision levels below the application limits can be used in our computing system by dynamically applying them only to certain data phases: sensor output varies over time, with active and passive phases, and active phases embed more information relative to passive phases and are therefore more precision-sensitive. *Accuracy aware scheduling or AAS* dynamically assigns active phases to high precision kernels and passive phases to low precision kernels. In MHM [4], faulty vibrations (active phases) are characterized by a relative increase in signal strength. The same is applicable in sleep apnea detection [23](rate of chest motion), sound monitoring [24], noise filtering, and ECG [6] monitoring. Signal strength is proportional to its variance, σ^2. A comparison of σ is used to classify the data into active/passive phases. The σ-thresholds are application dependent and are reliably obtained from the first 100 input samples. For phases that require high Signal-to-quantization-noise ratio (SQNR), the N-bit precision requirement can be computed as $SQNR_{\text{dB}} = 6.02N + 1.76$.

Fig. 4.7 shows an instance of σ-based AAS. Let the thresholds for 16 and 8-bit precision selection be $\sigma_{\text{16-bit}}$=0.03 and $\sigma_{\text{8-bit}}$=0.01 respectively. The ECG's QRS complex (samples 256 to 383), with σ=0.04, where $\sigma > \sigma_{\text{16-bit}}$, is assigned to an FFT *128RL* compute module at 16-bit precision (Fig. 4.6). T-wave (samples 384 to 511), with σ=0.021, where $\sigma > \sigma_{\text{8-bit}}$, is mapped to *128RR* at 8-bit. P-wave (samples 0 to 255), with σ=0.007, uses *256L* at 4-bit. Merge modules preserve accuracy from their inputs. A comparison of this optimal configuration, $C7 = \{$128RL-16bit, 128RR-8bit, 256L-4bit$\}$, against its alternatives in an FFT512 design space exploration is shown in Fig. 4.8. The configurations are sorted in increasing order of 128RL precision. It is evident that 128RL (having highest σ) needs to be processed precisely to achieve an overall higher accuracy. The optimal configuration achieves a tolerable error margin while utilizing energy efficiently.

Chapter 5
Methodology and Results

We interface our in-house cycle-accurate accelerator with NVP ARM processor modeled in GEM5 [2] as shown in Fig. 3.1. ARM processor schedules workloads to the accelerator. Accelerator's FFT backend uses SpiralFFT IP cores [25] synthesized using ASU's ASAP 7nm technology [5]. CONV backend and scheduler uses in-house IP cores based on Synopsys 32nm SAED library. Scheduler executes within one 10KHz tick consuming up to 0.1uJ. FFT/CONV modules are designed as unscaled and fully streaming and support 4, 8 ... 1024 dimensions. Each module supports 4, 8, 16, and 32-bit precisions. The energy (Fig. 5.1), area (Fig. 5.2), and timing (Table 5.1) requirements obtained by synthesizing these kernels in Design Compiler are loaded into ConfigNVM by MCU during initialization. The system can be extended to model any decomposable workloads by synthesizing their parameters.

We use STT-RAM as our NVM because of its relatively low read/write latency and power requirements [30]. Table 5.2 shows the specifications of STT-RAM used in our design, as obtained from the NVSim simulator [8]. STT-RAM read/write energy and access time for different kernel dimensions are computed based on these specifications. The STT-RAM is power-gated while kernel computation is in progress. We assume one *source-NVM* and one *destination-NVM* per compute module. They are configured as 2-port NVMs since each *destination-NVM* is a *source-NVM* for the next layer.

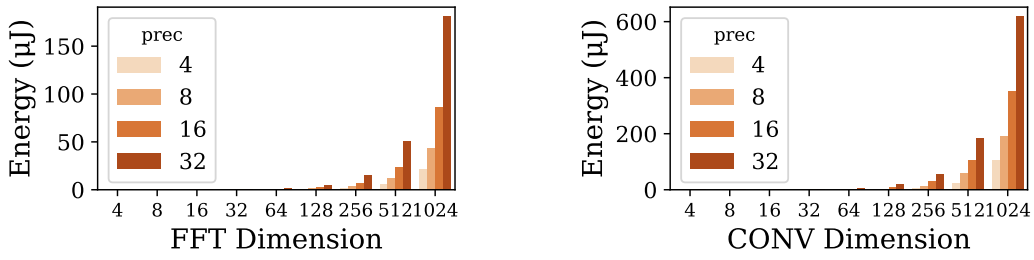

Figure 5.1. Energy requirements of FFT and CONV kernels.

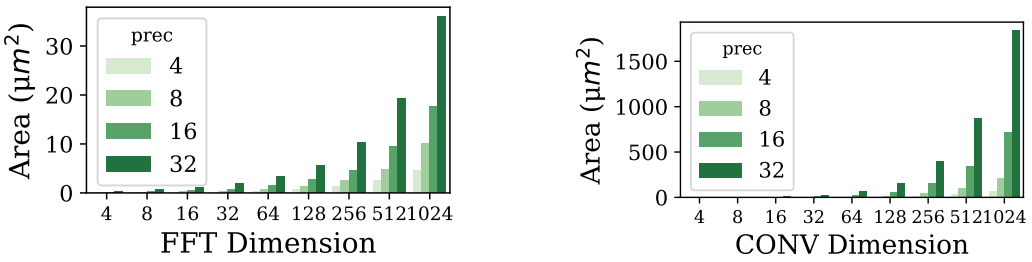

Figure 5.2. Area requirements of FFT and CONV kernels.

Table 5.1. Kernel execution time in cycles (10KHz).

Kernel	**Kernel Dimension**								
	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
fft	24	41	63	94	145	234	403	728	1373
conv	13	25	49	97	193	385	769	1537	3073

Fig. 5.3 shows the power trace, sampled at 10KHz from a piezo-electric transducer fitted on a bike. The mean, median and maximum power from this trace are 23.5 µW, 4.2 µW and 998 µW respectively. We model a 20 µF standard capacitor for energy storage. The capacitor has a leakage resistance path of 100 MΩ and an equivalent series resistance of 10 Ω. DC-DC converter efficiency is set to 85%. Fig. 5.4 shows a detailed DC-DC converter model [3], operating at $V_{in} = 0.6V$.

5.0.1 Conservative strategy (CNSV [7, 21])

CNSV area requirements are given in Fig. 5.2. CNSV's energy and execution time for MHM dataset [4] using FFT/CONV kernels at 32-bit precision is shown in Table 5.3. The exponential increase in these parameters with dimension shows that CNSV is impractical in energy-scarce IoT nodes.

Table 5.2. STT-RAM specifications.

Parameter	**Values**
Total Area	1.224mm^2
Read Latency	2.553ns
Write Latency	10.786ns
Read Dynamic Energy	750.043pJ
Write Dynamic Energy	872.985pJ
Word Width	512 bits
Capacity	2MB

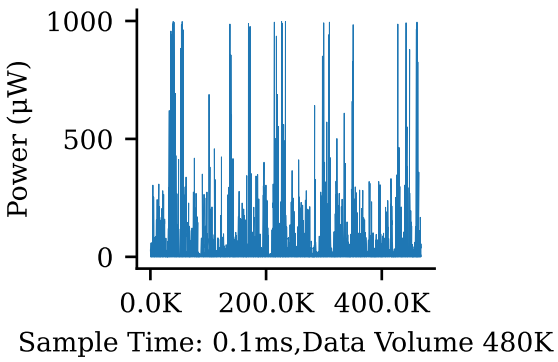

Figure 5.3. Piezo-electric power trace.

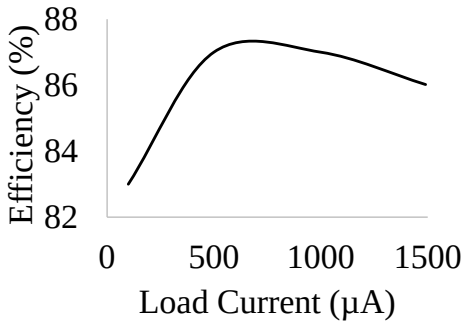

Figure 5.4. Non-linear DC-DC profile.

5.0.2 Greedy & Predictive Strategies

Fig. 5.5 shows the energy reduction achieved by MHM dataset [4] for 32-bit computations by different strategies compared to CNSV (Table 5.3). Kernel dimensions 4, 8, 16, and 32 are highly sensitive to energy variations as their absolute energy requirements are very low. For dimensions 64 and above, we observe energy reduction of up to 5.4x and 2.7x for FFT1024 and CONV1024 respectively.

Fig. 5.6 shows speedup of up to 3.7x and 2.4x for FFT1024 and CONV1024 compared to CNSV. ONLY_ONE strategy doesn't achieve gains with predictive scheduling as only one module is executed at a time. Even if N_OF_EACH assigns 100% of capacitor's energy to greedy work-set execution, predictive strategy can schedule anticipatory work

Table 5.3. Energy and cycles for CNSV [7, 21] strategy.

kernel	**param**	**Dimensions**				
		64	**128**	**256**	**512**	**1024**
fft	energy (uJ)	2.41	7.90	26.0	65.1	226.8
	cycles	598	1479	4314	7059	18226
conv	energy (uJ)	7.90	35.2	66.0	234.6	1150.2
	cycles	1476	4597	7432	18516	40376

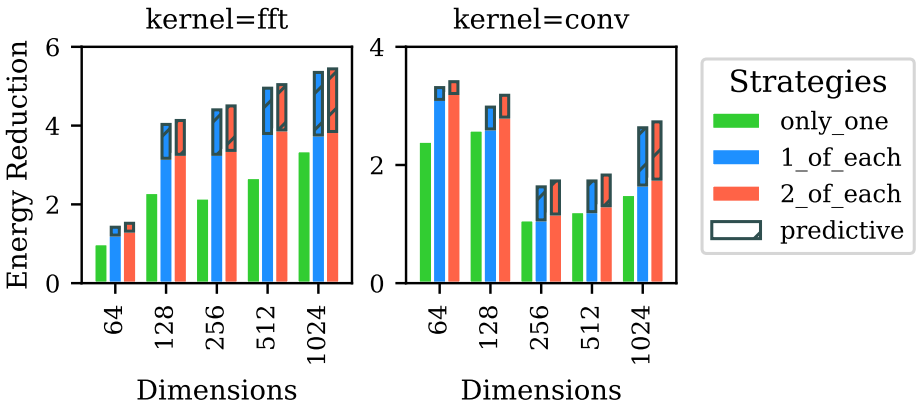

Figure 5.5. Energy reduction of greedy strategies relative to CNSV.

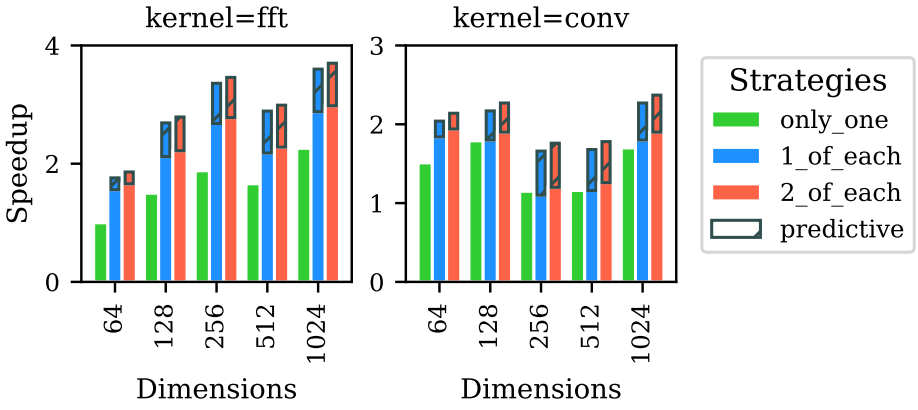

Figure 5.6. Speedup of greedy strategies relative to CNSV.

once at least 18% of capacitor energy is depleted by greedy work-set execution during run-time. Fig. 5.7 shows the area cost incurred by the strategies while reducing the Energy-Delay Product (EDP) compared to CNSV. For each kernel dimension, the strategy with the best EDP reduction and consuming the least number of hardware modules is circled as “optimal". EDP increases by up to 14% on switching from the constant efficiency of 85% to the non-linear profile shown in Fig. 5.4.

5.0.3 Selective and Fully Approximated Scheduling (SAS / FAS)

MHM dataset [4] achieves energy reduction (up to 9x and 10.4x) and speedup (up to 5.4x and 4.7x) for FFT1024 and CONV1024 on using FAS as shown in Fig. 5.8 and Fig. 5.9 respectively. However, as shown in Fig. 5.10, FAS incurs significant errors that render it practically unusable. The tolerable error is subjective to the application. Hence, we show error distribution in percentage to demonstrate run-time quality trade-off as

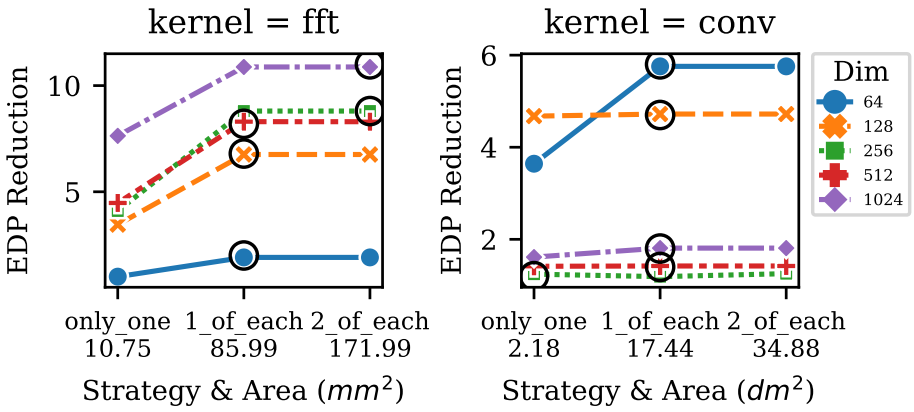

Figure 5.7. EDP reduction of greedy normalized to CNSV.

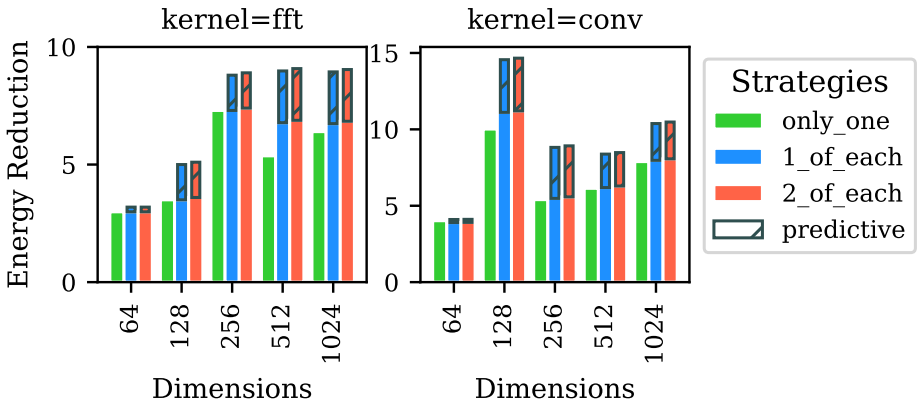

Figure 5.8. Energy reduction of FAS relative to CNSV.

opposed to choosing a fixed quality target per application. In Fig. 5.10, 'max' represents the maximum error incurred if all the bits discarded by precision adjustment had a value of 1. Impact of SAS and FAS on MHM and saliency detection is shown in Table 5.4.

5.0.4 Accuracy Aware Scheduling (AAS)

As shown in Table 5.4, FAS is a special case of SAS where all precision options are enabled. Since FAS achieves the best speedup and energy efficiency compared to SAS by trading accuracy, we compare AAS against FAS. Table 5.6 shows the improvements of AAS relative to CNSV. Table 5.5 shows the improvements of FAS relative to CNSV. Saliency [14], Sleep-apnea (FFT) [23], sound monitoring(CONV) [24], ECG(FFT) [6], noise filtering(CONV), and MHM (FFT/CONV) [4] datasets at 1024-dim are considered. By trading up to 2.8x energy and 1.5x speed compared to FAS, AAS achieves on average 6.6x energy reduction and 4x speedup while incurring an average error of up to 6.9%.

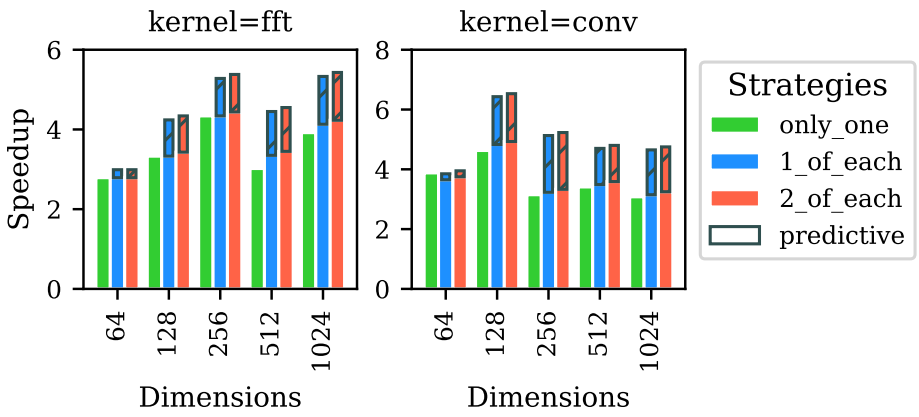

Figure 5.9. Speedup of FAS relative to CNSV.

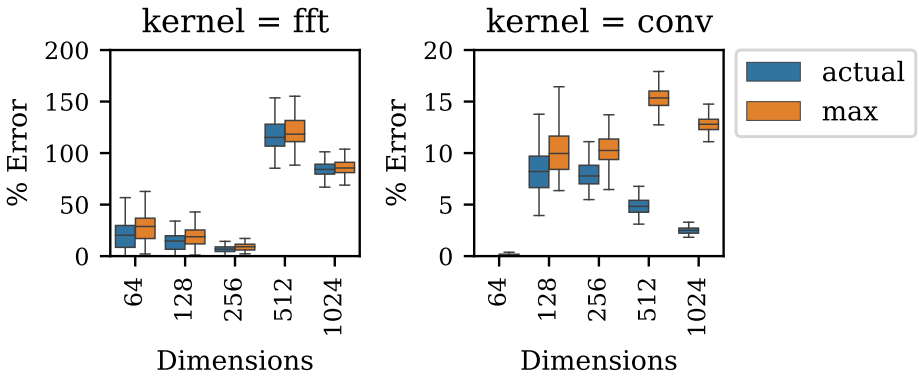

Figure 5.10. Error incurred by FAS.

Table 5.4. Impact of Selective approximated scheduling.

App	Enabled precisions	CNSV - Energy Energy (error=0%)	2_OF_EACH	
			Energy	Error %
Saliency	8	111.9 mJ	65.1 mJ	0
	8, 4	44.4 mJ	25.8 mJ	7
	8, 4, 2	19.1 mJ	11.1 mJ	23
	8, 4, 2, 1 (FAS)	14.7 mJ	8.5 mJ	52
MHM	32	31.9 uJ	10.6 uJ	0
	32, 16	14.4 uJ	4.7 uJ	3
	32, 16, 8	7.1 uJ	2.3 uJ	12
	32, 16, 8, 4 (FAS)	3.5 uJ	1.1 uJ	61

Table 5.5. Comparison of FAS relative to CNSV over 1024-dim datasets. FAS uses 2_OF_EACH with predictive.

Dataset	CNSV		FAS (8_OF_EACH)				
	energy (uJ)	cycles	energy reduction	speedup	error %		
					max	min	mean
ECG	226	18227	**6.74**	**4.13**	183.1	13.8	**76.9**
env_sound	1150	40377	**7.9**	**3.15**	38.5	0.4	**8.7**
noise_filter	1150	40377	**7.9**	**3.15**	15.9	0.3	**6.3**
sleep_apnea	226	18227	**6.74**	**4.13**	34.0	0.1	**9.9**
vib_conv	1150	40377	**7.9**	**3.15**	11.9	2.04	**6.2**
vib_fft	226	18227	**6.74**	**4.13**	335.2	63.5	**242**
saliency	111x10^3	124x10^6	**8.3**	**4.8**	221.9	65.7	**210**

Table 5.6. Comparison of AAS relative to CNSV over 1024-dim datasets. AAS uses 2_OF_EACH with predictive.

Dataset	CNSV		AAS (8_OF_EACH)								
	energy (uJ)	cycles	energy reduction factor			speedup			error %		
			max	min	mean	max	min	mean	max	min	mean
ECG	226	18227	6.73	5.9	**6.4**	4.1	3.4	**3.8**	8.0	0.06	**6.5**
env_sound	1150	40377	6.8	5.2	**5.7**	2.7	2.2	**2.4**	6.9	0.3	**2.2**
noise_filter	1150	40377	7.9	5.2	**5.9**	3.1	2.2	**2.5**	10.7	0.3	**2.7**
sleep_apnea	226	18227	6.4	5.9	**6.2**	3.9	3.3	**3.7**	0.27	0.004	**0.05**
vib_conv	1150	40377	6.3	5.2	**5.5**	2.6	2.2	**2.4**	0.41	0.013	**0.25**
vib_fft	226	18227	6.4	5.9	**6.2**	3.9	3.3	**3.6**	3.5	0.0	**0.72**
saliency	111x10^3	124x10^6	8.3	7.7	**8**	4.7	4	**4.4**	8.64	0.01	**6.92**

Chapter 6
Discussion of Related Works

Chinchilla [21] and Cleancut [7] propose general purpose NVP solutions where capacitor size restrained task boundaries are defined for sequential program execution. Program order execution of these tasks does not present any opportunity for parallelism. Moreover, Chinchilla and Cleancut do not consider approximate computing which is amenable in IoT workloads. What's Next [9] processes each kernel to an acceptable accuracy without taking advantage of parallelism or scheduling based on data relevance. In contrast, our system decomposes workload into independent kernels of varying sizes and energy requirements. These kernels can be executed whenever their energy constraints are satisfied. Our accuracy aware scheduling dynamically chooses compute precision for data segments based on data relevance, there by minimizing impact on overall accuracy.

ResiRCA [26] reconfigures a single loop tiling based CONV accelerator to match workloads to energy. It can execute already decomposed workloads sequentially. It doesn't explore parallel executions. Moreover, ResiRCA is limited to convolution kernels. Our work schedules decomposed kernels on a collection of dedicated hardware in parallel. Further, we boost system efficiency through approximate computing on a generalized framework that supports any decomposable kernel.

Nvalt [13] proposes a lookup table based approximation strategy. It uses offline processing to store most frequent input patterns and their outputs into a lookup table. During runtime, workloads are sent to the lookup table which generates a predicted output based on previous inputs. Large lookup tables required to increase prediction accuracy significantly increases energy. In contrast, our system intelligently identifies data segments that are amenable to approximation.

Chapter 7
Conclusion

Most IoT applications employ workloads that can be decomposed into smaller compute/merge kernels. Availability of many such independent kernels offers opportunity to tune them to the intermittent nature of ambient energy. We propose an energy and accuracy aware architecture for such workloads. Our FFT and CONV accelerators benefit from greedy and predictive scheduling strategies of this framework. Accuracy Aware scheduling, with minimal accuracy loss, achieves high speedups and energy efficiencies relative to existing systems.

Bibliography

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[3] A. Biswas, Y. Sinangil, and A. P. Chandrakasan. A 28 nm fdsoi integrated reconfigurable switched-capacitor based step-up dc-dc converter with 88 *IEEE Journal of Solid-State Circuits*, 50(7):1540–1549, 2015.

[4] A. Boudiaf, A. Moussaoui, A. Dahane, and I. Atoui. A comparative study of various methods of bearing faults diagnosis using the case western reserve university data. *Journal of Failure Analysis and Prevention*, 2016.

[5] L T. Clark, V Vashishtha, L Shifren, A Gujja, S Sinha, B Cline, C Ramamurthy, and G Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105 – 115, 2016.

[6] G. Cliford. *Signal Processing Methods For Heart Rate Variability Analysis*. PhD thesis, University of Oxford, 2002.

[7] Alexei Colin and Brandon Lucia. Termination checking and task decomposition for task-based intermittent programs. In *Proceedings of the 27th International Conference on Compiler Construction*, CC 2018, page 116–127, New York, NY, USA, 2018. Association for Computing Machinery.

[8] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, 2012.

[9] K. Ganesan, J. San Miguel, and N. Enright Jerger. The what's next intermittent computing architecture. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 211–223, 2019.

[10] A. Gatto and E. Frontoni. Energy harvesting system for smart shoes. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, 2014.

[11] A. Ghasemazar and M. Lis. Gaussian mixture error estimation for approximate circuits. In *Design, Automation Test in Europe Conference*, 2017.

[12] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 356–363, 2015.

[13] M. Imani, D. Peroni, and T. Rosing. Nvalt: Nonvolatile approximate lookup table for gpu acceleration. *IEEE Embedded Systems Letters*, 10(1):14–17, 2018.

[14] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[15] D. Jung, Z. Zhang, and M. Winslett. Vibration analysis for iot enabled predictive maintenance. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1271–1282, 2017.

[16] M. T. Lazarescu. Design of a wsn platform for long-term environmental monitoring for iot applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 3(1):45–54, 2013.

[17] V. Leonov, T. Torfs, P. Fiorini, and C. Van Hoof. Thermoelectric converters of human warmth for self-powered wireless sensor nodes. *IEEE Sensors Journal*, 7(5):650–657, 2007.

[18] V. Leonov, T. Torfs, R. J. M. Vullers, J. Su, and C. Van Hoof. Renewable energy microsystems integrated in maintenance-free wearable and textile-based devices: The capabilities and challenges. In *2010 IEEE International Conference on Industrial Technology*, pages 967–972, 2010.

[19] X. Li, U. Dennis Heo, K. Ma, V. Narayanan, H. Liu, and S. Datta. Rf-powered systems using steep-slope devices. In *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, pages 73–76, 2014.

[20] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 526–537, 2015.

[21] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 129–144, Carlsbad, CA, October 2018. USENIX Association.

[22] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. Wispcam: A battery-free rfid camera. In *2015 IEEE International Conference on RFID (RFID)*, pages 166–173, 2015.

[23] R. Nandakumar, S. Gollakota, and N. Watson. Contactless sleep apnea detection on smartphones. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 967–972, 2015.

[24] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015.

[25] M. Puschel, J. M. F. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, Jianxin Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo. Spiral: Code generation for dsp transforms. *Proceedings of the IEEE*, 93(2):232–275, 2005.

[26] K. Qiu, N. Jao, M. Zhao, C. S. Mishra, G. Gudukbay, S. Jose, J. Sampson, M. T. Kandemir, and V. Narayanan. Resirca: A resilient energy harvesting reram crossbar-based accelerator for intelligent embedded processors. In *2020 IEEE International*

Symposium on High Performance Computer Architecture (HPCA), pages 315–327, 2020.

[27] M. W. Shafer and E. Morgan. Energy harvesting for marine-wildlife monitoring. In *2014 ASME Conference on Smart Materials, Adaptive Structures and Intelligent Systems*, 2014.

[28] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[29] Fang Su, Kaisheng Ma, Xueqing Li, Tongda Wu, Yongpan Liu, and Vijaykrishnan Narayanan. Nonvolatile processors: Why is it trending? In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, pages 966–971, 2017.

[30] Guangyu Sun, Jishen Zhao, Matt Poremba, Cong Xu, and Yuan Xie. Memory that never forgets: emerging nonvolatile memory and the implication for architecture design. *National Science Review*, 5(4):577–592, 08 2017.