Incremental-Precision based Feature Computation and Multi-Level Classification for Low-Energy Internet-of-Things

Sandhya Koteshwara, *Student Member, IEEE,* and Keshab K. Parhi, *Fellow, IEEE*

***Abstract*—This paper presents a novel technique to reduce energy consumption of a machine learning classifier based on *incremental-precision feature computation and classification*. Specifically, the algorithm starts with features computed using the lowest possible precision. Depending on the classification accuracy, the features of the previous level are combined with features of the incremental-precision to compute the features in higher-precision. This process is continued till a desired accuracy is obtained. A certain *threshold* that allows many samples to be classified using a low-precision classifier can reduce energy consumption, but increases misclassification error. To implement hardware which provides the required updates in precision, an *incremental-precision architecture* based on data-path decomposition is proposed. One novel aspect of this work lies in the design of appropriate thresholds for multi-level classification using training data such that a family of designs can be obtained that enable trade-offs between classification accuracy and energy consumption. Another novel aspect involves the design of hardware architectures based on data-path decomposition which can incrementally increase precision upon demand. Using a seizure detection example, it is shown that the proposed incremental-precision based multi-level classification approach can reduce energy consumption by 35% while maintaining high sensitivity, or by about 50% at the expense of 15% degradation in sensitivity compared to similar approaches to seizure detection in literature. The reduction in energy is achieved at the expense of small area, timing and memory overheads as multiple classification steps are used instead of a single step.**

***Index Terms*—Energy reduction, Machine learning, Multi-level classification, Datapath decomposition, Incremental-precision computation, Approximate computing, Seizure Detection**

I. Introduction

With ever shrinking devices and embedded platforms, there is an increased demand for energy reduction in applications. Approximate computing has been proposed as an approach to address the energy requirements of modern devices. In this regard, several approximate computing techniques at algorithmic, circuit and synthesis levels have been proposed over the past decade [1]. Reducing word-lengths of components and wires is one of the approaches to achieve energy savings. In this work, we consider approximation as applied to feature extraction unit of classification systems. Feature extraction, which involves extraction of useful information from samples, helps in discriminating the data belonging to different classes. The datapaths of feature extraction consume significant amount of energy. For a complete system-on-chip classifier it is important to focus on the approximation of feature extraction unit which has not been addressed in previous literature.

This paper was supported in part by the National Science Foundation under grant number CCF-1749494.

To apply approximation techniques using word-length reduction we propose a novel *incremental-precision* feature extraction and classification algorithm. The rationale for using incremental-precision is that samples that are far away from the decision boundary can be easily classified using lower precision using a Level-1 classifier. Furthermore, the classifier may require lower precision and may be a simple classifier that requires thresholding. Samples that are not classifiable by the Level-1 classifier can be processed by a Level-2 classifier where the features are computed by incrementally *augmenting* higher precision to the features of earlier precision. This process is repeated as needed to achieve the desired classifier performance.

For the classification of samples from one step to the next, we design a *threshold calculation* unit. This paper also proposes the design of *incremental-precision feature computation* which are suitable for the incremental-precision classification paradigm. Specifically, an incremental-precision fast Fourier transform (FFT) design is proposed in this paper. These architectures are based on *data-path decomposition* approaches [2] which are commonly used in error correction schemes but have not been applied in an incremental-precision scenario. *Error models* to understand the errors due to approximation and estimates of power reduction, area reduction/overhead and timing overheads are also presented.

To demonstrate the applicability of incremental precision algorithm and hardware architecture, a complete case study using the seizure detection classification problem is presented. The proposed incremental-precision FFT architecture and its corresponding error and energy models are employed in this design. By studying the *trade-off* between classification accuracy and associated energy reduction, we identify different configurations of values that can be selected to construct the incremental-precision system. An approach to incremental-precision based classification specifically addressing linear classifiers is presented in [3]. The proposed incremental-precision algorithm, incremental-precision architecture and its application to feature extraction are the first in literature and to the best of our knowledge have not been presented before.

The rest of the paper is organized as follows. Section II discusses the current state of the art in approximate computing, incremental algorithms and seizure detection. Section III presents the incremental-precision feature extraction and classification algorithm by describing training, testing and thresh-

old calculation steps. The design approaches for incremental-precision datapaths are described in Section IV. In Section V we apply the incremental precision architecture design methodology on an FFT datapath and discuss associated error and energy models in Section VI. A detailed description of the seizure detection application and the experimental methodology for accuracy and energy measurements is provided in Section VII. Finally we present classification performance vs. energy tradeoff results using incremental-precision in Section VIII.

II. Related work

In this section, we discuss some of the recent literature which address approximate computing algorithms and architectures. Approximate computing has been used in the realm of signal processing applications [4], [5]. With the increasing use of machine learning based techniques, approximate computing has been applied to typical components such as classifiers [6] and synapses of Artificial Neural Networks (ANN) [7]. Bit-width reduction on feature extraction units has been discussed in [8] for the features of an object detection algorithm. An approach to energy reduction by incrementally improving precision of neural networks has been addressed in [9]. However, none of these literature has presented the application of multi-stage bit-width reduction.

The proposed incremental-precision classification approach is similar to the concept of boosting. Boosting is a technique that has been popularly adopted in several machine learning algorithms to design strong classifiers from multiple weak classifiers. The multiple classifiers used in boosting try to improve on the misclassifications of the predecessor [10]. Several applications of the AdaBoost technique to boosting have been outlined in [11]. In this paper, we apply the basic idea of boosting to classification but with the goal of energy reduction in mind. However, there are several differences between the proposed approach and the boosting strategy. These are outlined in Section III-D.

Data-path decomposition has been used to design error-resilient architectures [2]. In this technique, the data-path is divided into most significant bits (MSB) and least significant bits (LSB) components and it is assumed that the two are subject to different types of errors. By combining the two data-paths, error correction can be obtained. The proposed incremental-precision algorithm utilizes a similar approach to decomposing data-paths. The resultant architecture can provide outputs of different precisions by consuming minimal energy.

Incremental-precision architecture is demonstrated using fast Fourier transform (FFT) architectures. Several approaches to both real and complex FFT architectures using ideas of folding, pipelining and parallel processing for both real and complex inputs have been addressed in literature [12], [13]. However, none of these architectures can provide improvements in precision in an incremental manner as required for the proposed low energy algorithm. Approaches to incrementally increasing the length of FFT algorithms has been proposed in [14], [15]. These algorithms improve the SNR by increasing number of stages but do not address lowering precision which is better suited for low-energy multi-level machine learning classifiers. Dynamic Precision has been proposed to adapt the word-length to SNR; however, a single-precision architecture is used in this system [16]. No literature specifically addresses the challenges of FFT data-path decomposition including step wise increments and handling of scaling and other issues.

Seizure detection is used as an example to illustrate the incremental-precision algorithm and its associated architecture. Several techniques to detect seizures have been presented in the literature [17]–[19]. It has also been shown that using spectral powers as features results in high accuracy for seizure prediction [20]. A low complexity, high accuracy algorithm for seizure prediction has been proposed in [21], [22]. The features used in this algorithm include absolute spectral powers, relative spectral powers and ratios of spectral powers. These features have also been used to detect seizures [23]. However, the energy consumption of these algorithms is still high and can be reduced by using lower precision datapaths. Moreover, support vector machines (SVMs) which are resource consuming are used in this algorithm. [24] implements seizure detection using Logistic regression (LR) classifiers proving that LR classifiers require the least energy among classifiers such as k-nearest neighbor classifiers, support vector machines with linear and polynomial kernels, naive Bayes and neural networks. Thus, use of LR in seizure detection algorithms can reduce energy as well as complexity.

III. Incremental-precision feature extraction and classification system

The proposed incremental-precision algorithm operates on different precision data in an incremental manner starting from the lowest precision. We attempt to classify most of the sample points in low precision before we select samples which need to be processed in the next higher precision. Four important questions need to be addressed:

- How should the samples which need to be reprocessed in the next precision step be selected?
- After how many stages (precision updates), should the algorithm stop?
- What is the value of the lowest precision that the incremental-precision algorithm can start with?
- What are the trade-offs between classification accuracy and energy consumption?

The first two questions are addressed in this section. The last two questions will be discussed using a specific example in Section VI.

A. Training steps

The training steps of the incremental-precision algorithm are described in Fig. 1. The first step of the algorithm starts with the lowest precision N. The feature extraction unit operates in N-bit precision in this step. The generated features (F) are passed to a feature selection algorithm to reduce the dimensionality and obtain selected features (SF). A simple classifier is trained and predictions are obtained (denoted as L). We note that along with predicted labels, probability estimates

(denoted as P) for each sample are also required. Probabilities of the classified samples determine the probability of a sample belonging to a particular class. This information will be used for threshold calculation and, hence, the classifier should be able to provide good probability estimates. These probabilities can be obtained by training a logistic regression function on the decision variables of a classifier.

The generated labels (L), probabilities (P) and actual labels (AL) of the training data are then fed to a threshold calculation unit. We will discuss the details of the threshold calculation unit next. However, we observe that the threshold calculation unit generates two thresholds: minimum threshold ($T1$) and maximum threshold ($T2$). The threshold filter then uses P, $T1$ and $T2$ to determine which of the samples are to be reprocessed. This information is provided as indices to the next step of the algorithm. The selected features, classifier model and the threshold values $T1$ and $T2$ are stored and are used for testing of the data. This completes one step of the training process.

For subsequent steps, the unclassified samples from the *threshold filter* are reprocessed in higher precision feature extraction units of precision $N+2$. Note that other increments in precision can also be used depending on the requirements of the application. The samples are then subjected to the same process of classification and threshold calculation as before. Minimum and maximum thresholds ($T3$ and $T4$, $T5$ and $T6$ etc.) are also calculated for these steps. For the last step of the process, no threshold calculation is performed. The stopping criteria can be pre-determined to be equal to a specified number of steps (which simplifies the algorithm) or when the number of samples reselected is lower than a specified value. Stopping the algorithm when the number of samples reselected is too low prevents overfitting of data.

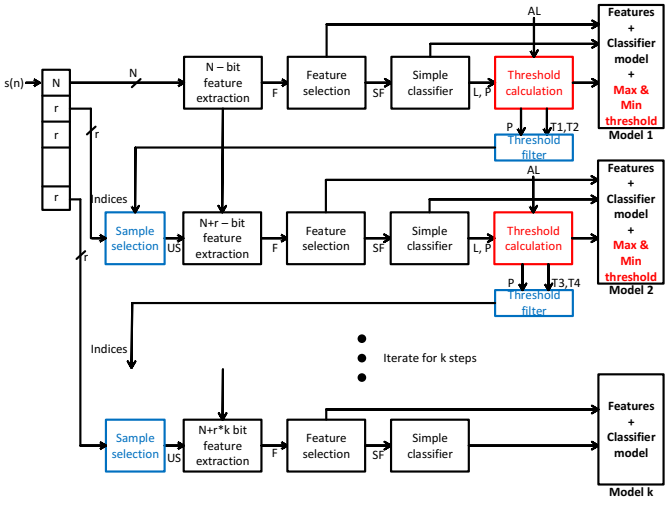


Fig. 1. Steps of training process of the incremental-precision algorithm using varying precision features.

B. Threshold calculation

The threshold calculation algorithm is an important step in the incremental-precision algorithm since it determines the number of samples to be selected for reprocessing in higher precision. This affects both the accuracy of the overall classification as well as the energy consumption of the system. The goal of this unit is to try to filter out as many correctly classified samples as possible from the misclassified samples.

A better visualization of the thresholds that need to be calculated is provided in Fig. 2. To generate this plot, four arrays are created representing the probabilities of misclassified samples and correctly classified samples in class 0 and class 1. The arrays are then sorted in ascending order of probabilities. For example, let us consider P to be probability of a sample in class 1. For correctly sampled class 1 points, this value is high. For incorrectly classified class 1 points this value is low. The opposite values hold for class 0 samples. The two thresholds $T1$ (minimum) and $T2$ (maximum) are also marked in this plot. It is to be noted that all sample points having probabilities between minimum and maximum thresholds need to be resampled. By adjusting the value of $T1$ and $T2$, we determine how many points should be reprocessed while ensuring high accuracy values.

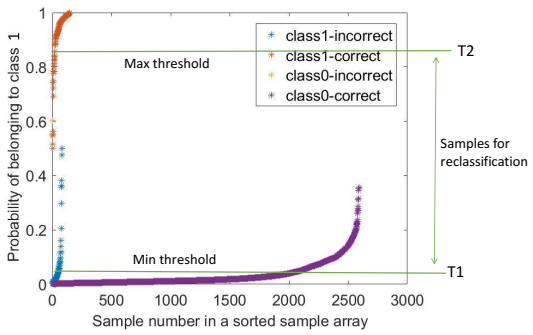


Fig. 2. Plot of sorted probability estimates of classified data samples for class 0 and class 1 samples. Initial values of thresholds can be set using this data.

Next, we discuss the algorithm for threshold calculation as outlined in Algorithm 1. First, consider the calculation of minimum threshold value ($T1$ for Step 1). Sample points lying above the minimum threshold are to be resampled in the next higher precision. This means that the higher the value of $T1$, less are the number of samples to be reprocessed. However, this also means that more number of blue points (incorrectly classified class 1 samples) are not selected for reprocessing. Thus, to obtain a desirable value, we start with a threshold value of $T1 = 0.1$ (for example) and set the maximum percentage of misclassified class-1 points allowed to a value $M1$. At each iteration, the threshold is lowered and the percent of misclassified class-1 points are calculated. The algorithm stops when the percent of misclassified class-1 points is lower than $M1$. The final threshold value $T1$ is the output of the algorithm. The initial value of $T1 = 0.1$ is based on the data from Fig. 2 and results in faster convergence of the threshold calculation algorithm. In general, the initial value ($init\_val1$) can be selected based on the characteristics of the data being classified. Similar steps can be followed to calculate the maximum threshold value $T2$ using allowed class-0 misclassification error, $M2$, and an initial value of $T2$

equal to $init\_val0$.

Algorithm 1: Algorithm for calculating maximum and minimum threshold values $T1$ and $T2$ of Step 1 of incremental-precision classification algorithm

Output: Final threshold = T1, T2
Input: P (Probability estimates from the classifier),
L (Predicted labels), AL (Actual labels),
M1 (Allowed misclassification error Class-1),
M2 (Allowed misclassification error Class-0)
Initialize: T1 = $init\_val1$; T2 = $init\_val0$;
Compute T1 (Minimum threshold):
while *Class-1 misclassification error* $>$ *M1* **do**
  T1 = T1 - 0.0005;
  for $i = 0;\ i < size(L);\ i = i + 1$ **do**
    if *P* $<$ *T1* **then**
      if *AL ! = L* **then**
        err1 = err1 + 1;
      end
    end
  end
  Class-1 misclassification error = err1 / Total class-1 points;
end
Compute T2 (Maximum threshold):
while *Class-0 misclassification error* $>$ *M2* **do**
  T2 = T2 + 0.0005;
  for $i = 0;\ i < size(L);\ i = i + 1$ **do**
    if *P* $>$ *T2* **then**
      if *AL ! = L* **then**
        err0 = err0 + 1;
      end
    end
  end
  Class-0 misclassification error = err0 / Total class-0 points;
end

Once these thresholds are obtained, the sample probabilities can be used to determine whether processing in the next precision is necessary or not. Thus, the sample is reselected if its probability $P > T1$ and $P < T2$. This is performed in the threshold filtering unit to complete the sample selection process and obtain unclassified samples (US). The number of unclassified samples are then checked to see if they are sufficiently high. For simplicity, we check if $size(US) > 1\% \times size(S)$ where $size(US)$ is the size of reselected sample set and $size(S)$ is the size of the original sample set. If this check passes, the feature extraction and classification continue in the next higher precision. Else, the algorithm is stopped. Note that the stopping criterion of the algorithm can be varied to ensure prevention of overfitting. Similarly the thresholds $T3$ and $T4$ of stage 2 can be calculated using maximum allowed class-1 misclassification error, $M3$, and maximum allowed class-0 misclassification error, $M4$. Thresholds $T5$ and $T6$ of stage 3 can be calculated using $M5$ (maximum allowed class-1 misclassification error) and $M6$ (maximum allowed class-0 misclassification error) and so on.

C. Testing steps

The testing algorithm follows the training process of incrementally increasing precision of feature computation. To move from one precision to the next, the threshold values calculated and stored at each step of the training process are used. If the probability estimate of the test sample does not lie in the window formed by the maximum and minimum threshold values, testing is stopped. Else, the sample is reprocessed in the next higher precision. This process is illustrated in Fig. 3. We note that even though the training process has an overhead with respect to threshold calculation, this overhead is not reflected in the testing process. During testing, only the stored classification model and calculated threshold values are used for sample selection. We observe that the only overhead in this case is the comparison of the probability with thresholds which can be done with a simple comparator.

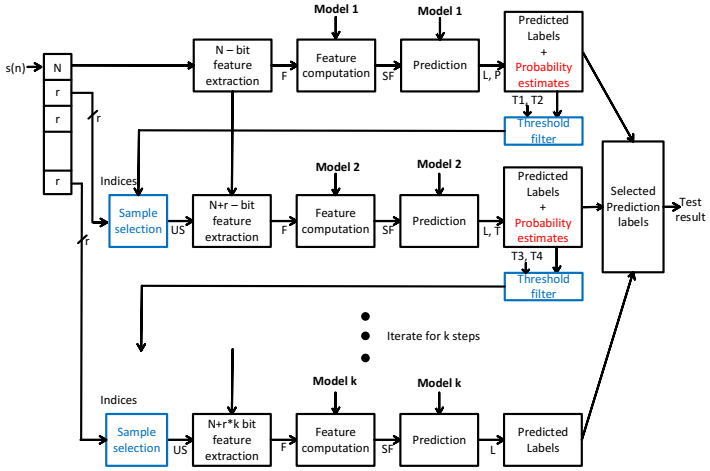


Fig. 3. Steps of the testing process of incremental-precision algorithm using varying precision features.

D. Similarities and differences between AdaBoost and incremental precision algorithm

The incremental precision feature extraction and classification algorithm is similar to the AdaBoost algorithm popularly used in machine learning algorithms [10]. The similarities and differences between boosting and our algorithm are presented next.

The AdaBoost algorithm operates by training multiple weak classifiers to the training data in a step-wise manner using weights, eventually resulting in a strong classifier. The weights selected determine the significance of the particular training sample. We summarize the AdaBoost algorithm below:

- Initialize the weights of all samples of the training data to be equal.
- Train a weak classifier and obtain the accuracy in terms of a weighted error (based on both weight of sample and its classification label).
- Update the weights of training samples using the accuracy of previous step.
- Repeat for specified number of steps or till no further improvement in accuracy is obtained.

- Test by applying all weak classifiers on the testing data and use a weighted average (based on the predictions and accuracy of each step) to determine the final prediction label.

Similarly, the target of incremental classification algorithm is to use multiple weak classifiers (due to low precision feature extraction) to obtain a final strong classifier. However, the goal of this system is to minimize energy by minimizing the number of samples that need to be processed in higher precision. Thus the incremental precision algorithm can be summarized as below:

- Begin with a low precision of $N-$bits on all training data.
- Train a simple classifier and obtain the threshold values $T1$ and $T2$.
- Pass data through a threshold filter and update the filtered indices of training data to next higher precision $N+2$. The increment by 2 bits could be varied in different applications. Note that, unlike AdaBoost, all the training samples are *not* reclassified in this step.
- Repeat for specified number of steps or till number of filtered samples is low. Note that the stopping criteria is based on the size of the training sample set since we are reducing this value in each step.
- Test by applying the classifiers in an incremental manner. Only test samples which are filtered through the thresholds need to be reprocessed. Unlike AdaBoost, all test samples need *not* go through all steps of training. Also, the proposed algorithm does *not* compute a weighted result like in AdaBoost.

‘

IV. Incremental-Precision Hardware Architectures using Data-Path Decomposition

The incremental-precision classification algorithm is based on the idea of incrementally improving precision of features at each level. This can be achieved by converting the feature extraction process to an incremental-precision system using data-path decomposition of the architecture. Data-path decomposition of adders, multipliers and several other components with respect to error-correction schemes have been addressed in [2]. However, in this paper, we approach data-path decomposition such that the complete data-path is decomposed. The goal is to separate the processing of most significant bits and the least significant bits of the system. After computation of the most significant bits of the output, the outputs are stored in memory. For subsequent improvements in accuracy, the output is accessed from memory and combined with the output generated by computation of lower bits. The details of data-path decomposition of a fast Fourier transform (FFT) module is discussed next.

A. Decomposition of adders and multipliers

The decomposition of addition of two operands x_1 and x_2 can be represented by Equation (1). In this equation N represents the number of most significant bits computed in the first part of the decomposed hardware.

$$\begin{aligned} S &= x_1 + x_2 \\ &= (x_{1M} + 2^{-N}x_{1L}) + (x_{2M} + 2^{-N}x_{2L}) \\ &= (x_{1M} + x_{2M}) + 2^{-N}(x_{1L} + x_{2L}) \end{aligned} \tag{1}$$

Similarly, the decomposition of multiplication of two operands x and W is represented by Equation (2). In this equation, the value x is decomposed while W is not. W represents coefficients such as filter weights or twiddle factors which are generally stored in memory.

$$\begin{aligned} Y &= x \times W \\ &= (x_M + 2^{-N}x_L)W \\ &= (x_M W) + 2^{-N}(x_L W) \end{aligned} \tag{2}$$

To implement these adders and multipliers in fixed-width, appropriate carry generation, approximation of multiplication and carry propagation have to be performed. The details of the decomposition are provided in Appendix A. From the decomposition architectures, it is to be noted that decomposition of addition followed by subsequent combination results in no error. However, decomposition of multiplication followed by combination results in approximation error. We discuss the associated error in Section V-A.

B. Decomposition of data-path of FFT to form incremental-precision architectures

The idea of incremental-precision architecture is demonstrated using a real FFT architecture as described in [25]. Fig. 4 illustrates a 16-point radix-2^2, 2-parallel folded real FFT. This architecture belongs to the class of hybrid architectures where the datapath includes both real and complex data-paths. Folding is a technique to reduction of hardware consumption by time-multiplexing data to reuse components [26]. The paths marked in blue are the complex paths while the rest are real datapaths. Individual components of the data-path which include butterflies (BFI, BFII and BFIV) and Delay-switch-delay (DS) are also defined in Fig. 4. The multiplier indicated in the figure is a complex multiplier which consists of 4 real multipliers. Details of the architecture including the control-path signals are not discussed in this paper and the reader is referred to [25]. Radix-2^2 architectures are low-complexity, low area architectures where the multiplier stage is present only in every other stage of the FFT. However, it is to be noted that the discussions below are not limited to these FFT architectures and can be applied in general.

Note that the main components of the FFT architecture include butterfly units and complex multiplication operation. The decomposition of adders and multipliers are applicable to these components. However, in the case of FFT, we *defer* the final combination to the output of the last stage of the FFT. Hence, the butterfly and multiplier operations allow for carry propagation, overflow propagation and any approximations. The details of these decompositions are provided in Appendix B. Note from the decompositions that for correct recombination, the butterfly stages need to increase by one bit

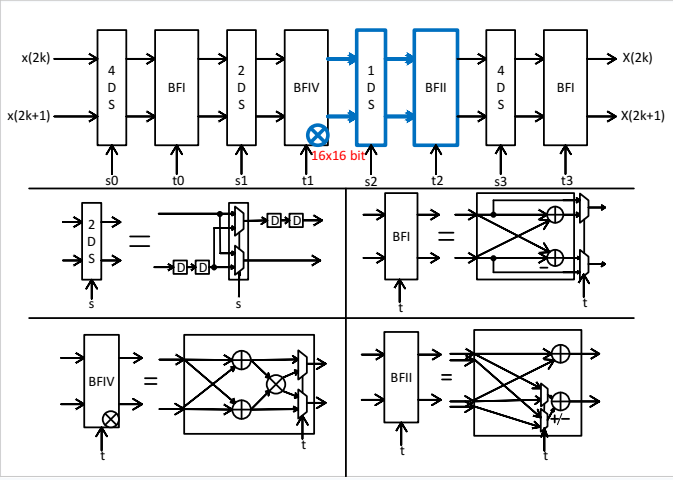


Fig. 4. 16-point, 2-parallel, folded, radix-2^2 real FFT architecture using a hybrid data-path. The components and paths marked in blue are complex data-paths.

at each stage. The decomposed 8-bit/4-bit data-path obtained after the required modifications is illustrated in Fig. 5.

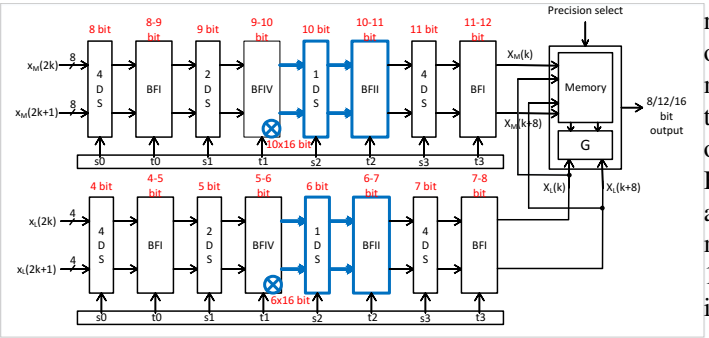


Fig. 5. Decomposition of 16-point, 2-parallel architecture into upper and lower data-paths with growing bit-widths at each stage. The components and paths marked in blue are complex data-paths. Note that varying precision multipliers are required. The controlpath is common for both the data-paths.

In this figure, the top datapath is used for computation of the first 8 bits of the output. After computation, these values are stored in memory. Next, when a 12-bit output is desired, the 8-bit output is accessed from memory and combined with the 4-bit output from the bottom datapath using a combination equation G to produce a 12-bit output. The 12-bit output now replaces the previously stored 8-bit outputs in memory. This process is continued to incrementally improve the precision of the outputs. The final combination equation G for every output $X(K)$ is given by:

$$G : X(k) = X_M(k) + sign\_extend(X_L(k), N) \qquad (3)$$

where the $sign\_extend$ function, extends the sign bit of the output of the bottom datapath by the number of bits computed in the top datapath (N).

V. Error and Energy Requirement Analysis

From Section IV, the decomposition and subsequent combination of multiplication operation introduces errors in the final output which is in addition to the error introduced due to lowering of precision of inputs. In this section, we model the errors due to these approximations using standard error models. We then contrast the error model with the reduction in resources due to incremental-precision architectures. This provides us with a method to determine if incremental-precision architectures are indeed beneficial to an application or not.

A. Modeling of error due to input approximation and approximate multipliers

Several studies have been performed to model noise propagated in fixed-point FFT architectures [27], [28]. A simple yet practical approach to model errors in fixed-point implementations of FFT is presented in [29]. Here, we apply similar concepts to the FFT architecture described previously i.e., a 2-parallel, folded, Radix-2^2 architecture. Note that the described error model can be used in general and is not limited to the architecture under discussion.

From [29], the addition and subtraction operations of butterfly units can be modeled as a gain unit of 2 followed by noise source due to rounding. However, if scaling is applied, the model is changed to a gain unit of $1/2$ followed by a noise source. For multipliers, recall that each multiplication operation is a complex multiplication consisting of four real multiplication operations. For example if the input is $a+jb$ and the multiplier is $Wr + jWi$, four multiplications are carried out and the sums $aWr - bWi$ and $bWr + aWi$ are generated. Hence, the total error due to multiplication at each output (real and imaginary) is equal to two times the error due to a single multiplication. Using these concepts, for the architecture of a 16-point FFT illustrated in Fig. 4, the error model is presented in Fig. 6.

In Fig. 6, V_{inp} is the variance of the quantization error due to fixed-width inputs. This can be measured using the *Fixed Point Designer ToolBox* of MATLAB and the command $sfi(xinp, prec, prec-1)$ where $prec$ is the total bit-width, $prec-1$ is the fractional bit-width and sfi converts the input $xinp$ to a signed fixed point variable. The inputs are assumed to be normalized and represented in the range of -1 to 1 as used in many practical applications. V_{rnd} is the rounding error after butterfly operation. The variance of error is dependent on the bit width of the adders and is proportional to 2^{-2prec} where $prec$ corresponds to the bit-width. For computing the variance of error in multiplication operation (V_{mult}), multipliers are coded in a hardware description language (Verilog) and simulated with 10,000 inputs. The variance of error between these simulations and infinite precision outputs of multiplication is represented as V_{mult}. The total error, V_{total}, which is a combination of V_{inp}, V_{rnd} and V_{mult}, is given by the expression in Fig. 6. Note that at each stage, the noise sources are added as linear sources. Due to scaling at each butterfly stage, the error is also scaled.

While decomposing the datapaths into top and bottom datapaths, we ensure that at each stage the scaled data for top datapath and carry information of bottom datapath are saved. This is achieved by construction of data-paths which grow in precision as described in Section IV-B. Hence, the

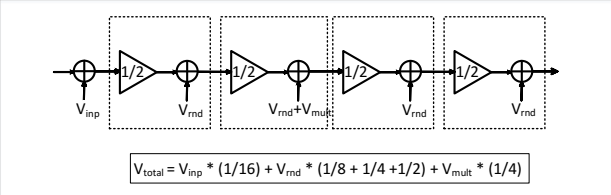


Fig. 6. Modeling of error in 16-bit datapath of a 16-point radix-2^2 FFT. Note that the butterfly structures are modeled by a gain of $1/2$ followed by rounding noise while the multipliers are modeled using an additional noise source at every alternate stage.

noise introduced at the output due to fixed-width input and rounding at butterfly stage remain the same as that of the error model described in Fig. 6. The change in the noise model occurs due to the multiplication operation. This is due to the approximation of multipliers as was discussed in Section IV. Hence the noise source V_{mult} of the direct implementation is modified to V_{mult10} and V_{mult6} as illustrated in Fig. 7. These variances are also obtained through simulation of RTL of the approximate multipliers of different bit-widths. For simplicity, this figure does not show the errors of butterflies and input error. The error due to approximate multipliers propagates to the outputs as V_{top} for the top datapath and V_{bot} for the bottom datapath. Total error V_{total} is now given by the equation in Fig. 7, where the multiplier error is modified to account for the approximate multiplier error and subsequent combination of top and bottom data-paths.

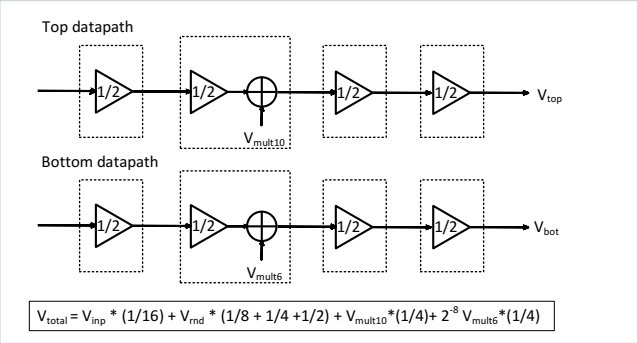


Fig. 7. Modeling of error in incremental precision architectures due to approximation. The noise due to addition operation V_{rnd} is included in the total error but not shown. The noise sources due to approximate multiplication propagate to the outputs and are added to the total error after combination of data-paths.

The described error model can be used for any incremental precision decomposition and for any N-point FFT. Fig. 8 describes the error obtained for an incremental-precision radix-2^2, 2-parallel, folded 1024-point FFT with various starting precision values, incremental step sizes and stopping precision. The figure also provides the plot of the error obtained with a direct implementation of the architecture at precisions of 4 bits to 16 bits. The starting precision and step increments are limited to powers of 2.

From this error model, we observe that for step increments of 2, the reduction in error stops after a precision of 12-bits is reached. This is because the errors due to approximation exceed the reduction in error due to increase of precision.

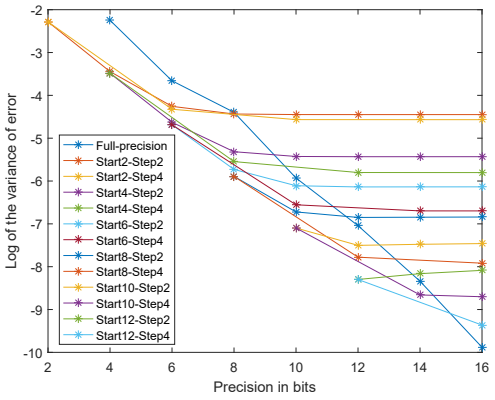


Fig. 8. Logarithm of the variance of error for different starting precision, step size and final precision for a 1024-point, 2-parallel, radix-2^2 FFT architecture.

Hence, only configurations upto 12 bits should be considered. Also observe that while starting at 6, 8 or 10 bits, we are able to reach the precision of 12 bit implementation. However, starting at 4 bits, we are able to only reach the precision of a 10-bit implementation. For step increments of 4 bits, the error reduces upto a precision of 16 bits. However, these increments are better suited for higher starting precisions such as 10 and 12 bits which are able to achieve almost the same accuracy as a direct 16-bit implementation.

B. Resource consumption and overhead for incremental-precision architectures

The main components of the folded FFT architecture include real multipliers, adders and delay elements. Hence, we consider the resource consumption in terms of these three main resources and understand the reduction of resources *vs.* overhead incurred for various configurations of the incremental-precision architecture. Note that only the major components of the datapath have been considered in our analysis. This is a fair estimate since controlpath of the design is small (less than 1% of the total architecture) and remains the same for both the direct and incremental implementation.

Considering an extension of the architecture as described in Fig. 4, we describe the number of resources required for a 16-bit precision 1024-point FFT in Table I. For real adders and multipliers, we list the number of resources as well the size of the components required. From Table I we observe that for starting precision of 10 bits or higher, the resource consumption is too high and comparable to that of a full 16-bit precision architecture. Hence, these incremental-precision architectures no longer provide sufficient resource reduction compared to accuracy improvement from Fig. 8. Hence, we consider starting precisions only upto 8 bits in our design.

To better understand the reduction in resources as well as the overhead incurred, we combine the area and power estimates of the top and bottom parts, and present area-power-timing estimates of different incremental-precision datapaths in Table II. The estimates of the total resources required for a 16-bit precision data-path are also reported. For power estimates, a similar strategy is applied by summing the power

TABLE I

HARDWARE COMPLEXITY OF A 1024-POINT FFT IN TERMS OF REAL ADDERS, REAL MULTIPLIERS AND REAL DELAY ELEMENTS. CELLS WITH TWO VALUES INDICATE THE RESOURCE COUNT AND CORRESPONDING BIT PRECISION IN PARENTHESES. THE MULTIPLIERS CORRESPOND TO 4 STAGES AND THE ADDERS CORRESPOND TO 10 STAGES.

Datapath	Starting precision	Steps	Real multipliers Count (Precision)	Real Adders Count (Precision)	Real Delay
	16	-	16 (16b)	28 (16b)	27264
Top	**4**	-	4 (6b), 4 (8b), 4(10b), 4(12b)	2(5b), 4(6b), 2(7b), 4(8b), 2(9b), 4(10b), 2(11b), 4(12b), 2(13b), 2(15b)	12876
	6	-	4 (8b), 4 (10b), 4(12b), 4(14b)	2(7b), 4(8b), 2(9b), 4(10b), 2(11b), 4(12b), 2(13b), 4(14b), 2(15b), 2(16b)	16284
	8	-	4 (10b), 4 (12b), 4(14b), 4(16b)	2(9b), 4(10b), 2(11b), 4(12b), 2(13b), 4(14b), 2(15b), 4(16b), 2(16b), 2(16b)	19180
	10	-	4 (12b), 4 (14b), 4(16b), 4(16b)	2(11b), 2(12b), 2(13b), 2(15b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b)	21552
	12	-	4 (14b), 4 (16b), 4(16b), 4(16b)	2(13b), 2(15b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b), 2(16b)	23872
Bottom	-	2	4 (4b), 4 (6b), 4(8b), 4(10b)	2(3b), 4(4b), 2(5b), 4(6b), 2(7b), 4(8b), 2(9b), 4(10b), 2(11b), 2(12b)	9468
	-	4	4 (6b), 4 (8b), 4(10b), 4(12b)	2(5b), 4(6b), 2(7b), 4(8b), 2(9b), 4(10b), 2(11b), 4(12b), 2(13b), 2(14b)	12876

TABLE II

ESTIMATES OF AREA, POWER CONSUMPTION AND TIMING OF CRITICAL PATH OF THE 1024-POINT INCREMENTAL-PRECISION ARCHITECTURE COMPARED TO A 16-BIT PRECISION ARCHITECTURE. THE RESOURCE CONSUMPTION VALUES AND PERFORMANCE NUMBERS ARE OBTAINED USING SYNTHESIS OF COMPONENTS WITH DESIGN COMPILER, 65NM TECHNOLOGY LIBRARY AND A CLOCK FREQUENCY OF 100MHZ

Starting precision	Steps	Area in um^2	Power in mW (top)	Power in mW (bottom)	Timing in ns (top)	Timing in ns (bottom)
16	-	337450	29.0134	-	9.85	-
2	2	223790	9.8392	9.8392	9.421	9.421
	4	264160	9.8392	13.408	9.421	9.469
4	2	264160	13.408	9.8392	9.469	9.421
	4	304540	13.408	13.408	9.469	9.469
6	2	305140	17.0053	9.8392	9.538	9.421
	4	345520	17.0053	13.408	9.538	9.469
8	2	346160	20.2719	9.8392	9.826	9.421
	4	386540	20.2719	13.408	9.826	9.469

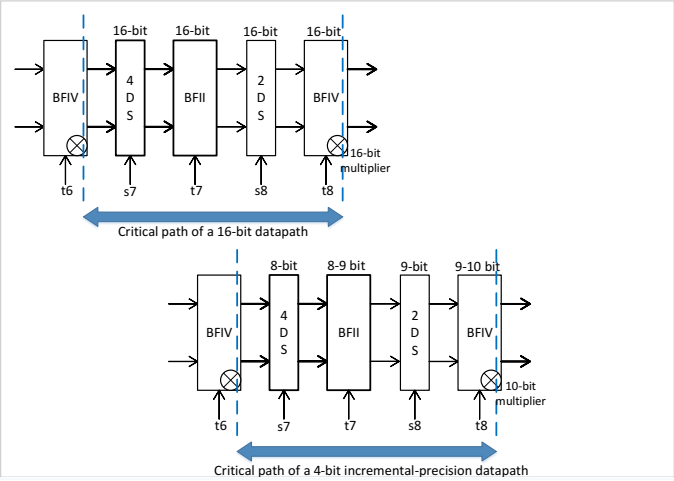


Fig. 9. Identification of critical path of a 1024-point 16-bit precision FFT and comparison with the critical path of a 1024-point incremental-precision FFT starting at 4 bits. Note that stages 7 and 8 form part of the critical path for incremental- precision architectures due to a growing data-path.

consumption of all the components. However, in this case the power consumption of the top and bottom data-paths are reported separately since it is assumed that only one of the data-paths would be active at any instant of time. The area and power consumption of individual components are obtained through synthesis. For this, Design Compiler synthesis tool is used and all components are synthesized at a frequency of $100MHz$ and a technology library of $65nm$. Note that this is a fairly good estimate of the resource consumption since the most energy hungry components are accounted for in our calculations.

To obtain timing overheads for the design, the critical path is identified and simulated for different precisions. A pipelined FFT architecture is considered in this paper with pipeline stages after every multiplier. Hence, it is understood that the critical path lies between two multiplier stages. For a 16-bit data-path, the components between each multiplier are constant and hence the critical path can be considered to be composed of BFII and BFIV of Fig. 4. This is indicated in Fig. 9. However, due to a growing data-path, the incremental-precision FFT architecture of Fig. 5 has a varying critical path. This is also indicated in Fig. 9 using a datapath with starting precision of 4-bit as an example. Using the identified paths and simulation results, timing requirements of Table II can be obtained.

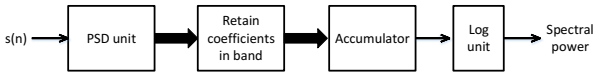
VI. CASE STUDY: SEIZURE DETECTION USING EEG SIGNALS

Seizure detection is a binary classification problem which involves classifying between ictal period (when seizure occurs) and interictal period (between seizures) of electroencephalogram (EEG) signals. We label the ictal samples as class 1 and the inter-ictal samples as class 0. For this problem, several distinguishing features can be used for classification. However, it has been observed that use of spectral powers in specific frequency bands of the EEG signals as features is known to demonstrate high separability between the two classes [30]. Implementations of seizure detection and prediction using ratios of spectral power have been presented in [22], [23]. Because of the high accuracy achieved by this system using a low complexity architecture, we use the same algorithm and architecture for our study but with required modifications for incremental-precision systems. For the extraction of spectral powers, an energy hungry power spectral density computation unit is required. Hence, this application is an ideal candidate for implementation using the incremental-precision approach.

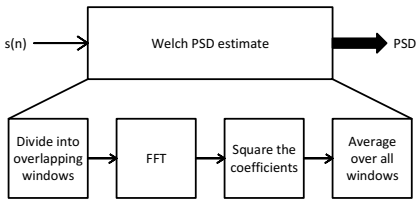
A. Dataset and System description

For our analysis, we use the database from UPenn and Mayo Clinic's Seizure Detection Challenge [31]. The feature extraction unit works on EEG data segments labeled as ictal or interictal each of duration $1s$. The sampling frequency is $5000Hz$ which results in 5000 samples to be analyzed for every data segment. 104 features are extracted from each electrode which include absolute spectral power, ratios of spectral powers and relative spectral powers in frequency bands of relevance [23]. Three of the most relevant electrodes are selected using the results published in [23] to generate a total of 312 features which are then ranked according to importance using Classification and Regression Tree (CART). Using these rankings, feature selection is performed to obtain the three best features for each data segment. We replace complex SVM classifiers originally used in [23] with Logistic Regression based classifiers. LR classifiers are known to provide good probability estimates [32] and have low hardware complexity [24]. Since the data is imbalanced, the number of positive samples is low compared to the number of negative samples.

A simplified hardware architecture for extracting spectral power values and power ratios has been proposed in [22]. The main components of the feature extraction unit are illustrated in Fig. 10. The extracted spectral powers are converted to logarithm units which makes it easier to perform addition and subtraction to obtain absolute, relative and ratios of spectral powers. The detailed components required for the power spectral density unit are also shown in this figure. We note that the PSD unit is based on the Welch PSD algorithm which uses overlapped data segments and applies fast Fourier transform (FFT) on each of these segments to extract frequency components of signals [33], [34]. We also observe that FFT is the largest component in the feature extraction unit. By application of the incremental-precision FFT architecture described in Section IV-B, we demonstrate reduction in resource consumption.



(a) Feature extraction



(b) Power spectral density estimation

Fig. 10. Low complexity implementation of feature extraction unit for computation of spectral powers. The Power Spectral density can be implemented using a Welch PSD estimate whose main component is an FFT.

B. Experimental setup

The errors due to approximation as well as the resource consumption analysis have been discussed in Section V. From Fig. 8, we obtain the errors for the various configurations of incremental-precision FFT. The errors are then incorporated into the feature extraction system to generate features of varying precision. Classification is performed using the proposed incremental-precision algorithm for various starting precisions, step sizes and threshold values. To model resource consumption, the estimates from Table II are applied to these configurations. For every experiment performed, the data is randomly separated into training and test data with 80% of the data forming the training set and 20% of the data forming the testing set. The experiments performed are based on varying the controllable parameters of the design as follows:

- Even though the algorithm has a stopping criteria dependent on the number of samples reselected, the maximum number of stages allowed can be restricted to a certain value. We experiment with two configurations with maximum values of 3 and 4.
- The maximum allowed misclassification error rates which in turn determine the maximum and minimum threshold values can be varied. When maximum number of stages is 3, the allowed class-0 misclassification error values ($M1$ and $M3$) and the allowed class-1 misclassification error ($M2$ and $M4$) are varied between 0% to 25% in increments of 5%. For simplicity, the values of allowed misclassification error at each stage can be equal, i.e., $M1 = M2$ and $M3 = M4$ etc. This results in 36 configurations. When the maximum number of stages allowed is 4, acceptable misclassification errors ($M1$, $M2$, $M3$, $M4$, $M5$ and $M6$) are varied in the range of 0% to 10% in increments of 5%, resulting in 27 configurations. These configurations are termed as configurations of M values with M ranging from 1 to 36 for a 3-stage algorithm and 1 to 27 for a 4-stage algorithm, respectively.
- The starting precision values are varied between 2 bits to 8 bits and step size is varied between 2 and 4 bits. The starting precision of 10 and 12 bits are not used since the error *vs* energy trade-off for these precisions is not favorable as observed from Fig. 8 and Table II.

To measure the improvement in accuracy with respect to varying parameter values, we use both training and test accuracy. Since the data is imbalanced, we use F1-score as a measure for training accuracy. The F1-score is given by Equation (4), where $precision = TP/(TP + FP)$ and $recall = TP/(TP + FN)$ and $TP =$ True Positive, $FP =$ False Positive and $FN =$ False Negative.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

The test accuracy is better represented in terms of the *Specificity* and *Sensitivity* of the data. While specificity is the true negative rate of the test data, sensitivity corresponds to the true positive rate.

For measuring area, power, timing and energy overheads, the estimates calculated in Table II are utilized. Assume that

the number of samples selected for classification in each stage is stored as t_1, t_2, t_3 etc. Equation (5) can then be used to calculate the overall overhead or reduction with respect to a full precision system. In this equation, $Estimate_{top}$, $Estimate_{bot}$ and $Estimate_{full}$ refer to the corresponding area, power or timing estimates of the top, bottom and full-precision data-path, respectively. For energy calculations, $Estimate_{top} = Power_{top} \times Timing_{top}$ where $Power_{top}$ is the power estimate of top datapath and $Timing_{top}$ is the timing estimate of the top datapath. Similarly, $Estimate_{bot} = Power_{bot} \times Timing_{bot}$ where $Power_{bot}$ and $Timing_{bot}$ are the power and timing estimates of the bottom data-path, respectively. Note that the obtained ratio values are calculated per test sample.

$$Ratio = \frac{Estimate_{top}.t1 + Estimate_{bot}.t2 + Estimate_{bot}.t3}{Estimate_{full}.(t1 + t2 + t3)} \quad (5)$$

VII. Results

Based on the proposed incremental-precision classification algorithm, derived error-energy models and the experimental setup described in the previous section, we present results for several configurations. The detailed experimental results are based on Patient 7 of the dataset whose most relevant electrodes have been identified in [23]. The seizure detection example serves as a demonstration of both the multi-level classification algorithm as well as incremental-precision architecture. Overheads of the system are also presented.

A. Training accuracy

First, we consider the effect of lowering the precision of data as well as the application of LR classifier. The error values obtained for the full-precision system in Fig. 8 are used to generate features for this experiment. The results are reported in the plots of Fig. 11. From this figure we observe that as the precision of data is increased, both training and testing accuracy improve. However, it can also be seen that LR classifier is not able to perform well since the maximum sensitivity that can be reached even with the highest precision is about 72%. For high accuracies, SVM classifiers which are energy consuming would be necessary [23] if an incremental-precision algorithm is not employed.

We consider the application of incremental precision algorithm by starting with a precision of 2 bits and incrementing it in step sizes of both 2 and 4. The results for the training accuracy at each increment of the algorithm are presented in Fig. 12. From the two subplots we observe that the training accuracy improves significantly from stage 1 to stage 3. Also, increments of step size 4 have higher final accuracies compared to step sizes of 2.

Extending the previous experiment to data-widths ranging from 2 to 8 bits and step sizes of 2 and 4, we obtain the results illustrated in Fig. 13. Also, the observations for the algorithm when maximum stages is 4 are presented. It can be observed that the accuracy is highest for the algorithm consisting of 3 stages and a step size of 4.

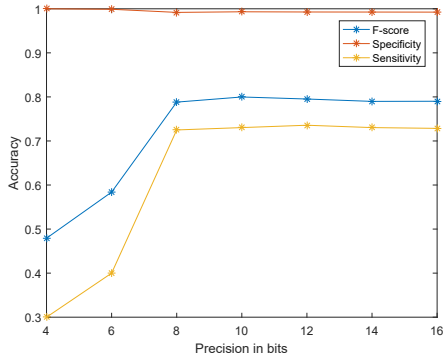
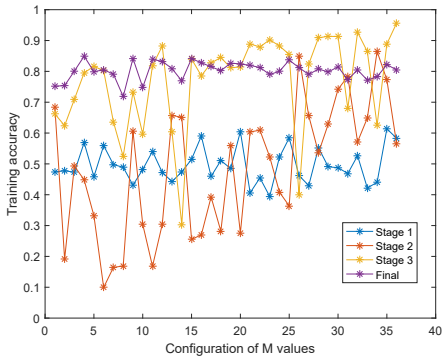
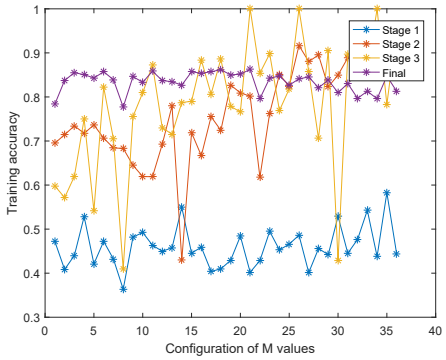


Fig. 11. Effect of using varying precision features on training accuracy (reported as F1-score) and testing accuracy (reported as Specificity and Sensitivity).

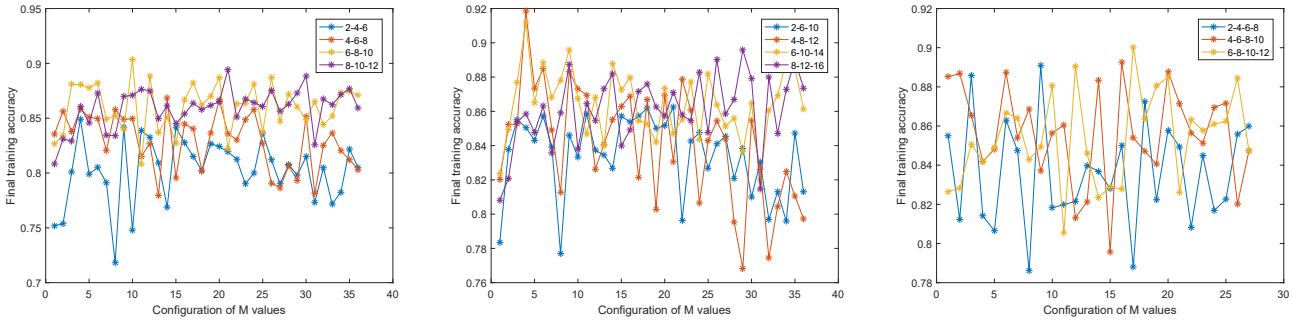


(a) Starting precision = 4 bits, Maximum stages = 3, Step size = 2 bits



(b) Starting precision = 4 bits, Maximum stages = 3, Step size = 4 bits

Fig. 12. Improvement in training accuracy at different stages of the incremental-precision algorithm for various configurations of M values. The x-axis represents different configurations ranging from 1 to 36 with varying maximum allowed misclassification errors $M1$, $M2$, $M3$ and $M4$ (termed as M values).



(a) Starting precision = 2,4,6,8, Step size = 2, Maximum stages = 3

(b) Starting precision = 2,4,6,8, Step size = 4, Maximum stages = 3

(c) Starting precision = 2,4,6, Step size = 2, Maximum stages = 4

Fig. 13. Final training accuracy for different starting precision, step size, maximum allowed stages and configurations of M value.

B. Testing accuracy and Power reduction

Next, we use the configurations discussed previously to understand the reduction in power *vs.* accuracy achieved. Depending on the number of samples that go through the testing process at each precision, the power consumption is calculated from Table II and Equation (5). Since the degradation in precision does not affect specificity, these values are not reported. However, it is ensured that high specificity values are maintained. The plots for trade-off of power reduction (as a ratio) *vs.* sensitivity of testing are presented in Fig. 14.

From these plots, we observe several configurations which provide reduction in power consumption while maintaining high accuracy values (bottom right corner of the plots). These plots demonstrate the effectiveness of the incremental precision algorithm. However, it is also important to understand the overhead associated with incremental-precision architectures which are studied in the next subsection for selected configurations.

C. Overhead of selected configurations

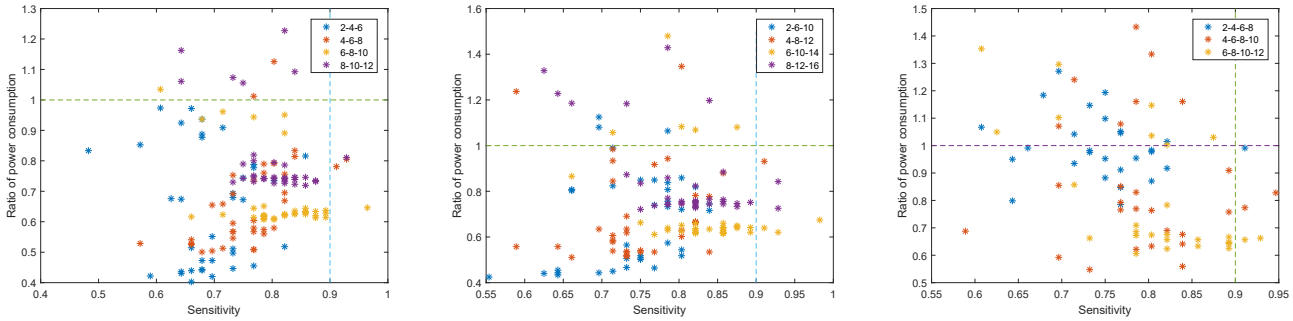
Several configurations selected from the power ratio *vs.* sensitivity trade-off plots of previous section are considered and detailed experimental results are presented in Table III. Configuration values including maximum allowed misclassification error rates $M1, M2$, etc., initial and final F1-scores, specificity and sensitivity are also reported. The power, area and timing ratios are reported as percentages of reduction with respect to a full-precision system. Note that there could be a reduction or overhead in area consumption depending on the configuration.

From Table III, we observe that there are several configurations that provide reduction in energy while maintaining high sensitivity values. The configuration with starting precision of 6 bits and step increments of 4 bits results in sensitivity values of 98% while reducing the energy consumption by almost 35%. Also note that a configuration of starting precision of 2 bits and step increment of 2 bits could provide almost 50% reduction in energy while providing sensitivity of 82%. All other reported configurations provide sensitivity values and energy reductions between that of these two configurations. Note that there is a cost associated with the incremental-precision algorithm in terms of the timing overheads. This is because if selected for the consequent stages, the feature extraction process repeats, increasing the computation time for these samples. However, note that for most configurations, the overhead is not too high. This can be attributed to the fact that most of the samples get processed in low precision which occurs faster than high precision computation. The additional memory overhead for this system is not presented in the table. However, it is understood that the 1024-point incremental-precision FFT should have a memory of approximately 2 KB (1024 outputs x 16 bit) to store intermediate values.

Finally, Table IV compares the test accuracy results of our proposed algorithm with the results from [18], [19], [23]. This table shows that the proposed algorithm maintains high accuracy values similar to other algorithms. For energy consumption comparison, we compare the results of our work with [22]. This is a fair comparison since the features used in both works are spectral energy ratios. Note that even though other seizure detection algorithms exist which claim lower energy, the goal of our work is to demonstrate applicability of the incremental-precision algorithm and architecture. Our ideas can be applied to other seizure detection or classification setups to result in configurations with lower energy while maintaining high accuracy.

VIII. Conclusions and Future work

We have presented a multi-level classification approach and corresponding incremental-precision datapath architecture methodology. The proposed algorithm and architecture are demonstrated on a seizure detection application which uses spectral powers and ratio of spectral powers as features. The most energy hungry component of spectral power calculation, the FFT architecture, was decomposed using datapath decomposition ideas. The resulting configurations were proven to reduce energy consumption while maintaining high test accuracy values using models which estimate error and resource consumption. The overheads in terms of area increase or timing overhead are also presented. It should be noted that the energy savings are demonstrated to a first-order approximation from synthesis of component parts. Future work will



(a) Starting precision = 2,4,6,8, Step size = 2, Maximum stages = 3

(b) Starting precision = 2,4,6,8, Step size = 4, Maximum stages = 3

(c) Starting precision = 2,4,6, Step size = 2, Maximum stages = 4

Fig. 14. Ratio of power consumption *vs* Sensitivity of test data for different starting precision, step size, maximum stages and configurations of M value. Note that configurations towards the right bottom corner of these plots have high sensitivity and low power ratios.

TABLE III
PERFORMANCE OF SELECTED CONFIGURATIONS WITH RESPECT TO FINAL TRAINING ACCURACY, SPECIFICITY, SENSITIVITY, POWER REDUCTION, AREA REDUCTION, TIMING OVERHEADS AND ENERGY CONSUMPTION

Precision increments	M1/M2 (Step 1)	M3/M4 (Step 2)	M5/M6 (Step 3)	F1-score (Step 1)	F1-score (Final)	SP	SS	Power reduction	Area reduction	Timing overhead	Energy requirement
16-0-0	-	-	-	-	0.790	0.992	0.728	0%	0%	0%	100%
6-10-14	5%	10%	-	0.786	0.874	0.988	0.982	32.59%	- 2.39%	15.13%	65.21%
6-8-10	5%	10%	-	0.789	0.855	0.985	0.964	35.41%	9.57%	13.68%	62.47%
4-6-8-10	0%	10%	10%	0.591	0.906	0.992	0.946	17.13%	21.72%	99.52%	79.49%
6-8-10	15%	0%	-	0.771	0.874	0.992	0.893	36.28%	9.57%	11.23%	61.64%
8-10-12	10%	20%	-	0.791	0.860	0.989	0.875	26.76%	- 2.58%	9.27%	72.93%
4-8-12	15%	25%	-	0.634	0.807	0.975	0.839	46.57%	9.75%	11.15%	51.37%
6-10-14	15%	25%	-	0.794	0.862	0.991	0.929	37.84%	- 2.39%	4.21%	60.16%
6-10-14	20%	5%	-	0.779	0.859	0.994	0.911	36.01%	- 2.39%	8.03%	61.93%
8-12-16	10%	25%	-	0.839	0.887	0.995	0.928	27.40%	- 14.54%	5.49%	72.35%
6-8-10-12	10%	0%	0%	0.771	0.879	0.992	0.928	33.68%	9.57%	18.57%	64.13%
6-8-10-12	10%	10%	0%	0.783	0.863	0.983	0.893	37.6%	9.57%	7.70%	60.44%
2-4-6	20%	0%	-	0.559	0.848	0.995	0.821	48.26%	33.68%	45.91%	49.48%

TABLE IV
COMPARISON OF TESTING ACCURACY AND ENERGY CONSUMPTION OF PROPOSED INCREMENTAL-PRECISION SYSTEM APPLIED TO SEIZURE DETECTION WITH SIMILAR APPROACHES FROM LITERATURE

Comparison of test accuracy		
Algorithm	**Specificity**	**Sensitivity**
Ratio of spectral power + SVM classifier [23]	99.90	100
Ratio of spectral power + LR classifier	99.24	72.86
[18]	99.19	91.29
[19]	94.89	91.72
Proposed approach	98.88	98.20
Comparison of energy consumption		
Algorithm	**Energy of feature selection**	**Energy of classifier**
Ratio of spectral power + SVM classifier [23]	226.26 nJ (Full-precision FFT)	∼60 uJ [35]
Proposed approach	∼ 147.07 nJ (65% of full-precision FFT)	∼0.15 uJ [35]

be directed towards synthesis of the complete system. Since the proposed incremental-precision algorithm and incremental-precision architectures are not limited to the demonstrated application, future work will involve applying these ideas to other machine learning applications. Application of data-path decomposition approaches to create incremental-precision architectures for other approximate computing paradigms is also a topic for further research.

IX. APPENDIX A

The idea of decomposition of fixed-width addition is presented in Fig. 15. From Equation (1), the input operands are split such that the most significant part of the operand is composed of N bits. The operands x_1 and x_2 are signed numbers. Hence, padding with zero is necessary to perform addition on the lower bits. Also, a carry bit is generated out of the lower adder which needs to be propagated to the final combination equation.

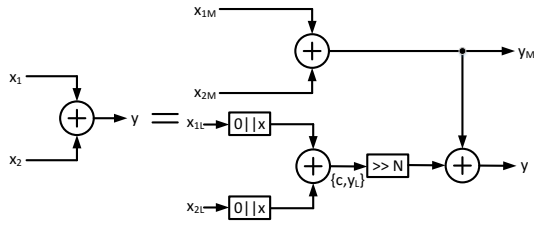


Fig. 15. Data-path decomposition of fixed-width adder operating on signed numbers x_1 and x_2 and subsequent recombination to produce y. The block $[0||x]$ indicates the operation of appending x with one leading zero.

Several techniques to fixed-width multiplication have been proposed in literature [37], [38]. In this paper, we propose tp

decompose fixed-width multipliers based on the design presented in [38]. The decomposition of the fixed-width multiplier is illustrated in Fig. 16. From Equation (2), observe that only the operand x is decomposed. The upper multiplier produces an output of size N bits along with a carry bit C. Similarly, the lower multiplier produces an output of N bits. Depending on the first bit of the lower multiplication output, addition or subtraction has to be performed as part of recombination. Hence, we check the sign and if required invert the output of the lower multiplier. The details of how to decompose the multiplier is discussed next. The carry signals need to be propagated to the final combination equation.

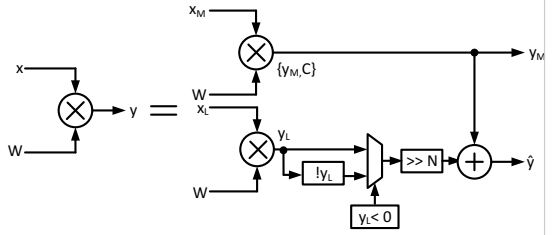


Fig. 16. Data-path decomposition of fixed-width multiplier operating on x and coefficient W and subsequent recombinaton to produce approximate output $\hat{y}$.

As an example of decomposition of the multiplier, we discuss an 8×8 bit fixed-width multiplication split into two 4×8 bit multiplications. Fig. 17 represents the required partial products and summation for a fixed-width 8×8 bit multiplication on inputs x and y. The multiplication performed here is based on modified Booth's algorithm where the operand y is first recoded to y' whose values can be -2,-1,-0,1 or 2. If y is a value greater 1, a second value y'' is set to 1. This architecture is presented in [38] and the details of calculation of carry signals $carry\_0$ and $carry\_1$ based on y'' can be referenced from the paper.

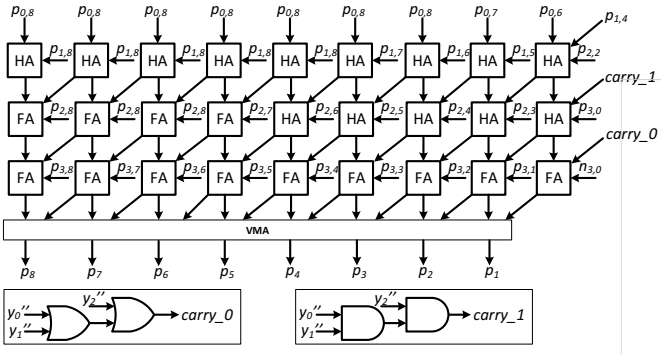


Fig. 17. Architecture of an 8×8 fixed-width multiplier based on modified Booth multiplication and carry generation.

To decompose the multiplier into two, we split the computation required for the most significant and least significant bits of the multiplier. The decomposed multiplier along with the new carry signals are presented in Fig. 18. Note that there is additional approximation involved in the computation of both the most significant and least significant bits of the multiplication operation. Hence, the final output of multiplication ($\hat{y}$) is an approximation of the final $8-bit$ output. The multiplier introduces an error in the data-path decomposition which needs to be accounted for while calculating the improvement in accuracy from stage to stage.

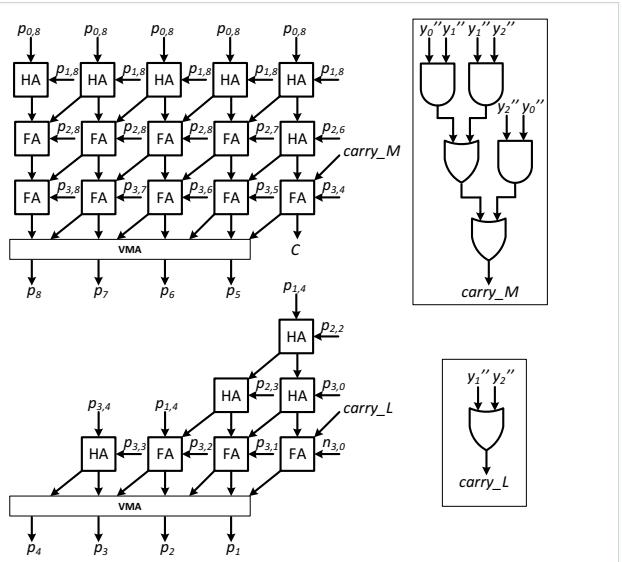


Fig. 18. Splitting of an 8×8 bit multiplier into two 4×8 bit multipliers by approximation. The carry information from the top multiplier is merged with the bottom multiplier during recombination.

X. Appendix B

In this section, we continue the decomposition to larger components such as butterfly units and complex multipliers. The main challenge in these decompositions is to defer the combination to the end of computation such that the two data-paths are truly decomposed. The decomposition of the butterfly unit is presented in Fig. 19. The butterfly operation involves addition and subtraction followed by a shift to avoid overflow. the butterfly can be decomposed using the concept of adder decomposition as presented in Fig. 15. However, there is an additional bit from the upper data path due to shifting (s_1 and s_2) along with the carry information from the lower data path (c_1 and c_2). This means that the top and bottom butterflies need to grow at each stage of the FFT.

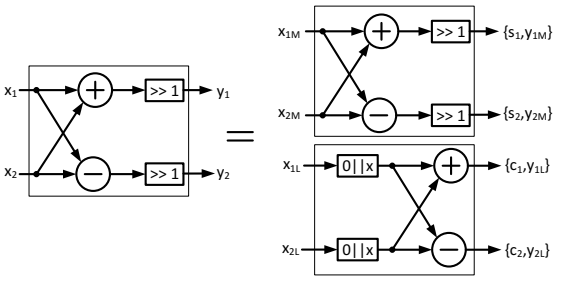


Fig. 19. Data-path decomposition of the butterfly unit which is composed of addition, subtraction and shift operations. Recombination of outputs is deferred to the last stage.

The decomposed multiplier presented in Fig. 16 is utilized to construct the complex multiplier required for the FFT

operation. The component consisting of four multiplication operations and subsequent addition operations is presented in Fig. 20. The carry signals from the top multiplier are included in the addition operation.

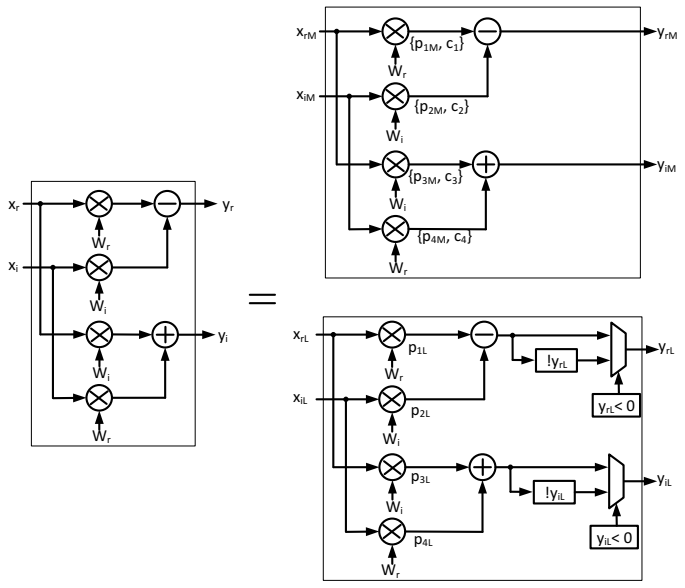


Fig. 20. Data-path decomposition of the complex multiplication unit composed of four real multiplications, addition and subtraction operations.

XI. Acknowledgment

The incremental-precision multi-level classification approach is inspired by the minimum uncertainty sample elimination feature ranking algorithm [36].

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

[2] S. Zhang and N. R. Shanbhag, "Embedded Algorithmic Noise-Tolerance for Signal Processing and Machine Learning Systems via Data Path Decomposition," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3338–3350, 2016.

[3] S. Koteshwara and K. K. Parhi, "Low-Energy Architectures of Linear Classifiers for IoT Applications using Incremental Precision and Multi-Level Classification," in *Proceedings of the Great Lakes Symposium on VLSI 2018*. ACM, 2018.

[4] S. H. Nawab, A. V. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig, "Approximate signal processing," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 15, no. 1-2, pp. 177–200, 1997.

[5] J. Park, J. H. Choi, and K. Roy, "Dynamic bit-width adaptation in DCT: an approach to trade off image quality and computation energy," *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 18, no. 5, pp. 787–793, 2010.

[6] M. Shoaib, N. Jha, and N. Verma, "A low-energy computation platform for data-driven biomedical monitoring algorithms," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*. IEEE, 2011, pp. 591–596.

[7] D. Kim, J. Kung, and S. Mukhopadhyay, "A Power-Aware Digital Multilayer Perceptron Accelerator with On-Chip Training based on Approximate Computing," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[8] A. Suleiman and V. Sze, "Energy-efficient HOG-based object detection at 1080HD 60 fps with multi-scale support," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 2014, pp. 1–6.

[9] M. Imani, M. Masich, D. Peroni, P. Wang, and T. Rosing, "CANNA: Neural network acceleration using configurable approximation on GPGPU," in *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*. IEEE, 2018, pp. 682–689.

[10] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[11] R. E. Schapire, "Theoretical views of boosting and applications," in *Algorithmic Learning Theory: 10th International Conference, ALT'99, Tokyo, Japan, December 1999. Proceedings*. Springer, 1999, p. 13.

[12] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2012.

[13] S. A. Salehi, R. Amirfattahi, and K. K. Parhi, "Pipelined architectures for real-valued FFT and hermitian-symmetric IFFT with real datapaths," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 8, pp. 507–511, 2013.

[14] J. M. Winograd and S. H. Nawab, "Incremental refinement of DFT and STFT approximations," *IEEE Signal Processing Letters*, vol. 2, no. 2, pp. 25–27, 1995.

[15] J. M. Winograd, S. H. Nawab, and A. V. Oppenheim, "FFT-based incremental refinement of suboptimal detection," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996*, vol. 5. IEEE, 1996, pp. 2479–2482.

[16] S. Lee and A. Gerstlauer, "Fine grain word length optimization for dynamic precision scaling in DSP systems," in *Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on*. IEEE, 2013, pp. 266–271.

[17] V. Bajaj and R. B. Pachori, "Epileptic seizure detection based on the instantaneous area of analytic intrinsic mode functions of EEG signals," *Biomedical Engineering Letters*, vol. 3, no. 1, pp. 17–21, 2013.

[18] L. M. Patnaik and O. K. Manyam, "Epileptic EEG detection using neural networks and post-classification," *Computer methods and programs in biomedicine*, vol. 91, no. 2, pp. 100–109, 2008.

[19] Q. Yuan, W. Zhou, Y. Liu, and J. Wang, "Epileptic seizure detection with linear and nonlinear features," *Epilepsy & Behavior*, vol. 24, no. 4, pp. 415–421, 2012.

[20] Y. Park, L. Luo, K. K. Parhi, and T. Netoff, "Seizure prediction with spectral power of EEG using cost-sensitive support vector machines," *Epilepsia*, vol. 52, no. 10, pp. 1761–1770, 2011.

[21] Z. Zhang and K. K. Parhi, "Seizure prediction using polynomial SVM classification," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 2015, pp. 5748–5751.

[22] ——, "Low-Complexity Seizure Prediction From iEEG/sEEG Using Spectral Power and Ratios of Spectral Power," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 693–706, 2016.

[23] Z. Zhang and K. K. Parhi, "Seizure detection using regression tree based feature selection and polynomial SVM classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 6578–6581.

[24] A. Page, S. P. T. Oates, and T. Mohsenin, "An ultra low power feature extraction and classification system for wearable seizure detection," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 2015, pp. 7111–7114.

[25] M. Ayinala and K. K. Parhi, "FFT architectures for real-valued signals based on radix-2^3 and radix-2^4 algorithms," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 9, pp. 2422–2430, 2013.

[26] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. Wiley, New York, 1999.

[27] W.-H. Chang and T. Q. Nguyen, "On the fixed-point accuracy analysis of FFT algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4673–4682, 2008.

[28] T. Thong and B. Liu, "Fixed-point fast Fourier Transform Error Analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 6, pp. 563–573, 1976.

[29] R. B. Perlow and T. C. Denk, "Finite wordlength design for VLSI FFT processors," in *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001.*, vol. 2. IEEE, 2001, pp. 1227–1231.

[30] M. Bandarabadi, C. A. Teixeira, J. Rasekhi, and A. Dourado, "Epileptic seizure prediction using relative spectral power features," *Clinical Neurophysiology*, vol. 126, no. 2, pp. 237–248, 2015.

[31] Upenn and mayo clinic's seizure detection challenge. USA. [Online]. Available: https://www.kaggle.com/c/seizure-detection
[32] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International conference on Machine Learning*. ACM, 2005, pp. 625–632.
[33] P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.
[34] K. K. Parhi and M. Ayinala, "Low-complexity Welch power spectral density computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 1, pp. 172–182, 2014.
[35] A. Page, C. Sagedy, E. Smith, N. Attaran, T. Oates, and T. Mohsenin, "A flexible multichannel EEG feature extractor and classifier for seizure detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 2, pp. 109–113, 2015.
[36] Z. Zhang and K. K. Parhi, "MUSE: Minimum uncertainty and sample elimination based binary feature selection," *IEEE Transactions on Knowledge and Data Engineering Large Scale Integration (VLSI) systems*, DOI: 10.1109/TKDE.2018.2865778, 2018.
[37] S.-M. Kim, J.-G. Chung, and K. K. Parhi, "Low error fixed-width CSD multiplier with efficient sign extension," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, no. 12, pp. 984–993, 2003.
[38] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 522–531, 2004.

Sandhya Koteshwara (S'16) received the B.E. degree in Electronics and Communication from Visvesvaraya Technological University, Belgaum, India, in 2010 and M.S. degree in electrical engineering from University of Minnesota, Twin Cities, USA in 2014. She is currently working towards a Ph.D. degree at the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, USA.

Her current research interests include hardware security, low power architectures for cryptographic algorithms, and approximate computing.

Keshab K. Parhi (S'85-M'88-SM'91-F'96) received the B.Tech. degree from Indian Institute of Technolgy, Kharagpur, India, in 1982, the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, PA, USA, in 1984, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 1988.

He has been with the University of Minnesota, Minneapolis, MN, USA, since 1988, where he is currently a Distinguished McKnight University Professor and Edgar F. Johnson Professor in the Department of Electrical and Computer Engineering. He has published over 600 papers, is the inventor or co-inventor of 29 patents, has authored the textbook VLSI Digital Signal Processing Systems (New York, NY, USA: Wiley, 1999), and coedited the reference book Digital Signal Processing for Multimedia Systems (Boca Raton, FL, USA: CRC Press, 1999). His research interests include the VLSI architecture design and implementation of signal processing, communications and biomedical systems, error control coders and cryptography architectures, high-speed transceivers, stochastic computing, secure computing, and molecular computing. He is also currently working on intelligent classification of biomedical signals and images, for applications such as seizure prediction and detection, schizophrenia classification, biomarkers for mental disorders, brain connectivity, and diabetic retinopathy screening.

Dr. Parhi has served on the Editorial Boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I AND PART II, the IEEE TRANSACTIONS ON VLSI SYSTEMS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE Signal Processing Magazine, and served as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I from 2004 to 2005. He currently serves on the Editorial Board of the Journal of Signal Processing Systems (Springer). He has served as the Technical Program Co-Chair of the 1995 IEEE VLSI Signal Processing Workshop and the 1996 Application Specific Systems, Architectures, and Processors conference, and as the General Chair of the 2002 IEEE Workshop on Signal Processing Systems. He was the Distinguished Lecturer of the IEEE Circuits and Systems Society from 1996 to 1998. He served as a Board of Governors Elected Member of the IEEE Circuits and Systems Society from 2005 to 2007. He is the recipient of numerous awards including the 2017 Mac Van Valkenburg award, the 2012 Charles A. Desoer Technical Achievement award and the 1999 Golden Jubilee medal, from the IEEE Circuits and Systems society, the 2013 Distinguished Alumnus Award from IIT Kharagpur, the 2013 Graduate/Professional Teaching Award from the University of Minnesota, the 2004 F. E. Terman award from the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, and the 2001 IEEE W. R. G. Baker Prize Paper Award. He is a Fellow of the AAAS.