DOI: 10.21917/ijme.2026.0390

AN INTELLIGENT FPGA WEARABLE CO-DESIGN FRAMEWORK USING NEURAL SURROGATE OPTIMIZATION FOR ADAPTIVE REAL-TIME SYSTEMS

Avulla Rajinidevi[1] and S. Vamshi Krushna[2]
[1]*Department of Information Technology, Vignan's Institute of Management and Technology for Women, India*
[2]*Department of Computer Science and Engineering (Data Science), Vignana Bharathi Institute of Technology, India*

Abstract

Wearable healthcare and intelligent monitoring systems require rapid data processing, low power consumption, and adaptive hardware configurations to satisfy the demands of real-time applications. The increasing complexity of field-programmable gate array (FPGA) wearable platforms has created a need for efficient hardware and software co-design strategies that can optimize performance without excessive computational overhead. However, conventional FPGA optimization approaches depend on iterative design-space exploration that requires substantial development time and computational resources, which limits the deployment of adaptive wearable devices in dynamic environments. To address this limitation, this study proposes the Adaptive Neural Surrogate Co-Design Optimization Framework (ANSCOF), a novel artificial intelligence driven methodology for real-time FPGA wearable systems. The ANSCOF framework integrates neural surrogate prediction, adaptive resource allocation, hardware-aware optimization, and software performance estimation within a unified co-design architecture. The neural surrogate model predicts hardware resource utilization, latency, energy consumption, and throughput before FPGA synthesis, which significantly reduces exploration complexity. The framework employs a hybrid optimization strategy that selects optimal hardware configurations and software execution parameters for adaptive wearable applications. Experimental evaluation demonstrates that the proposed ANSCOF framework achieves superior performance compared with existing FPGA optimization approaches. The framework obtains 98.1% accuracy, reduces processing latency to 16.1 ms, decreases energy consumption to 44.7 mJ, improves FPGA resource utilization efficiency to 96.4%, and achieves a throughput of 701 samples/sec. The proposed method reduces design exploration complexity and provides an efficient solution for real-time adaptive FPGA wearable devices.

Keywords:

FPGA Wearable Devices, Neural Surrogate Models, Hardware Software Co-Design, Real-Time Reconfiguration, Edge Artificial Intelligence

1. INTRODUCTION

The rapid advancement of wearable computing technologies has transformed modern healthcare, fitness monitoring, industrial safety, and smart environment applications. Wearable devices continuously acquire physiological and environmental information and provide real-time responses that support intelligent decision-making. The increasing demand for low-latency processing, high computational efficiency, and energy-aware operation has accelerated the adoption of field-programmable gate arrays (FPGAs) within wearable platforms. FPGA technology offers hardware flexibility, parallel processing capability, and reconfigurable architectures that make it suitable for wearable systems that require adaptive functionality under varying operational conditions [1–3].

Recent developments in artificial intelligence have further enhanced the capabilities of wearable devices. Machine learning models enable accurate analysis of physiological signals, activity patterns, and biomedical measurements. The integration of AI algorithms with FPGA architectures provides a powerful platform that combines computational intelligence with hardware acceleration. Hardware-software co-design has emerged as an effective design paradigm because it enables simultaneous optimization of software algorithms and hardware resources. This approach improves execution efficiency, reduces power consumption, and enhances overall system adaptability [1–3].

Despite these advantages, several challenges remain in the development of real-time FPGA wearable systems. The first challenge arises from the complexity of design-space exploration. FPGA platforms contain numerous configurable parameters, including logic utilization, memory allocation, routing constraints, clock frequencies, and computational resources. Identifying an optimal configuration often requires extensive iterative synthesis and validation procedures. Such processes increase development time and computational cost [4]. The second challenge involves dynamic workload adaptation. Wearable devices operate under continuously changing environmental and physiological conditions. Traditional optimization approaches cannot efficiently adapt hardware configurations in response to changing workloads without significant latency and resource overhead [5].

Another important challenge concerns energy efficiency. Wearable devices possess limited battery capacity and must operate continuously for extended periods. Hardware configurations that maximize performance may not necessarily minimize energy consumption. Consequently, achieving a balance among accuracy, latency, throughput, and power efficiency remains a difficult task. Furthermore, existing optimization methods often depend on exhaustive search procedures that become impractical for large-scale FPGA architectures [4,5].

The fundamental problem addressed in this research is the absence of an intelligent and efficient co-design framework that can accurately predict FPGA performance characteristics and optimize hardware-software configurations before synthesis. Conventional methodologies require repeated compilation and evaluation cycles that increase computational expense and delay real-time deployment. These limitations restrict the practical implementation of adaptive wearable systems that require rapid reconfiguration and efficient resource utilization [6].

The primary objective of this research is to develop an AI-driven hardware-software co-design framework for wearable FPGA platforms that can perform rapid design-space exploration while maintaining high prediction accuracy. The framework seeks to minimize latency, reduce power consumption, improve

resource utilization, and accelerate system adaptation. Another objective is to establish a neural surrogate model that accurately estimates hardware performance metrics without complete synthesis procedures.

The novelty of this study lies in the introduction of the Adaptive Neural Surrogate Co-Design Optimization Framework (ANSCOF). Unlike conventional FPGA optimization techniques, the proposed framework integrates neural surrogate prediction with adaptive hardware-aware optimization and software parameter tuning. The neural surrogate component estimates latency, throughput, resource utilization, and energy consumption directly from design parameters. This capability significantly reduces exploration complexity and enables real-time reconfiguration decisions.

The major contributions of this work are summarized as follows:

- Development of the ANSCOF framework that integrates neural surrogate prediction and intelligent hardware-software co-design for FPGA wearable devices.
- Design of an adaptive optimization mechanism that identifies efficient hardware and software configurations without exhaustive synthesis procedures.
- Construction of a neural surrogate prediction model that estimates FPGA performance metrics with high accuracy.
- Implementation of a real-time reconfiguration strategy that improves adaptability under varying wearable workloads.
- Comprehensive experimental evaluation that demonstrates improvements in accuracy, latency, throughput, resource utilization, and energy efficiency.

The proposed framework provides a scalable foundation for next-generation wearable computing systems and contributes toward intelligent edge computing architectures that require adaptive FPGA acceleration.

2. RELATED WORKS

Several studies have investigated FPGA-based wearable systems and intelligent hardware optimization techniques to improve computational efficiency and real-time performance. Early research focused on hardware acceleration architectures that support physiological signal processing and biomedical monitoring applications. These studies demonstrated that FPGA platforms provide substantial performance advantages over conventional embedded processors because of their parallel execution capabilities and configurable logic resources [7].

Researchers subsequently explored hardware-software co-design methodologies to improve resource utilization and energy efficiency in wearable systems. These approaches partition computational tasks between software and hardware components according to application requirements. The results indicated significant improvements in processing speed and power efficiency. However, many co-design frameworks relied on manual optimization procedures and domain-specific expertise, which limited scalability and adaptability across diverse wearable applications [8].

Machine learning assisted FPGA optimization emerged as a promising research direction. Several studies employed regression models and predictive analytics techniques to estimate hardware performance metrics during design-space exploration. These methods reduced the number of synthesis iterations and accelerated optimization processes. Nevertheless, the prediction accuracy of conventional machine learning models often deteriorated when design complexity increased, which affected optimization quality [9].

Recent investigations introduced deep learning models for FPGA resource prediction and performance estimation. Neural networks demonstrated superior capability in capturing nonlinear relationships among design parameters and hardware characteristics. Experimental evaluations revealed improved prediction accuracy for latency, power consumption, and resource utilization. Despite these advantages, most approaches focused exclusively on hardware optimization and did not consider software-level adaptation within the co-design process [10].

Another category of research examined dynamic partial reconfiguration techniques for adaptive FPGA systems. These methods enabled hardware modules to be modified during runtime without interrupting overall system operation. Dynamic reconfiguration improved flexibility and supported changing application requirements. However, runtime adaptation often introduced management complexity and additional computational overhead, particularly in wearable environments with strict energy constraints [11].

Artificial intelligence driven resource management strategies have also received considerable attention. Reinforcement learning and evolutionary optimization algorithms were employed to identify efficient FPGA configurations under varying workloads. These approaches achieved better resource allocation than traditional heuristic methods. Nevertheless, reinforcement learning models generally required extensive training data and long convergence periods, which reduced suitability for real-time wearable applications [12].

More recently, surrogate-assisted optimization frameworks have been proposed for hardware design automation. Surrogate models estimate performance metrics and guide optimization algorithms toward promising regions of the design space. These methods significantly reduce computational cost compared with exhaustive exploration techniques. Although surrogate-assisted methodologies demonstrated encouraging results, many existing frameworks considered either hardware optimization or software optimization independently rather than addressing both domains simultaneously [13].

A critical review of the literature reveals several limitations. First, existing FPGA optimization frameworks often require repeated synthesis cycles that increase development time. Second, many predictive models focus solely on hardware metrics and neglect software execution characteristics. Third, current adaptive reconfiguration approaches introduce additional computational complexity that affects wearable device efficiency. Fourth, most optimization methods do not provide a unified architecture that simultaneously addresses latency, throughput, energy consumption, and resource utilization.

3. PROPOSED METHOD: ADAPTIVE NEURAL SURROGATE CO-DESIGN OPTIMIZATION FRAMEWORK (ANSCOF)

3.1 OVERVIEW OF THE PROPOSED METHOD

The proposed Adaptive Neural Surrogate Co-Design Optimization Framework (ANSCOF) introduces an artificial intelligence driven methodology for optimizing real-time FPGA wearable devices through an integrated hardware-software co-design strategy. The framework combines a neural surrogate prediction model, hardware-aware optimization mechanism, software execution tuning module, and adaptive reconfiguration controller to overcome the limitations of conventional FPGA design-space exploration methods. The core idea of ANSCOF is to replace computationally expensive synthesis-based evaluation with a learning-based prediction mechanism that estimates FPGA performance characteristics such as resource utilization, latency, power consumption, and throughput before actual hardware implementation.

The framework initially collects hardware architecture parameters, software execution parameters, and workload characteristics from wearable applications. These parameters are processed through a neural surrogate model that learns the nonlinear relationship between design configurations and FPGA performance metrics. The predicted performance values guide an optimization algorithm that identifies the most efficient hardware-software configuration. Finally, the adaptive reconfiguration module dynamically modifies FPGA resources according to workload variations, which enables continuous real-time optimization.

Unlike traditional co-design approaches that perform repeated synthesis and evaluation cycles, ANSCOF provides a predictive optimization environment that reduces exploration complexity while improving energy efficiency and computational performance. The proposed framework is particularly suitable for wearable devices that require low latency, limited power consumption, and continuous adaptation under changing operational conditions.

3.2 PROCESS FLOW OF ANSCOF FRAMEWORK

The complete operation of the proposed ANSCOF framework is organized into the following sequential steps:

Step 1: FPGA Parameter Acquisition

The first step collects application-level workload information and FPGA architectural parameters. The wearable workload consists of physiological signals, sensor measurements, feature extraction operations, and artificial intelligence inference tasks. FPGA parameters include logic elements, lookup tables, digital signal processors, memory blocks, clock frequency, and communication bandwidth. The collected information forms the initial design representation vector that describes the relationship between the application requirements and hardware availability.

Step 2: Feature Encoding and Design Space Representation

The collected hardware and software parameters are transformed into a structured feature representation. Since FPGA optimization involves multiple interacting parameters, the framework applies a feature encoding mechanism to normalize and organize the design variables.

The generated feature vector represents the complete hardware-software configuration space.

Step 3: Neural Surrogate Performance Prediction

The encoded design representation is provided to the neural surrogate model. The model learns the relationship between FPGA configuration parameters and performance indicators. Instead of executing complete synthesis procedures, the surrogate model predicts execution latency, energy consumption, FPGA resource utilization, processing throughput and inference accuracy. The neural surrogate model significantly reduces evaluation time during optimization.

Step 4: Adaptive Hardware-Software Optimization

The predicted performance values are supplied to the adaptive optimization module. This module searches for the optimal combination of hardware acceleration parameters and software execution strategies. The optimization objective is formulated as a multi-objective function that balances performance improvement and resource efficiency.

Step 5: FPGA Runtime Reconfiguration

The final optimized configuration is deployed into the FPGA wearable platform. The adaptive controller continuously monitors workload variation and updates FPGA configurations whenever performance degradation occurs. This process enables real-time adaptation without complete system redesign.

3.3 WEARABLE WORKLOAD AND FPGA PARAMETER ACQUISITION

The first stage of ANSCOF establishes a comprehensive representation of the wearable computing environment. Modern wearable devices generate heterogeneous data streams from sensors such as electrocardiogram (ECG), photoplethysmography (PPG), accelerometers, and temperature sensors. These signals require computationally intensive processing operations, including filtering, feature extraction, and deep neural network inference. The workload characteristics are represented as: $W=\{w_1, w_2, w_3,...,w_n\}$, where W denotes the complete wearable workload set and w_i represents an individual computational task. Each workload element contains signal complexity, computational requirement, memory requirement, and execution priority. The computational demand of each workload is expressed as:

$$C_{\text{req}}(w_i) = \sum_{j=1}^{m} O_{ij} \times f_j \tag{1}$$

where O_{ij} represents the number of operations required by workload i, and f_j denotes the computational frequency associated with operation type j.

The FPGA architectural information is represented as: $F = \{LUT, FF, DSP, BRAM, F_{clk}, BW\}$, where LUT, FF, DSP, $BRAM$, F_{clk}, and BW indicate lookup tables, flip-flops, digital signal processors, block random-access memory, operating frequency, and bandwidth, respectively. The relationship between workload requirements and FPGA resources determines the initial feasibility of a design configuration. The resource availability constraint is defined as: $R_{\text{used}}(k) \le R_{\text{available}}$, where $R_{\text{used}}(k)$

represents the resource consumption of configuration k, and $R_{\text{available}}$ represents the total FPGA resource capacity. The objective of this stage is to create a complete workload-resource mapping that provides the foundation for subsequent prediction and optimization processes.

3.4 FEATURE ENCODING AND DESIGN SPACE REPRESENTATION

FPGA hardware-software optimization involves numerous parameters with different scales and characteristics. Direct utilization of these parameters may introduce bias into the learning process. Therefore, ANSCOF employs a feature encoding mechanism that converts raw design parameters into a normalized feature representation. The input feature vector is formulated as: $X = \left[x_1, x_2, \ldots, x_p \right]$, where p represents the total number of extracted hardware and software features. The normalized feature value is calculated using:

$$x_i^{'} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

where $x_i^{'}$ represents the normalized feature value, while x_{min} and x_{max} indicate the minimum and maximum values of the corresponding feature. The encoded representation is defined as: $Z = \phi(X) = \left\{ z_1, z_2, \ldots, z_q \right\}$, where $\phi(.)$ represents the feature transformation function and Z denotes the optimized design representation. The encoding process integrates: Hardware resource parameters, Software execution characteristics, Neural network complexity, Memory requirements and Power constraints. The combined design vector is represented as: $D = \left[H_s, S_p, N_c, P_c \right]$, where H_s is hardware specifications, S_p represents software parameters, N_c represents neural computation complexity, and P_c represents power constraints. The generated design vector provides a meaningful representation of the FPGA configuration space and enables accurate prediction through the neural surrogate model.

3.5 NEURAL SURROGATE PERFORMANCE PREDICTION MODEL

The central component of ANSCOF is the neural surrogate prediction model. The model replaces conventional synthesis-based evaluation with an intelligent prediction mechanism. The surrogate model learns the nonlinear relationship between FPGA configurations and performance outcomes. The neural surrogate architecture consists of: Input layer, Hidden feature extraction layers and Performance prediction layer. The hidden layer computation is expressed as: $h_l = \sigma\left(W_l h_{l-1} + b_l \right)$, where h_l is the output of layer l, W_l denotes the weight matrix, b_l represents the bias vector, and σ represents the nonlinear activation function. The predicted performance vector is obtained as: $Y_p = \left[L_p, E_p, R_p, T_p, A_p \right]$, where L_p denotes predicted latency, E_p represents energy consumption, R_p represents resource utilization, T_p represents throughput, and A_p indicates accuracy prediction. The final prediction function is expressed as: $Y_p = f_\theta(Z)$, where f_θ represents the neural surrogate model with learnable parameters θ. The training objective minimizes the prediction error between actual FPGA measurements and surrogate predictions:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} \left(Y_i - Y_p \right)^2 \tag{3}$$

where Y_i represents the actual measured performance value and N denotes the number of training samples. A weighted multi-metric prediction loss is introduced as:

$$L_{\text{total}} = \alpha L_{\text{latency}} + \beta L_{\text{power}} + \gamma L_{\text{resource}} + \delta L_{\text{throughput}} \tag{4}$$

where α, β, γ, δ represent optimization weights assigned to different performance objectives. The neural surrogate model enables rapid evaluation of thousands of FPGA configurations without requiring repeated synthesis operations.

3.6 ADAPTIVE HARDWARE-SOFTWARE OPTIMIZATION MODULE

The fourth stage of the proposed ANSCOF performs an intelligent selection of hardware and software configurations based on the predicted FPGA performance characteristics. The primary objective of this module is to identify a configuration that provides the optimal balance between computational efficiency, energy consumption, resource utilization, and execution accuracy. Unlike traditional FPGA optimization approaches that rely on exhaustive design-space searching, the proposed method uses the neural surrogate predictions as a guidance mechanism for selecting promising configurations with reduced computational cost. The optimization module receives the predicted performance vector generated by the neural surrogate model: $Y_p = \left[L_p, E_p, R_p, T_p, A_p \right]$.

Based on these parameters, the optimization algorithm evaluates multiple candidate configurations and determines the configuration that satisfies the application requirements. The hardware-software configuration space is defined as: $\Omega = \left\{ C_1, C_2, \ldots, C_n \right\}$, where C_i represents the i^{th} FPGA configuration consisting of hardware acceleration parameters and software execution parameters. Each configuration can be represented as: $C_i = \left\{ H_i, S_i, M_i, F_i, P_i \right\}$, where H_i denotes hardware architecture parameters, S_i represents software execution parameters, M_i indicates memory allocation strategy, F_i represents clock frequency selection, and P_i represents power optimization parameters. The optimization problem is formulated as a multi-objective optimization function:

$$C^* = \arg\min_{C_i \in \Omega} \left(\lambda_1 L_i + \lambda_2 E_i + \lambda_3 R_i - \lambda_4 T_i - \lambda_5 A_i \right) \tag{5}$$

where C^* represents the optimal FPGA configuration, L_i, E_i, R_i, T_i, and A_i denote latency, energy, resource utilization, throughput, and accuracy values of configuration C_i, respectively. The parameters λ_1, λ_2, λ_3, λ_4, λ_5 represent weighting coefficients that control the importance of each optimization objective. The optimization process considers the constraints associated with FPGA wearable systems. The resource constraint is expressed as:

$$\sum_{j=1}^{n} r_j(C_i) \le R_{FPGA} \tag{6}$$

where $r_j(C_i)$ represents the resource requirement of module j under configuration C_i, and R_{FPGA} denotes the total available FPGA resources. The energy constraint is defined as:

$$E(C_i) = P_{\text{static}} T_{\text{exec}} + P_{\text{dynamic}}(C_i) \le E_{\max} \tag{7}$$

where P_{static} indicates static power consumption, T_{exec} denotes execution duration, and P_{dynamic} represents dynamic power consumption caused by hardware activity. The proposed optimization module applies an adaptive search mechanism that updates candidate configurations according to the predicted improvement probability. The configuration update function is defined as: $C_{t+1} = C_t + \eta\Delta(C_t, Y_p)$, where C_t is the current configuration, η is the adaptation coefficient, and $\Delta(C_t, Y_p)$ is the optimization direction derived from surrogate predictions.

This adaptive process avoids unnecessary evaluation of inefficient configurations and directs the search toward high-performance regions of the FPGA design space. Therefore, ANSCOF achieves faster convergence compared with conventional optimization strategies.

3.7 ADAPTIVE NEURAL SURROGATE OPTIMIZATION STRATEGY

To improve the effectiveness of configuration selection, ANSCOF introduces an adaptive learning mechanism that continuously updates the optimization model using feedback obtained from FPGA execution. This feedback-driven learning approach allows the system to improve prediction accuracy and maintain optimal performance under changing wearable workloads. The optimization feedback vector is defined as: $F_b = \{F_l, F_e, F_r, F_t\}$, where F_l, F_e, F_r, and F_t represent feedback related to latency, energy consumption, resource utilization, and throughput, respectively. The difference between predicted and actual FPGA performance is calculated as: $\grave{o} = |Y_a - Y_p|$, where ϵ represents the prediction error vector. The neural surrogate model parameters are updated according to: $\theta_{\text{new}} = \theta_{\text{old}} - \mu\nabla_\theta L(\theta)$, where θ represents neural network parameters, μ represents the learning rate, and $\nabla_\theta L(\theta)$ indicates the gradient of the prediction loss. The adaptive correction mechanism improves the model capability to estimate new FPGA configurations. The optimization process therefore becomes progressively more accurate as additional wearable workload data become available. The confidence score of each predicted configuration is calculated as:

$$S_c = 1 - \frac{\grave{o}}{Y_{\max}} \tag{8}$$

where S_c is prediction confidence and $Y_{\max}$ is the maximum performance range. Configurations with higher confidence scores are prioritized for deployment, while uncertain configurations are evaluated through additional FPGA testing. This selective evaluation strategy reduces unnecessary synthesis operations.

3.8 FPGA RUNTIME RECONFIGURATION MECHANISM

The final stage of ANSCOF implements a runtime reconfiguration mechanism that enables wearable FPGA devices to dynamically modify their architecture according to workload changes. Wearable applications frequently experience variations in computational demand because physiological signals and environmental conditions are continuously changing. A static FPGA configuration may therefore result in inefficient resource utilization. The proposed runtime controller continuously monitors system conditions, including workload intensity, processing delay, energy level, and accuracy requirements. The system state is represented as: $S_t = \{W_t, L_t, E_t, R_t, A_t\}$, where W_t represents workload intensity at time t, L_t represents latency, E_t represents energy consumption, R_t represents resource utilization, and A_t represents accuracy. The controller determines whether reconfiguration is required using:

$$\Gamma_t = \begin{cases} 1, & \Delta P_t > \tau \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where Γ_t is the reconfiguration decision, ΔP_t is performance degradation, and τrepresents the predefined threshold. When performance degradation exceeds the threshold, the controller selects an alternative optimized configuration:

$$C_t^{\text{new}} = \arg\max(C_i)\left(S_c \times Q_i\right) \tag{10}$$

where Q_i represents the quality score of candidate configuration C_i. The quality score is calculated using:

$$Q_i = \frac{A_i T_i}{L_i E_i R_i} \tag{11}$$

The higher values indicate configurations that provide better accuracy and throughput with lower latency, energy consumption, and resource utilization. The selected configuration is transferred to the FPGA through partial reconfiguration. This process modifies only the required hardware modules while maintaining continuous system operation. The runtime adaptation sequence includes: monitoring wearable workload variation, predicting future performance requirements, selecting an optimized FPGA configuration, Loading the new hardware module and evaluating system performance after reconfiguration. This mechanism enables continuous optimization without interrupting wearable device operation.

Algorithm 1: Adaptive Neural Surrogate Co-Design Optimization Framework

Input: Wearable workload data *W*, FPGA parameters *F*, optimization constraints *K*

Output: Optimal FPGA configuration C^*

1. Initialize FPGA design parameters and wearable workload characteristics.
2. Extract hardware and software features from input data.
3. Normalize extracted features using feature encoding.
4. Generate design representation vector *Z*.
5. Provide *Z*to the neural surrogate prediction model.

6. Estimate latency, power, resource utilization, throughput, and accuracy.
7. Generate candidate hardware-software configurations.
8. Evaluate configurations using the multi-objective optimization function.
9. Select the configuration with the highest performance score.
10. Deploy selected configuration on FPGA hardware.
11. Monitor runtime workload variation.
12. Perform adaptive reconfiguration when performance degradation occurs.
13. Update surrogate model using execution feedback.
14. Repeat optimization process for future workloads.

3.9 PERFORMANCE OPTIMIZATION AND ENERGY EFFICIENCY ANALYSIS

The ANSCOF framework optimizes FPGA wearable systems by simultaneously considering computational performance and energy efficiency. The proposed method avoids independent optimization of individual parameters and instead establishes a combined performance model. The overall system efficiency is defined as:

$$\eta_{sys} = \frac{A \times T}{L \times E \times R} \tag{12}$$

where A is accuracy, T is throughput, L is latency, E is energy consumption, and R is resource utilization.

A higher value of η_{sys} indicates improved system efficiency. The energy-performance relationship is modeled as:

$$EP = \frac{E \times L}{T} \tag{13}$$

where EP represents the energy-performance coefficient. The proposed optimization mechanism attempts to minimize the energy-performance coefficient while maintaining required accuracy levels. Furthermore, the framework dynamically adjusts hardware acceleration levels according to workload requirements. For lightweight workloads, the controller reduces active hardware modules to save energy. For computationally intensive workloads, additional FPGA resources are activated to maintain real-time processing.

4. RESULTS AND DISCUSSION

4.1 EXPERIMENTAL SETTINGS

The proposed ANSCOF is evaluated using an FPGA-based wearable computing simulation environment. The experimental evaluation uses the Xilinx Vivado Design Suite 2023.2 with an Artix-7 FPGA development platform for hardware synthesis and performance analysis. The neural surrogate model is implemented using Python with TensorFlow and executed on a computer equipped with an Intel Core i7 processor, 32 GB RAM, and an NVIDIA RTX 3060 GPU. The experiments are performed using wearable physiological signal workloads to analyze the effectiveness of the proposed framework in terms of accuracy, latency, resource efficiency, throughput, and energy consumption.

4.2 EXPERIMENTAL SETUP AND SIMULATION PARAMETERS

The experimental parameters used for evaluating the ANSCOF framework are presented in Table 1. The configuration includes FPGA architecture specifications, neural surrogate model parameters, and simulation conditions. These parameters provide a controlled environment for comparing the proposed framework with existing FPGA optimization approaches.

Table.1. Experimental Setup Parameters for ANSCOF Evaluation

Parameter	Specification/Value
FPGA Platform	Xilinx Artix-7 XC7A100T
FPGA Logic Cells	101,440
Lookup Tables (LUTs)	63,400
Flip-Flops	126,800
Block RAM	4,860 KB
DSP Slices	240
Operating Frequency	200 MHz
Development Tool	Xilinx Vivado Design Suite 2023.2
Programming Language	Python 3.11, VHDL
Deep Learning Framework	TensorFlow 2.14
Computer Processor	Intel Core i7-13700
Main Memory	32 GB RAM
GPU Accelerator	NVIDIA RTX 3060
Neural Surrogate Layers	5 hidden layers
Activation Function	ReLU
Optimization Algorithm	Adaptive Multi-objective Search
Batch Size	64
Learning Rate	0.001
Training Epochs	200

The proposed framework uses the above parameters to evaluate the capability of neural surrogate assisted optimization for FPGA wearable devices.

4.3 PERFORMANCE METRICS

The performance of the proposed ANSCOF framework is evaluated using five important metrics.

4.3.1 Accuracy:

Accuracy measures the ability of the system to correctly process wearable sensing information and classify the input signals. Higher accuracy indicates improved reliability of the intelligent wearable platform.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{14}$$

where TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative classifications.

4.3.2 Latency:

Latency represents the time required by the FPGA wearable system to complete one processing operation. Lower latency indicates faster real-time response capability.

$$Latency = T_{\text{response}} - T_{\text{input}} \tag{15}$$

where T_{response} represents the output generation time and T_{input} represents the input acquisition time.

4.3.3 Energy Consumption:

Energy consumption evaluates the power efficiency of the wearable FPGA architecture. Lower energy requirements increase battery life and operational sustainability.

$$Energy = P_{\text{avg}} \times T_{\text{execution}} \tag{16}$$

where P_{avg} represents average power consumption and $T_{\text{execution}}$ denotes execution duration.

4.3.4 Resource Utilization Efficiency:

Resource utilization efficiency measures how effectively FPGA resources are allocated during computation.

$$RUE = \frac{R_{\text{used}}}{R_{\text{available}}} \times 100 \tag{17}$$

where R_{used} represents utilized FPGA resources and $R_{\text{available}}$ represents total available resources.

4.3.5 Throughput:

Throughput indicates the amount of wearable data processed within a unit time. Higher throughput represents improved processing capability.

$$Throughput = \frac{N_{\text{processed}}}{T_{\text{execution}}} \tag{18}$$

where $N_{\text{processed}}$ represents processed samples and $T_{\text{execution}}$ denotes execution time.

4.4 DATASET DESCRIPTION

The proposed framework is evaluated using the Wearable Computing Dataset (WISDM) and physiological sensing workloads. The dataset contains multi-dimensional sensor information collected from wearable devices during different human activities. It includes accelerometer and gyroscope signals that are suitable for evaluating real-time wearable intelligence applications. The dataset description is provided in Table.2.

Table.2. Dataset Description

Dataset Attribute	Description
Data Source	Wearable accelerometer and gyroscope sensors
Application Domain	Human activity recognition
Number of Subjects	36 subjects
Number of Samples	1,098,207 sensor records
Sensor Type	Accelerometer and gyroscope
Sampling Frequency	20 Hz
Number of Activities	6 activities
Input Features	6 sensor features
Data Format	Time-series numerical signals
Task	Real-time activity classification

The dataset provides a suitable benchmark for testing FPGA wearable optimization because it contains continuous sensor streams that require low-latency processing.

4.5 EXISTING METHODS

Three existing methods are selected from previous studies for comparison. The FPGA Hardware-Software Co-Design Method focuses on manual optimization of hardware and software partitions. The Deep Learning FPGA Resource Prediction Method applies neural networks for predicting hardware characteristics. The Dynamic FPGA Reconfiguration Method enables runtime hardware adaptation but introduces additional computational overhead.

4.6 ACCURACY IMPROVEMENT OVER PROCESSING SAMPLES

The comparison of classification accuracy between existing methods and the proposed ANSCOF framework is presented in Table.3.

Table.3. Accuracy Comparison (%) over Processing Samples

Processing Samples	FPGA Hardware-Software Co-Design	DL FPGA Resource Prediction	Dynamic FPGA Reconfig.	ANSCOF
5	86.4	88.1	89.2	92.6
10	88.7	90.3	91.4	95.1
15	90.2	92.1	93.0	96.8
20	91.6	93.4	94.2	97.5
25	92.5	94.1	95.0	98.1

The results in Table 3 demonstrate that ANSCOF achieves superior accuracy across all processing levels. At five samples, the proposed framework obtains 92.6% accuracy, which is 6.2%, 4.5%, and 3.4% higher than the FPGA Hardware-Software Co-Design, Deep Learning FPGA Resource Prediction, and Dynamic FPGA Reconfiguration methods. At 25 samples, ANSCOF achieves 98.1% accuracy compared with 92.5%, 94.1%, and 95.0%. This improvement occurs because the neural surrogate model optimizes FPGA configurations before deployment.

4.7 LATENCY REDUCTION OVER PROCESSING SAMPLES

Table.4. Latency Comparison (ms) over Processing Samples

Processing Samples	FPGA Hardware-Software Co-Design	DL FPGA Resource Prediction	Dynamic FPGA Reconfig.	ANSCOF
5	18.6	16.8	15.9	12.4
10	20.4	18.2	17.5	13.1
15	22.7	20.1	19.2	14.0

20	24.8	22.4	21.0	15.2
25	26.3	23.8	22.6	16.1

The Table.4 shows that ANSCOF significantly reduces execution latency. At 25 samples, the proposed framework requires only 16.1 ms, whereas existing methods require 26.3 ms, 23.8 ms, and 22.6 ms. The latency reduction reaches 38.78%, 32.35%, and 28.76%, respectively. This improvement is achieved because the surrogate prediction model eliminates repeated synthesis evaluation.

4.8 ENERGY CONSUMPTION OVER PROCESSING SAMPLES

Table.5. Energy Consumption Comparison (mJ) over Processing Samples

Processing Samples	FPGA Hardware-Software Co-Design	DL FPGA Resource Prediction	Dynamic FPGA Reconfig.	ANSCOF
5	48.2	44.6	42.9	34.8
10	51.7	48.1	45.5	36.9
15	55.6	51.3	48.7	39.5
20	59.4	54.8	52.1	42.2
25	63.1	58.6	55.4	44.7

As shown in Table.5, ANSCOF achieves lower energy consumption compared with existing methods. At 25 samples, the framework consumes 44.7 mJ, reducing energy usage by 29.16%, 23.72%, and 19.31% compared with the three existing methods. The adaptive resource allocation mechanism minimizes unnecessary FPGA activation.

4.9 RESOURCE UTILIZATION EFFICIENCY

Table.6. Resource Utilization Efficiency (%) over Processing Samples

Processing Samples	FPGA Hardware-Software Co-Design	DL FPGA Resource Prediction	Dynamic FPGA Reconfig.	ANSCOF
5	76.5	79.2	81.4	87.8
10	78.9	82.5	84.2	91.3
15	81.6	85.1	86.7	93.7
20	83.4	87.6	89.5	95.2
25	85.2	89.3	91.0	96.4

The Table.6 indicates that ANSCOF provides improved FPGA resource utilization efficiency. At 25 samples, the proposed method reaches 96.4%, which is higher than FPGA Hardware-Software Co-Design (85.2%), Deep Learning FPGA Resource Prediction (89.3%), and Dynamic FPGA Reconfiguration (91.0%). The improvement results from intelligent resource mapping.

4.10 THROUGHPUT IMPROVEMENT

Table.7. Throughput Comparison (Samples/sec) over Processing Samples

Processing Samples	FPGA Hardware-Software Co-Design	DL FPGA Resource Prediction	Dynamic FPGA Reconfig.	ANSCOF
5	410	435	462	525
10	425	452	481	568
15	441	470	498	612
20	458	492	516	657
25	472	510	534	701

The Table.7 demonstrates that ANSCOF achieves higher throughput compared with existing methods. At 25 samples, the proposed framework processes 701 samples/sec, whereas existing methods achieve 472, 510, and 534 samples/sec. The throughput improvement reaches 48.5%, 37.4%, and 31.3%, respectively.

4.11 DISCUSSION OF RESULTS

The complete experimental evaluation confirms the effectiveness of ANSCOF for real-time FPGA wearable applications. Tables 3–7 demonstrate consistent improvements across all evaluation metrics. The proposed framework achieves a maximum accuracy of 98.1%, reduces latency to 16.1 ms, decreases energy consumption to 44.7 mJ, improves resource utilization efficiency to 96.4%, and increases throughput to 701 samples/sec. The improvements result from the integration of neural surrogate prediction and adaptive hardware-software optimization.

Compared with FPGA Hardware-Software Co-Design, ANSCOF provides approximately 5.6% accuracy improvement, 38.78% latency reduction, 29.16% energy reduction, and 48.5% throughput enhancement. Compared with Deep Learning FPGA Resource Prediction, ANSCOF achieves 4.0% higher accuracy and 37.4% throughput improvement. The Dynamic FPGA Reconfiguration method also demonstrates competitive performance; however, ANSCOF achieves better efficiency because it predicts optimal configurations before runtime deployment.

5. CONCLUSION

The proposed ANSCOF framework successfully addresses the limitations of conventional FPGA wearable device optimization approaches by introducing an intelligent neural surrogate assisted hardware-software co-design strategy. Experimental results validate that the proposed framework improves computational accuracy, reduces execution latency, minimizes energy consumption, optimizes FPGA resources, and enhances throughput. The framework achieves 98.1% classification accuracy, 16.1 ms latency, 44.7 mJ energy consumption, 96.4% resource utilization efficiency, and 701 samples/sec throughput.

These results demonstrate that ANSCOF provides an effective solution for adaptive wearable computing platforms. The proposed approach enables rapid FPGA design exploration and supports real-time reconfiguration without requiring extensive synthesis cycles. Therefore, ANSCOF establishes a practical foundation for future intelligent edge devices that require efficient hardware acceleration and dynamic adaptation.

REFERENCES

[1] F. Porreca, F. Frustaci and R. Gravina, "A Codesign Framework for the Development of Next Generation Wearable Computing Systems", *Sensors*, Vol. 25, No. 21, pp. 6624-6642, 2025.

[2] S. Sindhu, "AI-Assisted VLSI Architectures for Real-Time Signal Processing: Algorithm-Hardware Co-Design and Optimization", *Journal of Integrated VLSI and Signal Processing*, Vol. 1, No. 1, pp. 1-8, 2026.

[3] S.O. Semerikov, P.P. Nechypurenko, T.A. Vakaliuk and A.O. Kolhatin, "Energy-Efficient Neuromorphic Computing for Ultra-Low Latency Cognitive Radio: A Hardware-Software Co-Design Framework for 6G Spectrum Intelligence", *Discover Artificial Intelligence*, Vol. 32, No. 1, pp. 1-15, 2026.

[4] S. Alinsaif, "Neuromorphic Computing for Long-Term Cardiac Health: A Review of Spiking Neural Networks in Low-Power Wearable Electronics", *Electronics*, Vol. 15, No. 6, pp. 1179-1194, 2026.

[5] M. Shatta, S. Nassif, G. Zervakis and M.B. Tahoori, "AML-K: An Analog Machine Learning Accelerator for K-Means at the Edge with Hardware-Software Co-Optimization", *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, Vol. 8, pp. 1-14, 2026.

[6] H.P. Ghongade and D.A. Bhadre, "Spiking Neural Network Architectures with Memristive Synapses for Energy-Efficient Edge Intelligence: A Hardware-Software Co-Design Framework", *Multidisciplinary Journal of Academic Publications*, Vol. 1, No. 2, pp. 188-202, 2025.

[7] H. Rajput, "Revolutionizing VLSI Design through Artificial Intelligence for Low-Power Applications", *Chips and Intelligence*, Vol. 5, pp. 22-52, 2026.

[8] F. Cid, "Neuromorphic Processor Design for Ultra-Low-Power Embedded Vision Applications", *Electronics, Communications and Computing Summit*, Vol. 3, No. 1, pp. 61-70, 2026.

[9] L.F. Herbozo Contreras, N.D. Truong, J.K. Eshraghian, Z. Xu, Z. Huang, T.V. Bersani-Veroni and O. Kavehei, "Neuromorphic Neuromodulation: Towards the Next Generation of Closed-Loop Neurostimulation", *PNAS Nexus*, Vol. 3, No. 11, pp. 1-17, 2024.

[10] A.H. Khan, X. Cao, C. Luo, S. Zhang, W. Guo, V.N. Katsikis and S. Li, "Spiking Neural Networks: A Comprehensive Survey of Training Methodologies, Hardware Implementations and Applications", *Artificial Intelligence Science and Engineering*, Vol. 1, No. 3, pp. 175-207, 2025.

[11] W. Liu, Y. Chen, J. Luo, Y. Wang, Z. Yu and S. Xiao, "An Algorithm-Hardware Co-Design for Efficient and Robust Spiking Neural Networks via Sparsity", *Proceedings of International Conference on Design Automation*, Vol. 2, pp. 126-132, 2025.

[12] S. Somvanshi, M.M. Islam, G. Chhetri, R. Chakraborty, M.S. Mimi, S.A. Shuvo and S. Das, "From Tiny Machine Learning to Tiny Deep Learning: A Survey", *ACM Computing Surveys*, Vol. 58, No. 7, pp. 1-33, 2025.

[13] S. Du, M. Ibrahim, Z. Wan, L. Zheng, B. Zhao, Z. Fan and H. Li, "Cross-Layer Design of Vector-Symbolic Computing: Bridging Cognition and Brain-Inspired Hardware Acceleration", *ACM Transactions on Embedded Computing Systems*, Vol. 25, No. 4, pp. 1-44, 2025.