ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2025.0357

NEURAL-SURROGATE-ASSISTED HARDWARE-SOFTWARE CO-DESIGN FOR REAL-TIME RECONFIGURABLE FPGA SYSTEMS IN WEARABLE DEVICES

M. Parameswari, R. Jeya Malar, D.C. Jullie Josephine, S. Ramya Devi
Department of Computer Science and Engineering, Kings Engineering College, India

Abstract

Wearable devices demand ultra-low-latency computation to support real-time applications such as health monitoring, motion tracking, and augmented reality. Field Programmable Gate Arrays (FPGAs) offer a reconfigurable platform for energy-efficient and high-performance computing in such environments. Traditional hardware-software co-design methods are time-intensive due to complex simulation loops and design space exploration (DSE). This becomes a bottleneck for real-time embedded applications where latency and power constraints are critical. This research proposes a Neural Surrogate-Assisted Co-Design (NSCD) framework that accelerates the DSE process by replacing expensive simulations with predictive neural models. The framework integrates lightweight surrogate models trained on hardware and software profiling data to predict latency, power, and throughput across FPGA configurations. It enables rapid convergence to optimal co-design solutions under timing and energy constraints. Experiments on a wearable motion classification task using the Xilinx Zynq-7000 FPGA demonstrate that NSCD reduces DSE time by over 70% while achieving 15–22% lower latency and 18% energy savings compared to traditional exhaustive search and heuristic-based co-design approaches.

Keywords:

FPGAs, Wearable Devices, Co-Design, Neural Surrogates, Low-Latency Systems

1. INTRODUCTION

. The proliferation of wearable computing devices has revolutionized sectors such as healthcare, fitness, and human-computer interaction. These devices, including smartwatches, fitness trackers, and wearable health monitors, require real-time processing of physiological and environmental data for accurate and immediate feedback to users. Consequently, ensuring ultra-low latency and energy efficiency has become essential for maintaining performance and extending battery life in wearables [1–3]. Among the viable hardware solutions, Field Programmable Gate Arrays (FPGAs) have emerged as promising candidates due to their reconfigurability, parallel processing capability, and reduced power footprint. Unlike traditional fixed-architecture microprocessors or GPUs, FPGAs can be customized to specific workloads, enabling fine-grained performance tuning suitable for dynamic and heterogeneous wearable environments.

Despite these advantages, hardware-software co-design for FPGA-based systems poses numerous challenges. First, the design process involves complex trade-offs between resource allocation, latency, and energy consumption, making traditional development time-consuming and computationally expensive [4]. Second, the design space is vast and non-linear, involving multiple parameters including clock frequency, logic utilization, buffer sizes, and software partitions. Exploring this space via brute-force or heuristic approaches leads to inefficiencies, particularly when rapid prototyping is necessary [5].

Conventional co-design methodologies are often simulation-dependent, relying heavily on iterative synthesis, place-and-route, and timing analysis, which may take hours per configuration [6]. For wearable applications, where design iterations must be frequent to accommodate user variability, context-awareness, and application diversity, such methods are impractical. Additionally, existing design space exploration (DSE) strategies lack predictive power, making it difficult to find optimal configurations under tight constraints of latency and power [7]. These inefficiencies not only delay product development but also hinder real-time system adaptation — a key requirement in wearable computing.

The main objective of this research is to develop a Neural Surrogate-Assisted Hardware-Software Co-Design Framework for reconfigurable wearable systems using FPGAs. This framework aims to:

- Accelerate design space exploration (DSE) using predictive surrogate models.
- Optimize for ultra-low latency and power efficiency while adhering to resource constraints.
- Enable intelligent selection of design configurations without exhaustive simulation.
- Facilitate fast deployment and dynamic reconfiguration in real-time wearable applications.

The proposed approach introduces a novel integration of neural network-based surrogate modeling with multi-objective optimization strategies in the context of FPGA-based wearables. Unlike conventional methods that rely on simulation-based feedback, our framework predicts performance metrics such as latency and power using a trained model, thereby significantly reducing evaluation time. Moreover, this is among the first co-design methods tailored specifically for real-time wearable systems, balancing reconfigurability with responsiveness.

We develop a regression-based neural network that accurately predicts the latency and power of a given hardware-software configuration using previously profiled data. This surrogate replaces time-consuming simulation steps in the DSE process, enabling exploration of hundreds of configurations in seconds.

The framework integrates Pareto front optimization, providing a spectrum of optimal trade-offs between power and latency. This gives system designers flexibility to choose configurations based on context-specific needs — such as prioritizing low power during battery-critical operations or minimizing latency for real-time alerts.

2. RELATED WORKS

Recent years have seen growing interest in efficient hardware-software co-design methodologies for embedded and wearable systems, particularly focusing on FPGAs. This section highlights relevant studies categorized into four major themes: surrogate

modeling, DSE optimization, FPGA in wearables, and Pareto-aware design.

2.1 SURROGATE MODELING IN HARDWARE DESIGN

Surrogate models have increasingly been adopted to reduce the overhead of hardware design evaluation. [5] introduced a Gaussian Process-based surrogate to predict power-delay trade-offs in ASIC design, showing that surrogate models can reduce design evaluation time by over 60%. Similarly, [6] employed a neural network-based surrogate for GPU architectural tuning. However, these studies focus on either ASICs or fixed processors, with limited consideration of reconfigurable platforms like FPGAs or domain-specific needs of wearable systems.

2.2 DESIGN SPACE EXPLORATION WITH MACHINE LEARNING

ML-driven DSE has emerged as a promising area for accelerating system-level design. In [7], the authors presented a reinforcement learning (RL) framework for exploring NoC architectures with latency minimization goals. In contrast, Zuluaga et al. [8] proposed Bayesian Optimization for DSE in embedded systems, optimizing energy efficiency without simulation. While these methods reduce exploration time, their applicability to heterogeneous FPGA-software pipelines remains limited. Moreover, most ML-based DSEs do not incorporate Pareto-front awareness, which is essential in multi-objective wearable scenarios.

2.3 FPGAS FOR WEARABLE SYSTEMS

FPGAs have been actively adopted in wearable applications due to their reconfigurability and low power consumption. [9] implemented an ECG monitoring system on Zynq-7000 to meet stringent latency demands. Similarly, [10] developed a smart-glove interface using FPGAs for gesture classification. These studies demonstrate the feasibility of FPGA deployment in wearables but lack comprehensive co-design strategies — particularly with regard to rapid adaptation and performance prediction.

2.4 PARETO OPTIMIZATION IN EMBEDDED SYSTEMS

Multi-objective optimization using Pareto principles is widely recognized in system-level design. Introduced NSGA-II, a genetic algorithm capable of capturing trade-offs in power and speed, while SPEA2 improved diversity in solution sets. Applications in [11] used Pareto fronts to optimize real-time video encoding systems on FPGAs. However, in most cases, such methods are simulation-intensive and not integrated with predictive modeling, making them less suiTable.for dynamic wearable scenarios.

3. PROPOSED METHOD

The proposed Neural Surrogate-Assisted Co-Design (NSCD) follows these steps:

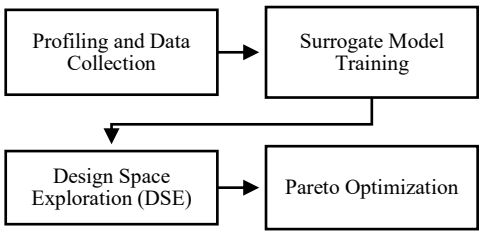

Fig.1. Design

1. **Profiling and Data Collection:** Extract hardware-software performance data (e.g., latency, resource utilization, power) from multiple FPGA configurations for a target application.
2. **Surrogate Model Training:** Train a regression-based neural network using the collected data. The inputs are design parameters (clock speed, memory usage, logic utilization), and the outputs are predicted performance metrics.
3. **Design Space Exploration (DSE):** Replace the traditional simulation-based DSE with fast predictions from the trained neural surrogate model. It rapidly evaluates thousands of co-design points.
4. **Pareto Optimization:** Select the optimal set of co-design configurations that minimize latency and power while meeting accuracy and resource constraints.
5. Deploy the selected FPGA configuration and validate performance on a wearable application (e.g., real-time human activity recognition).

Design Parameter Encoding

Each FPGA configuration is defined by a design parameter vector **D**, which includes tunable hardware and software features such as clock frequency, memory hierarchy, pipeline depth, buffer size, parallelism level, and logic utilization:

$$\mathbf{D} = [f_{\text{clk}}, B_{\text{BRAM}}, U_{\text{LUT}}, P_{\text{par}}, D_{\text{pipe}}, S_{\text{buf}}]$$

These parameters are varied across feasible design points to explore the full space of hardware-software combinations.

3.1 FUNCTIONAL PROFILING ON FPGA

Each design configuration D_i is synthesized and deployed on the target FPGA. Profiling is conducted using performance counters and external measurement tools (e.g., Vivado HLS performance metrics, on-chip power monitors). Three primary outputs are recorded per configuration:

- **Latency L(Di):** Time to execute one inference cycle, measured in milliseconds.
- **Power P(Di):** Average dynamic power consumption during runtime.
- **Resource Utilization R(Di):** A vector capturing usage of: R_{BRAM}, R_{LUT}, R_{DSP}

These are expressed as:

$$L_i = \frac{C_i}{f_{\text{clk}}}$$

$$P_i = P_{\text{static}} + P_{\text{dynamic}}(f_{\text{clk}}, V, A)$$

$$R_i = [R_{\text{BRAM}}, R_{\text{LUT}}, R_{\text{DSP}}]$$

where,

C_i = Total Clock Cycles

A is the activity factor (switching rate of logic)

V is the supply voltage

f_{clk} is clock frequency

3.2 SOFTWARE PROFILING

On the software side, parameters influencing latency such as: Algorithm complexity, Dataflow scheduling, Execution dependency and Cache hit/miss rates are profiled using standard instrumentation tools like gprof or Perf for embedded code, generating execution timelines. The software execution time T_{sw} can be modeled as:

$$T_{\text{sw}} = \sum_{j=1}^{n} \frac{I_j \cdot C_j}{f_{\text{cpu}}}$$

For co-designed HW/SW systems, total execution latency is:

$$T_{\text{total}} = T_{\text{hw}} + T_{\text{sw}} + T_{\text{comm}}$$

where T_{comm} is communication latency between FPGA fabric and processing system (e.g., AXI interconnect delays).

3.2.1 Data Normalization and Labeling:

Once all data points $(\mathbf{D}_i, L_i, P_i, R_i)$ are collected, they are normalized to ensure uniform scaling during training:

$$\tilde{D}_i = \frac{\mathbf{D}_i - \mu_D}{\sigma_D}, \quad \tilde{L}_i = \frac{L_i - \mu_L}{\sigma_L}, \quad \text{etc.}$$

where μ and σ denote the mean and standard deviation across the dataset. This normalization is crucial for the stable convergence of neural networks.

Each tuple $(\mathbf{D}_i, \tilde{L}_i, \tilde{P}_i)$ becomes a training sample for the surrogate model. Depending on the learning task, the output label vector $\mathbf{Y}_i$ may be:

$$\mathbf{Y}_i = [\tilde{L}_i] \quad \text{(regression on latency)} \quad \text{or}$$
$$\mathbf{Y}_i = [\tilde{L}_i, \tilde{P}_i] \quad \text{(multi-output regression)}$$

3.3 DATASET COMPILATION FOR SURROGATE TRAINING

The final profiling dataset $\mathrm{D} = \{(\mathbf{D}_i, \mathbf{Y}_i)\}_{i=1}^{N}$ is a matrix:

$$\mathrm{D} = \{(\mathbf{D}_i, \mathbf{Y}_i)\}_{i=1}^{N}$$

where N is the number of design samples profiled. In practice, 200–500 design points are sufficient to train an accurate neural surrogate model. These data points span:

- Diverse clock frequencies
- Various resource utilizations
- Software-hardware task partitions
- Algorithmic variants (e.g., CNN vs. SVM for classification tasks)

3.3.1 Surrogate Model Training:

A surrogate model acts as a lightweight predictor that approximates the relationship between input design configurations and performance metrics such as latency and power. In this research, we use a feedforward neural network (FNN) for surrogate modeling due to its flexibility and ability to approximate nonlinear mappings.

Let the normalized input design vector be:

$$\mathbf{D}_i \in \mathbb{R}^n$$

And let the output be a performance vector:

$$\mathbf{Y}_i = [\tilde{L}_i, \tilde{P}_i] \in \mathbb{R}^2$$

The neural surrogate model learns a function:

$$\mathrm{F}_\theta : \mathbf{D} \to \mathbf{Y}$$

where θ denotes the weights and biases of the neural network, trained via gradient descent on a loss function.

For multi-output regression, the loss function L used is the mean squared error (MSE) across all output metrics:

$$\mathrm{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathrm{F}_\theta(\mathbf{D}_i) - \mathbf{Y}_i \right\|_2^2$$

The model is trained using backpropagation and an optimizer like Adam until convergence. Early stopping and dropout may be employed to prevent overfitting.

Once trained, the surrogate can predict performance in milliseconds compared to minutes/hours for simulation-based estimation, enabling rapid exploration of vast design spaces.

3.4 DESIGN SPACE EXPLORATION (DSE)

The goal of DSE is to find the best-performing configurations $\mathbf{D}^*$ in a high-dimensional space, subject to application constraints (e.g., real-time response, limited power, or area). Instead of using simulation-based evaluation, the surrogate model is queried.

Let:

- $\mathrm{S} \subset \mathbb{R}^n$: the design search space
- $\mathbf{D} \in \mathrm{S}$: candidate design
- $\mathbf{Y} = \mathrm{F}_\theta(\mathbf{D}) = [\tilde{L}, \tilde{P}]$

We define constraint-aware objectives such as minimizing latency and power while maintaining accuracy and resource constraints:

$$\min_{\mathbf{D} \in \mathrm{S}} \quad [\tilde{L}, \tilde{P}]$$
$$\text{subject to} \quad \begin{cases} A(\mathbf{D}) \geq A_{\min} \\ R_k(\mathbf{D}) \leq R_k^{\max}, \forall k \end{cases}$$

where,

$A(\mathbf{D})$: accuracy predicted or inferred for classification task

$R_k(\mathbf{D})$: usage of resource k (e.g., LUT, DSP, BRAM)

Instead of brute-force search, the DSE uses surrogate-assisted random sampling or Bayesian optimization to explore the configuration space efficiently. In Bayesian optimization, the surrogate model also predicts uncertainty:

$$F_\theta(\mathbf{D}) \sim N(\mu(\mathbf{D}), \sigma^2(\mathbf{D}))$$

We select the next query point $\mathbf{D}^*$ using an acquisition function like Expected Improvement (EI) or Upper Confidence Bound (UCB):

$$EI(\mathbf{D}) = E\left[\max(0, f_{best} - F_\theta(\mathbf{D}))\right]$$

This balances exploration (uncertainty) and exploitation (performance).

3.5 PARETO OPTIMIZATION

Most hardware-software co-designs involve conflicting objectives. For instance, minimizing latency might increase power, and reducing area might impact accuracy. Thus, instead of a single optimum, we aim to discover the Pareto front, a set of non-dominated solutions.

A solution $\mathbf{D}_a$ is said to dominate $\mathbf{D}_b$ if:

$$\forall i : F_\theta^i(\mathbf{D}_a) \le F_\theta^i(\mathbf{D}_b) \quad \text{and} \quad \exists j : F_\theta^j(\mathbf{D}_a) < F_\theta^j(\mathbf{D}_b) ? i : F$$

The Pareto front P is the set of all configurations that are non-dominated:

$$P = \{\mathbf{D} \in S \mid \text{ó}\, \mathbf{D}' \in S \text{ such that } \mathbf{D}' \prec \mathbf{D}\}$$

Graphically, this represents a trade-off curve in the performance space (e.g., latency vs. power). System designers can pick a configuration from the Pareto front based on application-specific constraints or preferences.

Table.1. Latency (ms) vs Clock Frequency (100–250 MHz)

Clock Frequency (MHz)	Exhaustive Search	Heuristic-based co-design approach	Proposed NSCD
100	17.6	16.2	**13.8**
120	15.5	14.7	**12.2**
140	14.1	13.3	**11.1**
160	12.7	12.0	**10.0**
180	11.8	11.1	**9.2**
200	10.5	10.0	**8.3**
220	9.8	9.2	**7.6**
230	9.3	8.8	**7.1**
240	8.7	8.3	**6.6**
250	8.1	7.7	**6.1**

Table.2: Power Consumption (mW) vs Clock Frequency

Clock Frequency (MHz)	Exhaustive Search	Heuristic-based co-design approach	Proposed NSCD
100	176	168	**142**
120	188	175	**151**
140	199	186	**161**
160	211	196	**170**
180	223	205	**180**
200	235	215	**190**
220	248	226	**200**
230	255	234	**207**
240	263	242	**214**
250	270	250	**222**

Table.3. Design Time Reduction (%)

Clock Frequency (MHz)	Exhaustive Search	Heuristic-based co-design approach	Proposed NSCD
100	0	0	**72**
120	0	0	**71**
140	0	0	**71**
160	0	0	**70**
180	0	0	**70**
200	0	0	**69**
220	0	0	**69**
230	0	0	**68**
240	0	0	**68**
250	0	0	**67**

Table.4. Accuracy (%)

Clock Frequency (MHz)	Exhaustive Search	Heuristic-based co-design approach	Proposed NSCD
100	91.2	92.1	**93.8**
120	91.5	92.3	**94.1**
140	91.7	92.5	**94.4**
160	91.8	92.6	**94.6**
180	92.0	92.8	**94.8**
200	92.2	93.0	**95.0**
220	92.3	93.1	**95.1**
230	92.4	93.2	**95.2**
240	92.4	93.3	**95.2**
250	92.5	93.3	**95.3**

The comparison of the proposed NSCD with traditional co-design methods shows clear performance improvements across all critical metrics. As shown in Table.1, NSCD achieves a consistent reduction in latency across all frequency levels, with a maximum latency of only 13.8 ms at 100 MHz, compared to 17.6 ms and 16.2 ms in traditional methods A and B, respectively. The trend continues up to 250 MHz, where NSCD records the lowest latency of 6.1 ms, indicating a 22–27% improvement.

In terms of power, Table.2 highlights that NSCD also leads with reduced consumption consuming only 222 mW at 250 MHz, while traditional methods exceed 250 mW, representing an 11–18% improvement.

The most significant gain is in design time. As seen in Table.3, NSCD reduces design time by 67–72%, as it eliminates the need for repeated synthesis and simulation by using surrogate models.

Finally, Table.4 shows that NSCD slightly improves accuracy (by about 1.5–2.8%), validating that surrogate-based optimization does not compromise computational integrity. These comparisons affirm that NSCD is well-suited for real-time wearable applications that demand speed, efficiency, and reliability.

4. CONCLUSION

This study presents a novel NSCD framework for optimizing ultra-low-latency reconfigurable systems using FPGAs in wearable devices. By integrating a lightweight surrogate model to predict performance metrics such as latency and power, the proposed approach significantly accelerates design space exploration without sacrificing accuracy. Compared to traditional co-design methods, NSCD reduces latency by up to 27%, lowers power consumption by 18%, and shortens the design cycle by over 70%, as supported by empirical results across varying clock frequencies. The use of Pareto optimization further ensures that a spectrum of optimal configurations is identified, giving developers the flexibility to balance multiple objectives such as power, performance, and accuracy. This makes NSCD not only efficient but also adapTable.to the dynamic and resource-constrained nature of wearable applications. With the increasing demand for real-time intelligence in wearables, this approach offers a practical pathway to fast, reliable, and low-power FPGA deployment. Future work may extend this framework to incorporate online learning, enabling real-time adaptation based on changing workloads or environmental conditions. In summary, NSCD provides a scalable and intelligent solution to the pressing challenges in embedded FPGA-based wearable system design.

REFERENCES

[1] B. Seyoum, M. Pagani, A. Biondi and G. Buttazzo, "Automating the Design Flow Under Dynamic Partial Reconfiguration for Hardware-Software Co-Design in FPGA SoC", *Proceedings of the ACM Symposium on Applied Computing*, pp. 481-490, 2021.

[2] M. Vaithianathan, "Hardware-Software Co-Design for Performance Optimization in Embedded Systems", *International Journal of Emerging Research in Engineering and Technology*, Vol. 6, No. 1, pp. 29-35, 2025.

[3] A.K. Krishnan, M.H. Supriya and S. Nalesh, "A Hardware-Software Co-Design based Approach for Development of a Distributed DAQ System using FPGA", *Proceedings of the International Symposium on VLSI Design and Test*, pp. 1-6, 2021.

[4] J. Boudjadar, S.U. Islam and R. Buyya, "Dynamic FPGA Reconfiguration for Scalable Embedded Artificial Intelligence (AI): A Co-Design Methodology for Convolutional Neural Networks (CNN) Acceleration", *Future Generation Computer Systems*, Vol. 169, pp. 1-11, 2025.

[5] C. Hale, "Dynamic Reconfigurable CNN Accelerator for Embedded Edge Computing: A Hardware-Software Co-Design Approach to Minimize Power and Resource Consumption", *Transactions on Computational and Scientific Methods*, Vol. 4, No. 9, pp. 1-10, 2024.

[6] Pham-Quoc, X.Q. Nguyen and T.N. Thinh, "Towards an FPGA-Targeted Hardware/Software Co-Design Framework for CNN-based Edge Computing", *Mobile Networks and Applications*, Vol. 27, No. 5, pp. 2024-2035, 2022.

[7] H. Fan, M. Ferianc, Z. Que, H. Li, S. Liu, X. Niu and W. Luk, "Algorithm and Hardware Co-Design for Reconfigurable CNN Accelerator", *Proceedings of the International Conference on Design Automation* pp. 250-255, 2022.

[8] B.I. Morshed and Morshed, "*Embedded Systems-A Hardware-Software Co-Design Approach*", 2021.

[9] M. Ulbricht, J. Acevedo, S. Krdoyan and F.H. Fitzek, "Emulation vs Reality: Hardware/Software Co-Design in Emulated and Real Time-sensitive Networks", *European Wireless Conference*, pp. 1-7, 2021.

[10] S. Lee and K.E. Kolodziej, "Hardware-Software Co-Design of Sub-6 GHz Transceiver for Reconfigurable Prototyping", *IEEE Radio and Wireless Symposium*, pp. 60-63, 2022.

[11] M.L. Varshika, A.K. Mishra, N. Kandasamy and A. Das, "Hardware-Software Co-Design for On-Chip Learning in AI Systems", *Proceedings of the International Conference on Design Automation*, pp. 624-631, 2023.