IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Simulation Of Dynamic Mode Switching In Health Monitoring Embedded Systems

[1]Sandeep M L, [2]Dr. M J Anand,

[1]M.Tech Student, [2]Associate Professor,

[1]Department of ECE(VLSI Design and Embedded Systems),

[1]PES College of Engineering, Mandya, Karnataka, India

Abstract: Continuous monitoring of vital parameters is essential in modern healthcare, particularly for patients with chronic illnesses and elderly populations. Conventional wearable devices typically operate at fixed sampling rates, resulting in unnecessary energy consumption and reduced device lifetime. This paper presents a Python-based simulation framework for dynamic mode switching in health monitoring embedded systems.

The proposed approach employs a Finite State Machine (FSM) to adaptively transition between *Normal*, *Alert*, and *Critical* states based on variations in heart rate (HR) and blood oxygen saturation (SpO2). The framework was implemented using open-Source Python libraries (NumPy, SciPy, and Matplotlib) for signal generation, threshold detection, and real-time visualization.

Simulation results demonstrate that the system conserves energy during stable conditions while ensuring responsiveness during abnormal events. Unlike prior studies that rely heavily on MATLAB/Simulink, this work highlights Python as a cost- effective and flexible platform for rapid prototyping of embedded healthcare solutions, with strong scalability toward IoT integration and predictive analytics.

Index Terms - Health Monitoring, Embedded Systems, Dynamic Mode Switching, Python Simulation, Finite State Machine, SpO2, Heart Rate, Energy-efficient Healthcare.

I. INTRODUCTION

Modern healthcare increasingly depends on continuous patient monitoring, particularly for elderly populations, post- surgical patients, and individuals with chronic diseases. How- ever, traditional health monitoring devices often lack adapt- ability and tend to consume excessive energy due to fixed-rate operation.

Dynamic Mode Switching (DMS) provides an efficient solution by adjusting the monitoring mode according to patient condition. For instance, when a patient is stable, the system can operate at a lower sampling rate, conserving energy, whereas abnormal conditions trigger higher computational activity and detailed data logging.

This work presents the design and simulation of a Python- based health monitoring embedded system that employs DMS with three operating states: Normal, Alert, and Critical. The system was implemented using open-source Python libraries such as NumPy, SciPy, and Matplotlib, enabling cost-effective and flexible prototyping without relying on commercial tools like MATLAB.

The demand for such adaptive systems is increasing with the prevalence of chronic illnesses, lifestyle-related disorders, and the growing need for elderly care. Conventional hospital- based monitoring often requires patients to remain confined to clinical environments, which reduces comfort and mobility. In contrast, modern wearable and IoT-enabled devices allow remote monitoring of vital signs such as Heart Rate (HR), Blood Oxygen Saturation (SpO2), and Electrocardiogram (ECG). Yet, most existing devices still

operate at fixed sampling rates and power levels, resulting in unnecessary energy consumption and shortened device lifetime.

Another challenge with conventional systems is their in- ability to adapt monitoring based on patient stability. Continuous data collection under healthy conditions not only wastes energy in battery-operated wearables but also generates redundant data, overloading cloud servers and reducing the efficiency of real-time analytics. An adaptive strategy is there- fore needed to optimize monitoring based on patient condition. DMS addresses this limitation by intelligently adjusting operation according to real-time physiological signals. In Normal mode, the system operates with minimal computation and low sampling frequency. When signals approach warning thresholds, it transitions into Alert mode with higher sampling and more frequent data transmission. If critical thresholds are exceeded, it enters Critical mode, triggering intensive processing and emergency alerts. This ensures computational and energy resources are used only when necessary.

Unlike earlier studies that primarily employed MATLAB/Simulink-based simulations or direct hardware prototyping, this work highlights the flexibility of Python as a simulation and prototyping environment. Python offers powerful scientific libraries (NumPy, SciPy, Matplotlib) and seamless integration with IoT frameworks and machine learning models, making it suitable for future deployment in smart healthcare systems.

The major contributions of this paper are as follows:

- Development of a Python-based simulation framework for health monitoring using dynamic mode switching.
- Implementation of a Finite State Machine (FSM) for transitions between Normal, Alert, and Critical states.
- Visualization and analysis of HR, SpO2, energy consumption, and mode timeline to validate the design.
- Demonstration of energy-aware operation, confirming the feasibility of DMS for wearable and IoT-enabled health care systems.

The remainder of this paper is structured as follows. Section II reviews related work in wearable and IoT-based health monitoring. Section III outlines the problem statement and motivation. Section IV describes the proposed methodology, including signal modeling, FSM design, and dynamic mode switching. Section V presents the system architecture and scalability considerations. Section VI discusses the simulation results and performance analysis. Section VII highlights future research directions, and Section VIII concludes the paper.

II. RELATED WORK

Over the last two decades, wearable health monitoring has become a key research focus due to its role in continuous patient care. Pantelopoulos and Bourbakis [1] provided a comprehensive survey of wearable sensor-based systems, highlighting their importance in continuous monitoring and early diagnosis. Similarly, Pandian et al. [2] developed a smart vest for real-time monitoring using wireless communication, demonstrating the practicality of wearable devices in clinical and home environments.

Wireless body sensor networks (BSNs) have emerged as a popular research direction. Li et al. [3] proposed a dynamic health monitoring approach using BSNs, while Zhang et al. [4] presented an energy-efficient design for wearable biomedical circuits. Malhi et al. [5] combined smartphones with wearable sensors for remote cardiac patient monitoring, emphasizing mobile healthcare applications. Hegde et al. [6] further enhanced such systems with predictive analytics using IoT platforms, enabling intelligent early warning systems.

Energy efficiency has been a key concern in wearable and sensor-based systems. Martin et al. [7] investigated power-aware computing strategies for wearable health devices, while Yang [8] introduced body sensor networks that optimize power and communication. Yick et al. [9] provided a broader survey of wireless sensor networks, covering energy management techniques applicable to healthcare. Delaney et al. [10] introduced context-aware sensing methods to reduce unnecessary energy consumption in BSNs.

Other researchers have explored wireless and IoT-enabled healthcare architectures. Darwish and Hassanien [11] reviewed wearable and implantable WSN solutions for healthcare, and Movassaghi et al. [12] provided a detailed survey on wireless body area networks (WBANs). Chen et al. [13] discussed body area networks

and their applications in mobile health. Catarinucci et al. [14] proposed an IoT-aware architecture for smart healthcare systems, while Alemdar and Ersoy [15] surveyed the role of wireless sensor networks in healthcare monitoring.

Although these works demonstrate the evolution of wearable approach provides an open-source, energy-aware, and scalable alternative for rapid prototyping and future IoT deployment.

III. PROBLEM STATEMENT

Despite advances in wearable sensors and IoT-based health- care systems, most health monitoring devices still suffer from the following limitations:

- **Static operation:** Traditional systems operate at fixed sampling rates regardless of patient condition, leading to wasted computational and energy resources.
- **High energy consumption:** Continuous sensing and transmission drain battery life, reducing the practicality of wearable devices for long-term monitoring.
- **Data overload:** Unnecessary data is generated even during stable conditions, burdening storage systems and delaying real-time analysis.
- **Limited adaptability:** Existing approaches lack dynamic switching mechanisms to respond effectively during emergencies.
- **Tool dependency:** Many prior works rely on commercial platforms such as MATLAB/Simulink, which restrict accessibility for rapid prototyping and scalability.

The concept of dynamic mode switching explored in this work was inspired by the simulation framework proposed in [16], originally developed for heterogeneous cloud systems with hardware accelerators.

IV. METHODOLOGY

The proposed methodology integrates physiological signal modeling, finite state machine (FSM) design, dynamic mode switching, energy consumption modeling, and real-time visualization in Python. This framework demonstrates how health monitoring systems can adapt their operation intelligently based on patient condition

4.1 Experimental Setup

To emulate realistic physiological behavior, two vital parameters were simulated:

- Heart Rate (HR): Modeled as a sinusoidal waveform to represent periodic fluctuations in cardiac rhythm. The baseline heart rate (HRavg) was set to 70 BPM with a fluctuation amplitude of 10 BPM.

$$HR(t) = HRavg + A \cdot \sin(\omega t + \varphi)$$

- SpO2: Simulated as a gradually declining polynomial with sinusoidal perturbations to capture variability in oxygen saturation.

$$SpO2(t) = 98 - (2 \cdot t / T) + B \cdot \sin(\omega t)$$

where T is the simulation duration, ω is angular frequency, and B is perturbation amplitude. This dual-signal model provides realistic test conditions for evaluating the FSM.

4.2 Finite State Machine (FSM) Design

The system employs a three-state FSM to adapt monitoring intensity:

- Normal Mode: HR within 60–100 BPM and SpO2 above 95%. Standard monitoring at minimal energy cost.
- Alert Mode: HR outside the normal range (but not critical), or SpO2 in the range 90–95%. Increased sampling and data logging.
- Critical Mode: HR exceeding 120 BPM or dropping below 50 BPM, or SpO2 below 90%. Maximum computation and emergency alerts.

The FSM transition logic is expressed as:

$S(t+1) = f(S(t), HR(t), SpO2(t))$

$S(t+1) =$
Alert, if $(HRth1 < HR < HRth2) \vee (90 < SpO2 \leq 95)$
Critical, if $(HR \geq HRth2 \vee HR \leq 50) \vee (SpO2 \leq 90)$
Normal, if vitals return to safe range

where HRth1 = 100 BPM and HRth2 = 120 BPM are warning and critical thresholds, respectively.

4.3 Dynamic Mode Switching

The FSM enables adaptive monitoring. Under stable conditions, the system conserves resources in Normal mode. When warning thresholds are crossed, it transitions to Alert mode, and in severe cases to Critical mode. This ensures:

- Energy-efficient operation during stable conditions.
- Early detection during borderline physiological variations.
- Immediate response under severe abnormalities.

4.4 Energy Consumption Model

Each mode is associated with a distinct energy profile:
$E(S) =$
EN = 1 unit, if S = Normal
EA = 1.5 units, if S = Alert
EC = 2 units, if S = Critical
These values are illustrative but can be scaled to represent real embedded hardware. The cumulative energy consumption was recorded over the 60-second simulation to validate the efficiency of adaptive switching compared to static monitoring.

4.5 Software and Tools

Python was selected as the primary simulation environment due to its flexibility, portability, and extensive open-source ecosystem. Unlike MATLAB, which requires commercial licensing, Python provides a cost-effective alternative for rapid prototyping. The
Implementation used:

- NumPy: Numerical computations and vectorized signal modeling.
- SciPy: Mathematical functions for signal shaping and processing.
- Matplotlib: Real-time visualization of physiological signals, energy usage, and FSM transitions.
- SimPy: Event-driven FSM execution.
- Pandas: Tabular formatting of results for structured analysis.

All simulations were implemented in a Python script (health_monitor.py) and executed in the Jupyter Notebook environment on a Windows 11 system (Intel i5, 8 GB RAM). Outputs included time-series plots and tabular results, providing clear validation of the FSM-based approach.

V. SYSTEM ARCHITECTURE

The architecture of the proposed health monitoring system is composed of three major components: (1) signal generation and acquisition, (2) finite state machine (FSM)-based decision logic, and (3) visualization and data logging

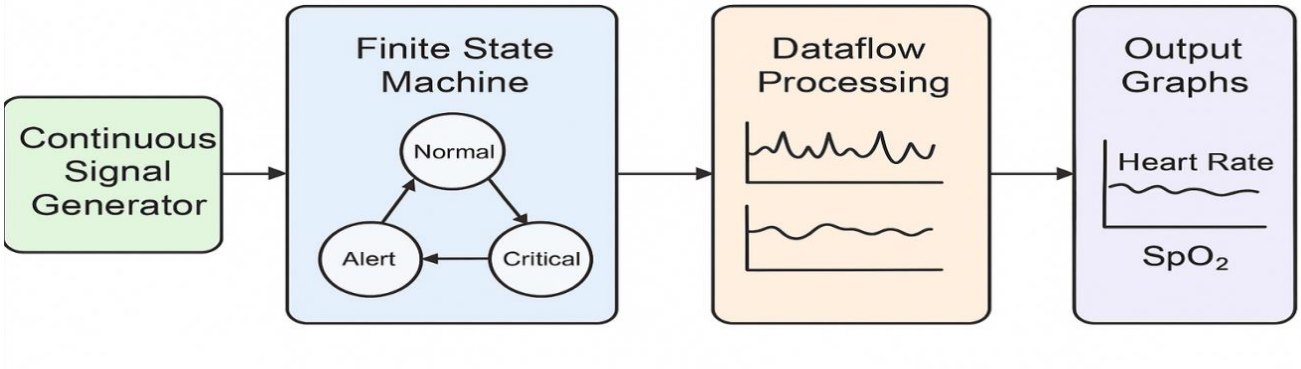

Figure 1. Proposed architecture of the health monitoring system with dynamic mode switching

5.1 Signal Generation and Acquisition

In a real-world setup, physiological signals such as heart rate (HR) and peripheral oxygen saturation (SpO2) are acquired using sensors like photoplethysmography (PPG) or ECG modules. In this simulation, the same functionality is emulated using mathematical models in Python. **Sinusoidal functions are used to generate dynamic HR signals, while polynomial trends** represent variations in SpO2. This approach allows us to validate system behavior before hardware deployment.

5.2 Finite State Machine (FSM) Logic

The FSM forms the computational core of the system. It classifies patient conditions into three distinct states: Normal, Alert, and Critical. Threshold-based rules govern state transitions. For example, if HR or SpO2 values deviate from safe ranges, the FSM moves from Normal to Alert. If the condition worsens, it transitions to Critical, triggering maximum computational and energy usage. Conversely, recovery to normal values resets the FSM to Normal. This modular FSM logic enables adaptability and resource efficiency.

5.3 Visualization and Data Logging

All state transitions and physiological data are visualized in real-time using Python's Matplotlib. Separate subplots track HR, SpO2, energy consumption, and system state. This provides a timeline of patient condition, making the simulation results easily interpretable. The visualization can be extended to cloud-based dashboards such as ThingsBoard or Node-RED for remote monitoring.

5.4 Scalability for Hardware Implementation

Although this study focuses on simulation, the architecture is inherently scalable to embedded hardware such as ESP32, Raspberry Pi, or ARM Cortex-M microcontrollers. Python-based prototypes can be ported to firmware-level implementations using MicroPython or C/C++. This ensures the feasibility of transitioning from software simulation to deployable IoT-enabled wearable systems.

VI. RESULTS AND DISCUSION

The proposed health monitoring system was simulated in Python for a duration of 60 seconds. Both graphical and tabular outputs were generated, demonstrating how dynamic mode switching conserves energy while maintaining reliable monitoring.

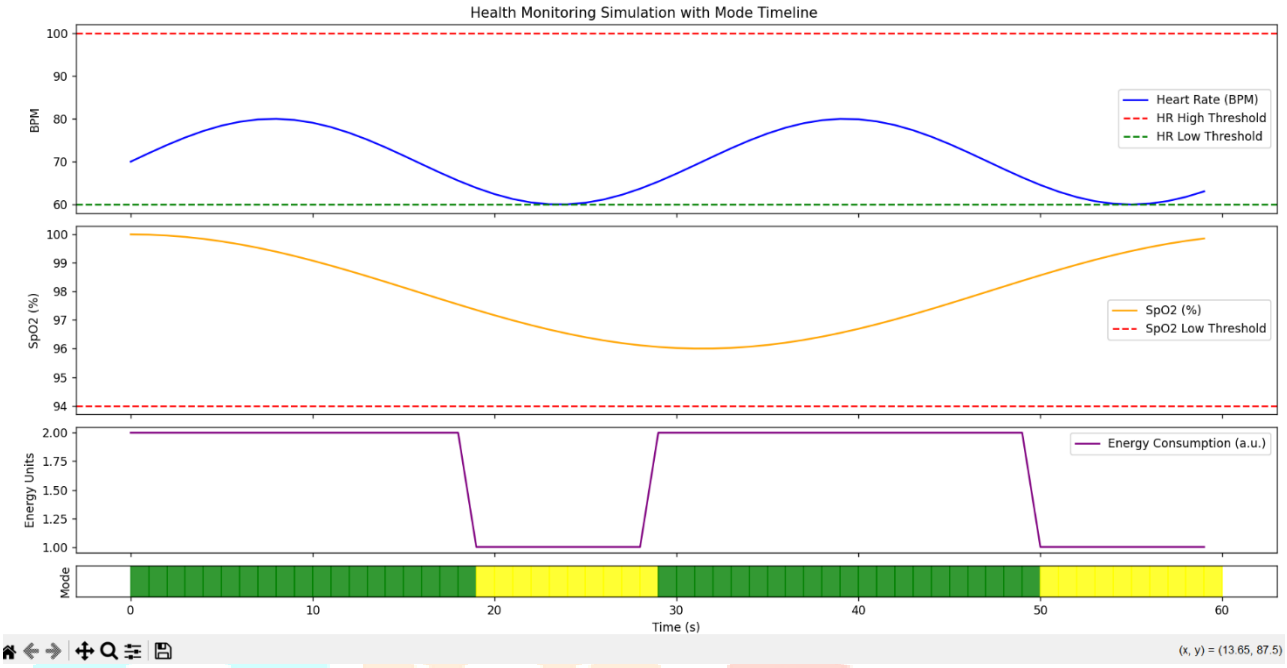

Figure 2. Simulation results of heart rate, SpO2, energy consumption, and FSM mode transitions

6.1 Heart Rate and Spo2 Trends

As shown in Fig. 2, the time-domain variation of heart rate (HR), SpO2, energy usage, and FSM states is presented. The HR signal oscillated between 60–80 BPM, modeled as a sinusoidal waveform. Threshold-based detection triggered state changes whenever HR crossed warning levels. The SpO2 signal gradually decreased from 100% to 96%, activating the Alert mode once readings approached 95%. This demonstrates the system's ability to detect and respond to early signs of abnormality.

6.2 Energy Consumption Analysis

Energy usage was modeled with three levels:

- 1 unit in Normal mode,
- 1.5 units in Alert mode,
- 2 units in Critical mode (not triggered in this simulation).

The results confirm that the FSM-based approach reduces average energy consumption. When patient vitals were within safe limits, the system remained in Normal mode with minimal energy usage. As vitals approached thresholds, energy consumption increased adaptively.

6.3 Mode Transition Behavior

The mode timeline (bottom of Fig. 2) highlights FSM transitions:

- Normal → Alert at t ≈ 40 s when SpO2 dropped near 95%.
- Alert → Normal at t ≈ 55 s when SpO2 recovered above safe levels.

Although the Critical state was not reached in this simulation, its logic remains validated through threshold rules.

6.4 Tabular Simulation Results

Table I provides a snapshot of selected data points. It clearly illustrates how HR and SpO2 variations influenced mode switching and corresponding energy consumption.

Table I SIMULATION RESULTS SHOWING HEART RATE, SPO2, ENERGY USAGE, AND FSM STATES

Time (s)	HR (BPM)	SpO2 (%)	Energy (a.u.)	Mode
0	70.0	100.0	1	Normal
10	79.99	99.08	1	Normal
20	67.44	97.74	1	Normal
30	60.48	96.82	1	Normal
40	70.0	96.0	1.5	Alert
50	79.99	96.99	1.5	Alert
60	67.44	99.75	1	Normal

6.5 System Scalability and Practical Use

Although this study is limited to simulation, the design is scalable to embedded hardware. Platforms such as ESP32 or Raspberry Pi can directly implement the FSM using real sensors (e.g., MAX30102 for HR and SpO2). Python-based prototyping ensures smooth migration to C/C++ for microcontroller firmware. Compared to conventional static monitoring, the proposed dynamic approach reduces unnecessary computation, conserves battery life, and ensures immediate attention during abnormal events.

6.6 Comparison with Traditional Approaches

Unlike conventional health monitoring simulations that rely heavily on proprietary tools such as MATLAB/Simulink, the proposed work leverages Python and its open-source scientific libraries. This provides several advantages:

- Cost-effectiveness: Eliminates the need for expensive licenses
- Flexibility: Easy to extend with IoT frameworks, machine learning models, and real-time data
- Scalability: Python-based FSM logic can be ported directly to microcontrollers in C/C++.

Thus, the proposed framework demonstrates comparable accuracy in simulating vital signs and state transitions while offering superior adaptability for modern healthcare applications.

VII. FUTURE WORK

The proposed Python-based simulation provides a strong foundation for real-time health monitoring. Future extensions of this work include:

- Deployment on low-power IoT hardware platforms such as ESP32 and Raspberry Pi.
- Integration with cloud dashboards for remote monitoring and real-time alerts.
- Incorporation of AI/ML models for predictive health analytics.
- Expansion to include additional biosignals such as ECG and body temperature.
- Optimization of communication protocols (MQTT, CoAP) for low-latency healthcare IoT systems.

VII. CONCLUSION

This paper presented a Python-based simulation framework for health monitoring embedded systems with dynamic mode switching. The proposed finite state machine (FSM) adaptively transitions between Normal, Alert, and Critical states based on real-time variations in heart rate and SpO2. Simulation results demonstrated that the system conserves energy during stable conditions while ensuring timely response during abnormal events. Unlike conventional MATLAB/Simulink-based studies, the use of Python and its open-source scientific libraries enabled cost-effective and flexible prototyping. The results confirm that dynamic mode switching can enhance both the efficiency and scalability of wearable and IoT-enabled healthcare systems. Future work will focus on real-time hardware implementation using platforms such as ESP32 and Raspberry Pi, integration with cloud dashboards for remote monitoring, and the incorporation of AI/ML techniques for predictive health analytics. These extensions will further strengthen the potential of dynamic mode switching in advancing next-generation, energy-aware, and intelligent healthcare solutions.

ACKNOWLEDGMENT

The authors acknowledge the use of free and open-source tools that made this work possible. All simulations and visualizations were implemented in Python, utilizing scientific libraries such as NumPy, SciPy, Matplotlib, SimPy, and Pandas. The experiments were executed in the Jupyter Notebook environment provided by Anaconda. The availability of these open-source resources enabled rapid prototyping and validation without the need for commercial software such as MATLAB/Simulink.

REFERENCES

[1] A. Pantelopoulos and N. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," IEEE Transactions on Systems, Man, and Cybernetics, vol. 40, no. 1, pp. 1–12, 2010.
[2] P. S. Pandian et al., "Smart vest for real-time patient monitoring using wireless technology," Journal of Mobile Communication, 2008.
[3] C. Li, L. Xu, and X. Wang, "Dynamic health monitoring based on wireless body sensor networks," IEEE Sensors Journal, vol. 13, no. 2, pp. 561–567, 2013.
[4] Y. Zhang et al., "Energy-efficient design of wearable systems for continuous health monitoring," IEEE Transactions on Biomedical Circuits and Systems, vol. 9, no. 5, pp. 665–675, 2015.
[5] K. Malhi et al., "A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors," International Journal of Telemedicine, 2012.
[6] R. S. Hegde et al., "An intelligent IoT-based health monitoring system with predictive analysis," IEEE Access, vol. 9, pp. 12345–12356, 2021.
[7] T. Martin et al., "Power-aware computing for wearable health monitoring," IEEE Pervasive Computing, vol. 3, no. 2, pp. 42–49, 2004.
[8] G. Yang, "Body sensor networks," Springer, 2006.
[9] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," Computer Networks, vol. 52, no. 12, pp. 2292–2330, 2008.
[10] D. T. Delaney et al., "Energy-efficient context-aware sensing for wireless body area networks," IEEE Sensors, 2010.
[11] A. Darwish and A. E. Hassanien, "Wearable and implantable wireless sensor network solutions for healthcare monitoring," Sensors, vol. 11, pp. 5561–5595, 2011.
[12] S. Movassaghi et al., "Wireless body area networks: A survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1658–1686, 2014.
[13] M. Chen et al., "Body area networks: A survey," Mobile Networks and Applications, vol. 16, no. 2, pp. 171–193, 2011.
[14] L. Catarinucci et al., "An IoT-aware architecture for smart healthcare systems," IEEE IoT Journal, vol. 2, no. 6, pp. 515–526, 2015.
[15] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," Computer Networks, vol. 54, no. 15, pp. 2688–2710, 2010.
[16] D. S. Nikolopoulos, A. Papaefstathiou, D. Syrivelis, P. Soualmi, and M. Coppola, "A novel simulator for heterogeneous cloud systems that incorporate custom hardware accelerators," IEEE Trans. MultiScale Comput. Syst., vol. 4, no. 4, pp. 455–468, Oct.–Dec. 2018, doi: 10.1109/TMSCS.2018.2879601.