Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 4246596, 29 pages
http://dx.doi.org/10.1155/2016/4246596

Research Article

FPGA-Based Design of an Intelligent On-Chip Sensor Network Monitoring and Control Using Dynamically Reconfigurable Autonomous Sensor Agents

Sani Abba and Jeong-A Lee

Computer Systems Laboratory, Department of Computer Engineering, Chosun University, Dongku SeoSuk Dong 375, Gwangju 501-759, Republic of Korea

Correspondence should be addressed to Sani Abba; saniabba2004@gmail.com and Jeong-A Lee; jalee@chosun.ac.kr

Received 15 September 2015; Revised 24 December 2015; Accepted 28 December 2015

Academic Editor: Johannes M. Karlsson

This paper proposes different low-level microarchitectural designs and frameworks for real-time monitoring and efficient control of on-chip sensor network for field programmable gate arrays (FPGAs). The main goals are to design low power, low-cost, and highly accurate monitoring and control mechanism using autonomous sensor agents and to dynamically reconfigure control of on-chip sensor networks by FPGAs. By collecting dynamic and real-time monitoring parameters such as voltage and temperature, the system becomes self-aware and is able to improve the utilization of FPGA resources and power consumption. The FPGA synthesis, place and route, and implementation were performed for the proposed design. The results after synthesis and implementation show a significant low usage of FPGA logic resources and efficient power consumption of all on-chip sensor components compared with previous approaches. Furthermore, the experimental results from the FPGA-measured on-chip sensor readings show high precision and accuracy in the measured voltage and temperature. Setting the dynamic reconfiguration refresh time at 1000 ms produces highly accurate FPGA-measured on-chip sensor readings compared with those at 100 and 500 ms. The proposed design technique and framework will assist network engineers and system designers by providing flexible and efficient real-time monitoring and control design of large and complex on-chip sensor networks and remote-sensing applications.

1. Introduction

The complexity imposed by on-chip sensor network monitoring and control increases with the scalability of the on-chip sensor network. Therefore, there is the need to have an intelligent and reliable low-level design of intelligent monitoring and control systems using autonomous sensor agents. Because different runtime physical parameters must be monitored, the need for accurate and efficient sensor communication mechanisms is very essential to ensure reliable monitoring and control of complex on-chip sensor network environment. This process also entails the design of low power and high-speed circuits for efficient and reliable network monitoring and control. The use of field programmable gate arrays (FPGAs) for autonomous sensor network monitoring and control is an interesting research domain in which low FPGA logic utilization, low power consumption, and accurate sensor readings are the major metrics for evaluation [1–5]. An autonomous sensor network is a special application for system monitoring, which is composed of several components such as a transducer, conditioning circuits, processing units, and data communication. The major functions of a sensor network are monitoring, collecting runtime ambient parameters, environmental adaptation, and making informed decisions about the observed control parameters [6–10].

As the flexibility and reliability of FPGAs increase, autonomous sensor agents can be embedded in FPGA to measure different runtime parameters. However, designing a sensor network on FPGA is a very difficult task to achieve in a short time because designers are limited to do beyond the FPGA manufacturer design specification. Similarly, the degree of complexity as well as sophistication of design

automation contained in the FPGA makes it difficult for designers to tweak the FPGA circuitry to design and implement sensor networks [4, 5].

The on-chip sensor network runtime parameters such as voltage, current, and cross talk noise are used in controlling and monitoring information from the on-chip sensor network environment. Therefore, as the signals are transmitted within the logic blocks, they encounter delays. This delay component of the logic circuit should be minimized. Hence, the use of FPGA to implement sensor networks will be an efficient solution for a reliable on-chip sensor network packet transmission and retransmission scheme because logic and critical-path delays in FPGAs are minimal. Several methods to overcome these limitations include the use of feedback control system [5, 6, 11–14].

Chu et al. [1] proposed a multiplexing scheme that is simple enough to synchronize the sampling in a multichannel acquisition system with different sampling rates and many combined analog inputs as well as an improved sampling control to enhance system performance. FPGA was used in their design implementation. They used a sampling lookup table (LUT) to design a rule-based synchronous sampling frame to reduce the internal wiring and enhance resource utilization. The results of their experiments demonstrated the benefit of their approach. The proposed scheme can be employed in other related applications. Perera et al. [2] presented the design of low-cost, reconfigurable, and programmable smart sensor node using ZigBee. Their design was implemented using FPGA that incorporated the basic functionalities of the IEEE 1451 standard. The design of the sensor nodes comprised a processing unit and transducers with control capabilities in a single core to ensure reliable processing speed-up as a result of interprocess communication within a die. The experimental results on the basis of the measured pH value and temperature of water samples demonstrated the benefit of their approach.

Campos et al. [15] proposed a distributed autonomic inference machine. This machine could allow the sensor nodes to self-manage and to contextualize the tasks based on fuzzy logic. The results of their experiments demonstrated that the proposed machine is energy efficient by minimizing the number of messages to 48.8% and achieving 19.5% reduction in the energy consumed by the network. The review paper by Portocarrero et al. [16] provided an overview of the recent wireless sensor network (WSN) middleware systems that addressed the autonomic properties. Their aims were determining the best autonomic computing method that will provide a self-management feature of WSNs for middleware systems and studying the various interactions and behavior in WSN components. Their conclusions were summarized as follows: first, they addressed the major concerns about self-configuration, self-healing, self-optimization, and self-protection properties of the autonomous systems. Second, their investigation used diverse methods to manage the dynamic behavior of middleware systems for WSN, which included policy-based reasoning, context-based reasoning, feedback control loops, mobile agents, model transformations, and code generation. Finally, they identified the lack of complete system architecture design that provides full autonomy to the sensor network.

Designing an intelligent monitoring and control system requires the use of smart or intelligent and reliable sensors. The recent work by Echanobe et al. [3] has proposed a system-on-chip-based intelligent multiprocessor embedded system to control ambient environmental parameters. They achieved intelligent capability of their proposed approach by employing the concept of neurofuzzy system, which can reason and adapt to situational environmental changes. The authors Huijsing et al. [12] and Kirianaki et al. [17] proposed the concept of intelligent sensors. These are sensors with embedded intelligence and control system that can perform diverse functionality for environmental monitoring, self-adaptation, runtime observability, and network packet monitoring. Approaches to provide a smart interface for application-specific integrated circuits and FPGAs are provided by the authors of [18–25].

Other approaches to intelligent sensor system monitoring are presented in [26–29]. These approaches focused on the functionalities and communication protocols using transducers to make network communication possible. Ring and delay sensors have been used to study temperature and variations in FPGA [4, 5]. In Franco et al. [5], a ring oscillator was proposed to monitor the temperature in FPGA. This work considered a quadratic frequency in relation to a temperature sensor transfer function and examined how voltage variations affect the sensitivity of a sensor in relation to the number of stages in the oscillator. The authors found that, for larger oscillating chains, the voltage variations can be easily analyzed. Xi et al. [18] used ring oscillators to measure variability and temperature within the processor die. They also measured the power consumption when the system processor die is operating and idle.

The use of sensor networks to measure diverse runtime parameters like cross talk noise, on-chip interconnect temperature, switching activity, clock duty cycle, and other parameters is given in [26–32]. Petrescu et al. [30] proposed a signal integrity and efficient architecture to monitor diverse network-on-chip physical parameters, particularly temperature and voltage. Similarly, McGowen et al. [31] described the control system of a 90 nm Itanium processor, which utilizes on-chip sensors to measure power and temperature and modulates voltage and frequency to optimize the system performance. Sohn et al. [32] proposed a sensor-based solution for static random access memory to mitigate the uncertainties and fluctuation that exist among different device parameters. Their work shows that the information gathered by the on-chip sensors can be used to efficiently enhance reliability and performance of different functional system units.

A small number of methods have been proposed to collect useful information at runtime for on-chip sensor networks. For instance, Chan et al. [33] proposed an approach to use system management bus for communication among IP cores and a thermal-aware power management IP for low-level power management functions. Furthermore, Alexander et al. [26] proposed a hierarchical architecture to collect runtime parameters using network on-chip. Another approach was

proposed by Ciordas et al. [27], which was a monitoring service framework, to support runtime observability of network-on-chip (NoC) behaviors, and application debugging was proposed.

As the complexity and scalability of on-chip sensor network continue to increase, a reliable and efficient low-level design that not only detects runtime ambient parameters but also offers better network performance, scalability, and flexibility of on-chip sensor network design using FPGAs is required. Hence, the use of FPGA for sensor network monitoring and control is very important. The authors in [34, 35] proposed a design of a web server for remote monitoring and control using FPGA. They provided a procedure for the implementation and application of their approach to remote monitoring of sensor networks on FPGA. Gomez Osuna et al. [35] proposed a monitoring infrastructure for FPGA self-awareness and dynamic adaptation. They employed sensor networks to provide dynamic adaptation and obtain data from the FPGA with reduced cost and easy implementation. They showed the significant benefit of their approach. Finally, the authors Kornaros and Pnevmatikatos [13] and García et al. [14] provided a detailed survey of the taxonomy of network-on-chip monitoring and control as well as FPGA-based design of sensor systems. However, the limitations of the previous approaches are as follows: (1) they did not provide real-time monitoring and a control mechanism for accurate measurement of the on-chip sensor network runtime parameters and (2) they did not provide efficient and convenient interface between the user and the FPGA to dynamically tune the operating ranges and reconfiguration refresh time to measure different on-chip sensor values.

In the current paper, we propose a low-level microarchitectural design infrastructure for real-time monitoring and control of on-chip sensor network-based systems for FPGA. We develop the design and implementation for efficient and reliable high-speed circuit technique of the on-chip sensor network runtime parameter monitoring and control, which use autonomous sensors that reside at the network interface (NI), to be dynamically configured and to communicate the runtime ambient parameters to the network monitor controller hardware. We provide detailed low-level design methodology and procedure and a user application for efficient and reliable on-chip sensor network monitoring and control for FPGAs. We also provide a webpage interface for the user to dynamically reconfigure the FPGA to conduct on-chip sensor readings and adjust the sensor ranges and reconfiguration refresh time. The proposed approach will provide engineers and system designers with a flexible and efficient real-time monitoring and control scheme for large and complex FPGA-based on-chip sensor networks. The prototype was implemented in FPGAs and the technology is applicable for ASIC as well.

The salient contributions of this work are as follows:

(i) We propose a low-level microarchitectural design and infrastructure for real-time intelligent network monitoring and control for reconfigurable on-chip sensor network for FPGA. The intelligent sensor agents collect runtime ambient parameters such as voltage and temperature for efficient monitoring and control of on-chip sensor networks.

(ii) We propose a real-time triggered protocol for efficient communication between the sensors and the monitoring and control unit.

(iii) We provide a detailed low-level design procedure and implementation of the on-chip sensor network, sensor NI, and monitoring and control unit using both Synopsys and Xilinx design tools. We implemented the proposed design as a case study using Xilinx Vivado 2013.3 integrated design environment and Zynq 7000 FPGA device. The results after synthesis and implementation, including the device resource utilization, showed low FPGA logic resource consumption and efficient power utilization by the on-chip sensor components.

(iv) We performed experiments using FPGA to measure and control the on-chip sensors. The experimental results from the FPGA-measured and controlled on-chip sensor readings show high precision and accuracy in the voltage and temperature readings. We found that setting the dynamic refresh time at 1000 ms produced accurate FPGA-measured on-chip sensor readings compared with those at 100 and 500 ms.

The rest of this paper is organized as follows: Section 2 presents the basic background; Section 3 discusses the design methodology; Section 4 discusses the simulation environment and simulation results; Section 5 presents the experimental setup, case study, and FPGA resource utilization and comparison with previous approaches; Section 6 presents the experimental results and discussions; and Section 7 provides the concluding remarks.

2. Background

The IEEE 1451 standard provides fast, efficient, and reliable means of converting transducer components into an intelligent sensor interface for efficient communication and transmission of sensor data to various NIs. A transducer is an electronic device that converts electrical signals from one form of energy to another. Therefore, a sensor is a special type of transducer that produces analog or digital electrical signals that represent the sensed physical, biological, or other environmental parameters. On the other hand, an actuator is a transducer that uses an electrical signal as an input and performs the required physical functionality [7–10].

An intelligent or smart sensor is a combination of both analog and digital transducer components, a central processing unit, and a network communication interface such as controller area network (CAN), interintegrated circuit (I2C), local interconnect network (LIN), and universal asynchronous receiver and transmitter (UART). It is also composed of hardware or software and conditioning circuits for sensor diagnostics and calibration, in addition to the communication mechanism and interface. Figure 1 shows the IEEE 1451 standard for a smart transducer interface (TI). The architecture comprises four subsystems: transducers

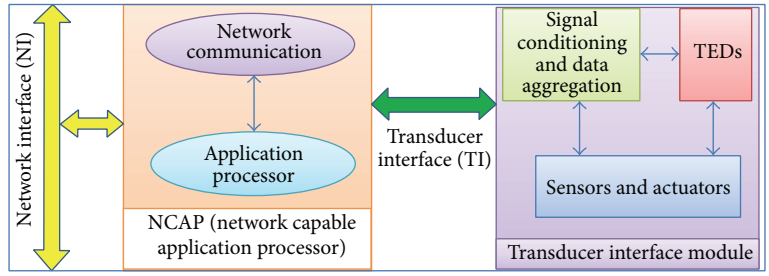


Figure 1: IEEE 1451 standard for smart TI [7].

consisting of sensors and actuators, signal conditioning circuit, data conversion subsystems, and application processor and network communication. The sensor output signal is conditioned by the conditioning circuit and is multiplied and converted into a digital signal using an analog-to-digital converter (ADC). The digital signal from the sensor is processed by a microprocessor with the help of a monitoring and control algorithm. The sensed parameters can be transmitted to the monitoring and control system using the network communication algorithm [7–9].

As defined by the IEEE 1451 standard, a smart transducer is an intelligent system that provides enhanced capability beyond sensing and provides a mechanism for control of the sensed information. This feature provides a means of combining transducers for different applications in a controlled network environment with a high level of efficiency and reliability. Furthermore, the IEEE 1451 architecture provides functionalities such as self-diagnosis, self-description, self-identification, self-calibration, data processing, location awareness, and time awareness. The various components of the IEEE 1451 smart transducer architecture are shown in Figure 1. The network capable application processor (NCAP) subsystem provides the functionality for application processing and network communication mechanism between the sensors and actuators. The TI module subsystem is composed of a transducer signal conditioning and data conversion, which consists of a number of sensors and actuators. The TI provides communication methods and algorithms for efficient transmission of sensed data. Finally, the NI provides a means of communication between the NCAP and the outside devices or networks [7–10].

2.1. On-Chip Sensor Network Monitoring and Control System. The on-chip sensor environment is a very complex and dynamic system. Monitoring of the on-chip sensor as a dynamic system demands communication of the sensed data to a controller and the basic information regarding the controller input parameters to the actuators. Therefore, the structure, topology, switching, and arbitration of the on-chip sensor network solely depend on the monitoring, control, and communication mechanism or protocol used by the system. Hence, two major challenges need to be addressed: provision of efficient and reliable communication mechanism used by the network and provision of central and distributed control system that ensures real-time sensing, monitoring, and efficient processing of the on-chip sensory information. Therefore, an autonomous monitoring and control system for on-chip sensor network must consider the communication protocol and the constraints in relation to the clock time, network topology, switching, and arbitration [36–38].

On-chip sensor network monitoring and control can be regarded as a closed-loop system, which means that controllers are provided to connect to a feedback system, and sensors and actuators are installed to determine the overall function of the whole system. The on-chip sensor network must consider and address the following fundamental issues in a controlled loop system [20–25, 36–39]:

(i) Suitability of the on-chip sensor network communication topology for a particular application.

(ii) Addressing the constraints imposed by the communication mechanism, such as packet loss, network latency, delay, routing, and switching.

(iii) Determining the monitoring and control goals that must be observed to facilitate proper coordination of each component in the whole network monitoring and control system.

To address these issues, there is the need to have an efficient and reliable monitoring and control infrastructure that will provide all the needed design methodology and framework to achieve the stated goals and objectives.

Figure 2 shows an intelligent monitoring and control system for on-chip sensor network using autonomous sensor agents. It shows the data transfer among different nodes in the system, that is, from the sensors to the controller and from the controller to the actuators. The on-chip sensor signal $K_n(\tau)$ is different from the on-chip network controller input signal $\widehat{K_n}(\tau)$, and the monitored controller output $L_m(\tau)$ is different from the actuator input $\widehat{L_m}(\tau)$. This information implies that signals $\widehat{K_n}(\tau)$ and $\widehat{L_m}(\tau)$ are the original network signals $K_n(\tau)$ and $L_m(\tau)$, respectively, which are transmitted through the communication links for the sensors, actuators, and controller. Signals R and K_{ref} are disturbances or noise that must be controlled both in the input and in the output to obtain the desired results. Figure 2 shows single-headed arrow lines representing continuous time signals, whereas the dashed arrow lines represent the data transmission links at a given transmission time, for example, τ_h, for $h = 0, 1, 2, \ldots$. During the transmission of the sensor data, the process produces a time delay that must be controlled and monitored. Similarly, time is another issue that needs to be considered:

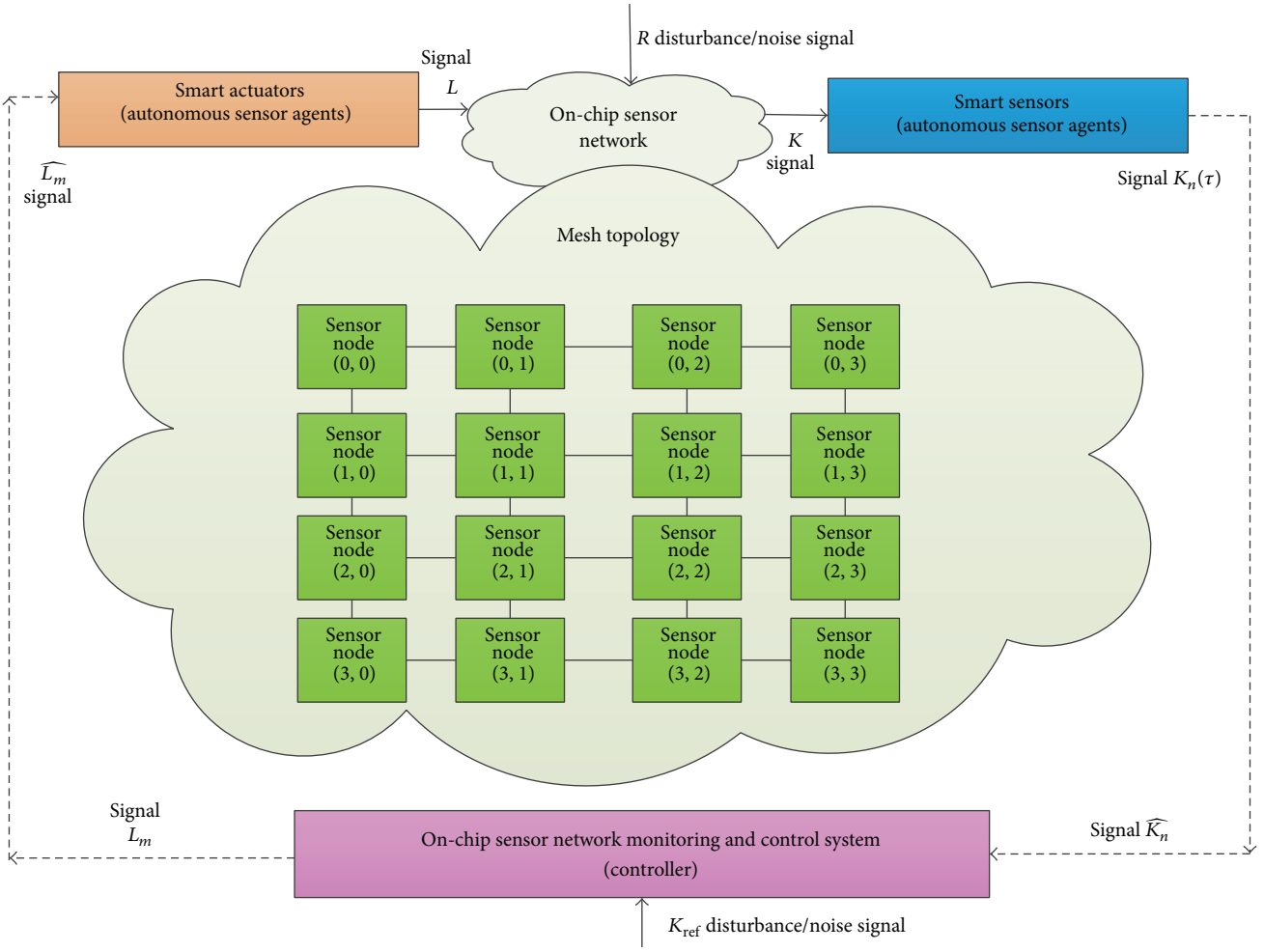


Figure 2: On-chip sensor network monitoring and control system using autonomous sensor agents.

for example, at what time interval the communication is needed and required. The transmission of the sensor data is constrained to a small number of bit rates, which implies that the values of signals $K_n(\tau)$ and $L_m(\tau)$ transmitted over the on-chip sensor network are constrained to a fixed bit length. Therefore, the monitoring and control system should ensure reliable and efficient communication at different time instances [36–38]. On the basis of these limitations, FPGA-based design and implementation of an intelligent on-chip sensor network monitoring and control are essential. Hence, our proposed design methodology provides a better solution for on-chip sensor network monitoring and control using autonomous sensor agents.

3. Design Methodology

This section presents the design methodology of the proposed on-chip sensor network monitoring and control system with dynamic reconfigurable capabilities [26–49]. The proposed design detects runtime ambient parameters and communicates the sensed data to the monitoring and control hardware unit. Figure 3 shows the architectural design of the proposed monitoring and control system using autonomous sensor agents that reside in the NI to sense and report various parameter values to the hardware controller for onward analysis and processing. The architecture is a mesh network that is well suited for on-chip sensor die structure [13, 14, 47, 49]. It consists of 12 nodes (gateway) and IP cores. The NI provides a means of communication among the sensors and actuators, network nodes, and hardware monitor controllers. At each NI, three intelligent sensors are installed, namely, voltage, thermal, and temperature sensors. These sensors continuously sense and transmit the sensed data to the hardware controller that then processes the data and provides the necessary actions to keep the system in a smooth and healthy condition. The sensors communicate the sensed data via a standard protocol. This protocol is shown in Figure 4.

The temperature sensors measure the temperature variations to ensure that the system temperature does not exceed a certain threshold. The sensed data from the sensors are aggregated between four or more temperature sensors in different NIs. These temperature sensors form a cluster of

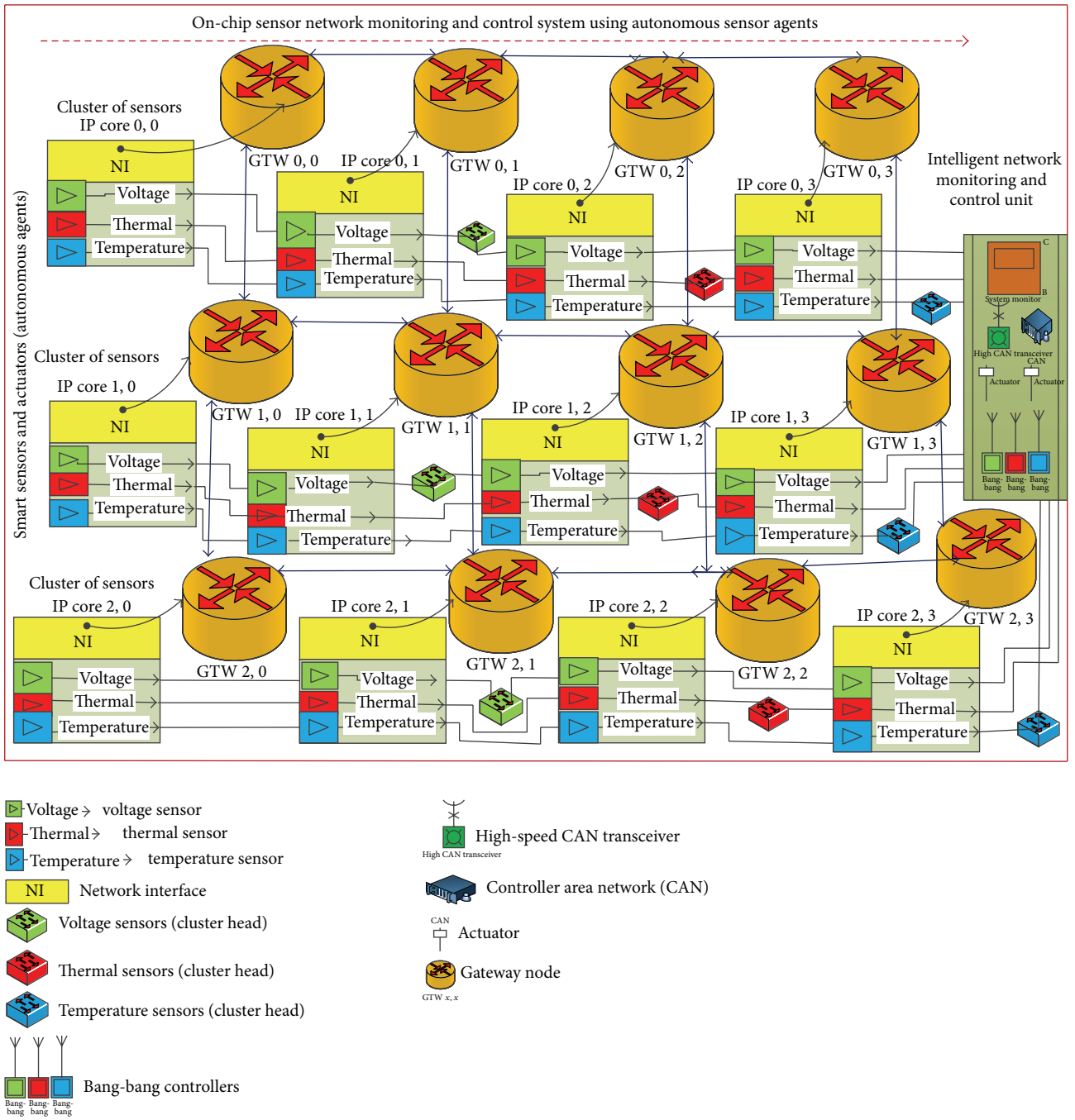


FIGURE 3: Proposed on-chip sensor network monitoring and control system using autonomous sensor agents on 4 × 3 mesh architecture.

sensors, and a cluster head (subcontroller) among them is set to collect and aggregate the sensed data from all sensors in the cluster and transmit them to the hardware monitoring and control system for further processing.

The thermal sensors measure the amount of heat dissipation resulting from the on-chip sensor cross talk noise and the data transmission and retransmission operations. For the thermal sensor operation, the procedure is similar to that of the temperature sensors where four or more thermal sensors form a cluster, and the cluster head (subcontroller) transmits the aggregated sensed data to the on-chip monitor controller hardware. The voltage sensors measure the voltage variations resulting from the current and resistance on the on-chip logic and the analog circuits. These sensors determine any transient and variation in the voltage drops across different components in the on-chip sensor network. Here, four or more voltage sensors in different NIs can form a cluster, and their sensed voltage input data are then combined for transmission to the monitoring and control unit. This process will ensure smooth and reliable

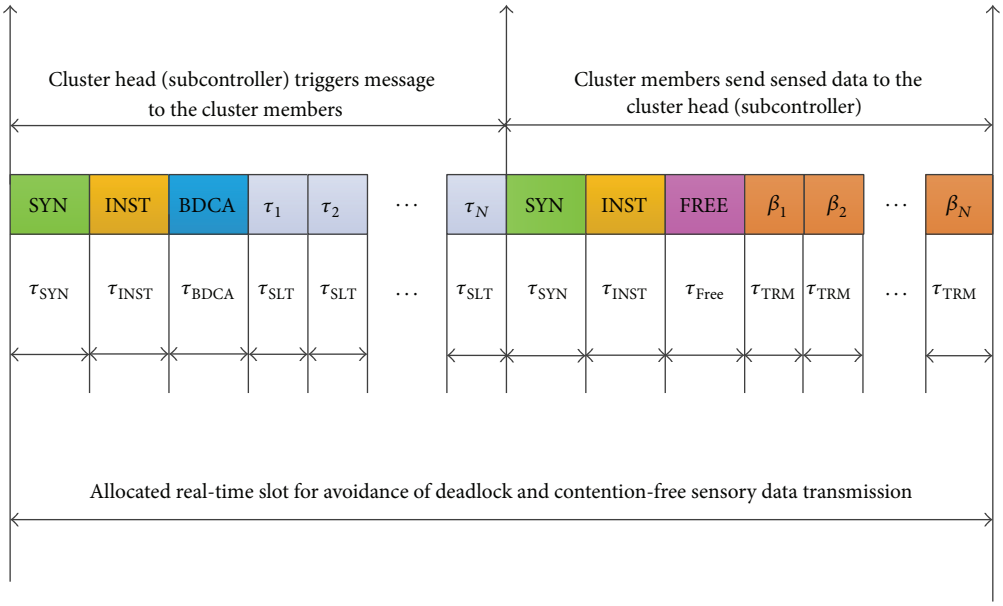


Figure 4: Proposed real-time triggered on-chip sensor network communication protocol.

monitoring and control of the entire on-chip sensor network [1–6, 11, 15–23].

Figure 3 shows that the intelligent monitoring and control system is composed of smart sensors and actuators that represent the autonomous agents, subcontrollers, main control unit, system monitor module, high-speed transceiver, and CAN controller. The monitoring and control system performs both central and distributed monitoring. Each sensor can communicate with its own cluster members and can perform data aggregation and forwarding to the main monitoring and control unit. Therefore, a distributed monitoring and control system is well established between the autonomous sensor agents and the main controller unit.

Figure 4 shows the real-time triggered protocol [6, 29, 48]. This protocol is deadlock and contention-free. The sensor nodes in each cluster are provided with real-time slots in which they receive sensed data from other nodes to communicate with the cluster head (subcontroller) or to transmit its own sensed data to the subcontroller. Communication among the sensors is synchronized in real-time to avoid delay and loss of sensor data. The subcontroller will trigger a message to all its cluster members to query if sensed data are available for transmission. The members will trigger a real-time message in form of response to the cluster if they have data to transmit, and a real-time frame will be allocated and synchronized in real-time for each member to avoid communication delays and loss of sensor information. The explanation of the variables used in the proposed protocol shown in Figure 4 is presented below:

SYN: it defines the real-time synchronization period between the cluster head (subcontroller) and cluster members.

INST: it defines the real-time interval between the transmission and reception of the sensor data.

BDCA: it determines the real-time in which sensory data are transmitted by the cluster head to the cluster members.

$\tau_1, \ldots, \tau_N$: they define the real-time slot during which the cluster members listen to triggered real-time message from the cluster head.

FREE: it is defined as a real-time slot in which the cluster head (subcontroller) receives data from the cluster members; the cluster head uses the free slot in real-time to communicate with the on-chip monitor controller hardware on the status of the network.

$\beta_1, \ldots, \beta_N$: they define the real-time interval in which the cluster members transmit data to the cluster head.

3.1. On-Chip Sensor Network Monitoring and Controller Interface Design. Figure 5 shows the logic design of the main on-chip sensor network monitoring and controller unit. The design consists of several components that form the controller unit and interface to the different sensor networks. The main monitoring and controller design include the following: three bang-bang (on/off) controllers, a time domain system analyzer implemented as the main monitoring unit, 8:1 multiplexer (MUX) interface, ADC, smart sensors, actuators, a high-speed CAN transceiver, autonomous voltage sources, CAN engine controller, clock sources, and voltage-monitoring circuits.

The reference voltage source supplies the required voltage to power the controllers and all the associated components of the monitoring and control system. The source is configured to generate different output sources such as a step and a periodic pulse. The initial pulse parameter and the amplitude

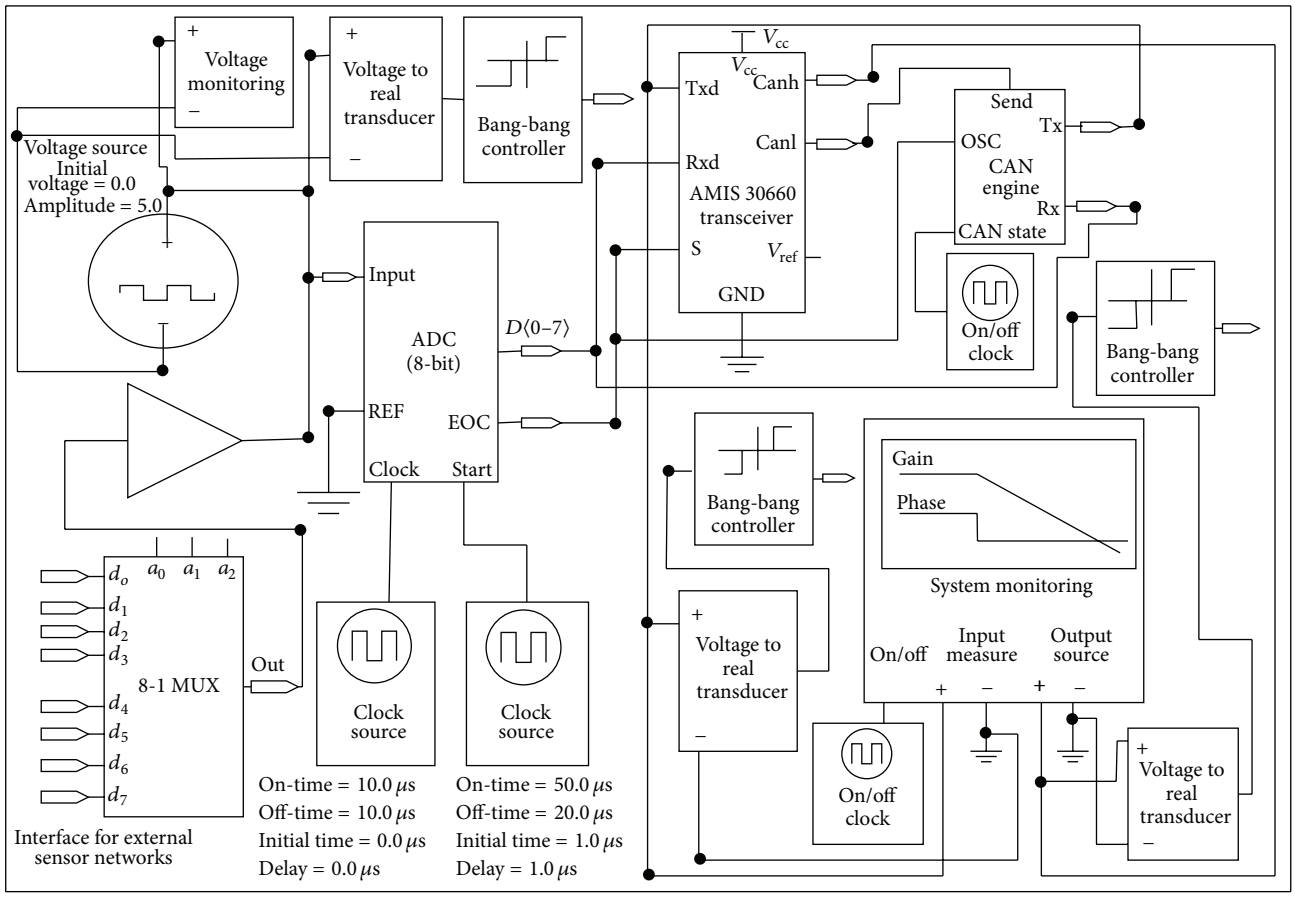


Figure 5: On-chip sensor network monitoring and controller hardware interface design.

values are set to 0.0 and 0.5 V, respectively. The 8:1 MUX interface provides the input from the output of the different sensors to the main controller system. It also acts as a conditioning circuit. Different sensor signals are conditioned (aggregated) to form a single strong analog signal, which is then amplified or buffered, as shown in the logic design. The use of a buffer is beneficial for interfacing with the sensors. It buffers the voltage to be fed into the ADC input. The output of the MUX is then fed into the ADC, which converts the analog signal into a digital form.

Two clock sources are connected to the ADC to provide a clocking mechanism for data conversion and transmission. The clock sources of the ADC are configured with the following settings: (on-time = 10.0 μs, off-time = 10.0 μs, initial time = 0 μs, and delay = 0.0 μs). Similarly, the clock source connected to the start pin of the ADC is configured with the following settings: on-time = 50 μs, off-time = 20 μs, initial time = 1 μs, and delay = 1 μs. These parameter values can be tuned to control both the input and output of the ADC. The output of the ADC is fed into AMIS-3660 high-speed CAN transceiver, which serves as an interface between the ADC and the CAN engine. The *eoc* pin of the ADC is connected to *s* pin of the CAN transceiver. The 8-bit digital output of the ADC is connected to the receiver (Rxd) input of the CAN controller. The Vcc input is connected to the Vcc source, and the ground pin is connected to the earth (ground). The transmitter (Tx) pin of the CAN transceiver is connected to the Tx pin of the CAN engine. The receiver (Rx) pin of the transceiver is connected to the Rx pin of the CAN engine, which provides a means of transmission and retransmission of sensor data and direct monitoring and control of sensor information [21, 40, 41, 50].

The CAN engine, together with the transceiver, provides a communication interface to the entire system and serves as NCAP, as defined by the IEEE 1451 smart TI standard [7–10]. The voltage-monitoring module block measures the voltages from the sensor input to the control unit. We set a limit for the upper and lower voltages. If the voltage falls below a certain level (0–5 V), it is reported to the control system, and appropriate actions are taken. Smart voltage transducers are attached to the controller input and output pins to monitor the variation in the input and output voltages from the sensors and actuators to observe and control the transmitted information.

The time domain system analyzer is the main monitoring interface that provides voltage analysis of the converted signals from sensor NI. The voltage can be swept within a given frequency range while performing both transient and time domain analysis of the monitored control system.

```
Pseudocode for the bang-bang controller module operation
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **//
Var Positive_Threshold_value = 5.0 V;
Var Constant_output_value = 1.0 V;
Begin {
If Controller_Input >= Positive_threshold
 Then
       Controller_Output = Positive_Constant_value;
If Controller_Input <= Negative_Threshold
 Then
       Controller_Output = Negative_Constant_Value;
Else if Controller_Input is between Positive_Threshold and Negative_Threshold
      Then
       Controller_output = 0;
      End if
 End if
End if
End
```

Pseudocode 1: Pseudocode of the bang-bang (on/off) controller module operation.

The controller compares the converted digital input signals from the CAN transceiver and the output signal from the CAN engine. It then calculates the phase and gain responses of the voltage frequencies during the sweeping process. This information is fed into the simulator, and waveform plots of the signals are displayed in the waveform window of the simulator for further analysis. Figure 5 shows two actuators attached to the system analyzer input and output ports to measure the actual transmitted sensor signals. This process will ensure that exact sensory information is transmitted and received at both input and output of the monitored and control system.

The bang-bang (on/off) controllers were used to control and monitor the sensor information between the sensor interface and the main hardware controller unit. We set the positive_threshold voltage to 5.0 V and the constant_output_value to 1.0 V in order to control the output voltages from the sensors, actuators, and main control unit [40–44]. The pseudocode for the bang-bang controller is shown in Pseudocode 1.

3.2. On-Chip Temperature Sensor NI Design. Figure 6 shows the on-chip temperature sensor NI design. The design consists of several components: an autonomous temperature source, four smart transducers (temperature sensors), four smart TIs to the temperature sensors, four operational amplifiers, 4-1 MUX, ADC, and two clock sources.

The temperature source is the main source of thermal energy for the sensors and the TI, which generates a temperature difference across its input ports in a form of a pulse or a pulse train. The generated temperature is not related to the amount of heat that flows through the source, but it depends on the differential voltage across the input terminals of the source. Furthermore, the positive polarity of the source defines the temperature differential between the highest ($\text{Temp}_{\text{high}}$) and lowest ($\text{Temp}_{\text{low}}$) values of the port of the temperature source [8–10, 40–44]. The temperature differential is expressed by the following equation:

$$\text{Temp}_{\text{diff}} = \text{Temp}\left(\text{Temp}_{\text{high}}\right) - \text{Temp}\left(\text{Temp}_{\text{low}}\right), \quad (1)$$

where $\text{Temp}_{\text{diff}}$ is the temperature difference and $\text{Temp}(\text{Temp}_{\text{high}})$ is the temperature in degree Celsius at the port ($\text{Temp}_{\text{high}}$). $\text{Temp}(\text{Temp}_{\text{low}})$ is the temperature at the port (Temp_{low}). The two parameters of the temperature source are set at the following conditions: initial voltage = 0.0 V and pulse = 5.0 V. This condition provides a continuous temperature source for the four smart temperature sensors and the TI. The control sensor senses the temperature signals from the source and converts them into a variable signal. The temperature model is expressed by the following equation:

$$\begin{aligned}\text{Output}_{\text{temp}} = R * \left(\text{Temp}\left(\text{Temp}_{\text{high}}\right)\right) \\ - \text{Temp}\left(\text{Temp}_{\text{low}}\right),\end{aligned} \quad (2)$$

where the two pins ($\text{Temp}_{\text{high}}$) and ($\text{Temp}_{\text{low}}$) are terminal pins. The difference in the two temperature values defines the input signal. Parameter R is the gain, which can take any arbitrary values. Here, we set the default value of R to be one. The output parameter has no specific signal value. Therefore, we need TI to convert the output signal into an electrical signal. To achieve this, we interface the output of the temperature sensor with the transducer (control to voltage interface) to convert the temperature into electrical signals. The TI is based on the characteristics expressed by the following equation:

$$\text{Input}_{\text{Sensor}_{\text{Signal}}} = R * \left(\text{Volt}\left(\beta\right) - \text{Volt}\left(\alpha\right)\right), \quad (3)$$

where the difference $\text{Volt}(\beta) - \text{Volt}(\alpha)$ is the differential voltage from the sensor output and R is the gain that is set to

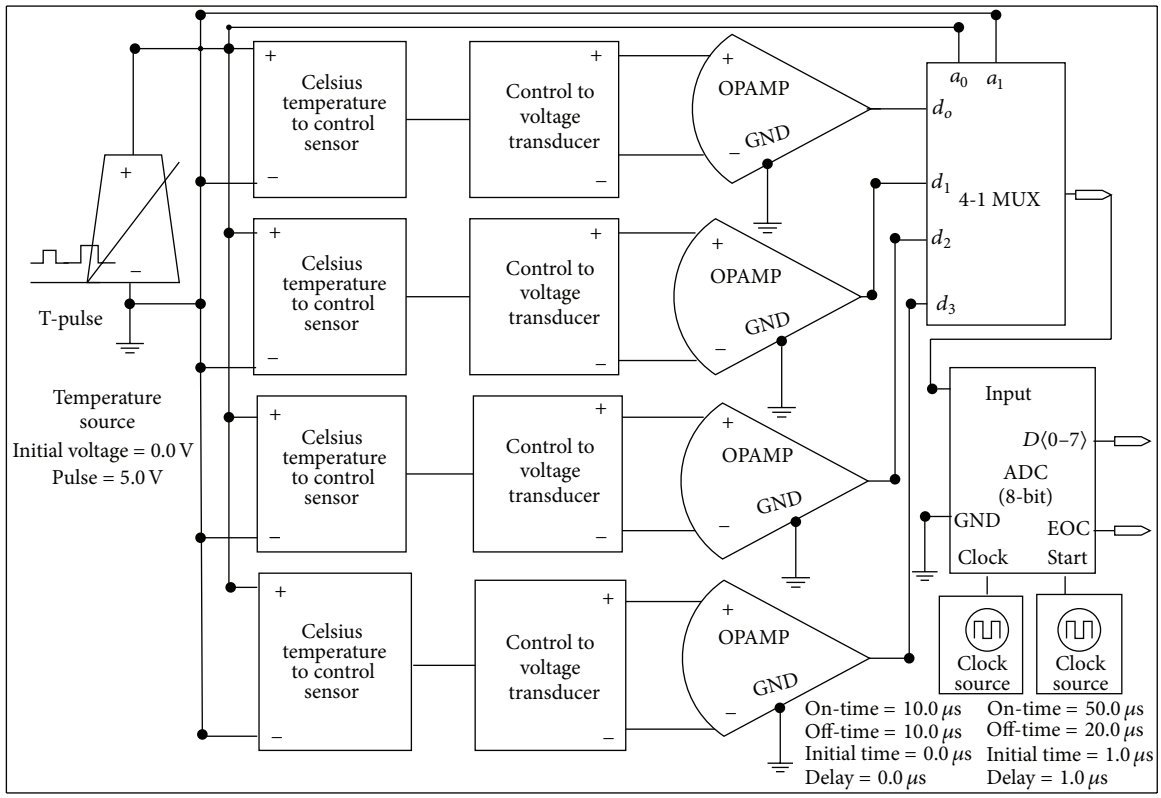


Figure 6: On-chip temperature sensor NI design.

one. The TI output is converted into a voltage and amplified using an operational amplifier. The converted output voltage from the TI ($\text{Input}_{\text{Sensor}_{\text{Signal}}}$) is used as an input to the operational amplifier because the operational amplifier is suitable for interfacing the sensor data and buffering the input voltage for maximum gain. The amplified voltages from the four operational amplifiers are used as input to the 4-1 MUX, which serves as a conditioning circuit for the sensor interface. The four inputs are multiplexed to form a single output that is fed into the ADC for conversion into digital signals. The 4-1 MUX has four data inputs, namely, d_o to d_3, and two address input pins a_1 and a_0, as shown in Figure 6. The 4-1 MUX functions as follows: one of the four inputs is chosen to drive the output state y. The selection is based on which among the four possible input combinations are available [1–7, 11, 15, 16, 40–44].

The output of the 4-1 MUX (y) is used as an input to the ADC. The ADC converts the analog sensor data into a digital form. However, the output (y) from the 4-1 MUX can be sent to the main controller representing one cluster of the four sensors' data. In this case, the conversion from analog to digital is performed by the ADC on the main monitor controller unit. To ensure timely and accurate data conversion by the sensors, two clock sources are connected to the *start* and *clock* pins of the ADC. The clk1 source is configured with the following parameter configurations: on-time = 10.0 μs, off-time = 10.0 μs, initial time = 0.0 μs, and delay = 0.0 μs. The start clock source is configured with the following settings: on-time = 50 μs, off-time = 20 μs, initial time = 1 μs, and delay = 1 μs.

3.3. On-Chip Voltage Sensor NI Design. Figure 7 shows the design of a voltage sensor NI. The design consists of autonomous voltage source, smart voltage sensors, four TIs, four operational amplifiers, one MUX, ADC, and two clock sources. The autonomous voltage source is used as the main source for power supply to the voltage sensors and TIs. The two parameters of the voltage source are configured to supply continuous voltage to all the components in the system. The values of the parameters are as follows: initial voltage = 0.0 V and amplitude = 5.0 V. The characteristic equations for the voltage sensors (voltage to control interface) and the TI (control to voltage) are similar to (1)–(3). The output of the TI is fed to the operational amplifiers, which is then amplified and sent to d_o to d_3 input pins of the 4-1 MUX [8–10, 40–44].

3.4. On-Chip Thermal Sensor NI Design. Figure 8 shows the smart thermal sensor NI of the main controller unit. The interface consists of several components similar to the two interfaces for the temperature and voltage sensors described in Sections 3.2 and 3.3. A thermal power source is used to power-on the four sensors and the TI. The two parameters for the thermal power source are configured with the following settings: amplitude = 5.0 V and frequency = 25 Hz. The source generates a sinusoidal heat at its input ports. The thermal

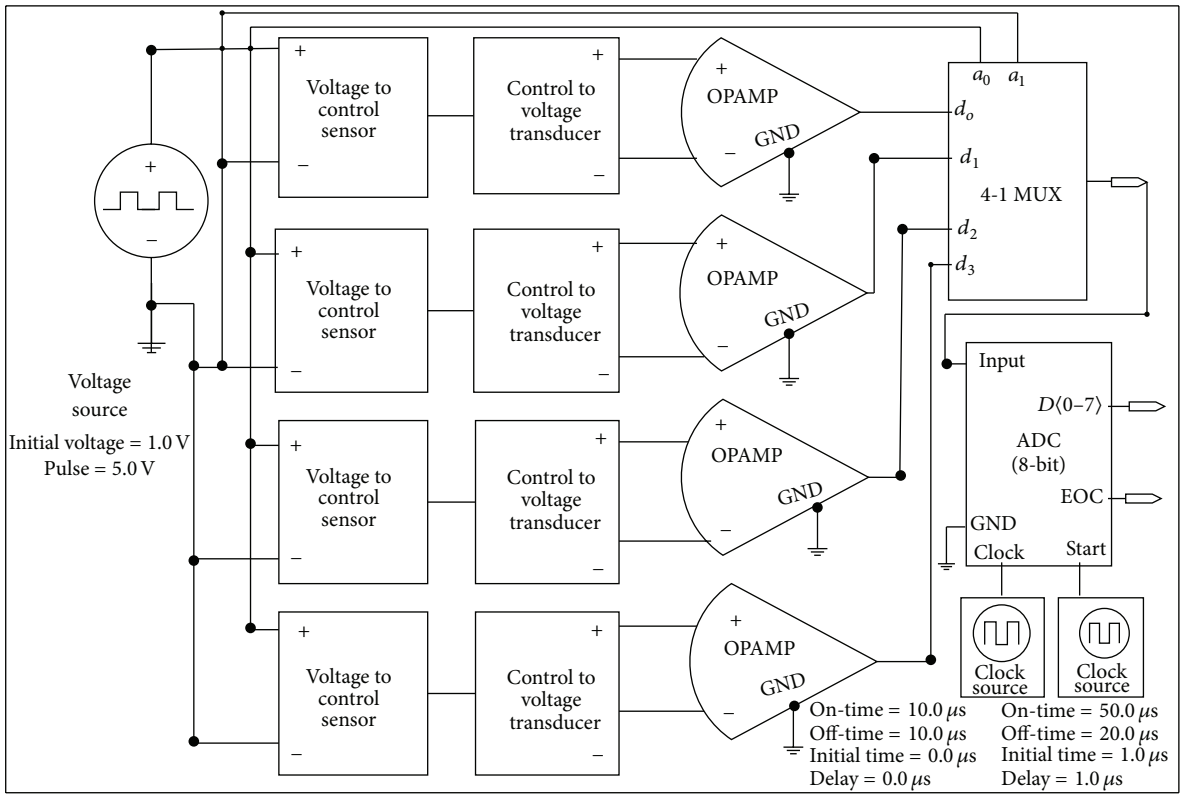


Figure 7: On-chip voltage sensor NI design.

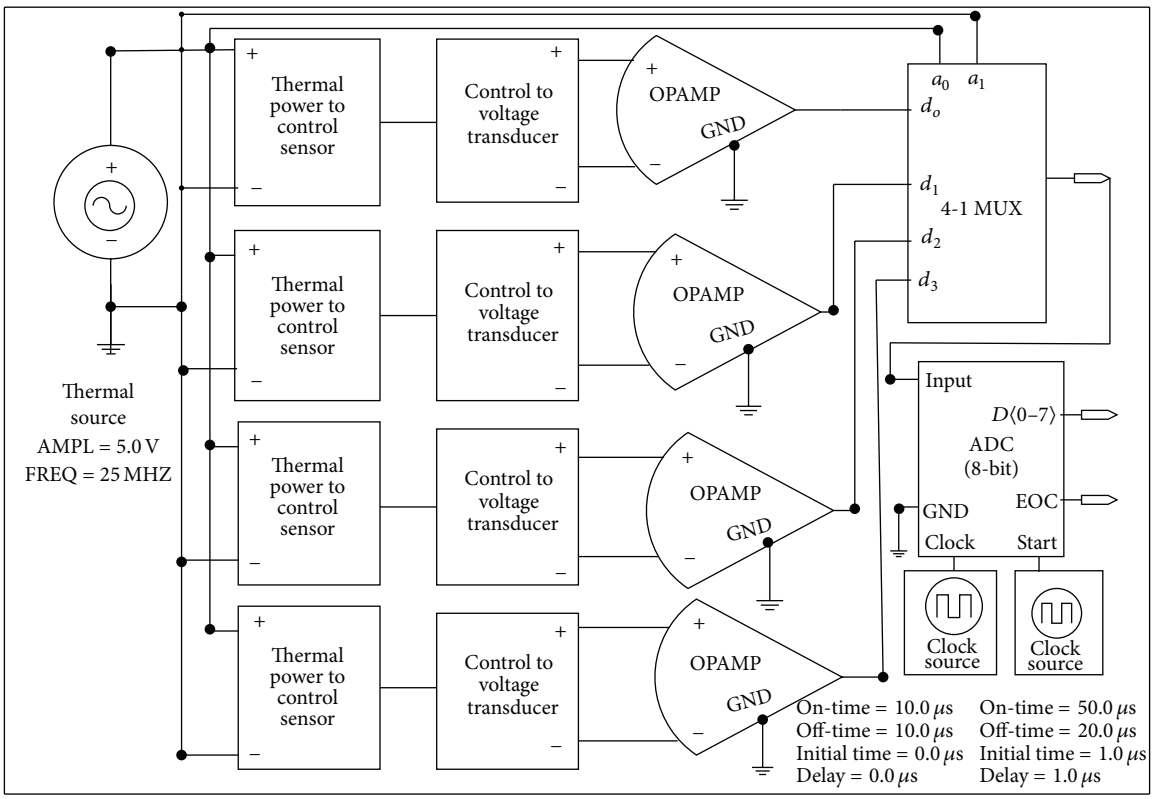


Figure 8: On-chip thermal sensor NI design.

sensors (thermal to power control) sense and transmit the sensor data to the smart TI. The characteristic equation for the thermal sensor model is expressed as

$$\begin{aligned} &\text{Output}_{\text{Sensor}_{\text{data}}} \\ &\quad = K * \left(\text{flux}\left(\text{Thermal}_{\text{high}} - \text{Thermal}_{\text{low}}\right)\right), \end{aligned} \tag{4}$$

where $\text{Thermal}_{\text{high}}$ and $\text{Thermal}_{\text{low}}$ parameters are the thermal pins that measure the temperature across the two input pins and flux is the thermal power which is expressed in degree Celsius. The input signal to the sensor is expressed by the function flux($\text{Thermal}_{\text{high}} - \text{Thermal}_{\text{low}}$). Parameter K is the gain constant. We set $K = 1$ during the simulation. The sensor output is then sent to the input port of the TI (control to voltage), which converts the sensed data into electrical voltage and then amplifies them using the operational amplifiers used as input by the 4-1 MUX [40–44, 50].

4. Simulation Environment

This section explains the detailed simulation procedure carried out to verify the proposed design. The design and implementation of the register transfer level (RTL) modules, models, and schematics of the proposed intelligent network monitoring and control system presented in Section 3 were performed using the Synopsys integrated design environment and tools [40, 41]. The Synopsys SaberRD provides hardware programming environment, design tools, and model libraries for designing low-cost and high-speed circuits and simulation environment. The tools are used to verify the functionality and efficiency of the circuit design under test. The different parameter values can be changed to verify the design functionality to ensure reliable and efficient network monitoring and control. We verified the functionality of the proposed design specification in the time domain using transient analysis to determine the variation in the sensor NI voltage signals and that in the monitoring and control unit over time. The simulation was run by setting the following parameter configurations:

Simulation time (end time) = 10,000 ms (10 s).

Time step = 10 ns.

Start time = default = 0 s.

Threshold voltage = 0–5 V.

ADC clock signal setting (on-time = 10 μs, off-time = 10 μs, initial time = 0 μs, and delay = 0 μs).

ADC start signal setting (on-time = 50 μs, off-time = 20 μs, initial time = 1 μs, and delay = 1 μs).

The simulation time (end time) defines the time in which the simulator finishes the transient analysis. Meanwhile, the time step is the period in which the simulator finds an initial solution point during the simulation to project the next solution point in the simulation process. In addition to the end time of the simulation process, the simulator also outputs the execution time. Four RTL gate-level netlist files for the entire design are provided with a simulator during the compilation and simulation process. Three netlist files are provided: one for each NI and the other one for the monitoring and control unit. The simulation results in waveforms are shown in Figures 9 and 10.

4.1. Simulation Results. Here, we provide the simulation results in waveforms using the Synopsys integrated design environment and tools [40]. Figures 9 and 10 show the transient analysis results in the time domain of the variation in sensor voltages over time for the different parts of the monitoring and control unit.

Figures 9(a)–9(c) show the transient time domain analysis for the three on-chip sensor NI designs. Figure 9(a) shows the simulation results of the voltage sensor NI and the voltage signal variation at different time instances. We observed that the sensor signals are alternating with an amplitude of 0–5 V corresponding to the reference voltage of 5 V. The four sensor signals are combined (multiplexed and amplified) to form an aggregated signal. As expected, the aggregated sensor signals are measured to be 5.0 V, which shows a high accuracy in the measurement of the combined sensor signals with a very negligible error in the sensor measurement.

Figure 9(b) shows the variation in the sensor voltage at different time instances for the thermal sensor NI. As observed, no difference exists between the reference temperature source (voltage) and the measured voltage of the four aggregated sensors. As expected, the reference voltage and the measured sensor voltages are the same (i.e., 5.0 V), which shows accurate precision in the sensor measurement of the NI and the efficiency of the intelligent thermal sensor network design.

Figures 10(a) and 10(b) show the simulation results for the variation in the sensor signals (voltage) over time for the main monitoring and control unit. Figure 10(a) shows the measured analog signals at different time instances of the various components of the monitor controller unit compared with the input sensor measured signals from the sensor NIs. The figure shows that the bang-bang_1 output signal oscillates between 0 and 5 V (maximum), which indicates the same measured input sensor signals of 5 V.

The transceiver_txd signal remains stable and is measured to be 5 V at different time intervals. This indicates a continuous conversion of signals and a high accuracy in the sensor signal conversion from analog to digital by the ADC. The output signals for both the high-speed transceiver and the CAN controller engine are controlled and regulated by the bang-bang_2 and bang-bang_3 controllers. We can observe that the controlled voltage measurement of the bang-bang_2 controller is 5 V measured at different time intervals. The bang-bang_3 controller voltage connected to the system monitor unit to control the monitor input and output is also measured to be 5 V, which shows that the input sensor signal from the NI is well monitored and performs control without losing sensor precision and accuracy.

Figure 10(b) shows the digital signal output of the converted sensor signals by the monitoring and control unit.

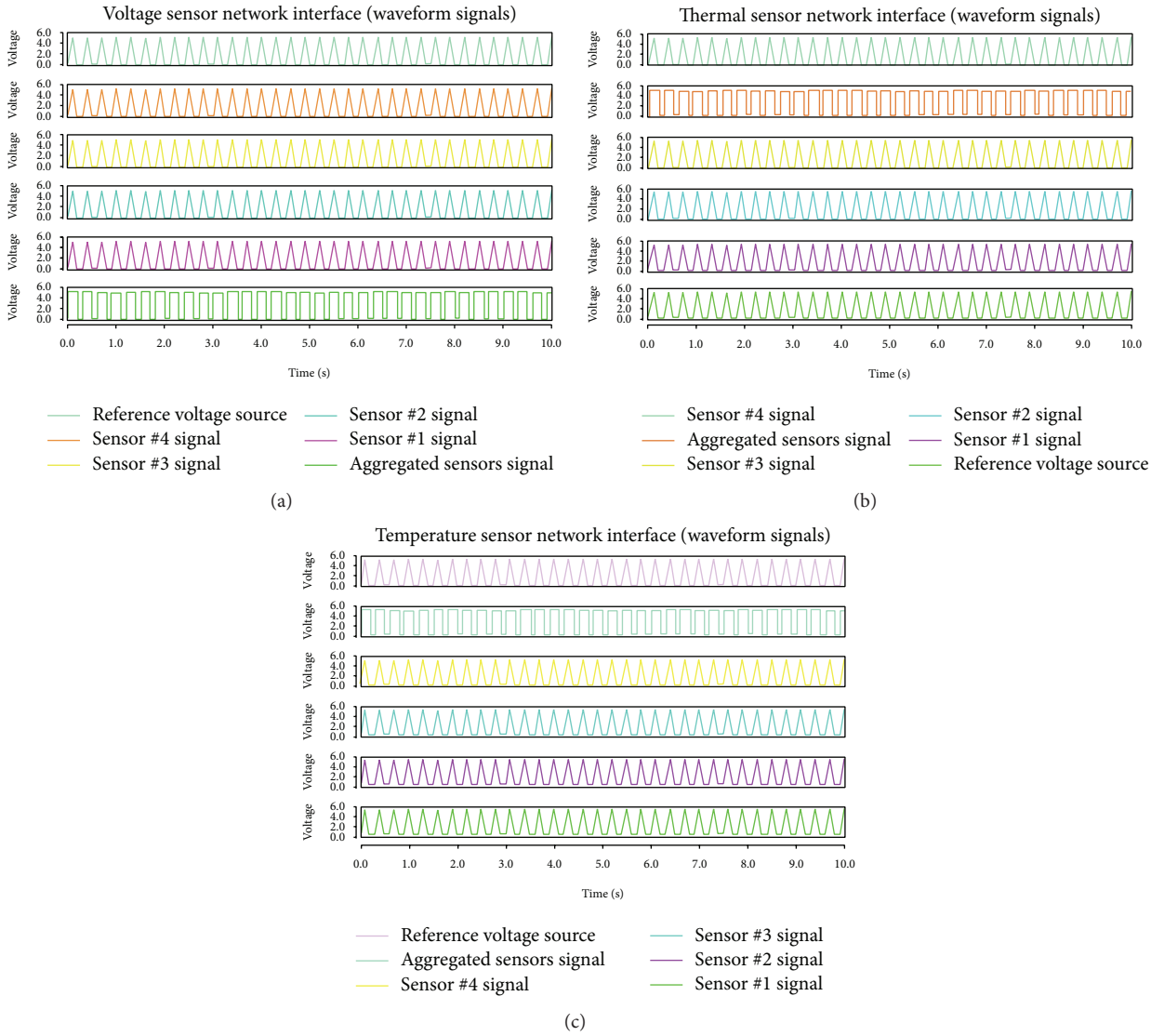


FIGURE 9: Simulation waveforms for transient analysis of the variation in the reference voltage and the measured voltages over time for the (a) voltage, (b) thermal, and (c) temperature sensors.

The different digital bits of the converted sensor analog signals are output through the digital bits (0–7) of the ADC output. The transceiver signal was observed to be stable and high at different time instances.

In conclusion, the simulation results verified the functionality of the various design components of the sensor NIs and the monitoring and control unit. The actual application of this proposed approach is implemented using FPGA as a case study. The details of the case study and FPGA implementation are presented in Section 5.

5. Experimental Setup

This section explains the experimental procedure used to design, implement, and verify the proposed on-chip sensor network monitoring and control system using FPGA. To validate the proposed approach through a real-life scenario, we implemented a case study for on-chip sensor network monitoring and control using FPGA. We used the Xilinx Vivado 2013.3 integrated design environment and the Zynq 7000 XC7z020 CLG484-1 AP SoC FPGA device for synthesis, placement, routing, and implementation of the proposed system. Figure 11 shows the on-chip sensor network monitoring and control unit implemented using Xilinx ADC (XADC) and SYSMON IP cores [43, 44]. The XADC and SYSMON IP cores provide an interface for designing on-chip sensors and user-defined external sensors to the main monitoring and control unit. The system provides capabilities for self-awareness and adaptation to dynamically tune the different parameter values at runtime to measure ambient parameters such as temperature, voltage, and thermal energy variations. It is based on a highly precise analog measurement system consisting of a 12-bit ADC and other circuit elements.

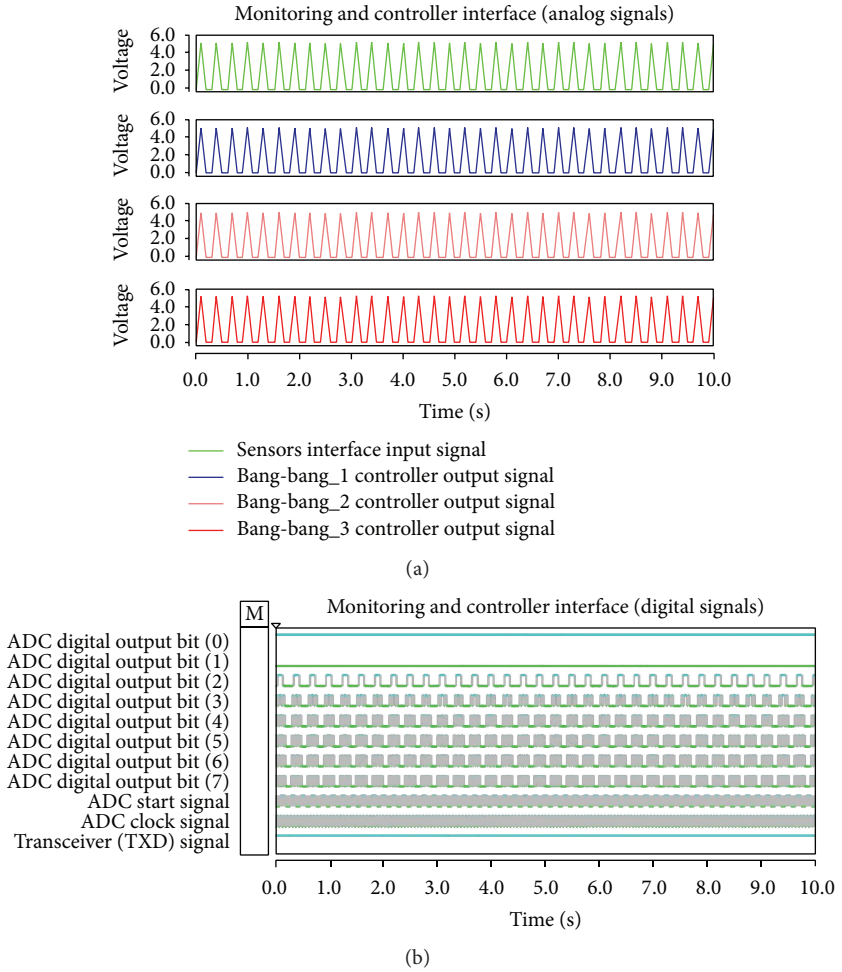


Figure 10: Simulation waveforms of the transient analysis of the variation in the reference and measured voltages over time for the (a) analog and (b) digital signals.

The ADC has a least significant bit with a size equivalent to 1 mV. Several input and output pins are provided to achieve high performance and accurate measurements from the on-chip sensors and the remote sensors connected to the system interface [26–35, 42–46].

Six different sections are present in the monitoring and control unit, as shown in Figure 11. The dynamic reconfiguration ports (DRPs) allow the system to be reconfigured according to certain values of input and output signals to the monitoring and control unit. The control section pins provide three signals that enable the conversion of the sensor analog signals to digital signals. These are the reset, conversion, and clock signals. The input to these signals allows the sensor data to be converted to digital signals. In addition to the on-chip sensors, remote sensors can also be interfaced to the main controller unit via the auxiliary input; up to 16 remote analog inputs can be connected to the system.

The DRP, advanced extended interface (AXI4), AXI4-stream, and I2C interfaces provide a dedicated communication mechanism for the system to allow interfacing with remote devices (external sensors). The monitor and controller unit provides the necessary monitoring and control of the variations in temperature, heat, and voltage of the on-chip sensors. The system is also configured to generate an alarm when a threshold value is exceeded. Figure 12 shows the RTL netlist schematic after synthesis, placement, and routing of the implemented monitoring and control system on Zynq 7000 XC7z020 CLG484-1 FPGA board using the Xilinx Vivado 13.3 integrated design environment.

The architecture shown in Figure 11 consists of the XADC and SYSMON IP interface that are present in all Xilinx 7 series FPGA devices, including the Zynq 7000 families [43, 44]. The central processing unit is based on ARM Cortex-A9 processing system and is used to monitor and control the on-chip sensor variation in the measured temperature, voltage, and power supply. Similarly, the remote sensors connected to the main unit can also be monitored and controlled. Furthermore, the system will set an alarm when over temperature, over voltage, and under voltage, and so forth, occur. This feature will ensure smooth running and proper functioning of the whole system.

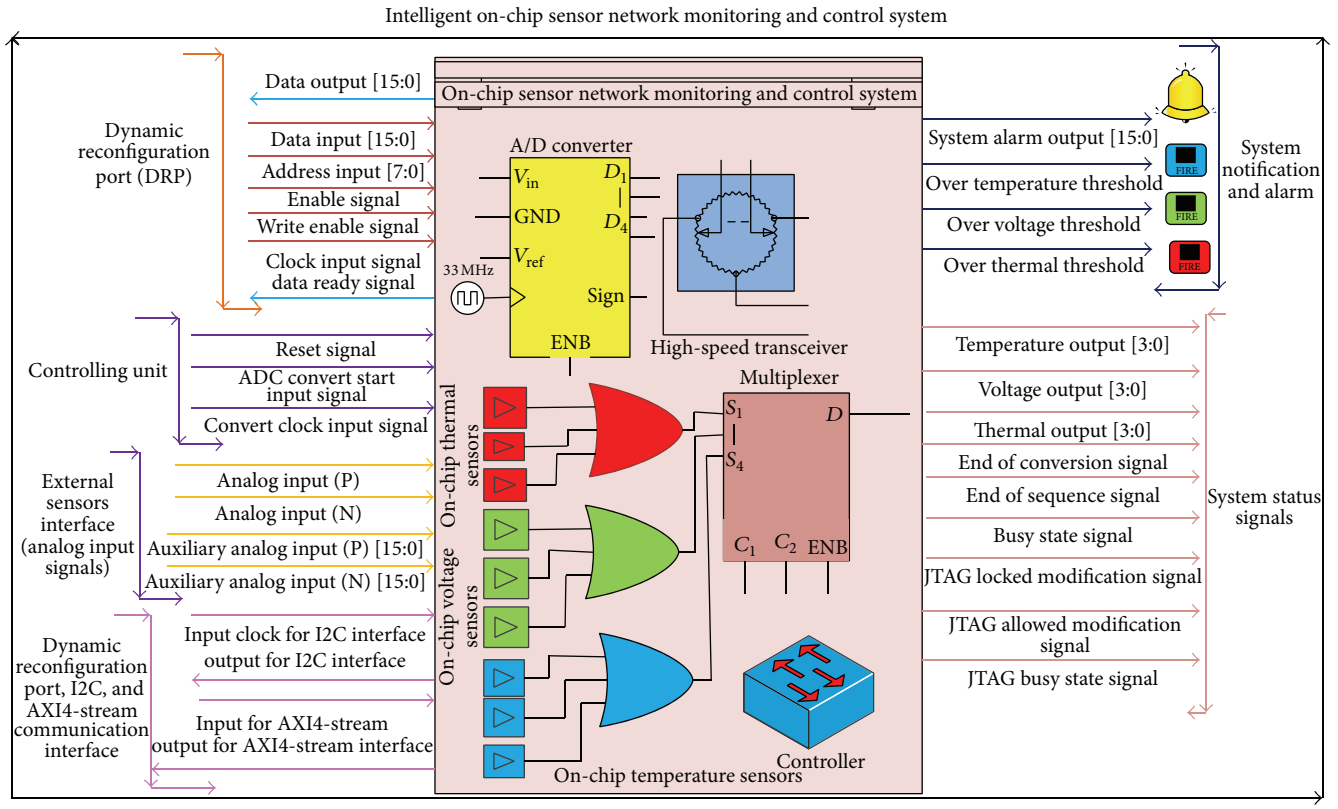
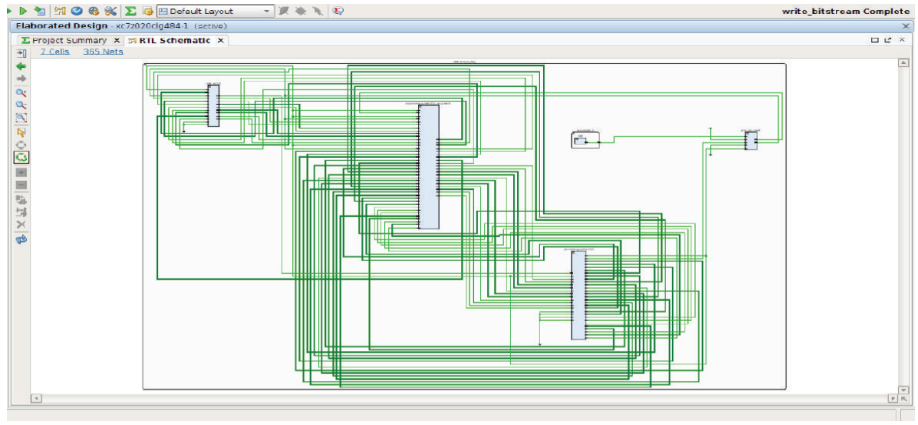


Figure 11: Block design of on-chip sensor network monitoring and control system using XADC and SYSMON IP cores.

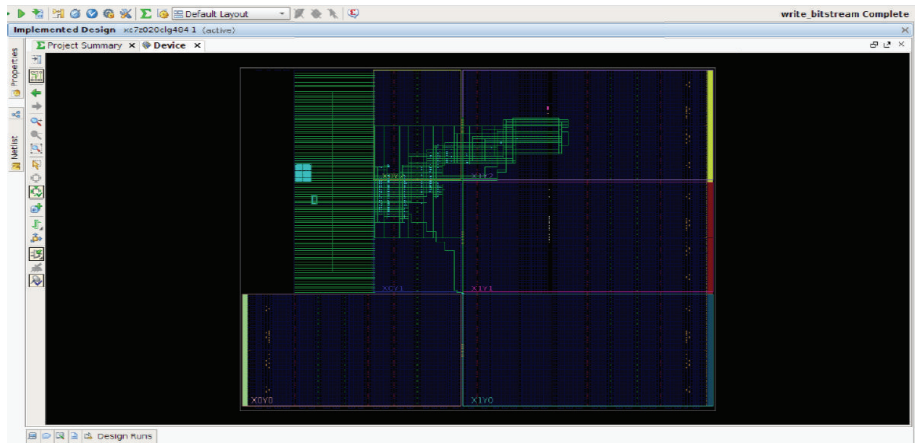
5.1. Case Study. This subsection provides the detailed explanation of the proposed case study using the Zynq 7000 FPGA board [43–47]. This case study uses a central processing unit (ARM Cortex-A9 core) to monitor and control the variation in temperature and voltage of the on-chip sensor and remote sensors connected to the FPGA fabric. The Xilinx Zynq 7000 is flexible and efficient architecture based on all programmable system-on-chip architecture. The device consists of a dual-core ARM Cortex-A9 MP Core processing system and enhanced Xilinx programmable logic in one integrated FPGA fabric. In the proposed design, the processing system is the main monitoring unit that provides basic functionalities with the Cortex-A9 MP Core central processing unit. The XADC acts as the control unit, which works together with the processing system. The other components installed in the processing system include the on-chip memory, on-chip sensors, input and output devices, auxiliary memory, and interface to the remote-sensing devices.

The monitoring and control system provides communication using different interfaces such as I2C, DRP, and JTAG. The control system uses another form of communication interface, which is the industrial input and output (IIO) framework-based Linux driver [44–46]. This driver works as a device driver for an application that uses a control system and the AXI and DRP communication interfaces. An interesting functionality of this driver is its ability to configure the control system to be used in different functionalities, for example, receiving data from the control system and providing information for different parameter states and configurations in the user space provided in the intended application. In this case study, we demonstrate how the IIO-based Linux driver is used for on-chip sensor network monitoring and control application. We show how the control system can be used to provide a hardware design in the programming logic, which creates a dedicated communication between the control and monitoring system using the DRP and AXI port communication interfaces. To make the application more user friendly, a web server-based design is provided for the user to interact with the monitoring and control system. The results of the dynamic monitoring and control system are displayed on a webpage for the users to see and change the configuration parameters to observe the variations in the measured on-chip temperature and voltage parameters, as shown in Figures 13 and 14.

The architecture of the hardware monitoring and control system as well as the connectivity between the control system in the programming logic and the processing system is shown in Figures 13 and 14. The DRP is used for reading and writing the sensor data into the control system DRP address register. The control system then translates the AXI4 transactions into the DRP address, which are used by the monitoring and control unit. An AXI interrupt controller is then instantiated, which translates the alarm output from the control system into various interrupt events for execution by the ARM core processing system. The control system interface to the monitoring system is set to control



(a)



(b)

Figure 12: Illustration of the on-chip sensor network monitoring and control system. (a) RTL schematics. (b) Routing resources of the Zynq 7000 AP SoC FPGA device.

the user-specified configuration executed in the context of the user application layer. Here, an instruction can be issued through the instruction interface to read and write the sensor values. For instance, a 32-bit instruction next to a no-operation instruction is written to first-in-first-out (FIFO) buffers. These instructions are transmitted to the control system using the communication interfaces, that is, DRP and AXI4. In this manner, any word that moves into the instruction FIFO buffer moves a similar word into the data FIFO buffer. This scenario also applies to the DRP. As data are moved out of the instruction FIFO buffer, the old values stored in the control unit dynamic reconfiguration register also move out. At the end of the specified time interval, the result of the current DRP read operation will be output at the control unit register for further processing by the monitoring and control system [3–14, 17–20, 43–47].

The web server or user application in this case study employs the IIO framework device driver, as mentioned earlier. This is a standard means of providing support for ADCs. It provides two basic functionalities: file system interface for communication with various devices (sensors) connected to the system and character driver interface to receive event information from the subsystems to the user application space in the monitoring and control system [43–47]. The basic sections of the on-chip sensor network monitoring and control user application are briefly explained as follows:

(i) On-chip sensor network monitoring and control: this subsystem provides a means of communication with the IIO system Linux device driver for functionalities like retrieving data from sensors, hardware configuration, synchronous/asynchronous event handling, and control mechanism.

(ii) Web server subsystem: this subsystem of the monitoring and control unit addresses the connection request from remote web users. The users can acquire sensor data and other information through the web server interface. Sensory information and event notification and updates are provided to the server through the monitoring and control unit.

(iii) Web interface subsystem: this subsystem provides the basic connection and interface for the web server and the monitoring and control system. The interface provides the functionality to set the threshold values

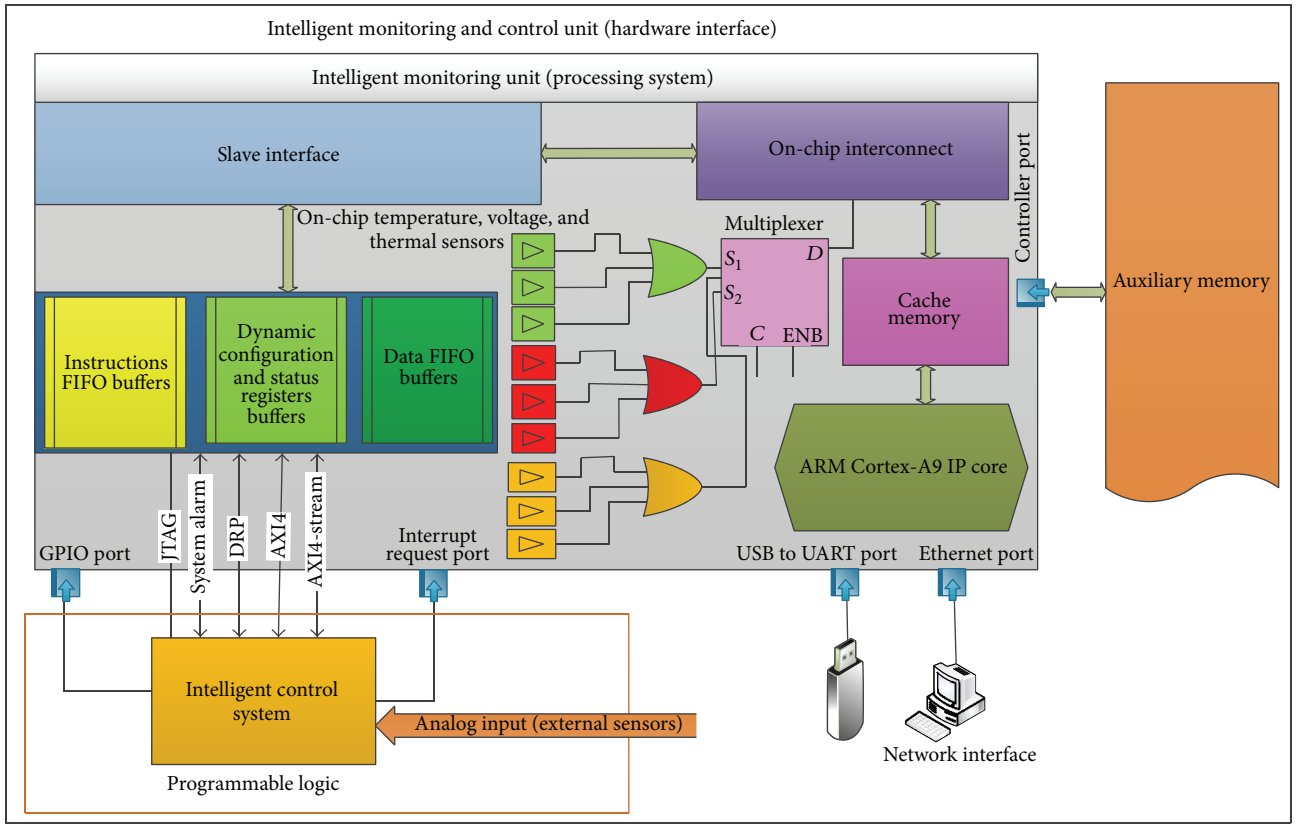


Figure 13: On-chip sensor network monitoring and control system (hardware interface).

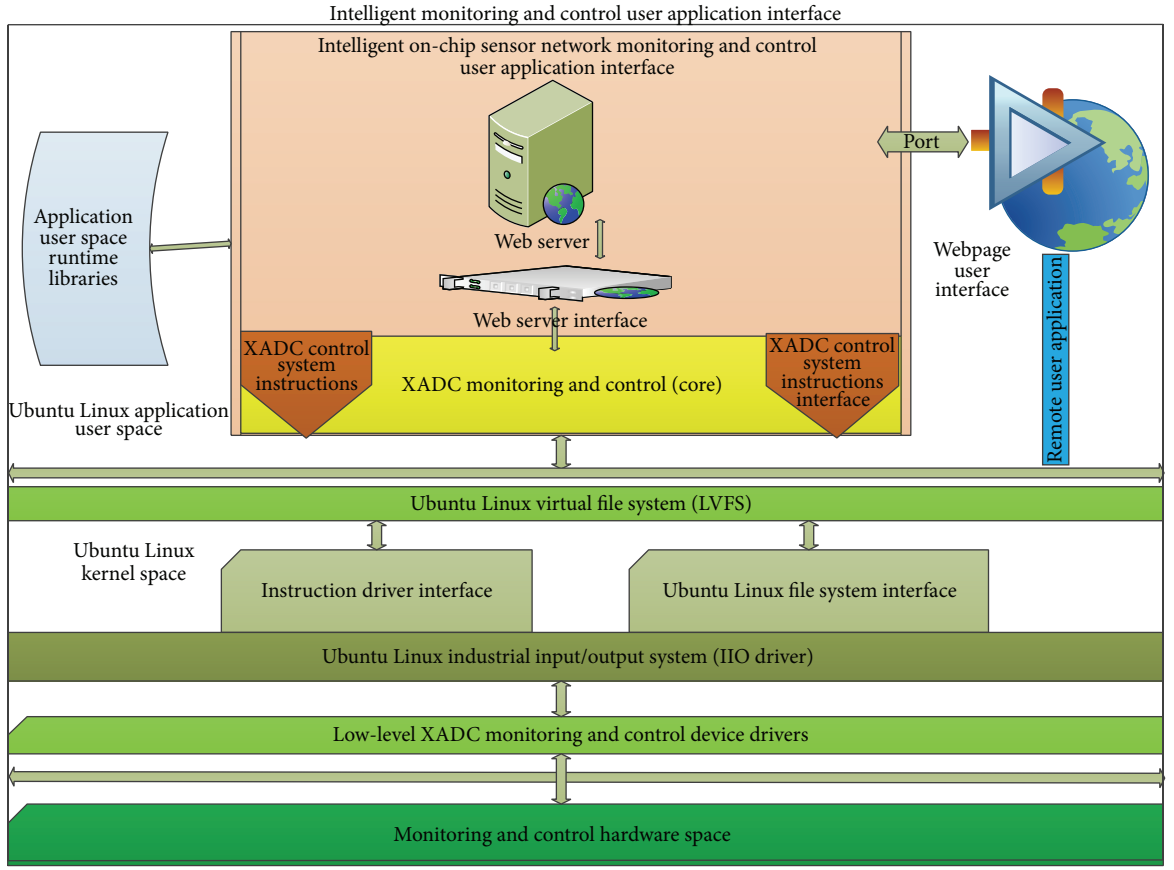
of the different alarms and communicates these values to and from the web server interface for efficient monitoring and control.

(iv) Web client subsystem: this subsystem provides functionality to users to tune the different parameter values of the temperature and voltage thresholds. The web client operates in a web browser and provides communication to the web server using a communication port. A graphical user interface is provided to obtain the sensed data from the sensors and the possibility of tuning the threshold values and dynamic reconfiguration refresh time. Figure 12 shows the implemented monitoring and control user application and web interfaces.

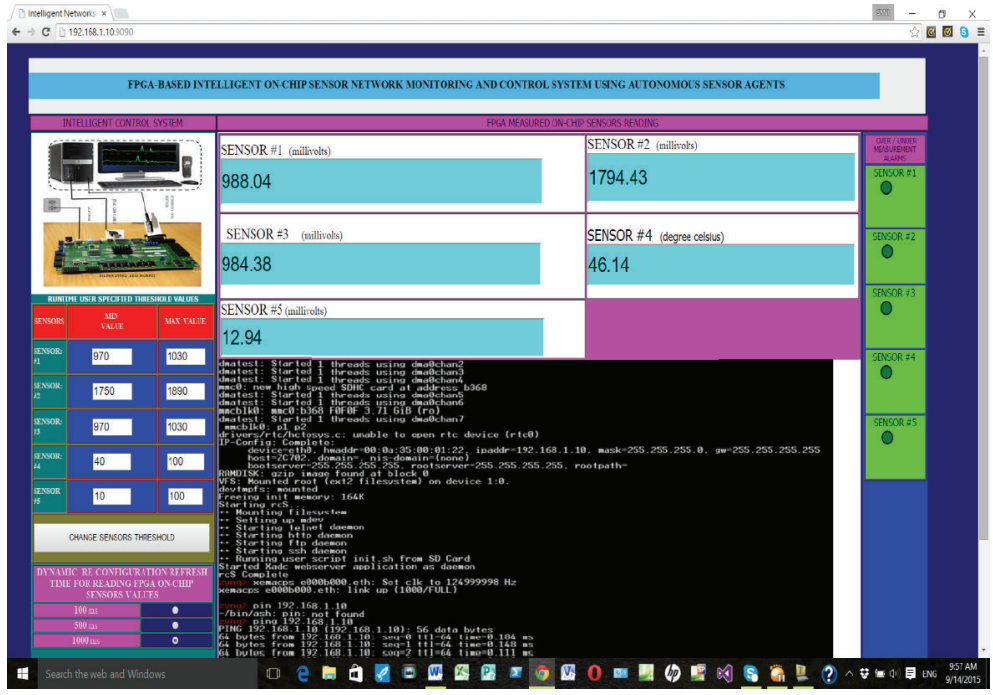
5.2. FPGA Resource Utilization. This subsection provides the results of the FPGA logic resource utilization of the postsynthesis and implementation (i.e., placement and routing) on the Xilinx Zynq evaluation kit ZC702 XC7Z020 CLG484-1, Zynq 7000 AP SoC FPGA device. Tables 1–4 list the resource usage at different design implementation stages. Tables 1 and 2 list the FPGA implementation results of the logic components for the on-chip sensor network monitoring and control system. We observe that the implemented design uses very small amounts of FPGA logic resources, that is, 2823 out of 482901, which is equivalent to 0.585%, compared with the results presented in [1–4, 23, 34, 35] as provided in Table 5. This result indicates low-cost utilization of the logic resources when implemented in a medium-sized seven-series FPGA device. This result validates our stated objective of achieving a low-cost intelligent monitoring and control system. After the design synthesis, we optimize the design using the Xilinx Vivado tool to efficiently route the design into the target

Table 1: FPGA postsynthesis (device placement) resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1 Zynq 7000 AP SoC FPGA device.

Resource type	Used	Available	Percentage utilization
Slice LUTs	641	53200	1.20%
LUTs as logic	571	53200	1.07%
LUTs as memory	70	17400	0.40%
LUT as shift register	70	—	—
Slice registers	770	106400	0.72%
Registers as flip-flops	770	106400	0.72%
Registers as latch	0	106400	0.0%
F7 Muxes	8	26600	0.03%
F8 Muxes	0	13300	0.00%
XADC	1	1	100%
Total	2901	482901	0.601%



(a)



(b)

Figure 14: On-chip sensor network monitoring and control system. (a) User application interface design. (b) Webpage user interface.

Table 2: FPGA postimplementation (device routing) resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1 Zynq 7000 AP SoC FPGA device.

Resource type	Used	Available	Percentage utilization (%)
Slice LUTs	607	53200	1.14
LUTs as logic	545	53200	1.02
LUTs as memory	62	17400	0.35
LUT as shift register	62	—	—
Slice registers	769	106400	0.72
Registers as flip-flops	769	106400	0.72
Registers as latch	0	106400	0.00
F7 Muxes	8	26600	0.03
F8 Muxes	0	13300	0.00
XADC	1	1	100%
Total	2823	482901	0.585%

Table 3: FPGA low-level primitive resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1 Zynq 7000 AP SoC FPGA device.

Name	FDRE	LUT3	LUT6	LUT5	BIBUF	FDSE	LUT4	LUT2	SRLC32E	SRL16E	CARRY4	MUXF7	LUT1	XADC	PS7	BUFG
Used	677	216	206	131	130	92	77	65	47	23	18	8	8	1	1	1

Table 4: FPGA on-chip sensor component after placement and routing power resource utilization implemented on Xilinx Zynq evaluation kit ZC702, XC7Z020 CLG484-1 Zynq 7000 AP SoC FPGA device.

Resource type	Used	Available	Power (W)	Percentage utilization (%)
Clocks	1	—	0.001	—
Slice logic	1757	—	0.001	—
LUTs as logic	545	53200	0.001	1.02
Register	769	106400	0.001	0.72
CARRY4	18	13300	0.001	0.14
F7/F8 Muxes	8	53200	0.001	0.02
LUT as shift register	62	17400	0.001	0.36
Others	189	—	0.001	—
Signals	1225	—	0.001	—
XADC	1	1	0.001	—
PS7	1	1	0.001	—
Static power	—	—	0.118	—
	Total		0.129 watts	2.26%

architecture. In Table 2, we can observe that because of the optimization done to route the design, a slight reduction in the amount of logic consumption occurs. Table 3 lists the low-level primitives used by the implemented design and their utilization numbers.

Table 4 lists the power distribution and utilization of the various on-chip sensor components after placement and routing. This distribution consisted of both dynamic and static power consumptions. We can see from the table that all on-chip sensor components show very low power consumption. We achieved 0.118 W (118 mW) of static power and 0.011 W (11 mW) of dynamic power consumption, a total of 0.129 W (129 mW), which represents a 2.26% of both dynamic and static power consumption, for all on-chip sensor components as compared with the results presented in [2, 3]. This result also agrees with our stated objective of achieving low power consumption in medium-sized seven-series FPGA device.

5.3. FPGA Resource Utilization Comparison with Previous Works. Table 5 shows the FPGA utilization results comparison with previous works [1–4, 23, 34, 35]. The comparison with the state of the art seems to be difficult due to the fact that there are a lot of differences in terms of the FPGA implementation architectures, sensors, and sensor networks and some of the proposed approaches did not provide FPGA power utilization result. Furthermore, the types of logic resources provided by the previous approaches are different

Table 5: FPGA resource utilization comparison with previous works.

Chu et al. [1]					
Flip-flops (%)	4-input LUTs (%)	Slices (%)	Block RAM (%)	Power utilization	Implementation architecture
Logic control method					
19%	35%	41%	25%	Not provided	XILINX SPARTAN-3 FPGA device (3XC3S400).
Lookup table method					
12%	29%	31%	37%	Not provided	XILINX SPARTAN-3 FPGA device (3XC3S400).
Perera et al. [2]					
Slice registers (%)	4-input LUTs (%)	Occupied slices (%)	Bonded IOBs (%)	Power utilization	Implementation architecture
50%	52%	97%	8%	Static power = 460.9 mW Dynamic power = 491.4 mW	XILINX SPARTAN-3E FPGA device (XC3S250E).
Echanobe et al. [3]					
FFs (%)	LUTs (%)	BRAM (%)	DSPs (%)	Power utilization	Implementation architecture
41%	52%	50%	83%	Static power = 3.071 mW Dynamic power = 0.649 mW	XILINX VIRTEX-5 FPGA device (XCVSX50T).
Osuna et al. [4]					
LUT + LATCH pairs (#)	Loop count (#)	Actual delay (ns)	LUT + D-FF pairs (#)	Power utilization	Implementation architecture
80	1024	180,000	110	Not provided	XILINX SPARTAN-3E FPGA device (XC3S100E).
Kornaros and Pnevmatikatos [23]					
Configuration	Slices (#)	RAMB16s (#)	Frequency (MHz)	Power utilization	Implementation architecture
CAM Plugin (16 × 192) Control Monitor	1364	18	211	Not provided	XILINX VIRTEX-4 FPGA device.
Manager (depth 32, trace, 16 bits)	3872	1	216		
Monitor (32-bit) with FSL Interface	432	3	418		
Magdaleno et al. [34]					
Implemented prototype uses 2065 LEs of a total of 7% logic utilization.				Not provided	CYCLON II ALTERA FPGA device (EP2C35).

Table 5: Continued.

Gomez Osuna et al. [35]							
Delay stages (#)	Loop counter (#)	Measured delay (ns)	Total area slices (#)	Delay area (%)		Power utilization	Implementation architecture
10	2,560	124.170	34	29%		Not provided	XILINX SPARTAN-3E FPGA device (XC3S100E).
20	1,280	117,500	43	47%			
30	853	116,340	50	60%			
4	4,096	92,370	14	29%		Not provided	XILINX VIRTEX-5 FPGA device (LX50T).
Proposed approach							
Slice LUTs (%)	LUT as logic (%)	LUT as memory (%)	Slice registers (%)	Registers as flip-flops (%)	F7 Muxes (%)	Power utilization	Implementation architecture
1.14%	1.02%	0.35%	0.72%	0.72%	0.03%	Static power = 0.118 W (118 mW) Dynamic power = 0.011 W (11 mW)	XILINX ZYNQ FPGA-7 device ZC702 (XC7Z020 CLG484-1).
Total logic utilization (implemented design) = 0.585%							

%: percentage of logic resource utilization.
#: number of logic resource.
Not provided: the authors did not provide the power utilization result.

from one implementation to another. However, the utilization results shown in Table 5 help to acknowledge the efficiency and flexibility of the proposed approach compared with the state of the art.

6. Experimental Results and Discussion

This section provides the results of the experiments to monitor and control the FPGA on-chip sensor readings. The essential metric for evaluating the FPGA on-chip sensor readings is its accuracy; that is, how accurate does the FPGA report the sensed on-chip voltage and temperature readings? Although the FPGA on-chip sensors are specified in terms of accuracy based on the manufacturer datasheet, the real significance is the on-chip sensor error. For instance, the temperature sensor specification for accuracy is ±4% full scale (FS) within the range −40°C to +100°C, which means that the on-chip sensor error will not exceed ±4% of its FS within its calibrated range. Therefore, the evaluation of the FPGA on-chip sensor accuracy represents the process of determining its maximum error [43, 44, 50].

We conducted several experiments using five different sensors, namely, Sensors #1–#5. Sensor #4 is a temperature sensor that measured the FPGA on-chip die temperature in degree Celsius and operates within a set threshold limit. The remaining four sensors are power sensors, which measure the on-chip voltage variations in millivolts within monitored and control threshold value. Tables 6–21 list the FPGA-measured on-chip sensor readings. Each table represents the four experiments we conducted, taking 10 readings in each experiment for a total of 40 measured readings from the FPGA on-chip sensors. The readings were averaged to remove the gross and systematic errors from the observed readings. The FS percentage error or accuracy was calculated for each observation and averaged across the 40 readings. We varied the dynamic reconfiguration refresh time to query the FPGA on-chip sensor readings from the webpage at 100, 500, and 1000 ms to determine the effect of the reconfiguration time with respect to the sensor accuracy. The details of the experimental results are presented in the next subsection.

6.1. Effect of Dynamic Reconfiguration Refresh Time on an FPGA-Measured On-Chip Sensor Reading Accuracy. Tables 6–10 list the FPGA-measured on-chip sensor readings with the dynamic reconfiguration refresh time set at 1000 ms. The range of operation and the ideal output of each sensor are specified and listed in the tables. The averaged FS percentage errors of the five sensors are as follows: Sensor #1 = −0.7% FS, Sensor #2 = −0.1% FS, Sensor #3 = 0.8% FS, Sensor #4 = 0.1% FS, and Sensor #5 = 0.4% FS. This result indicates that the accuracy of the measured sensor values is influenced by the reconfiguration refresh time, which shows an accuracy range between −0.7% FS and +0.8% FS.

Tables 11–15 list the FPGA-measured on-chip sensor readings with the dynamic reconfiguration refresh time set at 500 ms. Here, the averaged FS percentage errors for the five sensors are as follows: Sensor #1 = −0.6% FS, Sensor #2 = −0.6% FS, Sensor #3 = +0.2% FS, Sensor #4 = +1.1% FS, and Sensor #5 = 0.3% FS. The averaged FS accuracy lies between −0.6% FS and +1.1% FS, which illustrates a better accuracy compared with reconfiguration refresh time of 100 ms.

Table 6: FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 1000 ms.

S/number	Expt. 1 Sensor #1 measured reading (mV)	Expt. 2 Sensor #1 measured reading (mV)	Expt. 3 Sensor #1 measured reading (mV)	Expt. 4 Sensor #1 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	988.77	984.38	990.23	986.57	987.49	1030.00	970.00	987	0.49	0.8
(2)	982.91	985.84	988.77	986.57	986.02	1030.00	970.00	987	−0.98	−1.6
(3)	987.30	988.04	988.04	986.30	987.42	1030.00	970.00	987	0.42	0.7
(4)	985.84	985.11	986.57	987.30	986.21	1030.00	970.00	987	−0.80	−1.3
(5)	988.04	988.77	986.57	988.04	987.86	1030.00	970.00	987	0.86	1.4
(6)	987.30	985.11	986.57	990.23	987.30	1030.00	970.00	987	0.30	0.5
(7)	985.84	985.84	987.30	988.77	986.94	1030.00	970.00	987	−0.06	−0.1
(8)	985.84	983.64	985.84	986.57	985.47	1030.00	970.00	987	−1.53	−2.5
(9)	985.84	984.38	984.38	986.64	985.31	1030.00	970.00	987	−1.69	−2.8
(10)	985.11	985.84	985.84	985.84	985.66	1030.00	970.00	987	−1.34	−2.2
				Average					*−0.43*	*−0.7*

Table 7: FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 1000 ms.

S/number	Expt. 1 Sensor #2 measured reading (mV)	Expt. 2 Sensor #2 measured reading (mV)	Expt. 3 Sensor #2 measured reading (mV)	Expt. 4 Sensor #2 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	1788.57	1791.50	1795.17	1792.24	1791.87	1890.00	1750.00	1793	−1.13	−0.8
(2)	1789.31	1794.43	1794.43	1792.24	1792.60	1890.00	1750.00	1793	−0.40	−0.3
(3)	1795.17	1791.50	1796.63	1792.24	1793.89	1890.00	1750.00	1793	0.88	0.6
(4)	1792.24	1792.24	1797.36	1788.57	1792.60	1890.00	1750.00	1793	−0.40	−0.3
(5)	1788.57	1792.24	1795.90	1791.50	1792.05	1890.00	1750.00	1793	−0.95	−0.7
(6)	1792.24	1795.90	1792.97	1792.97	1793.52	1890.00	1750.00	1793	0.52	0.4
(7)	1790.04	1792.97	1801.03	1792.97	1794.25	1890.00	1750.00	1793	1.25	0.9
(8)	1785.64	1795.17	1793.70	1793.70	1792.05	1890.00	1750.00	1793	−0.95	−0.7
(9)	1793.70	1792.24	1791.50	1790.04	1791.87	1890.00	1750.00	1793	−1.13	−0.8
(10)	1794.43	1792.97	1795.90	1793.70	1794.25	1890.00	1750.00	1793	1.25	0.9
				Average					*−0.10*	*−0.1*

Table 8: FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 1000 ms.

S/number	Expt. 1 Sensor #3 measured reading (mV)	Expt. 2 Sensor #3 measured reading (mV)	Expt. 3 Sensor #3 measured reading (mV)	Expt. 4 Sensor #3 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	986.57	986.57	987.30	987.30	986.94	1030.00	970.00	986	0.93	1.6
(2)	985.11	987.30	985.11	985.84	985.84	1030.00	970.00	986	−0.16	−0.3
(3)	983.64	987.30	986.57	989.50	986.75	1030.00	970.00	986	0.75	1.3
(4)	986.57	987.30	988.04	985.84	986.94	1030.00	970.00	986	0.94	1.6
(5)	985.11	984.38	988.04	986.57	986.03	1030.00	970.00	986	0.02	0.04
(6)	985.84	985.84	987.30	988.77	986.94	1030.00	970.00	986	0.94	1.6
(7)	989.50	986.57	986.57	986.57	987.30	1030.00	970.00	986	1.30	2.2
(8)	983.64	988.77	985.84	986.57	986.21	1030.00	970.00	986	0.21	0.3
(9)	984.38	987.30	985.11	987.30	986.02	1030.00	970.00	986	0.02	0.04
(10)	985.84	984.38	988.04	985.11	985.84	1030.00	970.00	986	−0.16	−0.3
				Average					*0.48*	*0.8*

Table 9: FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 1000 ms.

S/number	Expt. 1 Sensor #4 measured reading (°C)	Expt. 2 Sensor #4 measured reading (°C)	Expt. 3 Sensor #4 measured reading (°C)	Expt. 4 Sensor #4 measured reading (°C)	Average Measured reading (°C)	Range High (°C)	Range Low (°C)	Ideal Output (°C)	Error Values	Full scale % error
(1)	55.12	47.49	47.62	47.25	49.37	100.00	40.00	50.00	−0.63	−1.1
(2)	54.26	48.36	47.49	48.23	49.59	100.00	40.00	50.00	−0.41	−0.7
(3)	55.49	48.23	47.74	47.74	49.80	100.00	40.00	50.00	−0.20	−0.3
(4)	55.12	48.85	47.74	49.09	50.20	100.00	40.00	50.00	0.20	0.3
(5)	55.00	49.09	48.23	48.23	50.14	100.00	40.00	50.00	0.14	0.2
(6)	55.37	49.22	47.74	48.48	50.20	100.00	40.00	50.00	0.20	0.3
(7)	55.12	49.09	47.99	48.48	50.17	100.00	40.00	50.00	0.17	0.3
(8)	54.88	50.20	47.86	48.48	50.36	100.00	40.00	50.00	0.35	0.6
(9)	55.49	50.08	47.99	48.60	50.54	100.00	40.00	50.00	0.54	0.9
(10)	55.25	49.83	48.23	48.72	50.51	100.00	40.00	50.00	0.51	0.8
				Average					*0.09*	*0.1*

Table 10: FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 1000 ms.

S/number	Expt. 1 Sensor #5 measured reading (mV)	Expt. 2 Sensor #5 measured reading (mV)	Expt. 3 Sensor #5 measured reading (mV)	Expt. 4 Sensor #5 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
(2)	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
(3)	13.43	12.94	13.18	13.43	13.25	100.00	10.00	13.00	0.24	0.3
(4)	13.43	13.43	13.18	13.43	13.37	100.00	10.00	13.00	0.37	0.4
(5)	13.67	13.18	12.70	13.43	13.25	100.00	10.00	13.00	0.24	0.3
(6)	13.67	13.43	12.94	12.94	13.25	100.00	10.00	13.00	0.24	0.3
(7)	13.67	13.43	13.18	13.67	13.49	100.00	10.00	13.00	0.49	0.5
(8)	13.43	13.18	13.43	13.18	13.31	100.00	10.00	13.00	0.31	0.3
(9)	13.43	13.43	13.18	13.18	13.31	100.00	10.00	13.00	0.31	0.3
(10)	13.92	13.43	12.70	13.67	13.43	100.00	10.00	13.00	0.43	0.5
				Average					*0.34*	*0.4*

Table 11: FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 500 ms.

S/number	Expt. 1 Sensor #1 measured reading (mV)	Expt. 2 Sensor #1 measured reading (mV)	Expt. 3 Sensor #1 measured reading (mV)	Expt. 4 Sensor #1 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	986.57	983.64	988.77	987.30	986.57	1030.00	970.00	987	−0.43	−0.7
(2)	985.11	985.84	986.57	987.30	986.21	1030.00	970.00	987	−0.80	−1.3
(3)	985.84	987.30	988.77	986.57	987.12	1030.00	970.00	987	0.12	0.2
(4)	985.11	985.84	985.11	988.77	986.21	1030.00	970.00	987	−0.79	−1.3
(5)	985.84	986.57	985.84	986.57	986.21	1030.00	970.00	987	−0.79	−1.3
(6)	986.57	986.57	988.77	987.30	987.30	1030.00	970.00	987	0.30	0.5
(7)	988.77	988.04	988.04	985.11	987.49	1030.00	970.00	987	0.49	0.8
(8)	987.30	985.11	985.11	988.04	986.39	1030.00	970.00	987	−0.61	−1.0
(9)	985.84	985.11	988.04	987.30	986.57	1030.00	970.00	987	−0.43	−0.7
(10)	988.04	985.84	984.33	988.04	986.56	1030.00	970.00	987	−0.44	−0.7
				Average					*−0.34*	*−0.6*

Table 12: FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 500 ms.

S/number	Expt. 1 Sensor #2 measured reading (mV)	Expt. 2 Sensor #2 measured reading (mV)	Expt. 3 Sensor #2 measured reading (mV)	Expt. 4 Sensor #2 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	1790.04	1792.97	1797.36	1792.97	1793.34	1890.00	1750.00	1793	0.34	0.2
(2)	1792.97	1792.97	1793.7	1794.43	1793.52	1890.00	1750.00	1793	0.52	0.4
(3)	1793.70	1792.24	1790.77	1792.97	1792.42	1890.00	1750.00	1793	−0.58	−0.4
(4)	1793.70	1795.90	1794.43	1795.90	1794.98	1890.00	1750.00	1793	1.98	1.4
(5)	1793.70	1796.63	1792.97	1795.17	1794.62	1890.00	1750.00	1793	1.62	1.2
(6)	1795.90	1792.24	1791.5	1795.90	1793.89	1890.00	1750.00	1793	0.89	0.6
(7)	1794.43	1792.97	1795.17	1790.77	1793.34	1890.00	1750.00	1793	0.34	0.2
(8)	1791.50	1792.97	1793.7	1792.97	1792.79	1890.00	1750.00	1793	−0.21	−0.2
(9)	1791.50	1792.24	1792.97	1795.17	1792.97	1890.00	1750.00	1793	−0.03	0.0
(10)	1792.97	1790.77	1792.24	1791.50	1791.87	1890.00	1750.00	1793	−1.13	−0.8
				Average					*−0.34*	*−0.6*

Table 13: FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 500 ms.

S/number	Expt. 1 Sensor #3 measured reading (mV)	Expt. 2 Sensor #3 measured reading (mV)	Expt. 3 Sensor #3 measured reading (mV)	Expt. 4 Sensor #3 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	986.57	987.30	986.57	983.64	986.02	1030.00	970.00	986	0.02	0.03
(2)	985.11	985.11	986.57	986.57	985.84	1030.00	970.00	986	−0.16	−0.3
(3)	984.38	987.30	983.64	985.11	985.11	1030.00	970.00	986	−0.89	−1.5
(4)	987.30	987.30	986.57	986.57	986.94	1030.00	970.00	986	0.94	1.6
(5)	985.11	986.57	985.84	987.30	986.21	1030.00	970.00	986	0.20	0.3
(6)	984.38	982.18	987.3	985.84	984.93	1030.00	970.00	986	−1.08	−1.8
(7)	986.57	984.38	988.77	985.11	986.21	1030.00	970.00	986	0.21	0.3
(8)	985.84	987.30	985.84	988.77	986.94	1030.00	970.00	986	0.94	1.6
(9)	989.50	988.04	986.57	987.30	987.85	1030.00	970.00	986	1.85	3.1
(10)	983.64	986.57	985.11	984.38	984.93	1030.00	970.00	986	−1.07	−1.8
				Average					*0.10*	*0.2*

Table 14: FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 500 ms.

S/number	Expt. 1 Sensor #4 measured reading (°C)	Expt. 2 Sensor #4 measured reading (°C)	Expt. 3 Sensor #4 measured reading (°C)	Expt. 4 Sensor #4 measured reading (°C)	Average Measured reading (°C)	Range High (°C)	Range Low (°C)	Ideal Output (°C)	Error Values	Full scale % error
(1)	54.75	50.20	48.23	48.72	50.48	100.00	40.00	50.00	0.48	0.8
(2)	55.37	50.32	48.48	48.36	50.63	100.00	40.00	50.00	0.63	1.1
(3)	55.49	49.83	48.36	48.97	50.66	100.00	40.00	50.00	0.66	1.1
(4)	55.98	50.32	48.11	48.67	50.77	100.00	40.00	50.00	0.77	1.3
(5)	55.86	49.71	48.36	48.48	50.60	100.00	40.00	50.00	0.60	1.0
(6)	55.61	50.57	47.86	48.72	50.69	100.00	40.00	50.00	0.69	1.2
(7)	55.61	50.20	48.29	48.60	50.68	100.00	40.00	50.00	0.67	1.1
(8)	55.37	50.20	47.86	48.60	50.51	100.00	40.00	50.00	0.51	0.8
(9)	55.61	50.08	47.62	48.85	50.54	100.00	40.00	50.00	0.54	0.9
(10)	56.48	50.57	47.99	49.22	51.07	100.00	40.00	50.00	1.07	1.8
				Average					*0.66*	*1.1*

Table 15: FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 500 ms.

S/number	Expt. 1 Sensor #5 measured reading (mV)	Expt. 2 Sensor #5 measured reading (mV)	Expt. 3 Sensor #5 measured reading (mV)	Expt. 4 Sensor #5 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	13.43	13.43	13.18	13.18	13.31	100.00	10.00	13.00	0.31	0.3
(2)	12.94	12.70	13.43	13.43	13.13	100.00	10.00	13.00	0.13	0.1
(3)	13.43	12.94	12.70	12.70	12.94	100.00	10.00	13.00	−0.06	−0.1
(4)	13.67	13.43	13.18	13.18	13.37	100.00	10.00	13.00	0.37	0.4
(5)	13.67	13.43	13.18	12.94	13.31	100.00	10.00	13.00	0.31	0.3
(6)	13.67	13.43	12.94	13.43	13.37	100.00	10.00	13.00	0.37	0.4
(7)	13.67	12.94	12.94	13.43	13.25	100.00	10.00	13.00	0.24	0.3
(8)	13.92	13.18	13.43	12.94	13.37	100.00	10.00	13.00	0.37	0.4
(9)	13.67	13.18	12.94	12.94	13.18	100.00	10.00	13.00	0.18	0.2
(10)	13.43	13.18	13.18	13.18	13.24	100.00	10.00	13.00	0.24	0.3
				Average					*0.24*	*0.3*

Table 16: FPGA-measured on-chip Sensor #1 reading with dynamic reconfiguration refresh time of 100 ms.

S/number	Expt. 1 Sensor #1 measured reading (mV)	Expt. 2 Sensor #1 measured reading (mV)	Expt. 3 Sensor #1 measured reading (mV)	Expt. 4 Sensor #1 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	985.11	985.11	990.23	986.57	986.76	1030.00	970.00	987	−0.25	−0.4
(2)	987.30	987.30	986.57	986.57	986.94	1030.00	970.00	987	−0.06	−0.1
(3)	986.57	986.57	985.11	988.77	986.76	1030.00	970.00	987	−0.25	−0.4
(4)	987.30	988.04	988.04	985.84	987.31	1030.00	970.00	987	0.31	0.5
(5)	986.57	986.57	986.57	985.11	986.21	1030.00	970.00	987	−0.79	−1.3
(6)	987.30	986.57	987.30	985.11	986.57	1030.00	970.00	987	−0.43	−0.7
(7)	986.57	987.30	986.57	986.57	986.75	1030.00	970.00	987	−0.25	−0.4
(8)	982.91	987.30	986.57	988.04	986.21	1030.00	970.00	987	−0.79	−1.3
(9)	986.57	987.30	989.50	985.11	987.12	1030.00	970.00	987	0.12	0.2
(10)	984.38	985.84	987.30	985.11	985.66	1030.00	970.00	987	−1.34	−2.2
				Average					*−0.37*	*−0.6*

Tables 16–20 show the FPGA-measured and controlled on-chip sensors readings with the dynamic reconfiguration refresh time set at 100 ms. The averaged percentage accuracy of Sensor #1 FS is −0.6% FS, Sensor #2 = −0.3% FS, Sensor #3 = −0.024% FS, Sensor #4 = 1.3% FS, and Sensor #5 = 0.4% FS. From Tables 16–20, we can observe that the FS percentage accuracy lies from −0.6% FS to +1.3% FS. This result shows a low accuracy at FS as compared with the dynamic reconfiguration refresh times of 500 and 1000 ms. Table 21 shows the comparison results of the sensor reading accuracy versus the dynamic reconfiguration refresh time. We can observe that the FS percentage error with dynamic refresh time of 100 ms is between −0.6% FS and +1.3% FS, that of 500 ms is between −0.6% FS and +1.1% FS, and that of 1000 ms is between −0.7% FS and +0.8% FS. This result shows that the lower FS percentage error range is almost the same (i.e., −0.6% FS and −0.7% FS) for all sensors irrespective of the dynamic refresh time. However, at higher FS percentage error rate, a significant difference in accuracy exists with different dynamic refresh time, that is, +0.8% FS, +1.1% FS, and +1.3% FS. Therefore, the dynamic reconfiguration refresh time of 1000 ms produces highly accurate FPGA-measured on-chip sensor readings.

In conclusion, the dynamic reconfiguration refresh time affects the accuracy of the FPGA-measured on-chip sensor readings. We determined that the dynamic reconfiguration

Table 17: FPGA-measured on-chip Sensor #2 reading with dynamic reconfiguration refresh time of 100 ms.

S/number	Expt. 1 Sensor #2 measured reading (mV)	Expt. 2 Sensor #2 measured reading (mV)	Expt. 3 Sensor #2 measured reading (mV)	Expt. 4 Sensor #2 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	1793.70	1791.50	1789.31	1794.43	1792.24	1890.00	1750.00	1794	−1.76	−1.3
(2)	1791.50	1796.63	1794.43	1792.97	1793.88	1890.00	1750.00	1794	−0.12	−0.1
(3)	1791.50	1791.50	1792.24	1792.97	1792.05	1890.00	1750.00	1794	−1.95	−1.4
(4)	1792.24	1795.17	1795.17	1795.90	1794.62	1890.00	1750.00	1794	0.62	0.4
(5)	1793.70	1793.70	1794.43	1795.17	1794.25	1890.00	1750.00	1794	0.25	0.2
(6)	1789.31	1795.90	1794.43	1795.17	1793.70	1890.00	1750.00	1794	−0.30	−0.2
(7)	1795.17	1790.04	1792.97	1792.97	1792.79	1890.00	1750.00	1794	−1.21	−0.9
(8)	1793.70	1793.70	1797.36	1792.97	1794.43	1890.00	1750.00	1794	0.43	0.3
(9)	1794.43	1792.97	1791.50	1794.43	1793.33	1890.00	1750.00	1794	−0.67	−0.5
(10)	1796.63	1795.90	1794.43	1792.24	1794.80	1890.00	1750.00	1794	0.80	0.6
				Average					*−0.39*	*−0.3*

Table 18: FPGA-measured on-chip Sensor #3 reading with dynamic reconfiguration refresh time of 100 ms.

S/number	Expt. 1 Sensor #3 measured reading (mV)	Expt. 2 Sensor #3 measured reading (mV)	Expt. 3 Sensor #3 measured reading (mV)	Expt. 4 Sensor #3 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	985.84	985.84	985.84	984.38	985.48	1030.00	970.00	986	−0.52	−0.9
(2)	985.11	986.57	985.84	987.30	986.21	1030.00	970.00	986	0.20	0.3
(3)	984.38	986.57	986.57	987.30	986.21	1030.00	970.00	986	0.20	0.3
(4)	986.57	984.38	986.57	986.57	986.02	1030.00	970.00	986	0.02	0.04
(5)	988.04	987.30	987.30	986.57	987.30	1030.00	970.00	986	1.30	2.2
(6)	985.84	983.64	982.18	986.57	984.56	1030.00	970.00	986	−1.44	−2.4
(7)	985.84	988.04	985.11	986.57	986.39	1030.00	970.00	986	0.39	0.7
(8)	986.57	986.57	987.30	987.30	986.94	1030.00	970.00	986	0.93	1.6
(9)	985.84	983.64	985.84	986.57	985.47	1030.00	970.00	986	−0.53	−0.9
(10)	985.84	986.57	984.38	984.38	985.29	1030.00	970.00	986	−0.71	−1.2
				Average					*−0.01*	*−0.024*

refresh time of 1000 ms provides highly accurate FPGA-measured and controlled on-chip sensor readings across the five different on-chip sensors.

7. Conclusions

In this research paper, we have proposed the design of efficient and high-speed circuit for real-time monitoring and control of on-chip sensor network for FPGAs. We developed the autonomous sensor agents implemented in the FPGA-based NI to be dynamically configured and to communicate the dynamic ambient parameter changes to the monitoring and control hardware unit. The ultimate goal of this research is to design low-cost, low power, and high accuracy real-time monitoring mechanism using autonomous sensor agents as well as a dynamic reconfiguration control of on-chip sensor network environment using FPGAs. We presented a detailed design procedure and a case study to demonstrate the applicability of the proposed approach. We showed that the proposed approach uses low FPGA logic resources and low power consumption compared with previous approaches, validating its suitability in real-system development. Furthermore, we demonstrated that, by collecting the dynamic and real-time monitoring parameters in terms of temperature and voltage variations, the system can adapt and improve the utilization of FPGA logic resources and power consumption. Experimental results from the FPGA-measured on-chip sensor readings showed high precision and accuracy in the measured voltage and temperature. We found that a dynamic refresh time of 1000 ms produces the best FPGA-measured and controlled on-chip monitored sensor readings as compared with the 100 and 500 ms refresh time. The proposed design techniques, framework, and protocol will assist network engineers and

Table 19: FPGA-measured on-chip Sensor #4 reading with dynamic reconfiguration refresh time of 100 ms.

S/number	Expt. 1 Sensor #4 measured reading (˚C)	Expt. 2 Sensor #4 measured reading (˚C)	Expt. 3 Sensor #4 measured reading (˚C)	Expt. 4 Sensor #4 measured reading (˚C)	Average Measured reading (˚C)	Range High (˚C)	Range Low (˚C)	Ideal Output (˚C)	Error Values	Full scale % error
(1)	55.49	50.45	47.99	48.85	50.70	100.00	40.00	50.00	0.70	1.2
(2)	55.98	50.20	48.23	48.72	50.78	100.00	40.00	50.00	0.78	1.3
(3)	56.23	50.08	47.49	49.46	50.82	100.00	40.00	50.00	0.82	1.4
(4)	56.61	49.96	48.60	48.48	50.91	100.00	40.00	50.00	0.91	1.5
(5)	55.49	50.45	48.23	48.48	50.66	100.00	40.00	50.00	0.66	1.1
(6)	55.74	50.57	48.36	48.60	50.82	100.00	40.00	50.00	0.82	1.4
(7)	55.74	50.57	47.37	48.97	50.66	100.00	40.00	50.00	0.66	1.1
(8)	55.74	50.69	47.62	48.72	50.69	100.00	40.00	50.00	0.69	1.2
(9)	55.37	50.20	48.23	48.48	50.57	100.00	40.00	50.00	0.57	0.9
(10)	55.74	50.32	48.23	49.34	50.91	100.00	40.00	50.00	0.91	1.5
				Average					*0.75*	*1.3*

Table 20: FPGA-measured on-chip Sensor #5 reading with dynamic reconfiguration refresh time of 100 ms.

S/number	Expt. 1 Sensor #5 measured reading (mV)	Expt. 2 Sensor #5 measured reading (mV)	Expt. 3 Sensor #5 measured reading (mV)	Expt. 4 Sensor #5 measured reading (mV)	Average Measured reading (mV)	Range High (mV)	Range Low (mV)	Ideal Output (mV)	Error Values	Full scale % error
(1)	12.94	13.18	12.94	13.18	13.06	100.00	10.00	13.00	0.06	0.1
(2)	13.92	13.18	13.43	13.43	13.49	100.00	10.00	13.00	0.49	0.5
(3)	13.67	13.18	13.18	12.94	13.24	100.00	10.00	13.00	0.24	0.3
(4)	13.67	13.43	13.43	13.18	13.43	100.00	10.00	13.00	0.43	0.5
(5)	13.67	12.94	13.18	13.43	13.31	100.00	10.00	13.00	0.31	0.3
(6)	13.18	13.67	13.18	13.18	13.30	100.00	10.00	13.00	0.30	0.3
(7)	13.67	13.67	12.94	13.18	13.37	100.00	10.00	13.00	0.37	0.4
(8)	13.43	13.18	13.43	13.43	13.37	100.00	10.00	13.00	0.37	0.4
(9)	13.43	13.43	13.67	13.18	13.43	100.00	10.00	13.00	0.43	0.5
(10)	13.43	13.43	12.94	12.94	13.19	100.00	10.00	13.00	0.18	0.2
				Average					*0.32*	*0.4*

Table 21: Comparison results of the sensor reading accuracy versus dynamic reconfiguration refresh time.

Sensors' accuracy/precision	Sensor #1 (mV) Range 970–1030 (mV)	Sensor #2 (mV) Range 1750–1890 (mV)	Sensor #3 (mV) Range 970–1030 (mV)	Sensor #4 (˚C) Range 40–100 (˚C)	Sensor #5 (mV) Range 10–100 (mV)
Dynamic reconfiguration refresh time = 1000 milliseconds (ms)					
Full-scale % error	−0.7	−0.1	+0.8	+0.1	+0.4
Error values	−0.43	−0.10	+0.48	+0.09	+0.34
Dynamic reconfiguration refresh time = 500 milliseconds (ms)					
Full-scale % error	−0.6	−0.6	+0.2	+1.1	+0.3
Error values	−0.34	−0.34	+0.10	+0.66	+0.24
Dynamic reconfiguration refresh time = 100 milliseconds (ms)					
Full-scale % error	−0.6	−0.3	−0.024	+1.3	+0.4
Error values	−0.37	−0.39	−0.01	+0.75	+0.32

system designers with a flexible and efficient real-time monitoring and control scheme for large and complex FPGA-based on-chip sensor networks and other related remote-sensing applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Chosun University Research Fund 2015. The authors would also like to thank Xilinx Corporation for providing the Zynq FPGA board and Vivado design tools. Special thanks are owed to the Associate Professors Antonio Filieri of Stuttgart University, Germany, and Martina Maggio of Lund University, Sweden, and Schloss Dagstuhl, Leibniz Centre for Informatics, Germany, for the Dagstuhl GI seminar 14382 "Control Theory Meets Software Engineering" 2014. Finally, they thank the Synopsys Corporation for providing the SaberRD integrated design environment and tools.

References

[1] C. Chu, Y. Ren, X. Liu, Y. Zheng, and W. Fang, "An efficiency multiplexing scheme and improved sampling method for multichannel data acquisition system," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 626307, 9 pages, 2015.

[2] M. D. R. Perera, R. G. N. Meegama, and M. K. Jayananda, "FPGA based single chip solution with 1-wire protocol for the design of smart sensor nodes," *Journal of Sensors*, vol. 2014, Article ID 125874, 11 pages, 2014.

[3] J. Echanobe, I. del Campo, K. Basterretxea, M. V. Martinez, and F. Doctor, "An FPGA-based multiprocessor-architecture for intelligent environments," *Microprocessors and Microsystems*, vol. 38, no. 7, pp. 730–740, 2014.

[4] C. G. Osuna, P. Ituero, and M. López-Vallejo, "A self-timed multipurpose delay sensor for Field Programmable Gate Arrays (FPGAs)," *Sensors*, vol. 14, no. 1, pp. 129–143, 2014.

[5] J. J. L. Franco, E. Boemo, E. Castillo, and L. Parrilla, "Ring oscillators as thermal sensors in FPGAS: experiments in low voltage," in *Proceedings of the 6th Southern Programmable Logic Conference (SPL '10)*, pp. 133–137, Ipojula, Brizal, March 2010.

[6] R. K. Umanath, *Designing Next Generation Low Power Autonomous Sensor Nodes Using Systems-on-Chip based Solutions*, Cypress Semiconductor, Published in EE Times Design, 2011.

[7] E. Y. Song and K. Lee, "Understanding IEEE 1451-networked smart transducer interface standard—what is a smart transducer?" *IEEE Instrumentation and Measurement Magazine*, vol. 11, no. 2, pp. 11–17, 2008.

[8] W. Elmenreich and S. Pizek, "Smart Tranducers Principles Commmunications and Configurations," http://www.vmars.tuwien.ac.at/~wilfried/papers/2003/rr-10-2003.pdf.

[9] H. Kopetz and T. Wien, "OMG smart transducer specification (II)," OGM Standard, 2003, http://www.omg.org/news/meetings/workshops/RT_2003_Manual/Tutorials/T4_Smart-Transducers_Kopetz_P2.pdf.

[10] Standard for a Smart Transducer Interface for Sensors and Actuators, "Common functions, communication protocols and Transducer Electronic Data Sheet (TEDs) formats," IEEE STD 1451.0-2007, IEEE Instrumentation and Measurement Society, TC-9, the Institute of Electrical and Electronics Engineers, inc, New York, NY, USA, 2007.

[11] V. Mattoli, A. Mondini, B. Mazzolai, G. Ferri, and P. Dario, "A universal intelligent system-on-chip based sensor interface," *Sensors*, vol. 10, no. 8, pp. 7716–7747, 2010.

[12] J. H. Huijsing, F. R. Riedijk, and G. Van der Horn, "Developments in integrated smart sensors," *Sensors and Actuators: A. Physical*, vol. 43, no. 1–3, pp. 276–288, 1994.

[13] G. Kornaros and D. Pnevmatikatos, "A survey and taxonomy of on-chip, monitoring of multi-core systems-on-chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 18, no. 2, article 17, 2013.

[14] G. J. García, C. A. Jara, J. Pomares, A. Alabdo, L. M. Poggi, and F. Torres, "A survey on FPGA-based sensor systems: towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing," *Sensors*, vol. 14, no. 4, pp. 6247–6278, 2014.

[15] N. G. S. Campos, D. G. Gomes, F. C. Delicato, A. J. V. Neto, L. Pirmez, and J. N. de Souza, "Autonomic context-aware wireless sensor networks," *Journal of Sensors*, vol. 2015, Article ID 621326, 14 pages, 2015.

[16] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires et al., "Autonomic wireless sensor networks: a systematic literature review," *Journal of Sensors*, vol. 2014, Article ID 782789, 13 pages, 2014.

[17] N. V. Kirianaki, S. Y. Yurish, N. O. Shpak, and V. P. Deynega, *Data Acquisition and Signal Processing for Smart Sensors*, John Wiley & Sons, Hoboken, NJ, USA, 2002.

[18] J. Xi, C. Yang, A. Mason, and P. Zhong, "Adaptive multisensor interface system-on-chip," in *Proceedings of the 5th IEEE Conference on Sensors*, pp. 50–53, IEEE, Daegu, South Korea, October 2006.

[19] S. Y. Yurish, "Digital sensors design based on universal frequency sensors interfacing IC," *Sensors and Actuators A: Physical*, vol. 132, no. 1, pp. 265–270, 2006.

[20] G. Song, A. Song, and W. Huang, "Distributed measurement system based on networked smart sensors with standardized interfaces," *Sensors and Actuators A: Physical*, vol. 120, no. 1, pp. 147–153, 2005.

[21] Smart Sensors Interface for IEEE 1451, http://www.jlm-innovation.de/products/ieee1451.

[22] K. M. Zick and J. P. Hayes, "On-line sensing for healthier FPGA systems," in *Proceedings of the 18th ACM SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '10)*, pp. 239–248, ACM, New York, NY, USA, February 2010.

[23] G. Kornaros and D. Pnevmatikatos, "Real-time monitoring of multicore SoCs through specialized hardware agents on NoC network interfaces," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW '12)*, pp. 248–255, Shanghai, China, May 2012.

[24] L. Fiorin, G. Palermo, and C. Silvano, "MPSoCs runtime monitoring through networks-on-chip," in *Proceedings of the Design Automation and Test in Europe Conference & Exhibition (DATE '09)*, pp. 558–561, Nice, France, April 2009.

[25] M. A. Aguirre, J. N. Tombs, V. Baena-Lecuyer et al., "Microprocessor and FPGA interfaces for in-system co-debugging in field programmable hybrid systems," *Microprocessors and Microsystems*, vol. 29, no. 2-3, pp. 75–85, 2005.

[26] W. Y. Alexander, L. Pasi, R. Pekka, N. Ethiopia, I. Jouni, and T. Hannu, "Hierarchical agent architecture for scalable NoC design with online monitoring sensors," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-41 '08)*, Lake Como, Italy, November 2008.

[27] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, "An event-based network-on-chip monitoring service," in *Proceedings of the 9th IEEE International High-Level Design Validation and Test Workshop —(HLDVT '04)*, pp. 149–154, IEEE, November 2004.

[28] Y. Wang, X. Jiang, H. Shengxi, L. Weichen, and Y. Huazhong, "A case study of on-chip sensor network in microprocessor system-on-chip," in *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09)*, pp. 11–16, ACM, Grenoble, France, October 2009.

[29] S. Andrew, "PSoC-based low-cost, intelligent network: physical and data link layer," Application Note AN2346, Revision A, Cypress Semiconductor, 2006.

[30] V. Petrescu, M. Pelgrom, H. Veendrick, P. Pavithran, and J. Wieling, "Monitors for a signal integrity measurement system," in *Proceedings of the 32nd European Solid-State Circuits Conference (ESSCIRC '06)*, pp. 122–125, IEEE, Montreux, Switzerland, September 2006.

[31] R. McGowen, C. A. Poirier, C. Bostak et al., "Power and temperature control on a 90 nm Itanium family processor," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 229–237, 2006.

[32] K. Sohn, N. Cho, H. Kim et al., "An autonomous SRAM with on-chip sensors in an 80 nm double stacked cell technology," in *Proceedings of the VLSI Circuits Symposium on Digest of Technical Papers*, pp. 232–235, IEEE, June 2005.

[33] C. Chan, Y. Chang, H. Ho, and H. Chiueh, "A thermal-aware power management soft-IP for platform-based SoC designs," in *Proceedings of the International Symposium on System-on-Chip*, pp. 181–184, Tampere, Finland, November 2004.

[34] E. Magdaleno, M. Rodríguez, F. Pérez, D. Hernández, and E. García, "A FPGA embedded web server for remote monitoring and control of smart sensors networks," *Sensors*, vol. 14, no. 1, pp. 416–430, 2014.

[35] C. Gomez Osuna, M. A. Sanchez Marcos, P. Ituero, and M. Lopez-Vallejo, "A monitoring infrastructure for FPGA self-awareness and dynamic adaptation," in *Proceedings of the 19th IEEE International Conference on Electronics, Circuits and Systems (ICECS '12)*, pp. 765–768, IEEE, Sevilla, Spain, December 2012.

[36] *Control Theory of Digitally Networked Dynamic Systems*, Springer, Cham, Switzerland, 2014.

[37] K. J. Phillip, *Feedback Control of Computer Systems (Introducing Control Theory to Enterprise Programmers)*, O'Reilly Media, Sebastopol, Calif, USA, 2014.

[38] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*, John Wiley & Sons, 2004.

[39] R. Dafali and J.-P. Diguet, "Self-Adaptive Network Interface (SANI): local component of a NoC configuration manager," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '09)*, pp. 296–301, IEEE, Quintana Roo, Mexico, December 2009.

[40] Synopsys, *SaberRD Integrated Design Environment (IDE)*, Student Edition, 2015, http://www.synopsys.com/cgi-bin/saberrd/reg1.cgi.

[41] Synopsys, "SaberRD User Guide, Version V-2004.06-SP1, Document Order Number: 00000-000 VA, Version W-2004.09," 2004.

[42] Xilinx inc, Xilinx Vivado 2013.3 Integrated Design Environment (IDE), 2015, http://www.xilinx.com/support/download.html.

[43] Xilinx, *SYSMON User Guide, UG58 (V1. 2)*, 2015, http://www.xilinx.com/support/download.html.

[44] Xilinx inc, XADC User Guide, UG480, 2015, http://www.xilinx.com/support/download.html.

[45] J. S. Mrinal and S. P. Radhey, *System Monitoring Using Zynq-700 AP SoC Processing System with the XADC AXI Interface*, Application Note: Zynq-7000 All Programmable SoC, XAPP1182 (V1.0), Xilinx, San Jose, Calif, USA, 2013.

[46] J. Pallav, A. Srinivasa, and J. S. Mrinal, *Using the Zynq-7000 Processing System (PS) to Xilinx Analog to Digital Converter (XADC) Dedicated Interface to Implement System Monitoring and External Channel Measurements*, Xilinx inc, Application Note: Kintex-7 Family and Zynq-7000 AP SoC, XAPP1172 (v1. 01), 2014.

[47] S. Hauck and A. DeHon, *Reconfigurable Computing the Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann Publishers, Elsevier, 2008.

[48] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[49] M. Platzner, J. Teich, and N. Wehn, *Dynamically Reconfigurable Systems: Architectures, Design Methods and Applications*, Springer, Dordrecht, The Netherlands, 2010.

[50] F. D. Patric, *Measurement and Data Analysis for Engineering and Sciences*, Taylor and Francis, Spencer, Ind, USA, 2nd edition, 2010.