IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 21, 2021, accepted January 25, 2021, date of publication January 29, 2021, date of current version February 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055650

Systematic Approach for State-of-the-Art Architectures and System-on-Chip Selection for Heterogeneous IoT Applications

RAMESH KRISHNAMOORTHY[1], (Member, IEEE), KALIMUTHU KRISHNAN[1], BHARATIRAJA CHOKKALINGAM[2], (Senior Member, IEEE), SANJEEVIKUMAR PADMANABAN[3], (Senior Member, IEEE), ZBIGNIEW LEONOWICZ[4], (Senior Member, IEEE), JENS BO HOLM-NIELSEN[3], AND MASSIMO MITOLO[5], (Fellow, IEEE)

[1]Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Chennai 603203, India
[2]Department of Electrical and Electronics Engineering, SRM Institute of Science and Technology, Chennai 603203, India
[3]Center for Bioenergy and Green Engineering, Department of Energy Technology, Aalborg University, 6700 Esbjerg, Denmark
[4]Faculty of Electrical Engineering, Wroclaw University of Science and Technology, 50370 Wroclaw, Poland
[5]School of Integrated Design, Engineering, and Automation, Irvine Valley College, Irvine, CA 92618, USA

Corresponding authors: Ramesh Krishnamoorthy (rameshk.tn@gmail.com) and Sanjeevikumar Padmanaban (san@et.aau.dk)

This work was supported by the Visvesvaraya PhD Scheme, Ministry of Electronics and Information Technology (MeitY), Government of India <MEITY-PHD-2822>.

ABSTRACT The Internet of Things (IoT) refers to a network of physical devices, which collects data and processes into a system without human intervention. In the commercialized market, IoT architectures are upgrading day by day to reduce data transmission costs, latency, and bandwidth usage for various application requirements. The extensively available IoT architectures and their specification resist the researchers to select a system-on-chip (SoC) for heterogeneous IoT applications. This paper seeks to comprehend the various IoT device specifications and their characteristics to support multiple applications. Moreover, microprocessor architectures and their components are detailed to facilitate developer knowledge in advanced methodology and technology. The various instructions set architectures (ISA) are implemented in a Zynq-7000 (xc7Zz20clg484-1) FPGA device to examine the feasibility of design space requirements for real-time hardware execution. To select specific system-on-chip (SoC) architecture for heterogeneous IoT applications, a genetic algorithm (GA) based optimization method is implemented in MATLAB. The proposed algorithm identifies the optimized SoC architecture concerning device parameters such as a clock, cache, RAM space, external storage, network support, etc. Further, the confusion matrix method evaluates the proposed algorithm's accuracy, which yields 84.62% accuracy. The outcome of SoCs attained through the GA are tested by analyzing their execution time and performance using various evaluation benchmarks. This article helps the researchers and field engineers to comprehend the microarchitecture device configurations and to identify the superior SoC for next-generation IoT practices.

INDEX TERMS Internet of Things, microprocessors, system-on-chip, heterogeneous architectures, and edge computing.

I. INTRODUCTION

The IoT refers to a universal presence of interconnected physical devices for a wide range of applications, such as consumer electronics, smart healthcare systems, and intelligent vehicles [1]–[3]. The IoT enabled gateway devices collects the necessary data from sensors or edge devices and sends to the cloud without human intervention [4]. Fig. 1 represents the number of IoT devices connected globally for upcoming years (2015-25). The data interpretation observed that several IoT devices installed for 2025 expect to grow more than fifty percent [5]. The critical requirement for using primary tools is to keep the bill of materials (BoM) as low as possible by utilizing low-cost microcontrollers with in-built memory

The associate editor coordinating the review of this manuscript and approving it for publication was Ramazan Bayindir.

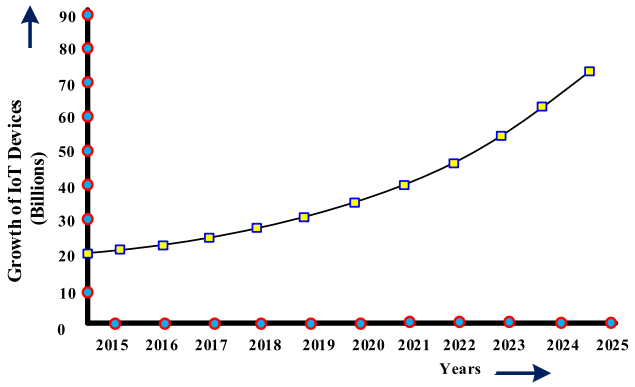


FIGURE 1. Statistics of IoT devices growth (Data source: Consumer electronics - 16 November 2016 [2]).

and storage. Based on an accurate study of device specifications, including a clock, RAM, Flash, Cache, power, and operating temperature, users can choose high-end devices. Besides, the advanced support features like camera, security, and audio/video support the devices should be considered for safety-critical applications. The SoC devices are semiconductor ICs with integrated components that enable the chip to operate as a standalone system. SoCs are more capable of increasing the system performance. The SoC lean towards to minimizing the latency delivered through various elements that are deliberately placed on the IoT boards to reduce the interference and interconnection delays and also to speed up the data transmission practice.

A. MOTIVATION

To develop the real-world embedded system SoC is inevitable for broad range of applications. The designers should follow the set of guidelines for selecting SoC devices. It is not necessary to find the proper match among the devices features and requirements during the initial stage. A small change in the addition of peripheral chips or requirements will impact the cost associated with the device. The evaluation process of several IoT devices should begin with categorizing the devices into basic, medium and advanced levels as determined by the application. The current researches are mainly focusing on the design of IoT applications for data sensing, centralized management, and automation services by making use of well-known devices familiar to the specific research group. In the industrial market, wide range of SoC devices are available, delivering high performance for diverse IoT applications. The recent research efforts focus on hardware acceleration on IoT and provided comprehensive survey of hardware devices for embedded prototyping. However, there is currently very little research that characterizes SoC device selection for IoT applications. This research gap motivates the author to develop a systematic methodology for optimal SoC device selection.

B. BACKGROUND ON IoT SoC

The SoC devices are semiconductor ICs with integrated components that enable the chip to operate as a standalone system. The SoCs has many features integrated on a single-chip, which is also capable of running an operating system on a smart device [6]–[8]. The microcontroller is the core of SoC technology, which is programmable and used for a wide range of embedded real-time applications such as home automation, smart appliances, security applications, etc. SoC devices are customizable, cost-effective, and more reliable. In contrast, advanced devices serve as a gateway to perform translation between different network protocols. These devices will manage the application control logic and wireless area network (WAN) interface. A smart home application where users can remotely monitor and control the status of appliances, whether turned ON or OFF, is achieved through necessary IoT based application development. In an industrial environment, the boiler's temperature can be remotely monitored and controlled using IoT devices.

The IoT devices embedded in the edge nodes are 32-bit or 64-bit microcontroller units (MCUs), which has versatile features in a single chip [9], [10]. The MCUs have more number of General-Purpose Input/Output (GPIO) signals for sensors integration. The wide range of embedded applications such as home appliances, wearable devices, building management systems and industrial automation, etc. makes use of IoT devices. Microcontroller's potential growth in smart appliances may frequently update the cloud server's data through the edge node. An edge node is a centralized node; it collects the data from devices and processes it to the cloud server. The evolution of the IoT increases the acquired and transmitted data considerably, which poses latency and bandwidth challenges. As IoT devices usage is rising; the devices should operate with low power and low latency. The devices and sensors placed under varying environments are battery-operated. Therefore, to preserve the battery lifecycle, IoT devices have to steadily wakeup from sleep mode to fetch new information. In case the device has less chance for data sharing, the device becomes dormant and runs at a slow speed [11]. Many IoT devices operate through a wireless connection. They utilize significantly less bandwidth, but due to more devices, they share the internet, and hence more bandwidth is required. That is, the amount of data collected and transmitted by the IoT devices will increase the bandwidth. For instance, video processing applications may significantly pose higher bandwidth. To alleviate the bandwidth challenges, the designer should consider the system won't always transmit or receive the data only through a wireless connection. These challenges force the embedded edge node resource constraints like complexity, battery capacity, cost, etc. Current research work progress on how IoT devices are connected and issues in establishing the device to device communication. But there are minimal research articles focused on addressing the critical microprocessor architecture device characteristics and selection requirements needed to design and develop applications.

Edge computing's impact is to reduce the latency and the data processing carried out at the cloud server via either Wi-Fi, 4G/5G networks to enable efficient networking

[12]. Ching Chen *et al.* presented intelligent IoT gateway architecture using multiple collaborative microcontrollers by applying master-slave principles [13]. The master MCU performs the local gateway computing there by the IoT system, saves the bandwidth costs, and reduces the communication response time between multiple devices. Here the gateway is realized by combining the multi-MCU design with a field-programmable gate array (FPGA) [14]. Increasing the data processing abilities of the cloud server does not shell out an increase in network latency. But it utilizes very minimal computational resources at the edge, which improves performance [15]. The more significant number of computational resources used in application domains has laid the path for designing the system by utilizing application-specific multicore processors to achieve high-performance requirements. Tiago Gomes *et al.* concluded that endpoint devices might significantly reduce computational resource availability by deploying a heterogeneous architecture-based edge device to handle parallel tasks [16], [17]. Each pipeline stage's activity rate will produce significant improvements in a pipelined processor [18]. Maryam *et al.* had proposed a low-cost instruction pre-fetching scheme with ultra-low power multicore processors. This mechanism allows the cache hit rate of 95%, thereby improving the performance double-times [19]. Due to more number of pipeline stages, the prefetching of instruction from the memory and processing through the queue enables the processors to achieve high performance. The modern processors like ARMv9 uses five stages of pipeline such as fetch, decode, execute, memory and writeback. Tobias Strauch has developed a thread controller using the ARM Cortex M3 processor with the supported peripherals. The standard CPU architecture may be transformed into multi-threaded CPU architecture by a well-known C-Slow Retiming (CSR) design transformation method [20]. The multithreaded CPU architectures with System Hyper Pipeline (SHP) method executes multiple threads, while the CSR method fails due to its limitations. In addition to scheduling threads in a dynamic fashion, the SHP method combines with parallel processing and CSR methods, which will deliver higher performance for critical tasks [21].

C. STRUCTURE AND CONTRIBUTION OF THE ARTICLE

The overview of the proposed work is shown in Fig. 2. In comparison, this paper aims to develop a genetic algorithm based optimization technique to identify the precise SoC for IoT applications. The main contribution of the paper as follows:

- This paper aims to provide a systematic methodology for the research community to determine the SoC selection requirements for IoT applications.
- We categorized the IoT devices based on basic, medium, and advanced devices, related the device characteristics and requirements with practical use-cases.
- We propose the design space investigation of various processor architecture. The results attained significantly proved that ARM architecture supports for fastest memory and I/O handling capability, registers utilization, and signal processing functionalities.
- This paper utilizes genetic algorithm optimization for optimal device selection. The genetic algorithm produces the results based on the SoC devices database loaded within the MATLAB.
- In this work, the GA results with multiple devices. The benchmark execution time and performance are experimented and tabulated to ensure the proposed work's effectiveness.
- The accuracy estimation for the proposed optimization algorithm is evaluated using the confusion matrix method and python programming model for the dataset of 26 devices.

D. RELATED WORKS

The existing works are focused on data sensing and image processing applications through IoT devices to enable the user to receive the information on time [22]–[24]. The IoT devices support for capturing of data from the environment without the intervention of the user. Linguaglossa *et al.* has proposed the extensive overview of the hardware architectures for network function virtualization (NFV) to provide abstractions for accessing components and physical resources into the ecosystem [25]. P. Shantharama *et al.* have demonstrated the Intel QAT hardware acceleration that performs image compression and decompression and achieved high network bandwidth [26], [27]. The work in [27] surveyed the emerging hardware platforms for executing softwarized network functions. P. Shantharama *et al.* has surveyed the emerging hardware platforms for executing softwarized network functions. There are few works in [25]–[27] that provides a comprehensive survey of hardware accelerated platforms associated for architectural enhancement by reducing the core utilization and expanding the ISA and cache memory access. Table 1 shows the works carried out in various literature. Initially, Kansakar *et al.* proposed a design space investigation methodology for choosing microarchitecture configurations for high-performance IoT processors. PARSEC and SPLASH2 benchmarks were taken and discussed the importance of low-power optimized processor design to evaluate the design space. The processor selection approach remained carried out using greedy search methods. The proposed processor configuration utilizes only 3% to 5% of the overall design space, significantly increasing the average speed up on processor design [28]. The work in [28] is significant towards the processor selection constraint. However, the greedy algorithms be unsuccessful to find the globally optimal solution for the reason that they do not consider all the data. Adegbija *et al.* seek to comprehend the microarchitectural characteristics to perform edge computing in the IoT. The four different CPU cores are considered and analyzed the impact of the stagnant energy on overall energy consumption using the matrixTrans_128 application benchmark. The power gating technique reduces the leakage power consumption by

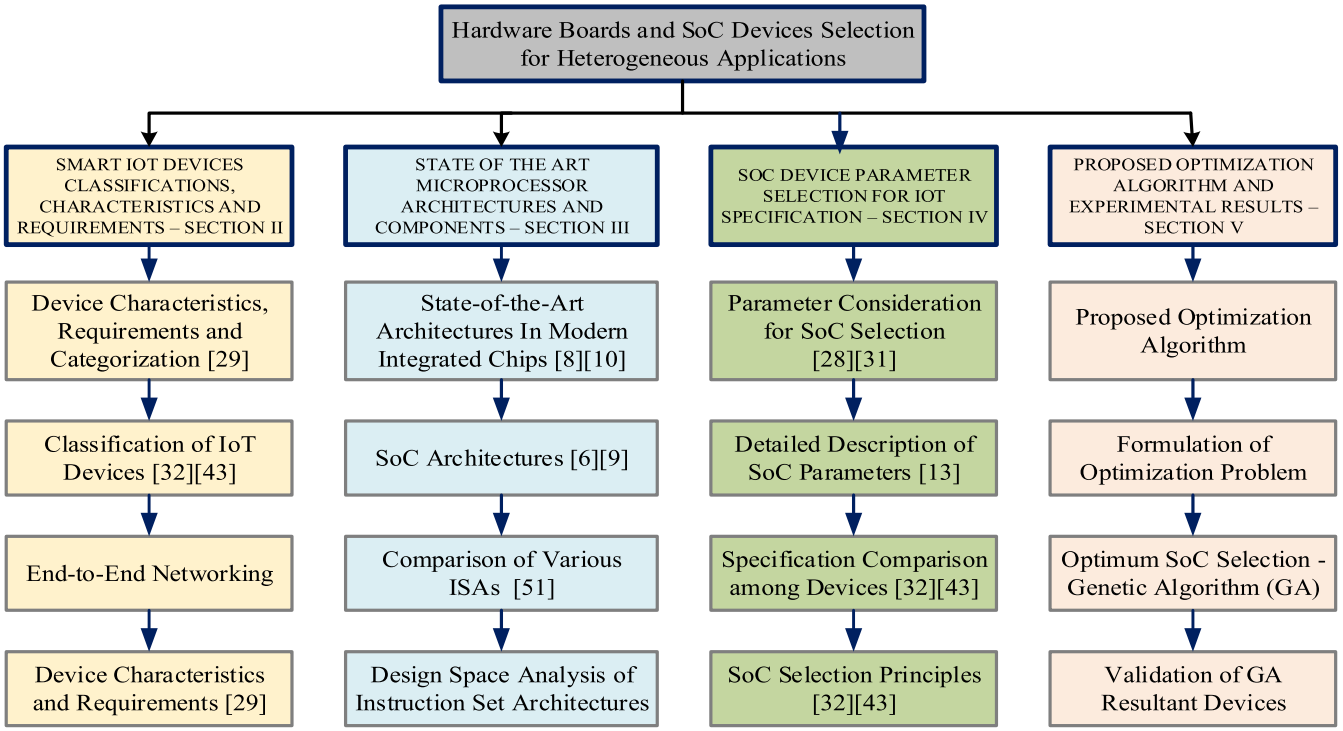


FIGURE 2. The overview of proposed work.

TABLE 1. Contribution of works in existing literatures.

Contributions	This Paper	Kansakar *et.al* [22]	Tosiron *et.al* [23]	Kansakar *et.al* [24]	Tosiron *et.al* [25]	Mike *et.al* [26]
Design Space Exploration	✓	✓	✓	✓	✓	×
Benchmarks Utilized	✓	✓	✓	✓	✓	×
Survey of Processor Architectures	✓	✓	✓	✓	✓	✓
Performance Optimization	✓	✓	✓	✓	✓	✓
Execution Time Estimation	✓	✓	×	✓	×	×
IoT Application Overview	✓	✓	✓	✓	✓	✓
Edge Computing	✓	×	✓	×	✓	✓
Survey of IoT Device Categories	✓	×	✓	×	×	✓
Optimum Device Selection	✓	×	×	×	×	×
Future Trends and Research Challenges	✓	✓	✓	×	×	✓

✓Topics Covered ×Topics Not Covered

shutting off the unused blocks, where the leakage power is reduced strictly by 95%. The performance and efficiency of conf1, conf2, and conf3 normalized to conf4 for all the benchmarks were computed in a non-energy constrained system. Results reveal that conf1, conf2, and conf3 degrade the performance approximately to 171x, 17x, and 8x, respectively. Compared to conf4, conf1 degraded the efficiency by 33x, while conf2 and conf3 degraded the efficiency by 4x. These results reveal the significant improvements achieved by using the larger configurations [29]. The work in [29] provides a foundation for advanced research to understand application requirements and architectures that support edge computing. However, the paper deals only with four specific hardware boards to analyze the benchmark implementation results.

Kansakar *et al.* has utilized exhaustive and greedy search algorithms for design space investigation and parameter optimization for multicore architectures. The design space estimation was performed using the cycle-accurate simulator and shared-memory computer benchmarks. The proposed algorithm using greedy search yields the best settings of 1.35% to 3.69% compared to a fully exhaustive search approach. The proposed methodology produces a better solution during the search phase upon considering more evaluation parameters [30]. The work in [30] followed the same approach presented in [28]. Moreover, the design space estimation for the considered benchmarks are again evaluated using the greedy algorithm which is computationally slow when compared to the evolutionary algorithms. Adegbija *et al.* presented a

comprehensive overview of architectures that explore the research challenges empowering the right provisioned architecture for IoT edge device computing [31].

The rest of the paper is organized as follows: In section II, this paper explores the smart IoT device characteristics and requirements. We discuss the comprehensive study of several microprocessor architectures and their internal components. In section III, we perform the design space analysis of various architectural configurations. In Section IV, we present the complete core factors that need to considered while selecting SoC configurations to improve the systems overall performance. Several microcontrollers available in the market with unique features and competitive improvements in their packing size, built-in communication interfaces, and memory capacity. It makes them adequate and inadequate for specific applications. Thus, repeatedly to circumvent the problem that comes while choosing the right microcontroller, developers usually choose the microcontrollers well-known. Finally, in section V, this paper presents our optimum algorithm for SoC selection, and the corresponding experimental results.

II. SMART IoT DEVICES CLASSIFICATIONS, CHARACTERISTICS AND REQUIREMENTS

The commercially available IoT devices are categorized as a basic, medium, and advanced, and their supported features are shown in Table 2. The low-end IoT devices operate with clock frequency ranging from 8 MHz to 400 MHz. These devices access the internal RAM ranging from 2 KB to 512 KB without cache memory. Flash memory supports to the range of 32 KB to 8 MB. These low-end devices do not have inbuilt advanced features, including security, camera, and audio/video. These devices are constrained to operate in windows or Linux OS environment with limited resources. Examples of basic devices are Atmel SmartD21, ESP32 Azure, and Libelium Waspmote.

However, for sensing and actuating applications, these devices are primarily used. In contrast, the medium-range IoT devices function with clock frequency from 1 GHz to 2.13 GHz. These devices contain 16 KB of level1 cache and 512 KB of level2 cache and involve a maximum of up to 2 MB of on-chip cache. These devices support internal RAM access from 512 MB to 2 GB (DDR3) and up to 4 GB of flash memory. The middle-end IoT devices, support advanced features like security, camera, audio/video processing, and consist of one or more communication interfaces, unlike basic devices. Examples for medium devices are Cubie Board, BeagleBone Black, and Intel Edison Compute Module. Considering high-end IoT devices, they operate with clock frequency ranging from 1 GHz to 2.13 GHz.

The devices consist of 16 KB of level1 and a maximum of 2 MB of on-chip Cache. It supports internal RAM access from 512 MB to 4 GB (DDR3) and up to 16 GB of eMMC memory. All the commercially available devices may support advanced features like security, camera, and audio/video in the high-end IoT devices. These devices are familiar with their onboard communication network support such as Ethernet, Wi-Fi, and Bluetooth, etc. Also, these can process multimedia applications using a camera serial interface (CSI) module and execute complex machine learning algorithms. These advanced devices are IoT gateways due to a high level of onboard features, making them more effective in performing intelligent data analytics at the edge network. These devices are useful for applications including media computer vision, virtual reality (VR), augmented reality (AR), industrial automation, etc.

The comprehensive survey of categorizing IoT devices into low, medium, and the high-end application was presented in [32]–[34]. The characteristics of IoT systems that differentiate from other linked systems are discussed [35]. During the boot sequence of a device, the processor executes the boot loader's software, target compatible operating system software, and other application softwares. Immediately upon the successful boot process execution, the device must operate more securely. The IoT solutions usually comprise sensors, connectivity, data processing, and a user interface.

A. THINGS

The IoT technology is essential for the smart home device, such as a camera for monitoring the home during abnormal conditions. An Internet protocol (IP) enabled camera framework can monitor, capture and send the video instantly to the user through the web interface. A smart clock is used to play music, weather display, cricket scores updates, and controlling smart home devices through voice. Simultaneously, the intelligent thermostat can self-learn about the individual users expected indoor temperature and indicate the user if something goes wrong. The IoT devices with sensors and actuators will detect the event, monitor the environment's transitions, and convert these physical quantities into electrical signals. Sensors are the entities that gather data and transfer through the internet for processing information from sensors such as image, smoke, light, and temperature. The sensor output are shared with the other devices with the availability of connectivity support. Therefore, wired or wireless connectivity among the devices is most vital. The processing capability and quantity of storage space required to handle this information is also significantly high. Thus, the interfacing of devices with cloud infrastructure enables higher support for processing and storage. In contrast, the actuator can operate in the environment to maintain and deliver the IoT devices' functionality. The actuator used in various categories can support interfacing motors, solenoids, and relays. The cluster of nodes associated with various protocols can communicate through the gateway.

B. GATEWAY

Extensive infrastructure has been provided by cloud platforms to allow authentication and identity control of devices. The gateway performs connectivity, processing, and data storage in the cloud infrastructure referred to as data models [36]–[38]. The gateway also performs authenticating, monitoring, controlling, software update, and maintenance.

TABLE 2. Devices categorization with supported peripherals.

Devices	CPU	Clock	Cache	RAM	Flash	Security	Camera	Audio/Video Support
BASIC DEVICES								
ATMEL Smart D21	ARM Cortex-M0 (32-Bit)	48 MHz	None	256 KB	32 KB	No	No	No
Light Blue Bean+	ATMega 328P MCU (8-Bit)	8 MHz	None	2 KB	32 KB	No	No	No
Particle Photon	ARM-based STM32F205 Cortex M3 processor, SOC-STM32F205 (32-Bit)	120 MHz	No Cache - No MMU	128 KB	1 MB Flash	No	No	No
Arduino UNO R3	Atheros AR9331, SOC-ATmega32U4 @ 16 MHz (8-Bit)	400 MHz	None	64 MB DDR2, 2.5 KB SRAM	16 MB Flash	No	No	No
LibeliumWaspmote	ATmega1281, SOC-ATmega1281 @ 14.74 MHz (8-Bit)	14.74 MHz	None	8 KB SRAM	128 KB Flash	No	No	No
SIPEED Maixduino RISC-V-AIIoT	RISC-V Dual core 64-bit with FPU, neural network processor (64-Bit)	400 MHz	None	256 KB	32 KB	No	No	No
NXP K64F	MK64FN1M0VLL12 (32-Bit)	120 MHz	None	256 KB	1 MB	No	No	No
MEDIATEK LINKIT ONE	ARM 7EJ-S, SOC-MT2502A (32-Bit)	260 MHz	None	4 MB	16 MB	No	No	No
Microsoft ESP32-Azure IoT Kit	Single core Xtensa 32-bit LX7 CPU (32-Bit)	240 MHz	None	320 KB	128 KB	Yes	No	Audio-Yes
Kinetics K22F	ARM Cortex-M4 MCU with FPU (32-Bit)	120 MHz	None	128 KB and 4 KB FlexRAM	1 MB	Yes	No	No
Intel Galileo Board	32-bit Intel Pentium ISA Compatible processor, SOC-Intel Quark SOC (32-Bit)	400 MHz	16 KB L1 Cache	256 MB DRAM and 512 KB SRAM	8 MB NOR Flash	No	No	NO
MEDIUM DEVICES								
Cubie Board	ARM Cortex-A8, SOC-AllWinner A20 SOC (32-Bit)	1 GHz	L1: 32 KB I-Cache + 32 KB D-Cache, L2: 512 KB Cache	1 GB DDR3	Up to 32 GB SD Slot, 4 GB Flash	Yes	No	Yes
Humming Board	i.MX6 based Quad Core processor CPU: ARM Cortex A9 Core (32-Bit)	1 GHz	None	2 GB DDR3	8 GB eMMC	No	Yes	Yes
BeagleBone Black	ARM Cortex A8, SOC-Sitara AM3358/9 (32-Bit)	1 GHz	L1: 32 KB I-Cache + 32 KB D-Cache, L2: 256 KB Cache	SDRAM 512 MB DDR3	4 GB eMMC Flash	No	No	Yes
Intel Edison Compute Module	Intel Atom Tangier Z3480 (64-Bit), SOC-Intel Quark MCU @ 100 MHz and Intel Atom Mcu @ 500 MHz	2.13 GHz	1 MB Smart Cache	1 GB DDR3	4 GB eMMC Flash	No	No	Audio-Yes
UDOO NEO	NXP i.MX6 soloX processor	200 MHz	None	1 GB	None	No	No	Yes
MINNOW BOARD MAX TURBOT QUAD-CORE	Quad Core Intel Atom E3845 (32-Bit and 64-Bit)	1.91 GHz	2 MB On-Chip Cache	2 GB DDR3L	SPI Flash/Boot Rom, 8 MB SPI Flash	No	No	Yes
AllWinner R8	ARM Cortex A8, SOC-AllWinner R8 (32-Bit)	1 GHz	L1: 32 KB I-Cache + 32 KB D-Cache, L2: 256 KB Cache	512 MB SDRAM	None	No	No	Yes

TABLE 2. *(Continued.)* **Devices categorization with supported peripherals.**

ADVANCED DEVICES								
SAMSUNG ARTIK 530S	ARM Cortex-A9 Quad Core, SOC-MT2502A (32-Bit)	1.2 GHz	L1: I Cache - 32KB D-Cache - 32KB and L2: 1 MB Shared	1 GB DDR3	4 GB eMMCv4.5	Yes	Yes	Yes
SAMSUNG ARTIK 5	ARM Cortex-A7 Dual Core 32-Bit)	1.0 GHz	L1: I Cache - 32KB D-Cache - 32KB and L2: 1 MB Shared	512 MB LPDDR3	4 GB eMMCv4.5	Yes	Yes	Yes
Qualcomm Dragon Board 410C	ARM Cortex-A53 Quad Core (64-Bit)	1.2 GHz	None	1 GB LPDDR3	8 GB eMMCv4.5	Yes	Yes	Yes
NXP i.MX 7 Dual Processors	Two ARM Cortex-A7 Plus and One ARM Cortex-M4 (32-Bit)	ARM Cortex-A7 @ 1 GHz and ARM Cortex-M4 @ 200 MHz	In Cortex-A7: 32KB Cache In Cortex-A4: 32KB Cache and 64KB TCM	None	None	Yes	Yes	Yes
Raspberry Pi 3 Model B+	ARM Cortex Quad-core A53, SOC-Broadcom BCM 2837B0 (64-Bit)	1.4 GHz	L1: 16 KB I-Cache + D-Cache, L2: 128 KB	1 GB LPDDR2 SDRAM	4 GB	Yes	Yes	Yes
MIPS Creator CI20	MIPS-Xburst Dual-core, SOC-Ingenic JZ4780 (32-Bit)	1.2 GHz	L1: 32 KB I-Cache + 32 KB D-Cache, L2: 512 KB Cache	1 GB DDR3	None	Yes	Yes	Yes
Intel Joule Compute Module	Intel Atom Processor Quad-Core, SOC-Intel Atom Broxton-M T5700	1.7 GHz	4 MB Level-1	4 GB DDR4	16 GB eMMC	Yes	Yes	Yes

The installed device can automatically establish communication between the other devices. After successfully collecting data, the device transfers the data to the cloud server

and receives messages serially from the cloud. The encryption algorithm can be ported in the device to enable secure data transmission among the devices. In the back end, the transport layer security and secure socket layer protocols to facilitate data security. The devices connected to the cloud environment can perform the software update and maintenance to ensure better compatibility with devices. The primary aspect is to provide extensive infrastructure to handle device applications and firmware. Thereby the infrastructure shall fix the bugs, manage security-oriented issues, and increases the functionality. In an IoT, the most commonly used end nodes are microcontrollers, human-machine interfaces, sensors, and actuators. In some applications, the end nodes directly communicate with the cloud without using a gateway or edge node. A generalized block diagram of the interaction between IoT devices and cloud servers is shown in Fig. 3.

C. PROTOCOLS

The MQTT protocol is a lightweight protocol intended for the machine to machine applications, which allows the publish/subscribe communication model [39]. It is beneficial for connections with distance locations where network bandwidth is exceptional. The CoAP is suitable to operate in a resource-constrained environment. CoAP enables request/response messaging model between endpoints, quickly interprets HTTP for more straightforward web integration, and allows multicast topology with reduced overhead. The HTTP protocol allows the client-server architecture model, which is primarily useful for web applications. It can be used by the IoT devices to publish a

bunch of data. It accepts a standard IP header for packet routing and runs over TCP and UDP protocol. The HTTP is essential to collect and process the big data from the entire ecosphere, and the message size of HTTP is large compared to MQTT. The AMQP is an application layer protocol designed for interoperable message-oriented middleware applications. The AMQP supports the client/server communication model in IoT device management. AMQP delivers a secure connection and message acknowledgment to ensure reliable message transfer and provide extensibility support.

D. END-USER DEVICES

The end node devices are low-cost microcontrollers with integrated sensors that utilize wireless protocols such as Bluetooth, ZigBee, etc. The long range communication can be established using TCP/IP supporting devices. These devices may also support energy harvesting with intensive power management strategies. On the other side, the edge nodes

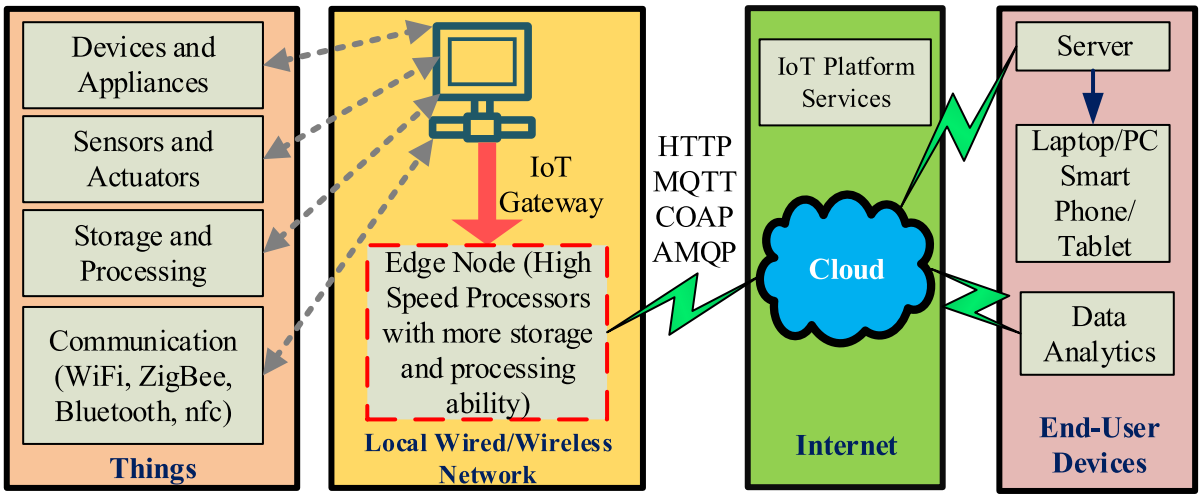


FIGURE 3. Things and end-user device networking.

TABLE 3. Security protocol stack layer.

Protocol	Corresponding Layer	Security Support
Zigbee	Physical/Mac layer	The Zigbee protocol uses AES-128 encryption which makes it secure for IoT applications while using mesh topology. The requirements include providing the best functionality with security measures in place.
MQTT	Application layer	On the transport level, the communication sequence is encrypted and identities are authenticated. The MQTT protocol provides a client identifier and username/password credentials to authenticate devices on the application level.
CoAP	Application layer	CoAP is a UDP protocol rather than a TCP protocol, TLS is not used by default. Instead, encryption is implemented using Datagram Transport Layer Security (DTLS) and alternately with IPSec.
6LowPan	Adaptation layer	Bootstrapping keys, IP Sec, RSA, ECC. This protocol used to shorten IPv6 packets and located between the network and data link layer.
AMQP	Application Layer	This protocol uses either SASL or TLS encryption on the transport layer and defends communication among nearby devices on wireless personal area networks.

are the most potent multiprocessor devices with high computation capabilities, and it can operate without an operating system for low-end applications. Moreover, the complex application requires an RTOS or GPOS, such as Linux, often being deployed. Real-time operating system (RTOS) provides functionalities such as device driver handling, memory management, scheduling, TCP/IP, upper-layer protocol stack interface, etc. Due to limited computation features, the basic devices do not use the operating system. To handle several RTOS functionalities, advanced devices with complex computation abilities will utilize OS effectively. These devices are built with a high-performance internet protocol stack to enable efficient networking. A general-purpose operating system (GPOS) is a primary component in system design and responsible for running the application. It supports the execution of multiple tasks and appropriate scheduling policy to dispatch threads and processes that enable higher throughput for desktop applications. For instance, Windows, Linux is GPOS. The significant limitations of GPOS are latency and no guarantee for high priority thread execution. RTOS is a software-based design used for time-dependent use cases where the processing time should be less than GPOS.

The porting of RTOS enables rewriting the application in embedded hardware. GPOS cannot handle real-time application tasks due to its latency issues.

In contrast, RTOS codes are scalable and suitable for real-time applications. Therefore, developers can choose the necessary kernel objects based on their application requirements. The future IoT devices require modern, optimized microprocessors that can operate under varying environments and characteristics. The device characteristics and requirements of IoT devices summarized as follows.

E. SMART

The IoT system implemented using hardware and software incorporating algorithms and computations makes the devices more intelligent. The intelligence in IoT devices strengthens its capabilities, which helps to react smartly and support the devices to complete the precise tasks. Artificial intelligence and machine learning create the devices to think cognitively and make appropriate decisions based on earlier experiences [40]. The smart devices placed in the critical environment can withstand even in hazardous locations to

enable real-time data transmission for robotics, industrial control, and self-driven vehicle applications.

F. DATA SENSING AND CONNECTIVITY

The IoT devices can understand the environment and process the analog data received from real-world applications. The sensors connected with the devices shall continuously monitor the real-time environment changes or appliances and report its status to the end-user [41]. The connectivity role is pivotal in enabling new market opportunities, compatibility, and networking among smart appliances. The devices state is dynamic, i.e., devices may operate either in power-down mode or default mode, connected or disconnected mode. Sometimes, the number of devices sharing the network also changes dynamically to time and place.

G. COLOSSAL SCALE

Due to the rapidly increasing number of smart consumer electronics devices, those devices must adequately manage to communicate with each other through internet technologies [42]. The enormous amount of data generated from smart appliances will cause data handling and interpretation more critical.

H. DIVERSIFICATION

IoT devices developed using different hardware boards and networks to work together with other devices over diversified networks [43]. The microprocessor architecture should support network connectivity directly in heterogeneous networks. The primary design necessities in IoT are scalable, extensible, and commutable.

I. ENERGY

The development of IoT brings a massive amount of edge devices. The edge node devices can be deployed quickly and do not require a physical connection to electrical infrastructure [44]. The microprocessor selection can bring the system to operate in low power mode and wait for an interrupt signal to awaken the system. Integrating a considerable number of devices that run on batteries can create a severe problem. Therefore, researchers and product development teams should focus on energy harvesting, charging topologies, and infrastructure to design a smart power ecosystem.

J. SAFETY AND SECURITY

The safety and security measures support various IoT protocols in different layers listed in Table 3. Multiple device interfaces, significant data handling, multiple network connectivity, and exploiting various IoT protocols will make the system more complex and less secure [45]. The early detection of deformities in safety applications is vital. Failing to locate anomalies may sometimes cause catastrophic failure. Therefore, it is mandatory to monitor the system consistently and gather the logging activities for troubleshooting abnormalities. Consequently, to secure the end nodes, edge nodes, networks, and data transfer across all nodes to create a security paradigm to prevent data from vulnerable attacks, additional efforts must be taken [46].

III. STATE OF THE ART MICROPROCESSOR ARCHITECTURES AND COMPONENTS

A. EXTENSIBLE OR SCALABLE ARCHITECTURE

The scalable microprocessor architecture-based design practices are becoming more complex due to increased networked devices. However, it provides the best possible functionality with faster response times and reduced power consumption. The deployment of scalable edge devices close to cloud infrastructures results in low latency, decreased reliability, and an increase in bandwidth [47]. The edge computing device significantly requires high-performance computing resources to address latency and bandwidth issues. Therefore, an essential requirement is to design reliable architectural resources and reduce the cost of hardware and software development for scalable architectures. However, we aim to address the cache memory implications within the scalable microprocessor architecture devices due to which it has a significant impact on processor performance. The IoT microprocessors should be adaptable following the IoT application necessities. The configurations of the microprocessor should be modified during design to achieve better efficiency. The reconfigured microprocessor with more internal components will be adaptable for various application design. The major factors, such as pre-fetch issue queue and pipeline mechanisms, have an essential role in performance improvement. The scalable architectures utilize the cache memory area efficiently rather than using the main memory. Since the cache has a potential impact on the microprocessors area, power, and performance in this section, we focus on the significance of cache memory. The register file is tightly coupled to the microprocessor core [48]. The memory hierarchy is categorizing as primary level and secondary level. In the primary level, Level-1 (L1) or primary cache and tightly coupled memories (TCMs) are associated with the processor through on-chip interfaces. In addition to this, the main memory is allied with the primary level (SRAM, DRAM, and flash memory) to store the programs while the processor is executing. In comparison, secondary level memory devices such as external disk drives and removable memory storage have a large area and are almost slow. The data accessed from the peripherals require too long access times. A cache can be included across any particular hierarchy. However, the access time varies among the memory components [49].

The cache memory takes data and instruction from disks, central memory (DRAM), off-chip cache (SRAM) and places near the processor core to increase the performance, ability, and cost per bit. Some of the IoT processor devices have Level-1 (L1), Level-2 (L2), and Level-3 (L3) caches. The L1 cache is an on-chip memory area that stores the program and data from onboard memory temporarily to reduce the access time required for both data and code. The L2 secondary cache is on the processor chip located between the

primary cache and board-level memory. While, the L1 cache is located directly on the CPU itself, and it ranges from 8KB to 64KB. However, to perform read operation from the CPU, L1 is the faster memory type. In multicore CPUs, each core has an L1 cache separately. Generally, L2 and L3 have a large volume of memory compared to L1, but it requires a long time to access data. The L2 cache memory usually a distinct chip located between the CPU and random access memory (RAM). The data types enable the processor core to access the main memory directly. The ARM7 through the ARM10 processor family uses a logical cache where the data can be accessed directly by the processor from the logical cache without the memory management unit (MMU). The cache has a significant implication in terms of security as the threat and backset vulnerabilities.

B. SHARED HETEROGENEOUS ARCHITECTURE

From the design perspective, the foremost benefits of heterogeneous architectures for IoT microprocessors core consists of CPUs, digital signal processors (DSPs), graphics processing units (GPUs), etc. and it can be able to reuse again in the implementation stage. This permits earlier design and verification exertions to be paying off. Though the design space offered by the processor architectures (HPA) is considerably smaller, unlike adaptable architectures [50].

The HPA requires diligent effort in finding the most exceptional processor core configurations that will fulfill the application necessities. Moreover, in a system with vast applications, the configuration optimization potential is realized lower in heterogeneous cores than configurable cores. During the cache miss operation, the CPU can't do anything, but in the meantime, another thread can utilize the processor resources. If the algorithm discloses high-level instruction parallelism, then the core available in sequence would be slower and inefficient. While performing floating-point operations and single instruction multiple data (SIMD) instructions, the memory access has long dormancy. A processor can achieve maximum throughput upon having more pipeline stages and considering the code optimization techniques explicitly for the processor architecture instructions. The processor takes more cycles to fill the pipeline stages, which may cause an increase in latency. The presence of multiple cores increases the speed execution speed; however, the power consumption will be slightly high. Notably, in ARM Cortex A8 IoT CPU core, the unavailability of multicore and order of execution pipelined architecture operates with reduced power consumption and area. The multicore processors are more beneficial to deliver high performance for edge computing applications such as audio-video processing and complex data acquisition systems [51]. The CPU can effectively exploit the branch prediction techniques. Thereby a considerable amount of power and area is reduced. Therefore, the multicore heterogeneous architectures are more proficient in executing operational codes in both out-of-order and in-order routines. Considerable earlier research works have directed heterogeneous cores in embedded systems and general-purpose computers, though their application to IoT microprocessors should be determined [52]. The applications designed with suitable cores and selections of appropriate cores are the prime challenges that need to address while designing and developing various microprocessors. The designers entail knowledge on core configurations and prior understanding and analysis of applications, exemplified in the microprocessor. The symmetric multiprocessing system consists of a single kernel, different cores utilizing the same instruction set architecture, and execute an available operating system using shared memory. The scheduler will decide the environment to empower load balancing, permitting processes to run on several cores at different times. On the other hand, an asymmetric multiprocessing (AMP) system consists of multiple cores, numerous software processing abilities, supports heterogeneous architecture using shared or separate memory [53]. Typically, the new OS is running on the system, which is unglued for each core architecture.

For example, a gateway requires high-speed internet connectivity to process graphical information, which runs on the Cortex-A core. In contrast, the process control and algorithms for monitoring purposes will run on the Cortex- M core separately. The AMP system is optimized to perform computation like off-loading audio processing towards low power Cortex-M processor core. Most of these operations require an RTOS to run with hard real-time constraints. The developed architecture near the Cortex-M core must enable single-cycle access at high speed between masters to the slave device to facilitate these necessities. The high-performance RTOSs such as Nucleus can deliver real-time processing on the ARM Cortex-A processor core. To attain higher efficiency in processing audio, video, data with reduced power consumption, a heterogeneous multicore configuration executing in the AMP approach is the right choice.

C. ENERGY SAVING ARCHITECTURE

Smart home devices are emerging in recent years, as they become innovative, ranging to not just automation and also provides safety and security. Handling smart electrical appliances such as microwave oven, a smart induction cooktop that tip-off before the malfunction, and protects the user from any terrible situations. The energy-saving in IoT smart home is another challenge; therefore, smart thermostats and smart illumination system supports the user in saving energy and the cost of electric bills. The sensors and home appliances will communicate information directly to the user using IoT endpoint devices with internet connectivity support. Therefore, battery-operated devices (BODs) are becoming an unavoidable option; however, battery replacement is not economical for specific applications. Since the device requires sufficient power, foraging energy is another approach that can address battery-oriented problems.

The energy harvesting can facilitate electronic systems to drive for years on varying power sources. Chen Pan *et al.* proposed an ENZYME software model to improve edge-nodes

TABLE 4. Comparison of instruction set architectures.

Architecture	Low Power Support	Industry standard Software Support	Cache Support	Pipeline Stages	Hardware Design	Instruction Size	Available Registers	Memory Access	Reference(s)
VLIW	Yes	Yes	Yes	5	Multiple Pipelining Stages	One size	Many GP Registers	Load/Store Architecture	[57]
MIPS	No	Yes	Yes	5	Support multiple stage of pipelining	4 Bytes	Many GP Registers	Load/Store Architecture	[58], [59]
RISC-V	Yes	Yes	Yes	6	Single stage pipeline implementation	32-Bits	Many GP Registers	Load/Store Architecture	[60], [61]
ARMv7	Yes	Yes	Yes	3	Three stage pipelining with microcoded	Varies	Many GP Registers	Load/Store Architecture	[62]
CISC	No	Yes	No	5	Support microcoded implementation	Varies	Registers with special instructions	Supports CISC instructions	[63]

energy efficiency using ultra-low energy harvesting (EH) power supplies [54]. The energy harvesting system (EHS) consists of sensors and data conversion circuits in which the sensor acquires energy from various sources and converts into electrical form. Lithium-ion (Li-ion), alkaline, nickel-cadmium (NiCd) batteries, and super-capacitors can be used to store and process the transformed electrical energy. The regulator will handle the power according to the system requirements.

D. APPLICATION-SPECIFIC PROCESSOR ARCHITECTURE

Table 4 illustrates the comparison of state-of-the-architectures instruction set architectures. To attain better performance, efficient energy requirements in IoT applications, the hardware-based design methods are utilized typically. The most common hardware design approaches such as application-specific integrated circuits (ASIC) and FPGA based solutions are well suited for IoT applications in which ASIC can deliver better performance. Still, a diligent effort must be taken in design [55]. Due to reduce overhead in architecture, the ASIC has a significant benefit compared to general-purpose processors (GPPs). Designing a processor with a limited set of functions required by the application may achieve very high performance, surpassing GPPs. Broadly ASIC design lack the competence to adapt and accord with changing application necessities [56].

The organization of cache differs from a general-purpose processor to DSP or ASIC devices. The industry is currently moving towards structured ASIC from a full custom or general-purpose processor based on the volume and end product shelf life. The application functionalities designed for microcontrollers must be programmed using register-transfer levels (RTL) logic programming languages such as Very High-Speed Integrated Circuits Hardware Description Language (VHDL) or system Verilog. Many high-level synthesis tools exist for ease of ASIC design. Using the appropriate platform, the designer can create hardware from software, providing a meaningful improvement in development time and effort to RTL design. The increasing acceptance of IoT move from the existing chip designs to custom silicon. Subsequently, there has been a substantial transformation in the complete IoT ecosystem. The semiconductor organizations can influence the market, considering they can design customized IoT chip at less cost.

Rather than contingent on using off-the-shelf components in IoT edge product design and development, the ASIC designers utilize the customized silicon design. The utilization of custom ASIC design for various IoT applications significantly improves the functionality and allows greater flexibility in design. To diminish the dynamic power consumption among multiple plants, the designers adopt power-saving methods appropriate to lower geometry systems, typically ranging from 180nm to 4nm technology. The researchers currently focus on attaining better dynamic power through clock gating techniques to strengthen the geometrical aspect, avoiding power leakage through proper biasing.

The foremost architectures presently used by IoT developers are Advanced RISC Machine (ARMv7), Microprocessor without Interlocked Pipeline stages (MIPS), and X86. The other traditional architectures, including RISC and variable-length instruction word (VLIW) [57] cannot adapt at runtime. In contrast, we have made a design space investigation of CPU architectures, including variable-length instruction word (VLIW), MIPS [58], [59], reduced instruction set computer (RISC) [60], [61], and ARMv7 [62], [63] architectures. This work is compiled and synthesized using the Xilinx Vivado v.2016.4 software tool and implemented using Xilinx Zynq-7000 (xc7Zz20clg484-1) FPGA Device. We tested the processors' data path structures and functional simulation of their internal components. We synthesized the VHDL script for processor architectures and the internal components using the inbuilt CPU clock frequency (FCPU) in the FPGA board. The processors are tested by writing a data word to an external

memory location in the FPGA devices to ensure the working of designed code modules. We developed and analyzed the PERL scripts code for the various sub-modules, including a register file, barrel shifter, ALU, address decoding, and control unit.

The implementation of processor architectures demonstrates resource utilization and area consumption in an FPGA device. The other coprocessors available on the board can utilize remaining memory resources. The processor's speed performance can be achieved by thoroughly analyzing and optimizing the architectures' pipelining flow. From the results (shown in Table-V) attained for the high-end design, the devices with higher performance such as ARMv7 architectures and MIPS are significant due to availability of bonded input-output (I/O) blocks. The I/O blocks are used to perform the I/O operations for interfacing the external memory and peripheral device in an FPGA.

In terms of image processing applications, the devices that support DSP operations are reliable such that ARMv7 uses three DSP modules, VLIW uses two DSP cores. In comparison, the other MIPS and x86 architectures are undependable.

The slice registers occupancy for ARMv7 followed by MIPS is higher than that of other processor architectures. It will support the fastest data access using the registers and allow the compiler to execute the instructions independently. We realize that the VLIW, X86, and RISC architectures use fewer slice LUTs and slice registers compared to ARMv7 and MIPS. The processor architecture's resource utilization in FPGA is verified based on the number of slice registers, slice LUTs, Bonded IOBs, memory usage, and signal processing capabilities. The result shown in Table 5 demonstrates ARMv7, and MIPS architectures utilize more slice logic, registers, and bonded IOBs, followed by RISC, X86, and VLIW architectures. In the future, design space study with contemporary architectures such as ARM Cortex A53, A57, and A72 configurations will be investigated for the growth of low-power IoT systems.

IV. SoC DEVICE PARAMETER SELECTION FOR IoT SPECIFICATIONS

The designer's primary concern is to fulfill the sensor node selection since an end node with an 8-bit MCU that can sense and transmits the limited data times in a day. Though for involved end nodes and smart gateway devices requires advanced algorithms; therefore, 32-bit MCU is the right option. The 32-bit microcontrollers, such as the ARM Cortex-M series MCU core, have floating-point units for implementing sophisticated algorithms. The higher processing power of the 32-bit MCUs empowers processing faster by switching to sleep mode and save power [64], [65]. Moreover, these MCUs have higher flash and RAM area, enabling the designers to develop the complete protocol stack and application program code on the microcontroller devoid of requiring an additional CPU in the system. While performing sophisticated filtering approaches for energy management applications like H-infinity and Kalman filtering, the processor depends on complex matrix computations. As a result, 32-bit MCUs with FPUs can be used [59]. The microcontroller choice is reliant on the functional requirements of the IoT product performance, low power requirement, wireless integration, or robust security. Many IoT products in industries and health are much complicated and have additional computational power and energy limitations. Thus, IoT requires more study and standards to appraise the needs of the microcontroller. This section describes certain principles to determine which microcontroller is precise for the application needs.

A. MEMORY

The processing speed and performance depends on the memory sizes of the chosen microcontroller. The commercial microcontrollers have different memory sizes and usually have two memory components, such as RAM and ROM. The RAM holds the data access by the processor and performs read and write operations. The ROM saves the application code inside the microcontroller. The device cost factor becomes high when the size of the ROM is large. Few microcontrollers are designed with memory protection regions by imposing rights and access rules to address the memory locations. Table 6 list the memory specifications of various IoT devices.

B. SPEED

The speed of processing is essential for IoT products. However, IoT products require high-speed microcontrollers to execute demanding real-time tasks. The clock determines the speed of processing delivered by the specific device to achieve a higher data rate. The IoT device can perform functions like sensing or sending raw video to the storage entity by collecting data from one or more source after performing economic analysis. The microcontroller requires enough processing power to execute the tasks and functions [28].

C. INBUILT NETWORK INTERFACES

The network interfaces of microcontrollers will communicate with devices nearby and transfer the data to the IoT device for any computational analysis. The IoT devices connected through a wireless standard such as Wi-Fi mesh, Zigbee, Z-Wave, KNX, Thread, and Bluetooth for short-range communication and LPWAN, LoRA, 2G/3G/4G cellular network technologies for long-range communication [66]. These network interfaces are necessary to share the data to enable end-to-end networking, as illustrated in Fig. 4. The inexpensive link, low energy consumption, and cost are the foremost considerations for choosing the radio frequency (RF) technology. The Wi-Fi and Bluetooth standards utilized in IoT applications can operate under real-time conditions since Bluetooth enables point-to-point connections with smart devices like mobile phones, laptops to control the appliances in connected home applications. At the same time, Wi-Fi is most appropriate for bandwidth-demanding applications like wireless cameras. Ultra-low-power Sub GHz radio transceivers ICs

TABLE 5. Design space requirements of instruction set architectures.

Architecture		VLIW	MIPS	RISC	ARMv7	X86
Site Type	Available	Used	Used	Used	Used	Used
Slice Logic						
Slice LUTs	53200	54	931	903	3392	104
LUT as Logic	53200	54	931	903	3320	92
LUT as Memory	17400	0	0	0	72	24
Slice Registers	106400	68	1055	534	2231	75
Registers as Flip-flop	106400	22	1055	534	2231	75
Registers as Latch	106400	0	0	0	0	0
F7 Muxes	26600	0	288	5	54	0
F8 Muxes	13300	0	92	0	0	0
Memory						
Block RAM Tile	140	2	0	0	4	0
RAM B36/FIFO	140	0	0	0	0	0
RAM B18	280	4	0	0	8	0
DSP						
DSPs	220	2	0	1	3	0
IO						
Bonded IOB	200	11	198	4	152	105

TABLE 6. Memory specifications of IoT devices.

SOC	CACHE	RAM	STORAGE	OPERATING TEMPERATURE	OPERATING VOLTAGE
AllWinner R8	L1: 32KB I-Cache + 32KB, D-Cache, L2: 256KB Cache	512MB SDRAM	MicroSD	-20˚C to 70˚C	3.3V
Broadcom BCM 2837B0	L1: 16KB I-Cache + D-Cache, L2: 128KB	1GB LPDDR2 SDRAM	MicroSD	0 to 50˚C	5V
Sitara AM3358/9	L1: 32KB I-Cache + 32KB, D-Cache, L2: 256KB Cache	SDRAM 512MB DDR3	MicroSD, 4GB eMMC Flash	-40˚C to 105˚C	1.8V to 3.3V
Ingenic JZ4780	L1: 32KB I-Cache + 32KB D-Cache, L2: 512KB Cache	1GB DDR3	MicroSD, 8GB Flash	-40˚C to 125˚C	3.6V
Intel Quark/Atom MCUs	1MB Smart Cache	1GB DDR3	MicroSD, 4GB eMMC Flash	0 to 70˚C	3.6V to 5.25V
Intel Atom Broxton-M T5700	4MB Level-1	4GB DDR4	MicroSD, 16GB eMMC	0 to 70˚C	3.6V to 5.25V
STM32F205	No Cache - No MMU	128KB	1MB Flash	-40˚C to 85˚C	1.8V to 3.6V
ATmega32u4	None	64MB DDR2, 2.5KB SRAM	MicroSD, 16MB Flash	-40˚C to 85˚C	2.7V to 5.5V
ATmega1281	None	8KB SRAM	16GB, eMMC, 128KB Flash	-40˚C to 85˚C	2.7V to 5.5V
AllWinner A20	L1: 32KB I-Cache + 32KB	1GB DDR3	Up to 32GB SD	-20˚C to 70˚C	1.7V to 3.6V
Tensilica Xtensa L106	512KB Cache, 32KB I-Cache, 80KB D-Cache	SRAM up to 160KB	Slot, 4GB Flash Internal 4MB Flash, External Flash up to 16MB	-40˚C to 125˚C	2.5V to 3.6V

(e.g., ADF7024) are used to operate the devices for several years without changing the battery allows better transmission range and effectively penetrate through walls.

D. WAKE-UP TIME

The ultra-low-power smart home appliances like connected lights, intelligent refrigerators, and the camera usually spend most of the time in low power operational mode (sleep mode) to handle the task and rapidly return to the low power mode. Therefore, it is necessary to choose an MCU with an active wake-up time. The MCU cannot carry out any additional tasks during sleep mode, which results in energy saving, and therefore devices can attain the optimum low-power consumption.

E. COST AND MANUFACTURER SUPPORT

The microcontroller cost can differ for different use cases and impose licensing charges for individual device drivers. Detailed documentation supports the microcontroller users to make an appropriate conclusion on the features and specifications. The manufacturer allows professional support apart from standard support for platform development by enabling direct contact with the manufacturing organizations experts [67]. Furthermore, community support must know the real problems, implementation issues, error occurrences, and their solutions to transform the product [68].

F. COMMUNICATION PORTS

The ports are inbuilt into the microcontroller chip, or a separate I/O controller can be used. The I/O port in the microcontroller performs data transfer between the central processing unit and the peripheral devices. The I/O processor on an embedded board can have a series of components connected like LCD, LED, Motor, and CRT. The digital I/O can be programmed as an input or output port. Besides, analog ports used for instances such as speed, and temperature

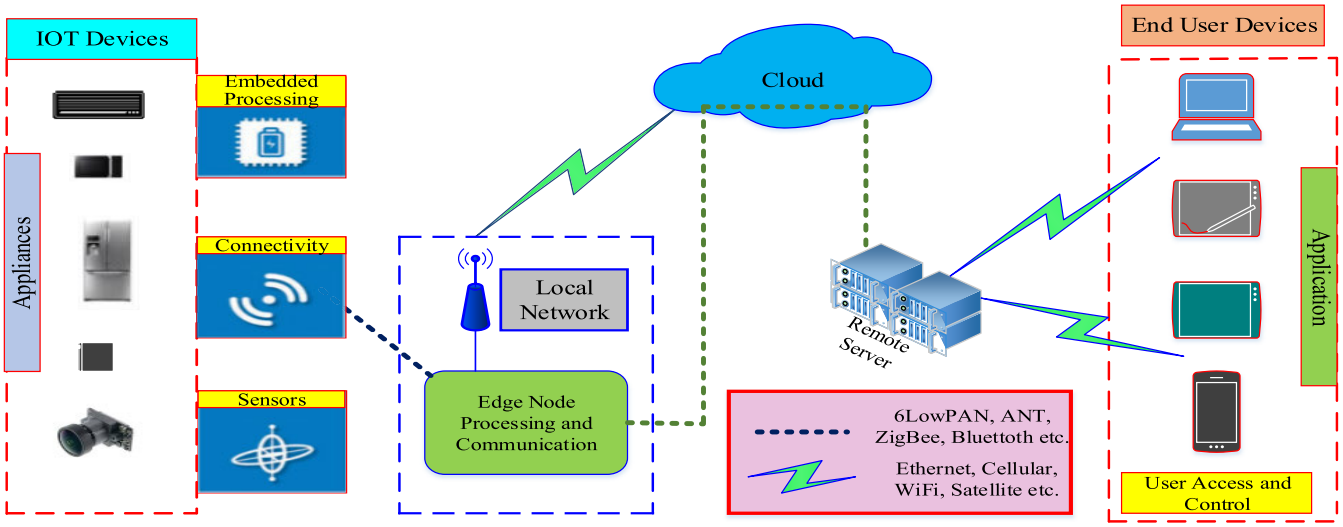


FIGURE 4. IoT end to end communication and networking.

measurement. The designer can select the microcontroller based on the type and specifications [69].

G. SECURITY AND STANDARDS

In the digital era, data security is the primary concern for microcontroller manufacturers and the end consumers working on the IoT domain [70]. Often chip manufacturers set up security measures to protect the system access from threat activities. Since several connected devices increases, vulnerability towards attacks may increases. In embedded design security procedures are essential through all layers similar to internal device storage, communication protocols, hardware interfaces, gateway implementation, and cloud computation. Security must be considered in the initial stage of the design by defining the security category that a specific product requires and examining the resources that necessitate protection. Thus, the designer will recognize the fundamental threat and security procedures that should be undergone to protect the design. Contingent on the IoT application needs, the security level can be examined. For instance, the application such as industrial control, nuclear engineering, medical devices, and traffic management requires a higher security level. However, smart home appliances such as a refrigerator, and the washing machine does not entail higher security measures. Additionally, designers are formulating more dedicated security chips that can be incorporated within the microcontroller to store user authorizations and encryption keys securely. Often attackers insert harmful code into the IoT microcontroller device; this situation leads to the worst consequences. To avoiding device attacking, few microcontrollers come with ingrained tamper detection. Usually, microcontroller chip manufacturers provide Crypto-Boot loader for security and firmware updates.

Ultimately, to prevent data leakages through the update process, the boot-loader encodes security keys in advance of the update. The encoding and decoding process utilizes internal microcontroller memory and clock cycles, lessens other events scheduled for execution. Some microcontrollers add accelerating algorithms in hardware to speed up the cryptography process. Conversely, Boot ROM is a write-protected flash set inside the microprocessor chip. It holds the initialization code, which is fetched and executed by the processor during power-on-reset. The standards developed by various industries, national and international bodies, are agreed upon by standard development organizations (SDOs), special interest groups (SIGs), and technology manufacturers. Table 7 shows the list of IoT and Machine to Machine (M2M) standards that are ideally suitable for IoT applications. The purpose of standards ensures interoperability and enables a cost-effective solution. The architectural framework described in the IEEE-P2413 standard will support cross-domain interactivity, assist system interoperability and operational compatibility, and further the IoT market [71]. IEEE Technology Report on Wake-Up Radio: An Application, Market, and Technology Impact Analysis of Low-Power/Low-Latency 802.11 Wireless LAN Interfaces describe how to attain low latency and low power at the same time. This report is beneficial for IoT device manufacturers and consumers to make superior business decisions [72].

H. ENERGY MANAGEMENT

At the outset, microcontrollers are essential to be energy efficient for IoT products. The wearable devices and other battery-operated devices encounter power and performance conditions [73]. The IoT based use-cases have been proposed for smart energy management to automate and analyze energy usage in different infrastructures [74]–[76]. The power management varies upon devices from the smart meter to laptops. In smart meters, the power-on hours (PoH) is always active.

While on laptops, power is not active all the time. It is essential to comprehend IoT products operating power modes to reduce power consumption peak hours. Consider the real-time control of edge node power consumption, during the non-peak hours; the power supply can be controlled remotely. The components like battery chargers and power supplies help cause standby and off-state energy consumption, influencing active usage of a products energy consumption. The active usage refers to which the appliance is carrying out its primary function.

In active-standby mode, the device can be ready for use, but it will not perform significant functions. In passive standby mode, the appliance can be either in standby state or off-state, but it seems off-state to the user. The appliance cannot activated by any means. Finally, off-mode points to which the appliance is turned off, and no function is currently being carried out. Active mode energy consumption ranges from 10 watts to more than 200 watts, which varies extensively by product category and accounts for roughly 60% of product power consumption. However, active-standby, inactive or passive standby, and off-state low power modes usually operate in the range of 1 watt to about 10 watts. However, few products can utilize more than 50 watts in low power mode [77]. During the sleep mode the power management chip is utilized that will be turned off to save the energy consumption. The modern chips are designed with dedicated power management features.

I. THERMAL MANAGEMENT

The IoT devices placed in a different environment should operate safely and reliably. The primary concern is that the process of designing IoT devices is thermal analysis. Integrating the device with modern technologies and processing power required to process the data may push towards the complexity of thermal issues. Therefore, as power density increases, it is necessary to impart cooling systems to manage IoT edge node devices in an operational state.

The Edge device can sustain for a wide-ranging environment temperature varying from $-30°C$ to $120°C$. The IoT device can also operate in the arduous enclosure and toughened to preserve the Edge Gateway in an unsecured indoor environment, outdoor applications, and industrial units. Thermal simulation can assist in developing a more complex solution and reducing the development time. In the future, the IoT will drive pioneering cooling solutions to come across varying thermal demands [78]. There has always been a trade-off between processor efficiency and thermal management. The processor consumes higher energy for higher performance, due to which the heat emission also increases as it operates in various switching frequencies. The IoT system-on-chip is to be designed so that its efficiency is not compromised even in extreme weather situations and temperatures. The processor's necessity is more crucial in specific applications where the latency to respond for an event would be in the range of micro-seconds. Conventional heat sinks made with high conductivity materials used to dissipate the heat generated from components like voltage regulators, audio amplifiers, etc. Placing a heat sink on the element alone does not clear the purpose. Alternative methods like thermal paste, thermal grease, or any thermally conductive sticky should create a reliable thermal connection between the heating and heat sink. Thermal management and safely shut down during out of the threshold temperatures will protect the SoCs.

J. COMPILER SUPPORT OPTIONS

The compiler selection should be explicit for different processors as it has to generate the machine programs during the code compilation process for a particular target board. Many compilers are available for both open source and paid versions in the market, including GNU C Compiler (GCC), ARM, Keil, and IAR. GCC is a well-known open-source compiler configured either native or cross compiler. In a cross compiler, the code will compile on the regular PCs. The generated object file get processed for linking to give the output as the executable that can run on the target processor boards such as ARM, AVR, Intel x86, etc. There is greater flexibility provided to the compiler through the practical usage of compiler options. Each compiler has got its possibilities to support the customized build environment. Through the compiler options, ground set rules made for the executable that will run on the specific target board. There are options supported by the compilers which provide the exhaustive configurable environment to the application and the target processor board.

K. SYSTEM-LEVEL PACKAGING

The packaging is yet another issue in IoT, where designers have integrated many blocks. After packaging done a single chip, then providing a solution becomes more challenging. IoT system-level packaging's standard requirements consist of fine RF shielding, power dissipation characteristics, low cost, and low power. The RF shielding in packages supports various RF standards like ZigBee, Wi-Fi, BLE, and others. Designing a small trace integrated with the product becomes essential rather than providing a new custom-built package for every IoT system. The other issues that arise in a packaged system are chip-package interaction, thermal stress, solder joint reliability, and more. To address these packaging complexities, the packaging experts and design engineers must involve in a combined system design integration process. Standardization for design and packaging will play a substantial role in achieving various design goals [79].

The smart object provides a massive amount of data; this data needs to be received, stored, and processed securely. The IoT offers computational support for many application fields, including intelligent transportation systems, utility, and health care services. The edge-connected system will provide the devices with adequate resources to carry out computations rather than performing through high-performance cloud devices. To appropriately distribute these devices, the application developer must initially understand the nature of applications executed on the IoT hardware device. Table 8

TABLE 7. IoT standards and application categorization.

Standards	Description
IEEE 1905.1™-2013	IEEE Standard for a Convergent Digital Home Network for Heterogeneous Technologies
IEEE 802.3™-2012	IEEE Standard for Ethernet
IEEE 1377™-2012	IEEE Standard for Utility Industry Metering Communication Protocol Application Layer (End Device Data Tables)
IEEE 21451-7™-2011	IEEE Standard for Smart Transducer Interface for Sensors and Actuators
IEEE 1900.4a™-2011	IEEE Standard for Architectural Building Blocks Enabling Network-Device Distributed Decision Making for Optimized Radio Resource Usage in Heterogeneous Wireless Access Networks
IEEE 1451.0™-2007	IEEE Standard for Smart Transducer Interface Standards - for connecting sensors or actuators to microprocessors, instrumentation systems, and control/field networks
IEEE P1828	IEEE Standard for Systems with Virtual Components
IEEE P1888.3™	IEEE Draft Standard for Ubiquitous Green Community Control Network: Security
IEEE 802.15.4t-2017	IEEE Standard for Low-Rate Wireless Networks
IEEE 802.11-2016	IEEE Standard for Information Technology - Local and metropolitan area networks
IEEE 802.16-2017	IEEE Standard for Air Interface for Broadband Wireless Access Systems
IEEE P1935	IEEE Draft Standard for Edge/Fog Manageability and Orchestration
IEEE P2301	IEEE Draft Guide for Cloud Portability and Interoperability Profiles(CPIP)
IEEE P2302	IEEE Draft Standard for Inter-cloud Interoperability and Federation (SIIF)
IEEE P2303	IEEE Draft Standard for Adaptive Management of Cloud Computing Environments
ERTICO	European Road Transport Telematics Implementation Coordination Organization
ETSI	European Telecommunications Standards Institute
One M2M	One Machine to Machine
GSMA	Global System for Mobile Communications
CEN	European Commission
CENELEC	European Committee for Electrotechnical Standardization
NLST	National Lung Screening Trial
UCA	Utilities Communication Architecture
3GPP	3^{rd} Generation Partnership Project
CONTINUA	Industry Standard for Connected Health Technologies
OMA	Open Mobile Alliance
IPSO	Internet Protocol for Smart Object Communications
OSGI	Open Services Gateway Initiative
IETF	Internet Engineering Task Force
W3C	World Wide Web Consortium

TABLE 8. IoT devices specifications and applications.

Devices	Specifications	Applications
Allwinner R8	SoC, CPU	IoT Module, Smart Devices, Gaming Peripherals, Audio Playback, and Video Boom box
Raspberry Pi 3 B+	GPU, Clock	Tablet computers, education, robotics, media streaming, gaming, home, industrial automation, commercial products
Beagle Bone Black	L1 Cache, L2 Cache	Robotics, media streaming, industrial IoT, home automation, wearable de vices, sensor platforms
MIPS Creator CI20	RAM, ROM	Smart home appliances, connected cameras, multimedia streaming, retail, remote device control
Intel Edison Compute Module	Price	Consumer electronics, IoT, wearable products, cyber-physical systems, Device to Device and Device to Cloud connectivity, Industrial IoT applications
Intel Joule Compute Module	Power Consumption	Computer vision, robotics, drones, industrial IoT, VR, AR, microservers, suitable for high-end edge computing
Particle Photon	Operating Temperature	To enable WiFi and Internet connectivity, Gaming, IoT, Industrial design, electronic and interactive prototyping
Arduino UNO R3	External Storage	Home automation, Game console, Drones, Robotics
Libelium Waspmote	Camera Support	Connected or smart home, wearable devices, retail, smart cities, healthcare, agriculture, automotive, industrial automation, energy management
Cubie Board	Media Streaming	Cluster management, Tablet, Smart TV, smart home, 3D printer, entertainment, education, personal storage cloud, Image processing
Node MCU ESP8266	On-Chip Networks	Wireless server, IoT, geolocation finding, Robotics, Industrial IoT monitoring applications

illustrates the combined specifications and applications of commercially available IoT devices. In the previous works, authors have proposed IoT applications through different components.

V. PROPOSED OPTIMIZATION ALGORITHM AND EXPERIMENTAL RESULTS

The genetic algorithm optimization function is the primary module that combines all other modules to perform

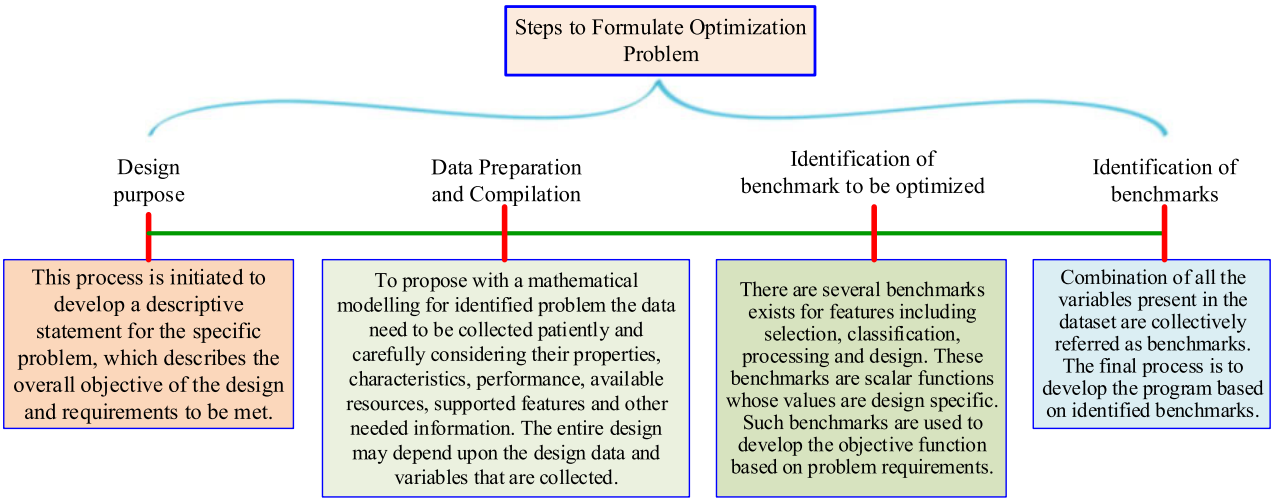


FIGURE 5. Optimization problem formulation.

optimization. This optimization function has many parameters that evaluate to find a specific object that contains complete information about the optimization. The core parameters chosen based on manual theoretical analyses and understanding of the SoC selection process handled by the academicians and organizations for application development. After analyses and design concepts are examined clearly, then the optimization problem is formulated according to the steps shown in Fig. 5. The steps to implement genetic algorithm-based optimum device selection are,

i) Chromosome and population: Initially, the algorithm developed with a limited number of chromosomes, usually referred to as a population. These chromosomes comprises of genes. The algorithm will calculate the fitness value for each chromosome. Following the design objective, the chromosome's fitness can check one by one by comparing the present chromosome's fitness value. Thus, one chromosome can yield better fitness results that update the desired vectors and its associated variable with the specific chromosome and its fitness value. The bit must be chosen from set of bit strings to execute genetic functions.

ii) Selection Operator: In the selection process, the main objective is to select two parents from the random population for mating to get the offsprings with the more excellent fitness value. This stage includes the selection of two or more parents from the available random population for mating. The selection process will filter individuals in the population to attain offsprings with higher fitness results. Each chromosome is counted relating to its fitness value to calculate the fitness selection. The best chromosomes have more fitness values.

The fitness value use to maximize the average system throughput in the selection constraint. The fitness value can represent the fitness value of the i^{th} chromosome.

This function will help identify the maximum value, where x takes the value from 1 to X.

$$F(i) = \max\left(\frac{1}{X}\sum_{x=1}^{X}\Delta^{x}\right) \tag{1}$$

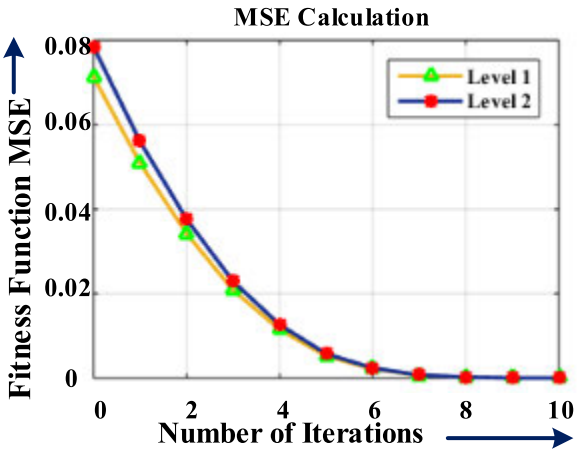


FIGURE 6. Fitness function versus number of iterations.

For the chromosome selection from the population, this paper applies the roulette wheel selection method. Relying upon the percentage of contribution to the available population fitness value is chosen for mating to form the next generation. Out of all the initial random population, two individual parents are selected based on the fitness in Pi, which can define by,

$$P_i = \frac{K_i}{\left(\sum_{K=1}^{n} K_j\right)} \tag{2}$$

where Pi is the probability of string I selected, n is the number of individuals in the population, and K_i is the fitness for series i in the available population.

iii) Crossover: The crossover operator can be either one point or two-point crossover. In the one-point crossover, the pair of selected strings cut at some random positions that are exchanged to form an updated pair of strings. In a two-point crossover, there are multiple breakpoints to arrange string pairs. The crossover will combine genes from various chromosomes. It is the concurrence of bit strings through reproducing the segments from chromosome pairs.

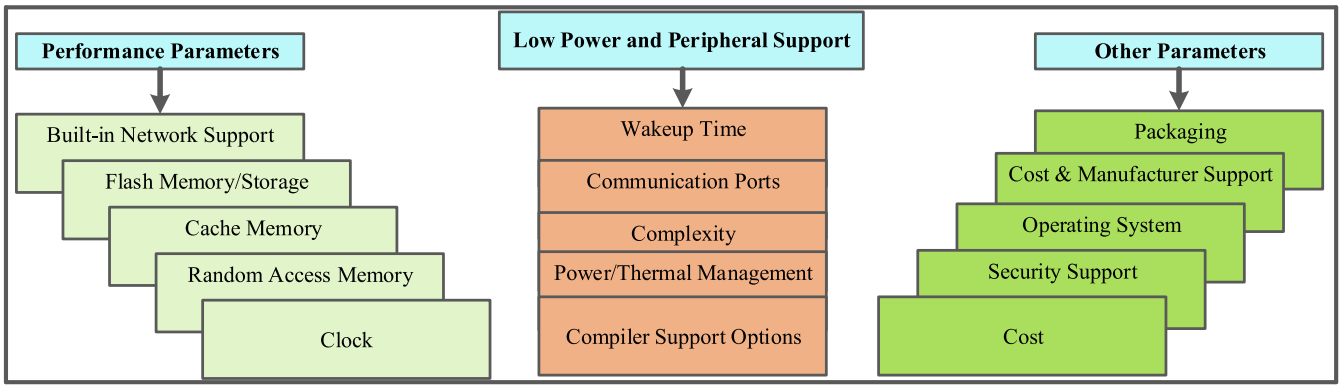


FIGURE 7. SoC device selection parameters.

iv) Mutation: The mutation is applied to every individual child after crossover, thereby bits are modified from 0 to 1 and 1 to 0 at randomly selected positions of selected string. The mutation produces a number randomly through the crossover position and then changes the gene value randomly. The selection, crossover, and mutation steps can repeated until the pre-determined number of generations is reached. The final optimal solution can be generated by the algorithm after the stop condition. The Fig. 6 shows that the proposed algorithm is superlative with greater fitness values due to different generations of populations. Thus, the value of the fitness function increases by increasing the number of iterations. Though there are many parameters to consider for the SoC selection, developers fail to view the real scenario's requirements. The incorrect selection of devices may push towards additional complexity in the system design or, unsupported features may cause the developers to find alternate solutions to test and implement the functionality. Moreover, the IoT system's reliability and the scalability of the system also be reduced. The SoC devices can be selected based on the following parameters, as shown in Fig. 7. The detailed explanation of the genetic algorithm parameters and the optimization algorithm discussed as follows.

A. REPRESENTATION OF CHROMOSOMES

The chromosome gives an overall solution to a particular problem. In this work, the chromosome significantly determine device performance and provide one accurate solution or device for a specific problem or application. The chromosomes of different population sizes generated using initial population. By assuming a device that supports five core parameters and may support for eleven applications then the chromosome representation is illustrated in Table 9.

B. CROSSOVER, MUTATION, AND SELECTION

There are totally 11 chromosomes present in which two parent chromosomes are combined and produce two child chromosomes. After determining uncrossed parent pairs, crossover and mutation is complete. The uniform crossover and mutation are carried out by which chromosome bits

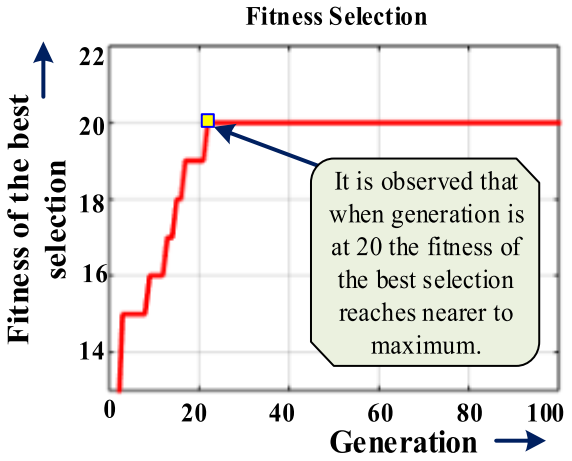


FIGURE 8. Fitness of the best selection.

changes for further improvement. The uniform crossover and mutation are selected from regenerated repositories of randomly selected binary values the crossover masks. The chromosomes are chosen based upon the fitness value from the population to be parents to crossover. The genes occupy a particular position from the chromosome of available population with a size considered by randomly generated binary value matrix. The fitness vector values are returned with a dimension of 1x population size to estimate the population fitness. The generation is considered as 20, and further these generations can be extended to improve the performance. The generation restricted to 20 to attain best selection. It is observed that when the generation is at 20, the best selection fitness reaches nearer to the maximum. Further, when a generation comes 100, the best selection shows just one point more than the fitness of the best selection shown in Fig. 8. The objective function written with two levels. The process is to select the IoT device that delivers maximum performance based on getting its input requirements. The level-2 function choose the MCU to target the maximum number of IoT applications. The level-1 Chromosomes produce the SoC

TABLE 9. Chromosome representation.

Parameters	Clock	Cache	RAM	Storage	Networks Support
Level-1 Chromosomes	1 GHz	64 KB	512 MB DDR3	SD Card	IEEE 802.11, Ethernet, USB, UART
Level-2 Chromosomes	IOT, Edge Computing	Industrial Automation	Smart Home, Media Streaming	Wearable Device, Commercial Products	Robotics, Drones, Gaming

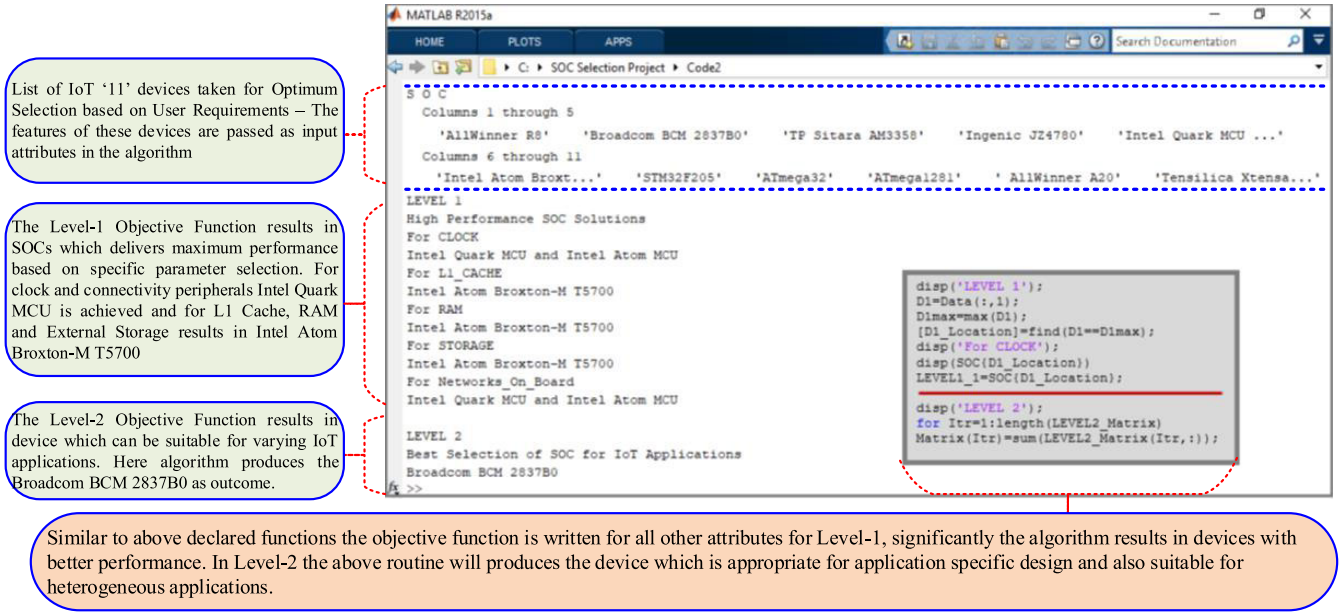


FIGURE 9. Objective function result for level-1 and Level-2 attributes.

solutions that offer high-performance devices appropriate for edge computing and high-end applications. The optimization algorithm output shows following the clock, L1-Cache, RAM, external storage, and on-board network requirements from the user, the algorithm produces two close matches, i.e., Intel Quark MCU and Intel Broxton- M T5700 MCU as a result. From this result, the user can select any one device that delivers maximum performance even after referring to the power and thermal requirements of the resultant devices. In level-2, the algorithm developed for the 'n' number of devices by loading the database in MATLAB R2015a. The devices supported by the applications are assigned binary input by referring to the specific device's datasheet. The level-2 algorithm uses the maximum input selection method, which can select a SoC device that supports a maximum number of applications from the database of commercially available devices. Fig. 9 illustrates the results of the objective function for both the level-1 and level-2 parameters.

The test data set contains 13 high-end devices, 13 medium and basic devices. After receiving the user's requirement, the algorithm starts predicting the best processors based on the level-1 and level-2 parameters. The experiment performed 26 times, and each processor tested to get the optimization algorithm's accuracy. The result yields 10 right positive results out of 13 high-end devices and 12 right negative results out of 13 virtual devices. Three high-end devices are categorized into necessary devices, and the algorithm produces one device as a high-end device. In the proposed work, a set of n performance parameters, $\mathrm{I} = I_1, I_2, ..., I_n$ preferred on various SoC devices. Where, I_1 clock, I_2 cache, I_3 RAM, I_4 storage, and I_5 networks on-board. The $J = J_1, J_2, J_3, \ldots, J_m$ is the various applications supported by the devices considered. Let $K = K_1, K_2, K_3, .., K_n$ be the SoC devices that bring higher performance and also used for IoT application development. Based on the user's input requirements such as the clock, L1 Cache, RAM, external storage, and on-board networks, the algorithm will be able to analyze and produce the close match device appropriate for the intended applications. The detailed information about the following devices such as All Winner R8, Broadcom BCM 2837B0, TP Sitara AM3358,

Ingenic JZ4780, Intel Quark MCU and Intel Atom MCU, Intel Atom Broxton MT5700, STM32F205, ATmega32, ATmega128P, All Winner A20, Tensilica Xtensa L106 are stored in the database and included in the primary function. During the run time, the output of the level-1 produces a better combination of SoCs for the desired user's input requirement. Similarly, the level-2 delivers the targeted SoC solution as a result of the precise IoT system development.

TABLE 10. Optimum SoC selection algorithm.

Genetic Algorithm based SOC selection for performance improvement and that specific SOC can also be used for heterogeneous IoT applications.
Input: (1) List of parameters (I_i) where $I_{1,2,3,4,5}$ denotes n^{th} SoC Clock, Cache, RAM, Storage, Networks on-board
Input: (2) A list of K SoCs supports applications J_m, (J_n — K_n). i.e., J_n and K_n represents the applications supported by specific device.
Output: An efficient mapping of SoC for attaining higher performance and utilized for different IoT applications.
1. Create initial population 2. Apply selection operation to determine number of chromosomes 3. Initialize gen=0; 4. while(gen=0) do 5. Select the parent chromosomes from available population randomly 6. Uniform crossover is applied to generate offspring's 7. if (Best Fitness > maxFitnessHist(gen+1)) then 8. [maxFitnessHist(1,gen+1),maxIndex] = max(fitnessVals); 9. avgFitnessHist(1,gen+1) = mean(fitnessVals) 10. else 11. BestFitness = maxFitnessHist(gen+1) 12. BestIndiv = pop(maxIndex,:); 13. end if 14. gen++; 15. end while 16. Select the maximum fitness value as final solution
Result with Finest Solution

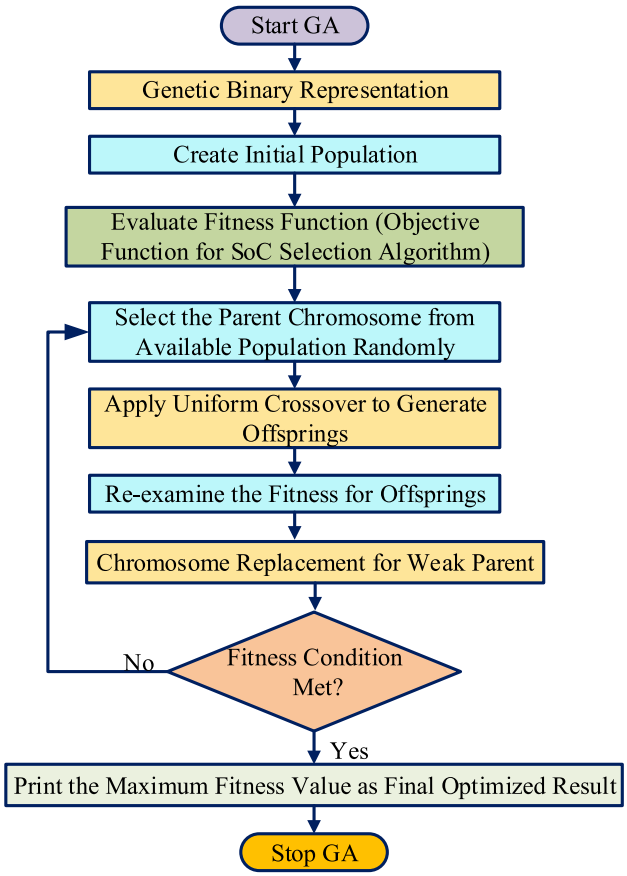


FIGURE 10. Operation flow of genetic algorithm for SoC selection.

The Table 10 illustrates the algorithm for the selection of a processor using a genetic algorithm approach. The Fig. 10. Illustrates the genetic algorithm functional flow diagram for SoC selection. In the algorithm, the parameters are chosen based on analyzing the interior features supported by individual SoC devices. The problem addressed using the linear programming method. Let α_{ij} be a Boolean value is shown as follows:

$$\alpha_{ij} = \begin{cases} 1, & \text{if } I_i \text{ present in } k_i \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Fig. 11 (a) and Fig. 11 (b) show the histogram plot for level-1 and level-2 attributes. Out of 5 attributes RAM reaches the maximum weight factor followed by L1 cache. Here in level-1, consists of five attributes and with level-2 offers the range of level-1 attributes associated to Broadcom BCM 2837B0. For future IoT applications, the weight factor of the SoC needs to be improved. Out of 5 attributes RAM reaches the maximum weight factor followed by L1 cache. Further, improving the external storage ability processor can deliver the ultimate performance. The crossover and mutation process reached the leading weight factor during the various generations. In the Level-1 attributes comparison, from the dataset shown in Table 11, we have taken samples of data to describe the priority-based weightage of the five different SoC from the core parameters. The x-axis represents the parameters, and the y-axis represents the SoC priority weightage of parameters. For instance, consider the clock parameter black color represents the clock weightage of Intel Quark MCU. It operates with 2.13 GHz clock followed by dark grey color representing Intel Atom Broxton M-T5700 MCU that supports internal clock frequency of 1.7 GHz and so on. Similarly, for the remaining core parameters, the weightage is described in the plot. The applications supported by each device shown in Table 12.

Through this priority based weightage assignment among the several SoC, we can quickly determine the overall parameter's weightage, which significantly supports better SoC selection. The Level-1 attributes can support for ultra-low

TABLE 11. Dataset for high-performance device selection.

SOC	CLOCK	CACHE	RAM	Flash	Networks On-Board
AllWinner R8	1 GHz	L1: 64KB Cache	512 MB SDRAM	Micro SD	None
Broadcom BCM 2837B0	1.4 GHz	L1: 16KB Cache	1 GB LPDDR2 SDRAM	Micro SD	IEEE 802.11.b/g/n/ac WLAN, Bluetooth 4.2, Ethernet over USB 2.0 (Max. throughput = 300 Mbps)
Sitara AM3358/9	1 GHz	L1: 64KB Cache	SDRAM 512 MB	4 GB eMMC Flash	10/100 Mbps Ethernet, USB
Ingenic JZ4780	1.2 GHz	L1: 64KB Cache	1 GB DDR3	8 GB Flash	1x 10/100Mbps using Davicom DM9000C controller , 80.11 b/g/n, Bluetooth 4.0
Intel Quark and Intel Atom	2.13 GHz	1 MB Smart Cache	1 GB DDR3	4 GB eMMC Flash	Broadcom 43340, IEEE 802.11 a/b/g/n, Bluetooth 4.0
Intel Atom Broxton-MT5700	1.7 GHz	L1: 4 MB	4 GB DDR4	16 GB eMMC	IEEE 802.11 ac w/MIMO, Bluetooth 4.2
AllWinner A20	1 GHz	L1: 64KB Cache	1 GB DDR3	4 GB Flash	Ethernet 10/100/1000 Mbps, Serial UART, SATA
ATmega1281 @ 14.74 MHz	14.74 MHz	None	8 KB SRAM	16 GB eMMC	RF (Xbee)-802.15.4 to 4G/GPRS/GSM router for Waspmote nodes, WiFi, ZigBee-Pro
STM32F205	120 MHz	No Cache - No MMU	128 KB	1 MB Flash	IEEE 802.11 b/g/n, On-Board Broadcom 43362 WiFi Module
ATmega32u4 @ 16 MHz	400 MHz	None	64 MB DDR2, 2.5 KB SRAM	Micro SD, 16 MB Flash	IEEE 802.3, 10/100 Mbps Ethernet, WiFi, USB 2.0
Tensilica Xtensa L106	80 to 160 MHz	L1: 32KB Cache	SRAM upto 160 KB	Internal 4 MB Flash , External Flash 16 MB	IEEE 802.11 b/g/n

TABLE 12. Dataset of devices that supports various applications.

SOC	IoT	High-end Edge Computing	Smart Home	Industrial Automation	Robotics	Wearable Devices	Drones	Gaming	Media Streaming	Commercial Products
AllWinner R8	✓	×	✓	×	×	×	×	✓	✓	×
Broadcom BCM 2837B0	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
Sitara AM3358/9	✓	×	✓	×	✓	✓	×	×	✓	×
Ingenic JZ4780	✓	×	✓	×	×	×	×	×	✓	✓
Intel Quark MCU @ 100 MHz and Intel Atom MCU	✓	✓	×	✓	×	✓	×	×	×	×
Intel Atom Broxton-M T5700	✓	✓	×	×	✓	×	✓	×	✓	×
AllWinner A20	✓	✓	✓	×	×	×	×	×	✓	✓
ATmega1281 @ 14.74 MHz	✓	✓	✓	✓	×	✓	×	×	×	✓
STM32F205	✓	×	✓	✓	×	×	×	×	×	×
ATmega32u4 @ 16 MHz	✓	×	×	✓	✓	×	✓	✓	×	×
Tensilica Xtensa L106	✓	✓	✓	✓	✓	×	×	×	×	×

✓Applications Supported ×Applications Not Supported

power and industrial applications device selection. The Level-2 attributes comparison describes the SoC appropriate for heterogeneous application development. Here, the proposed algorithm resulted in Broadcom SoC as the superior SoC, followed by dark grey color, represents the Intel Broxton M-T5700 SoC. The Fig. 12 shows the performance comparison between the probability of detection and throughput. Here the capacity of GA is considered as PoD, and efficiency is considered as throughput. However, PoD increases the efficiency of GA increases.

In both levels the performance attained nearer to PoD during complete measurements. Here two levels are considered for examining performance parameters and SoC selection for various IoT applications in which both the levels reach maximum efficiency. The number of attributes in level-1 as five and level-2 as 11 is static, though upon dynamic attributes consideration, the proposed algorithm can attain the maximum efficiency. The data obtained and used in the algorithm is static and expanded based on dataset size.

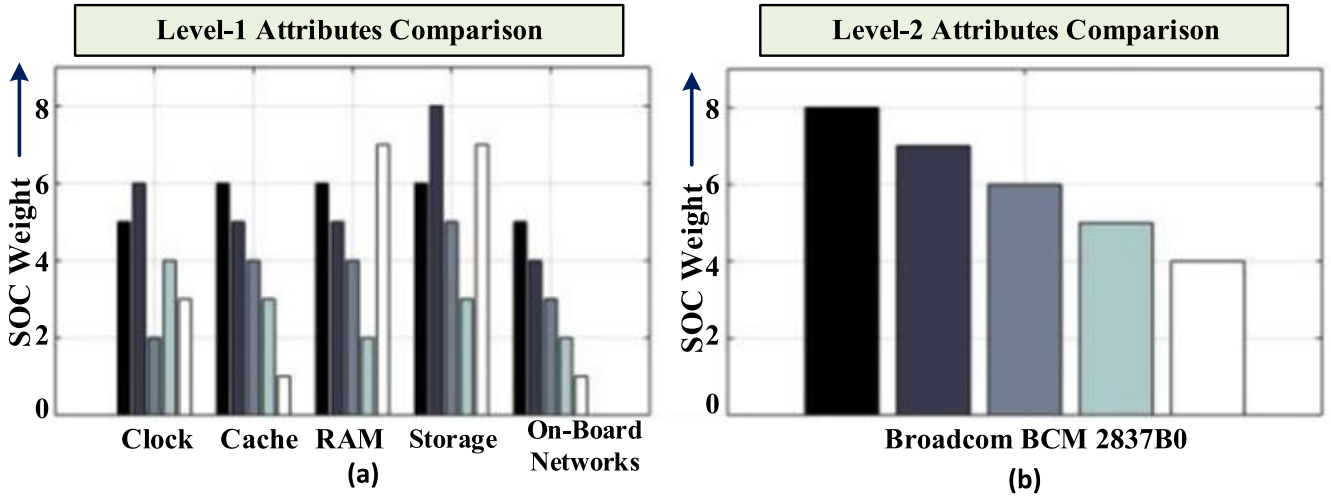


FIGURE 11. (a) Histogram plot for level-1 attributes and (b) Histogram plot for level-2 attributes.

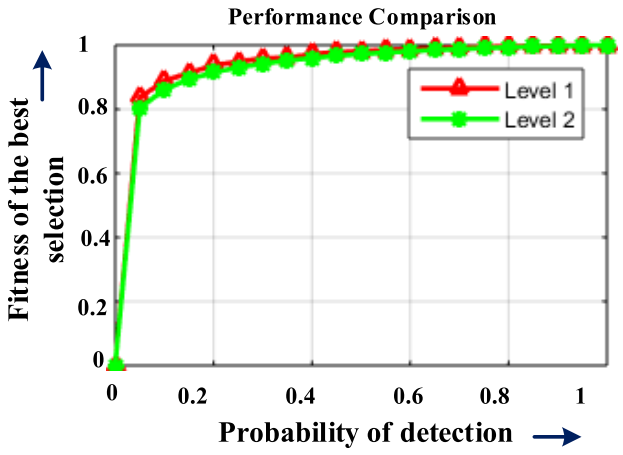


FIGURE 12. Probability of detection and throughput performance.

C. ACCURACY

We estimate the accuracy of the proposed algorithm using confusion matrix method. This method implied in the predictive analysis methodology to compute the efficiency of the model obtained to that of the actual model. Although the genetic algorithm predicts the best processor, the algorithm's output will not be the same for different data sets. The data set used for computing the model's accuracy divided into two classes, namely high-end devices and low-end devices. These devices are separated equally to support the predefined categories. The data set used for testing contains 26 devices along with their level-1, and level-2 parameters are inbound. A confusion matrix is formed using four output parameters. While an event occurs, the total probability of the event occurs is denoted by 2^n, where n is the number of times the event occurs. The output produced by the algorithm can be classified into a true positive, real negative, false positive, false negative.

TABLE 13. Confusion matrix table.

N=26	Basic Devices	Advanced Devices
Low-end Devices	12	1
High-end Devices	3	10

A true positive (TP) device is an outcome where the confusion matrix model correctly predicts the positive devices. The true positive devices belong to the category of high-end devices. Similarly, true negative (TN) devices are low-end devices, which are resulted when the model predicts the negative devices. A false positive (FP) is an outcome where the model incorrectly predicts the positive devices. And a false negative (FN) is an outcome where the model falsely predicts the negative devices.

Table 13 illustrates the confusion matrix table in which the devices assign with Boolean representation. For high-end devices, the Boolean value allocated as 1, and for basic devices, the value is 0. Two arrays for made computation of accuracy and the results are obtained. Accuracy of the algorithm is calculated as,

$$A = (TP + TN)/(TP + TN + FP + FN) \tag{4}$$

where A is the accuracy of the algorithm, TP is the number of actual positive outcomes, TN is the number of actual adverse outcomes, and N is the total number of times the task executes. The estimated algorithm accuracy using confusion matrix method in MATLAB.

The accuracy estimation algorithm also carried out using Watson studio and the python programming language. The input devices database loaded in CSV file format, from which the program fetches, and a multi-class classification is carried out. After the initial classification of the data, the user

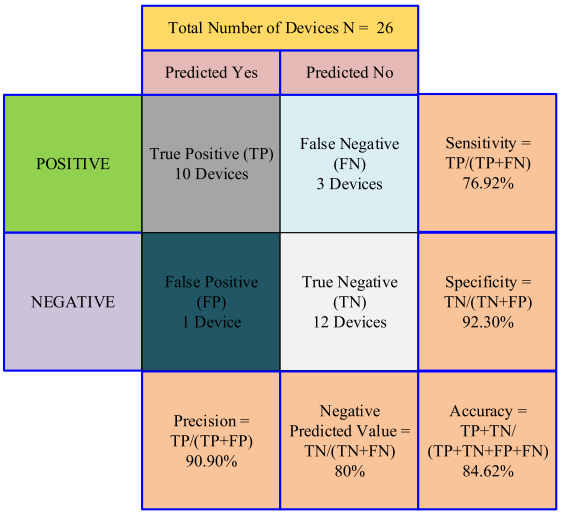


FIGURE 13. Estimated algorithm accuracy.

provides input, and the algorithm produces the result. These combined outputs of classified elements are correlated, and the accuracy of the algorithm is determined.

The classification metrics based on the confusion matrix is discussed below:

- These mathematically expression for these metrics are shown in Fig.13. It should be higher always. For instance, the proportion of devices foreseen among all predicted devices.

The Classification problem has predicted and non-predicted classes, and the dataset contains 26 examples, 13 predicted devices, and 13 are non-predicted by the algorithm.

Sensitivity is a measure of positive devices labeled as positive by the classifier and referred to as true positive rate. The 76.92% of devices are correctly classified and excluded from all non-predicted devices.

$$\text{Sensitivity} = \text{TP}/(\text{TP} + \text{FN}) = 10/(10 + 3) = 76.92\%$$

Specificity is a measure of negative devices labeled as negative by the classifier, referred as a true negative rate. There should be high specificity. For example, the proportion of devices that are non-predicted among all non-predicted devices.

$$\text{Specificity} = \text{TN}/(\text{TN} + \text{FP}) = 12/(12 + 1) = 92.30\%$$

The 92.30% non-predicted devices are accurately classified and excluded from all predicted devices.

Precision is the total number of correctly classified devices and the total number of predicted devices. It shows correctness achieved in positive prediction.

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) = 10/(10 + 1) = 90.90\%$$

The 90.90% of devices are classified as actually predicted devices.

Accuracy is the proportion of the total number of correct predictions.

$$\begin{aligned}\text{Accuracy} &= (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \\ &= (12 + 10)/(12 + 10 + 1 + 3) = 84.62\%.\end{aligned}$$

The classifier properly categorizes the 84.62% of devices.

In the existing works, fuzzy-based approach was used to improve accuracy, but the complexity is very high. Upon comparing the fuzzy-based approach, we evaluated the proposed algorithm accuracy in the confusion matrix method using MATLAB and Python. This method is implied in a predictive analysis methodology to compute the model's efficiency obtained to that of the actual model. Since the genetic algorithm predicts the best processor, the algorithm's output will not be the same for different data sets. The data set used for computing the accuracy of the model remains for high-end devices and low-end devices. These devices are divided equally to accommodate the predefined classes. Upon evaluating the confusion matrix using the python programming model. we achieve the same accuracy of 84.6% shown in Fig. 14. The Table 14, the various benchmark designs including 8-point and 64-point fast Fourier transform for communication applications.

The 512-bit dense matrix multiplication for image processing applications, data encryption standard, advanced encryption standard for security applications, and FIR filter for signal processing applications are implemented and computed for execution time and performance using the above device configurations. Table 15 shows the execution time and performance for benchmarks using resultant devices. Every benchmark is experimented with using Intel Atom and ARM instruction set architecture (ISA). The three devices *Intel Atom Tangier-Z3480 (Device1), Intel Atom Broxton M-T5700 (Device2), and ARM Cortex A53 quad core (Device3)* are adaptable for heterogeneous applications.

As shown in the Fig. 15, it is observed that, though the algorithm resulted with *Device1* and *Device2* configurations which offers improved performance. The *Device3* that resulted from supporting maximum number of applications and established ARM Cortex A53 (Broadcom BCM 2837B0 SoC). This SoC is chosen as base configuration and we compared the execution time and performance results for all the benchmarks. Each device is operated based on their internal clock configuration, and the corresponding execution time and performance. Though the selected devices can support multicore operations, authors have not discussed the implications of parallel processing with the chosen devices in this work.The resultant devices are high-end; they have more RAM capacity to process the benchmark codes. The benchmarks verification for analyzing the runtime and performance carried out using Keil ARM CC compiler for ARM Cortex A53 SoC and Intel C Compiler (ICC) for Intel SoCs.

As shown in Table 16, the 8-bit, 64-bit FFTs, matrix multiplication, and cryptocores are downloaded into the chip using microcontroller burner during the test phases. It is chosen

TABLE 14. Benchmarks and its applications.

Benchmarks	Description	Benchmark Application
8_Point FFT	small Fast Fourier Transform	Communication
64_point FFT	large Fast Fourier Transform	Communication
512_Matrix Multiplication	dense matrix multiplication	Image processing
DES	data encryption standard	Security
256_AES	advanced encryption standard	Security

TABLE 15. Execution time and performance analysis for various benchmarks.

Benchmarks	Device 1 (Intel Atom Tangier - Z3480)		Device 2 (Intel Atom Broxton M-T5700)		Device 3 (ARM Cortex A53 Quad Core)		Average	
	Execution Time (S)	Performance	Execution Time (S)	Performance	Execution Time (S)	Performance	Execution Time (S)	Performance
8_Point FFT	0.6	0.45	0.5	0.74	0.4	0.96	0.5	0.71
64_Point FFT	1.06	0.64	1.04	1	1.023	1.32	1.04	0.98
512_Matrix Multiplication	1.02	0.05	0.96	0.83	0.86	1.12	0.94	0.66
DES	0.83	0.26	0.8	0.56	0.77	0.99	0.8	0.6
AES	0.97	0.32	0.89	0.62	0.85	0.86	0.9	0.6

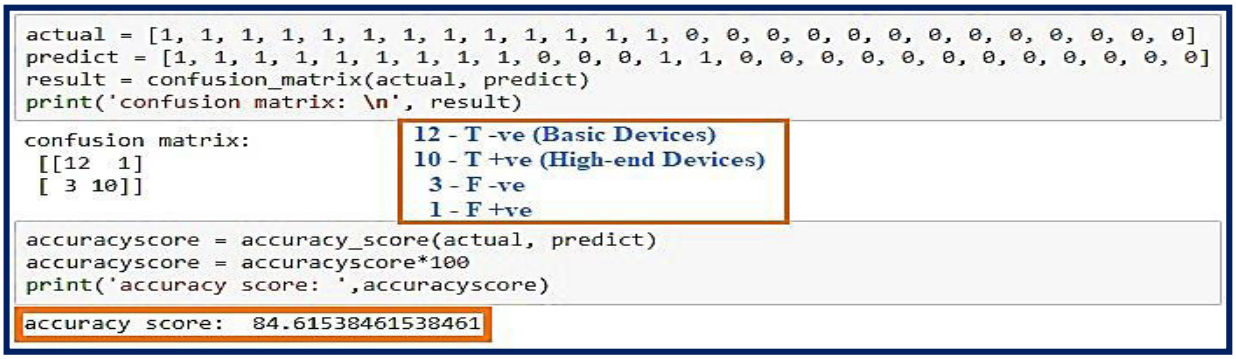


FIGURE 14. Accuracy estimation using python programming.

TABLE 16. Benchmarks and device testing for performance evaluation.

Devices	Bench Marks	Code Size	RAM	Flash	Cache	Clock
Intel Atom Tangier - Z3480	8-Point FFT	26.9 KB	4 GB	4 GB	1 MB	2.13 GHz
	64-Point FFT	86 KB				
Intel Atom Broxton M-T5700	Matrix Multiplication	213 KB	1 GB	16 GB	4 MB	1.7 GHz
ARM Cortex A53 Quad Core	AES	181 KB	1 GB	4 GB	16 KB	1.4 GHz

and classified as the benchmarks for signal processing and security applications. Also, benchmarks are downloaded in selected IoT devices that can perform computation such as, image, audio, and video compression. Due to the cross compiler's support and to save MIPS, the benchmarks are written in C language and tested using the resultant devices.

Relating to the performance and execution time measurement for the spectrum magnitude finding application is considered. The input voltage spectrum for 8-point and 64-point FFTs are got through the micro-controller device. For analysis, the spectrum magnitude and displaying the results in the PC are processed by the MCU. The algorithm written required 2N 16-bit variables for processing 16-bit values. The SoCs used for software implementation are advanced devices with large internal resources; however, they provide limited security support. Similarly, the matrix multiplication operation is implemented in these devices and the time taken for execution in each device is noted.

From the experiment carried out, it is observed that performing all the benchmark is depending upon the cache memory rather than the clock. The performance of the CPU is calculated based on the total number of instructions or

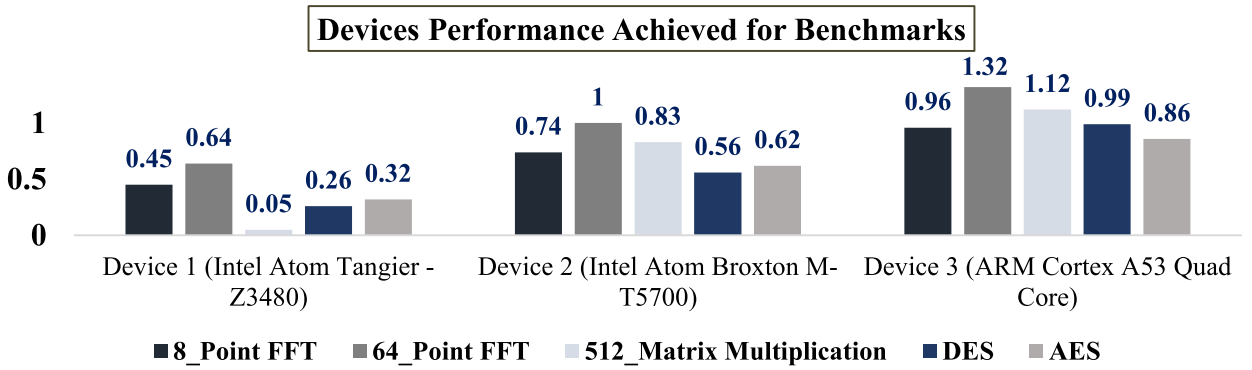


FIGURE 15. Performance comparison of various benchmarks.

instruction count, clocks per instruction, and clock time. For the security benchmarks, the subsequent devices are utilized and loaded the benchmarks to verify the cipher (encrypted) and decipher (decrypted) results. The algorithm uses the symmetric key, where the key size is chosen with 128-bit. After assigning the key size, the input for the encryption sequence (plain text) value is entered to produce the encrypted message. The same encrypted output is passed as an input to the decryption algorithm and entailed the plain text result achieved in the decryption algorithm. The same procedure is followed for both AES and DES benchmarks. These test cases cannot be tested in low-end microcontrollers where the internal RAM size is very less. The low-end microcontrollers they still require code optimization to be carried out from developer such that to decrease the instructions run the application. These SoCs have high MIPS range and support code optimization techniques to further reduce the number of instructions to execute the application. The compilers supported for these devices support various levels of code and compiler optimizations such as function inlining and multifile compilation. Through reducing the frequency of memory access by storing the data in the cache will increase the device performance. Besides, either by using the cache memory or performing the computations directly on the chip memory without the need for processor will increase the performance. Therefore, effective utilization of cache memory or in-memory processing significantly produces an adequate optimization for SoC devices.

From the result attained shown in Fig. 16, it is possible to understand that, the Device3 outperforms by achieving the least execution time and increased performance compared to Device1 and Device2 configurations. From the three different SoC device configurations, this analysis will provide the support for the better microprocessor architecture configuration. Though the proposed GA objective function is useful for finding the optimal solution, it becomes most prohibitive for complex multi-objective functions. In this work, we have multiple parameters that need to be analyzed based on which the algorithm produces an SoC outcome. This can be done in two different approaches; the first method is that the user can feed the input parameters to the variables declared inside the algorithm. The individual variables values gets compared with the entire parameter matrix, and the algorithm produces best possible outcome. The second method is that the fitness function developed to find the maximum value for every parameter. In the column matrix thereby input passed by the user for specific core parameters based on which the algorithm produces the SoC outcome.

- One of the foremost limitations in both the approach is that the algorithm will not end up with one common solution. It may results maximum of two or three solutions. Therefore, the user must go for validating approaches to justify the results.
- Another constraint is that when the number of parameters exposed to mutation is more, the size of the search space will be increased.
- The other limitation that we experienced is, GA produces optimal solution measures. If the scenario allows for success or failure test is repeatedly producing different results, then the success to failure ratio gives an appropriate suitable solution.

VI. FUTURE TRENDS AND RESEARCH DIRECTIONS

The quantity of data processed or stored expected to grow 3.42 ZB in 2025, which may upsurge the utilization of public cloud storage. Since massive IoT devices are deployed to handle various applications, Analytics of Things (AoT) need to be carried out by the devices, appliances, devices, hardware, and software systems interconnected in a top-notch fashion. The technologies such as cloud, fog, and embedded computing operate by developing IoT applications. These technologies create ample opportunities in the IoT domain, where developers are not only able to handle the embedded hardware. Besides, developers should be aware of recent tools for cloud development and the ability to develop high-level frameworks.

The machine learning and artificial intelligence make the devices conduct self-learning and bring a substantial rise in productivity to achieve business goals. The rapid adoption of IoT in the business landscape poses significant challenges, including new technology standards and interoperability factors. Influencing IT Architecture and Systems to pull the

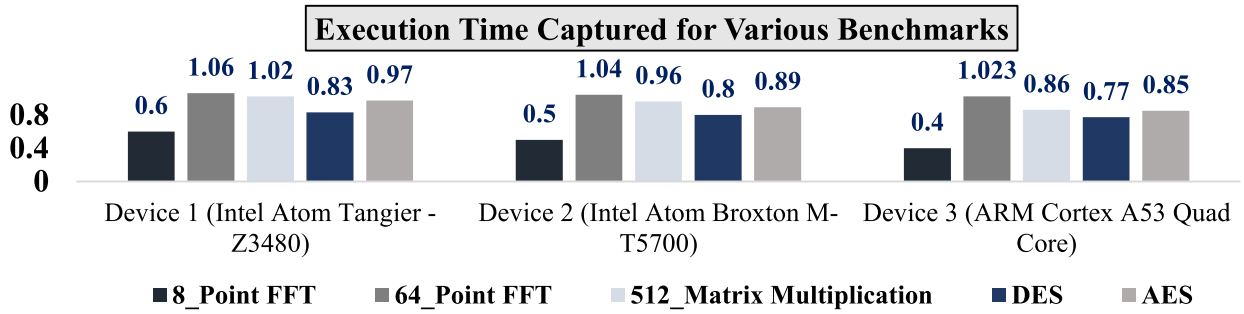


FIGURE 16. Execution time comparison between various benchmarks.

data and developing cybersecurity standards to avoid data privacy threats. The future IoT devices must have high speed and optimal decision-making ability at the edge node. The advanced devices enable efficient product management facilities to achieve a better quality of service (QoS), quality of monitoring (QoM), storage, and processing capabilities. The entire IoT ecosphere requires information and communication security functionality to be enabled among all the IoT reference architecture layers. For speeding up data created by cloud servers, server farms in organizations, or at the edge node server, priority must be assigned to achieve higher throughput. Future IoT-based research must focus on enabling multidisciplinary design, big data processing and analysis, strengthening industrial engineering solutions, infrastructure planning and management, smart energy management, legal aspects, innovative protocol development, political and business support. Through several IoT committees' collaborations actions, local and international bodies have a primary role in accomplishing the above research challenges.

VII. CONCLUSION

In this paper, a systematic approach to determine the technical requirements of IoT devices for different research communities is addressed. To validate design space requirements of state-of-the-art processor core architectures and their corresponding CPU cores are implemented in Xilinx Zynq-7000 (xc7Zz20clg484-1) FPGA device. The attained results disclose that ARMv7 and MIPS core utilizes more design space followed by RISC, X86, and VLIW architectures. The device attained through the GA outcome is wholly based on the input requirements from the user and the optimized simulation results of an algorithm. Based on the obtained result it is concluded that three devices, namely Intel Atom Tangier-Z3480 (Device1), Intel Atom Broxton M-T5700 (Device2), and ARM Cortex A53 Quad Core (Device3) meets the user's input requirements. The GA results are evaluated by implementing several benchmarks in the above resultant IoT devices. From the hardware implementation, the algorithm resulted that for high-performance ARM Cortex A53 the device can be utilized. Also, the algorithm results with Intel Atom Tangier-Z3480 the device that executes the benchmarks with the least execution time. In the future, the implementation of these algorithms reduces computation complexity to select optimal IoT devices for specific applications. By choosing an accurate IoT devices, it is possible to ensure that current cloud infrastructure can be made simple.

REFERENCES

[1] W. A. Jabbar, T. K. Kian, R. M. Ramli, S. N. Zubir, N. S. M. Zamrizaman, M. Balfaqih, V. Shepelev, and S. Alharbi, "Design and fabrication of smart home with Internet of Things enabled automation system," *IEEE Access*, vol. 7, pp. 144059–144074, 2019, doi: 10.1109/access.2019.2942846.

[2] A. Ahad, M. Tahir, and K.-L.-A. Yau, "5G-based smart healthcare network: Architecture, taxonomy, challenges and future research directions," *IEEE Access*, vol. 7, pp. 100747–100762, 2019.

[3] B. V. Philip, T. Alpcan, J. Jin, and M. Palaniswami, "Distributed real-time IoT for autonomous vehicles," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1131–1140, Feb. 2019.

[4] K. Ashton, "That 'Internet of Things' thing," *RFID J.*, vol. 22, no. 7, pp. 97–114, Jun. 2009.

[5] *Internet of Things (IoT) Connected Devices Installed Base World-Wide From 2015 to 2025 (in Billions)*, Statista, New York, NY, USA, 2018.

[6] A. Hafid, S. Benouar, M. Kedir-Talha, F. Abtahi, M. Attari, and F. Seoane, "Full impedance cardiography measurement device using raspberry PI3 and system-on-chip biomedical instrumentation solutions," *IEEE J. Biomed. Health Informat.*, vol. 22, no. 6, pp. 1883–1894, Nov. 2018.

[7] A. Villar-Martinez, L. Rodriguez-Gil, I. Angulo, P. Orduna, J. Garcia-Zubia, and D. Lopez-De-Ipina, "Improving the scalability and replicability of embedded systems remote laboratories through a cost-effective architecture," *IEEE Access*, vol. 7, pp. 164164–164185, 2019, doi: 10.1109/ACCESS.2019.2952321.

[8] M. Kim and Y. S. Shao, "Hardware acceleration," *IEEE Micro*, vol. 38, no. 6, pp. 6–7, Nov./Dec. 2018.

[9] M. S. BenSaleh, S. M. Qasim, A. A. AlJuffri, and A. M. Oheid, "Design of an advanced system-on-chip architecture for Internet-enabled smart mobile devices," in *Proc. 30th Int. Conf. Microelectron. (ICM)*, Dec. 2018, pp. 323–326, doi: 10.1109/ICM.2018.8704077.

[10] G. Schiele, A. Burger, and C. Cichiwskyj, "The elastic node: An experimentation platform for hardware accelerator research in the Internet of Things," in *Proc. IEEE Int. Conf. Autonomic Comput. (ICAC)*, Jun. 2019, pp. 84–94, doi: 10.1109/ICAC.2019.00020.

[11] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin, "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2018, doi: 10.1109/ACCESS.2017.2780087.

[12] M. Gusev and S. Dustdar, "Going back to the roots—The evolution of edge computing, an IoT perspective," *IEEE Internet Comput.*, vol. 22, no. 2, pp. 5–15, Mar./Apr. 2018, doi: 10.1109/MIC.2018.022021657.

[13] C.-H. Chen, M.-Y. Lin, and C.-C. Liu, "Edge computing gateway of the industrial Internet of Things using multiple collaborative microcontrollers," *IEEE Netw.*, vol. 32, no. 1, pp. 24–32, Jan. 2018, doi: 10.1109/MNET.2018.1700146.

[14] N. M. Qui, C. H. Lin, and P. Chen, "Design and implementation of a 256-bit RISC-V-Based dynamically scheduled very long instruction word on FPGA," *IEEE Access*, vol. 8, pp. 172996–173007, 2020, doi: 10.1109/ACCESS.2020.3024851.

[15] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018, doi: 10.1109/JIOT.2018.2805263.

[16] T. Gomes, F. Salgado, S. Pinto, J. Cabral, and A. Tavares, "A 6LoWPAN accelerator for Internet of Things endpoint devices," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 371–377, Feb. 2018, doi: 10.1109/JIOT.2017.2785659.

[17] W. C. Cave, R. E. Wassmer, H. F. Ledgard, A. B. Salisbury, K. T. Irvine, and M. A. Mulshine, "A new approach to parallel processing," *IEEE Access*, vol. 8, pp. 30287–30305, 2020, doi: 10.1109/ACCESS.2020.2972204.

[18] A. Gebregiorgis and M. B. Tahoori, "Fine-grained energy-constrained microprocessor pipeline design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 457–469, Mar. 2018, doi: 10.1109/TVLSI.2017.2767543.

[19] M. Payami, E. Azarkhish, I. Loi, and L. Benini, "A hybrid instruction prefetching mechanism for ultra low-power multicore clusters," *IEEE Embedded Syst. Lett.*, vol. 9, no. 4, pp. 125–128, Dec. 2017.

[20] T. Strauch, "Connecting things to the IoT by using virtual peripherals on a dynamically multithreaded cortex m3," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2462–2469, Sep. 2017.

[21] T. Strauch, "The effects of system hyper pipelining on three computational benchmarks using FPGAs," in *Applied Reconfigurable Computing* (Lecture Notes in Computer Science), vol. 9040. Cham, Switzerland: Springer, Mar. 2015, pp. 280–290.

[22] S. N. Swamy and S. R. Kota, "An empirical study on system level aspects of Internet of Things (IoT)," *IEEE Access*, vol. 8, pp. 188082–188134, 2020, doi: 10.1109/ACCESS.2020.3029847.

[23] H. Q. Al-Shammari, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Service embedding in IoT networks," *IEEE Access*, vol. 8, pp. 2948–2962, 2020, doi: 10.1109/ACCESS.2019.2962271.

[24] P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 30, no. 3, pp. 291–319, 2018.

[25] L. Linguaglossa, S. Lange, S. Pontarelli, G. Retvari, D. Rossi, T. Zinner, R. Bifulco, M. Jarschel, and G. Bianchi, "Survey of performance acceleration techniques for network function virtualization," *Proc. IEEE*, vol. 107, no. 4, pp. 746–764, Apr. 2019, doi: 10.1109/JPROC.2019.2896848.

[26] P. Shantharama, A. S. Thyagaturu, A. Yatavelli, P. Lalwaney, M. Reisslein, G. Tkachuk, and E. J. Pullin, "Hardware acceleration for container migration on resource-constrained platforms," *IEEE Access*, vol. 8, pp. 175070–175085, 2020.

[27] P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardware-accelerated platforms and infrastructures for network functions: A survey of enabling technologies and research studies," *IEEE Access*, vol. 8, pp. 132021–132085, 2020, doi: 10.1109/ACCESS.2020.3008250.

[28] P. Kansakar and A. Munir, "Selecting microarchitecture configuration of processors for Internet of Things (IoT)," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 4, pp. 973–985, Oct./Dec. 2020.

[29] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Microprocessor optimizations for the Internet of Things: A survey," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 7–20, Jan. 2018, 10.1109/TCAD.2017.2717782.

[30] P. Kansakar and A. Munir, "A design space exploration methodology for parameter optimization in multicore processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 2–15, Jan. 2018, 10.1109/TPDS.2017.2745580.

[31] T. Adegbija, R. Lysecky, and V. V. Kumar, "Right-provisioned IoT edge computing: An overview," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 531–536.

[32] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A review of low-end, middle-end, and high-end IoT devices," *IEEE Access*, vol. 6, pp. 70528–70554, 2018, doi: 10.1109/ACCESS.2018.2879615.

[33] J. O. Hamblen and G. M. E. van Bekkum, "An embedded systems laboratory to support rapid prototyping of robotics and the Internet of Things," *IEEE Trans. Educ.*, vol. 56, no. 1, pp. 121–128, Feb. 2013, doi: 10.1109/te.2012.2227320.

[34] D. Raskin, A. Vassiliev, V. Samarin, D. Cabezas, S. Hiererra, and Y. Kurniawan, "Rapid prototyping of distributed embedded systems as a part of Internet of Things," *Procedia Comput. Sci.*, vol. 135, pp. 503–509, 2018, doi: 10.1016/j.procs.2018.08.202.

[35] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.

[36] H. T. El Kassabi, M. A. Serhani, R. Dssouli, and B. Benatallah, "A multi-dimensional trust model for processing big data over competing clouds," *IEEE Access*, vol. 6, pp. 39989–40007, 2018, doi: 10.1109/access.2018.2856623.

[37] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018, doi: 10.1109/ACCESS.2018.2866491.

[38] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018, doi: 10.1109/MNET.2018.1700202.

[39] E. Al-Masri, K. R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, and C. Yan, "Investigating messaging protocols for the Internet of Things (IoT)," *IEEE Access*, vol. 8, pp. 94880–94911, 2020, doi: 10.1109/ACCESS.2020.2993363.

[40] E. Oyekanlu and K. Scoles, "Towards low-cost, real-time, distributed signal and data processing for artificial intelligence applications at edges of large industrial and Internet networks," in *Proc. IEEE 1st Int. Conf. Artif. Intell. Knowl. Eng. (AIKE)*, Sep. 2018, pp. 166–167.

[41] S. K. Roy, S. Misra, and N. S. Raghuwanshi, "SensPnP: Seamless integration of heterogeneous sensors with IoT devices," *IEEE Trans. Consum. Electron.*, vol. 65, no. 2, pp. 205–214, May 2019, doi: 10.1109/TCE.2019.2903351.

[42] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[43] A. Polianytsia, O. Starkova, and K. Herasymenko, "Survey of hardware IoT platforms," in *Proc. 3rd Int. Sci.-Practical Conf. Problems Infocommun. Sci. Technol. (PIC S&T)*, Oct. 2016, pp. 152–153, doi: 10.1109/INFOCOMMST.2016.7905364.

[44] A. R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, "A smart home energy management system using IoT and big data analytics approach," *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 426–434, Nov. 2017, doi: 10.1109/TCE.2017.015014.

[45] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 265–275, Jun. 2014, doi: 10.1109/JIOT.2014.2323395.

[46] Y. Cheng, H. Zhang, and Y. Huang, "Overview of communication protocols in Internet of Things: Architecture, development and future trends," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Dec. 2018, pp. 627–630, doi: 10.1109/WI.2018.00-25.

[47] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018, doi: 10.1109/JIOT.2017.2767608.

[48] J. Abella and A. Gonzalez, "On reducing register pressure and energy in multiple-banked register files," in *Proc. 21st Int. Conf. Comput. Design*, Oct. 2003, pp. 14–20, doi: 10.1109/ICCD.2003.1240867.

[49] G. Jia, G. Han, J. Du, and S. Chan, "A maximum cache value policy in hybrid memory-based edge computing for mobile devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4401–4410, Jun. 2019, doi: 10.1109/JIOT.2018.2878872.

[50] P. Kansakar and A. Munir, "A two-tiered heterogeneous and reconfigurable application processor for future Internet of Things," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 690–696, doi: 10.1109/ISVLSI.2018.00130.

[51] A. Gamatie, G. Devic, G. Sassatelli, S. Bernabovi, P. Naudin, and M. Chapman, "Towards energy-efficient heterogeneous multicore architectures for edge computing," *IEEE Access*, vol. 7, pp. 49474–49491, 2019, doi: 10.1109/ACCESS.2019.2910932.

[52] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–10, doi: 10.1145/2463209.2488734.

[53] P. I. Martos and A. Garrido, "Software patterns for asymmetric multiprocessing devices on embedded systems: A performance assessment," in *Proc. Eight Argentine Symp. Conf. Embedded Syst. (CASE)*, Aug. 2017, pp. 1–6, doi: 10.23919/SASE-CASE.2017.8115373.

[54] C. Pan, M. Xie, and J. Hu, "ENZYME: An energy-efficient transient computing paradigm for ultralow self-powered IoT edge devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2440–2450, Nov. 2018, doi: 10.1109/TCAD.2018.2858478.

[55] T. Gomes, S. Pinto, T. Gomes, A. Tavares, and J. Cabral, "Towards an FPGA-based edge device for the Internet of Things," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2015, pp. 1–4.
[56] F. Sun, S. Ravi, A. Raghunathan, and N. K. Jha, "Application-specific heterogeneous multiprocessor synthesis using extensible processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1589–1602, Sep. 2006.
[57] S. Hu and J. Haung, "Exploring adaptive cache for reconfigurable VLIW processor," *IEEE Access*, vol. 7, pp. 72634–72646, 2019, doi: 10.1109/ACCESS.2019.2919589.
[58] S. Harris, D. Harris, D. Chaver, R. Owen, Z. Kakakhel, and E. Sedano, "MIPSfpga: Using a commercial MIPS soft-core in computer architecture education," *IET Circuits, Devices Syst.*, vol. 11, no. 4, pp. 283–291, 2017, doi: 10.1049/iet-cds.2016.0383.
[59] W. Hu, L. Yang, B. Fan, H. Wang, and Y. Chen, "An 8-Core MIPS-compatible processor in 32/28 nm bulk CMOS," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 41–49, Jan. 2014, doi: 10.1109/jssc.2013.2284649.
[60] P.-F. Chiu, C. Celio, K. Asanovic, B. Nikolic, and D. Patterson, "Cache resiliency techniques for a low-voltage RISC-V out-of-order processor in 28-nm CMOS," *IEEE Solid-State Circuits Lett.*, vol. 1, no. 12, pp. 229–232, Dec. 2018, doi: 10.1109/lssc.2019.2900148.
[61] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gurkaynak, and L. Benini, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017, doi: 10.1109/tvlsi.2017.2654506.
[62] T. Jia, R. Joseph, and J. Gu, "An instruction-driven adaptive clock management through dynamic phase scaling and compiler assistance for a low power microprocessor," *IEEE J. Solid-State Circuits*, vol. 54, no. 8, pp. 2327–2338, Aug. 2019, doi: 10.1109/jssc.2019.2912510.
[63] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 1–12.
[64] G. Lallement, F. Abouzeid, M. Cochet, J. Daveau, P. Roche, and J. Autran, "A 2.7 pJ/cycle 16 MHz, 0.7 μW deep sleep power ARM cortex-M0+ core SoC in 28 nm FD-SOI," *IEEE J. Solid-State Circuits*, vol. 53, no. 7, pp. 2088–2100, Jul. 2018, doi: 10.1109/JSSC.2018.2821167.
[65] M. Rana, "Architecture of the Internet of energy network: An application to smart grid communications," *IEEE Access*, vol. 5, pp. 4704–4710, 2017, 10.1109/ACCESS.2017.2683503.
[66] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, and J.-C. Prevotet, "Internet of mobile things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs standards and supported mobility," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1561–1581, 2nd Quart., 2019.
[67] S. N. Han, G. M. Lee, N. Crespi, N. V. Luong, K. Heo, M. Brut, and P. Gatellier, "DPWSim: A devices profile for Web services (DPWS) simulator," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 221–229, Jun. 2015.
[68] J. Glova, T. Sabol, and V. Vajda, "Business models for the Internet of Things environment," *Procedia Econ. Finance*, vol. 15, pp. 1122–1129, 2014, doi: 10.1016/S2212-5671(14)00566-8.
[69] S. Nuratch, "A universal microcontroller circuit and firmware design and implementation for IoT-based realtime measurement and control applications," in *Proc. Int. Electr. Eng. Congr. (iEECON)*, Mar. 2017, pp. 1–4, doi: 10.1109/IEECON.2017.8075906.
[70] V. Sivaraman, H. H. Gharakheili, C. Fernandes, N. Clark, and T. Karliychuk, "Smart IoT devices in the home: Security and privacy implications," *IEEE Technol. Soc. Mag.*, vol. 37, no. 2, pp. 71–79, Jun. 2018.
[71] *IEEE Approved Draft Standard for an Architectural Framework for the Internet of Things (IoT)*, IEEE Standard P2413/D0.4.6, Mar. 2019, pp. 1–265.
[72] D. K. McCormick, *IEEE Technology Report on Wake-up Radio: An Application, Market, and Technology Impact Analysis of Low- Power/Low-Latency 802.11 Wireless LAN Interfaces*, Standard 802.11ba Battery Life Improvement: IEEE Technology Report on Wake-Up Radio, Nov. 2017, pp. 1–56, doi: 10.1109/IEEESTD.2017.8055459.
[73] Y. Ramadass, "Powering the Internet of Things," in *Proc. IEEE Hot Chips 26 Symp. (HCS)*, Aug. 2014, pp. 375–380, 10.1145/2627369.2631644.
[74] B. Chokkalingam, S. Padmanaban, P. Siano, R. Krishnamoorthy, and R. Selvaraj, "Real-time forecasting of EV charging station scheduling for smart energy systems," *Energies*, vol. 10, no. 3, p. 377, Mar. 2017.
[75] K. Ramesh, C. Bharatiraja, S. Raghu, G. Vijayalakshmi, and P. Sambanthan, "Design and implementation of real time charging optimization for hybrid electric vehicles," *Int. J. Power Electron. Drive Syst. (IJPEDS)*, vol. 7, no. 4, p. 1261, Dec. 2016.
[76] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt, and B. Zakeri, "Internet of Things (IoT) and the energy sector," *Energies*, vol. 13, no. 2, p. 494, Jan. 2020.
[77] A. Prasad and P. Chawda, "Power management factors and techniques for IoT design devices," in *Proc. 19th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2018, pp. 364–369, doi: 10.1109/ISQED.2018.8357314.
[78] A. Ali, "Thermal challenges and industry trends of consumer electronic devices," in *Proc. 34th Thermal Meas., Modeling Manage. Symp. (SEMI-THERM)*, Mar. 2018, doi: 10.1109/SEMI-THERM.2018.8357336.
[79] W. W.-M. Dai, "Historical perspective of system in package (SiP)," *IEEE Circuits Syst. Mag.*, vol. 16, no. 2, pp. 50–61, 2016, doi: 10.1109/MCAS.2016.2549949.

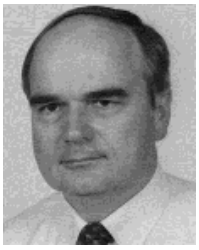
RAMESH KRISHNAMOORTHY (Member, IEEE) received the M.Tech. degree in embedded systems from the SRM Institute of Science and Technology, Chennai, India, in 2008. He is currently pursuing the Ph.D. degree in embedded systems and the Internet of Things with the Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Chennai, under the supervision of Kalimuthu Krishnan, under Visvesvaraya Ph.D. Scheme, MeitY, Government of India (2822). He is also working as an Assistant Professor with SRM Institute of Science and Technology. His research interests include embedded systems, the Internet of Things, and real-time operating systems.

KALIMUTHU KRISHNAN received the bachelor's degree in electronics and communication engineering from MIT, Anna University, in 2003, the master's degree in communication system from SRM University, in 2007, and the Ph.D. degree in wireless communications from the Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Chennai, India. He is currently working as an Associate Professor with the SRM Institute of Science and Technology. His areas of interests include wireless communication, signal processing, the Internet of Things, and embedded systems.

BHARATIRAJA CHOKKALINGAM (Senior Member, IEEE) received the bachelor's degree in electrical and electronics engineering from the Kumarguru College of Technology, Bharathiyar University, India, in 2002, the M.Eng. degree in power electronics and drives from the Government College of Technology (Anna University Coimbatore), India, in 2006, and the Ph.D. degree in electrical engineering from the SRM Institute of Science and Technology, Chennai, India, in 2015. He completed the post Centre for Energy and Electric Power, Faculty of Engineering and the Built Environment, Tshwane University of Technology, South Africa. He is currently a Postdoctoral Fellow/Visiting Research Scientist with Northeastern University, Boston, MA, USA. He is also currently working as an Associate Professor with the Department of Electrical and Electronics Engineering, SRM Institute of Science and Technology. He has authored more than 300 scientific articles. He is a member in of the IEEE Industrial Electronics, the IEEE Power Electronics, and the IEEE Power and Energy Societies. He serves as a Reviewer for more than 50 refereed journals, in particular, the IEEE.

SANJEEVIKUMAR PADMANABAN (Senior Member, IEEE) received the bachelor's degree in electrical engineering from the University of Madras, Chennai, India, in 2002, the master's degree (Hons.) in electrical engineering from Pondicherry University, Puducherry, India, in 2006, and the Ph.D. degree in electrical engineering from the University of Bologna, Bologna, Italy, in 2012.

He was an Associate Professor with VIT University, from 2012 to 2013. In 2013, he joined the National Institute of Technology, India, as a Faculty Member. In 2014, he was invited as a Visiting Researcher at the Department of Electrical Engineering, Qatar University, Doha, Qatar, funded by the Qatar National Research Foundation (Government of Qatar). He continued his research activities with the Dublin Institute of Technology, Dublin, Ireland, in 2014. Further, he served an Associate Professor for the Department of Electrical and Electronics Engineering, University of Johannesburg, Johannesburg, South Africa, from 2016 to 2018. Since 2018, he has been a Faculty Member with the Department of Energy Technology, Aalborg University, Esbjerg, Denmark. He has authored more than 300 scientific articles. He was a recipient of the Best Paper cum Most Excellence Research Paper Award from IET-SEISCON'13, IET-CEAT'16, IEEE-EECSI'19, IEEE-CENCON'19, and five best paper awards from ETAEERE'16 sponsored Lecture Notes in Electrical Engineering, Springer book. He is a Fellow of the Institution of Engineers, India, the Institution of Electronics and Telecommunication Engineers, India, and the Institution of Engineering and Technology, U.K. He is an Editor/Associate Editor/Editorial Board of refereed journals, in particular the IEEE Systems Journal, IEEE Transaction on Industry Applications, IEEE Access, *IET Power Electronics*, *IET Electronics Letters*, and *International Transactions on Electrical Energy Systems*, Subject Editorial Board Member—*Energy Sources—Energies Journal*, MDPI, and the Subject Editor for the *IET Renewable Power Generation*, *IET Generation*, *Transmission and Distribution*, and *FACTS* journal (Canada).



ZBIGNIEW LEONOWICZ (Senior Member, IEEE) received the M.Sc., Ph.D., and Dr.Sci. degrees in electrical engineering from the Wroclaw University of Science and Technology, Wroclaw, Poland, and the Habilitate Ph.D. degree from the Bialystok University of Technology, Bialystok, Poland, in 1997, 2001, and 2012, respectively. Since 1997, he has been with the Department of Electrical Engineering, Wroclaw University of Science and Technology, where he is currently an Associate Professor. His current research interests include power quality, control and protection of power systems, renewables, industrial ecology, and applications of advanced signal processing methods in power systems.

JENS BO HOLM-NIELSEN was born in 1954. He received the Ph.D. degree. He is currently the Head of the Research Group of Bioenergy and Green Engineering, Department of Energy Technology, Aalborg University, Denmark. He has 30 years of experience in the field of Biomass feedstock production, Biorefinery concepts, and Biogas production. He was a Board Member of Research and Development, committees of the cross-governmental body of biogas developments, Denmark, from 1993 to 2009. He is also the Secretary and/or Chair of NGO biogas and bioenergy organizations. He is also a Chair and presenter of Sustainable and 100 per cent Renewables and SDG-17 goals. Experience of a variety of EU projects, organizer of international conferences, workshops and training programs in EU, USA, Canada, China, Brazil, India, Iran, Russia, Ukraine, and among others. His research interests include managing research, development and demonstration programs in Integrated agriculture, environment, and energy systems. He is fulfilled with the biomass and bioenergy Research and Development Projects. His principle research's interests include on biofuels, biogas, and biomass resources. He is also EDU and Supervising the M.Sc. and Ph.D. degree students in these research fields. His training programs are International courses, training programs, and supervision for Ph.D. students and academic staff, governmental bodies and experts in bioenergy systems.

MASSIMO MITOLO (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Napoli Federico II, Italy, in 1990. He is currently a Full Professor of electrical engineering with the Irvine Valley College, Irvine, CA, USA, and a Senior Consultant in Electric Power Engineering with Engineering Systems Inc., ESi. He has authored more than 118 journal articles and the books *Electrical Safety of Low-Voltage Systems* (McGraw-Hill, 2009) and *Laboratory Manual for Introduction to Electronics: A Basic Approach* (Pearson, 2013). His research interests include the analysis and grounding of power systems and electrical safety engineering. He was a recipient of numerous recognitions and best paper awards, including the IEEE-I&CPS Ralph H. Lee Department Prize Paper Award, the IEEE-I&CPS 2015 Department Achievement Award, and the IEEE Region Six Outstanding Engineer Award. He is currently the Deputy Editor-in-Chief of the IEEE Transactions on Industry Applications. He is active within the Industrial and Commercial Power Systems Department of the IEEE Industry Applications Society (IAS) in numerous committees and working groups. He also serves as an Associate Editor for the IEEE IAS Transactions. He is also a registered Professional Engineer with the state of California and in Italy.