2020 International Conference on Information Technology Systems and Innovation (ICITSI)
Bandung - Padang, October 19 - 23, 2020
ISBN: 978-1-7281-8196-7

Edge Computing for Internet of Things Based on FPGA

Rian Ferdian
Computer Engineering Department
Universitas Andalas
Padang, Indonesia
rian.ferdian@it.unand.ac.id

Ratna Aisuwarya
Computer Engineering Department
Universitas Andalas
Padang, Indonesia
aisuwarya@it.unand.ac.id

Tati Erlina
Computer Engineering Department
Universitas Andalas
Padang, Indonesia
tatierlina@it.unand.ac.id

***Abstract*—The Internet of things (IoT) is a device that can perform mechanical or electrical interaction to its environment while also transferring data through a network. The exponential rise in IoT implementation starts to affect human-to-human communication recently. Furthermore, it is hard to achieve a secure transmission as the microprocessor-based IoT nodes do not have enough computing power. Edge computing in IoT (edge-IoT) promises low bandwidth and secure transmission. Edge-IoT is the ability of a node to perform high computation before sending its data. Thus, instead of sending raw data, an edge node is sending a post-processed the data with much lower bandwidth. An edge-IoT node also can perform a complex encryption computation before the transmission to secure the data. This paper proposes an edge-IoT using field-programmable gate arrays (FPGA) as an IoT node. The edge-IoT with FPGA can perform secure, parallel data acquisition and low bandwidth data transmission with almost the same cost as the microprocessor-based IoT.**

***Index Terms*—internet of things, edge computing, FPGA, advanced encryption standard, edge IoT**

I. Introduction

The internet of things (IoT) has a vital role in our life in recent years. The IoT applications are vast and range from smart home, smart healthcare, remote sensing, and factory automation. The mainstream applications of IoT are using microcontrollers such as Arduino UNO [1]–[4]. Because the computation in the microcontroller is limited, a node in IoT mostly only sends the data to the cloud and leaves heavy computation into the server. This method makes a secure and fast implementation of IoT device is hard to achieve. In a conventional IoT system, one node can only do one task only due to microcontroller limitations. The other issue in the conventional IoT system is the communication bandwith, the exponential increase in IoT machine-to-machine needs also begins to affect the human-to-human communication network.

Edge computing is another trend in remote sensing methods. Edge computing is a distributed computation technique that allows a node to process the data itself to increase the response time, improve the bandwidth, and secure the data. The application of edge computing in IoT reduces the bandwidth in the communication traffic by sending post-processed data instead of raw data. Moreover, the edge computing node also can have an encryption part to achieve secure transmission. Because the nodes in edge-IoT are required to perform some additional computation, an implementation using an application-specific processor (ASP) is superior compared to the general-purpose processor (GPP) such as a microcontroller.

The ASP includes field-programmable gate arrays (FPGA) and application-specific integrated circuit (ASIC). The interconnection of digital logic and finite state machine (FSM) implements an application in ASP, unlike a series of looping of memory and logic units such in the GPP. Because of that, to reprogram an ASP is not as easy as programing a GPP. However, annexing additional computation to the ASP will not affect the computational speed since all the logic units and FSMs are running in total parallel. Furthermore, ASP is suitable for IoT applications as it is unnecessary to change its functionality frequently.

In the ASP group itself, there are some tradeoffs between FPGA and ASIC. ASIC is suitable for mass-production output, and it will bring less cost and power while having the best performance. While FPGA tradeoff the performance and energy for the reconfigurability. Furthermore, FPGA able to change its hardware while the ASIC has a permanent form. FPGA essentially is a prototyping tool for ASIC before the mass-production stage. However, FPGA starts became the central core in a system application development since its cost became cheaper. Moreover, for applications which not addressed for the mass-production, FPGA becomes preferable.

There have been several proposed approaches for edge-IoT hardware using FPGA [5]–[7]. The IoT proposed in [5] proposed a cost-effective communication stack for different IoT standards using edge computation with the Xilinx FPGA system-on-chip (SoC). The approach in [6], [7] already showed how an edge-computing approach for IoT could improve security and bandwidth efficiency. However, the edge-computing methods in [6], [7] are still using minimize data centers near the IoT nodes, which is not cost-efficient. In this paper, we propose an edge-IoT method using FPGA to gain secure and low bandwidth communication for an IoT node while maintaining a reasonable cost.

II. Description of The System

A. *FPGA DE10-lite*

In this paper, we use DE10-lite, which is the low tier of FPGA from Intel. The cost of the DE10-lite almost the same as the middle lineup of Arduino processors. The size of

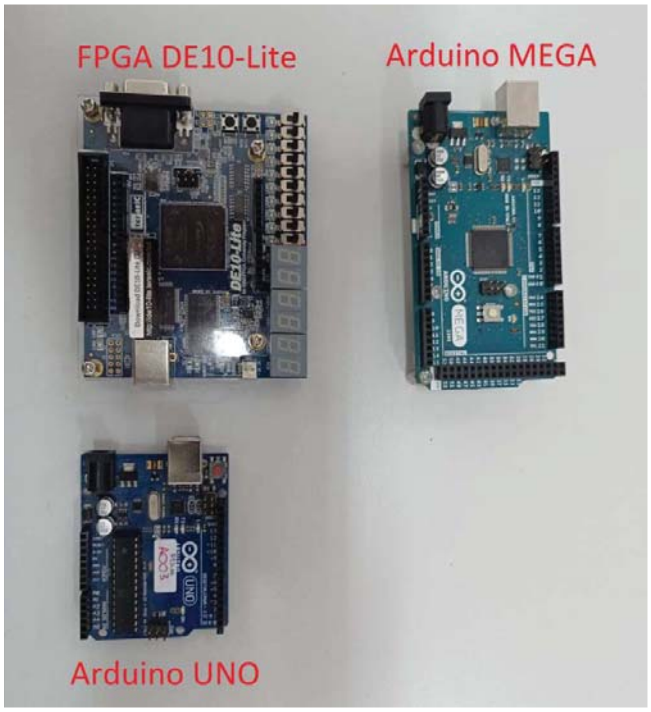

Fig. 1. FPGA DE10-lite size comparison with Arduino

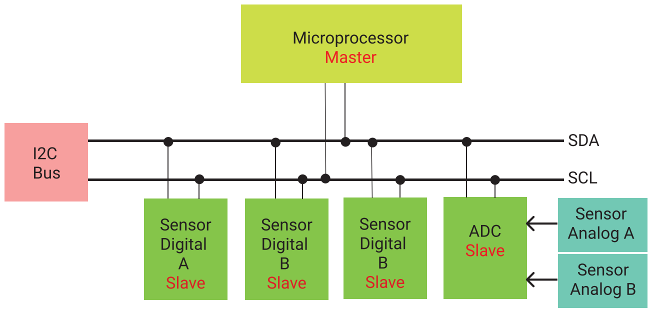

Fig. 2. Arduino I2C sensor data acquisition

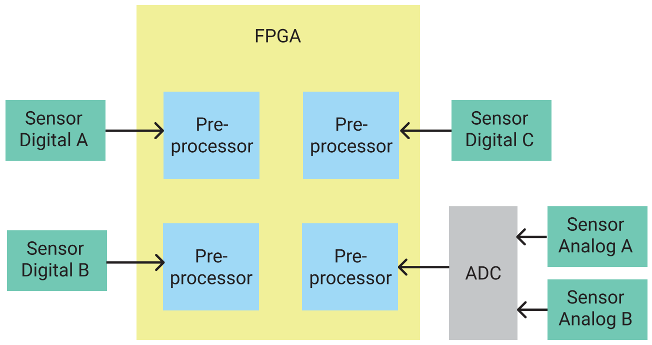

Fig. 3. FPGA sensor data acquisition

the DE10-lite FPGA also only 50% bigger compared to the Arduino Mega as shown in the Fig.1. This FPGA includes 32 pins of Arduino's connectors and 40 pins of general-purpose I/O (GPIO). The number of connected sensors increases due to the I/O pins large number. Furthermore, the DE10-lite FPGA also has integrated dual analog to digital converter (ADC). Thus, the analog sensor also can be connected without an additional module. All the operations in the FPGA are black-box operations implemented by lookup tables (LUTs) and flip-flops. The DE10-lite has 50K of programmable LUTs. This paper uses Verilog hardware description language (HDL) to design the hardware in the FPGA. The Intel's Quartus is the design software provided by intel to program the DE10-lite FPGA. It is also possible to enhance the FPGA's design flexibility by creating a system on chip (SoC) [8]. SoC can accelerate the development cycle by creating hardware-software collaboration. The Quartus software provides the SoC builder tool using the NIOS II processor, which is a 32 bit proprietary RISC softcore processor from Intel. The connection between the NIOS II processor and the custom hardware in the DE10-lite FPGA is established trough the Avalon system bus interface.

B. *Sensor's Data Acquisition*

Acquisition speed is an essential aspect of an IoT node for the real-time system application. The node needs to catch up with the sensor's response time, which usually in milliseconds order. Arduino system commonly uses I2C serial protocol to communicate with the sensor attached to its I/O. The I2C is a low-speed master-slave bus protocol that controls the peripheral order that the processor can access. Furthermore, it is impossible to achieve concurrency for multiple sensor readings in I2C communication because it can only access one peripheral at a time. The structure of I2C diagram can be seen in Fig 2.

The FPGA has a different approach for reading the sensor's data acquisition concurrently. The digital sensor can be directly connected to the digital I/O or GPIO. Instead of applying a software driver to read the data, the FPGA needs to have a sensor preprocessor to convert the sensor electrical parameter into the actual physical parameter presentation. However, DE10-lite only has one ADC module with six channels, which only allows it to read one analog sensor simultaneously. The structure of sensor data acquisition with FPGA is shown Fig 3.

C. *Advanced Encryption Standard (AES)*

Advanced Encryption Standard (AES) is a symmetric block cipher developed by Belgian's researcher Vincent Rijment and Joen Daemen. Before selected as the new standard that supersedes the data encryption standard (DES) in 2001, it is called the Rijndael algorithm. The AES has two central processes: encryption and cipher-key scheduling, which ensure its security.

The encryption consists of four sub-operations, which are subByte, shiftRows, mixColumns, and addRoundKey. The subByte replaces each byte of the input data into a new byte from a pre-defined lookup table. Then, shiftRows will shift the last three rows of the prior step state cyclically in a determined number of shift operations. Furthermore, The

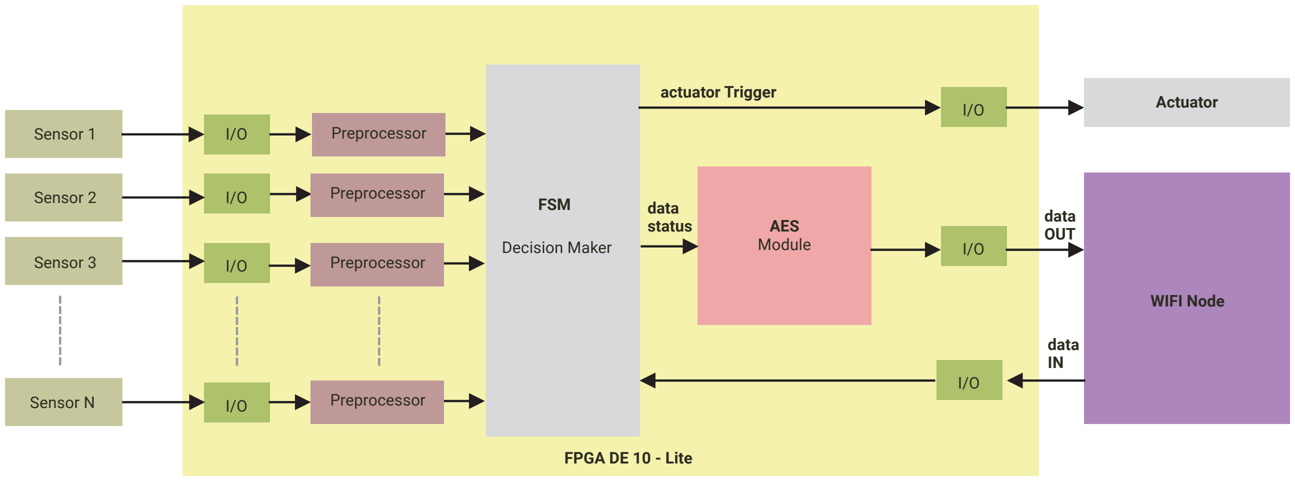

Fig. 4. Proposed edge-IoT block diagram

mixColumns then mix each four columns of the data of the former state by using a Galois Field Matrix multiplication. Last, the addRoundKeys performs a bitwise xor operation of the data state and the received round keys.

The second operation of the AES is the scheduling to create the chipper key. The cipher key in AES is an array of 32 bits number. The round constant is retrieved from the first column of the current state cipher key array. This column is shifted cyclicly then non-linear transformed in subByte operation similar to the sub-process in encryption. The only way to crack the AES algorithm is to obtain the exact number and configuration in this cipher key scheduling.

III. Proposed Method

The proposed block diagram of edge-IoT node is shown in the figure 4. It can be seen that a node here consists of 4 parts. The first one is the sensor reader, which connects the N parallel sensors to the FPGA. In this part, the reading process is done in parallel, so the number of sensors will not affect the reading speed. The maximum number of connected sensors depends on the I/O port that available. It is different compared to the microcontroller implementation, where the number of sensors will affect the reading rate.

The second part is the finite state machine (FSM) unit, which is the decision-maker in this IoT node. The FSM process all the data from the sensors and combine it with the trigger from the cloud. Then the FSM decides the status of the environment around the IoT node. A remote user also can send a trigger to the node through the network that can affect the actuator's decision to the environment. Instead of sending all the sensors data to the cloud, this node only sends the FSM's decision output.

The AES encryptor is the third part of this edge-IoT that guarantee the security in the data transmission. The AES is still unbreakable at the time of this paper published. All the data from the FSM unit are encrypted before the node transfers it to the cloud. In this paper, we use the 128-bit version of the AES module. The encryptor has two main processes which are encryption and cipher key scheduling. The XOR operator implements the AES input and cipher multiplication efficiently in the FPGA.

The last section of this edge-IoT node is the wireless transceiver module. This part responsible for transferring the encrypted output from the node to the cloud. The transceiver module can also send some defined information or instruction from the cloud to the node. In this paper, we use node MCU as the wireless transceiver module. Since the node only needs low rate data transmission, the low-bandwidth wide-area wireless module utilization such as LoRa also possible for applications in an extremely remote area.

IV. Results

The proposed technique can be applied to any IoT application. However, this paper implemented the method into a weather station. The weather station has four sensors, which are temperature, humidity, anemometer, and rainfall sensor. All the sensors are connected to one FPGA DE10-lite. The FSM in the FPGA is using the decision tree computation for the weather prediction algorithm. The node only sends the post computed weather information instead of the sensors'raw data. Furthermore, the weather data are encrypted before the wifi module sends it.

The synthesis results from Intel Quartus is shown in the Table I below. The table indicates the FPGA utilization for the proposed technique implementation of the edge-IoT.

TABLE I
Synthesis Results for The Proposed method

Device	10M50DAF484C6GES
Family	MAX10
Total Logic Elements	6,877 / 49,760 (14 %)
Total Register	2994
Total Pins	76 / 360 (21 %)
Maximum Clock Speed	92.18 MHz

The table I shows that the FPGA DE10-lite is sufficient to include the AES encryption module in its computation. The

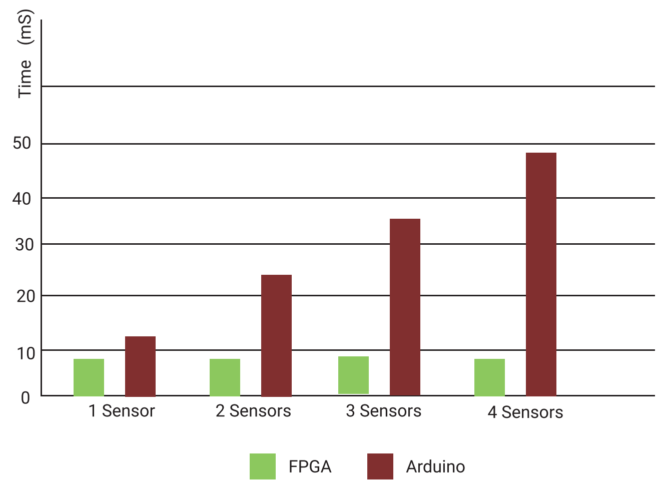

Fig. 5. Timing Performance Comparison FPGA vs Arduino

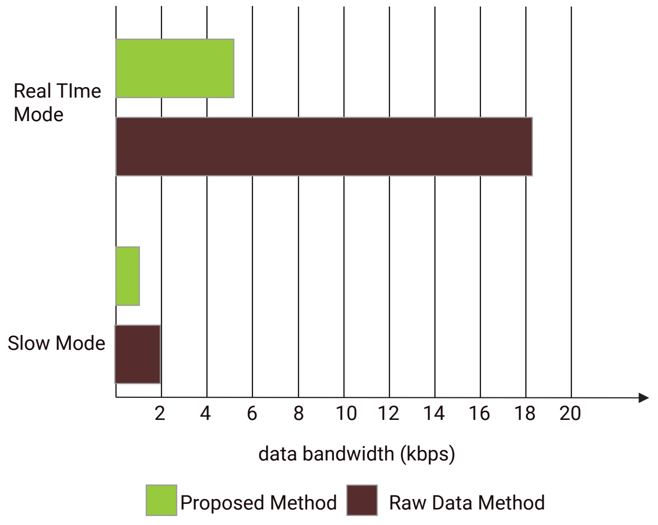

Fig. 6. Bandwidth Comparison of the Proposed Method vs Raw Data Method

results indicate the total utilized logic elements are 14%, and the total pins are 21%. The AES module here will guarantee the security of the data transmission in this IoT system. Furthermore, the FPGA utilization still below 20% that makes it possible for other applications or improvements.

The timing performance comparison between the proposed method with the FPGA and Arduino is shown in Fig 5. The diagram indicates that the edge-IoT with FPGA can maintain the timing performance of the 7mS, while the timing performance for the IoT node using Arduino decrease in linear to the number of the sensors attached to its node. The proposed edge-IoT is suitable for strict real-time applications.

The bandwidth performance of the proposed technique is shown in Fig 6. The figure compares the proposed method and the raw data transmission method. It also shows the bandwidth comparison between the real-time mode achieved by the edge-IoT and slow-mode used in the conventional IoT. The proposed method delivered 41.5 kbps in the real-time mode, and 0.67 kbps in the slow-mode. It is shown that the proposed method requires much less data bandwidth compare to conventional IoT.

V. Conclusion

The edge-IoT implementation using FPGA is presented in this paper. The FPGA used in this paper is DE10-lite, which has almost the same price compared to the Arduino Mega. The proposed IoT node can accommodate the AES encryption computation to secure the data and still leaving much room for further expansion. Then, the post-processed calculation in the node can guarantee a much lower bandwidth requirement for data transmission. Furthermore, real-time sensor data acquisitions with fast response time also achievable with the absolute parallel processing that is not affected by the sensor number.

References

[1] G. Acciari, M. Caruso, R. Miceli, L. Riggi, P. Romano, G. Schettino, and F. Viola, “Piezoelectric rainfall energy harvester performance by an advanced arduino-based measuring system,” *IEEE Transactions on Industry Applications*, vol. 54, no. 1, pp. 458–468, 2018.

[2] Q. A. Al-Haija, “Design and on-field testing of wireless sensor network-based air quality monitoring system,” *JITCE (Journal of Information Technology and Computer Engineering)*, vol. 3, no. 2, pp. 54–59, 2019.

[3] R. Putri and T. Wibowo, “Prototype of smart minimarket,” *JITCE (Journal of Information Technology and Computer Engineering)*, vol. 3, no. 1, pp. 39–53, 2019.

[4] R. Aisuwarya, Melisa, and R. Ferdian, “Monitoring and notification system of the position of a person with dementia based on internet of things (iot) and google maps,” in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2019, pp. 396–400.

[5] T. Gomes, S. Pinto, T. Gomes, A. Tavares, and J. Cabral, “Towards an fpga-based edge device for the internet of things,” in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, 2015, pp. 1–4.

[6] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, “A secure iot service architecture with an efficient balance dynamics based on cloud and edge computing,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2019.

[7] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, “Integration of iot, transport sdn, and edge/cloud computing for dynamic distribution of iot analytics and efficient use of network resources,” *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1420–1428, 2018.

[8] T. Adiono, R. Ferdian, N. Ahmadi, F. Dawani, and I. Abdurrahman, “Flexible data sharing architecture of wimax heterogeneous multiprocessor system on chip,” in *2015 International SoC Design Conference (ISOCC)*, 2015, pp. 131–132.