# A Low-Power High-Performance SoC Platform for IoT Applications

Hyewon Jeong, Jinhwan Lee, Hoyoung Yoo and In-Cheol Park[a]
Department of Electronical Engineering, KAIST
E-mail: {hwjeong.ics, jhlee.ics, hyyoo.ics}@gmail.com, icpark@kaist.edu

***Abstract* - The rapid growth of wirelessly connected devices introduces a diverse range of applications and requires intelligent hardware platforms that integrate computing, sensing, and wireless connectivity in a compact system-on-chip (SoC). This paper presents a low-power, high-performance SoC platform that supports dynamic power management and secure communication. The SoC platform consists of 16-/32-bit programmable ARM9 cores, a power management unit with multiple low-power modes, analog and digital peripherals, and security engines. A complete tool-chain with an automatic platform generator has also been developed to ease and accelerate the application development. Fabricated in a 65 nm CMOS technology, an implementation of the proposed platform occupies an area of 1.0 × 1.7 mm$^2$.**

## I. INTRODUCTION

Recent years have seen a rapid increase of mobile computing devices such as smartphones, smart watches, and intelligent home devices. These smart devices inter-connected via wireless networks enable ubiquitous computing and collectively form the Internet of Things (IoT) [1]. As described in Fig. 1, these IoT systems will be employed in a wide range of applications ranging from home automation, health monitoring, and smart man-ufacturing to many others. Since these applications require not only computing but also sensing and wireless communication, there is a need for high-performance hardware platforms well-quipped with various sensor interfaces. In addition, as these platforms must operate with long lifespan in power-constrained environments, they must manage the system-level power consumption intelligently [2]. Moreover, many issues have risen around security and privacy, since these devices are always connected and communicate with each other [3], making the system design task more challenging.

Meanwhile, a development of IoT applications utilizing the hardware platforms requires a software development environment that is suitable for the development of sophististicated software. Thus, complete software development tools for the proposed SoC platform have been constructed. In addition, application programming interfaces (APIs) are developed for all the peripherals integrated on the proposed platform. To ease and speed up the developing processes, furthermore, an automatic platform generator is introduced, enabling flexible system configurations and generation with user-friendly graphical interfaces.

Fig. 1. IoT applications.

Previous IoT SoC platforms produced by several semiconductor companies achieve high integration and low-power consumption, but they tend to operate at relatively low operating frequencies, providing limited performances [4-6]. This work demonstrates a highly integrated, high-performance SoC platform composed of a synthesizable ARM9 core and various on-chip peripherals targeting a wide range of IoT applications. The platform achieves low standby power consumption by leveraging a number of well-known low-power techniques including clock gating, power gating, and several low-power modes. The SoC platform also integrates cryptographic engines capable of data encryption and decryption to alleviate security concerns.

The rest of this paper is organized as follows. Section II provides an overview of the proposed hardware platform. Software development environments and constructed tools for the proposed platform are explained in Section III. Section IV summarizes the proposed SoC platform and its implementation results. In Section IV, implementation results are discussed. Finally, conclusions are drawn in Section V.

a. Corresponding author; icpark@kaist.edu

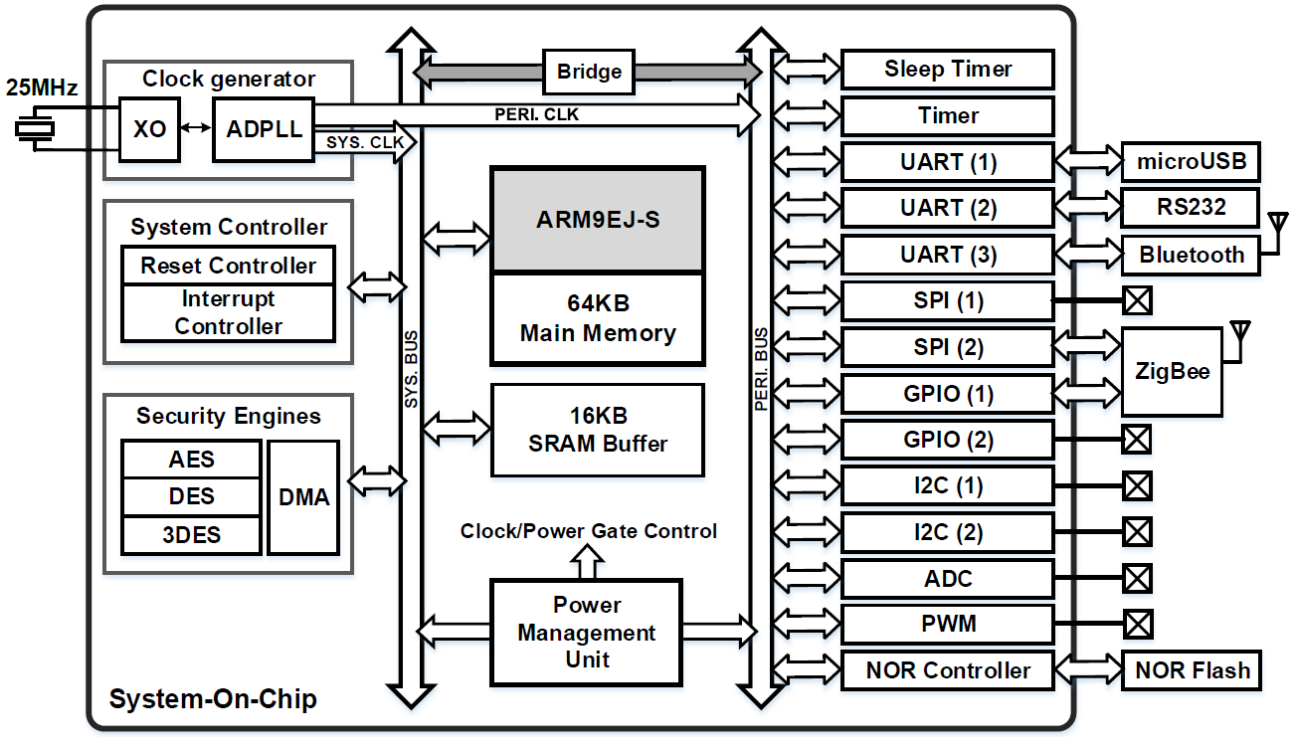


Fig. 2. Overall block diagram of the SoC platform.

## II. Overview of Hardware Platform

### *A. Overall Architecture*

Fig. 2 shows the main building blocks of the proposed SoC platform. The chip utilizes a synthesizable ARMv5-compatible processor called ARM926EJ-S with rich DSP instructions [7]. The ARM926EJ-S incorporates both 32-bit ARMv5 and 16-bit Thumb2 instruction set architectures (ISAs), thereby guaranteeing the performance at a very reasonable power consumption. The core is implemented based on a 5-stage pipeline architecture and achieves 512 MHz operating frequency with 44 k equivalent gates in a 65 nm CMOS process. A RAM of 64 KB is employed as the main memory.

The core serves as a parent to peripheral devices such as timers, UARTs, and GPIOs. The core and the memory-mapped peripheral devices communicate over two 32-bit buses; a system bus for fast memory access, and a peripheral bus for reduced complexity and low-power consumption. The peripheral bus is attached to the system bus via a bridge. The system bus to which the core and high-speed peripherals are attached operates at 200 MHz frequency, while the peripheral bus operates at 50 MHz frequency. The chip has a flexible clocking scheme where the system clock can be derived directly from an external clock or an on-chip clock generator of which frequency is ranging from 100 kHz to 1 GHz. Starting from a 25 MHz crystal oscillator, the clock generator drives the desired system clock by employing a programmable all digital phase locked loop (ADPLL).

The SoC platform contains a family of peripheral devices to perform various tasks assigned by IoT applications. Two 32-bit timers, watchdog timers, and a PWM controller are implemented and one of the timers can be designated as a sleep timer to wake up the platform from the low-power modes. The single-channel 12-bit SAR ADC is integrated to support an analog-input interface. Serial and parallel interfaces such as SPIs, I2Cs, UARTs, and GPIOs are employed to cope with a wide variety of sensors, LCD modules, and communication modules. In particular, the UART interface and the SPI with GPIO interfaces can be configured to serve as Bluetooth and ZigBee wireless communication modules, respectively. The NOR flash controller is to read the stored boot code from the NOR flash memory. Though the default booting device is the NOR flash, the platform can boot from the USB memory by passing through the UART controller.

### *B. System Controller*

The system controller consisting of a reset controller and an interrupt controller manages the initialization and the interrupt service. The reset controller takes the role of initialization. At startup, the clock generator allows the reset controller to reset the entire platform after the system clock is stabilized. If the system is in the low-power mode, the reset controller wakes up the platform at every system reset. The reset controller is also responsible for the soft reset of individual peripheral devices.

The interrupt controller gathers up the interrupt signals generated from peripheral devices, and signals an IRQ interrupt to make the ARM core enter the interrupt service routine corresponding to the highest-priority peripheral. Up to 32 interrupt signals can be handled by the controller and the priority of each peripheral is programmable. If the system is in the low-power mode, the IRQ interrupt signal is

TABLE I.
Clock and power conditions for power domains

| Power Domain | Power Mode | Conditions | | Power Gating | Clock Gating |
|---|---|---|---|---|---|
| | | Supply (V) | Frequency (MHz) | | |
| CPU, On-Chip Memory | Active | 1.2 | 200 | Y | Y |
| | Halt | 1.2 | 0 | | |
| | Snooze | 1.2 | 0 | | |
| | Shut down | 0 | 0 | | |
| High-Speed Peripherals | Active | 1.2 | 200 | N | Y |
| | Halt | | 0 | | |
| | Snooze | | 0 | | |
| | Shut down | | 0 | | |
| Low-Speed Peripherals | Active | 1.2 | 50 | N | Y |
| | Halt | | 50 | | |
| | Snooze | | 0 | | |
| | Shut down | | 0 | | |
| Always-On Units | - | 1.2 | 200/50 | N | N |

also sent to the power management unit to wake up the system.

*C. Security Engines*

The SoC platform has a suite of cryptographic hardware accelerators to alleviate security issues in IoT applications. Three encryption algorithms, which are most widely used, are implemented in hardware: the Advanced Encryption Standard (AES) [8], Data Encryption Standard (DES) [9], and triple DES (3DES). The 3DES applies the DES algorithm three times in succession to strengthen security. The AES hardware processes a 128-bit block at a time by using a 128-/192-/256-bit symmetric key, while the DES and 3DES deal with a 64-bit block using 64-bit and 64-/128-/192-bit keys, respectively. The security engines can be configured either for encryption or decryption simply by setting a single-bit signal. The engines are implemented to achieve high throughput, i.e., 2.56 Gbps at 200 MHz for AES-128, and low-area complexity by exploiting the similarities of encryption and decryption. Since it is usually required to transfer a block of data from or to the security engines, a separate DMA is used to manage the data transfers without the intervention of the core. The encrypted data are transmitted via wireless networks and the received data can be decrypted.

*D. Power Management Unit*

The SoC also contains a power management unit (PMU) to control the operating mode of the system effectively and thereby reduce the standby power consumption. The four operating modes and their transitions are depicted in Fig. 3. The clock gating and power gating techniques are also used in circuit level to support the power management [10-12]. Fig. 4 shows a simplified diagram of the clock gating applied in circuit level. In the PMU, there is a 32-bit configuration register to control the clock gating circuits residing in the peripheral block wrapper. The power gating is realized by means of header switches placed between the

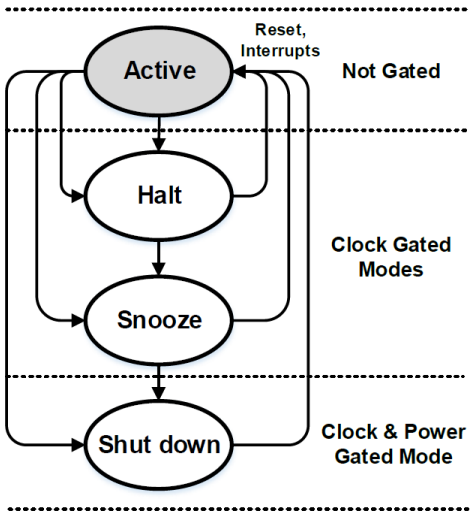


Fig. 3. Low-power modes.

power supply and the gated logic as shown in Fig. 5.

The platform has two power domains. The core and the memory are in a power domain, and the rest of the system including peripherals and buses is in the other power domain which is not gated. The functional units attached to the system bus reside in a high-speed clock domain, while all the peripheral devices belong to a low-speed clock domain. There are several units that are neither clock gated nor power gated, which are called always-on units. The clock generator, PMU, system controllers, and sleep timer are defined as always-on units.

There are four operational modes: Active, Halt, Snooze, and Shut-down. Table I presents the operating frequency and supply voltage conditions applied to the power mode and the power domain. The active mode is the normal operating mode of the platform where all the functional units are operational. The platform enters the active mode after power-up and reset. In the halt or snooze mode, the clocks to several functional units are disabled by utilizing the clock

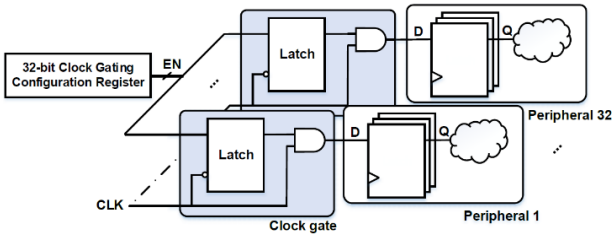


Fig. 4. Clock gating circuits.

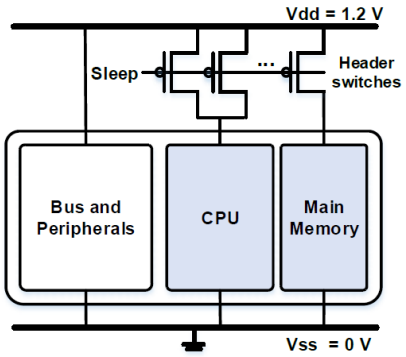


Fig 5. Power gating circuits.

gating circuit shown in Fig. 4. Meanwhile, the units responsible for the mode control, such as the clock generator, system controller, sleep timer, and power management unit, remain in operation. The halt mode allows a software application to stop the core, while continuing to monitor the interrupt-request and reset signals. The snooze mode disables the peripherals along with the core, but leaves the sleep timer running. The transition from the snooze mode to the active mode occurs when the sleep timer expires after a specific period of time or the reset signal is triggered. In the halt or snooze mode, the core can return to the active mode quickly when an interrupt occurs. The shut-down mode provides great power savings by powering off the core and the main memory as illustrated in Fig. 5. The system comes back to the active mode on expiration of the sleep timer or reset. As the wake-up process from the shut-down mode accompanies complicated processes such as booting and initialization, it leads to a long wake-up penalty.

Each low-power mode involves a trade-off between the wake-up overhead and the power saving, and thus, it is possible to allow the designer to manage the power consumption dynamically according to the required conditions.

## III. Software Development Environments

### A. Software Supports

A software toolchain for the proposed SoC platform has been established to support the development and production of IoT applications. A cross-compilation toolchain is built based on the GNU gcc v4.6.1 [15] and related binutils such as assembler, linker, *etc*. The GNU debugger (GDB) 6.8 [16] is adopted in the toolchain for software and hardware debugging. For graphical and a command-line interfaces, the

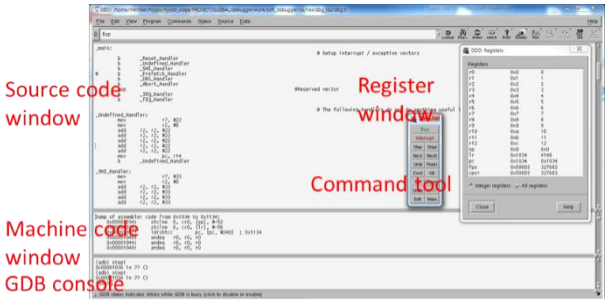


Fig 6. GNU DDD interface.

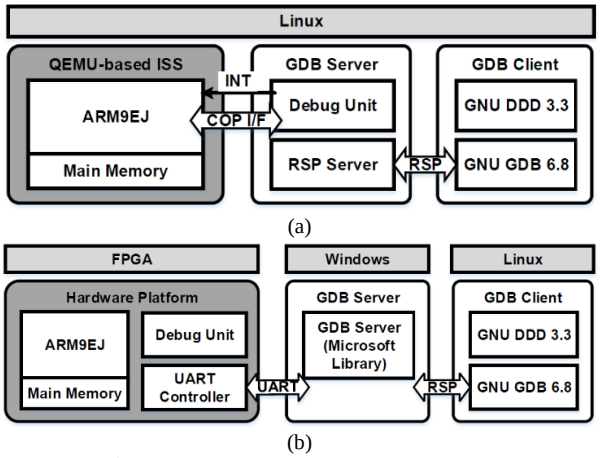


Fig. 7. (a) Software and (b) hardware debuggers developed on top of the GNU debugger.

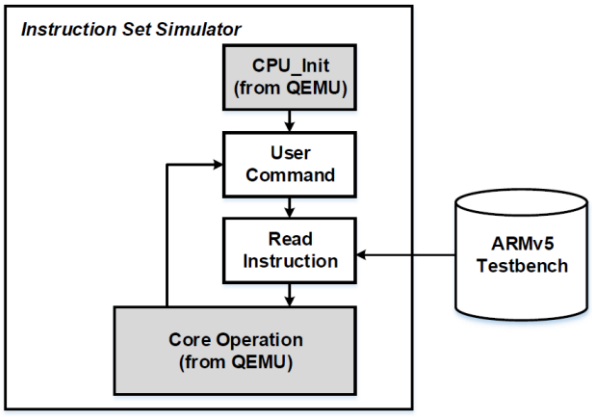


Fig. 8. QEMU-based instruction set simulator.

GNU data display debugger (DDD) 3.3 [17] shown in Fig. 6 is employed along with the GDB.

Fig. 7 (a) shows a software debugger consisting of an in-house QEMU-based instruction set simulator (ISS), a GDB server, and a GDB client. The ISS can execute either an instruction step by step or a number of instructions at one go. Meanwhile, the contents in registers and memories can be examined on request. In addition, breakpoints can be set to specific addresses. As shown in Fig. 8, the ISS utilizes source codes extracted from the QEMU core initialization and operation program [22]. The debugging is processed by issuing an interrupt request to invoke the debug handler of the ISS. The debug unit inside the GDB server asks for debugging and communicates with the ISS using coprocess-

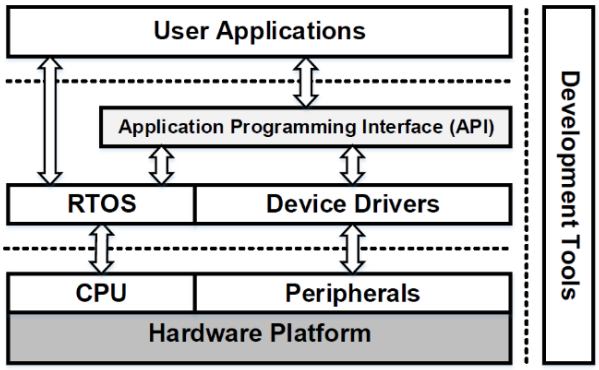


Fig. 9. The complete IoT platform including hardware architecture and software.

or (COP) interface. The GDB client, on the other hand, communicates with the GDB server using remote serial protocol (RSP) packets over TCP/IP. For the hardware debugger illustrated in Fig. 7 (b), the hardware platform mounted on the FPGA board replaces the ISS. The debug request from the GDB server is delivered to the on-chip debugger through the UART interface. Then, the on-chip debugger interacts with the core using the COP interface.

Fig. 9 illustrates a complete IoT platform including hardware architecture and software supports. The complete platform includes a real-time operating systems (RTOS) as many embedded systems must do. The RTOS allows the core to perform multiple tasks simultaneously by exploiting an advanced scheduling algorithm. Unlike the general-purpose OS, the RTOS aims to process the application in real time by managing hardware resources such that a specific process satisfies strictly defined time constraints. Thus, the RTOS guarantees the performance, and sometimes accelerates complicated software applications. Two widely-used open-source embedded RTOS, FreeRTOS 7.6.0 [13] and uC/OS-II 2.5.2 [14], are ported to the proposed platform. Application software interacts with the RTOS through APIs, and in turn, the RTOS controls peripheral devices through device drivers programmed with the APIs. The APIs and the device drivers developed for peripheral devices such as UARTs, SPIs, and timers have also been tested on the platform.

*B. Automatic Platform Generator*

This work introduces a novel platform generator that assembles only the necessary IPs into a synthesizable platform. Given the list and configuration of Verilog IPs that are actually used, in other words, the platform generator generates a synthesizable platform containing only the required IPs. The automatic platform generator embeds reusable IPs including an ARM9 core, a system bus, a peripheral bus, and peripheral devices such as UART, SPI, GPIO, timer, PWM, and NOR flash controller. The input to the automatic generator is a set of parameters describing the desired platform, such as operating frequency, the number and the type of buses, internal bit-width, IP locations, base addresses, and so on. The proposed generator goes through four steps: 1) defines a bus hierarchy and sets the bit-width and operating frequency of each bus; 2) selects and configures the IPs to be integrated and locates them in the

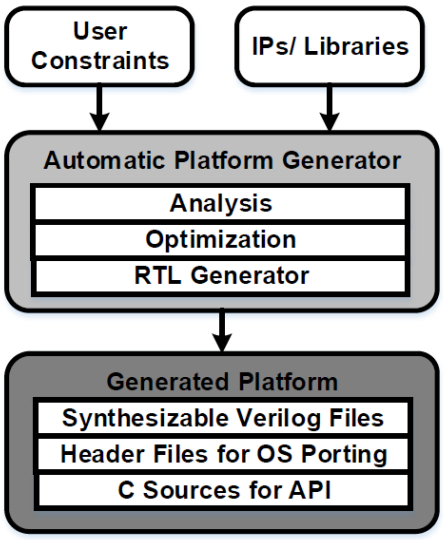


Fig. 10. A block diagram of automatic platform generation.

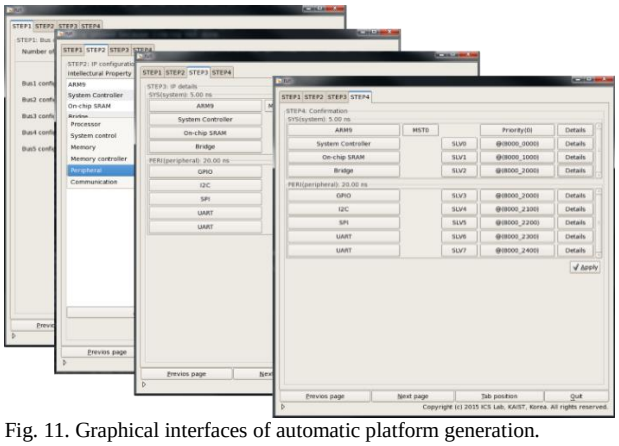

Fig. 11. Graphical interfaces of automatic platform generation.

bus hierarchy, i.e., connects the core to the system bus and some peripherals to the peripheral bus; 3) creates the address map of peripherals and defines the priority of each IP; 4) generates a synthesizable platform. As shown in Fig. 10, the outputs of the generator include synthesizable Verilog files as well as header files required for RTOS porting, and C sources for API functions, and thus form a complete ready-to-use hardware and software platform. For the sake of convenience, the graphic user interface (GUI) for the proposed generator is developed from GTK+ 2.1 library [18] based on X-window on Linux, which is exemplified in Fig. 11.

*C. Example of Application System Development*

Due to the automatic generation of a specific platform and the diverse IPs available, the SoC platform can easily support a number of applications. Fig. 12 shows a temperature sensor application as an example that can be implemented on the proposed SoC platform.

The example platform consists of the ARM9 core, main memory, system controllers, the timer, GPIO, I2C, and UART controllers. The I2C interface is used to communicate with temperature sensors. The gathered temperature data can

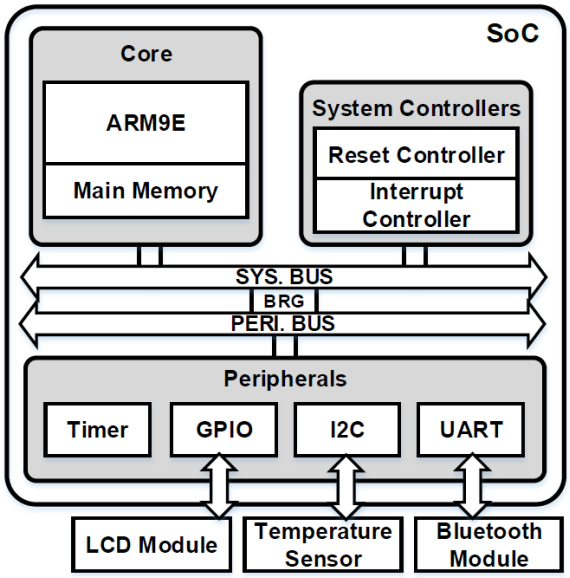


Fig. 12. A temperature sensor module.

```
/* Pseudo application code */

// The platform wakes up from low-power mode

// Temperature sensor data measurement using I2C
i2c_init (CONFIG_I2C);
i2c_recv_data (ADDRESS, NUM);

// Bluetooth communication via UART
uart_config (CONFIG_UART);
uart_transmit_data (MEASUREMENT_DATA);
...

// LDC display with GPIO
gpio_set_direction (0xFF, OUT);
gpio_send_data (MEASUREMENT_DATA);

// Set sleep timer and low-power mode
timer_set_period (55DA280); // 30 min.
timer_ctrl (COUNT_ENABLE, INT_ENABLE);
pmu_set_peripherals (0xF0);
pmu_set_mode (ACT_TO_SNOOZE);

// The platform goes back to low-power mode

/* Pseudo code ends */
```

Fig. 13. A pseudo code of the temperature sensor application.

be transferred to other devices, such as smartphones and PCs, via a Bluetooth communication module connected to the UART interface. The LCD module can also be supported by connecting it to the GPIO. The software application exploits the timer to manage dynamic power consumption according to its usage scenario. For example, the platform spends a predefined period of time in the snooze mode (e.g., 30 minutes) and wakes up in response to an expiration of the timer. Once awake, the core polls the temperature sensor and runs a short processing and sends the measurement data to the PC. After completing the routine, the core returns to the low-power mode. Fig. 13 shows a pseudo code of the example above, written with API functions.

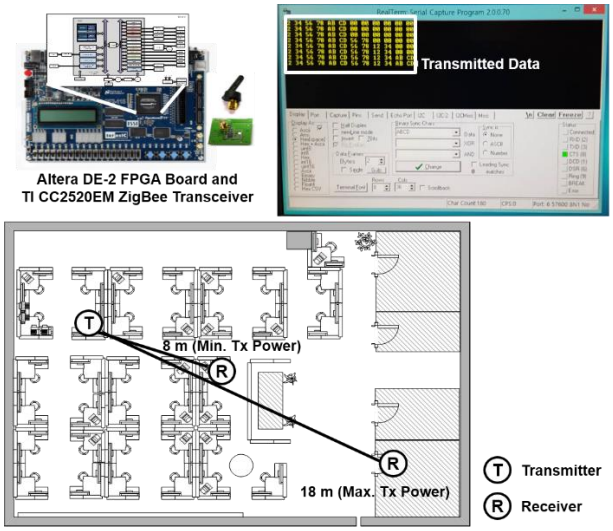


Fig. 14. ZigBee communication tests.

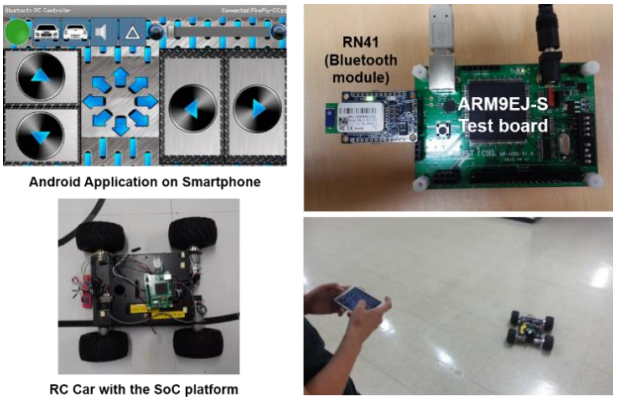


Fig. 15. RC car demonstration with the proposed SoC platform.

## IV. Implementation Results

### A. Demonstrations

The SoC easily implements a number of applications with diverse operating conditions, due to the 16-/32-bit ARM core ISAs, power management unit, and various peripheral interfaces. Note that the peripherals can be configured to play different roles. For example, the UART interfaces can either be connected to a Bluetooth module or a RS 232 module. The I2C interfaces can be used to deal with sensors, such as temperature sensor, humidity sensor, and acelero-meter. The SPI interface can also be exploited as a sensor interface or a ZigBee communication module along with the GPIO interface. To prove the usability of the proposed SoC platform, two demonstration systems, a ZigBee comm-unication module and a RC car controller, are implemented and presented below.

An indoor experiment on ZigBee communication is done using the proposed ARM9 platform. Fig. 14 illustrates the ZigBee communication test platform built with two Altera DE2 FPGA boards [20] and the TI CC2520EM ZigBee transceivers [19]. The developed ARM9 platform is mounted on each FPGA board. A series of data is sent from

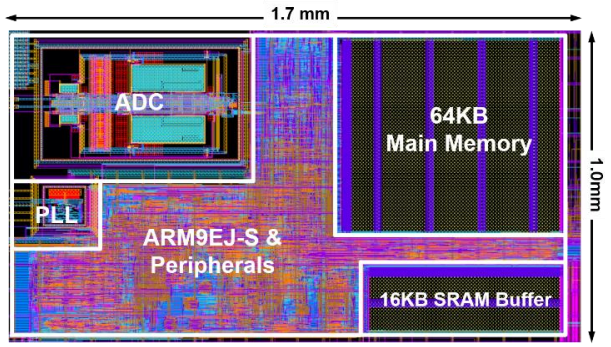


Fig. 16. Chip micrograph.

TABLE II.
Characteristics of the implemented chip

| Chip Specifications | |
|---|---|
| Technology | 65nm CMOS |
| Supply Voltage | 1.2 V |
| Chip Size | 1.70 $mm^2$ |
| Max. Frequency | 277 MHz |
| Equivalent Gate Counts | 569 k |

a transmitter and the validity of data is confirmed at a receiver. On the presence of obstacles and walls, the transmission distance ranges from 8 to 18 meters when the transmit power is minimum (16 uW) and maximum (3.2 mW), respectively.

In addition, we implemented a RC car controller with the SoC platform. An application program running on an Android smartphone is used to receive user commands in the demonstration system. Fig. 15 shows a developed SoC platform with the smartphone application. The RN41 Bluetooth module [21] is connected to the UART interface to communicate with the smartphone. Another UART interface is used to control the motor driver through a UART -to-RS232 converter [23]. The LEDs are connected to a GPIO interface to display current control status. In the demonstration, the RC car moves forward or backward and takes turns according to the user commands received at the smartphone application which in turn sends Bluetooth messages to control the peripherals of the SoC platform.

### *B. Summary*

Fig. 16 shows a micrograph of the chip fabricated in a 65nm CMOS process. The chip size is 1.7 x 1.0 $mm^2$ and the equivalent gate count is 569 K as summarized in Table II. The maximum operational frequency of the chip is 277 MHz at 1.2 V supply voltage. Table III compares the proposed platform with the state-of-the-art IoT SoC products. The proposed SoC is equipped with a suite of peripherals required for diverse IoT applications like other platforms and has the highest operating frequency among them.

## V. CONCLUSIONS

This paper has presented a low-power, high-performance SoC platform with full software supports for a variety of IoT applications. The hardware architectures are described including the core, peripherals, clock generator, system controllers, and power management unit. Software development tools and an automatic platform generator are also described, and an application system is exemplified for better understanding. To show the practicality of the proposed platform, two demonstration systems are also presented.

Based on the achievements, a multi-core version of the platform and its software supports including multi-core RTOS will be developed for high-performance applications. In addition, both system-level and circuit-level ultra low-power techniques will be studied for battery-powered applications.

TABLE III.
Comparison of the proposed platform with the state-of-the-art

| Chip Specifications | | This Work | Atmel SAM R21 [4] | TI CC2538[5] | Freescale MKW2x [6] |
|---|---|---|---|---|---|
| CPU | | 16-/32-bit ARM9EJ-S | 32-bit ARM Cortex-M0+ | 32-bit ARM Cortex-M3 | 32-bit ARM Cortex-M4 |
| Operating frequency | | 200 MHz | 32 MHz | 32 MHz | 50 MHz |
| On-chip SRAM | | 64 KB | 32/16/8 KB | 32 KB | 64 KB |
| Power management | | Y | Y | Y | Y |
| Security engines | | AES128/192/256, DES, 3DES | AES128 | AES128/256, SHA256 | AES128, DES/3DES, MD5 SHA1/256 |
| Peri. | UART | Y | Y | Y | Y |
| | SPI | Y | Y | Y | Y |
| | I2C | Y | Y | Y | Y |
| | PWM | Y | Y | Y | Y |
| | GPIO | Y | Y | Y | Y |
| | Timer | Y | Y | Y | Y |
| | Watchdog | Y | Y | Y | Y |
| | ADC | Y | Y | Y | Y |

## ACKNOWLEDGMENT

This work was supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as Global Frontier Project (CISS-2011-0031860) and IC Design Education Center (IDEC).

## REFERENCES

[1] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans. Industrial Informatics*, vol. 10, no. 4, pp. 2233-2243, Nov. 2014.
[2] S. Gangopadhyay, S. B. Nasir, A. Raychowdhury, "Integrated Power Management in IoT Devices under Wide Dynamic Ranges of Operation," in *Proc. IEEE Design Automation Conference (DAC)*, San Francisco, CA, 2015, pp. 1-6.
[3] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things J.*, vol. 1, no. 1, pp. 3-9, Feb. 2014.
[4] Atmel SAM R21E / SAM R21G Datasheet. [Online]. Available: http://www.atmel.com/images/atmel-42223-sam-r21_datasheet.pdf.
[5] Texas Instruments CC2538 Datasheet. [Online]. Available: http://www.ti.com/lit/ds/swrs096d/swrs096d.pdf.
[6] Freescale Semiconductor MKW2xDxxx Datasheet. [Online]. Available: http://www.nxp.com/files/rf_if/doc/data_sheet/MKW2xDxxx.pdf.
[7] ARM926EJ-S Revision: r0p5 Technical Reference Manual. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0198e/DDI0198E_arm926ejs_r0p5_trm.pdf
[8] *Advanced Encryption Standard (AES)*, FIPS PUB 197, 2001.
[9] *Data Encryption Standard (DES)*, FIPS PUB 46-3, 1999.
[10] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 3, pp. 299- 316, Jun. 2000.
[11] W. Dargie, "Dynamic Power Management in Wireless Sensor Networks: State-of-the-Art," *IEEE Sensors J.*, vol. 12, no. 5, pp. 1518-1527, May 2012.
[12] D. Flynn, "Power Gating Applied to MP-SoCs for Standby-Mode Power Management," in *Proc. IEEE Design Automation Conference (DAC)*, Austin, TX, 2013, pp. 1-5.
[13] FreeRTOS 7.6.0. [Online]. Available: http://www.freertos.org/.
[14] uC/OS-II 2.5.2. [Online]. Available: https://www.micrium.com/rtos/ucosii/overview/.
[15] GNU gcc v4.6.1. [Online]. Available: https://gcc.gnu.org/.
[16] GNU debugger. [Online]. Available: https://www.gnu.org/software/gdb/.
[17] GNU DDD 3.3. [Online]. Available: https://www.gnu.org/software/ddd/.
[18] GTK+ 2.1 Library. [Online]. Available: http://www.gtk.org/.
[19] TI CC2520EM ZigBee RF Transceiver. [Online]. Available: http://www.ti.com/lit/ds/swrs068/swrs068.pdf.
[20] Altera DE2 Development and Education Board. [Online]. Available: http://wl.altera.com/education/univ/materials/boards/de2/unv-de2-board.html.
[21] RN-41 Class 1 Bluetooth Module. [Online]. Available: http://www.mouser.com/catalog/specsheets/rn-41-ds-v3.3r%5B1%5D.pdf.
[22] QEMU, An open-source processor emulator. [Online]. Available: http://wiki.qemu.org/Main_Page.
[23] MAX3232. [Onlien]. Available: http://pdfserv. Maxim integrated.com/en/ds/MAX3222-MAX3241.pdf

**Hyewon Jeong** received the B.S. and M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2008 and 2012, respectively, where she is currently working toward the Ph.D. degree. Her current research interests include VLSI architectures for error-correction coding, architectures for microprocessors, HEVC video coding and associated VLSI architectures.

**Jinhwan Lee** received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2013, where he is currently working toward the M.S. degree. His current research interests include architectures for micro-processors, and VLSI designs for embedded systems.

**Hoyoung Yoo** received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea, in 2010, the M.S. degree and the ph.D. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2012 and 2015, respectively. He is currently working for the Samsung Electronics Corporation, in Hwa-sung, Korea. His current research interests include VLSI architectures for error-correction coding.

**In-Cheol Park** received the B.S. degree in electronic engineering from Seoul National University, Seoul, Korea, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1988 and 1992, respectively.

Since June 1996, he has been an Assistant Professor and is currently a Professor with the Department of Electrical Engineering, KAIST. Prior to joining KAIST, he was with the IBM T. J. Watson Research Center, Yorktown, NY, USA, from May 1995 to May 1996, where he researched high-speed circuit design. His current research interests include computer-aided design algorithms for high-level synthesis and very large scale integration architectures for general-purpose microprocessors.

Dr. Park received the Best Design Award at ASP-DAC in 1997 and the Best Paper Award at ICCD in 1999.