**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 8, No. 4, August 2018, pp. 2595~2601
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i4.pp2595-2601 

# Instruction Set Extension of a Low-End Reconfigurable Microcontroller in Bit-Sorting Implementation

**Sani Irwan Md Salim, Yewguan Soo, Sharatul Izah Samsudin**
Centre of Telecommunication Research and Innovation, Faculty of Electronic and Computer Engineering,
Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

**Article Info**

***Article history:***

Received Mar 12, 2018
Revised May 21, 2018
Accepted May 28, 2018

***Keyword:***

Instruction Set Extension
Reconfigurable Processor
Bit-Sorting

**ABSTRACT**

The microcontroller-based system is currently having a tremendous boost with the revelation of platforms such as the Internet of Things. Low-end families of microcontroller architecture are still in demand albeit less technologically advanced due to its better I/O better application and control. However, there is clearly a lack of computational capability of the low-end architecture that will affect the pre-processing stage of the received data. The purpose of this research is to combine the best feature of an 8-bit microcontroller architecture together with the computationally complex operations without incurring extra resources. The modules' integration is implemented using instruction set architecture (ISA) extension technique and is developed on the Field Programmable Gate Array (FPGA). Extensive simulations were performed with the and a comprehensive methodology is proposed. It was found that the ISA extension from 12-bit to 16-bit has produced a faster execution time with fewer resource utilization when implementing the bit-sorting algorithm. The overall development process used in this research is flexible enough for further investigation either by extending its module to more complex algorithms or evaluating other designs of its components.



***Corresponding Author:***

Sani Irwan Md Salim,
Centre of Telecommunication Research and Innovation,
Faculty of Electronic and Computer Engineering,
Universiti Teknikal Malaysia Melaka,
Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia.
Email: sani@utem.edu.my

## 1. INTRODUCTION

Embedded systems are rapidly evolving with numerous applications now are executed on its various platforms. The embedded system variant has evolved from the traditional PC-based integrated system to stand-alone and self-sustaining system with equivalent processing power and features compare to the former. As the system becomes more ubiquitous, the design process for the embedded system hardware must be in line with its software applications counterpart in order to provide the most streamlined and satisfying experience for the end-users. To achieve this, the designer must consider several performance parameters that ultimately lead to the introduction of a cluster of hardware called the reconfigurable system.

The relevance of 8-bit microcontroller has been the subject of intense debate within the computing community. Many predicted since the 90's that the decline of the 8-bit microcontroller is imminent and all applications would be based on 16-bit architecture [1] and later 32-bit architecture [2] with the emergence of the ARM processor. However, despite all things happened in the world of technology, the 8-bit microcontrollers are still in demand and being adopted in wide-ranging applications. For example, the Microchip's 8-bit device has the ability to maintain functionality with voltage varies from low-level 1.3V to TTL 5V [3] which is beneficial due to the long sleep-mode required for most of the IoT device. It is

important to note that most of the data processing tasks in an IoT system are executed in the cloud server while the IoT board only responsible for data acquisition. The IoT board is connected to various sensors and minimal pre-processing tasks are required before the data are relayed to the cloud server through the internet connection. As the IoT device transmits its data serially in a protocol, having an optimized 8-bit architecture would suit the requirement of the IoT board better than the 32-bit architecture, without all the unnecessarily peripherals that are integrated into the 32-bit architecture.

A soft-core processor is a microprocessor fully described in software which can be synthesized in programmable hardware, such as FPGAs. Soft-core processors implemented in FPGAs can be easily customized to the needs of a specific target application. A microprocessor defined in software code usually in hardware description language (HDL). As the processors are available in programming code, the developers are able to modify and reuse pre-designed hardware components in the form of intellectual property (IP) cores. This method would reduce design time at the expense of area and performance penalty.

In general, having a soft-core processor provide several advantages in embedded system development. Soft-core processors have a higher level of abstraction that makes it easier to understand. Flexibility is also offered to the developer that enable the core to be edited and changed the parameters. Notable applications of the soft-core processor are in motor control system [4], pulse width modulation (PWM) generation [5] and cryptographic algorithm [6]. However, there is very little detail of the implementation and choices made during the development process i.e. design decisions and performance trade-off [7], [8]. There is the need to established a methodology that promotes incremental design improvement for a system that is based on the soft-core processor.

UTeMRISC0 [9, 10] is a softcore processor that is essentially based on the PIC architecture, that is also the fundamental architecture of several other soft-core processors previously mentioned in this thesis. The UTeMRISC0 soft-core processor was first introduced in [11] as a part of the experimental setup to maximise the capability in delivering highest performance achievable by a processor core compared to their physical IC counterpart. The previous PIC-based soft-core processors are heavily reliant to the MPlab to compile and generate the processor's bit file. For UTeMRISC0 processor, a retargetable assembler is designed exclusively from the ground up to replace the MPlab as the de-facto assembler. In the early phase, the assembler is designed to match the existing instruction set in the PIC data sheet within a simulated computer architecture environment called the CPUSim.

One of the prominent paper with regards to the instruction set extension on reconfigurable processors was published by [12]. The overall target for manipulating the instruction set is to achieve higher parallelism to the instruction execution. Over the years, different techniques to increase the level of parallelism have been introduced at instruction level: for instance, techniques such as instruction pipelining, superscalar execution, out-of-order execution, and register renaming. Parallelism has also been exploited at other levels: bit-level, data-level, and loop-level parallelism. Although the level of parallelism has been increased over the years, it is still relatively limited for highly parallelizable applications, which become poor candidates for implementation of these architectures.

## 2. RESEARCH METHOD

The instruction set extension technique applied in this research is not necessarily focused on adding new instructions to the instruction set [13]. It is also could encompass modification to the current instruction, altering the behavior of the available instruction to perform a command that is totally different from its original purpose and also omitting the unrelated instructions that are not involved in the specific application. These customizations are done through manual configuration [14].

Manual extensions involved a degree of human effort to identify and implement the instruction set extensions. Human ingenuity in the manual creation of custom capabilities creates high-quality results which within the context of the application's result. As the design grows more complex and constraint on the time-to-market requirement, automatic design flow is desirable for the use of these new capabilities. For a large set of instruction candidates, selection of multiple custom instructions involves complex trade-off and can be hard to execute manually. However, as this research is focusing on low-end digital signal controller platform, the custom instructions that are manually identified and created. These instructions are considered sufficient to perform DSP tasks successfully within the hardware limitation.

### 2.1. 12-bit ISA to 16-bit ISA

Using the original 12-bit ISA [15] as the reference point, the ISA is extended to become 16-bit in width that in the end becomes the configured ISA for the UTeMRISC01 microcontroller. The customized instruction set is created by expanding the available instruction tokens to vacate more space to execute more data-transfer tasks such as register swapping. Figure 1 shows the instruction format for both the 12-bit ISA

and the 16-bit ISA with three different modes; (1) Literal operation, (2) byte-oriented operation with direction and (3) bit-oriented operations.

The new 16-bit ISA now comprised of 6-bit of the opcode and 10 bits of the operand. The total number of 16 is chosen as it is byte addressable and could easily accommodate more bits for future expansion. Opcode bit is set to 6 bits that could stretch the maximum number of instructions to up to 64 instructions. The file register address is extended to 7 bits that enable theoretically up to 128 registers that can be addressed by each instruction. The literal value maintained at 8-bit of raw data with a 3-bit select bit to point to any position in the 8-bit data. Finally, the single directional bit is used to indicate the destination of the data either to the working register or the register that is addressed in the operand. Technically, the 16-bit ISA provide the best setup for the low-end digital signal controller whereby there a lot of room to exploit in adding new instructions, expanding the number of registers and also offered extra bits for literal values for precision and sign extended data.

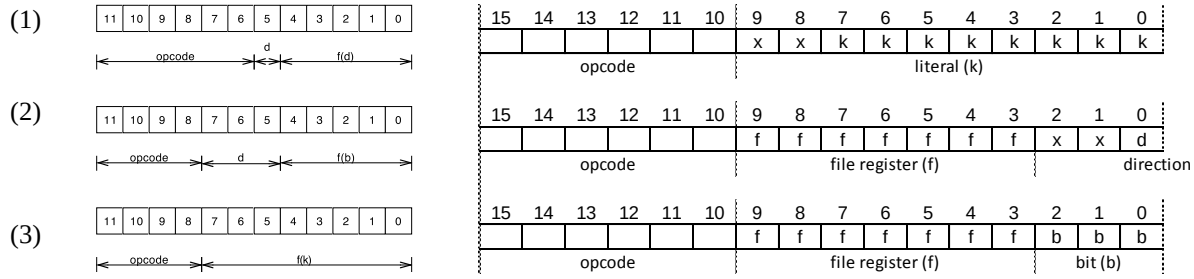



Figure 1. Conversions from 12-bit ISA in (a) to 16-bit ISA in (b) for all instruction types

Compiler generator or retargetable code generator is used to synthesize code for the particular applications. Once the instruction set is generated, the retargetable assembler is developed to accommodate all changes made in the instruction set. The assembler then is responsible to decode and generate the object code of the program based on the new instruction set [16]-[18].

### 2.2. Workflow and related tools

Specific tasks and procedures are followed to ensure smooth transactions as well as getting the right tools. The UTeMRISC01 processor is implemented in the Xilinx ML605 FPGA board and the retargetable assembler is developed from the ground up using the Visual Basic software by Microsoft. Once implemented, the performance parameters such as resource utilization and the execution times are obtained through the synthesis and Post-PAR timing report. Further measurements are also conducted using the logic analyzer to verify the execution times of the bit-sorting program.

During the early phases of the design, the new instruction is tested and simulated using the CPUSim software. CPU Sim is a Java application that allows users to design simple computer CPUs at the microcode level and to run the machine-language or the assembly-language programs on those CPUs through simulation [19].

The tools used in the hardware implementation phase are exclusively done on Xilinx environment, including its Integrated Synthesis Environment (ISE) Design Suite. The Xilinx ISE Design Suite is responsible for the bulk of the work by providing the tools to develop HDL modules in the processor architecture. The behavioral simulations are completed by using the ISE simulators, also known as ISim, which its main purpose is to perform functional and timing simulations for Verilog designs. Xilinx ISE will perform the logic synthesis and design implementation to generate the bit file. Using a software called iMPACT (integrated with the Xilinx ISE), the bit file is configured and loaded to the FPGA board, in this case, the ML605 FPGA board).

### 2.3. Bit-sorting algorithm

There are various methods of sorting, such as interchange or bubble sort, shell sort, bucket sort and radix interchange sort. Bubble sort is one of the popular methods, although the execution is not very efficient. The key idea of bubble sort is to take adjacent pairs of elements in the list of data and put them in order by interchanging their positions when it is necessary. By continuously executing the interchanging operations,

the data are sorted ascendingly within the data range. Figure 2 shows the step-by-step implementation flow of the bit sorting algorithm.

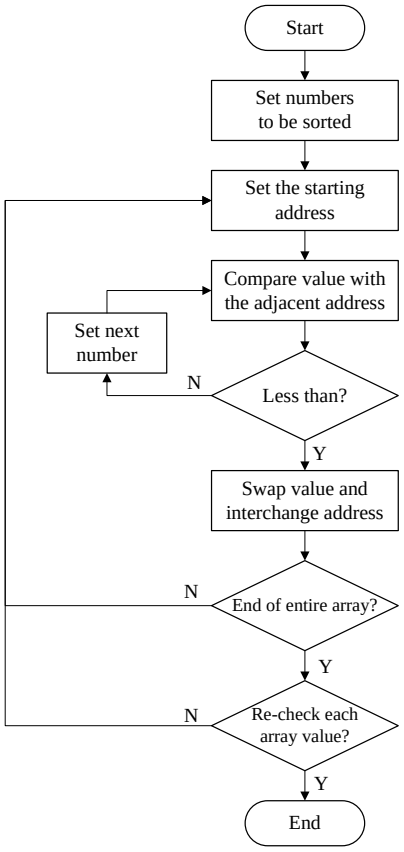



Figure 2. Flowchart for bit-sorting algorithm

## 3. RESULTS AND ANALYSIS

Establishing the benchmark of the soft-core processor is vital at this stage. The performance parameters for the modified architecture are evaluated and compared with the performance of the original processor architecture. In this case, the 8-bit soft-core processor called UTeMRISC0 resemble greatly to its packaged-IC counterpart which is the PIC16C57. The UTeMRISC0 will be the starting point for the architecture expansion and modification.

The instruction set architecture extension is performed on the UTeMRISC0 architecture. Originally, the ISA consisted of 12-bit and is expanded to become 16-bit. The expansion does not solely involve the lengthening of the ISA register. It also required further modifications to other modules and registers such as the instruction decoder module, the ROM module and instruction registers. New opcodes are also assigned to the instruction set. At this point, the original list of instruction is maintained as per original while a single new instruction called 'swapfw' is created to demonstrate the capability of the architecture in handling and executing the new instruction. To reflect the major changes being made due to the ISA expansion, the modified processor now is called UTeMRISC01 [20], [21].

### 3.1. Instruction set generation

A new instruction is created and called 'swapfw' that would take both of its operands as register address. To optimize the instruction set, the 'swapfw' instruction would occupy the space for another instruction called 'movf f,1'. Originally, this instruction only functions to transfer data from a register back to its own address which it impractical unless a certain condition bit is required to be generated. Therefore, the new 'swapfw' instruction is assigned to the previous opcode allocation but now with different function, as shown in Table 1.

Table 1. New Instruction for Bit-Sorting

| Opcode | CPU Sim Mnemonics | Format | 16-bit Opcode |
|---|---|---|---|
| 0B | swapfw f | 6 7 -3 | 0010_11ff_ffff_fxxx |

### 3.2. Code synthesis

Referring to Figure 3, the new instruction the is simulated using the CPUSim software with the UTeMRISC01 architecture is loaded as the main processor execution. The new instruction's micro-operation is defined in this process to iterate the sequence of the instruction execution**.**

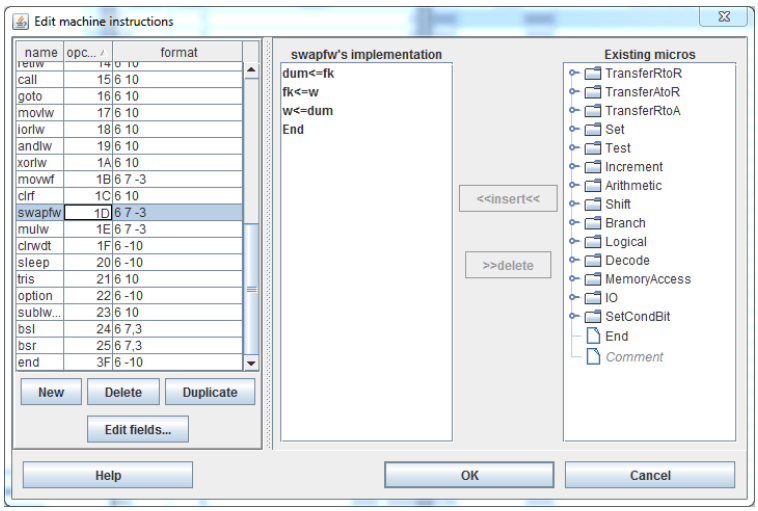



Figure 3. Machine instruction editing in CPUSim

### 3.3. Hardware synthesis

The findings from the code synthesis stage are brought forward to the FPGA implementation by replicating the instruction execution sequence, this time in Verilog code. The instruction's opcode and ISA are updated in the instruction decoder and ALU module. Once the architecture is successfully synthesized and implemented, the performance parameters are observed and measured using the logic analyzer.

When the UTeMRISC01 architecture is implemented for the bit-sorting program, an increase of registers and LUTs have been recorded. As the UTeMRISC01 are equipped with the new ISA and an improved instruction register, the increase in the slice register number is expected, as shown in Table 2. Further optimization of the ALU module, especially in reducing the redundant instructions also contributed to a smaller usage of slice LUTs. For the bit sorting program, the new instruction is introduced by overwriting an unused instruction and this ensures the overhead of resource utilization is kept to a minimum. The new 'swap' instruction only involved register data transfer, which is already embedded in the original architecture, hence there is no additional hardware is required to execute the instruction.

Table 2. Resource Utilization for Bit-Sorting Program

| Parameter | UTeMRISC0 (12-bit ISA) | UTeMRISC01 (16-bit ISA) | Difference |
|---|---|---|---|
| Number of Slice registers | 102 | 119 | ⇧16.7% |
| Number of Slice LUTs | 457 | 355 | ⇩22.3% |
| Number of LUT Flip Flop pairs used | 494 | 409 | ⇩17.2% |

In terms of execution times, the UTeMRISC01 architecture consistently produced lower execution times throughout the bit-sorting program implementation. Therefore, the program is ended much quicker, as shown in Figure 4. By calculations, the UTeMRISC01 architecture could finish executing the same bit-sorting program faster than the UTeMRSIC0 architecture by 12%. This is clearly due to the lower instruction cycle, resulted in the introduction of the new swap instruction.

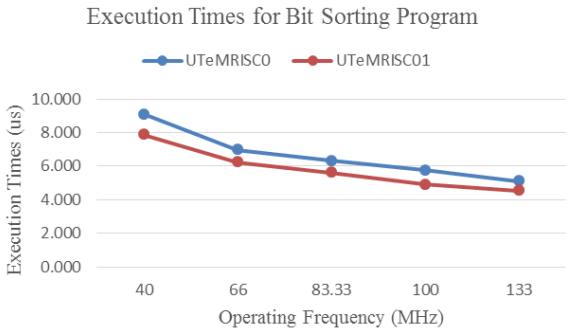



Figure 4. Execution times for bit-sorting program

## 4. CONCLUSION

In the bit-sorting programs, the number of registers in the UTeMRISC01 is reduced due to several optimization steps are done to the original UTeMRISC0 architecture. There is no new module added to the architecture and the new instruction is introduced by occupying the existing instruction slot in the instruction set. The execution times also improved that makes the program finished earlier. The reduced time is not because of the less-complexity of the bit-sorting program or the architecture itself but it is due to the efficiency of the new instruction that replaced the repetitive instruction previously existed in the original architecture. It is demonstrated here that the adoption of instruction set extension technique on the UTeMRISC01 architecture has optimized the processor capability in dealing with a more complex task. The overall development process used in this research is flexible enough for further investigation either by extending its module or to adopt more complex algorithms in the future. While the focus of this research is to reduce the execution times and the reduce utilization, power consumption will be considered in the future iterations of the processor.

## ACKNOWLEDGEMENTS

The author would like to thank Universiti Teknikal Malaysia Melaka and Ministry of Higher Education Malaysia for the financial support through the research grant number PJP / 2016 / FKEKK-CETRI / S01496.

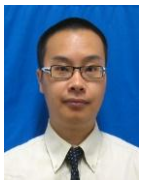
## REFERENCES

[1] R. Cravotta. (2012, 11 July). *One Processor to Rule Them All?* Available: http://www.edn.com/design/systems-design/4398890/One-processor-to-rule-them-all
[2] J. Ganssle. (2012, 2 July). *Is 8-bits dying?* Available: http://www.embedded.com/electronics-blogs/break-points/4389890/Is-8-bits-dying-
[3] Microchip, "8-bit PIC® Microcontroller Solutions," Microchip Technology Incorporated 2014.
[4] T. Sutikno, N. R. N. Idris, A. Z. Jidin, and A. Jidin, "A Model of FPGA-based Direct Torque Controller," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 11, pp. 747-753, 2013.
[5] A. Zemmouri, R. Elgouri, M. Alareqi, M. Benbrahim, and L. Hlou, "Design and Implementation of Pulse Width Modulation Using Hardware/Software MicroBlaze Soft-Core," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 8, pp. 167-175, 2017.
[6] C. Kannan, "NIOS II Based Secure Test Wrapper Design for Testing Cryptographic Algorithms," *International Journal of Reconfigurable and Embedded Systems*, vol. 4, 2015.
[7] F. Plavec, B. Fort, Z. G. Vranesic, and S. D. Brown, "Experiences with Soft-Core Processor Design," in *IPDPS*, 2005, p. 167.2.
[8] F. Plavec, "Soft-core processor design," Master of Applied Science, Graduate Department of Electrical and Computer Engineering, University of Toronto, 2004.
[9] A. J. Salim, S. I. M. Salim, N. R. Samsudin, and Y. Soo, "Conversion of an 8-bit to a 16-bit Soft-core RISC Processor," *International Journal of Electronics Communication and Computer Technology*, vol. 3, pp. 393-397, 2013.
[10] N. R. Samsudin, S. I. M. Salim, and A. J. Salim, "Designing UTeMRISCII Processor for Multiply-Accumulate Operation," in *3rd International Conference on Engineering and ICT (ICEI2012)*, 2012, pp. 88-91.
[11] L. E. Yong and A. J. Salim, "Implementation of an 8-bit RISC Microcontroller Chip," in *4th International Symposium on Broadband Communication*, 2010, pp. 1-4.

[12] C. Galuzzi and K. Bertels, "The Instruction-Set Extension Problem: A Survey," *ACM Trans Reconfigurable Technol. Syst.*, vol. 4, pp. 1-28, 2011.
[13] A. J. Salim, S. I. M. Salim, N. R. Samsudin, and Y. Soo, "Instruction Set Extension Through Partial Customization of Low-End RISC Processor," *Australian Journal of Basic and Applied Sciences*, vol. 7, pp. 678-687, 2013.
[14] D. Liu, *Embedded DSP Processor Design, : Application Specific Instruction Set Processors*: Morgan Kaufmann, 2008.
[15] T. Coonan. (1999, 4 January). *RISC8 Verilog Core*. Available: http://pldworld.info/_hdl/2/_ip/mindspring/~tcoonan/risc8doc.html
[16] S. I. M. Salim, H. A. Sulaiman, R. Jamaluddin, L. Salahuddin, M. N. S. Zainudin, and A. J. Salim, "Two-pass assembler design for a reconfigurable RISC processor," in *IEEE Conference on Open Systems (ICOS)*, 2013, pp. 77-82.
[17] S. I. M. Salim, H. A. Sulaiman, R. Jamaluddin, L. Salehuddin, M. N. S. Zainudin, and S. Yewguan, "Assembler Design Techniques for A Reconfigurable Soft-Core Processor," *Journal of Theoretical and Applied Information Technology*, vol. 64, 2014.
[18] S. I. M. Salim, H. A. Sulaiman, M. N. S. Zainudin, R. Jamaluddin, and L. Salahuddin, "One-pass assembler design for a low-end reconfigurable RISC processor," in *International Symposium onTechnology Management and Emerging Technologies (ISTMET)*, 2014, pp. 492-496.
[19] D. Skrien, "CPU Sim 3.1: A Tool for Simulating Computer Architectures for Computer Organization Classes," *Journal on Educational Resources in Computing (JERIC)*, vol. 1, pp. 46-59, 2001.
[20] A. J. Salim, N. R. Samsudin, S. I. M. Salim, and S. Yewguan, "Modification of Instruction Set Architecture in a UTeMRISCII Processor," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 4, pp. 1196-1201, 2013.
[21] A. J. Salim, N. R. Samsudin, S. I. M. Salim, and S. Yewguan, "Multiply-accumulate instruction set extension in a soft-core RISC Processor," in *10th IEEE International Conference on Semiconductor Electronics (ICSE)*, 2012, pp. 512-516.

## BIOGRAPHIES OF AUTHORS

**Sani Irwan Md Salim** obtained his first degree in B.Eng (Electronic Engineering), from the Universiti Teknologi Malaysia (UTM), in 2002. Following that, he obtained his M.EngSc (Computer & Communication) from Queensland University of Technology (QUT), Australia. He is currently working as senior lecturer in the Computer Engineering Department, Faculty of Electronic and Computer Engineering, UTeM. His main research area is in reconfigurable computing with interest in FPGA applications.



**Dr Soo Yew Guan** joins Universiti Teknikal Malaysia Melaka (UTeM) as academician since 2001. Currently he is the Senior Lecturer in Microprocessor and Microcontroller Technology in the Faculty of Electronics and Computer Engineering (FKEKK). As a professional trainer in Embedded System, he has conducted CAN Bus Communication in in Embedded System training to Western Digital (M) Sdn Bhd and Internet of Thing (IoT) programs in Faculty. His research interests include Embedded System and Robotics, particularly on the myo-electric robotic prosthesis and sensory substitution for amputees. In addition, his is also interested in software development for various mobile platforms (iOS, Win8 Phone, and Android) and Internet Technologies (Linux, Apache, MySQL, PHP, and JSON**).**

**Dr Sharatul Izah Samsudin** graduated for her his first degree in B.Eng (Electronic Engineering) at Universiti Sains Malaysia (USM). Following that, she obtained her M.Eng. and PhD from Universiti Teknologi Malaysia (UTM). She specializes on control system modeling with interest in water quality control environment and IoT applications. She is currently working as a senior lecturer in the Industrial Electronic Department, Faculty of Electronic and Computer Engineering, UTeM.