Overview of an Adaptive Approach for Implementing RTOS in Hardware

Tetsuo Miyauchi[1,a)] Kiyofumi Tanaka[1,b)]

Abstract: In recent years, along with a growth of IoT (Internet of Things), many embedded devices are equipped with processors/controllers, where a real-time OS (RTOS) is accommodated to make full use of complicated functions of the devices. It is desired that RTOS runs fast with as small memory usage as possible since it is overhead from an application program's viewpoint. Therefore, it is expected that providing hardware for a part of RTOS processing reduces memory usage while it makes the processing fast. Under the circumstances where several examples of hardware implementations of RTOS are found, we implement functions of the μITRON specification in FPGA hardware.
In addition, we propose an approach to adapting it to applications' requirement.

Keywords: RTOS,adaptive approach,μITRON,FPGA

1. Introduction

Along with the popularization of IoT (Internet of Things), microprocessors are more than ever being embedded in lots of appliances, which have complicated functions with communication. In order to implement microprocessors in various things, cost is one of the most important factors. For reducing the cost, it is desirable that processing resources which software and hardware use should be reduced as much as possible while functions to be provided and performance are maintained.

From the viewpoint of embedded system development, RTOS (Real-Time Operating System) is commonly used to make developing a system with strict time constraint more efficient.

Nevertheless, as an RTOS kernel itself is just overhead for an application program, the smaller footprints of an RTOS kernel mean the better implementation and it is desirable that execution time is short enough.

It is expected that implementing RTOS functions in hardware can make it possible to reduce software code size and shorten system call execution time. In our approach, RTOS kernel functions for μITRON4.0 specification are implemented in an FPGA. Compared with a full software RTOS, we aim at reducing software code size and execution time.

While there are several studies for implementing RTOS functions in hardware ([1], [2],[3],[4]), characteristics of our approach are: building RTOS functions, removing error checking functions in RTOS system calls if possible and deleting hardware functions for unused system calls. We show that this approach can be implemented in an FPGA and evaluate the number of hardware resources used in an FPGA, software code sizes, and execution time for system calls, when the system is adapted to an application program so that unused functions are eliminated.

[1] Japan Advanced Institute of Science and Technology, Asahidai 1-1, Nomi, Ishikawa, 923–1292, Japan
[a)] t-miyauc@jaist.ac.jp
[b)] kiyofumi@jaist.ac.jp

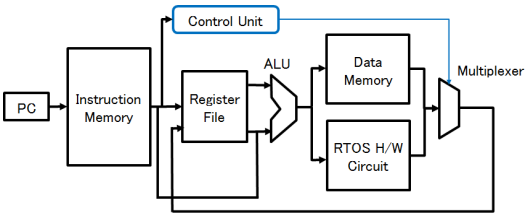

Fig. 1 Processor structure with RTOS hardware.

2. Implementation

2.1 Hardware Structure

Hardware structure which we have implemented is explained below. We have implemented principal functions of the μITRON4.0[5] standard profile specification. Figure 1 shows a structure of a processor core and RTOS hardware circuit we implemented.

RTOS hardware proposed in this paper consists of not only primitive functions but also all functions including error checking in system calls. Therefore, compared with software-only RTOS implementation, execution time of an RTOS system call can be reduced and the software code size can be decreased.

Figure 2 shows an RTOS hardware structure. In this figure, the part of the "RTOS Hardware Core" is the fundamental functions related to TCB (Task Control Block) queue operations.

An RTOS hardware operation command and data are delivered to the "RTOS Hardware Wrapper" part by a software program in a processor. RTOS hardware operation command is designated with a memory address of a memory reference instruction and an RTOS hardware operation is decided with a pair of an address and data.

2.2 Software Structure

The software reads from or writes to a specific address to use hardware. Figure 3 is a flow in the software-side processing for act_tsk(). Other system call functions follow a similar flow.

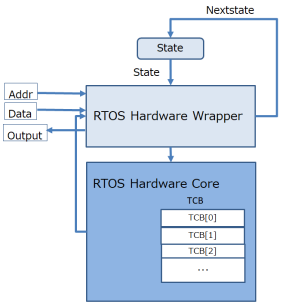

Fig. 2 RTOS Hardware Structure.

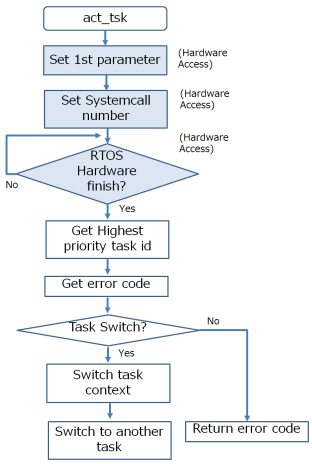

Fig. 3 Systemcall Software flow.

3. Evaluation

The processor core and RTOS hardware described in the previous section are implemented in an FPGA, Xilinx Spartan-6 (XC6SLX16CSG324C) [7] with the evaluation board of Digilent NEXYS3. The processor core runs at 50MHz in the FPGA and executes MIPS instruction set [6].

Table 1 and Table 2 show the number of resources occupied by the processor core and RTOS hardware and minimum period/maximum clock frequency for the configuration with error checking including five tasks, four semaphores and three event flags, and with/without error checking, with five tasks, four semaphore and no eventflags, respectively.

Table 1 FPGA Resources (Full)

Resources	# of Used Resources	Usage (%)
Register	2076	11%
LUT	4699	51%
Slice	1444	63%
Minimum period	19.972ns	
Maximum frequency	50.07MHz	

Table 2 FPGA Resources (Semaphore w/ and w/o Error Check)

	w/ Error Check		w/o Error Check	
Resources	# of Used Resources	Usage (%)	# of Used Resources	Usage (%)
Register	1431	7%	1405	7%
LUT	3601	39%	3538	38%
Slice	1111	48%	1085	47%
Minimum period	18.583ns		17.655ns	
Maximum frequency	53.813MHz		56.641MHz	

Table 3 shows the sizes of binary codes of system calls and common routines. “Soft Only” means the software-implemented RTOS kernel. “With Hard” means the proposed implementation where the main processing for the RTOS kernel is preformed by the hardware. Since the main processing is hidden by the RTOS hardware, the size of the software-side system call is reduced. From the table, it is confirmed that the hardware implementation reduces the code sizes of all system calls and common functions.

Execution time for each system call is shown in Table 4. Since execution of system calls can involve task switching, the table includes execution times with task switching and without it.

Table 3 RTOS Kernel Software Size (bytes)

Systemcall	Soft Only	With Hard	Hard/Soft
act_tsk	416	256	61.5%
chg_pri	1008	272	27.0%
ter_tsk	1248	640	51.3 %
rel_wai	720	288	40.0%
sig_sem	528	304	57.6%
wai_sem	672	336	50.0%
pol_sem	304	256	84.2%
set_flg	704	368	52.3%
wai_flg	864	464	53.7%
pol_flg	432	336	77.8%
Soft Kernel	1280	0	0%
Common Routine	1184	1152	97.3%
Total	9360	4672	49.9%

Table 4 Execution Time for System Calls (μsec@50MHz)

Systemcall	Task switch	Soft (μsec)	Hard (μsec)
sig_sem	×	2.3	2.0
sig_sem	○	6.5	3.9
wai_sem	×	1.8	1.8
wai_sem	○	7.0	3.7
pol_sem	×	1.8	1.5
set_flg	×	2.4	2.1
set_flg	○	8.2	4.2
wai_flg	×	2.4	2.3
wai_flg	○	7.5	4.1
pol_flg	×	2.4	2.1
Average	w/o switch	2.2	2.0
	w/ switch	7.3	4.0

4. Conclusion

We presented the hardware implementation of an RTOS kernel based on μITRON 4.0, where functions of system calls including error checking are built in FPGA hardware resources. In the system, the RTOS kernel functions are invoked via memory reference instructions of a processor core with the MIPS instruction set. The hardware-implemented RTOS functions are selectable not only on a function basis but on a finer-unit basis, e.g., error-checking code fragment, which enables the system to adapt to the application codes and reduce the usage of hardware resources.

In the evaluation, it is confirmed that the hardware implementation proposed in this paper can simplify the software processing and reduce the size of software as well as the execution times. In addition, the results show that the proposed strategy can further reduce the hardware amount according to the application program by providing only functions/mechanisms required by it.

References

[1] A.B. Lange, K.H. Andersen, U.P. Schultz, A.S. Sorensen: HartOS – a Hardware Implemented RTOS for Hard Real-Time Applications, 11th IFAC, IEEE International Conference on Programmable Devices and Embedded Systems, Volume 45, Issue 7, pp. 207–213, 2012.

[2] N. Maruyama, T. Ishihara, H. Yasuura, An RTOS in Hardware for Energy Efficient Software-based TCP/IP Processing, IEEE 8th Symposium on Application Specific Processors (SASP), 2010.

[3] H. Mori, K. Sakamaki, H. Shigematsu, Hardware Implementation of a real-time operating system for embedded control systems, Tokyo Metropolitan Industrial Technology Bulletin of Study No.8, pp55–58, 2005. (In Japanese)

[4] T. Nakano, A. Utama, M. Itabashi, A. Shiomi, M. Imai, VLSI Implementation and Evaluation of a Real-Time Operating System, IEICE Trans. Inf.&Syst. (Japanese Edition) Vol.J78-D1 No.8, pp.679–686, 1995 (In Japanese)

[5] μITRON4.0 Specification Ver.4.01.00, ITRON Committee, TRON ASSOCIATION.

[6] MIPS® Architecture For Programmers, Volume II-A: The MIPS32® Instruction Set.

[7] ”Spartan6” [Online] Available `http://www.xilinx.com/products/silicondevices/fpga/spartan-6.html`