Development and Implementation of OS Functions for a Computer System Having FPGA Devices as Reconfigurable Resources

Kazuya Tokunaga, Akira Kojima and Tetsuo Hironaka
Graduate School of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima, 731-3194, Japan
E-mail : {tokunaga,kojima,hironaka}@csys.ce.hiroshima-cu.ac.jp

Abstract: Many researchers and companies have been interested in computer systems which have reconfigurable devices as programmable resources. Current Reconfigurable systems (RC systems) are basically single task systems, but the number of reconfigurable devices in these systems has increased so that multitasking environment for the systems are required to use the devices efficiently. In this paper, we develop the OS functions to support executing user programs which use FPGAs as customized hardware in multitasking environment. The target system consists of a host compute and FPGAs. The functions are implemented on the target system as RC-OS server. RC-OS server supports management of FPGA resources and communication control between host computer and FPGA resources. We develop the programs which use the RC-OS functions, and executed them in multitasking environment on our prototype RC system. As the result, it shows that RC-OS server improves the usability of target system.

1. Introduction

Reconfigurable Systems based on FPGA have been studied and developed by many researchers and companies in these days [1] [2] [3] [4]. The RC systems manage FPGA devices as reconfigurable resources (FPGA resources). These systems can execute various hardware/software (HW/SW) codesign programs by usign FPGA resources as customized HW. The resource management of the existing RC systems focuses on high speed execution for one program. However, it is difficult to keep system utilization efficiency high in the situation with running just single program, because many FPGAs are enough large to configure circuits in many cases and the parts of FPGA resources have much idle time. To improve the system utilization efficiency, multitasking technique is used in traditional computer systems, so we attempt to adopt the multitasking environment to the RC systems. To realize the environment, OS functions are developed which are called RC-OS functions. The major features of the RC-OS functions are the followings:

1. Multitasking and multiuser environment
2. Easiness of developing program
3. Management of reconfigurable resources

In this paper, we implement RC-OS server which has these functions on a prototype RC system. Further, we evaluate the performance of the RC server in multitasking environment.

2. Target System and Program model

The target system consists of a host computer, which has a microprocessor to execute software processes, and several FPGAs to execute hardware processes(Fig.1). In the system, each FPGA is managed as one FPGA resource. The program executed on the system is a HW/SW codesign program which includes the part to use one or more FPGAs as customized HW. The construction of program is shown in Fig.2. The part of HW process in the executed program is called “HW task” and the part of SW process is called “SW task.” The pro-

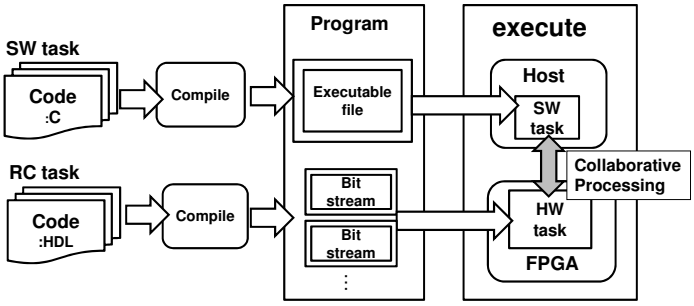

Figure 2. Construction of program

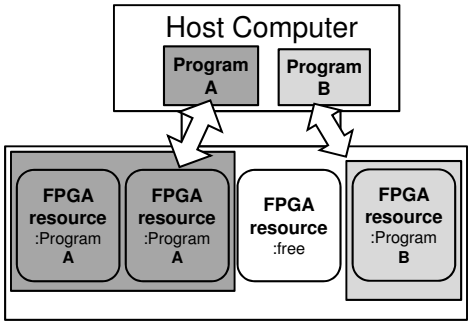

Figure 1. Programs on the target system

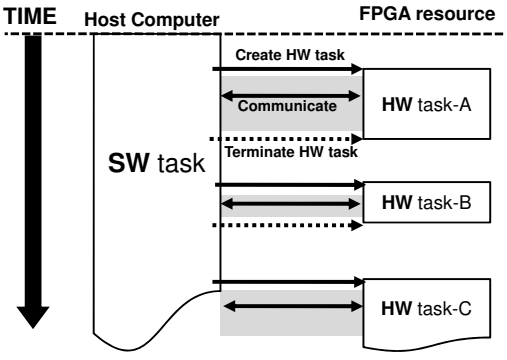

Figure 3. Behavior of program on the target system

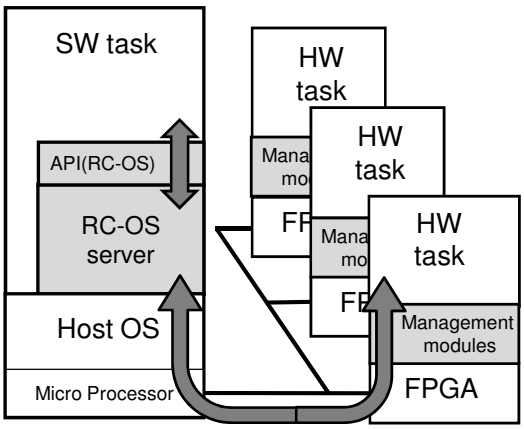

Figure 4. RC-OS server in relation to the system

grams executed on the target system can create multiple HW tasks. For example, program A in Fig.1 runs two HW tasks simultaneously using two FPGAs. On the other hand, program can run multiple HW tasks just one FPGA resource. Fig.3 shows the behavior of a program along the timeline, which runs multiple HW tasks on one FPGA resource in different times.

3. The RC-OS Server

We implement an OS server program which is called RC-OS server and provides the RC-OS functions on the target system. The RC-OS server runs at the host computer, supports collaborative processing between HW task and SW task (Fig.4), and provides the multitasking environment to programs. RC-OS server manages and controls all FPGA resources in the target system, and does not allow direct access to the FPGA resources by SW task in programs. To access the FPGA resources, the SW task needs to send request to the RC-OS server by using system calls. To provide multitasking environment for programs, RC-OS server has several functions. These functions are shown in 3.1. APIs and HDL templates which called HW task frame are introduced to use RC-OS server functions easily. These are explained in 3.2.

3.1. RC-OS server functions

RC-OS server has following functions to provide multitasking environment and manage FPGA resources.

3.1.1. Management of FPGA resources

This function manages the status of FPGA resources by using the FPGA resource status table shown in Fig.5. The FPGA resource status table manages FPGA resource number, SW task's process ID (PID) which has right to use FPGA resource, status of FPGA resource, and status of HW task configured on FPGA resource. FPGA resource status has three kinds of statuses.

a)RUNNING Created HW task is running on FPGA.
b)ALLOCATED FPGA is allocated, but HW task has not been created yet.
c)FREE FPGA is not allocated to any program.

FPGA NO	PID	Resource status	HW task status
1	230	running	sendable
2	-	free	-
3	245	allocated	-
4	-	free	-

Figure 5. FPGA resource status table

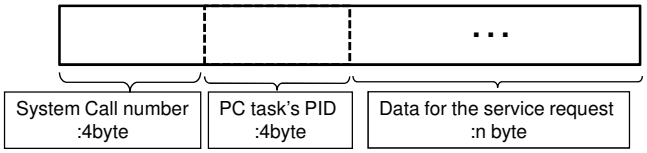

Figure 6. System call format

HW task status has four kinds of statuses.

a)SENDABLE Having data to send RC-OS server
b)RECIEVABLE Waiting data from RC-OS server
c)RUNNING HW task is running
d)STOP HW task is stopped

The FPGA resource status table is also used by other functions.

3.1.2. Acceptance of service requests from programs

The RC-OS server accepts requests from the programs. When a program requests the services to RC-OS server, system call is sent from SW task to RC-OS server. System call consists of system call number, process ID (PID) of SW task which sends system call, and data for the service request. The format of the system call is shown in Fig.6

3.1.3. Scheduling of service requests from programs

System calls are sent from multiple SW tasks in multitasking environment. This function accepts system calls from multiple SW tasks and schedules to execute them. The scheduling algorithm is FIFO, and the system call are processed sequentially.

3.1.4. Execlusive control for FPGA resource

RC-OS server manages the access right to FPGA resources of program by using "FPGA NO" and "PID" in FPGA resource status table. When RC-OS server executes the system call to control FPGA resource, RC-OS server checks whether it has access right to FPGA resources or not. This function provides exclusive control for FPGA resources in multitasking environment.

3.1.5. Allocation of FPGA resources

When programs request the RC-OS server to allocate or free FPGA resources, RC-OS server controls resource statuses in the FPGA resource status table. When the program requests

RC_alloc();
Allocate a usable FPGA resource.

RC_free(Resource_no);
Free the held FPGA resource.

RC_create(Resource_no, RC_task);
Create HW task on the FPGA resource.
RC_task in API's argument show the absolute path of HW task data.

RC_write(Resource_no, data, data_size);
Send data from SW task to HW task.

RC_read(Resource_no, data, data_size);
Send data from HW task to SW task.

Figure 7. APIs

```
1  : int  main(){
2  :    int source_no;
3  :         char  send_data[]= “abc”;
4  :
5  :   RC_connect_server();
6  :   source_no  = RC_alloc();
7  :   RC_create( source_no,  file_place );
8  :
9  :   RC_write( source_no,  send_data , 1);
10:
11:   RC_free(source_no);
12:   RC_unconnect_server();
13:   return 0;
14: }
```

Connect RC-OS server

HW task filename

Unconnect RC-OS server

Figure 8. Sample program using APIs

allocation of FPGA resource, RC-OS server find a usable FPGA resource and allocates it.

3.1.6. Creation and management of HW task

This function creates HW tasks on the FPGA resources, and manages them. When HW task is created, the RC-OS server launches configuration tool provided by FPGA bender. To manage the status of HW task, the programmer inserts a hardware module to his hardware description. We prepare HDL templates to support describing the hardware module which manages HW task status, called as HW task control module. The HW task control module has a resister which shows HW task status. The RC-OS server gets HW task status from the resister by polling and manages the status by writing it to the FPGA resource status table.

3.1.7. Communication control

When a program transfers data between HW task and SW task, this function synchronizes them, and transfers the data from the sender to the receiver properly. Programmer inserts a hardware module to manage the synchronization of HW task and SW task in his hardware description. We introduce connection control module into the HDL templates. When the SW task communicates with the HW task, the RC-OS server checks the HW task status in FPGA resource status table. If the HW task status is receivable or sendable, the RC-OS server synchronizes connection control module and communicate with it.

3.2. Programming environment

Programmers must consider several matters in HW tasks and SW tasks when developing programs on the target system. HW tasks have to have management modules which support communication and control HW task status. SW tasks have to use system calls to request RC-OS server. Programmer can use the APIs and HW task frame to write programs.

3.2.1. APIs

To send system calls to the RC-OS server, APIs are used in SW task. The APIs are provided as C functions. The APIs implemented for the prototype system is shown in Fig.7. A sample code of SW task which uses the APIs is shown in Fig.8. It shows that the SW task transfers data to HW task on FPGA resource.

3.2.2. HW task frame

HW task includes the management modules to use services of the RC-OS server. Connection control module and HW task control module are implemented as management modules. HW task frame is a template code described by HDL, which supports management modules and hardware interfaces. Programmer can describe HW tasks without knowledge of management modules by using HW task frame.

4. Evaluation

The prototype RC hardware for the evaluation of RC-OS server is shown in Fig.9. The FPGA resources are Xilinx Spartan3 sub-boards [5], which are connected to the host computer by ISA bus. PC is used as the host computer whose CPU is Pentium3 933MHz and the host OS is Linux (kernel 2.6). The evaluation items for RC-OS server in this paper are the following:

- Easiness of writing programs
- Overhead of RC-OS server's services

Evaluation methods and results of two items are described in 4.1and 4.2.

4.1. Easiness of writing programs

Easiness of writing program can be measured approximately by the number of program lines. The sample program showed in Fig.8 of section 3 is used for this evaluation as the program using RC-OS server APIs, and it is called **sample A**. For comparison with it, the equivalent program of **sample A** without using RC-OS server APIs is used, and it is called **sample B**.

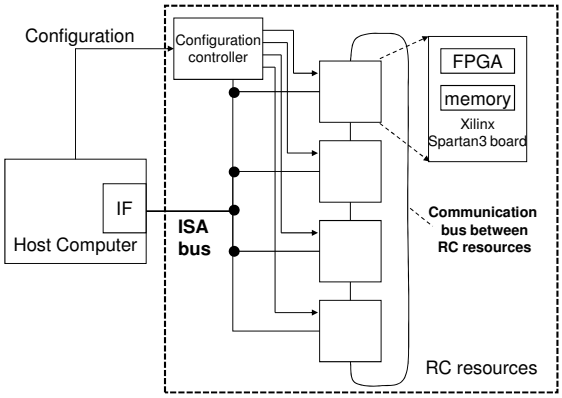

Figure 9. Prototype HW

Sample B must have functions to manage FPGA resources and run multitasking environment. Therefore, the number of program lines in **sample B** becomes larger than 120 lines. In contrast, **sample A** can be written in only 14 lines by using APIs. Programmer must treat various issues to describe **sample B** for the multitasking environment. For example, management of FPGA resources, communication protocol and synchronous processing between HW task and SW task, exclusive control, etc. It is necessary to know the detail of target system to write **sample B**. On the other hand, the knowledge for programming the **sample A** is only how to use APIs. Programmer can write the program easily without knowledge about the target system and consideration about multitasking environment. Therefore, we can say that the RC-OS server provides us more easier development environment.

4.2. Overhead of RC-OS server's services

The execution time of each API is shown in Table.1 which is measured to evaluate the performance of RC-OS server. The time of RC_create in Table.1 means the overhead of RC-OS server which does not include the configuration time, and the configuration time is shown in the parentheses. The time of RC_read and RC_write means the transfering time of 1KB data between SW task and HW task . According to Table.1, execution time of the RC_alloc and the RC_create is large. Especially execution time of the RC_create exceeds 6 seconds include configuration time. Whenever HW task is created, the configuration tool is started by RC-OS server at this moment. The most of overhead in RC_create is starting up and exiting time of the configuration tool. To resolve this matter, the configuration tool should be kept running. The configuration time depends on the FPGA devices in the target system. For these reasons, there is a chance to decrease the configuration time. The processing to select FPGA resources in RC_alloc consumes much time. Therefore, the algorithm to select FPGA resources should be improved.

The graph in Fig.10 shows the relation between the percentage of overhead and transfer data size. The horizontal axis of the graph shows the transfer data size. The vertical axis shows the percentage of overhead caused by RC-OS server in the transfer time. According to the graph in Fig.10

Table 1. Overhead of RC-OS server's servicies

API	Overhead(ms)
RC_alloc	101.1
RC_free	4.7
RC_create	1507.5(4864.9)
RC_write	9.5
RC_read	7.6

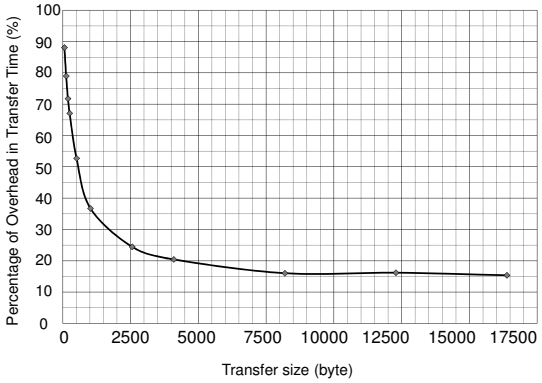

Figure 10. Ratio of overhead and transfer size

when the transfer size is small, the ratio of overhead in transfer time becomes large. However, when the transfer size is more than 8KB, the ratio of overhead is stable with less than 17%. To improve the performance of communication with RC-OS server, it is important to transfer enough large size data.

5. Conclusions

In this paper, we propose the RC-OS functions, and implement a prototype RC-OS server which has the RC-OS functions. The RC-OS server provides multitasking environment and easiness of writing programs on the target system which has multiple FPGA resources. As a future work, we plan to measure the performance of various applications using the RC-OS functions.

References

[1] Cray.Inc, "CRAY XD1 DATA SHEET," http://www.cray.com/global_pages/downloads/20050715CrayXD1JP-ver1.3.01web.pdf.

[2] Hidetoshi Sueyoshi, Hideharu Amano, "RECONFIGURABLE SYSTEM," Ohmsha, August, 2005.

[3] Starbridge Inc., "Hypercomputing HC-62," http://www.starbridgesystems.com/hypercomputing/HC-62.

[4] SRC Computers, "SRC-7 Overview," http://www.srccomp.com/products/src7.asp.

[5] Xilinx, "Spartan-3 Startar Kit," http://www.xilinx.com/products/devkits/HW-SPAR3-SK-UNI-G.htm