MODUL FOR RUN-TIME MONITORING IN PC HARDWARE BASED REAL-TIME SYSTEM

Bojan Jovanovic
Faculty of Electronic Engineering ,University of Nis, Serbia

Milun Jevtic
Faculty of Electronic Engineering ,University of Nis, Serbia

Abstract
This paper presents one way of implementation of hybrid on-line monitoring in real-time systems. Monitoring module is described in VHDL programming language and tested on Altera DE2 development board.

Keywords: on-line hybrid monitoring, real-time systems, VHDL, FPGA.

INTRODUCTION

For proper functionality of real-time systems (RTS) it is necessary not only to give the correct results on the outputs, but to give it in exactly defined time interval. This is especially true for hard real-time systems (HRTS), because untimely execution of the tasks can lead to disaster. Tracking the course of events in RTS we can make conclusions about meeting the timing requirements. Therefore, can be said with good reason that on-line monitoring of processes and events in HRTS is of enormous importance because it provides its predictable behavior.

EXPOSITION

Monitoring system is the process or set of possible distributed processes whose function is dynamic acquisition, interpretation and partitipation in information concerning application, during the application execution [1], [2]. Therefore can be said that monitoring systems improve vitality , security, fault tolerance and adaptability of RTS. Since the testing of the timing requirements of RTS directly depends on process, tasks and events monitoring, it is necessary to say a few words about testing strategies. There are three basic strategies for testing and correct functioning validation for RTS.

Sequential environment shown in Figure 1 is least complex for implementation. Testing scenario is created in off-line mode and test excitation is generated before execution of the test procedure. Data are stored and reproduced during the RTS operation. During the test, system response (the result of the test) is stored in real time and later analyzed in off-line mode to make a conclusion about functioning of RTS. The disadvantage of this approach is inability of the dynamic changes in the test scenario as well as inability of tracking the course of the test events. Also, the success or the failure of the test is known only after the analysis.

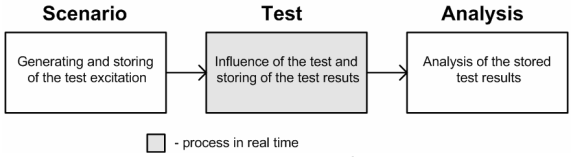

Fig. 1. Sequential organization of the testing process

More complicated testing systems generate data for test scenario in real-time, during the system operation (Figure 2). Only the test results are analyzed in off-line mode.

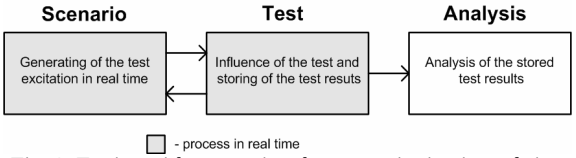

Fig. 2. Testing with generating the test excitation in real-time

Only configuration shown in Figure 3 provides complete testing in real-time and therefore on-line RTS testing.

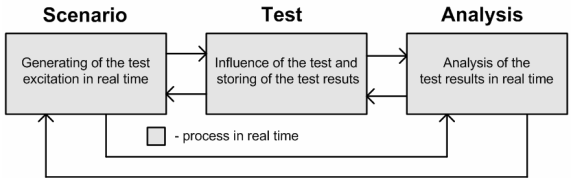

Fig. 3. Testing scenario when all processes run in real-time

RTS process monitoring

Monitoring system is obtrusive if it requires the use of application resources (CPU time, I/O devices, communication channels etc). Monitoring systems are mainly obtrusive in some level. Completely unobtrusive monitoring system use specialized hardware designed for monitoring. Ideal monitoring system which is completely transparent to the target system is very difficult to achieve in practice.

There are three basic aproaches in implementation of system monitoring:

- *software,*
- *hardware and*
- *hybrid approach.*

Software implementation of RTS monitoring is flexible, but largely obtrusive and therefore significantly disturb RTS timing characteristics.
Hardware based approach in implementation of RTS monitoring is unobtrusive in some level but requires specialized hardware. Whereas the target system must support the possibility of its installation, the use of this approach is inflexible and clumsy. It should be planned during the design of the target system.
Hybrid monitoring enables both, unobtrusive nature of the hardware approach and the flexibility of a software approach. That's why the hybrid monitoring system is a trade-off between pure hardware and pure software monitoring.

Realisation of the RTS monitoring

Posing the demand that on-line monitoring do not require significant CPU time and clumsy additional specialized hardware, this paper presents one way of implementation of hybrid on-line RTS monitoring. It is intended for RTS based on an industrial PC and Linux operating system which is widely accepted and available open source system in RTS.
Implemented system monitors up to 32 processes i.e. tasks and events that execute in parallel. The number of monitored processes is relatively small, but it should be said that HRTS in industrial applications do not have a lot of processes. Increasing the additional hardware , the number of monitored processes can be easily expanded.

Implemented system is based on additional hardware module with 32 programmable timer-counters and interrupt logic [3], [4]. Unlike [3] and [4], system monitors more processes that can be executed simultanously, and has some additional features. In addition, simple software primitives for on-line monitoring implementation are realized. They can be activated from desired place in application program code. For monitoring of the processes and tasks in RTS without modification of application program code, simple modification of the operating system task sheduler and dispatcher is predicted. Modification ensures that sheduler or dispatcher, with every change of the process/task status, activates appropriate software primitive for controlling timer and checking the time constraints. Block diagram of the monitoring system is shown in Figure 4.

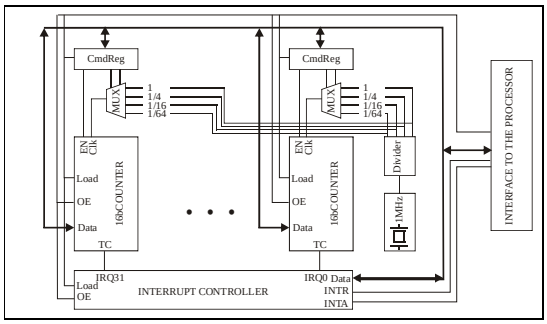

Fig. 4. *Schematic description of an 32-channel Online Monitor.*

Timers-counters are used as devices for defining the moments of events' time occurence as well as watchdog i.e. monitoring timers for checking the correct timing execution of the tasks. Command register, CmdReg selects the time quantum of 1, 4, 16 or 64 μs which gives the maximum time for task execution of 65.5, 262, 1048 and 4194 ms. Command register, through EN input enables or disables counter operation. In the monitoring module, there is also the decoder which, depending on its control inputs, passes the write enable, load and output enable signals to the desired command register and counter. Writing to command register and counter is done using the Data bus. Counter counts backward, from the specified set value to zero. If task is not complete within the given time interval, counter's TC output sets to logic one. Using interrupt controller, interrupt request for occured error processing is sent to the microprocessor.
Software primitives control the module which has the following functions:

- Setting the operating mode of timer-counters,
- Setting the time constraints,
- Enabling timer-counters,
- Disabling timer-counters,
- Reading value from timer-counters,
- Timer-counter interrupt processing,
- Comparison of the timer-counter value with the timing constraints.

During the system verification phase monitoring system provides informations about system timing characteristics and creates the log file. During the system operation it should detect deviations from predicted timing behavior of the system which is the consequence of a failure in RTS. Thereby, monitoring system has two operating modes. First mode refers to the system analysis. It performs with the purpose to measure the time of execution of every task. Obtained informations can be used for the future control of the RTS.
In the second mode monitoring system has the function of the built-in selftesting based on the watchdog function. It checks upper and lower time limit at the tasks and periodic quasi-periodic events level. Each task activation initiates the procedure of starting timer-counter. Monitoring timer-counter sets to the previously defined maximum task execution time and starts its countdown. If excess of the time interval happens,

monitoring module sets interrupt request. If the task is complete before time excess, timer-counter stops its countdown with the end of task execution. Monitoring module reads its state and checks whether the task is executed before the time (incorrectly performed). If the task is executed in regular time intervals, RTS continues to work, otherwise provided procedure for system recovery from detected error starts. In this way, predicted behavior of HRTS is ensured.

Checking the timing parameters of RT tasks on-line monitoring checks correctness of their execution. Real-time task τ_i can be characterized with the following timing parameters (Figure 5): r – moment of occurence of the request for task execution; B – maximum delay to the start of task execution; C – task execution time (needed CPU time); D – time limit for task execution; T – period of occurence of periodic tasks.

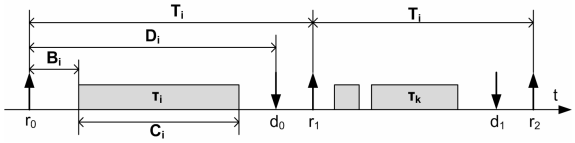

Fig. 5. *Timing parametes of RT task*

Possible course of non pre-emptive task (τ_i) execution is shown on Figure 6.

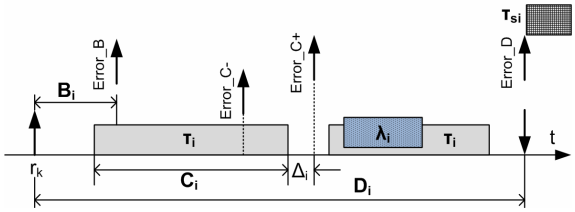

Fig. 6. *Monitoring scenario of non pre-emptive task execution*

From the moment – event r_k when request for task τ_i execution occured, allowed delay to starting the task execution can be checked at first.

This is important for the tasks that do not initiate with external interrupt event, but for the tasks which are „set“ in the queue for execution by some internal event. In the case of exceeding the interval B_i monitoring timer-counter generates a hardware interrupt request, and error Error_B is detected. Another monitored time interval is task execution time (CPU time). For task execution time which is shorter than C_i (minimal required time for correct task execution – detected when timer-counter is in zero state), marker Error_C- is set. In case of exceeding the task execution time $C_i + \Delta_i$ (maximum time for correct task execution) monitoring timer-counter generates interrupt request to detect error Error_C+. Such monitoring performs over each RT task. Upon detection of any of these errors, it is the policy of the planner and available time what will be taken. Restarting of the same task or starting alternative task λ_i execution which will overcome given situation can be done. For each task, deadline D_i for his execution should also be monitored. Special counter –timer is most suitable for this purpose. In the case of his exceeding interrupt request is generated and hardware-software security task τ_{si} is started. This security task should recover RTS or place it in a safe contidion.

Monitoring of pre-emptive tasks τ_i (Figure 7) differs from the previous monitoring scenario. While his execution is stopped because of higher priority task τ_j, its monitoring timer-counter should be stopped (during C_i).

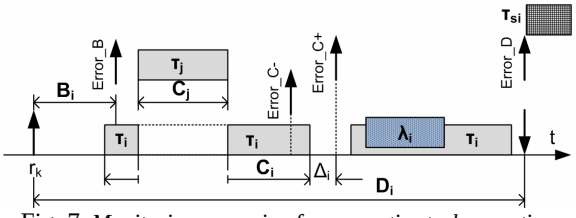

Fig. 7. *Monitoring scenario of pre-emptive task execution*

Monitoring module is described in VHDL programming language and implemented on FGPA programmable device of Altera DE2 development board [5], [6].

CONCLUSION

Implementation of hybrid on-line monitoring module was successfully achieved. Testing proved the correct functioning of monitoring module. The next task that imposes is the interface type to the microprocessor (USB, PCI or some other). Interface type is very important from the point of obtrusion to RTS. Interface type which will use application resources as less as possible should be chosen.

REFERENCE

[1] Jane W. S. Liu, *Real-Time systems*, Prentice Hall, 2000.

[2] N. Nissanke, *Realtime Systems*, Prentice Hall, 1997.

[3] M. Jevtic, V. Zerbe, S. Brankov, *Multilevel validation of on-line monitor for hard real time systems*, Proc. 24th International Conference on Microelectronics – MIEL 2004, Vol. 2, pp. 755-758, Nis, Serbia, 16-19. may, 2004.

[4] S. Brankov, M. Jevtic, “Module for Run-Time Monitoring of Real Time Processes Realized in VHDL”, *Proceedings of the XLVII ETRAN Conference*, Vol.1, pp. 80-83, Herceg Novi (Serbia and Montenegro), June 2003.

[5] Clive Maxfield,*The Design Warrior’s Guide to FPGA*, Elsevier, New York, 2004.

[6] Altera DE2 user manual: ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf