functions, as a result, the data gathered from different RTOS components cannot be compared.

Efforts have been made to measure the performance of RTOS as shown in Anh and Tan (2009), Martinez et. al (1996), Kar and Porter (1989) and Sacha (1995). These RTOS performance analyzers are based on benchmarking method of frequently used system services. The performance of an RTOS is measured verifying if the task complies with its deadline. As there are various types of applications with each having very different requirements, benchmarking against any generic applications will not be reflective of the RTOS strengths and weaknesses.

Another approach is presented by Lamie and Carbone (2007), where they provide some APIs which can be used in different RTOS to verify the performance. But in this proposal, there is software interference on the performance measurements. According to the target, some changes also may be required. The JEWEL tool proposed by Lange et. al (1992) was developed to analyze the performance of distributed systems. Also Wybranietz and Haban (1988) proposed integrated tool for monitoring distributed systems continuously during operation.

The main objective of the test environment proposed in this paper is to measure absolute time characteristics of a RTOS, that influences performance, for embedded applications, without the insertion of overhead, neither including changes on the embedded software. This approach allows time comparisons that may be helpful to analyze different RTOS performances in different processors. To show the proposed real-time system based on FPGA to measure the transition time between tasks in a RTOS, this work contains 5 sections. In Section 2 an overview of the proposed solution is presented, introducing the elements used on the implementation. Section 3 presents in details the system implementation. In Section 4, experimental results are described. The paper ends with a conclusion, acknowledgment, and references.

2. Proposed Solution

This paper proposes a real-time system based on FPGA to measure the transition time between tasks in a RTOS. The use of FPGAs allows many processes to start at the same time [D'Amore, 2007] which creates an efficient tool to analyze the performance of a device running RTOS, and there is no overhead due to software initialization or software modifications that usually occur on the debuggers.

In order to determine the transition time between the tasks in a RTOS, the first step was to decide which device should be chosen as the target. In this case, the Device Under Test (DUT) is an ERC32 development kit [ATMEL, August 2003]. The ERC32 is a radiation-tolerant 32-bit RISC processor developed for space applications, [ATMEL, August 2004] and [ATMEL, March 2005].

The second step was to choose a RTOS. The chosen DUT accepts the real time operating system RTEMS (Real-Time Executive for Multiprocessor Systems) [OAR, 2006].

The third step regards to the choice of the task model to be implemented on the DUT. The task should produce stimulus for the data acquisition module, where these stimulus could be read and stored in a temporary memory. Once the data acquisition

The running software was developed in C language. The software was build using RTEMS development tools. The main characteristics of the implemented tasks and the RTEMS configuration are:

- Four equivalent tasks with the same priority level were implemented using RTEMS.
- Each task was implemented as an infinite loop that produces a periodic square wave in the GPI.
- The RTEMS was configured to use round-robin scheduling with a fixed time slice.

Figure 2 shows the general idea of the task model used in the software architecture. The cooperative context switching task model proposed by Lamie and Carbone (2007) was chosen.

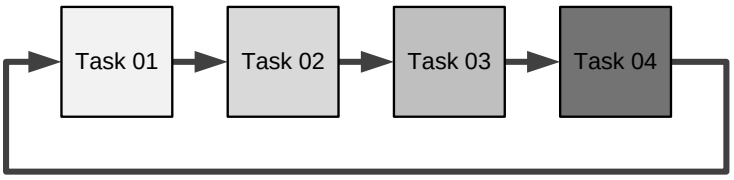


Figure 2: Task model

Figure 3 shows an overview of the system under test and the output expected on the GPI.

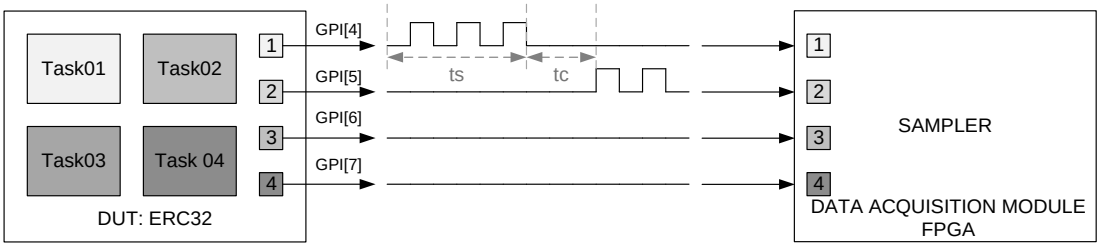


Figure 3: General overview of the device under test and the output on GPI[4~7]

In Figure 3, on the left, each task produces a square wave in one GPI output. On the right, the output expected on the GPI, when the scheduler achieves the time set on the time slice, represented by (ts). There is transition time, represented by (tc), between the Task01 and Task02. Along the time, after each time slice, there is a transition time (tc) between the tasks. It is important to emphasize that each task produces a square wave on a specific GPI output and keeps all the other GPI interfaces in zero, e.g, task01 produces a square wave on GPI[4] output, task02 produces a square wave on GPI[5], task03 outputs to GPI[6] and task04 outputs to the GPI[7]. The period of the square wave can be changed through an internal counter on the task routine. The RTEMS configuration parameters were configured as follows.

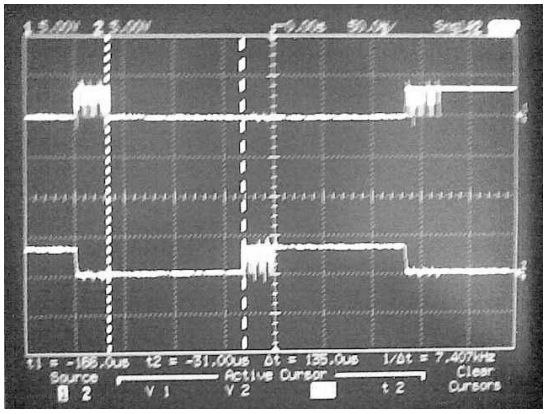


Figure 4: Output on GPI measured on the scope

3.2 Data Acquisition Module

The Data Acquisition Module makes use of an ALTERA FPGA. The development environment for the FPGA is composed by the Quartus® II software [ALTERA, October 2007] and a Stratix II development kit [ALTERA, May 2007] and [ALTERA, January 2007].

The FPGA was programmed with VHDL and ALTERA Mega-Function Blocks. The Data Acquisition Module can be divided in two functional blocks: the Sampler and the Communication block.

Figure 5 shows the block diagram of the Data Acquisition Module implemented in FPGA. The communication controller transfers all the data storage to the PC.

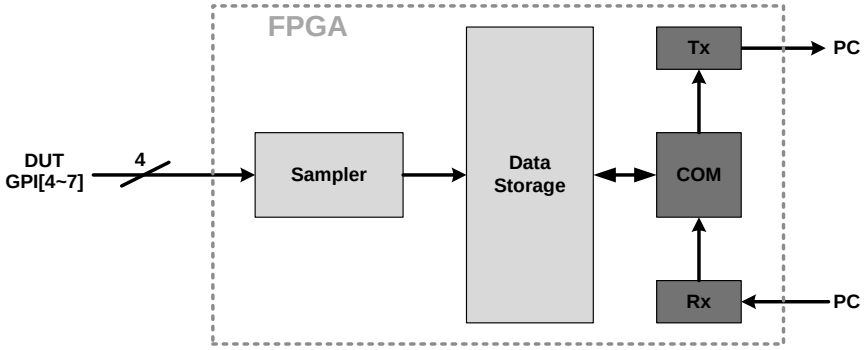


Figure 5: Overview of FPGA implementation

The Sampler Block performs the monitoring and analyzing function. This block has a free-running counter (ticks counter). This free-counter tick value is stored when the hardware detects a task transition on the GPI. At this moment, the minimum historical data are stored, to permit infer the transition time between the tasks. To monitor the GPI, the data acquisition module reads the GPI as an 8 bits register. The four low significant bits (GPI[0~3]) are always read as zero.

Figure 6 represents the moment of a transition time. The samples occur on the rising edge of the data acquisition module clock. A First In First Out (FIFO) register stores the three transition in the GPI that represents the transition time, ticks t_5, t_7 and t_{14}. The data are stored in the FPGA memory and when the FPGA receives a command from PC, the FPGA dumps the memory to PC.

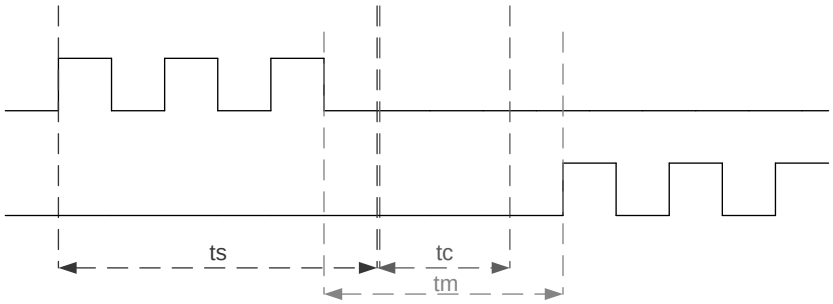


Figure 8. Maximum non-desirable contribution on the transition time measured by the data acquisition module

The non-desirable contribution can be reduced increasing the frequency of the square wave for each task, reducing consequently the period, as shown in Figure 9. Once the data acquisition module works with a clock more than 1000 times faster than the frequency of the square waves on GPI, it's possible to increase the square wave frequency without loss of accuracy of the measurements.

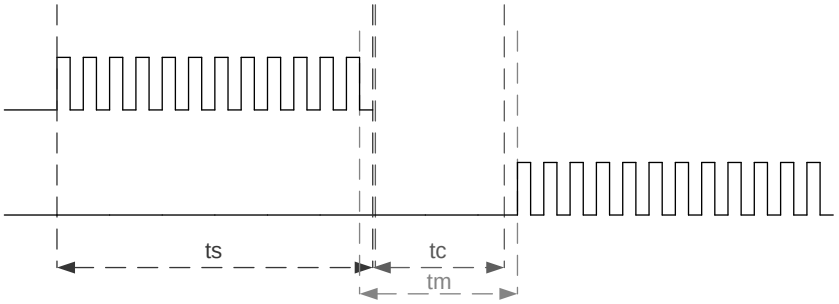
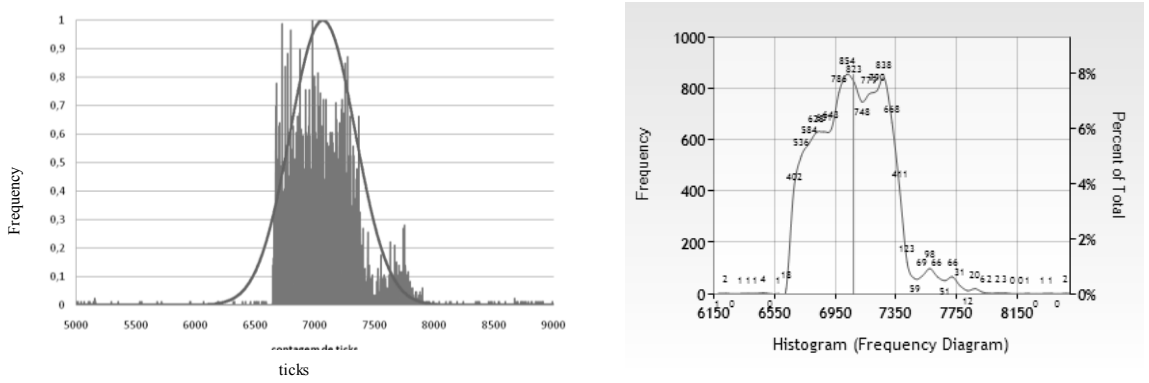


Figure 9. Square wave with higher frequency reduces the worst condition for the data acquisition module.

3.3 Analysis Module

The main function of the Analysis software is to receive, to verify and to generate the graphs. The development tools used for analysis module are composed by the Visual Studio .NET Framework to acquire the data and the Dundas software to the graphical interface [Dundas, 2009].

The PC host, with the developed analyzer software uses the serial interface. The transfer rate, frame format (start bit, parity and stop bit) must be configured, using also the developed software. The Figure 10 shows the developed software window.



(a) Ticks (b) Histrogram

Figure 11. Results one acquisition: (a) ticks and (b) Histogram

Figure 11a, on the right, depicts the results of one acquisition from the PC host software and the Figure 11b, on the left, depicts the results of the same acquisition, but analyzed manually, using the Excel. As one can see, both pictures have the same shape. The attenuation on the small picture can be explained once the developed software for the PC host considers a range of values (ticks) to plot the graph and the manual analysis is plotted using the ticks (x axis) incremented one by one. These transition time values were compared with the expected value of 135us, observed on the oscilloscope in the lab.

As it was explained on Section 3.2, the results presented in both graphs in Figure 11 show the measurements considering all the possibilities of additional contributions due to the task implementation. The difference between the maximum and minimum value taking into account the average value is about ±395 ticks, which represents a range of ±7,9us.

5. Conclusion

The results of this implementation show that it is possible, using the proposed real-time system based on FPGA, to measure the transition time between tasks in a running RTOS. These measurements can be helpful to model more precisely the tasks that one RTOS shall perform and to estimate with a better level of accuracy the real time performance accuracy of the software, particularly, contributing to the study of safe critical applications.

6. Acknowledgements

We want to thanks AEB and UNIESPAÇO program for the financial support of the project.

7. References

ALTERA (January, 2008) “Stratix II Device Handbook, Volume 2 – SII5V2-4.4”,January.

ALTERA (May, 2007) “Stratix II Device Handbook, Volume 1 – SII5V1-4.3”, May.

ALTERA (October, 2007) “Introduction to the Quartus® II Software – Version 7.2”, October.