Design of a Test Environment for Real-Time Operating Systems in Satellite On Board Computer using FPGA

Guilherme B. Resende[1], Osamu Saotome[2], Paloma M. S. Rocha Rizol[1],
Lidia H. S. Sato[2], Fernando G. Nicodemos[2]
1 Univ Estadual Paulista - UNESP – Dep. de Engenharia Elétrica - Av. Dr. Ariberto Cunha, 333 - Pedregulho– Guaratinguetá – SP
2 Instituto Tecnológico de Aeronáutica - Praça Eduardo Gomes, 50 – Vila das Acácias –São José dos Campos – SP

***Abstract* — Nowadays, the availability of microcontrollers from 8 to 16 bits and 32-bit microprocessors capable of executing 100 million instructions per second, allows the existence of applications that require the use of Real-Time Operating Systems (RTOS) in their embedded systems. This paper presents ameasurement environment to test an RTOS, in order to validate its use in special applications, such as satellite onboard software's, once it must respondto critical constraints, like timing. In this case, the objective is to measure the interrupt latency time, through a high-performance FPGA,of an embedded system running the RTOS Real-Time Executive for Multiprocessor System (RTEMS), widely used by the international spatial community, for on board computer systems.**

***Keywords* — RTOS , RTEMS , Embedded Systems, On Board Computer**

I.INTRODUCTION

The continuous advances in the development of semiconductors, microcontrollers and microprocessors have become more powerful each day, thus, being able to execute millions of instructions per second. Along with this evolution, applications that were not possible to imagine years ago, today, can be easily performed due to the existence of these devices. Certain embedded applications have reached such a level of complexity, that they shouldbe managed by an operating system. According to [1], an embedded system is any system or computing device that performs adedicated function or, it is designed to be used with a specific embedded software application. And according to [2], real-time operating systems are operating systems that have the ability to respond to an event in a limited and deterministic time, this is one of the key factors for performance of a RTOS.Systems that operate under a RTOS are classified as a hard real-time system or a soft real-time system. The soft real-time is a system that tolerates, but do not want, a bit more than the deadline of the task in a operating system, like, for example, in video streaming. However, in hard real-time systems, absolute deadlines must be reached in any ways and any failure is not acceptable. Some examples of systems like these are nuclear reactor systems, ABS brake systems, Air-Traffic Systems, and satellite onboard software's.

The main goal of this environment, is the measurement of performance parameters of a RTOS, because such parameters are essential for deep tests and the decision on whether using or not a RTOS in an on-board computer of a satellite. For instance, RTOS are known and built to fulfill tasks in a well-known amount of time (deadline), and not being able to do so, raises the issue on its reliability concerning its choice.

This work presents the measurement results of the interrupt latency time. The interrupt latency time must be as small as possible, once the RTOS stops executing a certain task, which has a deadline to meet, to serve an Interrupt Service Routine (ISR) and then, switches back to the task the RTOS had been executing before the interrupt request happened. Basically, an interrupt request occurs as the following: an asynchronous pulse, which is, that can happen at any time, arrives at the processor. Meanwhile, the processor is doing its work normally incrementing the Program Counter Register. When the interrupt is acknowledged, the processor stores the address of the Program Counter in the stack (push operation), and sets the Program Counter to the address that the Interrupt Service Routine is located. When the processor begins the ISR, in this work, it masks all the other interrupts to avoid interrupt nesting, executes the ISR and then sets back the Program Counter to the address it had before the ISR arrived (pop operation). For this paper, there has been created an environment that consists of a software for analysis and data storage for PC, a hardware unit for stimuli generation and measurement with the use of programmable logic technology, in this case, a Field Programmable Gate Array (FPGA) of high performance and a Test Device (TD) using the processor ERC32 running the Real-Time Operating System (RTOS) Real-Time Executive for Multiprocessor Systems (RTEMS). Both ERC32 and RTOS RTEMS were chosen because they are aligned with the guidelines for the development of new satellites at the National Institute of Space Research (INPE) and with the development line of the European Space Agency (ESA). The high performance FPGA used in this work is Altera's Stratix II FPGA Family. Basically, this environment can be understood as shown in Figure 1.

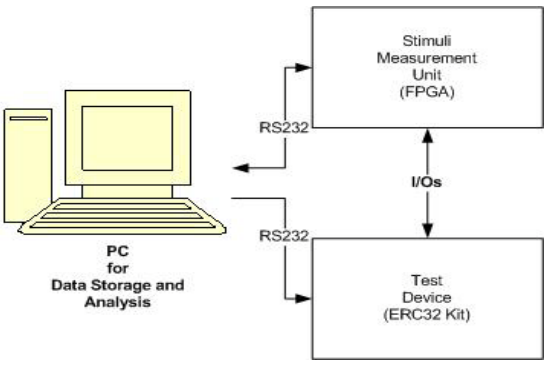

Fig.1 - Test Environment for Real-Time Operating Systems

Each block of the Figure 1ispresented in the following section. The Stimuli Measurement Unit, however, is detailed in section 3, once it is the focus of this paper. The present article is organized in 5 sections. Section 2 presents the description of the components of the test environment. Section 3, describes the Stimuli Measurement Unit. Results are shown in Section 4 and conclusions in Section 5.

II.THE ENVIRONMENT

A more detailed view of the system can be seen in Figure 2.

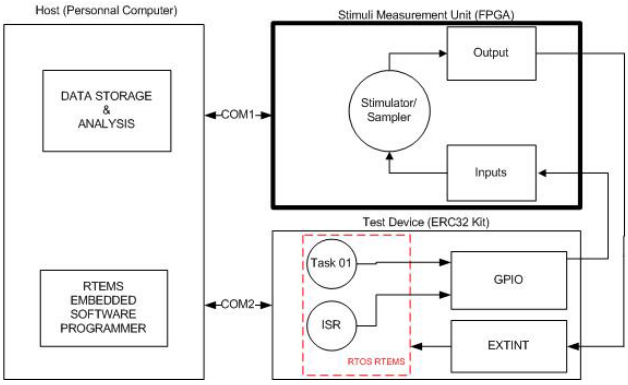

Fig.2 - Block Diagram of the Environment

Three steps have been defined in order to develop the system:

1. Development of a hardware for stimuli measurement from a Test Device (TD);
2. Development of an embedded software for RTEMS to simulate a task being executed and the interrupt handling;
3. Development of software for PC that is a data base for analysis of the results from the Test Device.

And also, have defined how the Test Device (TD) is going to be connected to the Stimuli Measurement Unit. Two ports of the General Purpose Input/Output (GPIO) from the Test Device were assigned to be the outputs for reading the behavior of the embedded processor. The Test Device has also a pin for receiving external interrupts, and the pin chosen is called EXTINT4, the interrupt requests are detailed in section3.2.1. The connections between the Test Device and the Stimuli Measurement Unit are shown in Figure 3. The pin assignments can be found in [4] and [5].

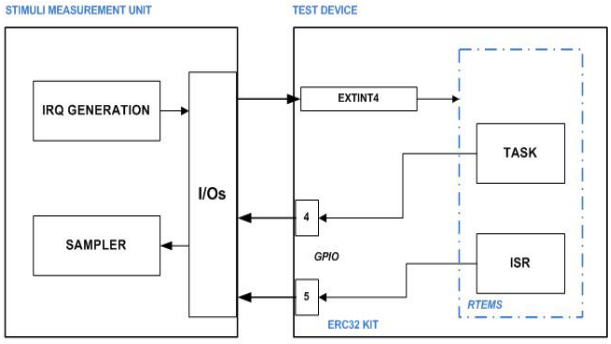

Fig.3 - Connection between the Test Device and the Stimuli Measurement Unit

A brief explanation of the blocks found in Figure 2 is given below.

2.1 Stimuli Measurement Unit (FPGA)

The stimuli measurement unit is the interface between the PC and the Test Device. This block generates pulses simulating Interruption Requests (IRQs), and storing the changes in logic levels of the GPIO pins (Figure 3), these changes are captured by the sampler and later are sent to the PC through a serial interface for storage and analysis. The Stimuli Measurement Unit block is highlighted in Figure 2 since it is the scope of this paper, this block will be further detailed in section 3.1.

2.2 RTEMS Embedded Software

The next block is the development ofan embedded software for the SPARC ERC-32 embedded processor, running the RTOS Real-Time Executive for Multiprocessors System (RTEMS).

The software consists of a task running indefinitely generating pulses (square waveform) in one port of the board's General Purpose Input/Output (GPIO) while an Interrupt Request does not require the use of the processor and does not preempt the current task.

When an IRQ arrives, the Interrupt Service Routine (ISR) generates pulses (also in a square waveform) in another pin of the GPIO ports available on the board for a certain amount of time, and then, it releases the processor for the task that had been running before the Interrupt Request arrived. Since there will be a change in logic levels in both pins, the Sampler (Figure 2) in the FPGA records this change and stores this data in a memory that have this data sent when it is completely filled (32768 samples), through the software for PC (section 2.3), is possible to verify the interrupt latency time t , where t is shown in the Figure 4.

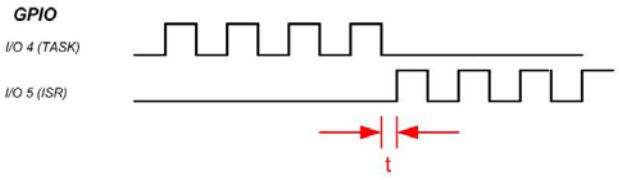

Fig.4 - Interrupt Latency Time

The flowchart of the embedded software is shown in Figure5.

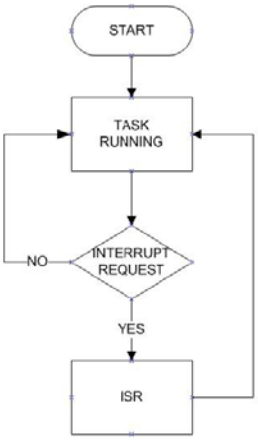

Fig.5 - Software Flowchart

2.3 Software for PC for Data Storage & Analysis

This software has the function to receive all the data that comes from the Stimuli Measurement Unit, process these data and generate graphics for statistical analysis. The Eclipse IDE [6], has been used for the development of this software, and it was programmed in JAVA language. The database was built through a Java Database called HyperSQL [7], and the tool for report generation is the JFreeChart [8]. The software communicates with the Stimuli Measurement Unit through a serial interface and must be configured to the same configuration of the units for receiving and transmitting data from and to the FPGA (parameters like baudrate, parity, stop-bit) . The main screen is shown in Figure 6.

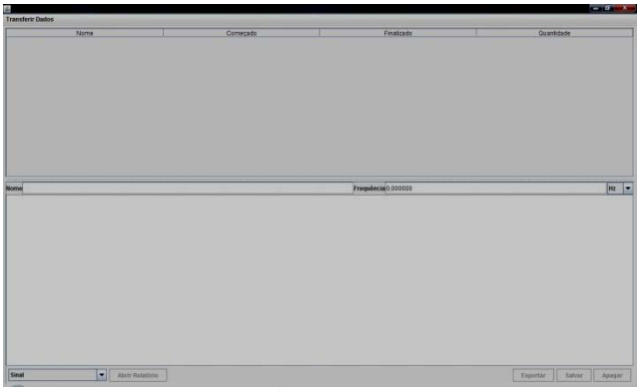

Fig.6 - Software Main Screen

The top window presents a table that shows all the saved data, showing an identifier, the date of start and end and the amount of data in each capture. Before generating the reports, the desired frequency must be entered in the software for a correct design of the reports, in this case, it is 50 MHz. The command window where configuration parameters are set and sampling begins is shown in Figure 7.

Fig.7 - Command Window

III.HARDWARE FOR STIMULI MEASUREMENT FROM THE TEST DEVICE

3.1Description of the Hardware for Stimuli Measurement from a Test Device (TD)

The focus of this hardware is to perform the measurement of important parameters of the RTOS's. Parameters like time between context switches, interrupt latency, message passaging, among others, are examples of these important parameters.

The hardware was named Stimuli Measurement Unit and it consists of a high-performance FPGA programmed in VHDL (VHSIC Hardware Description Language). The benefits of using a FPGA, is thatthe delay times using this technology, are smaller than the delay times that appear when using a software for measuring these same parameters.

In its first version [3], the hardware unit for measurement consisted on context switch measurements, called cooperative context switching. In this second version, the aim is to measure interrupt latency time. The following tools have been used for the development of this hardware unit:

-Altera's development software Quartus II

-Altera's development kit FPGA Nios II Development Board

The functional diagram for the Stimuli Measurement Unit is shown in Figure 8.

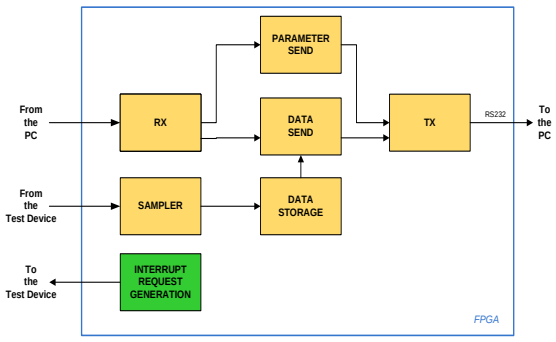

Fig.8 - Functional Diagram for the Stimuli Measurement Unit

Each of these blocks will be described in the following sections.

3.2 Description of the blocks of the Stimuli Measurement Unit

3.2.1 Interrupt Request Generation

The interrupt requests were chosen to be automatically generated by the FPGA, since the memory used in the Data Storage block is capable of storing 32768 samples, it would be unpractical and not precise, to manually generate these requests using a push-button, for instance. There would be some problems like noise in the push-button (bouncing), and it would be a tiring task for an operator to press the switch 32768 times. It was chosen that the FPGA will generate an Interrupt Request signal with a period of 2 seconds and t_{on} of 100 milliseconds. The signal waveform is captured in an oscilloscope and it is shown along with its period and its t_{on} in section 4. It has been used the FPGA clock to be referred as time base for the pulse generation. When the programmer tool (from Quartus II) finishes programming the hardware, the interrupt request signal does not exist, the FPGA sends logic level 0 to the Test Device and the embedded processor does not recognize an IRQ. A push-button was assigned in the FPGA board to indicate the hardware to start generating interrupt pulses, another push-button was assigned to stop this generation at any time, when the memory is totally filled and there is no more sampling, for instance. Two leds have been used to indicate the start of interrupt generation and the stop of such.

3.2.2. Sampler

The Sampler Block has the function to monitor the GPIO ports that are connected between the FPGA and the Test Device (Figure 3). The FPGA clock (50 MHz) works as a the time basis. Every high-edge of the clock, the sampler will read the logic levels at GPIO 4 and 5 and registers the state of these pins. Figure 9 explains the basic function of the sampler.

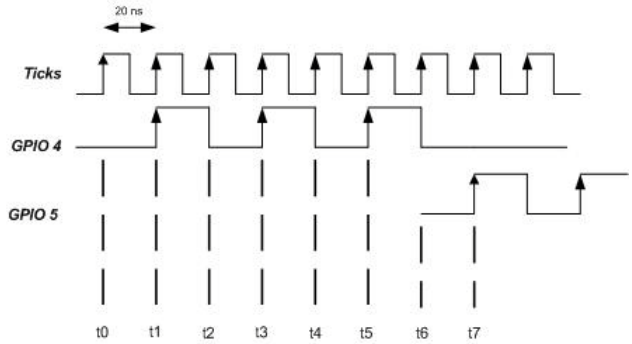

Fig.9 – ISR latency time capture

As mentioned before, at each high-edge of the board's clock, the sampler will read the logic state of GPIO 4 and 5. The task that will be running indefinitely (Figure 5) , generates a square waveform at GPIO4, so for this pin, the sampler will read either 00h or 01h. According to Figure 5, when a IRQ is acknowledged by the processor, the ISR generates a square waveform at GPIO5, having the sampler to read either 00h or 02h (there are 8 GPIO ports available, but the most significant are mapped in this project). At each instant of time, the sampler registers what is happening at the GPIO ports and stores this sample in the FPGA memory as shown in Table I.

TABLE I–VALUES STORED IN THE MEMORY BY THE SAMPLER

I/O 4-5	t	Ticks	Value stored in the memory
01h	t_5	00000005h	0100000005h
00h	t_6	00000006h	0000000006h
02h	t_7	00000007h	0200000007h

The Table I show the values stored in the FPGA's memory for the example given in Figure 9. The interrupt latency time, which is the goal of this project, can be calculated by doing the difference between t_7 and t_6 , since the clock period is 20 ns, for this example, there would exist an interrupt latency time of (7*20) – (6*20) = 20 ns. All of these data (values stored in the memory) are sent to the software for PC when the memory is totally filled (32768 samples of 40 bits each) . There is a counter inside the sampler to detect if all the samples have been reached.

3.2.3 Reception Block (RX)

The reception block works with serial port RS232 with a baudrate of 115200 bps, even parity and one stop-bit. This block works with data of 8 bits (1 byte). Inside the Reception Block, there is a unit called "uart_rx" that was built in VHDL, responsible for receiving a bit from the serial port. Each word received by "uart_rx" is stored in registers to form a Frame of Command. After the storage of these words, it is done a calculus of *checksum* to verify and validate the data stored in the set of registers, a Parameter Frame or Data Frame will be generated by the Parameter Send Block or Data Send Block, according to the Frame of Command stored in the registers. An "enable transmission signal" is also created to send for transmission either parameter or data. The Parameter Frame is detailed in Section 3.2.4 and the Data Parameter in Section 3.2.5. The only component of the Reception Block that was built in VHDL was "uart_tx", others were made by components available in Altera's Quartus II library. The Frame of Command is shown in Figure 10.

STX	Command ID	ETX	CKS
8 Bits	8 bits	8 bits	8 bits

Fig.10 - Frame of Command

The frame Command consists of a 4-byte frame that is sent from the Software for PC to the Stimuli Measurement Unit. The description of each byte of the frame is given on Table II.

TABLE II - FRAME OF COMMAND

Field	Code	Description
STX	02h	Start of Transmission
ID	11h	Request Data Send
	12h	Request End of Data Send
	13h	Request Parameter Send
ETX	03h	End of Transmission
CKS		Least significant byte from the sum of all fields

A block diagram of RX is shown inFigure 10.

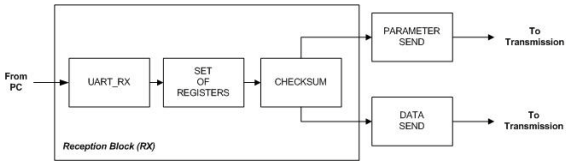

Fig. 11 - Reception Block (RX) Diagram

3.2.4 Parameter Send

The Parameter Frame is a 4-byte frame that is sent from the Stimuli Measurement Unit to the Software for PC, and it contains the FPGA's operating frequency. The frame

and its description are given in Figure 12, and its content as described in Table III.

STX	PAR	ETX	CKS
8 Bits	8 bits	8 bits	8 bits

Fig.12 - Parameter Frame

TABLEIII- PARAMETER FRAME

Field	Code	Description
STX	02h	Start of Transmission
PAR	32h	Parameter:Clock frequency of 50 MHz
ETX	03h	End of Transmission
CKS		Least significant byte from the sum of all fields

This block is responsible for creating this frame. The circuit for Parameter Send can be understood by the block diagramshown in Figure 13.

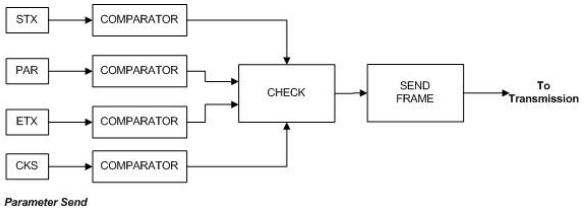

Fig.13 - Block Diagram for Parameter Send

Each field received by the circuit, is compared with the values given above (in Table 2), if all of them are correct, than the frame is sent to transmission.

3.2.5 Data Send

The Data Frame is also a 4-byte frame and it is sent from the Stimuli Measurement Unit to the Software for PC. It contains the data from the samples measured at the I/O pins of the Test Device. The frame and its description are givenin Figure 14,and its content as described in Table IV.

STX	DF	ETX	CKS
8 Bits	256 x 5 bytes	8 bits	8 bits

Fig.14 - Data Frame

TableIV - DATA FRAME

Field	Code	Description
STX	02h	Start Of Transmission
DF	256 data of 5 bytes	Data
ETX	03h	End Of Transmission
CKS		Least significant byte from the sum of all fields

This block is responsible for creating this frame and to control an address counter for sending the data that is stored in the FPGA's memory.

3.2.6 Data Storage

The Data Storage block uses the FPGA's internal memory, limiting the capacity of storage for 32768 words of 40 bits each. It is responsible for saving the data that comes from the Sampler and making them available for the Data Send block to be sent for the Software for PC for analysis.

3.2.7 Transmission Block (TX)

The Transmission Block has a component built in VHDL that has the same parameters for serial communication like the component "uart_rx" from the Reception Block (3.2.3). The component is called "uart_tx".

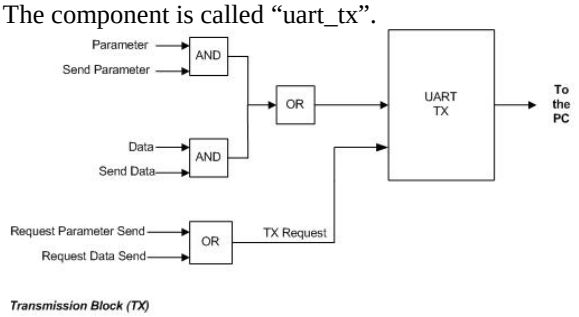

Fig.15 - Transmission Block (TX) Diagram

This block will either send parameter or data, as mentioned before, and "uart_tx" only makes the data available if a transmission request signal is present. A block diagram is shown in Figure 15 for better understanding.

IV. RESULTS

As mentioned in Section 3.2.1 , the waveforms for the Interrupt Generation are shown in Figures 16 and 17. The Figure 16 shows how the signal behaves through time and the time period for this signal, it possesses a high-logic level at each 2 seconds. And Figure 17 displays the amount of time that the IRQ signal is in its ON state. This signal is this one that generates a IRQ to the Test Device. The results for the other components of the Stimuli Measurement Unit can be found in [3].

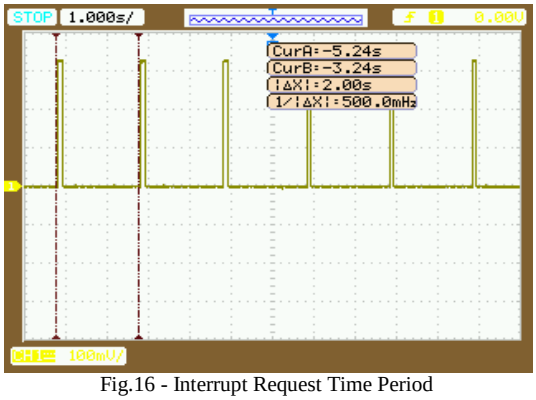

Fig.16 - Interrupt Request Time Period

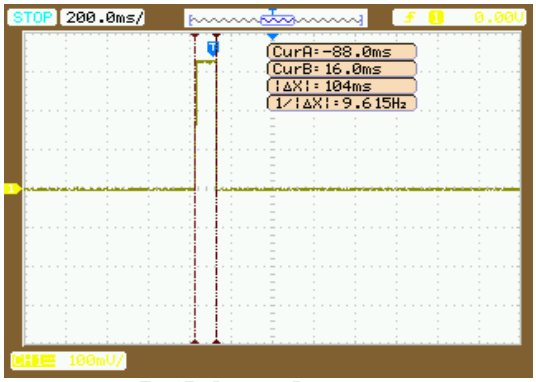

Fig.17 - Interrupt Request ton

V. CONCLUSIONS

This environment has been developed due to the need of testing critical spacial embedded systems. This paper focused on the development of a Stimuli Measurement Unit using a high-performance FPGA for measuring the interrupt latency time of anembedded processor for space application ERC32, running the RTOS RTEMS. The implementation of the embedded software will be done in future works.

REFERENCES

[1] A. Guerrouat, H. Richter, "A Formal Approach for Analysis and Testing of Reliable Embedded Systems". Electronic Notes in Theoretical Computer Science, v.141, n. 3, p.91-106, December 2005.

[2] D. Simon, An Embedded Software Primer. 1ed Pearson Educational, 1996.

[3] L. H. Shibuya, S. Sato, O. Saotome, F. Nicodemos. "A real-time system based on FPGA to measure the transition time between tasks in a RTOS." Publishedat WSE 2010, 2010.

[4] Altera's Nios Development Board Stratix II Edition – Reference Manual

[5] Atmel's TSC695F SPARC 32-bit Space Processor User Manual

[6] "http://www.eclipse.org/downloads", accessed in July 10th, 2011.

[7] "http://www.hsqldb.org" , accessed in July 10th, 2011.

[8] "http://www.jfree.org/jfreechart" , accessed in July 10th, 2011.