Monitoring of system memory usage embedded in FPGA

Víctor Asanza Armijos, Nathaly Sánchez Chan, Rommel Saquicela, Luis Macas Lopez
Facultad de Ingeniería en Electricidad y Computación, FIEC
Escuela Superior Politécnica del Litoral, ESPOL
Guayaquil, Ecuador
{vasanza, nssanche, rsaquice, luianmac}@espol.edu.ec

***Abstract* – At this moment in the field of FPGA, only RAM tests have been carried out to evaluate its performance but these works have not focused on tracking memory usage in real time, this paper proposes a design for monitoring the memory of an embedded system, in the logical part, making use of the communication between the FPGA and the HPS. In addition, the HPS has implemented a web service that allows to visualize a graph of the monitoring in real time. The proposed design can be an introduction to the development of applications that can be specifically monitored for a component of the embedded system in FPGA, because FPGA is currently being used for different purposes such as machine learning, real-time image processing, mining of Bitcoins, among others. These applications are quite robust, which implies a high demand for processing for the embedded system.**

Keywords- FPGA; HPS; Embedded System; RAM; DE10 Standard.

I. INTRODUCTION

All types of microprocessors use Random-Access Memory (RAM) as temporary storage for execution of processes and tasks, this is because it is much faster to access it. For monitoring and performance tests is necessary to verify its high availability and fault tolerance [1].

Current Printed Circuit Board (PCB), such as the DE10 Standard Development Kit designed by Terasic company, incorporate chips with portions of System on Chip (SoC) and Hard Processor System (HPS) so they can create and implement two embedded systems in real time, each can work independently or together with their own Central Processing Unit (CPU) [2]. One of the advantages of communication between these two systems is that not only they can share physical resources, they can also verify response times and application execution [3].

Currently, the use of Field Programmable Gate Array (FPGA) has increased quite high because it is simpler and cheaper than CPU / Graphics Processing Unit (GPU) implementations. It can be reprogrammed at any time to perform a different task than the one that was being executed initially [4]. In addition, this allows the same algorithm to be modified to make it much more robust and complex, in order to obtain better results [5].

As indicated above, there are already many designs implemented in both complex and simple FPGA. However, it has not been observed any project in which the communication between FPGA and HPS is verified, nor the monitoring of one with respect to the other. To make this communication possible, there are physical and logical bridges between the two parties, these bridges provide the possibility of working in many ways, among which can be master-slave or parallel and independent work, the latter is the one that was implemented in this design using shared memory that is accessed by Nios II from FPGA and Advanced RISC Machine (ARM) processor from HPS [6].

II. RELATED WORKS

There are several related work areas that talk about testing RAM modules in FPGA based on SRAM using a minimum number of test configurations but these works focus only on functional tests in the part of physics that makes up the RAM of the FPGA. This work is responsible for monitoring the RAM of the embedded system that is in the FPGA part. In addition, this monitoring of memory usage can be viewed from a webpage hosted on the hard processor of the DE10 standard card [3, 7].

To carry out the RAM memory monitoring of the embedded system from the hard processor the device tree was created, which allows to configure a kernel at runtime. To use device trees, it is needed a textual representation called Device Tree Source (DTS). The DTS is compiled into a binary representation called Device Tree Blob (DTB). The DTB is delivered to the kernel at boot time. Device Tree Compiler (DTC) is compiled as part of the Linux kernel compilation and is also available as part of the SoC FPGA Embedded Development Suite (SoC-EDS), this operation is detailed on [8].

One of the main objectives in real-time applications is to use memory in a optimized way during the execution of tasks based on SoC architecture. Wang et al. Demonstrated in their work that real-time Electrocardiogram (ECG) signal monitoring systems can be performed using an FPGA with two 8GB Dual Data Rate Synchronous Dynamic Random Access Memories (DDR3 SDRAM) , 72 Mb Static Random Access Memory (SRAM) and its system clock frequency is 156.25 MHz [9].

Supported by Escuela Superior Politécnica del Litoral (ESPOL) and NationalSecretariat of Higher Education, Science, Technology and Innovation of Ecuador (SENESCYT).

ISBN 978-80-261-0892-4, © University of West Bohemia, 2020

III. METHODOLOGY

For the monitoring of the RAM in the FPGA part of the DE10 Standard card, a 50MHz global clock was used. The size of the configured RAM is 65536 bytes, the core used is the Nios II/e version, that also uses a Very High Speed Integrated Circuit Hardware Description Language (VHSIC-HDL) block which does the work of a "for" or "while" cycle to access memory addresses in write or read mode, all this project was carried out on the DE10 Standard card which has a Cyclone V 5CSXFC6D6F31C6 chip [10].

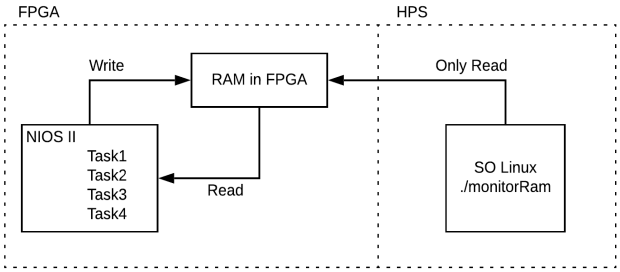

Fig. 1. Representation of communication between FPGA and HPS.

Algorithms were implemented in both the logic and the HPS portion. In the latter, a web server was also installed which allows visualizing by means of a graph the memory usage.

A. FPGA implementation

Using the Eclipse Kepler Software, the algorithm was implemented which will execute Nios II to access the input and output peripherals. In the same way, the use of the RAM when writing in it. Additional software allows the creation of a .hex file that is recorded in memory to run the algorithm once it starts [9].

The Nios II microprocessor performs several tasks in parallel and independently, the execution of these depends on the interaction with the peripherals Input / Output (I/O), when the microprocessor executes some task it uses the memory to store temporary data during its execution. All tasks interact with the peripherals available on the DE10 Standard card, are shown in the Fig. 1 and were described below:

- Task 1 and 2 read values of the switches, write on the 7-segment display and turn on the Light-Emitting Diode (LED). This is shown in Fig. 2.
- Task 3 and 4 write directly in the memory registers and on the display. To write in several memory addresses, a VHDL block was used in order to travel through them and thus be able to write faster since the Nios II no longer has that execute the process. Shown in Fig. 2.
- There is an additional task known as the main one that is always running which is responsible for validating the execution of the tasks detailed above.

B. HPS implementation

Since the system embedded in FPGA works in parallel to the HPS, in this last one runs a Linux OS based on Debian, in which an algorithm written in C language was implemented that accesses the logical part, reads the memory and begins to see the changes of the same while the Nios II microprocessor makes use of it when executing tasks and interacting.

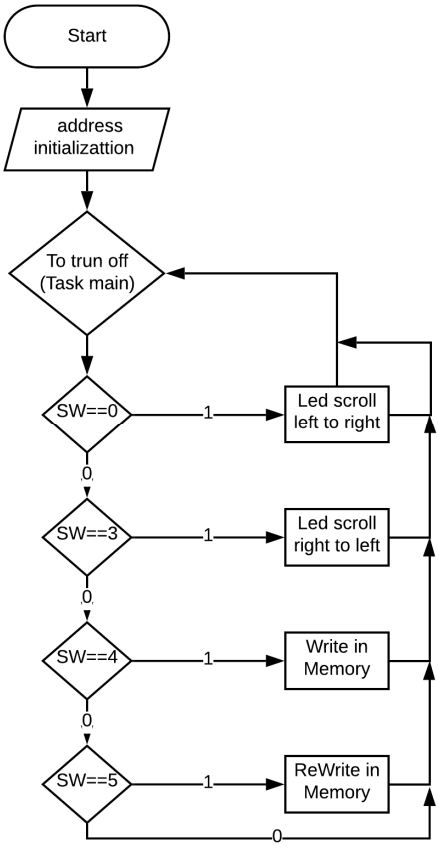

Fig. 2. Flow chart of the Nios II process.

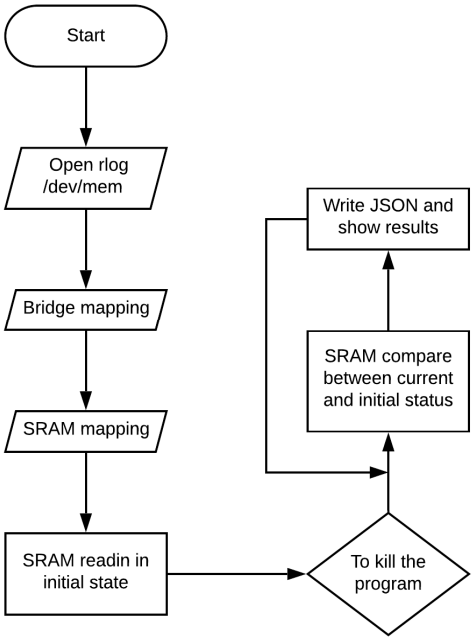

Fig. 3. Process flow chart in Linux (HPS)

To perform the correct monitoring, the algorithm verifies at high speed the changes in the memory registers and according to these changes a percentage is calculated allowing to validate its memory usage.

Fig. 3 shows the process flow of values saved in a JSON file to be able to visualize them in real time on a web page in graphic form and with this perform a better analysis. In addition, about 1670 samples were taken from the memory records during the execution of each of the tasks and then compared between them.

IV. RESULTS

The system allowed to observe the variations of the SRAM memory records incorporated in the FPGA part when it is running of several and different tasks like interactions with 42% of I/O peripherals as shown in table 1, write and read in memory. This also allowed estimate a percentage of use and a time of execution of the tasks, as well as the time that in that the Nios II processor writes in random and specific records, in parallel the ARM from the HPS is validating this writing by reading these records at high speed. As a contrast a similar algorithm was carried out in the part of the HPS to write in the Double Data Rate 3 (DDR3) and verify its usage with respect to the SRAM.

TABLE I. RESOURCES USED OF FPGA

Name	Used	Total	Percentage
Logic utilization (in ALMS)	1544	41910	4.00%
Total pins	210	499	42.00%
Total block Memory bits	535808	5662720	9.00%
Total PLLs	1	15	7.00%
Total DDLs	1	4	25.00%

Fig. 4 shows the average percentage of SRAM consumed by each of the tasks when they are executed. The task2 consumes much more since it is written directly in the records, while the others only interact with the peripherals who already have their own addresses on the card and should not load all these in memory.

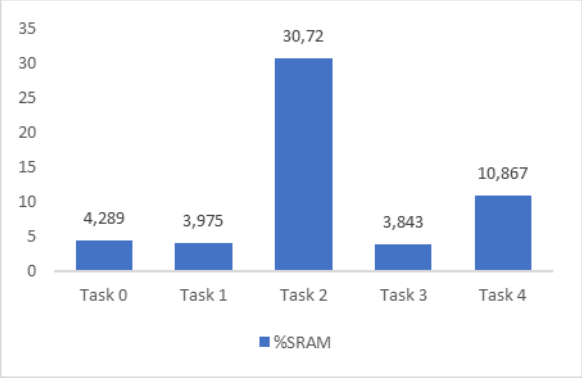

Fig. 4. Average SRAM usage when executing tasks.

In addition to the task4, that fulfills the main role, it consumes a little more than the one of the peripherals since it has to validate the execution of the other tasks so it must load the entire code in memory every time it is executed.

Fig. 5. Shows the time [ms] it takes for memory to load the data of each of the tasks when they are executed, and proportionally it is observed that the tasks that consume the most memory are the ones that take the longest time in loading your data into it.

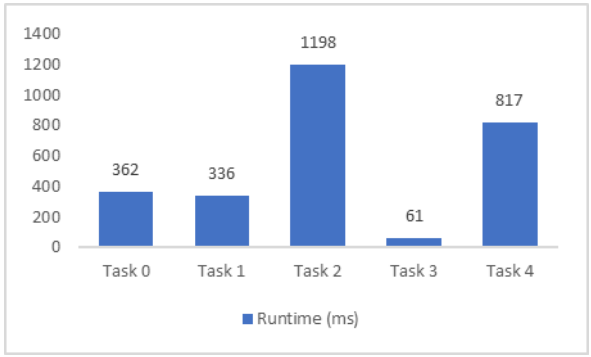

Fig. 5. Task Execution Time in SRAM

At the beginning, DDR3 memory usage by task 0 remains constant, because no instance of the webpage has been opened, but as several instances open, the percentage of memory usage increases. To perform tasks 1, 2, 3 and 4, the memtest tool was used, which covers the amount of memory that was put in it, this will make the use of DDR3 memory vary according to the amount that is being used for the execution of each of the tasks. For the task1 100MB, task2 118MB, task3 65MB and task4 115MB were placed, the percentage of DDR3 memory that was used to execute each task as shown in Fig. 6.

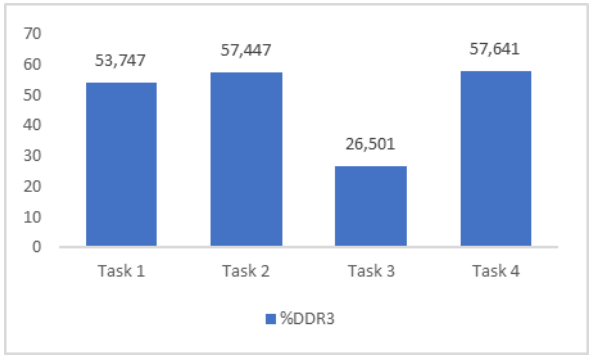

Fig. 6. Task usage percentage in DDR3

Fig. 7. shows the average time it takes for the DDR3 to reserve space to proceed to write on it, you can see that in the initial state of the OS only serving clients that connect to the web server the average time is the lowest of all hovering around 113 ms, again it is observed that as more space is required to write more time it takes to reserve this space.

V. DISCUSSION AND CONCLUSIONS

Thanks to the chips that have a portion of SoC and HPS such as the Cyclone V and taking advantage of the resources that the DE10 Standard card has, it was possible to implement these applications that work in parallel, which allows to use the HPS to monitor the embedded system in FPGA. All this is possible thanks

to the communication established between the two parties.

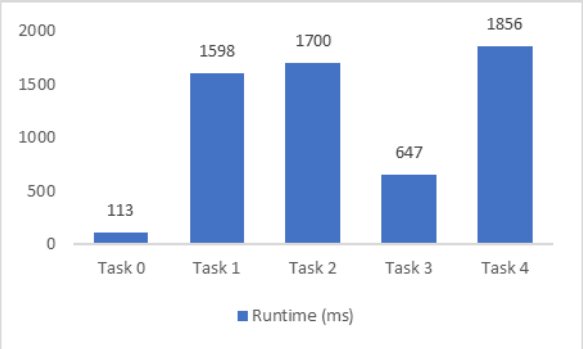

Fig 7. Average Execution Time in DDR3

Fig. 8 shown the SRAM is working in the logical part executing several tasks and it is validated that as time passes the memory usage increases. In addition, the writing times will depend on the amount of memory to be written and this varies according to the task that is being executed by the user or those that he has programmed in the Nios II.

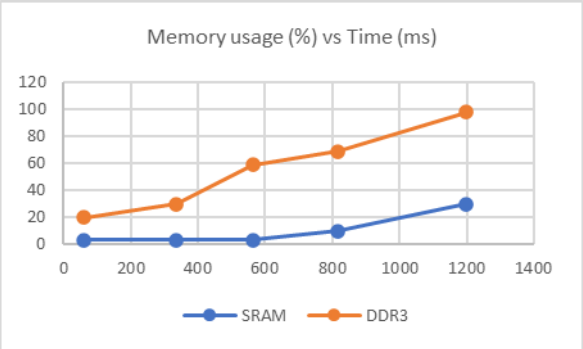

Fig. 8. Comparison in Usage of memory vs. Time

As for the DD3, it is executing the Linux OS as a basis and additionally, a size proportional to the size of the SRAM is reserved for the respective comparisons, so it is observed that it has a higher memory usage and longer response times.

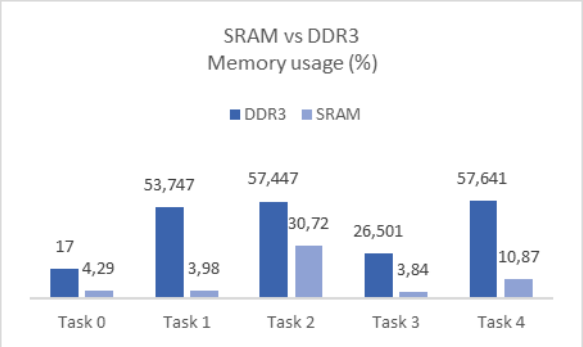

Fig. 9. Comparison of memory usage

It should be considered in this comparison that the DD3 in addition to running the OS, also has the web server implemented which memory usage varies according to the clients that are connecting to the webpage where it can be seen the memory monitoring of the embedded system. Also, thanks to the part of the HPS it is possible to monitor the memory of the embedded system without affecting its memory usage.

Fig. 9 shown the SRAM is not under the same workload since it is only responsible for storing what Nios II needs for the execution of the tasks.

Finally, it was consider that the HPS portion to be very important for a clean monitoring not only of the SRAM but also of any core that is implemented in the FPGA portion, since if this application is implemented on a chip that only has FPGA the application would affect the memory usage and performance of it, therefore you could not have completely reliable results.

REFERENCES

[1] Michael Daum (2017). GSRD v13.1 - Device Tree Generator. [online] RocketBoards.org. Available at: https://rocketboards.org/foswiki/view/Documentation/DeviceTreeGenerator131 [Accessed 25 Sep. 2019].

[2] Terasic. (2017). LINUX X64 INSTALLATION. En DE10 Standard Control Panel(14-39). United statess: Terasic.com

[3] Renovell, M., Portal, J., Figueras, J. et al. SRAM-Based FPGAs: Testing the Embedded RAM Modules (1999) 14: 159. https://doi.org/10.1023/A:1008326111919

[4] Wei Kang Huang ; F.J. Meyer ; Xiao-Tao Chen ; F. Lombardi, et al. Testing memory modules in SRAM-based configurable FPGAs (1997). https://ieeexplore.ieee.org/abstract/document/619399

[5] Intel FPGA (2015). Combining a Nios II ELF executable into a Hardware Project SOF file. [video] Available at: https://www.youtube.com/watch?v=joFaxLY-rUE [Accessed 26 Aug. 2019].

[6] Cyclone V Hard Processor System Technical Reference Manual. (2018). 4th ed. [ebook] San Jose, CA 95134. Available at: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_54005.pdf [Accessed 26 Jul. 2019].

[7] Intel.com. (2019). Intel SoC FPGA Embedded Development Suite User Guide. [online] Available at: https://www.intel.com/content/www/us/en/programmable/documentation/lro1402536290550.html [Accessed 26 Jul. 2019].

[8] Using Linux on the DE1-SoC. (2017). [ebook] Intel Corporation-FPGA University Program. Available at: https://software.intel.com/en-us/fpga-academic [Accessed 26 Jul. 2019].

[9] Wang, X., Zhu, Y., Ha, Y., Qiu, M., & Huang, T. (2017). An FPGA-based cloud system for massive ECG data analysis. IEEE Transactions on Circuits and Systems II: Express Briefs, 64(3), 309-313.

[10] C. Cedeño Z., J. Cordova-Garcia, V. Asanza A., R. Ponguillo and L. Muñoz M., "k-NN-Based EMG Recognition for Gestures Communication with Limited Hardware Resources," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, United Kingdom, 2019, pp. 812-817.