A fast codesign approach for low cost application-specific-system on programmable chip (SoPC) : Application to sensor network

Zied BEN SALEM
CES Laboratory, E.N.I.Sfax
ALPHA ENGINEERING TUNISIA
A33, 34 rue du Lac Lancey
Les Berges du Lac, 1053, Tunisie

Med. Wassim YOUSSEF
CES Laboratory, I.S.I.Tunis
2, Rue Abou Raïhan El Bayrouni
2080, Ariana, Tunisie

Mohamed ABID
CES Laboratory, E.N.I.Sfax
Route de Soukra, km 3
3052, Sfax, Tunisie

ABSTRACT
Systems on programmable chip, as system on chip, are composed by software and hardware. Therefore a codesign based approach is needed to meet the functional requirements. While classic codesign flows are long and involve complex and expensive design tools, this paper presents a fast but efficient codesign flow. To reduce the design cycle and the total cost of ownership (TCO), the presented design flow is based on massive reuse of Hw Intellectual propriety (IP) and Sw components, and involves costless design tools. As a proof of concept we designed and implemented a low cost standalone monitoring gateway for sensor network entirely based on royalty free Hw and Sw components. The hardware design was based on an AMBA bus system and a SPARC V8 compliant CPU subsystem. The software stack is based on a ported Linux kernel and a lightweight Round Robin Data Base Tool acting as a kind of middleware processing hardware-independent-software. The design was a submitted to series of stress test to evaluate its performance and capacity.

Keywords
Codesign; System on Programmable Chip (SoPC); design reuse; royalty-free-IP; sensor network monitoring.

1. INTRODUCTION
Although sensor network research was initially driven by military applications such as battle-field surveillance and enemy tracking, this technology is spreading quickly in civil applications. In fact, sensor networks are being deployed for many indoor applications like habitat, health, data centers, and factories. Actually, outdoor applications like environment observation and forecast system are particularly targeted by sensor networks to ensure a remote supervision. Indeed these systems could provide important information in real-time, like rainfall and water level information to evaluate the possibility of potential flooding [1]. Another application consists of deploying sensor network at Volcanoes edges to monitor remotely its activity [2].

Unfortunately sensor networks are usually expensive and complex to deploy for many considerations. The main fact is their non-standard communications protocols and their different electrical proprieties. Consequently, a unified management console for a global reporting on the status of the deployed sensors involves usually unaffordable costs. Using sensor networks over standard communication protocol like Internet Protocol (IP) is attracting today more and more users. In fact, Ip-sensor platforms are currently supplied with a monitoring gateway driven by a microprocessor and an operating system. These platforms generally support Hyper Text Transfer Protocol (HTTP) for the presentation layer, Simple Mail Transfer Protocol (SMTP) for notification, and Simple Network Management Protocol (SNMP) [3] for monitored data collection.

Nevertheless, such solutions have many constraints. Indeed, monitoring gateways are deployed on dedicated servers or in expensive hardware appliances. Besides, they require extra devices dedicated for routing and security [4]. Such devices are either not suitable for deployment in unconditioned environments, or over sized compared to the low data stream circulating in the Ip-sensor network.

On the other hand, today's market is offering more and more efficient and highly dense integrated circuits in terms of logic resources. Thanks to Very Large Scale Integration (VLSI) technologies, the integration of a complete system on a single chip (SoC) is easier and less expensive [5]. In this context, Field Programmable Gate Array (FPGA) has brought more flexibility to embedded systems design. Allowing the integration of a full system based on single or multi processor cores and dedicated hardware accelerators for the critical functional tasks [6], FPGAs are a suitable target for system on programmable chip (SoPC), able to implement complex systems designs.

Thereby, the idea to implement servers on embedded systems starts to take hold in many different applications. Despite the fact that several studies have demonstrated the possibility to implement low-end embedded application servers such as web servers with limited resources; the security aspects have been marginalized [8] [9] [10]. Consequently such systems are not suitable for standalone run-mode.

Today embedded application servers are taking place on critical applications like home appliances [11], renewable energy control logic [12], and sensor network monitoring. In many cases, these embedded devices are directly connected to the Internet in order to be remote-accessible. Thus, they are exposed to an increasing number of threats at each layer of the Open Systems Interconnect (OSI) Seven Layer Network Model.

In this context, we present a fast codesign approach for low cost application-specific-system on programmable chip (SoPC). As an application, we designed and implemented a low cost monitoring gateway node for Ip-sensor network. The design was based on royalty free Sw components and Hw intellectual propriety (IP) to decrease the design time cycle and the total cost of ownership (TCO).

The remaining part of this paper is organized as follows: section 2 presents a typical codesign approach toward a cost effective and fast design cycle. Section 3 details the implementation of sensor network monitoring gateway based on a SoPC and presents its associated experimental results.

2. Typical fast codesign approach for low system on programmable chip

2.1 Typical Application Specific System on programmable chip architecture

System–on–Programmable-Chip (SoPC) is composed by different processing elements and/or other specific electronic subsystems, implemented into a single reconfigurable logic chip like Field Programmable Gate Array (FPGA).

A SoPC may contain many types of computing subsystems, memories, input/output devices (I/O), and other peripherals. A SoPC designer could select hardware intellectual propriety (IP), and then connect them together around an on-chip bus system.

The on-chip bus connects a central CPU subsystem and standard components like memory, I/O peripherals, and application-specific components generally called Hardware Accelerator.

Because of increasing number of hardware components to be connected, today's SoPC are built today around complex hierarchies of buses, with multiple bridges between them. This approach has many advantages such as power savings, higher integration density, and lower systems costs.

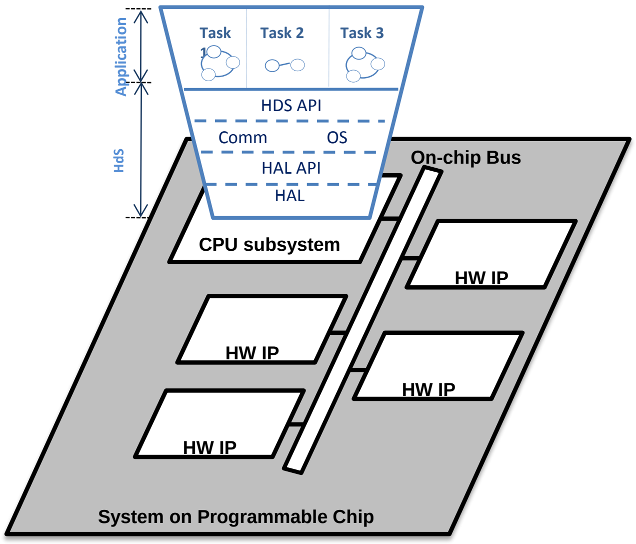

Figure 1. Typical Hw/Sw SoPC architecture based on system bus

In figure 1, the software stack running on the CPU subsystem implemented by the application specific SoPC must fulfill at least three requirements: reducing the size of memory, optimizing the overall performance, and minimizing the power consumption. A typical software stack is structured in two layers: the task/application-software layer and the hardware-dependent software (HdS) layer [13].

The HdS provides application- and architecture-specific services, like applications/task scheduling and communication, hardware resource management and control. Indeed, the HdS layer represents the software layer which is directly in contact with, or significantly affected by, the hardware that runs it; or can directly influence the behavior of that hardware [14]. The HdS integrates all the software that is directly depending on the underlying hardware, such as hardware drivers or boot strategy.

In depth, the HdS layer [15] is made of five components:

- HdS API: This application programming interface (API) is used by the subset tasks of the application to abstract the underlying HdS software layer.
- The operating system (OS): this component manages the sharing of the architecture's resources by providing services like tasks scheduling, context switch, synchronization, and interrupt management.
- Specific I/O communication software: This component manages the interaction between hardware components composing the design and the other software subsystems.
- Hardware Abstraction Layer: It is a thin software layer which depends on the processor architecture that will run the software stack. Implementing drivers for the I/O operations, the HAL is responsible for all the processor-specific operations like the boot code.

Thereby, a structured the software stack in several layers offers many advantages to SoPC designers, especially flexibility and portability. The flexibility means the possibility to re-use software components while changing the OS and/or the communication software components. Portability means possibility to re-use software components on other CPU subsystems architecture by changing the HAL software layer. Thus, meeting application-specific SoPC architectural requirements needs a suitable design flow.

2.2 Fast Codesign flow for system on programmable chip

Classic hardware/software codesign flow starts from a very high abstracted system specification in order to produce an efficient implementation that reaches the required performance and minimizes the TCO[16]. Besides, the large variety of technological targets motivates the designer to explore all possible solutions. This phase may take a lot of efforts and time. However, as systems become more complex, an increasing amount of time is spent exploring the hardware/software design space to find the optimal target architecture [17][18]. Figure 2 describes a fast but efficient SoPC-targeting codesign flow. It starts by a functional-partitioned system specification and takes advantage from "design reuse" based on royalty free components to reduce Hw/Sw design space exploration.

Design reuse consists of assembling ready-to-use components called Intellectual Property (IP) to design a more complex system [15]. Although classic design workflow assumes that a unitary design of each element and/or resource composing the targeted application as well as its respective unitary test sets is necessary, this phase becomes optional in a design reuse-based flow, or at least restricted to custom "in-house" made IP components.

The remaining part of this section presents a typical codesign flow for application-specific system on programmable chip, and lists the suitable design tools associated to each phase of the flow.

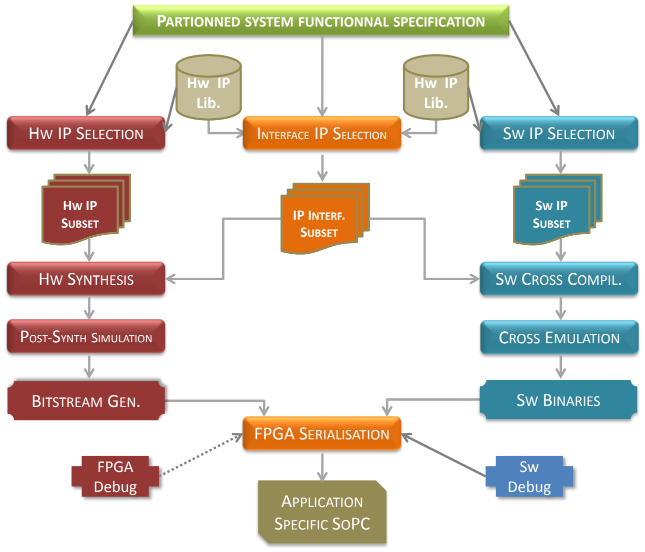

Figure 2. Fast codesign flow for SoPC target

(i)Partitioned system specification: This typical codesign flow, assumes that the design process starts from a given functional-level partitioned system specification. To ensure short time cycle design and low cost, this partitioning must consider the computational complexity of the application as well as the availability of reusable Intellectual Propriety (IP) for each function. The partitioned system specification lists the different functions that the final system would implement in order to fulfill its application requirements, and precise whether they are going to be implemented in hardware or in software.

(ii)Hardware, Software and interface IPs Selection: In this phase, we should select the suitable IPs to perform each function of the system according to the partitioning specification. To accelerate the design time, it would be interesting to reduce the integration constraints between the unitary components used in the design. Therefore, using a global IP library ensures inter-components compliance and easier integration. Indeed some global IP libraries [19] provide a generic HDL model mapping all its components, and a built-on configuration wizard that generates a configuration template specifying the selected IP to be implemented in the design. Further, in order to decrease the design cost, it would be interesting to use open source royalty-free-IP [20]. Indeed, thanks to the General Public License (GPL) licenses, and its derived licenses [21], designers can take advantage of many "ready-to-use", "royalty-free" and "open-source-coded" software components and hardware IP libraries.

(iii)Hardware Synthesis: This phase processes Hw IP subset specified and mapped at a structural and/or behavioral HDL level. The output of this phase is a low level Register-Transfer-Logic specification of system on a chip. Depending on the FPGA reference, this operation could be performed by FPGA vendor tools as well as third party vendors.

(iv)Software cross-compilation: The cross compilation operation transforms a set of software packages provided as high level language-coded source into a binary code for a specific execution environment. This phase is performed by CPU-Specific toolchain that compiles the source code to specific hardware architecture, and links it to the corresponding software libraries to interface it with a given operating system. While each CPU vendor provide a specific toolchain, the GNU toolchains powered by GNU Cross Compiler GCC [22], provided as free software fit the majority of the hardware architecture used in embedded systems industries like x86, SPARC-Vx, ARMx, MIPSx, PPCx and 68xx.

(v)Post synthesis simulation: The behavioral simulation makes a number of approximations to make a fast simulation. The concept of time is simplified since it does not consider the time to spread. Nevertheless, post-synthesis simulation simulates finely the code produced by the synthesis tool. It reproduces most of the problem that can occur on chip. While classic codesign flows assume that the behavioral simulation is necessary, the presented codesign flow in figure 2 avoids it. This choice was made in order to accelerate the design cycle, especially that the targeted design is IP-library-centric, and that its components have been tested by the library provider. This simulation could be performed by FPGA vendors IDE as well as many tools released under the GPL license like GHDL [23].

(vi)Sw cross-emulation: The cross emulation aims to imitate a physical behavior of a specific hardware architecture by software. Unlike the simulation that attempts to emulate an abstracted model by trying to extrapolate some of the unknown variables, the emulation reproduces the behavior of a model in which all variables are known to reproduce any constraints and/or functional system features. In the codesign context, cross emulation allows designers to validate at an early stage, unitary-software components and the major part of it, and even the whole software stack composing the design in some specific cases. In a context, of a single or homogenous multiprocessor on chip, QEMU [24] architectural cross emulator is a convenient tool because it emulates the most used CPU architectures in system on chip and provides a set emulation models for standard I/O peripherals like Ethernet controller, UART interfaces, and even USB and VGA controllers.

(vii)Hw/Sw debugging: Debugging is a key step in the design flow, to validate the obtained SoPC. While debugging an application, designers usually perform the following actions: Breakpoint set/stop/reset to inspect the design state, and to interact with; View/trace/analyze any signal or memory contents in the design. While many codesign flow targeting FPGA implementation present Hw FPGA debugging as a must, the suggested design flow consider this phase as optional. Indeed, a key distinction between Sw debugger and Hw FPGA debugger is that the Sw debugger provides a rich featured-debug environment while Hw debugger provides limited debug capabilities especially for embedded high/mid-end software. The GNU GDB debugger [25] provided by as free software by the GNU toolchain is one of the most suitable Sw debug tool for efficient and cost effective design thanks to its multiple features as well as its multi-architecture support.

3. Application for sensor network monitoring

The typical codesign flow described in the previous section could be used to implement efficiently an Ip-sensor network monitoring gateway in a SoPC. This section describes the implementation of a cost-effective sensor monitoring gateway. The first subsection details the functional requirements of sensors monitoring applications. The second subsection describes the hardware and software architecture of the sensor network monitoring gateway. Finally, the last subsections will present consequently the implementation results on FPGA and the global monitoring gateway performance through series of stress tests.

3.1 Sensor network monitoring functional requirements

An Ip-sensor network consists of many sensor nodes generally communicating using the IP protocol. Such networks are used to monitor critical and hostile environments. Therefore, an Ip-sensor monitoring gateway must provide a good quality of service ensuring reliability and security.

Figure 3 provides a general overview of a classic Ip-sensor network. The monitoring gateway has to provide at least three features. First, it has to ensure the collection of runtime events like availability and sensor-related physical metrics. Second, it has to send notification messages if an appropriate event occurs by several means like email or short message service (SMS). Finally it has to provide a complete dashboard to the sensor network supervisor reporting a continuously updated global status of the monitored environment.

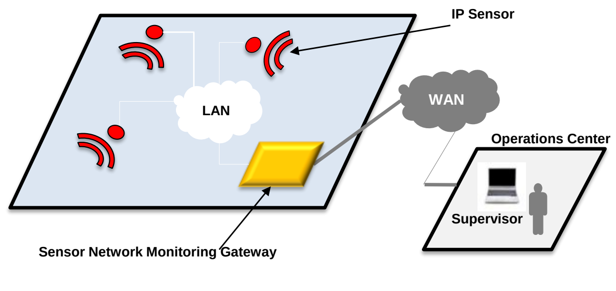

Figure 3. Overview of a classic Ip-sensor network

In reality, a global monitoring system is a distributed architecture composed of a set of sensors and a gateway node. While a sensor behaves as a simple agent providing its instantaneous status, the monitoring gateway must be structured in three layers. The lowest layer must ensure physical metrics collection from the Ip-sensors. The middle layer ensures the processing of the collected data and their transformation into metric data expressed in unit metrics. The top layer ensures reporting. It provides a coherent dashboard presenting the monitored environment metrics and listing its associated events. In the same time, the reporting layer performs the notifications operations.

This functional specification of the Ip-sensor network monitoring gateway shows the complexity of the application. Meeting efficiency, low cost and performance constraints requires hardware/software implementation compliant with the typical SoPC architecture through the fast design cycle described on the previous section.

3.2 Fast Hw implementation of the sensor network monitoring gateway

To satisfy complex application requirements, application-specific SoPCs are designed as a combination of processors, memory blocks, I/O peripherals, integrated on a single chip. Consequently the design could become time consuming and expensive. Dealing with integration problems and reducing the design cycle time is a major issue to bridge design complexity, increase designer's productivity and reduce time to market (TTM). Therefore the Hw design should be exclusively based on a single and reliable IP library providing all the required components for networking applications.

Furthermore, to master the total cost of ownership (TCO), the suggested design is based on open source IP. Organized in libraries, these components are more and more available. Thus a fast comparison between the most common royalty free IP libraries would be interesting to make the right choice for implementation. We will focus particularly on GRLIB [19], OPENCORES [26] and LatticeMico [27].

Generally, each IP library is especially featured by two main elements: the on-chip bus system and the CPU soft-core subsystem. Ensuring a centric role in the design, the global performance depends on the bus system. On the same time, the availability of, as much as possible, "ready-to use" bus-compliant peripherals is an attractive advantage for the designers, who prefers in general to focus only on the global system integration. The second characterizing element of an IP library is the CPU soft-core. This element is also particularly decisive for the system performance because it will run the software stack.

Therefore, the most interesting comparison criteria in a SoPC implementation would be CPU features: Pipeline, MMU, PMU, FPU and registers; and bus system (performance, availability of compliant IP like Ethernet MAC Core for networking). Symmetric Multi-Processors (SMP) support, which depends on both the bus and the CPU systems, could be also a decisive criterion because it guarantees to the designers a high scalability of CPU resources in case of complex software treatments.

Table 1, shows that GRLIB is particularly interesting, because it suitable for performance as well as space and/or power optimization. Indeed, the built-on LEON3 [7,19,28] soft-core supports Symmetric Multi-Processors (SMP), has 7-stage-configurable pipeline and implements up to 520 configurable register windows. This CPU softcore is also full compliant to the SPARC V8 standard. This fact gives a master key to the designers since they can take advantage of huge SPARC portable

GNU software packages because of its compatibility with the GNU cross-compiler GCC.

Table 1. Royalities free IP cores libraries comparaison

Features	IP Core Libraries		
	GRLIB	**OpenCores**	**LatticeMico**
BUS	AMBA	Wishbone	Wishbone
CPU SoftCore	LEON 3	OR1k2	LattMico3 2
Pipline	7-stages	5-stages	6-stages
Registers	40-520	32	32
FPU	Yes	Yes	No
MMU	Yes	Yes	No
PMUa	Yes	Yes	No
SMPb support	Yes	No	No
Ethernet MAC	Yes	Yes	Yes

a. Power Management Unit
b. Symmetric Multi-Processor

Thereby, we can conclude that the GRLIB IP library is suitable for the sensor network monitoring gateway implementation since it offers many read-to-use cores compliant with the AMBA 2.0 bus as well as a highly scalable soft-core CPU subsystem supporting until 8 CPU cores of 32-bits RISC SPARC V8 compliant.

Table 2 lists the hardware IP cores required to implement a SoPC dedicated for Ip-sensor network monitoring gateway application.

Table 2. List of the IP cores composing the SoPC

IP Core	Function	Library
LEON3	Soft-core processor	GRLIB
AHBRAM	Single-port RAM with AHB interface	
AHBROM	ROM generator with AHB interface	
GRETH	Gaisler Research 10/100 Mbit Ethernet with AHB I/F	
APBUART	Programmable UART with APB interface	
AHBCTRL	AMBA AHB bus controller with plug and play	
APBCTRL	AMBA APB Bridge with plug and play	
MCTRL	8/16/32-bit PROM/SRAM/SDRAM controller	
SDCTRL	PC133 SDRAM controller	
FoC[a]	Layer2/Layer3 Packet Filter	Custom Design

a. Firewall on a Chip

All theses cores are provided by the GRLIB IP Library except the Firewall on Chip (FoC) packet filter which is custom design [29]. This component has to ensure access level security on OSI layer 2 and layer 3, based on packets inspection. The received Ethernet packets are filtered according to their MAC addresses, IP addresses, protocols and/or ports. These rules are implemented on an Access Control List (ACL) ROM. This ACL-ROM is programmable only via a Universal Asynchronous Receiver Transmitter (UART) unconnected to the on-chip bus system to ensure higher security.

3.3 Fast Sw implementation for the sensor network monitoring gateway

The purpose of this section is to present the software specification of the sensors network monitoring gateway using a small footprint Web server. This specification must be as flexible and efficient to provide a fast, lightweight, and cost-effective implementation. Therefore, we present an architecture that provides simple but powerful application interface for Ip-sensor network monitoring entirely powered by GPL licensed software components.

The monitoring gateway acts as server providing a global environment dashboard in the form of HTML pages and a reliable alarm notifications engine. On the other hand from the side of sensors, the monitoring gateway acts as a client that collects, processes, and analyzes the information retrieved from the network sensors via (SNMP) flow.

Figure 4, presents the designed architecture. The software stack implements a hardware dependant software layer around a ported Linux Kernel to a SPARC V8 architecture, since it is very suitable for embedded networking applications [30,31] ; and a hardware independent layer made of four essential parts.

(i)The HTTP engine: It serves the client's request. The minimum requirement for an HTTP engine is that it must be compliant with HTTP specifications [32] for communicating with commercial Web browsers. Unlike general Web servers that start a new thread or process whenever a new connection is made, normally an HTTP engine supports multiple simultaneous users while running as a single process. The number of processes that the server requires can impact on both RAM usage, due to the stack space per task, and CPU usage. The httpd web server, as a light implementation of Apache web server provided by the Busybox package, is an interesting alternative because it works only with one single process.

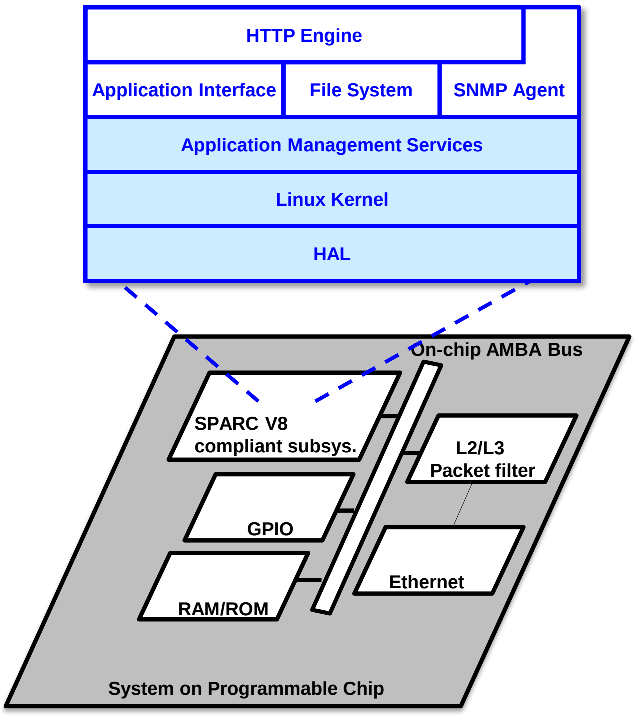

Figure 4. Monioring gateway architecture

(ii)The Management application: This module is the core of monitoring gateway. It processes and provides the collected physical metrics from sensors to the http engine, in order to present it in a coherent dashboard in forms of tables and/or charts. Besides, it ensures alarm notifications when a special event occurs. RRDtool, acronym for Round-Robin Database Tool [33] is suitable to implement the management application. Indeed RRDtool ability to handle time-series data is a key feature in monitoring application. In sensor network context, time-series data could be temperatures, humidity, water level, gas density, or any other discrete event like a physical intrusion or a smoke detection. RRDtool includes also functions to present the collected data in tables as well as charts format. The data are stored in a round-robin database (circular buffer). This feature is definitely the most interesting in our case since the system storage footprint remains constant over time. Finally RRDtool acts like a middleware that could be programmed by PERL, a simple and highly portable language very used for SNMP traps analysis.

(iii)The application interface module: It enables developers to add new monitoring functionalities. With any off-the-shelf Web authoring tool, it can merge Web documents with management application programs to generate specific dynamic management information. CGI type interface is provided for interacting with the embedded application. CGI type interface is used for generating Web document based on parameters submitted by operator through Web browser.

(iv)SNMP agent: It mediates between the sensors and the management application. It collects metric data when extracting management information base (MIB) [3] from the SNMP stream provided by IP sensor, in order to interface it easy with RRDtool.

3.4 Implementation results

To demonstrate the efficiency of the design, an Altera Cyclone2 EP2C35 package was selected as an implementation target because this series is featured by a small silicone area, and very low power consumption.

Table 3. Altera Cyclone 2 EP2C35 Resources Occupation Summary

Ressource	**Value**
Logic Elements	27.454
Combinatorials functions	24.217
Registers Logic	19.202
Pins	153
Memory Bits	185304
PLL	1
Global Ressources Use	97,31%

a. Firewall on Chip

Table.3, summarizes the resources occupation on the Cyclone2/EP2C35 FPGA. The high global resource usage rate shows that the cyclone 2, despite its small silicone area, was a convenient option for the monitoring gateway application since 97.31% of the available logic resources were used.

Finally, figure 5 shows a screenshot of the monitoring gateway dashboard which lists the Ip-sensors and graphs their metric. This dashboard is implemented in HTML and DHTML, and can be viewed through any web browser. The graphic layout of the dashboard is inspired from a classic monitoring platform [34].

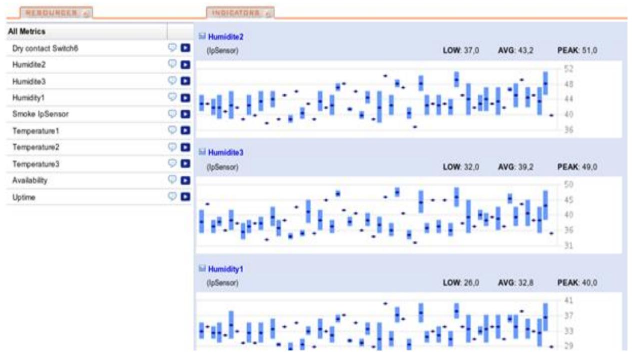

Figure 5. Screenshot of the Ip-sensor Monitoring Dhasboard

In order to evaluate the global performance of the monitoring gateway, the frontal web application server was stress tested. The system was configured to use 64MB off-chip SDRAM available on the FPGA development board, during the benchmark.

The stress tests are based on two scenarios:

- Ramp Test: gradual incrementing of the number of users
- Time Test: keeping the server loaded during a specified period and observing his reaction.

The Stress tests results showed that the designed IP sensor monitoring gateway supports up to 283 concurrent connections with an average bandwidth near of 2687kbit/s. This means that the presented design is suitable for Ip-sensor network monitoring application.

4. Conclusion

Sensor Networks are used in numerous indoor and outdoor applications. They have been adopted especially in critical environments like industries and weather forecast systems. Therefore a monitoring gateway node is needed to collect, process, and report sensitive data. To meet these requirements at a mastered cost, we presented a fast but efficient codesign flow, based on design reuse, and targeting low cost SoPC. Then we implemented a prototype of an Ip-sensor network monitoring gateway. To decrease the TCO, the designed system was based on an assembly of software (SW) and Hardware (HW) royalty free components. To reduce the power consumption, the selected FPGA target was Cyclone2/EP2C35. Finally the designed monitoring gateway was submitted to a series of stress tests to demonstrate its efficiency for sensor network monitoring. The future work is to specify an efficient design environment that automates the typical codesign flow presented in this paper to take advantage of all its features.

5. Acknowledgements

This work was supported by ALPHA ENGINEERING TUNISIA, Tunisia's leading Monitoring and High Availability Solutions and the national Federated Research Project (PRF) on security and embedded systems, leaded by Professor Mohamed ABID. A special thanks to Mr. Chaker ZAAFOURY, senior information systems security consultant in ALPHA ENGINEERING TUNISIA, who contributes in this work.

6. REFERENCES

[1] Yick, J. Mukherjee, B. Ghosal, D. "Wireless sensor network survey, Computer Networks"; The International Journal of Computer and Telecommunications Networking, 2008, ISSN:1389-1286.

[2] Werner-Allen, G.; Lorincz, K.; Ruiz, M.; Marcillo, O.; Johnson, J.; Lees, J.; Welsh, M. "Deploying a wireless sensor network on an active volcano", Internet Computing, IEEE, 2006, ISSN: 1089-7801.

[3] Case, J. Fedor, M. Schoffstall M. and Davin, C. "The Simple Network Management Protocol (SNMP) RFC 1157"

[4] Raskovic, D. Revuri, V. Giessel, D. Milenkovic, A. "Embedded Web Server for Wireless Sensor Networks", IEEE 41st South eastern Symposium on System Theory, 2009, ISBN:978-1-4244-3324-7.

[5] Tuomi, I. Bogdanowicz, M. "The Future of Semiconductor Intellectual Property Architectural Blocks in Europe", Joint Research Centre Institute for Prospective Technological Studies, European Commission, 2009, ISBN: 978-92-79-13058-8.

[6] Smach, F. Mitéran, J. Atri, M. Dubois, J. Abid, M. Gauthier, J.P : "An FPGA-based accelerator for Fourier Descriptors computing for color object recognition using SVM" Journal of Real-Time Image Processing 2, 2007.

[7] Ben Salem, Z. Megdich, M.F. Salhi, A. Zaafouri, C. Abid, M. "Implémentation et Evaluation des Performances d'Un Serveur Web Dédié sur SoPC autour de NIOS II", 11th International conference on Sciences and Techniques of Automatic control & computer engineering, 2007.

[8] Fioretti, F. Pasqualini, S. Andreoli A. and Pierleoni, P. "Permanent Switchboard Monitoring using Embedded Web Server", International Conference on Renewable Energies and Power Quality (ICREPQ'09), 2009.

[9] Mahanta, S. Sarma, A.K. "A comparative study on client server technology and web technology in design and implementation of an embedded system used for monitoring and controlling of physical parameters", Internet Technology and Secured Transactions ICITST, 2009.

[10] Grediaga, S. Llorens, A. Albero, H. " Performance Evaluation of FPGA-Embedded Web Servers", 14th IEEE International Conference on Electronics Circuits and Systems ICECS, 2007, ISBN: 978-1-4244-1377-5.

[11] Huaiyu, X. Ruidan, S. Xiaoyu, H. Qing, N. " Remote Control System Design Based on Web Server for Digital Home", Ninth International Conference of Hybrid Intelligent Systems (HIS), 2009, p457-461, ISBN: 978-0-7695-3745-0.

[12] Chen, Po. Ho, S. Lee, W. Chu, C. Pan, C. "An internet based embedded network monitoring system for renewable energy systems", 7th International Conference on Power Electronics (ICPE), 2007, ISBN: 978-1-4244-1871-8.

[13] Popovici, K. and Jerraya, A. "Virtual Platforms in System-on-Chip Design", DAC.COM KNOWLEDGE CENTER ARTICLE DAC'47, 2010.

[14] Pospiech, F. Hardware dependent Software (HdS). Multiprocessor SoC Aspects. An Introduction, MPSoC 2003, Online: http://www.mpsoc-forum.org/2003/slides/MPSoC2003_HdS_1.1.pdf.

[15] Popovici, K. Rousseau, F. Jerraya, A. Wolf, M. "Embedded Software Design and Programming of Multiprocessor System-on-Chip", 2010, ISBN 978-1-4419-5566-1.

[16] ABID, M. "Exploration of Hardware/Software Design Space in the Co-design Process", Real-Time Systems Magazine , 2001.

[17] Ktari, J. Abid, M. "A Low Power Design Space Exploration Methodology Based on High Level Models and Confidence Intervals". Journal of Low Power Electronics , 2009,

[18] Maalej, I. Gogniat, G. Philippe, J.L. Abid, M. "System Level Design Space Exploration for Multiprocessor System on Chip". ISVLSI, 2008

[19] GRLIB IP Library User's Manual, Gaisler Research, Online: http://www.gaisler.com/products/grlib/grlib.pdf, last visit 25/07/2010.

[20] A.K. Swain, A.K.and Mahapatra, K.K."Low Cost System on Chip Design for Audio Processing", Proceedings of The

International MultiConference of Engineers and Computer Scientists, 2010, ISBN: 978-988-18210-4-1.

[21] The GNU General Public License - GNU Project - Free Software, Online : http://www.gnu.org/licenses/gpl.html, last visit 25/07/2010

[22] GCC, the GNU Compiler Collection – The GNU project- Online: http://gcc.gnu.org/onlinedocs/ last visit 25/07/2010.

[23] GHDL guide; Online: http://ghdl.free.fr/ghdl/index.html, last visit 25/07/2010.

[24] Bellard, F. "QEMU, a fast and portable dynamic translator", Proceedings of the annual conference on USENIX Annual Technical Conference table of contents, 2005.

[25] Stallman, R. Pesch, R. Shebs, S. et al Published by the Free Software Foundation. Debugging with GDB, MA 02111-1307; ISBN 1-882114-77-9

[26] Opencore, Online: http://www.opencores.org last visit 25/07/2010

[27] LatticeMico Development Hardware, Online: http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/mico32developmenthardware.cfm last visit 25/07/2010.

[28] Großschadl, J. Tillich, S. Szekely, A. "Performance Evaluation of Instruction Set Extensions for Long Integer Modular Arithmetic on a SPARC V8 Processor", 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007, ISBN: 0-7695-2978-X.

[29] Ben Salem, Z. Youssef W., Abid, M. "Designing Secure Application Server on chip (SASoC) powered by open source intellectual proprieties Application to sensor network monitoring gateway", 2010, International Review on Computers and Software, Print ISSN: 1828-6003

[30] Yaghmour, K. "Building Embedded Linux Systems", Published by O'Reilly & Associates, Inc., 2003, ISBN: 8173666598.

[31] Gao, F. Li, F. Bao, S. Coll, X.W. "Analysis and implementation of secure console server based on embedded Linux", IEEE International Conference of Industrial Technology, 2008, ISBN: 978-1-4244-1705-6.

[32] Fielding, R. Gettys, J. Mogul, J. Frystyk Nielsen, H. Masinter, L. Leach, P. and Berners-Lee, T. "Hypertext Transfer Protocol - HTTP/1.1," RFC 2616 IETF HTTP WG, June 1999.

[33] RRDtool Official Documentation, Online: http://oss.oetiker.ch/rrdtool/doc/index.en.html last visit 25/07/2010.

[34] HQ system monitoring, Online http://www.hyperic.com, last visit 25/07/2010