Journal of Theoretical and Applied Information Technology
15th December 2021. Vol.99. No 23
© 2021 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN AND IMPLEMENTATION OF A SECURE AND AUTONOMOUS WEB SERVER USING FPGA FOR REMOTE CONTROL OF INTELLIGENT SENSORS

[1]KHAMLICH FATHALLAH, [2]EL GHOLAMI KHALID, [3]KHAMLICH SALAHEDDINE, [4]EL JOURMI MOHAMMED, [5]BENRABH MOHAMED

[1,5]LTI Lab. Faculty of Sciences Ben M'sik Hassan II-Casablanca University, Casablanca-Morocco

[2]National School of Applied Sciences, Research teams "TCI" LaSTI Laboratory, Sultan Moulay Slimane University, KHOURIBGA, Morocco

[3]National School of Applied Sciences, Research teams "SEIA"; LaSTI Laboratory, Sultan Moulay Slimane University, KHOURIBGA, Morocco

[4]Department of Telecommunications, Networks and Computer Science, National School of Applied Sciences, Chouaib Doukkali University, El jadida, Morocco

E-mails : [1]khamlich.fathallah@gmail.com, [2]k.elgholami@usms.ma, [3]s.khamlich@usms.ma, [4]eljourmi.med@gmail.com, [5]benrabh@yahoo.fr

ABSTRACT

This paper introduces the design and implementation of an FPGA-based webserver to communicate with sensors in smart cars. It's a better solution to replace traditional web servers in terms of processing speed, cost and power consumption. Our solution created using FPGA technology (i.e., Cyclone2, device EP2C70F896C6) is connected to the network and can play the role of a central WEB server achieving real-time remote processes control or remote data transmission via the Internet. The main contribution of this work is creating an embedded WEB server using RISC processors called Nios II configurable for general use. This Nios II, which is a soft-core processor developed by ALTERA, can models specific processors using HDL and then can be customized (and synthesized) for any application. Network connectivity is tested between an embedded Web server on Nios II and a standard web client. Messages sent from the client-side can be displayed over LCD on Webserver. The client can send commands to the board for controlling IO's, for reading from RAM, and for writing on RAM. The web server consists of an HTML interface, a MySQL database that contains user queries and results, a set of databases, a library of FPGA configurations, encryption algorithms, a responding host application user requests, and an FPGA coprocessor to speed up the alignment operation sequence.

Keywords: *HTTP / TCP, Nios II, FPGA, Soft Core Processor, Embedded Web Server, AES, RSA, Real-Time.*

1. INTRODUCTION

Nowadays, thanks to the Internet and embedded web servers, it is possible to carry out technological remote monitoring operations at a very low cost [1]. A web server is a computer server that allows you to order, store and publish web pages, basically written in HTML, on the internet or intranet. These web pages can be requested by any client web browser. The language used to exchange data between a web client and a web server is the standardized HTTP (Hypertext Transfer Protocol) [2] communication protocol as simplified in figure 1.

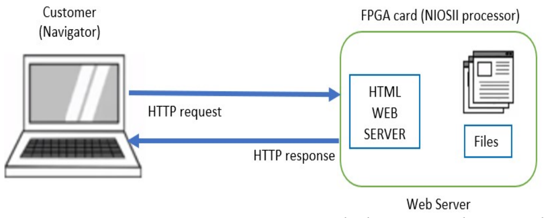


Figure1: Communication principle between client and FPGA webserver

The web server can be either software, hardware or even a combination of both. At the hardware level, the webserver is a computer that stores the files that belongs to a specific website (e.g., HTML documents, images, CSS sheets and JavaScript files) [3].

This research field present several challenges related to embedded systems developpement and types of targeted applications. In fact, IoT application, that are mainly based on embedded systems are evolving very fast wich requires solutions with a fast and low complexity implementations. In addition, being embedded, the system is constrained by hardware ressources limitations such as memory, CPU, etc. Besides, in many IoT applications, devices are not mainly powered and requires low energy consumption.

In this we fucus on a specific use case represented by an embedded web server for remote monitoring, using the Nios II embedded processor. This allows us to create a high-performance computer-based web server based on an FPGA card. The latter provides web pages to any other client on the network. It can also intelligently control the inputs/outputs of the Web server (e.g., reading data according to sensor inputs). The choice of the FPGA technology to build our hardware web server is an ideal solution to reduce costs. Indeed, one of the biggest challenges facing developers in electronics and embedded systems is to select a processor that meets the needs of their applications [4] [5]. With literally hundreds of integrated processors available on the market, each with a different set of peripherals, memories, interfaces, and performance, designers often end up buying more processors than necessary (to get the right mix of peripherals, interfaces, etc.) or settle for less. FPGA technology makes the embedded web server small (and then portable), flexible, reconfigurable and reprogrammable with the benefits of good customization, cost, effectiveness, integration, accessibility and extensibility [6]. This technology allows us to design the hardware, the software and the kernel simultaneously, which greatly reduces the design cycle [7].

Integrating an embedded web server is a remarkably powerful mechanism for easily implementing a variety of key features and functions in your device. Although the word "server" is often associated with heavy elements, the approach of the Nios II softcore processor allows designers to build efficient web servers with a reduced footprint. The embedded web server allows users to monitor and control their embedded applications using any standard browser. According to the TCP/IP model (which is at the heart of network interconnection communications [8], and the practical model running the whole Internet), the client-server web communication uses HTTP application protocol over TCP transport protocol over IP routed protocol. On the client-side, we only need the server's IP address or URL (Unified Resource Locator) so that the web browser navigate the web pages. Our implementation was tested and validated and shows no need for a dedicated high configuration computer to run a webserver. We consider this approach as the fastest and cheapest solution with low power consumption to replace traditional web servers.

With the Nios II softcore processor approach, designers can create a "perfect fit" in terms of processors, peripherals, memory interfaces, performance characteristics and cost. This is can be accomplished using a Cyclone II FPGA card from Intel-ALTERA, and create a custom system-on-chip or a System on Programmable Chip (SOPC). SOPC designers benefit from great versatility in product characteristics, performance, costs and life cycle management.

The rest of this paper is organized as follows. In section 2 we will review previous works using FPGA to create processors for various applications. Section 3 introduces the concept of hardware-software co-design. In section 4, we present the design and characteristics of our FPGA-based web server. Section 5 present an example use case of our web server, followed by a performance evaluation in section 6. This paper ends with a conclusion and future work.

2. LITERATURE REVIEW

Work has already been done for similar reasons by some researchers. we studied and examined these existing implementations before designing our system. Some of these works are summarized in this section.

Raghuwar and Asati designed in [9] an embedded web server based on MicroBlaze using the Xilinx FPGA development board. The system was connected to the internet for remote control and monitoring of a desktop computer.

Nandini et *al.* developed in [10] an FPGA-based embedded web server using a softcore processor (implemented on a Zynq MPSoC FPGA/Xilinx ZCU102). The proposed web server aims to control the GPIO using a client/server architecture. For network communication, the authors used the open-source Lightweight IP (LwIP) standalone TCP/IP stack over Ethernet.

Thapliyal and Kumar described in [11], based on a network of sensor nodes, the design of a prototype for monitoring and movement of parameters in real-time. A system for detecting critical/restricted compartments on platforms navies with data logging capacity (possibility). Raspberry Pi was used as the main server in the system and is responsible for connecting the sensors to the internet.

Selvi and Rathnakannan proposed in [3] the design of the integrated web server on the flexible MicroBlaze kernel available on the Spartan 3a FPGA kit using Xilinx Platform Studio. They declared it as a special type of file server and indicated that the system allows us to control and monitor the integrated system anywhere, away from the field of work.

Joshi et *al.* Described in [12] the implementation of a Web server using the Altera Nios II software kernel to download data relating to the department. They claimed it to be the best and cheapest solution with reduced power consumption to replace traditional web servers.

A.Bilakanti [13] proposed a fully homomorphic cryptosystem based on the learning with errors (LWE) problem, which is a modified scheme of the BVG cryptosystem for image encryption that provides fewer computations over the cloud. This scheme modifies basic LWE based on fully homomorphic encryption to process decimal inputs directly as like in BGV cryptosystem.

In [14], S. Chandel and Patel use the RSA algorithm to encrypt images to ensure image security in communication. The selected image file is split. Each part is encrypted using the RSA algorithm and then combines all the obtained cryptograms to produces an encrypted image. The decryption process follows the same scheme. To evaluate the security of the proposed approach, histogram and entropy are used. The experimental results illustrate the effectiveness of the proposed method.

3. HARDWARE/SOFTWARE CO-DESIGN

The Hardware/software co-design is an interesting concept that allows creating application-specific hardware (dedicated hardware). This paradigm gained more popularity thanks to the increasing similarities between hardware and software design (e.g., the hardware circuit can be described using modeling or programming language) [15]. It tries to exploit the synergy of hardware and software to optimize and/or satisfy design constraints such as performance, cost and the power of the final product [16].

However, in current electronic systems, we have to make many trade-offs, and mostly in conflicting directions; whether to go for more hardware or more software in this design. Table 1 summarizes the main driving factors for each direction in hardware/software codesign [17].

Table 1: Driving Factors in Hardware/Software Codesign

On-ship dedicated hardware	on-chip software
Performance	Design Complexity
Energy Efficiency	Design Cost
Power Density	Shrinking Design Schedules

4. WEBSERVER IMPLEMENTATION

In this work, we used the Nios II Soft-Core processor, which is a microprocessor fully described in software (generally in HDL hardware language) and that can be synthesized in FPGAs manufactured by Altera. A soft-core processor targeting FPGAs is flexible because its parameters can be changed at any time by reprogramming the device [18]. Our proposed web server is implemented using an embedded Nios II IP Core processor integrated within an FPGA circuit. The Nios II processor is a general-purpose 32-bit RISC processor with an integrated peripheral architecture [12]. The Nios II processor system is equivalent to a microcontroller or "computer on a chip" that includes a processor and a combination of peripherals and memory on a single chip. To implement our embedded web server, we used the Altera card, Cyclone II EP2C70 [12]. This FPGA contains 68,416 logic elements, 250 blocks of 4 Kbits RAM, 150 on-board multipliers, 4 PLLs, 622-pin I/O and can operate at a maximum frequency of 402.5MHz.

4.1. Creation of a Nios II hardware system in fast version:

Our Nios II hardware processor, created by the Qsys or Sopc-Builder tool, is composed of a Nios II processor core, a set of peripherals created on the FPGA circuit, a memory on a chip and a communication controller with a memory external circuit, all implemented on a single Altera FPGA circuit by Quartus II software [19]. The latter is valid for creating and implementing the SoftCore processor in all families and devices manufactured by the Intel-ALTERA company. It automates the task of integrating hardware components into a larger system. It can specify system components in a graphical user interface (GUI) and automatically generates interconnection logic.

Basic Architecture: The Nios II processor and the interfaces necessary to connect to other chips on the DE2 card are implemented in the Cyclone II FPGA circuit. These components are interconnected via the interconnection network called Avalon Bus. Its main advantages are the simplicity of its architecture, a design intended to optimize the use of material resources and multi-master support. The Nios II processor diagram created for the webserver application and the controllers for communicating with devices on the FPGA card are shown in Figure 2.

The elements used to create our Nios II hardware processor are:

– Nios II/Fast heart
– A JTAG UART interface, (communication controller between the FPGA circuit and the programmer linked with a computer. It allows performing operations such as downloading programs into memory, starting and stopping execution, defining breakpoints and real-time execution trace data collection. Indeed, all parts of the Nios II system and that are

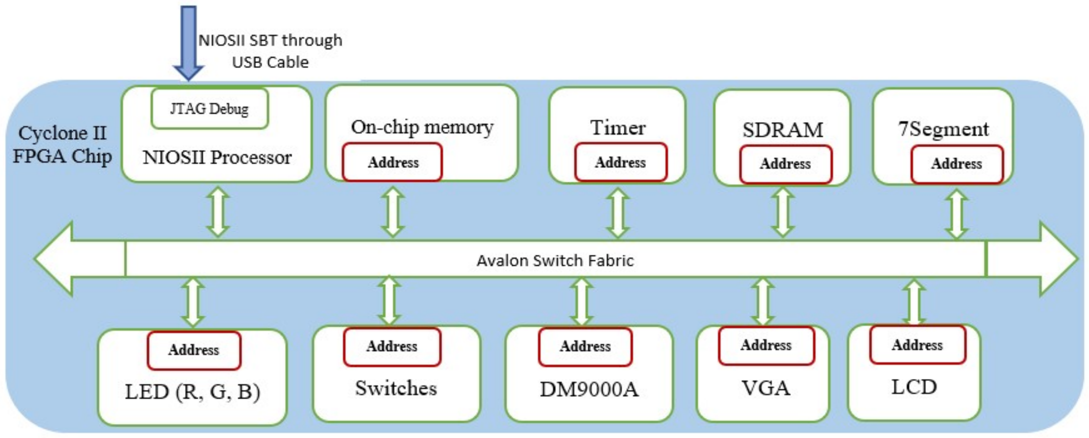


Figure 2: Communication diagram between Avalon Bus and the controllers of our Nios II HW processor.

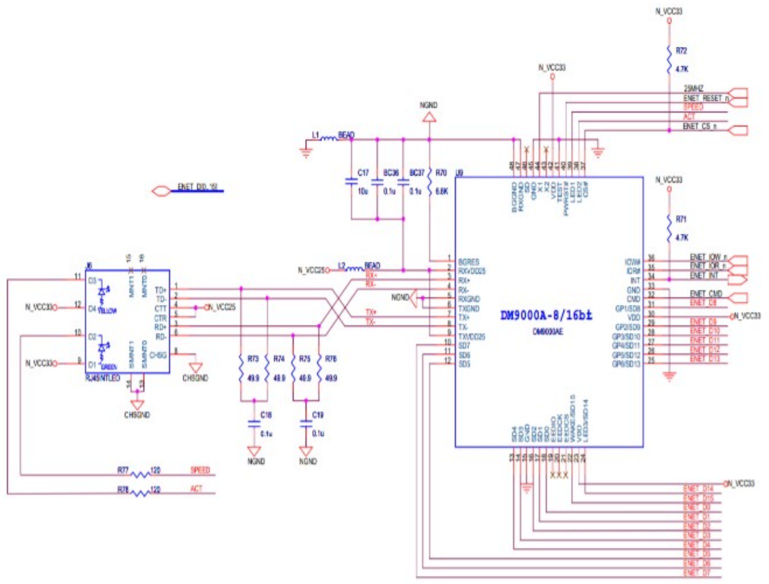


Figure 3: Ethernet converter scheme

implemented on the FPGA chip are defined using a hardware description language (HDL).

- Chip memory (internal program memory of FPGA circuit).
- An SDRAM controller (an external SDRAM memory controller for an FPGA circuit, used to store data).
- A DM9000A Ethernet controller (the communication interface with the DM9000A Internet controller).
- An LCD controller (controller of the LCD display used to display MAC and IP addresses and other parameters).
- A PIO interface to control the green LEDs (LEDG) on 9 bits.
- A PIO interface to control the red LEDs (LEDR) on 18 bits.
- A PIO interface to control the 18-bit switches.
- A SEG7_LUT_8 control interface to control the 7 segment displays (seven_segs).
- VGA controller.
- A PLL with three outputs:
    - 50 MHz which will be the clock of the Nios II processor
    - 50 MHz with -65 ° phase shift which will be the SDRAM clock
    - 25 MHz which will be the clock of the DM9000A circuit

Avalon interface: it simplifies system design by easily allowing the connection of several components in an FPGA. The Avalon switch fabric enables multiple simultaneous data transactions for unmatched system throughput.

Note: The JTAG UART interface is used to connect to circuits that provide a USB (Universal Serial Bus) Blaster link to the programming computer to which the DE2 card is connected. The DE2-70 card also provides Ethernet support via the DM9000A controller. Figure 3 shows the diagram of the Fast Ethernet Interface and the associated pin assignments between the RJ45 port and the FPGA circuit.

We used the FPGA circuit from Intel-ALTERA to create the Nios II processor as hardware technology. When determining the generation of the system (shown in Figure 2) in the SOPC-Builder tool, we go to Quartus II software to connect the inputs/outputs of the Nios II processor with the inputs/outputs of the FPGA circuit. After the interconnection of these interfaces, we move on to the simulation of our Nios II processor in order to check errors. When the software finishes the simulation without errors, we obtain as a result an extension file ".sof" which defines the Nios II hardware processor. This file contains all the required instructions to modify the internal structure of the reprogrammable FPGA circuit implementation. Figure 4 shows the complete design corresponding to our Nios II processor created and implemented in the FPGA circuit specifically for our web server application.

After the simulation of our hardware processor (Figure 4), the Quartus II software will send the ".sof" file of this processor to the FPGA circuit.

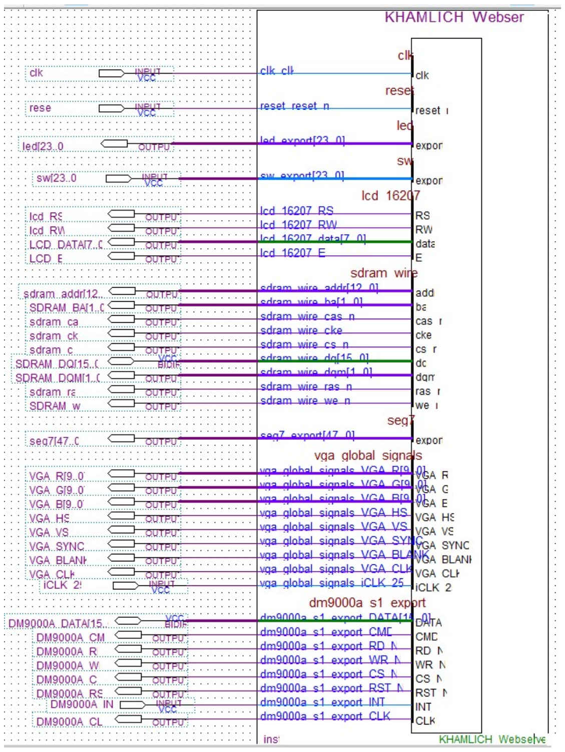
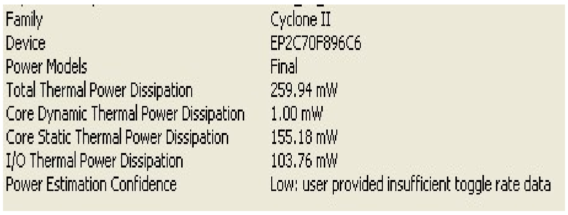


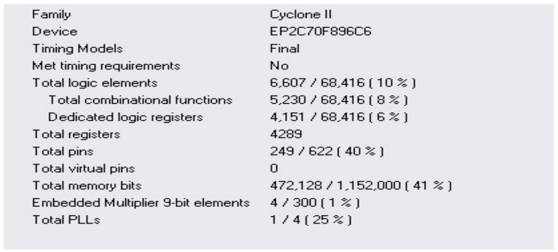
Figure4: Nios II Hardware processor created by Qsys and Quartus II software

When the hardware compilation of our processor under Quartus II software is completed without any errors, we get a set of valuable information (i.e., the energy and the material space reserved by our hardware architecture in the FPGA circuit) as illustrated in figures 5 and 6.



Family	Cyclone II
Device	EP2C70F896C6
Power Models	Final
Total Thermal Power Dissipation	259.94 mW
Core Dynamic Thermal Power Dissipation	1.00 mW
Core Static Thermal Power Dissipation	155.18 mW
I/O Thermal Power Dissipation	103.76 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Figure 5: power result of our Nios II equipment



Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,607 / 68,416 (10 %)
Total combinational functions	5,230 / 68,416 (8 %)
Dedicated logic registers	4,151 / 68,416 (6 %)
Total registers	4289
Total pins	249 / 622 (40 %)
Total virtual pins	0
Total memory bits	472,128 / 1,152,000 (41 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

Figure 6: material reservation of our Nios II on the FPGA circuit

Based on these outputs, we note that the hardware implementation of a single processor requires a reservation of a reduced space in the FPGA circuit (we used circuit version EP2C70F896C6). In the remaining space of the FPGA circuit, we can add other peripherals or physically integrate parallel processors or encryption algorithms, wich very usefull in IoT applications. Besides, the energy consumption is low wich is switable fow low anery applications. Or to reduce the overall system energy cost.

Finally, the Nios II processor is implemented in the FPGA circuit with the interfaces or communication controllers between the external FPGA circuit peripherals and our Nios II architecture. This is implemented using the Altera Quartus II environment where it is added to the Nios II interface, the DM9000A interface to connect the development board to an Ethernet network by the RJ-45 port, a storage memory, a controller VGA and a JTAG interface for downloading Hardware and Software processors. Communication between external devices and our architecture is shown in Figure 7.

After building the hardware part, a set of files are simultaneously created to facilitate the programming of our Nios II processor in computer language for the software part. We used Eclipse [20] software to create the Nios II software processor. In the next section, we will discuss the software part of our work.

4.2. Creation and implementation of Nios II Software Processor and our Web server:

After the creation of our hardware Nios II (Webserver.qpf) by the Qysys and Quartus II tools, we obtain a “.sof “ executable file which defines the hardware Nios II processor (following the internal modification of the FPGA circuit). To go to the software part, we used the file "KHAMLICH_Webserver.sopcinfo" in the directory KHAMLICH_HW / SW for the creation of the new project in the Eclipse software (see figure 8).

The Template selected in Figure 8 is a minimalist web server application using sockets based on the TCP/IPv4 stack from Interniche [21]. This application also uses the real-time operating system MicroC/OS-II from Micrium [22]. The web pages are stored as an uncompressed .zip file in the memory of the DE2 card. A basic file is located in the "KHAMLICH_Webserver\system" directory of our project.

All the IPs present in the hardware construction are listed in a file called system.h, which is generated by SOPC builder or Qsys. It is part of the libraries called by the Eclipse software. The program files, as well as the libraries of the Soft part, are illustrated in the software folder in Figure 9.

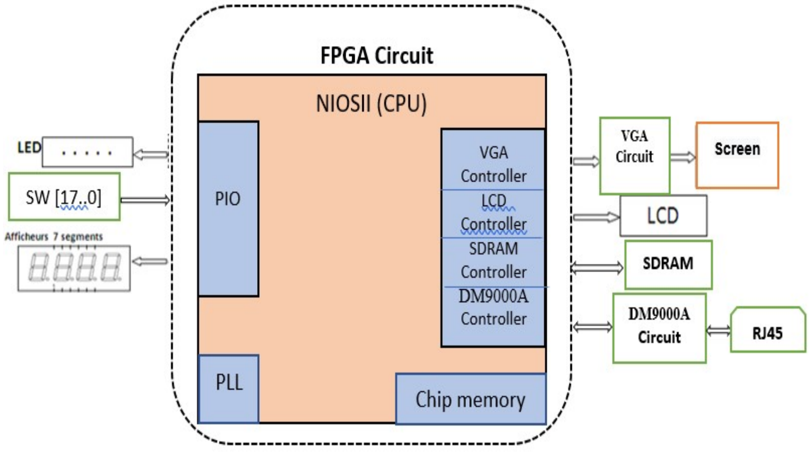
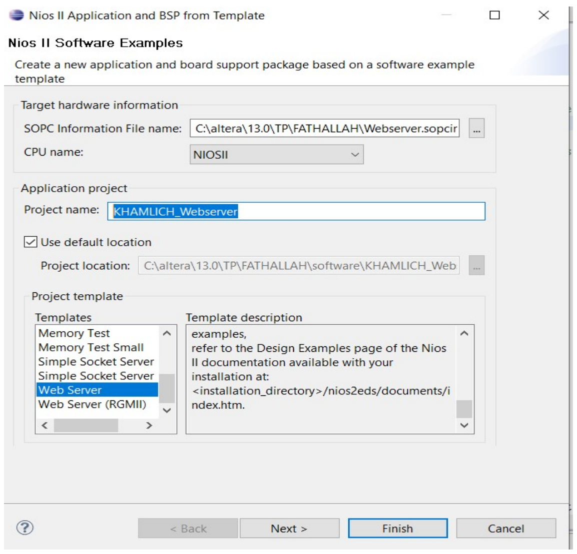


Figure 7: internal structure of our Nios II architecture

The contents of the main Software folder are the following subfolders (KHAMLICH_Webserver and KHAMLICH_Webserver_Syslib) containing the files of the programs, subroutines, declaration libraries, Web server, etc. These are files created in C, PHP, MySQL, etc.
For example:

– Creation of FPGA configuration libraries in the "KHAMLICH_Webserver_Syslib" folder and files with the extension .C and .H in folder “KHAMLICH_Webserver” in the form of subroutines (e.g., DM9000A circuit configuration, display on "Screen by VGA, LCD, 7SEGEMENT and LEDs".
– Main program creation int main in C.
– Creation and encryption of the web page files by languages (e.g., Php, MySQL ...).
– The website’s files are saved on the memory of the Cyclone2 FPGA card.


Nios II Application and BSP from Template

Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: C:\altera\13.0\TP\FATHALLAH\Webserver.sopcir

CPU name: NIOSII

Application project

Project name: KHAMLICH_Webserver

☑ Use default location

Project location: C:\altera\13.0\TP\FATHALLAH\software\KHAMLICH_Web

Project template

Templates: Memory Test, Memory Test Small, Simple Socket Server, Simple Socket Server, Web Server, Web Server (RGMII)

Template description: examples, refer to the Design Examples page of the Nios II documentation available with your installation at: <installation_directory>/nios2eds/documents/index.htm.

Figure 8: Creation of the new web server project on Eclipse.

Before creating the main program for our Hardware processor for the webserver application, we worked on an algorithm in the form of mathematical functions used to encrypt our data. The process of making the information unreadable for any unauthorized entity is called encryption. The result of the encryption is a ciphertext or a cryptogram. Reversing this process and recovering the original readable information is called decryption. Many methods of security analysis have been implemented on proposed techniques that used many types of images with many experiments.

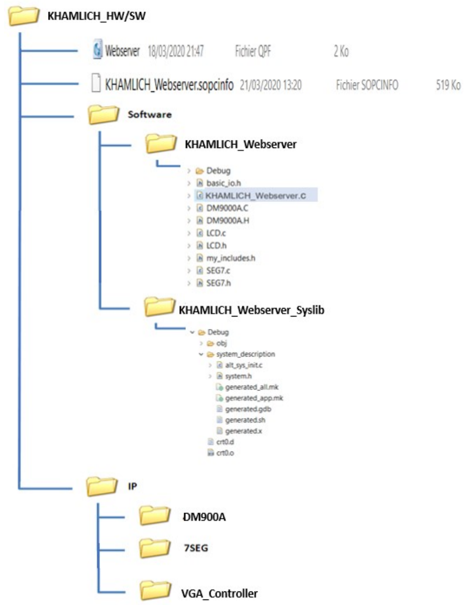


Figure 9: program files created by the Nios II IDE tool (Eclipse)

Our browser's HTTPS connections are based on a mixture of AES symmetric encryption [23] [24] and RSA asymmetric encryption [25]. The AES algorithm takes as input a block of 128 bits (16 bytes), the key is 128, 192 or 256 bits. The 16 input bytes are swapped according to a previously defined table. These bytes are then placed in an array of 4x4 elements and its rows are rotated to the right. The increment for the rotation varies depending on the row number. A linear transformation is then applied to the matrix, it consists of the binary multiplication of each element of the matrix with polynomials resulting from an auxiliary matrix, this multiplication is subject to special rules according to GF (28) (Galois group or finite body) as illustrated in the example of figure 10. This linear transformation guarantees better diffusion (i.e., propagation of the bits in the structure) over several turns. Finally, an exclusive OR (i.e., XOR) between the matrix and another matrix allows obtaining an intermediate matrix [23] [24]. These different operations are repeated several times and define a "round". For a key of 128, 192 or 256, AES requires 10, 12 or 14 turns respectively.

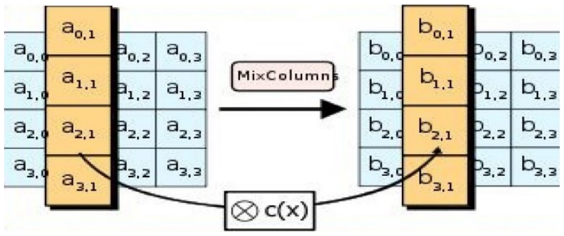


Figure 10: Example of the Mixcolumn operation

The implementation uses the VHDL programming language which is currently a is frequently used language and very convenient for FPGA [26].

To show that these changes give a better result in terms of speed and space than the subsequent effort, we evaluate the encryption /decryption codes) based on the Several AES Variants. The comparison considered two criteria: speed and area utilization. The design was implemented on a Cyclone III (EP2C70F896C6 model) device. The encryption block double_aes is shown in figure 11, where the main signals used by the implementation are displayed.

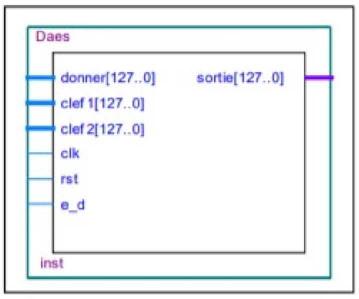


Figure 11: Double-AES (ciphering & decoding) block.

The design and the simulations are performed using the Quartus II software, and the obtained results showed that our implementation is efficient in terms of security and execution speed when implementing the algorithm in parallel with the Nios II processor. The main goal of this hardware implementation is security efficiency, faster processing speed, and a small hardware reservation in the internal FPGA circuit structure.

5. ARCHITECTURE OF OUR SYSTEM AND NETWORK PROTOCOL

We have configured the network card of our PC with a fixed IP address in the same subnet as the DE2 card. The Ethernet cable between the PC and the DE2 card is also connected (see figures 12 and 13). When you type on a web browser 192.168.1.12 in the address bar the following page opens:

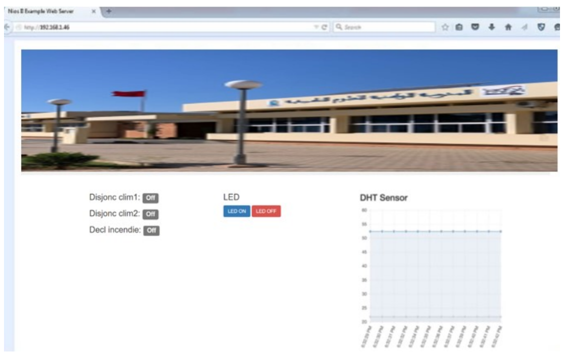


Figure 12: web control of sensors and input outputs PIO

The code was customized in order to allow the configuration and control of the webserver using pushbuttons. These changes also allow us to remotely control the 7-segment displays of the DE2 card. The created Hardware and Software processor implementing our proposed web server starts automatically and successfully allowing the client to download the published webpages and execute PHP programs stored in the FPGA card's memory. The exchange between two network elements is based on a time-triggered communication where the data flow is controlled by a global clock [27].

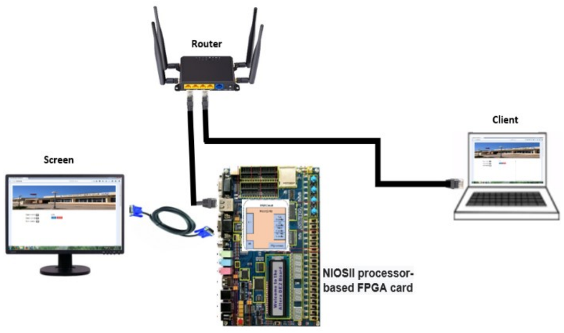


Figure 13: System Circuit diagram

The FPGA board is connected to the network using an Ethernet cable that is tied with the router through the RJ45 Ethernet port. All addressing and configuration of both the router and FPGA card are performed to successfully connect the latter to the Internet. Figure 13 shows the Diagram of the system network. That allows LAN (Local Area Network) clients to connect to our embedder web server (using any standard web browser). Accessing this webserver from the Internet requires only a public IP address on the server-side. This simple topology can be extended for other advanced applications like smart cars and smart cities, which will be covered in our future works.

Table 2: change of addresses by switches and Push Button

Pushbuttons	Description of the function
SW [0]	ON, activation of the display of the IP address on LCD OFF, activation of the display of the MAC address on LCD
SW [1]	ON, activation of change of IP and MAC addresses. OFF, change stop and end of the menu display.
SW [2]	ON, use a static IP address OFF, get the address of the DHCP server
PB [3]	To decrement the last two bytes of the static IP address or the last byte of the MAC address.
PB [2]	For incrementing the static IP address or MAC address.
PB [1]	Is used to move the LCD cursor to the left
PB [0]	Used to move the LCD cursor to the right

6. PERFORMANCE EVALUATION

In this part, we will discuss the operating principle of the FPGA card based on a Nios II embedded processor and its performance. The version of the FPGA card studied is the DE2-70 version which is designed by the Intel-ALTERA company.
As an example, we use Toggle Pushbuttons in our card to set the static IP address and MAC address (i.e., EUI-48).

Example:
Static IP address: 192. 168. ---.---
MAC address: A2-B2-C3-BD------

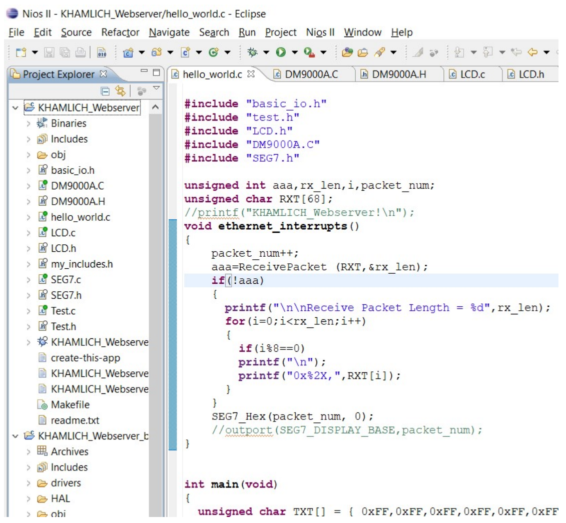
Note: To avoid MAC addresses conflict with existing commercialized network cards, the manually configured MAC address should set the second-least-significant bit of the first octet of the address, also named U/L bit, to 1 (i.e., Locally Administered).

Table 2 summarizes the switches and Push Button's states and the action for each state.

The role of the switch SW0 is to activate the LCD display menu (if SW0 = 1 the LCD will display the IP address if not it will display the MAC address.

The 16x2 LCD display is used to display MAC and IP addresses. The 7-segment displays are used to display the time. A VGA controller was created to ensure the communication between the FPGA card and a screen for displaying the web page. LEDs are used to indicate the speed of the internet network.

After the design and production of our processor, created for a Web Server application using the presented development tools, we obtained encouraging result, either at the material reservation level (Figure 14 and 6), Energy consumption (Figure5), processing and communication speed, or in terms of the easy configuration of Nios II. Which makes this implementation suitable for several Internet of Things (IoT) applications. The proposed processor's version can be easily modified and its internal architecture of this dynamic processor can be shared with other researchers and developers interested in our project.



```c
#include "basic_io.h"
#include "test.h"
#include "LCD.h"
#include "DM9000A.C"
#include "SEG7.h"

unsigned int aaa,rx_len,i,packet_num;
unsigned char RXT[68];
//printf("KHAMLICH_Webserver!\n");
void ethernet_interrupts()
{
    packet_num++;
    aaa=ReceivePacket (RXT,&rx_len);
    if(!aaa)
    {
      printf("\n\nReceive Packet Length = %d",rx_len);
      for(i=0;i<rx_len;i++)
      {
        if(i%8==0)
        printf("\n");
        printf("0x%2X,",RXT[i]);
      }
    }
    SEG7_Hex(packet_num, 0);
    //outport(SEG7_DISPLAY_BASE,packet_num);
}


int main(void)
{
  unsigned char TXT[] = { 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
```

Figure14: programs created and compiled by the Nios II IDE tool (Eclipse)

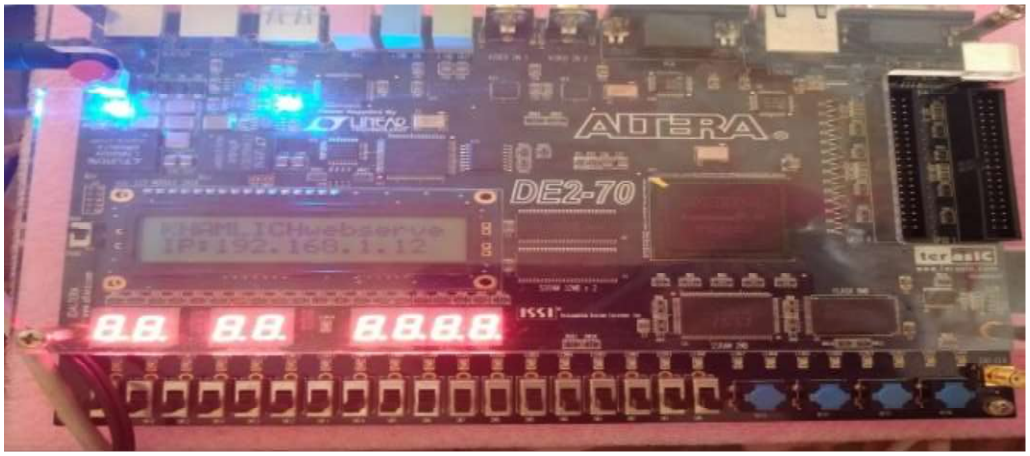


Figure15: Web server created by the FPGA card

Figure 15 illustrates the concretization of our Hardware/Software implementation of Nios II processor and web server algorithms.

7. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

The Internet of Things has now become a reality thanks to the known development in the field of embedded systems and communication technologies. We are seeing more and more new types of connected devices in the industry (e.g. Industry 4.0, Farming 4.0 …). However, in IoT environements, embedded systems should respond to specific constraints like low complexity, low hardware ressources and low energy consumption.

Inorder to addresse these issues, we proposed in this work, an embedded web server dedicated to affordable and versatile control systems. The proposed implementation can be used to control a large number of sensor nodes or instruments, either remotely via the Internet or within a private network. The design was carried out using a Nios II microprocessor integrated with the Cyclone II FPGA circuit.. The use of these technologies allowed us to implement a flexible system with a relatively short development time. The implementation of our system control application was programmed using C language reducing its complexity.We also animated our web application using several standard web programming tools such as JS, CSS, etc. In addition, the chosen interface can easily contributes to the acceptance of the system by its users. Besides, thanks to the concept of FPGA, we were abel to provide an application specific hardware implementation. Our proposed system achieved good results in terms of energy consumption and hardware resource reservation.

As aforementioned, the proposed implementation can be suitable for various IoT applications. Therefore, in our future research works, we aim to use the proposed imbedded web server in smart city applications, specifically, in an automatic road accident detection and control system. We also proposed to use external memory (e.g., SD card or USB key) to store the collected data from the sensors installed on a vehicle instead of using SDRAM memory. Thus, it will solve a technical problem where each of the sites will be hosted "at the root" of the external memory, making it possible to create links and run immediately in the production phase.. Besides, other main commands can be added to control more devices if they have the necessary communication channels to control the new slaves.

REFERENCES:

[1] J. W. Szymanski, “Embedded Internet technology in process control devices,” in *2000 IEEE International Workshop on Factory Communication Systems. Proceedings (Cat. No.00TH8531)*, Sep. 2000, pp. 301–308, doi: 10.1109/WFCS.2000.882562.

[2] “HTTP - Hypertext Transfer Protocol Overview.” https://www.w3.org/Protocols/ (accessed Apr. 03, 2021).

[3] V. Selvi and D. K. Rathnakannan, “Embedded Web Server using Soft Processor on FPGA Platform,” 2012.

/paper/Embedded-Web-Server-using-Soft-Processor-on-FPGA-Selvi-Rathnakannan/220de0427cd50db6e4f3ba622f3bb580a7ed6a75 (accessed Mar. 18, 2021).

[4] H. Cheng and H. Qin, "A design of IEEE 1451.2 compliant smart sensor based on the Nios soft-core processor," Nov. 2005, pp. 193–198, doi: 10.1109/ICVES.2005.1563640.

[5] T. Fukatsu and M. Hirafuji, "Field Monitoring Using Sensor-Nodes with a Web Server," *J Robot. Mechatron.*, 2005, doi: 10.20965/jrm.2005.p0164.

[6] S. Cuenca, A. Grediaga, H. Llorens, and M. Albero, "Performance Evaluation of FPGA-Embedded Web Servers," in *2007 14th IEEE International Conference on Electronics, Circuits and Systems*, Dec. 2007, pp. 1187–1190, doi: 10.1109/ICECS.2007.4511208.

[7] I. D. Agranat, "Engineering Web technologies for embedded applications," *IEEE Internet Comput.*, vol. 2, no. 3, pp. 40–45, May 1998, doi: 10.1109/4236.683798.

[8] A. F. Tenca and C. K. Koc, "A scalable architecture for modular multiplication based on Montgomery's algorithm," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1215–1221, Sep. 2003, doi: 10.1109/TC.2003.1228516.

[9] R. S. Soni and D. Asati, "Development of Embedded Web Server Configured on FPGA using Soft-core Processor and Web Client on PC," *Int. J. Eng. Adv. Technol. IJEAT*, vol. 1, no. 5, pp. 295–298, Jun. 2012.

[10] K. Nandini, A. Viswapriya, and H. Umadevi, "Development of embedded web server configured on ZCU102 FPGA using softcore processor," *Int. J. Eng. Appl. Sci. Technol.*, vol. 4, pp. 110–115, Jun. 2019, doi: 10.33564/IJEAST.2019.v04i02.020.

[11] A. Thapliyal and C. Kumar, "Development of Data Acquisition Console and Web Server Using Raspberry Pi for Marine Platforms," 2016, doi: 10.5815/IJITCS.2016.11.06.

[12] N. N. Joshi, P. K. Dakhole, and P. P. Zode, "Embedded Web Server on Nios II Embedded FPGA Platform," in *2009 Second International Conference on Emerging Trends in Engineering Technology*, Dec. 2009, pp. 372–377, doi: 10.1109/ICETET.2009.89.

[13] A. Bilakanti, Anjana N.B., Divya A., K. Divya, N. Chakraborty, and G. K. Patra, "Secure computation over cloud using fully homomorphic encryption," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Jul. 2016, pp. 633–636, doi: 10.1109/ICATCCT.2016.7912077.

[14] G. S. Chandel and P. Patel, "Image Encryption with RSA and RGB randomized Histograms," *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, vol. 3, no. 5, pp. 9750–9757, May 2014.

[15] G. D. Michell and R. K. Gupta, "Hardware/software co-design," *Proc. IEEE*, vol. 85, no. 3, pp. 349–365, Mar. 1997, doi: 10.1109/5.558708.

[16] J. Teich, "Hardware/Software Codesign: The Past, the Present, and Predicting the Future," *Proc. IEEE*, vol. 100, no. Special Centennial Issue, pp. 1411–1430, May 2012, doi: 10.1109/JPROC.2011.2182009.

[17] P. Schaumont, *A Practical Introduction to Hardware/Software Codesign*. Springer US, 2010.

[18] S. KHAMLICH, "Design and production of a voice recognition system based on MFCC, artificial neural networks and NIOS II embedded processors," Chouaib Doukkali University, El jadida (Morocco), 2013.

[19] Altera Corporation, "Quartus II Software and Device Support Release Notes Version 14.0." Altera Corporation, Jul. 2014, Accessed: Apr. 07, 2021. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/rn/archives/rn_qts_140_dev_support.pdf.

[20] E. F. Inc, "The Community for Open Innovation and Collaboration | The Eclipse Foundation." https://www.eclipse.org/ (accessed Apr. 07, 2021).

[21] "Using NicheStack TCP/IP Stack – Nios II Edition." https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/intellectual-property/embedded/nios-ii/exm-using-nichestack.html (accessed Apr. 07, 2021).

[22] "µC/OS Real-Time Operating System | Micrium." https://www.micrium.com (accessed Apr. 07, 2021).

[23] S. arrag, A. Hamdoun, A. Tragha, and S. eddine Khamlich, "Several AES Variants under VHDL language In FPGA," *ArXiv12104962 Cs*, Oct. 2012, Accessed: Mar. 18, 2021. [Online]. Available: http://arxiv.org/abs/1210.4962.

[24] S. Arrag, A. Hamdoun, A. Tragha, and S. Khamlich, "Replace AES key expansion algorithm by modified genetic algorithm,"

vol. 7, pp. 7161–7171, Jan. 2013, doi: 10.12988/ams.2013.38482.

[25] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.

[26] V. Fischer and M. Drutarovský, "Two Methods of Rijndael Implementation in Reconfigurable Hardware," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Berlin, Heidelberg, 2001, pp. 77–92, doi: 10.1007/3-540-44709-1_8.

[27] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proc. IEEE*, vol. 91, no. 1, pp. 112–126, Jan. 2003, doi: 10.1109/JPROC.2002.805821.