Implementation of Miniature Embedded Server With Nios II Processor

Deepali Shelke , Jayu Kalambe and Meghana Hasamnis

***Abstract*-In general server is a software program that runs on a computer and delivers web pages to a browser. The proposed server can process basic requests to serve HTML, JPEG and GIF files from the Altera read–only zip file system. It is an example of HTTP server using lightweight internet protocol (LWIP) on MicroC/OS-II platform, which uses a standard sockets interface. The LWIP has two application program interface (API) i.e. a call back interface and a standard sockets. In the board /host set up it requires an Ethernet cable connected to the development board's RJ–45 jack and joint test action group (JTAG) connection with the development board. If the host communication settings are changed from default JTAG UART to a conventional UART then a serial cable between board, DB-9 connector and the host is required. If dynamic host configuration protocol (DHCP) is available then the server will attempt to obtain an IP address from DHCP server. Otherwise, a static IP address which is defined in user.h file will be assigned after a 120 second time out. The DHCP time out can be adjusted in telnet.h file. The server is implemented on NIOS II embedded processor. The Altera FPGA is used to create a custom system-on-a-chip or, more accurately, a system on-a-programmable-chip (SOPC). To run the server it is required to program the file system, generated as filesys.flash in<system library project>/ Release folder using flash program utility of the NIOS-II IDE (integrated development environment)**

Keywords **: Nios II Processor, SOPC, MicroC/OS-II**

I. INTRODUCTION

A web server is a software program that runs on a computer and delivers web pages to a browser. The proposed web server in the project is an example of HTTP server using light weight internet protocol (LWIP) on MicroC/OS-II platform. The web server can process basic requests to serve HTML, JPEG and GIF files from the Altera read –only zip file system. The LWIP has two application program interface (API) i.e. a call back interface and a standard sockets. The proposed web server is implemented on NIOS II embedded processor. The Nios II processor is a general-purpose RISC processor with embedded peripheral architecture.

The Nios II processor system is equivalent to a micro controller or "computer on a chip" that includes in the Quartus II software and is available to all Altera customers. It automates the task of integrating hardware components into a larger system. It can specify the system components in a graphical user interface (GUI), and generates the interconnect logic automatically.

Ms. Deepali Shelke, Lecturer , S.R.K.N.E.C, Nagpur
Email:deepshelke@yahoo.co.in Mobile No.: 09881713176/70
Ms. Jayu Kalambe Lecturer, S.R.K.N.E.C, Nagpur
Email:jayu_kalambe@rediffmail.com Mobile No.: 09822462988
Ms. Meghana Hasamnis Lecturer, S.R.K.N.E.C, Nagpur
Email:meghanahasamnis@rediffmail.com Mobile No. : 09373284084

The purpose of the SOPC Builder GUI is to allow one to easily define the structure of a hardware system and then generate the system. The GUI is designed for the tasks of adding components to a system, configuring the components and specifying how they connect together.

To run the server it is required to program the file system, generated as filesys.flash in<system library project>. Release folder using flash program utility of the NIOS II IDE integrated development environment. The Nios II integrated development environment (IDE) is the software development GUI for the Nios II processor. All software development tasks can be accomplished within the Nios II IDE, including editing, building, and debugging programs. The Nios II IDE is the window through which all other tools can be launched. The hardware abstraction layer (HAL) system library provides a hosted C runtime environment based on the ANSI C standard libraries. The HAL provides generic I/O devices, allowing one to write programs that access hardware using the C standard library routines, such as prints(). The HAL minimizes or eliminates the need to access hardware registers directly to control and communicate with peripherals. [1]

By using Quartus II software customized system design is created that interfaces with the component provided on NIOS II development board. The whole system is realized by IP cores provided by Altera. The database for web site is created in flash memory through flash programmer. The program is written in NIOS IDE using C/C++ language by using NIOS drivers. [2]

II. SYSTEM HARDWARE

The proposed web server system required following hardware: The Nios II Development Board, Cyclone II Edition. The following peripherals are used in system. Ethernet MAC (named "LAN91C111" in SOPC Builder) STDOUT device (UART or JTAG UART) LCD Display (named "lcd_display" in SOPC Builder) Board/Host Requirements: The server requires an Ethernet cable connected to the development board's RJ-45 jack and a JTAG connection with the development board. If the host communication settings are changed from JTAG UART (default) to use a conventional UART then a serial cable between board DB-9 connector and the host is required.

A. Connecting to the Board via Ethernet:

The Nios II development board is factory-programmed with a reference design that implements a web server. The Nios II development kit includes an Ethernet (RJ45) cable and a male/female RJ45 crossover adapter.
There are two methods for connecting to the board via Ethernet, which are:
1).LAN Connection — to use Nios development board on a LAN (for example, connecting to an Ethernet hub) do the following:
a) Connect one end of the RJ45 cable to the Ethernet connector on the development board (RJ1).
b) Connect the other end to LAN connection (hub, router, wall plug, etc.).
2). Point-to-Point Connection — To use Nios development board connected directly to a host computer point-to-point (not on a LAN), do the following:
a) Connect one end of RJ45 cable to the female socket in the crossover adapter and insert the male end of the crossover adapter into RJ1 on the Nios development board as shown in Figure 4.9
b) Connect the other end of the RJ45 connector directly to the network (Ethernet) port on host computer.

B. Obtaining an IP Address:

In order to function on a network (either LAN or point-to-point), the board must have an IP address. This section describes the methods to assign an IP address to the board.
If board is connected to a LAN, then it will either obtain a dynamic IP address using DHCP, or a static IP address stored in flash memory. If it is not known whether or not LAN supports DHCP, it is easiest to try DHCP first. Upon reset, the web server attempts to acquire an IP address via the DHCP protocol. The board continues to attempt DHCP self-configuration for two minutes. One can determine if DHCP has succeeded, or if it is still in progress, by reading status messages on the LCD screen. If the LAN does not support DHCP then DHCP configuration ultimately fails, and the web server defaults to a static IP address.
If DHCP succeeds, the board displays a success message and the IP address on the LCD screen. The web server is now ready to display web pages. If the DHCP process fails, the board uses a static IP address stored in flash memory.

C .Browsing to the Board:

Once the board has a valid IP address (obtained from either DHCP self-configuration or from flash memory), it can be accessed via a web browser (e.g., Microsoft Internet Explorer). To browse to this site, open a web browser and type the IP address of the board (four numbers separated by decimal-points) as a URL directly into the browser's Address input field. One can determine the board's IP address by reading the messages displayed on the LCD screen.

III. SYSTEM CREATION

The SOPC Builder is a powerful system development tool for creating systems based on processors, peripherals, and memories. It enables one to define and generate a complete system-on-a-programmable-chip in much less time than using traditional, manual integration methods. It is included in the Quartus II software and is available to all Altera customers. It automates the task of integrating hardware components into a larger system. It can specify the system components in a graphical user interface (GUI), and generates the interconnect logic automatically. Fig 1 shows Example of a System Module Generated by SOPC Builder

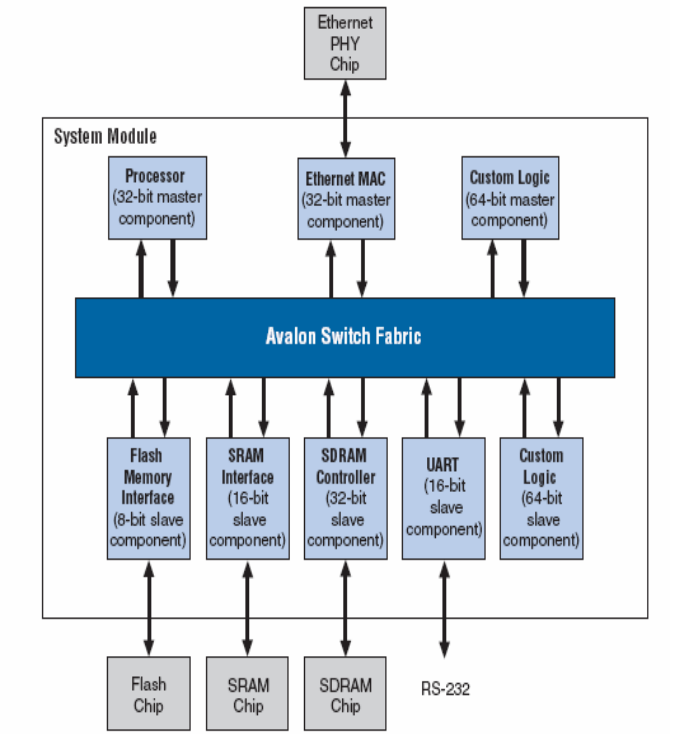

Fig.1 Example of a System Module Generated by SOPC Builder

A. SOPC Builder Components:

SOPC Builder components are the building blocks of the system module. It components use the Avalon interface for the physical connection of components, it is use to connect any logical device (either on-chip or off-chip) that has an Avalon interface. The Avalon interface uses an address-mapped read/write protocol that allows master components to read and/or write any slave component.

Altera provide ready-to-use SOPC Builder components, such as: Microprocessors, such as the Nios II processor, Micro controller peripherals, Timers, Serial communication interfaces, such as a UART and a serial Peripheral interface (SPI) ,General purpose I/O Digital signal processing (DSP) functions Communications peripherals ,Interfaces to off-chip devices ,Memory controllers buses and bridges ,application-specific standard products (ASSP) ,application-specific integrated circuits (ASIC) Processors.

B. User-Defined Components:

SOPC Builder provides an easy method to develop and connect components. With the Avalon interface, user-defined logic need only adhere to a simple interface based on address, data, read-enable, and write-enable signals.[5]

The following design flow is use to integrate custom logic into an SOPC Builder system:

1. Define the interface to the user-defined component.

2. If the component logic resides on-chip, write HDL files describing the component in either Verilog HDL or VHDL.

3. Use the SOPC Builder component editor wizard to specify the interface and optionally package HDL files into an SOPC Builder component.

4. Instantiate all component in the same manner as other SOPC Builder Ready components.

C. Location of the Component Hardware:

There are two types of components, based on where the associated component logic resides:

Components that include their associated logic inside the system module & components that interface to logic outside the system module

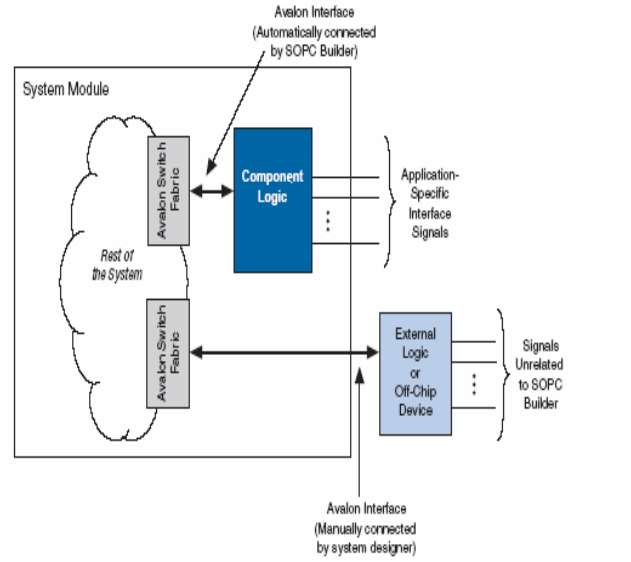

Fig .2 Component Logic Inside and Outside the System Module

D. Components that include logic inside the System Module:

In this case, the component files provide a full description of the component hardware. During system generation, SOPC Builder instantiates the component logic inside the system module and automatically wires the component to the rest of the system. Internal to the system module, the component connects to the rest of the system through its Avalon interface. The component can also have non-Avalon signals that SOPC Builder exposes on the top-level system module.

E. Components that Interface to Logic outside the System Module:

In this case, the component files describe only the interface to logic external to the system module. During system generation, SOPC Builder does not instantiate any logic for this component. Instead, SOPC Builder exposes an Avalon interface for this component on the top-level system module. One must manually wire the interface to external logic, such as a separate HDL module or an off-chip device .As shown in Fig.2, in this case there is no physical embodiment of the SOPC Builder component. Components that interface to external logic describe only the shape of the Avalon interface; they do not include logic inside the system module.

F. Steps for creating system standard design in SOPC Builder:

1. Adding component in the standard example design of Nios II processor

2. Nios II CPU Configured in SOPC Builder:

3. System Generation in SOPC Builder

4. Integrate SOPC Builder O/P in Quartus II

IV. IMPLEMENTATION SETUP

A. Flow of system design:

- Start the server.

- Initialize TCP/IP Stack.

- Run up the server.

- If DHCP is available the server will take dynamic address other wise it will take static IP address stored in software file.

- After getting the suitable IP address the server will start serving pages.

- The database is stored in flash memory which is present on NIOS II development board.

B. Software source files use:

- Web_server.c: Contain main ( ) and LWIP callback to install tasks once LWIP has been properly initialized.

- http.c: Implementation of an HTTP server including all necessary sockets calls to handle a multiple connections and parsing basic HTTP commands to handle GET and POST requests.

- http.h: Header information defining HTTP server implementation and common HTTP server strings and constant.

- user.h: Definition for the entire example application.

- Network_utilities.c: Contains MAC address, IP address, and DHCP routines to manage addressing.

Application software:

The following steps are performing to start with the Nios II IDE from the Quartus II software.[4]

1. Click the System Generation tab of the Altera SOPC Builder window; then click the Run Nios II IDE button.

2. From the opening window, choose File New C/C++ Application.

3. Complete or check the following:

Select Count Binary from the Select Project Template field. The Name field automatically becomes count_binary_0. Click Finish. At this point, the project is added to workspace and ready to be built and run.

Fig 3. Nios II IDE Window after Creating the Project

4. To build the project, right-click on the project in the navigator pane and select build Project. The build process begins. The Nios II IDE determines all of the project's dependencies, warns of any potential problems, catches errors, and compiles the source code into a <project name>.elf file. When the build process successfully completes, one can run the project. [2]

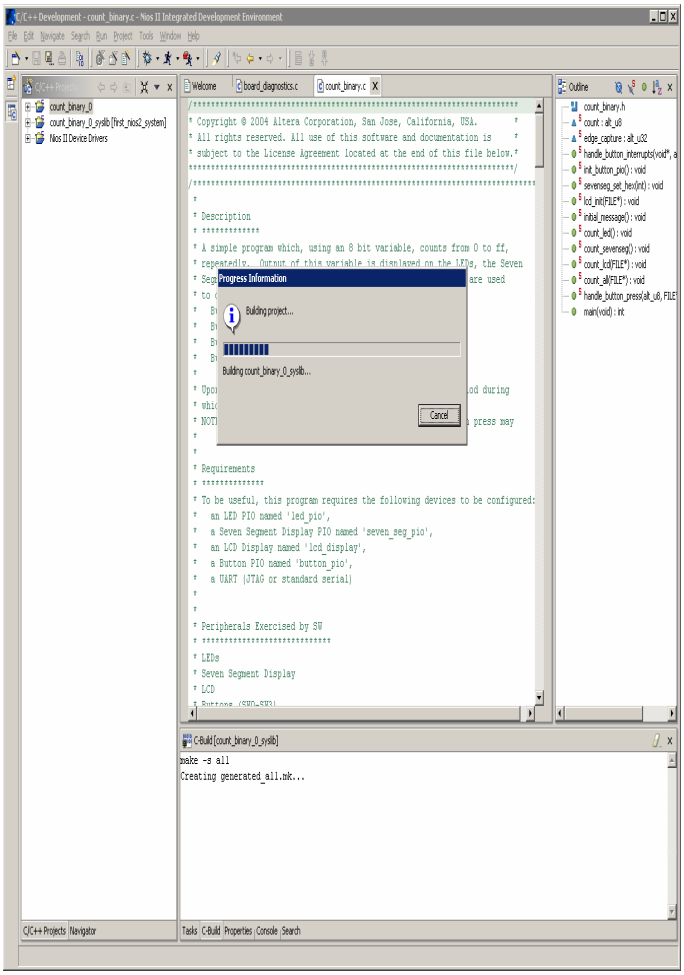

Fig.4 Nios II IDE Build Process

5. After completion of build process, the project is run as Nios II hardware.

6. After successful completion of build and run, the result is displayed on LCD display, which is connected to the Nios II development

V. CONCLUSIONS

The proposed web server perform following functions as compare to conventional Web server

- HTTP 1.0/1.1 compliant.
- Multiple concurrent requests.
- Small memory footprint (7KB-11KB ROM)
- GET, POST support
- Transfer chunked encoding support
- Full access and exposure to headers
- Form item decoding
- Compliant with IETF RFC 2616
- Easily portable

VI. REFERENCES

[1] Embedded System Security Designing Secure Systems with Windows CE Lawrence Ricci (eMVP) Larry McGinnes (CPL) Applied Data Systems COACT.

[2] Designing *System on a Chip* Products using Systems Engineering Tools Graham R. Hellestrand, CEO and President, VaST Systems Technology Corporation, 2700 Augustine Drive, Santa Clara, CA 95054 USA

[3] Nios documentation http:// www.altera.com /Literature/manual .pdf

[4] Nios Embedded Processor Software Development Reference Manual (Doc. version 3.2).

[5] Nios Hardware Development Tutorials (Doc. Version 1.2)

[6] Sriram V Iyer, Pankaj Gupta, "Embedded Realtime System Programming", New Delhi: McGraw-Hill, 2006.

VII. BIOGRAPHIES

MsDeepali Shelke, born in Yeotmal in India on November 15, 1979. She graduated from the Government College of Engineering, Amravati (Maharashtra State) and studied her post graduation in Electronics Engineering from YCCE, Nagpur (Maharashtra State).

Her employment experience includes five years of teaching at graduate level. Her special fields of interest include VLSI and embedded system.

Ms. Jayu Kalambe, born in Katol in India on November 17, 1977. She graduated from the Priyadarshini College of Engineering, Nagpur (Maharashtra State) and studied her post graduation in Electronics Engineering from VNIT, Nagpur (Maharashtra State).

Her employment experience includes Seven years of teaching at graduate level. She is having to her credit many International and National Conference papers. Her special fields of interest include Neural Network and embedded system.

Ms. Meghana Hasamnis, born in Kolhapur in India on May 09, 1975. She graduated from Yeshwantrao Chavan College of Engineering, Nagpur (Maharashtra State) and studied her post graduation in Electronics Engineering from VNIT, Nagpur (Maharashtra State).

Her employment experience includes Seven years of teaching at graduate level. She is having to her credit many International and National Conference papers and received best paper award in one of the conference. Her special fields of interest include Embedded System and Fuzzy Logic.