Home Automation Using Zybo FPGA

Lavanya T
Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, Tamil Nadu, India
lavanya.takur@gmail.com

Rajalakshmi K
Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, Tamil Nadu, India
krl.ece@psgtech.ac.in

Abstract- **In this modern world, a requirement of manpower everywhere is quite hard and replacement by smart technology is needed. People are more familiar with the usage of smartphones and tablets and work can be done easily. Also, the microcontroller cannot interface with high-power devices directly and only performs a limited number of executions simultaneously. The main aim of this work is to develop a smart home automation system using FPGA (ZYBO Z-7010) that is IoT-based and wirelessly controls the appliances using a mobile phone. The design and implementation of home automation systems have been described as an embedded system and while implementation, the new IPs and customized IPs are designed with Verilog coding. It is implemented in hardware using FPGA (ZYBO Board) with the help of the Xilinx Vivado tool and communicated with IoT using the Petalinux platform. To communicate with IoT, a web server has been created using bottleneck API and python. The IO devices connected to the ZYBO hardware can be controlled from the web page and the status of the IO devices connected to the board can be received on a web page as well. Sensors such as MQ-6 to detect gas leakage, LM35 to detect temperature, IR sensor, and rain sensor are integrated with the FPGA board and any desired indication in them will be displayed on the web page and the devices can be controlled accordingly. Also, analog data from the sensor is read through one of the ports of the XADC module in ZYBO and is displayed on the webserver. Therefore, a prototype of a smart home automation system is implemented as an embedded system using SoC (System on chip).**

Index Terms: FPGA, home automation, System on Chip.

DOI Number: 10.48047/nq.2022.20.19.NQ99081 ***NeuroQuantology 2022; 20(19): 874-882***

1. Introduction

Home automation is the automatic control of electronic devices in our homes and also sometimes automatically respond to certain conditions or scenarios [1]. Home automation is the building automation of a house, called a smart home. Home automation systems monitor and/or control home attributes such as lights, air conditioning, entertainment systems, and appliances. Home devices are becoming an important part of the Internet of Things ("IoT") by connecting to the Internet [2]. In home automation, the control system user interface uses a wall terminal, tablet or desktop computer, mobile phone application, or web interface, and can also be accessed remotely via the Internet. While there are many competing vendors, the drive for open-source systems is growing.

Home automation has huge potential to share data with family members or confidants for personal safety and could lead to future energy-saving measures that have a positive impact on the environment. In this work, a smart home automation system using FPGA is done as an embedded system using a system on chip (SoC design). However, currently available IoT platforms for home automation are still limited, and the use of Field Programmable Gate Arrays (FPGAs) is still relatively uncommon in these platforms [3]. In many areas, FPGAs have the potential to achieve higher performance than more commonly used embedded computers such as the Raspberry Pi. FPGA chips are ideal for home automation because of their long-life cycle and more than 100 general-purpose inputs and outputs (GPIOs). This paper designs and develops an embedded architecture to exploit hardware parallelism in FPGAs as a local analysis engine in IoT applications for home automation with VHDL and investigates the potential of FPGAs in IoT applications [7].

There were many works related to automation and they are described as follows. In this paper [1], the design is described using VHDL and implemented using hardware (FPGA). The system is based on SMS technology and uses a GSM network to establish the communication between mobile and controller but in this method, only a limited number of sensors can be interfaced, and changing sim cards is tedious. This paper [2], aims at controlling home appliances via Smartphone using Wi-Fi as communication protocol and raspberry pi as a server system but in this method, there is no security for the user, and there is a use of a processor instead of a controller. In this paper [3], the proposed methodology is that the system is controlled using FPGA which receives commands from IoT modules but this methodology requires an app and is limited to android mobile phones.

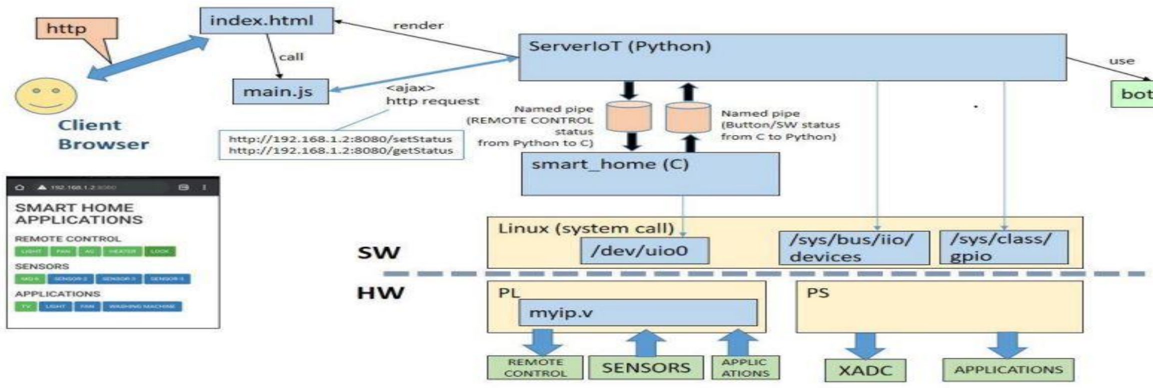


Fig .1 Proposed Home Automation block diagram

In this paper [4], the methodology is designed and implemented using an ARM 7 processor and simulated using Keil software but there is no security for the user, and there is a use of a processor instead of a controller. In this paper [5], various sensors like LM35, IR sensors, LDR module, Node MCU ESP8266, and Arduino UNO have been used however the use of Arduino limits the performance as compared to FPGA.

This paper describes the proposed methodology, results and discussions, conclusion, and future work in further sections.

2. Proposed methodology and implementation

Our work presents the design and implementation of home automation systems. The system can be described as an embedded system and is coded in VHDL, and is implemented using FPGA (Field Programmable Gate Array) ZYBO board Z-7010 [10]. In the IoT (Internet of Things) system, wireless technology is used to improve the standard of living. Initially, to communicate with IoT, a web server is created, and then a mobile phone connects wirelessly to the FPGA controller (Zybo board) using the server.

2.1 Development environment

We use XILINX VIVADO, XILINX SDK, Visual Studio code, Zybo board, Ubuntu Virtual Box, and PetaLinux for development. In addition to HDL design synthesis and analysis, the Vivado Design Suite adds capabilities for system-on-chip (SoC) development and advanced synthesis to Xilinx's ISE. Vivado Advanced Synthesis Compiler allows users to inject C, C++, or SystemC code directly into Xilinx devices without having to create RTL manually. Vivado HLS has been thoroughly reviewed to enhance developer productivity, and it supports C++ class, template, function, and operator overloading. Vivado 2014.1 introduced support for automatically converting OpenCL kernels to Xilinx devices. Various CPU, GPU, and FPGA platforms support OpenCL kernels. Additionally, Xilinx's Software Development Kit (XILINX SDK) lets you build embedded applications for any Xilinx microprocessor: Zynq®

UltraScale+ MPSoC, Zynq-7000 SoC, and the industry-leading MicroBlaze™ soft microprocessor. As the first IDE to allow the simultaneous design of heterogeneous and homogeneous multi-processors, debugging, and also performing performance analysis, Xilinx SDK sets the standard.

2.2 Proposed work

Our proposed work smart home automation using FPGA is shown in the Fig.1. It describes an overview of how the hardware communicates with IoT using a web server.

- Server IoT: it uses Python script and it is developed on a Windows PC for debugging MSYS and ZYBO real machines. It provides Web API and views as a server. The output LED control command to a named pipe as instructed by WebAPI (setStatus). GPIO is controlled by the user. After receiving WebAPI (getStatus), get the status of the button and switch from the named pipe. Subsequently, convert it to JSON format and return it to the caller (main.js).
- myip_controller: it is a C ++ application and developed / debugged with Xilinx SDK on Windows PC. It also controls the user-created IP (/dev/uio0, /dev/uio1).
- Linux image (kernel / RootFS): it is created with PetaLinux on Ubuntu, with installed Python and C ++ libraries. This includes Server IoT, myip_controller, and also, start these as startup daemons. System-user.dtsi- it makes your IP recognized as a UIO device by setting the device tree ( ).
- Hdf: Hardware information (PS settings and PL bitstream) which is developed in Vivado on Windows PC.

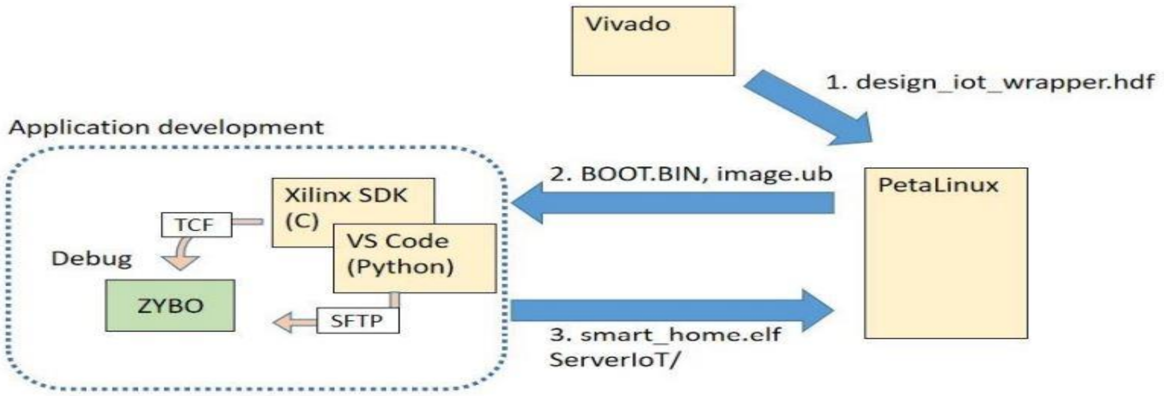


Fig.2 IoT integration with ZYBO

The block diagram for the integration of IoT with the ZYBO Board using Petalinux is shown in the Fig.2 On the ZYBO Z7-20 board, there are 4 LEDs, 4 buttons, 4 switches, 2 color LEDs connected to the PL, and 1 LED, 2 buttons connected to the PS. Create an IP named myip for controlling LEDs, buttons, and switches on the PL side. This is almost the same as AXI GPIO, but my IP was made for practice. It is a user-created IP myip that controls as User space IO (UIO). Each /dev/uio0, /dev/uio1 will be set in the device tree. Create a C application (myip_controller) to control IP. It has a named pipe as input and output. The input pipe accepts LED control commands (ON / OFF and color LED duty ratio if wanted). The output pipe outputs the status of buttons and switches. There is a Python script (Server IoT) at the top layer. Server IoT Acts as a server using a bottle. As a Web API, setStatus and getStatus are provided in POST (JSON) format. In fact, using /dev/uio0, , /dev/uio1 it is possible to omit the C application and control it only with Python. For C and python integration, Boost.Python or ctypes can be used. In this case, the C side is the shared library (.so). Named pipes can be used to interact with strings. This makes the Python side and the C application binary loosely coupled, which makes testing very easy. When developing on the C side, the input and output are pipes, so the operation can be checked with echo from the console. The same is true on the Python side. Furthermore, since the input and output are pipes, hardware control is eliminated and development/confirmation can be performed on the host PC. (However, since it uses a pipe, it cannot be used at the Windows command prompt. Also, PS GPIO cannot be confirmed.)

- The hardware (hdf) and pre-built binaries are already included in the PetaLinux project. So, only the PetaLinux project is needed to be built.
- A new project is created in Vivado. The Hardware is made and the hdf wrapper file is exported along with bitstream.
- The block design has been made by selecting the IPs (Intellectual properties) either new IPs or customized IPs and it is displayed in Fig.3. The ZYNQ7 processing system is the main processing unit of the embedded system designed using Xilinx Vivado [9]. For IoT communication, an IP has been added and named TRIAL_IOT here.
- Top-level designs can be based on schematics, and lower-level designs can be built using any of the following source types: HDL files, state diagrams, CORE GeneratorTM cores, Architecture Wizard IP, or schematic files. To instantiate a lower-level module in the top-level design, a schematic symbol must be created from the lower-level module, and the schematic symbol instantiated, resulting in the schematic design.
- VIVADO implementation encompasses all the steps required to complete the placement and routing of a netlist onto the FPGA device while fulfilling the logical, physical, and timing requirements of a design.
- The IP named 'myip' has been created to control LEDs, switches, and buttons of the ZYBO board. The IP can be customized as per the user.

2.3 Creating Base LINUX IMAGE (PETALINUX ON UBUNTU):

Embedded Linux solutions are built and deployed on Xilinx processor platforms using Petalinux [13,14]. It is an Embedded Linux Systems Development Kit that targets Xilinx SoC designs. PetaLinux tools will automatically generate a custom, Linux Board Support Package including device drivers for Xilinx embedded processing IP cores, kernel, and boot loader configurations. A new PetaLinux project was created in Ubuntu. Then, based on this hardware information, the project must be configured. Before proceeding to the build, the device tree must be changed. myip is a UIO device. Next, the RootFS config is changed and the necessary tools and

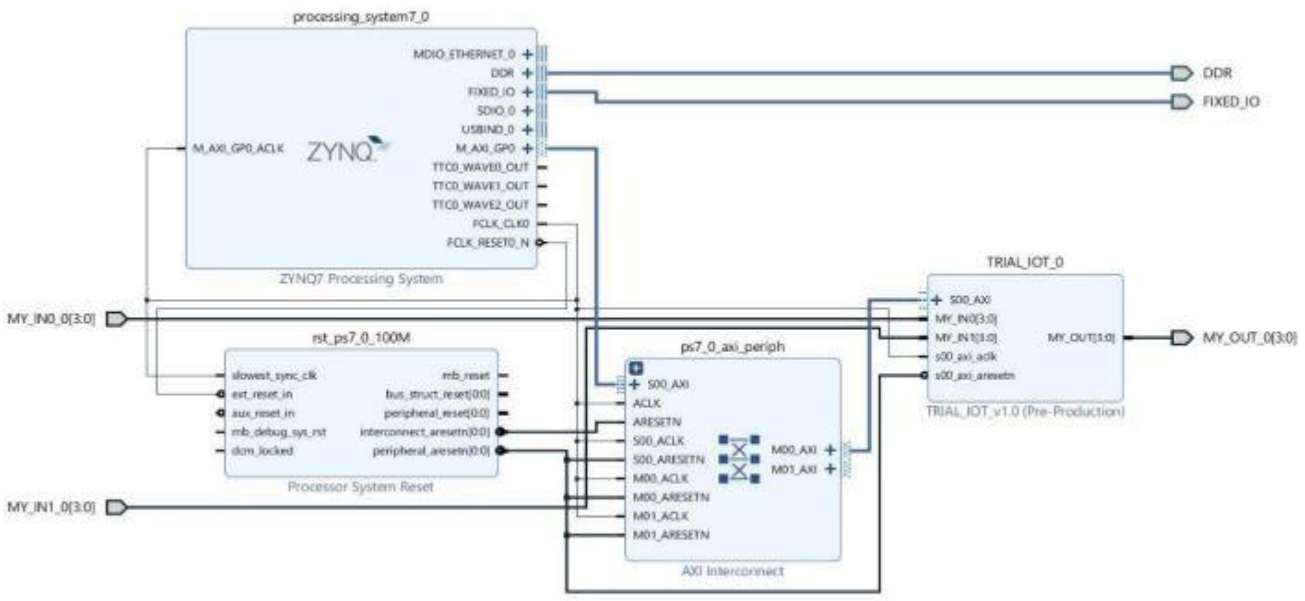


Fig.3 Block design of Embedded System

libraries are installed. Python, curl and libstdc ++ on menuconfig also shall be installed. After setting, to build the project the following command flow are used.

petalinux-config -c rootfs

# Filesystem Packages → devel → python → python

# Filesystem Packages → console → network → curl

# Filesystem Packages → misc → gcc-runtime-> libstdc ++

petalinux-build

petalinux-package --boot --force --fsbl images / linux / zynq_fsbl.elf --fpga images / linux.

- The completed image (images/linux/BOOT.bin and images/linux/image.ub) is inserted into the SD card (FAT32) and ZYBO is started. Having made sure python is installed, the IP address can be checked with ifconfig.
- The BOOT.bin file and the image.ub file has been obtained from the Petalinux and is copied to an SD card. The SD card is inserted into the PC in the Linux platform after copying the necessary files into it. The Operating system has been booted in the hardware ZYBO Z-7010 board.
- In this work for serial monitoring, a software called TERATERM is an open-source, free, software-implemented, terminal emulator (communications) program. It emulates different types of computer terminals, from DEC VT100 to DEC VT382. It also supports telnet, SSH 1 & 2, and serial port connections. The Petalinux project is opened with a login ID and password. After entering the project, the IP address of the hardware is obtained by giving the command 'if config'.
- In Linux, the pipe command can be used to send the output of one command to another. It can redirect the output, input, and error of one process to another for further processing. Here the pipe_c2p (C to python) is used to transfer the data when a switch is turned on or a button is pressed from the hardware (ZYBO Board) to the webserver (python). Similarly, the pipe_p2c (python to C) is used to transfer data when the LED icons are pressed on the web page which makes the particular LED glow. The interface by pipe should be the following character string.
  - pipe_p2c (Python → LED control command to C app) "ON, OFF, ON, OFF, 100,100,100,100,100,100"
  - pipe_c2p (C app → Notification of button and switch status to Python) "{" btn0 ":" OFF "," btn1 ":" OFF "," btn2 ":" OFF "," btn3 ":" OFF "," sw0 ":" OFF "," sw1":" OFF ", "sw2": "OFF", "sw3": "OFF"}"
- finally, Target Communication Framework must be set up which is nothing but a protocol for driving embedded systems by using system debugger and program is debugged by specifying the debug type as Linux Application Debug.

2.4 Creation of web server

Microsoft Visual Studio is an IDE provided by Microsoft and its various applications such as web apps, web services, computer programs, and mobiles app, are developed using visual studio. Visual Studio-25 works on different platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store, and Microsoft Silver light provided by Microsoft. Visual studio builds both managed code and native code. In development so far, the C application has debugged the elf binary with TCF and the Python script has been uploaded with SFTP. For Python scripts (servers), you need to install a directory that contains multiple files, such as html, instead of just one file. The multiple files include static, views, pipes created, python main files which are necessary for the creation of the web server, and these necessary files are uploaded to a web server by providing the IP address in the code in JSON (JavaScript Object Notion) format.

- Serial Port Monitor: in this, open-source software Tera Term window has been used for serial port monitoring. It works on different types of computer terminals, from DEC VT100 to DEC VT382 are reproduced by Teraterm, and also supports SSH 1 & 2, telnet, and serial port connections. The created server gets started after opening the main python file.
- Web Page (User View): After giving the correct IP address in the URL, the web page will be opened and the processes can be verified and the FPGA (ZYBO Board) can be controlled from the web

server and vice versa. After the integration of sensors, the web page can be programmed as required by the user.

2.5 Sensors Integrated with ZYBO

i. Gas sensor – MQ-6: The gas sensor (MQ135) is integrated with the system and when gas is detected, the indication can be shown in the web page created or can be indicated with a glowing LED.

ii. Temperature sensor – LM35: LM35 sensor can detect the temperature and that value can be displayed on the web page. Since the data is an analog value, the XADC comes into the picture.

iii. IR sensor: IR sensor is used to detect an obstacle.

iv. Rain sensor: A rain sensor is used to detect water droplets.

The digital data of these sensors are given to the ports of the ZYBO board and the indications are shown on the web page. The analog data of the temperature sensor LM35 can be read using the XADC block of the ZYBO board through one of its ports with a specific IP module, XADC Wizard [15].

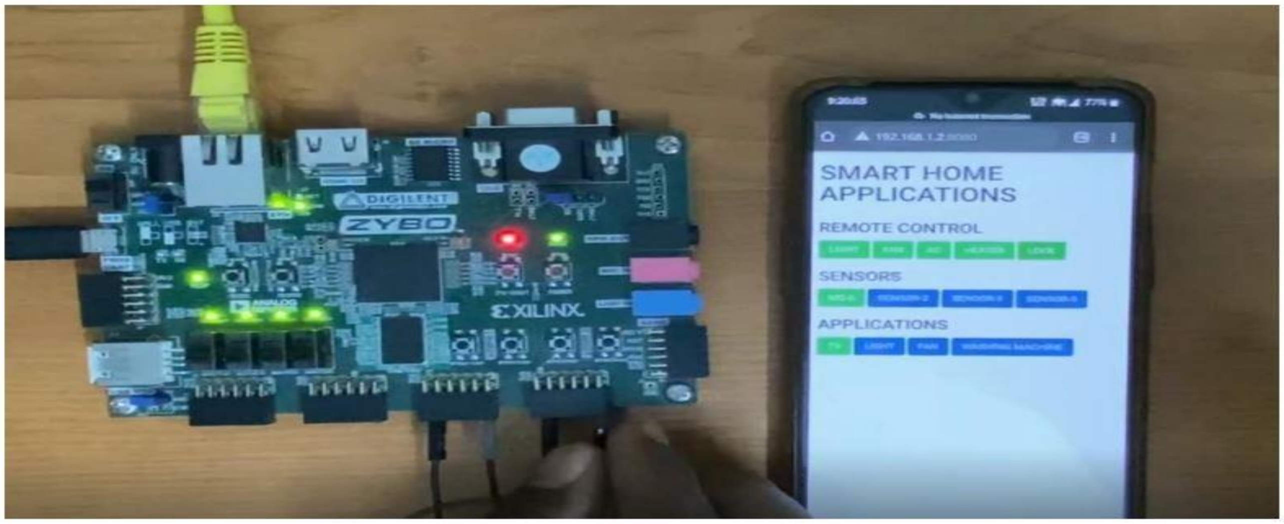


Fig.4 ZYBO control using webpage

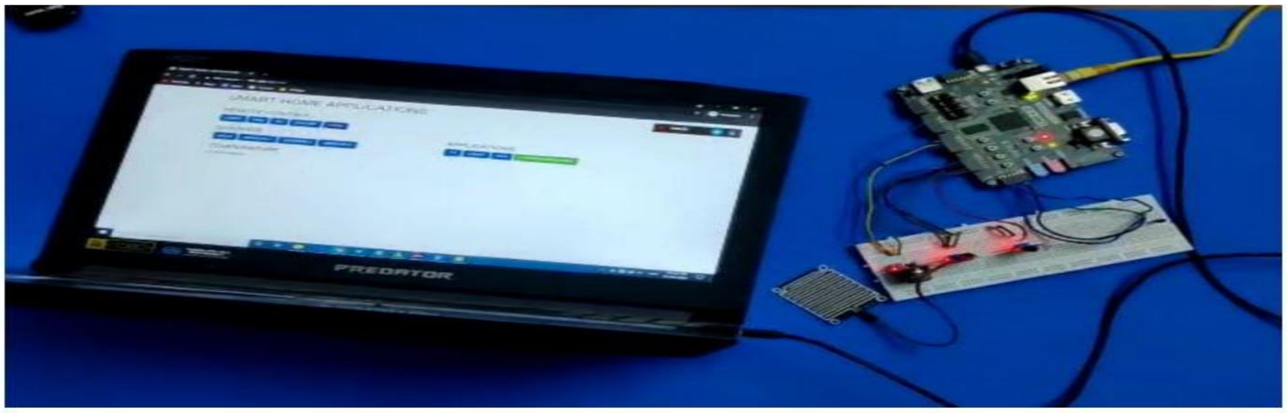


Fig.5 Experimental Setup of FPGA

3. Results and Discussion

In this section, the work proposed is implemented, and how the Zybo board communicates with a web server and vice versa are discussed below.

i. CONTROLLING ZYBO FROM WEB SERVER: As shown in the Fig.4, the ZYBO Board is communicating with IoT using a web server. The Remote-control icons on the web page when pressed, the LEDs on the board glow accordingly.

ii. OVERALL SET UP: The overall setup of the work, Smart home automation has been shown in the Fig.5. Four sensors namely gas sensor MQ-6, temperature sensor LM35, IR sensor, and rain sensor have been integrated with the ZYBO Board and the web page has been opened on the PC by specifying the IP address in the URL.

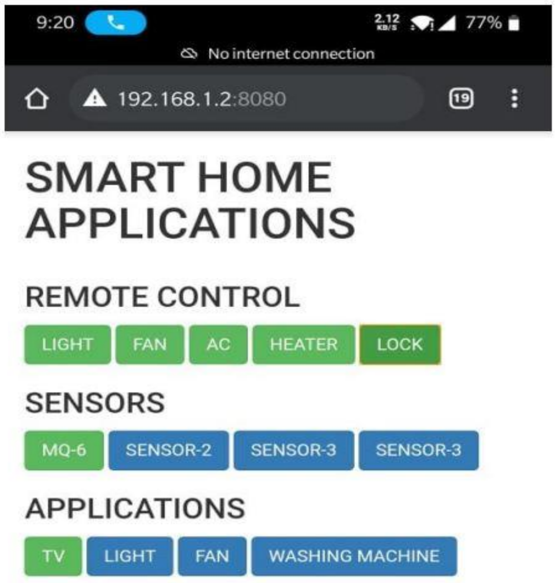
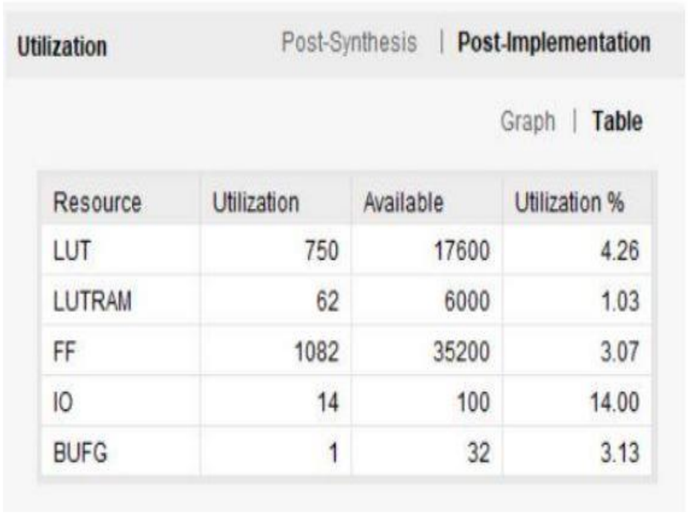


Fig.6 User Webpage for appliance control

iii. SMART HOME APPLICATIONS WEB PAGE: The web page that has been designed to implement the project is shown in the Fig.6. Using the IP address in the URL, the desired web page is opened and the operations can be performed. With the icons in the remote control, the light, fan, AC, heater, and the lock can be turned ON. The icons in the sensors turn green when there is an interrupt. The icons in the application display the status of the devices and glow green when switched ON.

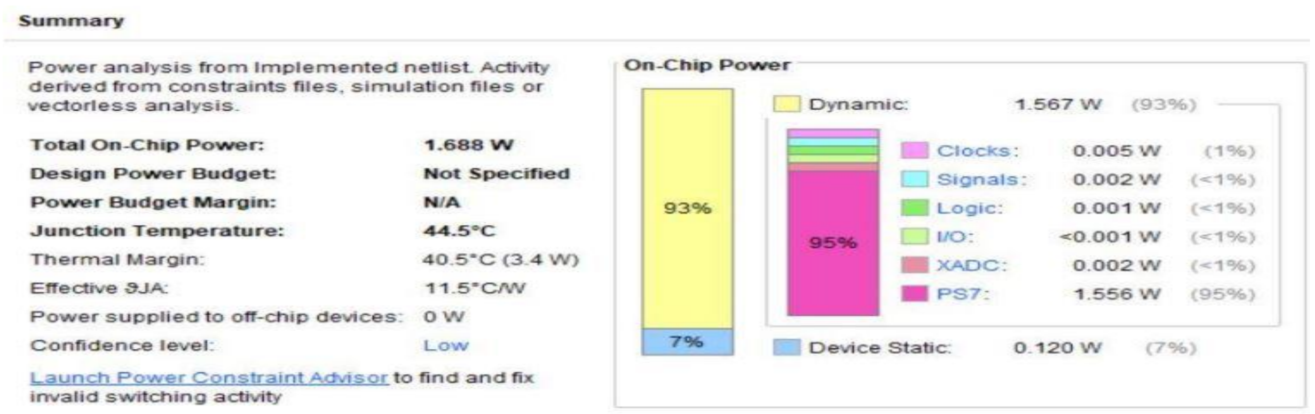
iv. AREA ANALYSIS: The area analysis of the project is depicted in the Fig.7. The utilization of area is less than 25% and the individual utilization of the resources is clearly shown in the Fig.7.

v. POWER ANALYSIS: The power utilization and analysis are depicted in the Fig.8. The total on-chip power consumed is 1.688 W among which 1.567 W is dynamic power and 0.120 W is static power.


Utilization
Post-Synthesis
Post-Implementation
Graph
Table

Resource	Utilization	Available	Utilization %
LUT	750	17600	4.26
LUTRAM	62	6000	1.03
FF	1082	35200	3.07
IO	14	100	14.00
BUFG	1	32	3.13

Fig.7 Area Utilization



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.688 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**44.5°C**
Thermal Margin:	40.5°C (3.4 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨 Dynamic:	1.567 W	(93%)
Clocks:	0.005 W	(1%)
Signals:	0.002 W	(<1%)
Logic:	0.001 W	(<1%)
I/O:	<0.001 W	(<1%)
XADC:	0.002 W	(<1%)
PS7:	1.556 W	(95%)
🟦 Device Static:	0.120 W	(7%)

Fig.8 Power Utilization

4. Conclusion

A low-cost control system using FPGA for home automation is developed as an embedded system using SoC. Using the system, you can monitor real-time as well as control and sense several different electrical devices at the same time in home automation. The hardware implementation of the system was performed using the VHDL language and Zybo board 7010 FPGA. Mobile phones receive the information by way of a web page with an IP address and a web server is integrated with IoT. A simulation and verification of the whole system's operation was performed. In addition to being able to control numerous home appliances remotely, the system is available and easy to use.

5. Future work

The objective of this work is to implement various sensors with the FPGA board. All the used sensors either give a digital or analog output. Sensors that transmit data through serial communication cannot be integrated with this current design. Hence, the future work is to make a design that implements serial communication possible either by I2C or UART protocols.

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. S. Chinchansure and C. V. Kulkarni, "Home automation system based on FPGA and GSM," 2014 International Conference on Computer Communication and Informatics, 2014, pp. 1-5, doi: 10.1109/ICCCI.2014.6921803.

[2] D. Pavithra and R. Balakrishnan, "IoT based monitoring and control system for home automation," *2015 Global Conference on Communication Technologies (GCCT)*, 2015, pp. 169-173, doi: 10.1109/GCCT.2015.7342646.

[3] Azeem Mohammad Abdul, "IoT BASED HOME AUTOMATION USING FPGA", Journal of engineering and applied sciences,1931-1937,2016.

[4] K. Keshamoni and S. Hemanth, "Smart Gas Level Monitoring, Booking & Gas Leakage Detector over IoT," *2017 IEEE 7th International Advance Computing Conference (IACC)*, 2017, pp. 330-332, doi: 10.1109/IACC.2017.0078.

[5] H. Singh, V. Pallagani, V. Khandelwal and U. Venkanna, "IoT based smart home automation system using sensor node," 2018 4th International Conference on Recent Advances in Information Technology (RAIT), 2018, pp. 1-5, doi: 10.1109/RAIT.2018.8389037.

[6] Intel. (2014). "Could Smart Homes Be as Commonplace as Smartphones by 2025?". [Online]. Available: http://download.intel.com/newsroom/kits/iot/pdfs/IntelSmartHomeSurveyBackgro under.pdf

[7] Amine Rghioui, AbdelmajidOumnad, "Challenges and Opportunities of the Internet of things in Healthcare" IJECE, vol 8, No 5, Oct 2018, pp2753-2761.

[8] Rajina R Mohamed et.al, "Provisioning of Street Lighting based on Ambience Intensity for Smart City" Vol 12, No 3, Dec 2018, pp 1401-1406.

[19] The Zync Book Tutorials, the vivado tutorials.

[10] Implementation of a home automation system through a central FPGA controller: https://ieeexplore.ieee.org/abstract/document/6196513

[11] https://forums.xilinx.com/t5/Embedded-Linux/zynq-7000-pl-dtsi-syntax-error/tdp/1114022

[12] https://forums.xilinx.com/t5/Embedded-Linux/Device-Tree-Error-unable-to-parseinput-tree/td-p/754945

[13] https://forums.xilinx.com/t5/Embedded-Linux/petalinux-config-Is-it-stuck-in-plnxstep/td-p/872414

[14] https://forums.xilinx.com/t5/Embedded-Linux/error-petalinux-build/td-p/1005383

[15] XADC Wizard v3.2 (LogiCORE IP Product Guide)