FPGA ASSISTED ZIGBEE COMMUNICATION FOR LOW-POWER HOME AUTOMATION DESIGN

by

Furkan Cayci

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Masters of Science in Electrical and Computer Engineering

Spring 2010

© 2010 Furkan Cayci
All Rights Reserved
FPGA ASSISTED ZIGBEE COMMUNICATION FOR LOW-POWER HOME AUTOMATION DESIGN

by

Furkan Cayci

Approved: 

Fouad E. Kiamilev, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: 

Kenneth E. Barner, Ph.D.
Chairman of the Department of Electrical and Computer Engineering

Approved: 

Michael J. Chajes, Ph.D
Dean of the College of Engineering

Approved: 

Debra Hess Norris, M.S.
Vice Provost for Graduate and Professional Education
ACKNOWLEDGMENTS

To my advisor, with my deepest gratitude, Fouad Kiamilev, who is a great and humble mentor. Without him, this work would not be possible, and I would not be where I am right now.

To my family, without whom my growth as a human being and the opportunities they have worked hard to give me would not have been possible. I dedicate this work to my parents and my sister Nihal who have always believed in me and supported me with their unfailing love even though they were far away.

To Ministry of Education, Turkiye for giving me the opportunity to study abroad and for funding my graduate education.

To all of the members of CVORG, especially Tyrell Fawcett, Stephen Janansky, Nick Waite, Rodney McGee, Joshua Marks, and Michael Roman. You have created a unique environment and did not hesitate to help me out with even the smallest problems that I have encountered. I profoundly believe that I could not have accomplished what I did at CVORG without you.

To Ron Reisor, for sharing his amazing research information.

Finally, I would like to express my gratitude to my Turkish friends especially Esra Ozturk and Abdulkadir Ozden for supporting me and being like a family during our time in Newark.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
</tbody>
</table>

Chapter

1 INTRODUCTION .......................... 1

1.1 Architecture ......................... 2

2 BACKGROUND INFORMATION: PARTS ........ 4

2.1 FPGA .................................. 4

2.1.1 Architecture ....................... 4

2.1.1.1 Logic Cells, Slices, CLB ...... 4
2.1.1.2 Block RAMs, Multipliers ....... 5
2.1.1.3 Hard and Soft Processors ..... 6

2.1.2 Typical Applications .......... 6

2.2 ZIGBEE ............................... 7

2.2.1 ZigBee Networks ................... 8
2.2.2 Typical Applications .......... 9
3 PRINTED CIRCUIT BOARD DESIGN

3.1 Peripheral Selection

3.1.1 FPGA

3.1.2 Platform FLASH

3.1.3 DDR-SDRAM chip

3.1.4 Ethernet Controller

3.1.5 ZigBee Module

3.1.6 Clock Generator

3.1.7 Power Regulators

3.2 FPGA Constraints

3.2.1 Reserving Important Pins

3.2.2 Preparing DDR-SDRAM Pins

3.2.3 Complete Placement

3.2.4 Design Check

3.3 Schematics

3.3.1 FPGA, Platform Flash, JTAG

3.3.2 DDR-SDRAM

3.3.3 Ethernet

3.3.4 ZigBee Module

3.3.5 SD/MMC Card

3.4 PCB Layout & Routing

3.4.1 DDR-SDRAM Design Rules

3.5 Fabrication

3.6 Debugging Process

3.6.1 DDR-SDRAM problem

3.6.2 Ethernet Mode Pins Problem
4 SOFTWARE WORKBENCH, APPLICATIONS, AND TESTING

4.1 MicroBlaze soft processor core ........................................... 27
4.2 Preparing the Boot loader .................................................. 28
4.3 Preparing the Kernel ............................................................ 29
4.4 Building Applications .......................................................... 30

  4.4.1 microcom ........................................................................ 31
  4.4.2 ZigBee Application ............................................................ 31

4.5 Testing .................................................................................... 33

5 CONCLUSIONS AND FUTURE WORK ..................................... 36

  5.1 MMU support ....................................................................... 36
  5.2 Wake-On-Lan Support ............................................................ 37
  5.3 Expanding the environment .................................................... 37

Appendix

A BOARD SCHEMATICS .............................................................. 38

BIBLIOGRAPHY ........................................................................... 48
# LIST OF FIGURES

1.1 Home automation architecture ........................................... 2

2.1 A Basic FPGA Architecture ................................................ 5

2.2 A Multiplier ........................................................................ 6

2.3 ZigBee 802.15.4 CSMA-CA Center Frequencies (MHz) ............... 8

2.4 An Example ZigBee Network Model ........................................ 8

3.1 User I/Os Per Bank for the XC3S500E in the PQ208 Package ..... 11

3.2 FPGA Banks for the XC3S500E in the PQ208 Package ............... 11

3.3 FPGA Pin Layout according to the banks ................................ 16

3.4 Generated Pad file for the FPGA design ................................. 17

3.5 FPGA - Platform Flash PROM - JTAG Connection for Master  
                Serial Configuration mode ...................................... 18

3.6 Voltage divider circuit .......................................................... 19

3.7 Final board layout test before fabrication ............................... 23

3.8 Back from fabrication .......................................................... 23

3.9 Soldering process ................................................................. 24

3.10 SPI bus tapping for SD Card Initialization commands ............. 26

3.11 The power consumption measurements for the main board ...... 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>MicroBlaze Core Block Diagram</td>
<td>28</td>
</tr>
<tr>
<td>4.2</td>
<td>FPGA Device Utilization Summary for the design</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>ZigBee application Flowchart</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Base Station for the design</td>
<td>34</td>
</tr>
<tr>
<td>4.5</td>
<td>ZigBee beacon device for the user</td>
<td>34</td>
</tr>
<tr>
<td>4.6</td>
<td>ZigBee power switch circuitry attached to the light bulb</td>
<td>35</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Estimated cost for the components</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Ethernet Mode pins and their definitions</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>SPI Interface Signals</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Board layers</td>
<td>21</td>
</tr>
</tbody>
</table>
ABSTRACT

In recent years, rapid development of wireless communication systems has enabled developers to design and increase the connectivity of devices within the home environment. At the same time, smart methods for lowering power consumption have been studied to help combat rising energy costs.

This thesis presents the design and implementation of an energy smart wirelessly controllable home automation network. The design consists of a base station that has an FPGA, acting as a coprocessor, running a Linux distribution on the Xilinx MicroBlaze soft core. The FPGA has many peripherals including Ethernet connectivity and allows for further prototyping of possible communication methods. Also connected to the FPGA is a ZigBee 802.15.4 wireless radio which allows for data to be passed between ZigBee wireless networks to coordinate turning items within the house on and off. This is accomplished by tracking a user through a house using a wearable wireless beacon and automating power saving responses. A simple prototype of the design is implemented using simple electrical devices for validating the basic functions and testing has shown successful results were achieved.

The overall goal of the project is to learn to design an embedded system from scratch and have a platform that can be used as an educational module for future projects, implementations and modifications.
Chapter 1

INTRODUCTION

In recent years, the rapid development of wireless technologies helped our daily lives become easier and more integrated with wireless devices. These developments brought the possibility to have automated home control systems in order to lower our power consumption and have easier management over home devices.

A lot of research has been done about having a home network that provides some kind of a smart solution for making home automation and control easier and lowering power consumption. Different technologies and approaches have been used to have a reasonable and cost-effective solution for this purpose. For example, a bluetooth based home automation [1], or a java-based home automation system which enables you to control your home devices remotely [2]. However, In 2004, with the introduction of the ZigBee standard, having a low-cost and low-power home network became more feasible.

What ZigBee offers is to accomplish a reliable and low-data rate communication, while keeping the system very low-power and cost efficient. There has also been various ZigBee related research and publications for this purpose [3] [4] [5].

Another technology that has been widely used for mostly where the computational power is needed is Field Programmable Gate Arrays (FPGAs). These devices are known as being configurable by the users, and having the ability to have a high parallel processing power.
1.1 Architecture

This thesis presents an FPGA assisted ZigBee home automation system. Figure 1.1 shows the basic architecture of the design. It has a main control board that has the processing power and resources to execute any command that has been passed. A low-cost, and efficient enough FPGA was used to accomplish this task. This board has an Ethernet port to have a connection to the home wired network or to the router to accomplish a remote access from outside, or to send the required packets to the network. It also has a ZigBee module acting as a ZigBee Network Coordinator to establish and coordinate the home network.

The end device boards have a ZigBee module that act as a ZigBee End Device and have a power switch circuitry that works basically as a switch to turn on and off the power line. These boards are connected to the devices that do not need to stay active when nobody is at home such as monitors, TVs, lights, and etc...

Figure 1.1: Home automation architecture
Lastly, the beacon device is carried by the user to provide the basic idea of if the user is inside or outside of the house. It will have a ZigBee End Device that will enable it to talk to the Coordinator module, and batteries to supply the necessary power. The main principle is that, if the user is not present at home, or has left the house, the coordinator will detect this and start the procedure of the automated tasks like turning off the devices that has the ZigBee End Device attached. Whenever the user enters the house, the coordinator will detect it and execute the other sets of procedures.

Chapter 2 introduces some background information about both FPGAs and ZigBee standard to help readers understand the basic idea about these technologies. In Chapter 3, the main board design process is discussed with the selection of the components, board layout and debugging processes. Chapter 4 is about the software workbench and testing of the design. Finally, Chapter 5 presents a summary of the design and introduces some ideas for the future work of the design, and how to improve it to have a better and more efficient solution.
Chapter 2

BACKGROUND INFORMATION: PARTS

2.1 FPGA

A Field Programmable Gate Array (FPGA) is an integrated circuit that is configurable by the user using a *Hardware Description Language* such as VHDL [6] or Verilog HDL [7]. It consists of logic blocks and reconfigurable interconnects that connect these logic blocks together. These logic blocks can be configured to work as logic gates or combinational functions.

There are two big companies in the FPGA market, Xilinx, and Altera. Generally speaking, different vendors name things differently, an example of that would be logic blocks. Xilinx calls them *Configurable Logic Block* whereas Altera calls them *Logic Array Block*. Since the Xilinx Spartan 3E FPGA chip was used in the design, its definitions will be used in this document.

2.1.1 Architecture

An FPGA is a complex device which has different kinds of embedded peripherals. Figure 2.1 shows a basic FPGA architecture which has logic blocks that are connected through the interconnection resources. There are programmable switches on these connections that help with configuring those interconnects.

2.1.1.1 Logic Cells, Slices, CLB

A logic cell is the very basic part in the Xilinx Spartan 3E architecture. It consists of *D-type Flip-Flop* (D-FF) and a four input *look-up table* (LUT). The
logic cell can be used for arithmetic functions or as multiplexers. The other nice feature about the LUTs is that, they can be configured as 16 x 1-bit RAMs, and different implementations can be achieved by combining some of them together. Also they can be configured as 16-bit shift registers using some sort of special dedicated connections.

In some of the Xilinx architectures, two logic cells are called a slice, and four slices are called a configurable logic block (CLB). These numbers may vary depending on the family of the FPGA.

2.1.1.2 Block RAMs, Multipliers

In addition to these CLBs, Xilinx has some extra resources inside the FPGA that are available to use. One type of these extra resources is the Block RAMs. Since applications need memory, these BRAMs can hold several kilo-bytes of data in them depending on the architecture.
Another example for this resources would be *multipliers*. Since most programs require multiplication operations, dedicated multiplier blocks are far more efficient than CLBs [9]. The Spartan 3E family has 18 x 18 bit multipliers dedicated for this purpose [10]. The basic implementation of this multiplier would be to have two 18-bit inputs and one 36-bit output shown in Figure 2.2.

![Multiplier Diagram](image)

**Figure 2.2:** A Multiplier

### 2.1.1.3 Hard and Soft Processors

There are two types of processor cores for an FPGA. These are hard and soft processors. Hard processor is a separate part of the integrated circuit, whereas soft processor is actually implemented by configuring the logic cells. The two examples for these processors would be the PowerPC 405D5 Embedded Processor Core [11] and MicroBlaze Soft Processor Core [12].

### 2.1.2 Typical Applications

The FPGAs have a very wide usage area. Being configurable by the users and having a high parallelism feature are very desired features for most of the applications nowadays. Some of the application areas in that perspective are software-defined radios, cryptography, password cracking, digital signal processing, etc.
2.2 ZIGBEE

ZigBee is a low data rate, low power consumption wireless network protocol based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs) [13] [14]. Its main target areas are radio frequency applications (RF) that requires long battery life, reliability, a wireless connection, and the type of applications that do not require high data rates.

ZigBee products operate in unlicensed frequency bands, such as 2.4 GHz (world-wide), 868 MHz (Europe), and 915 MHz (America). Each band has its own max data rate. 250K bits per second can be achieved in 2.4 GHz, 40K bits per second can be achieved in 915 MHz and 20K bits per second can be achieved in 868 MHz frequency bands. Depending on the frequency, power and environmental characteristics, its range can vary between 10 to 100 meters [15].

ZigBee uses three different kinds of modulation: offset-quadrature phase-shift keying (O-QPSK), binary phase shift keying (BPSK) and amplitude shift keying (ASK). In the 868 Mhz band, there is 1 channel and in the 915 MHz band, there are 10 channels (1 through 10), whereas in the 2.4 GHz band there are 16 ZigBee channels (11 through to 26) with each channel having 2 MHz of bandwidth with a 5 Mhz of channel spacing. For 2.45 Ghz band the center frequency for each channel can be calculated with:

\[ CF = (2405 + 5 \times (Ch - 11)) \text{MHz} \]

where,

CF = Center Frequency

Ch = Channel Number such as 11, 12, 13, ..., 26

For example, if we calculate the center frequency for channel 20 from the formula, we find that:
\[ CF_{ch20} = 2405 + 5 \times (20 - 11) = 2450 \text{MHz} \]

Also center frequencies of each channel can be seen in Figure 2.3.

![ZigBee 802.15.4 CSMA-CA Center Frequencies (MHz)](image)

**Figure 2.3:** ZigBee 802.15.4 CSMA-CA Center Frequencies (MHz)

### 2.2.1 ZigBee Networks

ZigBee Network nodes can be separated into three different kinds based on their roles. *Network Coordinator* (ZC) is basically the one that controls the network. *Router* (ZR) is the one that connects to the coordinator and routes the packets across the network. Lastly, an *End Device* (ZED) is the one that is placed at the end of

![ZigBee Network Model](image)

**Figure 2.4:** An Example ZigBee Network Model
the network. It is either connected to the router or the coordinator, and can receive and transmit, but does not support routing process \[16\].

ZigBee supports multiple network topologies like star and mesh. Each having advantages and disadvantages, they all might be used in different applications. Figure 2.4 shows an example topology that might be used in an application.

2.2.2 Typical Applications

As stated in the beginning of the chapter, ZigBee is used for various embedded systems that require low-data rate and low power consumption. Some of the application areas include home control, building automation, industrial plant monitoring, automated meter reading, healthcare, and etc \[17\].
Chapter 3

PRINTED CIRCUIT BOARD DESIGN

Embedded systems are everywhere in our daily lives. They all come with different configurations and designs depending on their purposes. For that reason some companies provide design tools and programs. There are several choices of programs to design a fairly complicated board like this. For the FPGA design, it was very convenient and helpful to use the Xilinx tools for the certain parts of the flow, because a Xilinx FPGA chip was used. For the PCB design part, Cadence Allegro PCB Design Software was used. The rest of the chapter will be about this hardware design process and important considerations for the parts that are being used.

3.1 Peripheral Selection

Two of the most important issues for this design are to have enough memory for storing and running the applications and to have enough resources for the functionality. In addition to that, a working environment such as a Spartan3E Demo board was considered in order to avoid any unnecessary compatibility issues, and it also provided an example layout of a similar environment.

3.1.1 FPGA

Since a very fast and powerful process speed is not desired, it is convenient to use a low-cost FPGA that has enough I/O pins for the rest of the components and has enough resources for the soft processor core that will be implemented later in the
process. Spartan 3E family was a good candidate for that purpose, therefore, a Xilinx Spartan3E XC3S500E-PQ208 package is chosen to be used. Table 3.1 shows a detailed view of the user I/O pins per bank for the specific package that was used for the design. Figure 3.2 shows the bank layout for the chip.

**Figure 3.1:** User I/Os Per Bank for the XC3S500E in the PQ208 Package

<table>
<thead>
<tr>
<th>Package Edge</th>
<th>VO Bank</th>
<th>Maximum i/O</th>
<th>All Possible I/O Pins by Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VO</td>
<td>INPUT</td>
</tr>
<tr>
<td>Top</td>
<td>0</td>
<td>38</td>
<td>18</td>
</tr>
<tr>
<td>Right</td>
<td>1</td>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>Bottom</td>
<td>2</td>
<td>40</td>
<td>8</td>
</tr>
<tr>
<td>Left</td>
<td>3</td>
<td>40</td>
<td>23</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>158</td>
<td>58</td>
</tr>
</tbody>
</table>

**Figure 3.2:** FPGA Banks for the XC3S500E in the PQ208 Package

### 3.1.2 Platform FLASH

There are several options to configure the Xilinx FPGA. The first one is to download the configuration file directly from a PC using a JTAG programmer.

---

1 A more detailed pin layout can be found in *Spartan-3E FPGA Family: Data Sheet* [10]
For that purpose a 6-pin header was chosen to be placed on the board. However, in order to have a non-volatile memory and have the opportunity to configure the FPGA when it boots up, the XCF04 Platform FLASH chip was added into the design.

3.1.3 DDR-SDRAM chip

DDR-SDRAM was the most challenging part among all of the components because of the fact that it was a high-speed device. The important point on choosing the DDR-SDRAM is to make sure that the selection is compatible with the FPGA package that is used in the design. Since the FPGA package has a limited amount of I/O pins per bank, the suitable DDR-SDRAMs for the XC3S500E-PQ208 package were the ones that have a maximum width of 8 data pins [20]. The reason for that is there are some design restrictions about the data lines and they need to be placed into the same bank. Therefore, Micron [21]’s MT46V64M8 package seems to be the most suitable one for the purpose. Also, a complete list of compatible DDR-SDRAM devices for some of the Xilinx FPGAs and design considerations can be found in *Xilinx Memory Interface Generator (MIG), User Guide* [20].

3.1.4 Ethernet Controller

The Ethernet chip selection was based on the demo board which has a 10/100 Mbit connector, and the related chip was SMSC’s LAN83C185, which is fully compliant with IEEE 802.3 standard, 100BASE-TX support, and runs with +3.3V as with most of the components on the board.

3.1.5 ZigBee Module

XBee-PRO ZB RF Module from Digi [22] is chosen to be used in the design. It operates within the ISM 2.4 GHz frequency band and compatible with RS232
Adapter. It has 250,000 bps RF data rate, and supports *Point-to-point, Point-to-multipoint, Peer-to-peer*, and *Mesh* network topologies. Also it runs within the +3.0V - +3.4V range.

### 3.1.6 Clock Generator

A 50 Mhz Oscillator was chosen that runs with +3.3V level for the main clock source of the FPGA.

### 3.1.7 Power Regulators

The component selection was done in such that they have common voltage levels. The chosen FPGA needs three different voltage levels: 1.2V, 2.5V and 3.3V. The DDR-SDRAM needs 2.5V and a 1.25V reference voltage, and the rest of the board is compatible with 3.3V level. Since there are three common voltage levels (1.2V, 2.5V and 3.3V), three power regulators were used that have a 5V input voltage level. In addition to that, a divider circuit was used to get 1.25V for the DDR-SDRAM reference voltage which is discussed later in this chapter in section 3.3.2.

Table 3.1 shows the estimated cost of the each individual components and the total cost.

### 3.2 FPGA Constraints

The logical initial start point of the design would be to have a pin layout for the FPGA chip since it does not have any specific pin mapping like other chips do. The pins need to be carefully reserved for the other components of the design to have a stable, and efficient communication. Also it would be better to have a rough layout of the components on the board so that the correlated pins can be placed to the closer locations in order to make the routing easier. Xilinx offers a great tool for this purpose called PlanAhead Analysis and Design Tool [23]. The pin mapping was done with the help of this software.
Table 3.1: Estimated cost for the components

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Estimated cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>$28</td>
</tr>
<tr>
<td>Platform Flash</td>
<td>$7</td>
</tr>
<tr>
<td>Oscillators</td>
<td>$3</td>
</tr>
<tr>
<td>Power Regulators</td>
<td>$20</td>
</tr>
<tr>
<td>5V Power Adaptor</td>
<td>$5</td>
</tr>
<tr>
<td>DB-09 Connector</td>
<td>$2</td>
</tr>
<tr>
<td>MAX232 Chip</td>
<td>$2</td>
</tr>
<tr>
<td>DDR-SDRAM</td>
<td>$10</td>
</tr>
<tr>
<td>Ethernet port</td>
<td>$5</td>
</tr>
<tr>
<td>Ethernet chip</td>
<td>$6</td>
</tr>
<tr>
<td>XBee module</td>
<td>$34</td>
</tr>
<tr>
<td>SD Card Slot</td>
<td>$2</td>
</tr>
<tr>
<td>Switches/Buttons/LEDs</td>
<td>$5</td>
</tr>
<tr>
<td>Resistors/Capacitors/Inductors</td>
<td>$15</td>
</tr>
</tbody>
</table>

Estimated Total Cost: $144

3.2.1 Reserving Important Pins

Some pins are reserved for the input voltages and programming purposes. Before starting to map the pins for our design, these pins need to be reserved. As a matter of fact, some of them might still be assigned for the general purpose I/Os, since they are not needed after the configuration process. However, because there are more than enough pins, it is a better approach to reserve and not use them as general purpose I/Os.

Here is a list of these reserved pins and their purposes [24];

- M0, M1, M2: The mode select pins. These pins define the configuration mode that the FPGA uses to load its configuration data such as: Master/Slave Serial (Platform Flash) Mode, Master SPI Mode, and JTAG Mode.

- PROG_B: The program pin. This pin initiates the configuration process, and also forces a master reset on the FPGA.
• **DONE**: When the FPGA successfully completed loading its configuration data, it goes high.

• **CCLK**: The configuration clock pin. It defines the timing for the FPGA’s configuration process. The purpose also depends on the Mode pins. If the mode select pins define a Master mode, then the FPGA internally generates CCLK. If the mode select pins define a Slave mode, then CCLK works as an input to the FPGA from an external timing reference.

• **INIT_B**: In the beginning of the configuration, this pin goes low meaning the FPGA is clearing its configuration memory. After that, it goes low again if the CRC check fails when the FPGA is actively loading its configuration data.

• **TDI, TMS, TCK, TDO**: This pins are for programming the FPGA.

• **HSWAP**: This pin is for user I/O control purpose. When it is low during the configuration, it enables pull-up resistors in all I/O pins to receive I/O bank $V_{CCO}$ inputs respectively.

• **VCC, VCCAUX, VCCINT**: The voltage input pins. These pins must get the correlated voltage values. Each banks’ working voltage level can be changed by supplying different voltage levels to their VCC inputs.

### 3.2.2 Preparing DDR-SDRAM Pins

The second important part of the design was to route the pins for the DDR-SDRAM. Since it is a high speed device, it was very important to route the pins according to both manufacturers’ design considerations. My first attempt was to route these signals by hand just considering the ease of the routing process, which failed and ended up with a non-working RAM. After some research and collaboration, a tool from Xilinx called Memory Interface Generator (MIG) was used.
for this part. It generated an appropriate constraints file for the DDR-SDRAM according to the design considerations and timing issues.

Because of the fact that you cannot have different voltage levels in a bank, Bank 1 and 2 were allocated for the DDR-SDRAM pins, and a +2.5V input was supplied to these banks.

3.2.3 Complete Placement

The rest of the placement was done according to the bank voltage and the rough layout of the board. Bank 0 and 3 were supplied by +3.3V input voltage to be compatible with the rest of the design. Figure 3.3 shows a rough FPGA pin layout.

![FPGA Pin Layout](image)

**Figure 3.3:** FPGA Pin Layout according to the banks
3.2.4 Design Check

In this step, we had a User Constraints File (.ucf) which consists of both the DDR-SDRAM constraints file and the mapping output from PlanAhead software. The next step would be to create a sample project using Xilinx Platform Studio and EDK software [26] and check this constraints file to make sure everything is working as they were intended. After running the Place & Route process, a Pad file would be generated which is basically the design’s pin layout. An example view of this file is given in Figure 3.4. Fortunately, Allegro Part Developer [18] software has an option to import the Pad file generated by Xilinx EDK software. This was very helpful for making the footprint and surely saved me a lot of time.

```
OUTPUT FILE: system.pad
PART TYPE: xc3s500e
SPEED GRADE: -5
PACKAGE: pq208

Pinout by Pin Number:

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Signal Name</th>
<th>Pin Usage</th>
<th>Pin Name</th>
<th>Direction</th>
<th>IO Standard</th>
<th>IO Value</th>
<th>IO Register</th>
<th>Signal Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>PROG_B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>fpga_0_Btns_4Bit_GPIO_in_pin&lt;0&gt;</td>
<td>IBUF</td>
<td>IO_L01P_3</td>
<td>INPUT</td>
<td>LVC莫斯33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>fpga_0_Btns_4Bit_GPIO_in_pin&lt;1&gt;</td>
<td>IBUF</td>
<td>IO_L01N_3</td>
<td>INPUT</td>
<td>LVCamos33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>fpga_0_Btns_4Bit_GPIO_in_pin&lt;2&gt;</td>
<td>IBUF</td>
<td>IO_L02P_3</td>
<td>INPUT</td>
<td>LVCamos33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>DIFS</td>
<td>IO_L02N_3</td>
<td>VREF_3</td>
<td>UNUSED</td>
<td>[3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>fpga_0_Btns_4Bit_GPIO_in_pin&lt;3&gt;</td>
<td>IBUF</td>
<td>IP</td>
<td>INPUT</td>
<td>LVCamos33</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 3.4:** Generated Pad file for the FPGA design

### 3.3 Schematics

#### 3.3.1 FPGA, Platform Flash, JTAG

Xilinx has a helpful diagram shown in Figure 3.5 [24] that shows the connections between the FPGA, Platform Flash PROM, and JTAG pins for the Master Serial Configuration mode. Since there are no Flash devices other than the Platform Flash, and it is also a part of the on-board JTAG logic for providing in-system programming, the Master Serial Configuration mode was the only mode that should
Figure 3.5: FPGA - Platform Flash PROM - JTAG Connection for Master Serial Configuration mode
be used in the design. That’s why, the mode pins were routed to the ground and Platform Flash was added to the JTAG chain.

3.3.2 DDR-SDRAM

The DDR-SDRAM chip that was selected for the design requires an input voltage level between +2.5V and +2.7V, and since we have chosen a +2.5V power regulator already, this was an easy issue to deal with. However, the I/O reference voltage needed for the DDR-SDRAM requires a voltage level between 0.49 \times V_{IN} and 0.51 \times V_{IN}, which is equivalent to 1.225 < V_{REF} < 1.275 for the input voltage level that was chosen for the DDR-SDRAM (+2.5V). In order to get this voltage, a voltage divider circuit was used as shown in Figure 3.6 and +1.25V was derived from +2.5V source.

![Voltage divider circuit](image)

**Figure 3.6:** Voltage divider circuit

The schematic between DDR-SDRAM and FPGA was pretty straight forward. The only important keynote here would be the resistors. There were some termination resistors that need to be placed on the Data and Address lines. Since they were mostly the same values, and 4-array resister packs were used in order to make routing and soldering easier. In addition to that, because one of them was as big as a single 0805 size resistor, this approach saved a lot of routing space around the
DDR-SDRAM. Also, the problem that was encountered with the DDR-SDRAM is discussed in Section 3.6.1.

3.3.3 Ethernet

As stated before, the ethernet chip uses a +3.3V supply voltage as an input voltage. The tx/rx pins routed directly to the FPGA and a 25 MHz Crystal was used for the PLL input. In addition to those, the mode pins control the configuration of the 10/100 digital block. Table 3.2 shows the definitions of the mode pins [27]. Also, the problem that was encountered with these mode pins is discussed in Section 3.6.2.

<table>
<thead>
<tr>
<th>MODE[2:0]</th>
<th>Mode Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>10Base-T Half Duplex. Auto-negotiation disabled.</td>
</tr>
<tr>
<td>001</td>
<td>10Base-T Full Duplex. Auto-negotiation disabled.</td>
</tr>
<tr>
<td>010</td>
<td>100Base-TX Half Duplex. Auto-negotiation disabled.</td>
</tr>
<tr>
<td>011</td>
<td>100Base-TX Full Duplex. Auto-negotiation disabled.</td>
</tr>
<tr>
<td>100</td>
<td>100ase-TX Half Duplex is advertised. Auto-negotiation enabled.</td>
</tr>
<tr>
<td>101</td>
<td>Repeater mode. 100Base-TX Half Duplex is advertised. Auto-negotiation enabled.</td>
</tr>
<tr>
<td>110</td>
<td>Power Down mode.</td>
</tr>
<tr>
<td>111</td>
<td>All capable. Auto-negotiation enabled.</td>
</tr>
</tbody>
</table>

3.3.4 ZigBee Module

The schematic for this module was very easy due to the fact that the module works serially. Therefore, directly connecting each of the Rx and Tx pins to the FPGA was enough. In addition to that, having a range between +3.0V and +3.4V meant there was no need to use another circuitry for the voltage input. For that purpose, a +3.3V input was enough.
3.3.5 SD/MMC Card

The SD/MMC Card connector was placed and connected to the FPGA in Serial Peripheral Interface (SPI) mode to make the implementation easier. There are several third-party Intellectual Properties that can be bought for SD/MMC mode, and, there are also some open-source projects that are freely available [28].

SPI bus is a serial interface in which the data is simultaneously transferred in both directions one bit at a time [29]. One of the important features is to be able to achieve full-duplex data transfers. The SPI Interface Signals are shown in Table 3.3. The transfer rate can be as high as 20 Mbps depending on the device, and it is mostly used for ADC, DAC applications.

<table>
<thead>
<tr>
<th>Table 3.3: SPI Interface Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCLK</td>
</tr>
<tr>
<td>MISO</td>
</tr>
<tr>
<td>MOSI</td>
</tr>
<tr>
<td>CS</td>
</tr>
</tbody>
</table>

The board schematics can be found in Appendix A.

3.4 PCB Layout & Routing

Although, according to the design it would be very convenient to make the board 6 layers due to the fact that there were 3 different voltage levels, it was decided to make the board 4-layers in order to lower the cost of the fabrication. However, 4-layers was good enough to accomplish a nice and successful routing. Table 3.4 shows the layers of the board and what they are used for.

<table>
<thead>
<tr>
<th>Table 3.4: Board layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Layer</td>
</tr>
<tr>
<td>Layer 2</td>
</tr>
<tr>
<td>Layer 3</td>
</tr>
<tr>
<td>Bottom Layer</td>
</tr>
</tbody>
</table>
3.4.1 DDR-SDRAM Design Rules

The most challenging part of routing the DDR-SDRAM signals was to be careful about *the signal integrity* since it is a high speed device. The trace lengths, the line gaps, and even placing the termination resistors closer to DDR-SDRAM or closer to the FPGA make a difference. Fortunately, both Xilinx and Micron provide some feedback about what should be done in order to minimize these negative effects. The other consideration about the DDR-SDRAM signals is the differential pair. These signals need to be routed very close to each other, and need to have some distance from the power lines in order to have a healthy and synchronized signal.

3.5 Fabrication

The board was fabricated by an outside company, and assembled in CVORG labs [30] at the University of Delaware. Pictures of the board can be seen in Figures 3.7, 3.8, and 3.9.
Figure 3.7: Final board layout test before fabrication

Figure 3.8: Back from fabrication
3.6 Debugging Process

3.6.1 DDR-SDRAM problem

After loading the linux kernel on to the FPGA chip, there was some instability issues and random lock-ups. Sometimes lock-ups happened while loading the kernel and sometimes they happened after running the kernel for sometime. Since the occurrences were at random, it has to be about writing and reading the data from memory locations. In order to test that, a software debugging approach was taken. The best option that seems reasonable was to use a program that comes with u-boot[31] which was discussed in section 4.2.

Memory tests showed that, in some memory locations, one or two bits were flipping. My first guess was that the clock speed of the DDR-SDRAM was too fast for the data lines to catch up, and it results with a timing issue that ends up with some possible flipped bits.
The DDR-SDRAM was running at 100 MHz according to the Xilinx configuration. After changing the frequency a little bit, it was seen that less bits were flipped, and decreasing the speed to 80 Mhz solved the problem completely.

### 3.6.2 Ethernet Mode Pins Problem

The problem with the mode pins was related to a confusion in the data sheet. The mode pins were connected to pull-down resistors that are going to +3.3V in the design.

The Ethernet cable that was plugged in to the board was not detected by the PC automatically. In order to see the properties of the connection on the Ethernet, a tool called `ethtool` was used. It was seen that, the other end of the cable was set on 10Base-T and half duplex mode. In addition to that, the auto-negotiation was disabled. Trying to set that automatically did not help much, and ended up the same.

Some hardware debugging on the Ethernet chip was done, but no problem in particular was encountered. However, after going through the data sheet, I realized that the problem might be related to the mode pins. Fortunately, just disconnecting them from the resistors and leaving them unconnected resulted in a successful connection.

Moreover, some additional debugging processes is shown in Figures 3.10 and 3.11.
Figure 3.10: SPI bus tapping for SD Card Initialization commands

Figure 3.11: The power consumption measurements for the main board
Chapter 4

SOFTWARE WORKBENCH, APPLICATIONS, AND TESTING

Since hand-held devices are becoming more and more popular, running a fully functional operating system in an embedded system is a very neat feature. Also it enables the opportunities to have various features like having a web server or having remote access on a hand-held device.

The soft processor core was mentioned in section 2.1.1.3, but it was not explained in enough detail. The FPGA chip that was used in the design has the capability and resources to have a MicroBlaze soft processor core. Developers have been working on Linux support in this architecture, and they have been able to support it for quite some time now.

In that context, it was decided to run a Linux operating system on the main board. One of the other important reasons for this approach is to integrate the developed applications easily to the working system and have an expandable workbench.

4.1 MicroBlaze soft processor core

The MicroBlaze is an embedded 32-bit soft processor core as mentioned in section 2.1.1.3 and is developed by Xilinx. It has a 32-bit RISC Harvard architecture and comes with a native PLB bus interface. A block diagram of MicroBlaze core can be seen in Figure 4.1.
MicroBlaze architecture has some fixed features like a 32-bit address bus and 32 general purpose registers, and most of it is configurable by the user such as the pipeline depth (3 or 5 stage), instruction and data cache sizes, Floating Point Unit, and Barrel Shifter. In addition to those, v7.0 recently added support for the Memory Management Unit, which is basically designed to make the processor operate in a virtual address space [35]. Unfortunately, the resources that the Xilinx chip, which was selected for the design, is not enough to implement MMU support, which is discussed as future work in section 5.1.

Also in 2009, MicroBlaze was merged into and became a part of the mainline Linux Kernel Source tree [36].

### 4.2 Preparing the Boot loader

Most of the systems need a boot loader in order to boot the kernel. One of the most common ones in embedded systems is Das U-Boot (the Universal Boot-loader) [31] commonly referred to as u-boot. It supports many different architectures
and has the option to load the kernel from an NFS share or using TFTP, which decreases the kernel image load time.

The kernel image was put into a machine that has a TFTP server running. After booting into u-boot, a predefined script automatically copies the kernel image into the next available memory location on DDR-SDRAM, and starts to boot into the kernel.

There are two ways to prepare the u-boot image that makes it easier than the others. The first one is from Petalinux [37] and the second one is from Xilinx [38]. They both have their own scripts for preparing the image. However, Xilinx is only for kernels that have MMU support, and Petalinux has both of the options like the traditional uClinux way, as well as the newer stack kernel. Therefore the preparation of the u-boot image depends on which approach is chosen. For this design, Petalinux was chosen since it was not possible to implement MMU support into the design because of the limited resources.

In general, in order to prepare the u-boot image, the first thing to do is to create an EDK project that has all the peripherals added and configured properly, and also include the scripts to generate the files that is needed for u-boot configuration. After generating these files, which basically have the information about peripherals’ locations and network configuration, they were inserted into the u-boot project. u-boot image was obtained using the default MicroBlaze cross compiler that Petalinux offers: The utilization of the board can be viewed from the EDK Project and is shown in Figure 4.2.

4.3 Preparing the Kernel

There are a few options for running a linux kernel on an FPGA and one of the best options is uClinux, which is described as a Linux/Microcontroller project that is a port of Linux to systems without a Memory Management Unit [39]. Also,
Petalinux has a nice software development kit that helps users save some time by doing some of the work automatically with various scripts.

The previously generated files from EDK project was transferred into the specific board configuration folder in the kernel tree. After defining architecture and compiling the kernel, the image files for the kernel were generated.

These two steps were done repetitively in order to have a working kernel. There were some problems in every step that ended up with a non-working boot loader or kernel. Some of them were related to mapping nets to the wrong pins and some of them were related to the small mistakes in peripherals’ configurations in the EDK project.

For detailed explanations, Petalinux and Xilinx pages both have very useful information about these steps.

4.4 Building Applications

The base Linux system was ready and running on the board on this step, all the peripherals were tested under Linux and verified that they are working except ZigBee. Some functions like an SSH server, or an HTTP server were installed,
and they were ready to use. The missing parts were the applications that are not installed by default and needed for this project like the main application for the ZigBee network.

### 4.4.1 microcom

The first step was to test the ZigBee module with the help of a serial terminal. After some searching on the internet for a small sized one, I came across microcom [40], which is a small terminal emulator. It was compiled with the help of cross compiler and was ready to use.

### 4.4.2 ZigBee Application

Since the ZigBee was tested with the help of microcom, everything was good to move to the last step which is the write the main application.

The ZigBee network has three different types of nodes. The first one is the **Network Coordinator** which is attached to the main board and responsible for controlling of the other ZigBee devices. The second type is the **ZigBee Beacon** which is responsible for notifying the coordinator about the presence of the user. This device is carried by the user all the time, and basically consists of a microcontroller, a ZigBee module, and a battery. The role of this device is to send out presence packets at a given interval of time, and inform the Network Coordinator that the user is around. Lastly, the third type is the **ZigBee Client** which is connected between the power plug and the dump devices like monitors, light bulbs, music players, etc. This device has a microcontroller and a ZigBee module attached to a power switch circuit which basically switches on or off the line with an input signal.

The main board has a ZigBee application running in the background all the time. A flowchart for the basic functionality is given in Figure [4.3].

This application initializes and maintains the ZigBee network. It listens for a presence packet in order to decide whether to turn on or off the clients.
Figure 4.3: ZigBee application Flowchart
minutes for a presence packet, and if no packet arrives in that interval, it sends out a request packet before turning off the clients as a last step. Then, again if no replies are received, it sends out the packet for turning off the devices. In addition to that, whenever the user comes within the range of the network, the Network Coordinator detects the user and sends out the packet for turning on the devices.

The Presence Indicator node has an application that basically sends out presence packets in a given interval of time. Also, it has a function to reply back when it receives a request packet.

Lastly, as stated before, the ZigBee Client boards goes between the electrical devices and the power plug. They enable or disable the connection between them depending on the received packets from the main board.

After everything was completed, all the parts brought together and the system was ready to test.

4.5 Testing

A test of a simple home automation network system for validating the basic functionality of the design was created. It consisted of the base station connected to a router, a ZigBee beacon for presence indication, and a ZigBee power switch circuitry attached to a light bulb. The pictures of the system are shown in the Figures 4.4, 4.5, and 4.6.
**Figure 4.4:** Base Station for the design

**Figure 4.5:** ZigBee beacon device for the user
Figure 4.6: ZigBee power switch circuitry attached to the light bulb
Chapter 5

CONCLUSIONS AND FUTURE WORK

This thesis has presented a design approach for a low-cost low-power home automation network using ZigBee wireless radios. The results of the test show that the base station successfully detects the user’s presence and turns the item on and off, and can be used as a smart solution for home automation network control while lowering power consumption.

The design may still be improved in a variety of ways. These improvements and additions might be about improving the system usability for users to have a better experience as well as improving the system performance and reliability. Some of the initial ideas for future work are discussed in the next sections.

5.1 MMU support

For the hardware side of the design, the first thing that should be improved is to replace the FPGA chip with a more resourceful one in order to improve the reliability of the system. As stated in section 4.1, the chip that was used in the design does not have enough resources to have an MMU enabled kernel which would prevent some of the potential errors in memory management, and also improve the system’s reliability and dependability.

A GUI design for remote control might be implemented in order to help users manage the environment easier.
5.2 Wake-On-Lan Support

Wake-On-Lan [41] is a protocol allows a computer to be woken up or powered on by sending a specially crafted network packet. It is only available on Ethernet cards and motherboards which have a 3-pin Wake-on-LAN connector.

The basic principle of WoL is that a Network Card listens for a specific packet from the network at which point it will switch on the machine, even though the machine is powered off. By having an Ethernet port on the FPGA board, and a connection to the local area network, users can take advantage of this feature to save more power combined with an ACPI aware server running on each machine, systems can be powered on and off over the network.

This would be a very helpful feature for the design since the main board can detect the users presence, it can automatically turn on or wake up the user’s computer when the users enters to the ZigBee Network coverage.

5.3 Expanding the environment

The design that was done for this thesis was intended for home environment where there is only one user present. The design however, has the ability to scale up to a multi-user environment if needed. This presents the possibility of implementing the system in a work place, where there are more than one user and each has their own monitor, computer, light, cooling system, and etc.
Appendix A

BOARD SCHEMATICS

The schematics for the board design is shown here.
BIBLIOGRAPHY


[8] FPGA Architecture Figure. [http://www.design-reuse.com/articles/10991/back-to-the-basics-programmable-systems-on-a-chip.html](http://www.design-reuse.com/articles/10991/back-to-the-basics-programmable-systems-on-a-chip.html). 2.1.1


