An architecture and technology for Ambient Intelligence Node

Tomasz M. Jankowski

University of New Hampshire, Durham

Follow this and additional works at: https://scholars.unh.edu/thesis

Recommended Citation
https://scholars.unh.edu/thesis/422

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars’ Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact Scholarly.Communication@unh.edu.
AN ARCHITECTURE AND TECHNOLOGY FOR
AMBIENT INTELLIGENCE NODE

by

Tomasz M. Jankowski

MS, Technical University of Gdansk, Poland, 2002

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Electrical and Computer Engineering

December, 2008
This thesis has been examined and approved.

Thesis Director, Dr. Andrzej Rucinski, Professor

Dr. John LaCourse, Professor and Department Chairman

Dr. W. Thomas Miller, Professor

Dr. Thaddeus P. Kochanski, Affiliate Professor

Dr. Pieter Mosterman, Senior Research Scientist, The MathWorks

Date
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE OF CONTENTS</td>
<td>IV</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>VI</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>VII</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>VIII</td>
</tr>
<tr>
<td>LIST OF ACRONYMS</td>
<td>IX</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>X</td>
</tr>
<tr>
<td>STATEMENT OF WORK</td>
<td>XII</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Prototyping Specifications for Ambient Node</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Scope of the Thesis</td>
<td>4</td>
</tr>
<tr>
<td>AMBIENT INTELLIGENCE ENVIRONMENT</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Concepts and Definitions</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Overview of Ambient Intelligence Technology</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Autonomous Micro Device</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Portable Mini Device</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Static Watt Device</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Ambient Intelligence Network Structure</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Topology</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Communication range</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3 Function</td>
<td>16</td>
</tr>
<tr>
<td>2.3.4 Ambient Network</td>
<td>17</td>
</tr>
<tr>
<td>AMBIENT TECHNOLOGY MAPPING</td>
<td>19</td>
</tr>
<tr>
<td>3.1 Concepts and Challenges</td>
<td>19</td>
</tr>
<tr>
<td>3.2 SoC FPGA vs ASIC</td>
<td>19</td>
</tr>
<tr>
<td>3.3 SoC FPGA</td>
<td>21</td>
</tr>
<tr>
<td>3.4 SoC Bus Technology</td>
<td>24</td>
</tr>
<tr>
<td>3.4.1 CoreConnect</td>
<td>25</td>
</tr>
<tr>
<td>3.4.2 AMBA</td>
<td>26</td>
</tr>
<tr>
<td>3.4.3 WishBone</td>
<td>27</td>
</tr>
<tr>
<td>3.4.4 MicroBlaze Local Memory Bus and Fast Simplex Link (Xilinx)</td>
<td>28</td>
</tr>
<tr>
<td>3.5 SoC Processor</td>
<td>29</td>
</tr>
<tr>
<td>AMBIENT INTELLIGENCE NODE : ARCHITECTURE OVERVIEW</td>
<td>31</td>
</tr>
<tr>
<td>4.1 AIN Node System Architecture</td>
<td>31</td>
</tr>
<tr>
<td>4.1.1 AIN Node Base System Overview</td>
<td>32</td>
</tr>
<tr>
<td>4.1.2 AIN Node Operation System Overview</td>
<td>34</td>
</tr>
<tr>
<td>4.2 AIN Node DSP Overview</td>
<td>34</td>
</tr>
<tr>
<td>AMBIENT INTELLIGENCE NODE PROTOTYPING</td>
<td>37</td>
</tr>
<tr>
<td>5.1 Node Design</td>
<td>40</td>
</tr>
<tr>
<td>5.2 Node Software Detail</td>
<td>42</td>
</tr>
<tr>
<td>5.3 DSP Block Design Detail</td>
<td>44</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

Dr Barbara Rucinska and Dr Andrzej Rucinski for the opportunity.

Donald P. Bliss, NI2 Center for Infrastructure Expertise, Director and Dale Ferriere, NI2 Center for Infrastructure Expertise, Deputy Director for the financial support

Dr. Thaddeus P. Kochanski, Affiliate Professor and Frank Hludik, ECE Department, Faculty Member for the knowledge.

Chris, Len, George and the other team members from the CIDLab for the good time we spent.

Xilinx Inc. and The MathWorks for the tools.

My wife Ewa for her time and support.
LIST OF FIGURES

Figure 1: Block Diagram of Thesis Elements .......................................................... 4
Figure 2: Communication Technologies Mapped on Power/Communication Range
  Graph .................................................................................................................. 9
Figure 3: Clustering of Technologies in Devices ..................................................... 10
Figure 4: Functional Block Diagram of MicroWatt node for Contextual Awareness .. 11
Figure 5: Functional Block Diagram of MilliWatt node ........................................... 12
Figure 6: Watt node for Natural Interfaces ............................................................. 13
Figure 7: Network Topology [IPB05] ................................................................. 15
Figure 8: Current (a) and Ambient (b) Networks ................................................ 17
Figure 9: Hardware Architecture with Separate DSP Banks .................................. 20
Figure 10: Hardware Architecture with Separate CPU for Separate Tasks .......... 20
Figure 11: Hardware Architecture FPGA/ASIC Hybrid vs Multi core Processor .. 21
Figure 12: FPGA Altera and Xilinx Family Group ................................................. 24
Figure 13: The structure of CoreConnect Bus ..................................................... 26
Figure 14: The Structure of the AMBA Bus ......................................................... 27
Figure 15: The Structure of WishBone Bus .......................................................... 28
Figure 16: Block Diagram of Hard-Core CPU Embedded System (Courtesy of Xilinx). 29
Figure 17: Block Diagram of Soft-Core CPU Embedded System (Courtesy of Xilinx) . 30
Figure 18: AIN Node Functional Block Diagram ................................................. 33
Figure 19: Node Hardware Components Block Diagram ....................................... 33
Figure 20: Node Hardware Components Block Diagram ....................................... 40
Figure 21: Flowchart for AIN Implementation .................................................... 41
Figure 22: System Base Elements ....................................................................... 42
Figure 23: Relationship Diagram of EDK Flow Steps and Key Files (Courtesy of Xilinx) ................................................................. 44
Figure 24: System Generator ............................................................................. 45
Figure 25: System Generator ............................................................................. 46
Figure 26: System Generator – Hardware in the Loop Testing ......................... 48
Figure 27: System Generator – Resource Estimator ............................................. 48
Figure 28: Peripherals Memory Address Ranges ................................................... 49
LIST OF TABLES

Table 1: Statement of Work ................................................................. XII
Table 2: Ambient Node Classification Parameters .................................. 9
Table 3: Network Topology Classification with Parameters ...................... 15
Table 4: FPGA Actel, Altera and Xilinx Parameters Comparison .............. 22
Table 5: FPGA Actel, Altera and Xilinx Parameters Comparison 2 ............ 23
LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/V</td>
<td>Audio/Video</td>
</tr>
<tr>
<td>AIN</td>
<td>Ambient Intelligence Network</td>
</tr>
<tr>
<td>AMBA</td>
<td>Advanced Microcontroller Bus Architecture</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuits</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>DVT</td>
<td>Design Verification Test</td>
</tr>
<tr>
<td>EDK</td>
<td>Embedded Development Kit</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GAIN</td>
<td>Global Ambient Intelligence Network</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IPC</td>
<td>Information Processing Capacity</td>
</tr>
<tr>
<td>ISE</td>
<td>Integrated Software Environment</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistants</td>
</tr>
<tr>
<td>SoC</td>
<td>System-on-a-Chip</td>
</tr>
</tbody>
</table>
ABSTRACT

AN ARCHITECTURE AND TECHNOLOGY FOR AMBIENT INTELLIGENCE NODE

by

Tomasz M. Jankowski

University of New Hampshire, December 2008

The era of separate networks is over. The existing technology leaders are preparing a big change in recreation of environment around us. There are several faces for this change. Names like Ambient Intelligence, Ambient Network, IP Multimedia Subsystem and others were created all over the Globe. Regardless of which name is used the new network will combine three main functional principles – it will be: contextual aware, ubiquitous access and intelligent interfaces unified network.

Even if there are many faces at the heart there is always one – the technology. Currently no technology exists on the market that would provide a comprehensive solution to fit this concept. Moreover, no architecture standards or design specifications have been defined yet.

Within this thesis two major aspects are defined. First, the definition of the Ambient Intelligence Environment concept is presented. Secondly the architecture vectors for the technology are named. A short overview of the existing technology is followed by details for the chosen technology – FPGA. The overall specifications are
incorporated in the design and demonstration of a basic Ambient Intelligence Node created in the System on the Chip (SoC) FPGA technology.
STATEMENT OF WORK

The project was divided into two simultaneous theses. The modular approach in
to the hardware design together with the automation and deep overview of the software
components of the project is described in a parallel thesis “Application Development
Process for GNAT, a SOC Networked System” by Christopher L. Plumlee, BSEE. The
following table lists the credits for the overall project:

<table>
<thead>
<tr>
<th>Tomasz Jankowski</th>
<th>Joint Effort</th>
<th>Chris Plumlee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Statement of Work</td>
<td>Introduction</td>
</tr>
<tr>
<td>Ambient Intelligence Environment</td>
<td>Node Design Requirements</td>
<td>Model Driven Architecture</td>
</tr>
<tr>
<td>Ambient Intelligence Environment</td>
<td>Implementation</td>
<td>Node Design</td>
</tr>
<tr>
<td>Ambient Technology Mapping</td>
<td>Integration</td>
<td>DSP Block Design</td>
</tr>
<tr>
<td>Ambient Intelligence Environment: Architecture</td>
<td>Node Software</td>
<td>Results</td>
</tr>
<tr>
<td>Ambient Intelligence Node Implementation</td>
<td>Development Roadmap</td>
<td>GNATDVD</td>
</tr>
</tbody>
</table>

Table 1: Statement of Work
CHAPTER 1

INTRODUCTION

This chapter presents the motivation of the thesis. It also includes the prototypic specification together with scope of the project.

Def. 1

Network – any method of sharing information between two systems” [WNET]

1.1 Motivation

We live in the Network. The gap between the human location and the entrance to the Network ended in the last century. These days, we are surrounded by the Network or, to be more precise, many networks. If we only are able to hear all the network frequencies around us, we would realize that we live in a loudly and crowded corner of the galaxy. Last century was the era of the Network creation. The telecommunication, TV, radio, mobile and Internet networks were invented and designed. The first computers, TV, cell phones and telephones as the interfaces to the networks were created. But the era of closed unique signalization networks has ended. A new era is beginning - a time of the networks integration. What it brings is the challenge of all network redesign together with the creation of new interfaces among them. What it means from the engineering point of view is a big challenge in the redesign of the systems and the elements of the networks. Moreover, the definition of the network is changing. A new
network will bring not only the medium capabilities to share the information – instead it will become a method of ubiquitous meaning sharing as well – it will become an Ambient Intelligence Network (AIN).

The challenge in the definition of interfaces and signalization for integrated AIN is significant. But the biggest challenge in this new vision is the hardware capabilities. It is easy to realize that every new concept is as good as the hardware capabilities. The challenge became more important when at the beginning of this century the hardware designers realized the capabilities of a unicore computer had ended. New approaches to this problem were created. The existing processor industry is changing from the single processor unit to multi core technology trying to reuse already created modules. From another side a new approach of hardware design was born – the programmable hardware industry. The FPGA industry is demonstrating that not all processes need to run on the processor. The idea is that some of the processing work can be done on the separate optimized DSP modules controlled easily by processors. This approach already found some industry sponsors as the cost of design and the implementation of the FPGA based systems is much less lower than that of ASIC based systems. Unfortunately the gap between the concepts and the implementation is still wide.

There are two major tasks in this thesis. First, standardize the Ambient Intelligence Network concept. Secondly proves that it is possible to create an element of the AIN concept in FPGA technology based on the open and reusable hardware architecture. The implementation part of the thesis focuses on building the basic block, proving that it can be achieved to create a complete, reusable node in the System on the Chip technology using finite resources and time.
1.2 Prototyping Specifications for Ambient Node

While a successful Ambient Intelligence Network (AIN) system will generally include a wide variety of Nodes, design templates and methods abound for both the large and small of scale - it is the intermediate scale of node, the portable mini device which it will be focused on in this project. Before a decision between the technology choices was made, the design requirements of a node in this scale were defined:

- **Information Processing Capacity (IPC)** - The node does not merely collect and relay data. It must make intelligent decisions about all the data and noise presented to it, in order to avoid idle chatter within the overarching AIN system.

- **Ranged Communication** - in order to be an Ambient Node, each device must be capable of communications over at least one medium.

- **Power requirements** - Capabilities do not come free; power goes in, data comes out. Nodes on this scale should be relatively small and unobtrusive. The ability to run temporarily on batteries or permanently from, for example, solar panels would further expand this scale of node's marketability.

- **Adaptability** - Needs change. In order for a Node to remain useful for its entire operating life, it will need to be re-purposed. If this can be done remotely, the node becomes exponentially more useful and linearly more profitable.

- **Autonomy** - The basic node design must capture all this and yet be self-contained. Any node must be able to carry out its function independently in case it is not deployed in, or becomes cut off from an overarching system.
- **Short Design Cycle** – Devices in this scale are not deployed in great enough numbers to justify a lot of engineering investment.

### 1.3 Scope of the Thesis

The work presented in this thesis was divided into several steps:

- Define Ambient Network Technology specifications, thus determine requirements for the Node
- Research technology bases applicable to the Node
- Propose Software and Hardware Architecture for Ambient Intelligent Environment
- Implementation

The scope of the work divided into several steps is presented in the block diagram below.

![Figure 1: Block Diagram of Thesis Elements](image-url)
CHAPTER 2

AMBIENT INTELLIGENCE ENVIRONMENT

This chapter introduces and defines the ambient environment. The scope of the project is to synchronize theoretical concepts of ambient environment with the up to date technology capabilities. This chapter introduces design requirements for an ambient intelligence network containing reprogrammable nodes, where both network and node topology, together with their functions can be variable vectors. The results contain design specifications for a network of nodes integrated within an ambient environment which all together provides a universal skeleton for a user defined system.

2.1 Concepts and Definitions

The Ambient Network assumes that people are surrounded by an intelligent environment. The intelligent environment is a combination of devices that function to share information and meaning in the right place and in the right time, and of course in the right way. The basic structure of these environments includes a combination of autonomous, yet networked, nodes, that act in concert. This network of nodes, properly managed, forms a coherent system that can be utilized to perform any type of service.

Ambient Intelligence, “the State of the Art”

The Ambient concepts vary:
Def. 1

"Ambient Intelligence envisions a world where people are surrounded by intelligent and intuitive interfaces embedded in the everyday objects around them. These interfaces recognize and respond to the presence and behavior of an individual in a personalized and relevant way."[AIP05]

Def. 2

"Where the electronic functions spontaneously organize themselves in networks to provide their human companions with communication..."[AIT05]

Similar to the variation in the ambient environment definition there are several scopes for the ambient environment research:

- **Project Oxygen, Pervasive, human-centered computing**
  - "communicate naturally, using speech and gestures that describe our intent ...
    ...and leave it to the computer to carry out our will "[MPO04]

- **Ambient Intelligence Group**
  - "radically rethink the human-machine interactive experience. By designing interfaces that are more immersive, more intelligent, and more interactive we are changing the human-machine relationship and creating systems that are more responsive to people's needs and actions, and that become true "accessories" for expanding our minds. "[MAI06]

- **Philips HomeLab**
  - "The stylized electronic home of the future won't have a mind of its own, but it could be so finely programmed that it will know – even anticipate – what its occupants are thinking." [PHL02]
• European’s Information Society

  o “A key feature (of Ambient Intelligence) is the ability for seamless movement
  ... : people on the move will become networks on the move as the devices
  they carry network together and connect with the different networks around
  them.” [EIS04]

Regardless of which definition is used, there is a consensus regarding three elementary requirements for ambient environment:

• **Contextual awareness** - Sensors integrated into environment communicate events. Events are interpreted, i.e. when you meet a ball as big as the Eiffel tower? Is it normal? Why do you think not?

• **Ubiquitous access** - Available anywhere, integrated in physical environment & objects around us (environment as a storage and communications access).

• **Intelligent interfaces** - Existing human/machine interfaces are keyboard, mouse, and display. We can design more than that and utilize additional human senses like hearing and touch.

Two kind of ambient aspects are defined in this thesis:

• The construction of a reprogrammable, adaptable, self-contained hardware node as a base for both hardware and software additional modules.

• The node-to-node communication which enables creation of a scalable, intelligent network.
2.2 Overview of Ambient Intelligence Technology

Ambient Nodes can be characterized by their performance along three vectors:

1. **Information Processing Capacity** (IPC) – while this is a soft vector in FPGA-based Nodes, it plays a major role in determining the purposes for which a given node is suited. This computation vector describes device’s capabilities for data processing and storage.

2. **Ranged Communication** (RC) – in order to be an Ambient Node, each device must be capable of communications over at least one medium. Media compatibility determines the relationships that must be present in a system for successful deployment of each Node. While range and bandwidth are the primary parameters of interest, they are almost entirely determined by the supported communications media, whether it be a 3-meter radius low-speed Bluetooth connection or a broadband satellite link.

3. **Power requirements** (P) – Processing capabilities do not come free; power needs to go in, data can come out. The quantity and kind of power required determine the deployment possibilities for a given Node design.

We can define the superspace of the Ambient Intelligence Technology vectors as:

\[
\text{AIN superspace} = \text{IPC} \times \text{RC} \times \text{P}
\]

By generalizing Nodes to these three measurements, one proposed classifications system[ICD03] groups all of the elements into three categories: micro, mili and maxi devices. These classes are generally suited to different purposes as follows:
- Autonomous Micro - "Awareness"
- Portable Mini - "A/V everywhere"
- Static Maxi - "Natural interfaces and Servers"

The dividing lines for these criteria generally agreed upon are presented in the table below:

Table 2: Ambient Node Classification Parameters.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Dimension</th>
<th>Power</th>
<th>IPC [bps]</th>
<th>Communication Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro</td>
<td>Micrometer</td>
<td>Micro-Watt</td>
<td>1k</td>
<td>LAN (mm – 100m)</td>
</tr>
<tr>
<td>Mini</td>
<td>Millimeter</td>
<td>Milli-Watt</td>
<td>1M</td>
<td>WAN (m – km)</td>
</tr>
<tr>
<td>Maxi</td>
<td>Centimeter</td>
<td>Watt</td>
<td>1G</td>
<td>GLOBAL (K km)</td>
</tr>
</tbody>
</table>

The next two pictures categorize common technology examples by power needs, communication range and information processing parameters. While there is a continuous spectrum, the technologies do cluster well into three ambient technology classes according to the availability of already developed architectures.

Figure 2: Communication Technologies Mapped on Power/Communication Range Graph.
Figure 3: Clustering of Technologies in Devices.

In this project the milliWatt Node element of the Ambient Network was developed. The detailed information regarding each cluster of Ambient technology is presented below.

2.2.1 Autonomous Micro Device

The Autonomous Micro Device is the class of autonomous micro devices that characterizes the smallest size, the lowest energy needs, and the lowest processing capabilities. This class of device meets the functional needs of awareness elements -- to collect basic information data. These elements can be implemented either as small hardware elements or as software agents. The hardware implementations range from basic probes, passively collecting data, to active environmental function monitors, emitting waves and processing reflected distortions. Existing technologies available today include all kinds of chemical/radiation detectors, vibration/impact sensors, temperature, sound, and video recorders. Software agents, on the other hand, monitor the virtual world coexisting on top of the milliWatt or Watt elements. These tools allow us
to monitor the environment in which the node is installed, finding bottlenecks or detecting intrusion. They power, processing and networking capabilities limits the node on which are they installed on. While all AIN nodes contain the same general building blocks, microWatt nodes tend towards reliance on the smallest and least expensive technologies in each category. The figure below presents functional block diagram of microWatt structure, including the underlying technologies used.

![Functional Block Diagram of MicroWatt node for Contextual Awareness.](image)

2.2.2 Portable Mini Device

The Portable Milli (or Mini) device built on the milliWatt scale will arbitrate the most important functions in ambient environment: the connection bridge. The bridge concept includes both a human/machine interface, and machine/machine networks interconnect. Current examples of Portable Mini devices include the cellular phone, the portable media player, and the personal digital assistants (PDA). These elements are equipped with higher digital processing capabilities and higher bandwidth, enabling text/audio/video communication. Devices in this group often perform a primary function: bridging communication between two networks. This function requires varying combinations of traffic prioritization, address translation, flow management and data compression.
Relative to the other two classes of nodes, the Portable Mini device is generally most customizable for the individual human user. Said user in this environment can customize the device’s interface based on his or her experience and needs. Some interfaces offer more than usual display/keyboard elements, taking advantage of the concept of “sense utilization”. But these devices are not ambient nodes yet because they do not utilize the integrated ambient intelligence environment concept. The general design recommendation for milliWatt-scale Ambient Nodes interfaces were already presented [AID03]. To list the most important:

- Ambient interfaces should be effective and efficient – a new user should understand the basic function of the ambient environment without additional learning steps. The user should be able to use it to shorten the time to complete a task.

- Ambient interfaces should be safe – This aspect brings into consideration two areas, protection from dangerous anomalies - the first. The second is the ergonomical aspect of the device.

- Ambient interfaces should have good utilization and feedback – It is important that the user and ambient environment have good communication. To achieve this, feedback should be provided from the environment.

![Functional Block Diagram of MilliWatt node](image)
The ambient environment should have system awareness of the current situation and the user who are participating. The dynamics of the situation should trigger exact interaction provided only to defined clients.

### 2.2.3 Static Watt Device

The largest class of Ambient Devices generally uses fast computing, wide bandwidth, and relatively large volumes of storage. The natural implication of these capabilities will be extended need for energy, tending towards static installations. Examples of such devices are multi-user servers, database farms, and most data routers. These Nodes can most inexpensively support multiple communication media, often utilizing broadband, wireless, and/or optical transport technology. One of the most important characteristics of fixed computing elements is their ability to correlate and interpret data to form system intelligence. Data mining, pattern recognition, statistical analysis, and other artificial intelligence algorithms are best run on these larger machines with greater resources.

![Figure 6: Watt node for Natural Interfaces](image_url)
2.3 Ambient Intelligence Network Structure

The Ambient Network as said before will integrate other networks. Moreover, its main role is human - human interaction regardless of place and time. The basic network is defined as the communication link between at least two elements of the AIN environment. The specifications of the network depend on the technology and signalization used in implementation. The huge expansion of the Internet and the popularization of Internet Protocol (IP) have directed nearly all network development into packet switched IP networks.

The classification of the networks depends on:

- Topology
- Physical range
- Function

Where the topology presents the structure of the interconnections of network devices, physical range describes the link capabilities and function defines the services utilized over the network.

2.3.1 Topology

There are three main structures of network topology:

a) Centralized
b) Decentralized (Hierarchical)
c) Distributed

The picture below presents these network structures, and it is followed by a table contrasting their features. The first two structures, centralized and hierarchical, are
common existing communication topologies. The last structure, the distributed network, was proposed by Pawel Baran, a Polish-American engineer. This network structure was designed for continuity of service in the event of nuclear attack. This invention leverages the natural structure of the brain, where function does not rely on a single set of dedicated cells. There is not single point of failure.

![Network Topologies](image)

Figure 7: Network Topology [IPB05].

The table below presents the parameters of each network.

<table>
<thead>
<tr>
<th>Centralized</th>
<th>Hierarchical</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single (backup-ed) Master topology</td>
<td>Master-Slave topology</td>
<td>Flat Topology</td>
</tr>
<tr>
<td>Easy Management/Security</td>
<td>Distribution of functions/processes</td>
<td>Distribution of functions/processes</td>
</tr>
<tr>
<td>Not too scalable</td>
<td>Built in Redundancy</td>
<td>Full Redundancy</td>
</tr>
<tr>
<td>High demand on Performance (CPU)</td>
<td>More scalable</td>
<td>Highly scalable</td>
</tr>
<tr>
<td>Big demand on “Wide” interface bandwidth</td>
<td>Less demand on Performance (CPU) and interface bandwidth</td>
<td>Less demand on Performance (CPU) and interface bandwidth</td>
</tr>
<tr>
<td></td>
<td>Harder to manage</td>
<td>Harder to manage</td>
</tr>
<tr>
<td></td>
<td>Cross servers dependency for information gathering</td>
<td>All information gathering at each single location</td>
</tr>
</tbody>
</table>
2.3.2 Communication range

Communication range of a given network depends on physical parameters of the transport medium where a message is transported. The most popular medium for electromagnetical waves in today’s communication links are:

- Wired
  - Electric
  - Optical
- Wireless

The technology utilized in ambient network for milli and micro devices mostly based on wireless technology e.g. ZigBee, Wifi, IRDa. Static Watts nodes are able to benefit from higher bandwidth supplied by a wired data transportation.

2.3.3 Function

The function of the network depends on the function of its elements. The most popular networks functions are:

- Data Transportation (mid point; several input to several outputs) e.g. network gateway between two networks communicating over different mediums
- Data Storage (end point function; several input to several output) e.g. Storage area networks.
- Data Processing (end- or mid-point function; several inputs to several outputs) e.g. network system management.
- Data Collection (start point function; several inputs to one output) e.g. sensor network
2.3.4 Ambient Network

In order to complete the AIN environment the concept of ubiquitous access for milliWatt nodes, the paradigms of wireless connection needs to be taken addressed. The new concept of Ambient Networks brings also many new challenges. In discussions during the Ambient Network project meetings (phase II of the Ambient Networks project cosponsored by the European Union) [ANP06], the idea of separation of the access/transport network from the service providers was developed. The new system view for the current and the new ambient network is presented below.

Figure 8: Current (a) and Ambient (b) Networks

The main principles of the integrated networks were defined as:
- "Ambient Networks communicate with each other through an open, feature rich, internetworking interface
- The control functions of Ambient Networks form a modular control space with a defined, extensible architecture
- Ambient Networks can operate over any type of connectivity infrastructure
- Ambient Networks support open interfaces for service creation and deployment
- Ambient Networks configure and manage themselves and their relationships"

Another important factor was the network integration on the transport level type of network among technological different type of networks. This process can take two possible paths: a unification of all current networks into one network, or development of high density border elements which will provide the network-to-network translation both in signalization and traffic. Another key element is the development of new technology which will be utilized on the access level to the network. All mentioned ideas call for new network elements development based on an open, adaptive structure.
CHAPTER 3

AMBIENT TECHNOLOGY MAPPING

This chapter presents an overview of the technology used to reach the Ambient Intelligence Environment concepts. The challenges for future technology presented in chapter 3.1 are followed by a short discussion of the possible node technology choices. A detailed overview of chosen technology – FPGA – is provided in the rest of the chapter.

3.1 Concepts and Challenges

The era of uniprocessor core is over [FCA06]. This is a very strong statement but if you hear it from one of the fathers of the RISC processor that means at least the big change is coming. The use of a single processor is reaching its limitation and new ideas arrived. The new basis for prototyping was forced by the increase of the processing requirements per system, bringing the new challenges for processing architecture. There are several possible designs to reach this goal like: multicore, FPGA or ASIC/FPGA hybrids. All of these solutions have one common factor – parallel processing.

3.2 SoC FPGA vs ASIC

The trend introduced by one of FPGA leaders is to gain reprogramming advantages with embedded “Hard-IP” microprocessors in an FPGA device. The result of this hybrid technology is the ability to design a universal System on a Chip (SoC). A
microelectronic design including a combination of system components such as CPU, memory, and other functional blocks in a single chip is classified as a SoC. The main advantage of this architecture is a reuse of designed components such as CPU and RAM for each system design. Another important factor in favor of FPGA boards is the parallelism of task processing. The processor industry is trying to develop a many-core processor but the software on top of it and the cost of development can bury this solution, even the new architectures base on tested, reliable modules.

The current migrations in the technology are to:

- move as many processing tasks as possible away from the processor onto separate DSP cards – picture 10 where the arrows present the task interfaces.
- utilize separate CPUs for separate tasks – picture 11
- utilize multiple cores to load-balance main processes, or offload CPU processes to DSPs within each CPU - picture 12

Figure 9: Hardware Architecture with Separate DSP Banks

Figure 10: Hardware Architecture with Separate CPU for Separate Tasks
The ASIC technology solution is the most expensive solution. The cost of development, with long design and verification process eliminates this solution for many small designs. In return for the costs, the developed ASIC product brings a very good quality and a very strong base for processes running on top of it. The era of multi core processors will allow multiplying the numbers of these processes. From the other side, it is important to realize that the solution of multi core can be easier and faster to achieve using FPGA board! It was proven by the “Research Accelerator for Multi Processors” (RAMP) [RAM] project where the development of the multi processor environment was based on FPGA boards. During this project four solutions were taken into consideration: Multi core processor, cluster of servers, simulation, and the RAMP. While the first three solutions lost based on the cost of development (Multicore processor), the cost of ownership/ power & space (Cluster) and the performance (Simulation) the FPGA took the credits in all metrics in the evaluation. The FPGA was also chosen to fit the design requirements in this project. The detailed information about the FPGA cons and pros are overviewed below.

### 3.3 SoC FPGA

A Field-Programmable Gate Array is a programmable and erasable semiconductor device. The basic component of the FPGA is a logic block consisting of
AND and OR gates with programmable interconnect. The matrix of blocks arbitrarily connected via programmable interconnections allows the designer to synthesize an FPGA hardware structure capable of natively solving any mathematical, combinatorial, or sequential logic function. High-level combinations of embedded functions allow an FPGA to form a complete system. There are several popular producers of FPGA chips; the two most advanced of which are Xilinx[XLW07] and Altera[ALW07]. They both utilized the volatile technology as a programmable basis for their backplane. One low-cost competitor for both of them is Actel. The two tables below present a short parameter comparison of these manufactures [SPB04].

<table>
<thead>
<tr>
<th>Company</th>
<th>Actel</th>
<th>Altera</th>
<th>Xilinx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>FPGA (non-volatile)</td>
<td>FPGA (volatile)</td>
<td>FPGA (volatile)</td>
</tr>
<tr>
<td>Product</td>
<td>AX2000</td>
<td>Stratix EP1S120</td>
<td>XC2V8000</td>
</tr>
<tr>
<td>Memory</td>
<td>339 kb</td>
<td>10 Mb (without reducing logic</td>
<td>Max 3 Mb RAM in 18 kb blocks + 1.5 Mb distributed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>resources)</td>
<td></td>
</tr>
<tr>
<td>Maximum Gates</td>
<td>2 M</td>
<td>5 M</td>
<td>8 M</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>500 MHz</td>
<td>250 MHz (DSP blocks)</td>
<td>420 MHz</td>
</tr>
<tr>
<td>Integration (on-system functionality)</td>
<td>IEEE Std 1149.1† (JTAG)</td>
<td>IEEE Std 1149.1† (JTAG)</td>
<td>IEEE Std 1149.1† (JTAG)</td>
</tr>
<tr>
<td>Voltage</td>
<td>Core 1.5 V, I/O 3.3 V</td>
<td>Core 1.5 V, I/O 3.3 V</td>
<td>Core 1.5 V, I/O 3.3 V</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>Not provided by manufacturer</td>
<td>Industrial -40°C to 100°C</td>
<td>Industrial -40°C to 100°C</td>
</tr>
<tr>
<td>Packaging</td>
<td>896/1152 FBGA</td>
<td>1,923 BGA</td>
<td>Flip Chip, 1152 FBGA</td>
</tr>
<tr>
<td>Size</td>
<td>35 x 35 mm or 31 x 31 mms</td>
<td>45 x 45 mm</td>
<td>3.5 x 3.5 mm</td>
</tr>
</tbody>
</table>

Table 4: FPGA Actel, Altera and Xilinx Parameters Comparison
Table 5: FPGA Actel, Altera and Xilinx Parameters Comparison 2.

All of these solutions present very innovative but also very experienced views in SoC design. The main difference between these two volatile FPGA manufactures (Altera and Xilinx) from the node design point of view is the option for embedded hard-core microprocessors. Only the Xilinx products (Virtex II & IV) include on-die integrated CPUs. From an energy perspective, the Actel non-volatile solution enables very interesting opportunities for meeting low power restrictions. Figures below present diagrams of Xilinx and Altera FPGA dies. The first group of figures presents the Altera
Stratix/Stratix II family contrasted with the Xilinx Virtex II/ Virtex IV family.

Figure 12: FPGA Altera and Xilinx Family Group.

3.4 SoC Bus Technology

In order to support IP Block reuse, it is necessary to use some standard bus architecture to interconnect all components. There are several bus types available for the SoC developer; the most popular are:

- Advanced Microcontroller Bus Architecture AMBA
- WishBone
- CoreConnect

The primary purpose of each of these standards is to allow various parts of a System on the Chip to intercommunicate. The secondary goal behind each standard is to
provide IP cores with technology independence, allowing modules to be migrated across platforms and among different projects. Every bus designer wants his bus to become standard; each new bus begins its product lifetime in a race to determine which standard will become the leader. From one side, the producers with the hardware background (ARM, IBM) promote their vision based on the ASIC standards. From the other side, technology-independent open source standards such as WishBone came into the game.

3.4.1 CoreConnect

CoreConnect is a bus architecture developed by IBM for System on the Chip designs. Like the standards mentioned below, it is designed as a universal standard for connecting reusable cores as peripherals to IBM embedded processors. The standard includes two main bus types: Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB). The difference between these two busses is the speed and width. High performance components such as DRAM and the OPB peripheral bus controller connect to the PLB. Most peripherals are connect to the second tier OPB bus, although the option remains to connect very high-performance peripherals to the PLB bus. In thesis design the architecture utilize the OPB bus because it is natively supported by Xilinx, and it has plenty of bandwidth to reach the project specification. The following figure presents an example PowerPC processor with CoreConnect busses.
3.4.2 AMBA

AMBA is supported only by ARM processors. It encompasses three different 32-128+ bit buses interfaces:

- Advanced High-performance Bus (AHB)
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)

The information for each of bus type can be found at AMBA 2.0 Specifications [AAS07]. The main objective for the ARM-dependent, SoC-focused bus standards is the mobile mark, with the following advantages:

- low power consumptions
- minimal design area
• reusable module design

The diagram below illustrates the AMBA bus structure.

![AMBA Bus Structure Diagram]

Figure 14: The Structure of the AMBA Bus

The AMBA solution is very similar to the CoreConnect architecture but not as sophisticated. It also belongs to the hardware architecture type bus structures.

3.4.3 WishBone

The WishBone standard [OCA02] provides a very flexible approach to interconnect the system elements. It can be a 8, 16, 32 or 64 bits wide, and supports point to point, many to many, and other topologies. The WishBone bus standard focuses more on logical topology, leaving designers a very flexible approach similar to software design. This standard is popular enough that it has become the preferred bus standard of
the OpenCores [OPW07] website, a repository of free IP cores. The diagram below illustrates the bus structure.

![Diagram of WishBone Bus Structure]

Figure 15: The Structure of WishBone Bus

3.4.4 MicroBlaze Local Memory Bus and Fast Simplex Link (Xilinx)

MicroBlaze is a soft-core processor available for Xilinx FPGAs. This processor supports two bus types natively; Local Memory Bus (LMB) and Fast Simplex Link (FSL). The MicroBlaze processor will be discussed in further detail later on, because it is an option on all Xilinx FPGA’s, including those that do not include hard CPU cores. The LMB is a 32-bit local bus that is used for low-latency single-cycle access to on-chip Block RAM.

The MicroBlaze supports eight input-only FSL and eight output-only FSL connections. Each FSL link is a 32-bit wide bus, which provides easy connection to MATLAB® IP Blocks.
3.5 SoC Processor

In the project environment, based on the Virtex FPGA board and EDK application (Xilinx application for Virtex boards development) two processors are available:

- PowerPC — a dedicated “hard core” processor standard on Virtex-II and optional Virtex-IV FPGAs.
- MicroBlaze — a soft-core processor that can be implemented on any sizable Xilinx FPGA.

Both processors are based on the Reduced Instruction Set Computer (RISC) microprocessor architecture. The PowerPC was commercially developed by the Apple-IBM-Motorola collaboration with 32-bit data-side and 64-bit instruction-side On-Chip Memory. The MicroBlaze CPU is a Xilinx design based on the DLX architecture soft-logic processor with a 32-bit Local Memory Bus (LMB), from academic roots [UCP96].

The block diagrams below show how each processor fits into an embedded system; one commonality available in either design is the OPB bus:

Figure 16: Block Diagram of Hard-Core CPU Embedded System (Courtesy of Xilinx)
Figure 17: Block Diagram of Soft-Core CPU Embedded System (Courtesy of Xilinx)
CHAPTER 4

AMBIENT INTELLIGENCE NODE : ARCHITECTURE

OVERVIEW

This chapter explains the architectural basis of the project. A brief overview of the components involved in creating the Node is presented: a Base System is presented in chapter 4.1 and the DSP - chapter 4.2. It also presents the short overview of the AIN network architecture - chapter 4.3.

4.1 AIN Node System Architecture

The AIN design is based on the FPGA Virtex board utilizing reusable IP building blocks. The base IP blocks (UART, Ethernet MAC, DDR) for the Node architecture are already included as plug-and-play modules in the EDK environment with pre-defined configurations compatible with the development board. The whole project, from the architectural perspective, was divided into the following three steps:

- Implementation of the Base System node in the EDK environment.
- Building the Operating System on top of the base system node.
- Development of the DSP module and its integration with the system using the OPB bus connection.
4.1.1 AIN Node Base System Overview

The Base System for the Node was designed to meet the ambient milliWatt node specification. The main specifications of the developed node were:

- **Miniaturization:** A design and implementation process of a microelectronic system starting with an algorithm (MATLAB®), hardware design using hardware description language (VHDL), prototype implementation in FPGA

- **Interoperability:** The ability to provide an interface for additional development.

- **Reconfiguration:** Adjustment of the internal structure of the ambient intelligent node, possibly on line.

- **Scalability:** The ability to offer the above at any level and scale including the AIN node level and the AIN network level.

The main XUP Virtex-II Pro board modules integrated into the Base System were:

- Environment and Communication Interfaces – Ethernet MAC module enabling TCP/IP network communication
- CPU – embedded on the Virtex FPGA, PowerPC processor
- Memory – the DRAM memory with additional Non-Volatile (Boot Flash) memory

The functional block diagram of the node is presented in the picture 18.
Detailed information about development of the system base is presented in the Implementation chapter below. The critical function of the function of the node is its communication capabilities. This enables an autonomous node to become a part of the overall ambient network within the ambient environment. The structure of the AIN environment with identified layers is presented in the picture below.

The elements of this environment named NODE A are the scope of this project development.
4.1.2 AIN Node Operation System Overview

The purpose of the implementation of an operating system on top of the embedded processor was to increase the reusability of the base system. It allows the designers to move the application development to a higher level of abstraction. It also brings many already completed out-of-the-box components which can be reusable. There are several embeddable OS's available for SoC designers:

- MontaVist Linux
- μClinux
- VxWORKS
- Xilinx OS

MontaVista Linux 2.4 was selected for this design together with its networking components:

- Network Hardware Driver
- TCP/IP stack

Another major function of the Operating System (OS) on top of the base system communication capabilities is to provide a software base for a software application built on top of it.

4.2 AIN Node DSP Overview

The new approach in the hardware design is a modular approach. The processor architects in new multicore designs are trying to reuse as much as they can of the already used, proven, reliable modules, but these modules are not available for all. From the other side the FPGA industry is trying to develop its own library of IP cores. The
programmable characteristic and very flexible approach of FPGA solution together with a huge efficiency and short time to market encourage the FPGA vendors, other companies and educational institution [ERF05] to design and optimize IP cores. Each of the leading vendors entertains their IP cores trying to bring the potential developers to their solutions sharing its boards capabilities:


The beauty of this FPGA modular approach is a global access to complex solution available to individual designer. The million dollar pricing cost doors of embedded microelectronic fabrication, which were closed for individual and small companies, had been opened. Unfortunately it is still a very relatively small market and the number of the designs ready to use is limited and the quality of the projects is still weak. Another Achilles' heel of this crawling industry is that there is still missing one unified bus standard for the modules interconnect.

The good news for these potential designers who want to "do-it-yourself" is that the FPGA vendors and software companies are creating easy-to-use development kits. DSP block design can be achieved in many ways. From one side, the FPGA industry leaders provide easy-to-use development and integration applications. Unfortunately this approach has one small hidden requirement – the development application fits only the application provider FPGAs. Having this problem the universal application developers like MATLAB® and Simulink® came into play providing the vendor agnostic design prototyping tools. But universal hardware components for the designer are still in the
development stage and limited. It is still easier to use the FPGA vendor’s application like Xilinx’s System Generator embedded in the MATLAB® and Simulink® environment. This joint solution provides excellent DSP design capability as well as synthesis and verification environment and was used in the development of this project.

The design process in Simulink® software and System Generator utilizes the Simulink® base for DSP development and simulation. System Generator provides a toolset for the Virtex FPGA including the predestined hardware modules, an engine for system modeling, and automatic HDL code generation. A significant added feature of this environment is the possibility of the project verification not only in the software simulation but also in hardware-in-the-loop testing. A project created and optimized using the software simulation is translated to the target specific hardware code and implemented on the target board. The validation of hardware and software code can be processed simultaneously, which enables fast result comparison and project verification. The same environment provides additional features like the resource estimator based on the target board and the flexibility to import generic HDL code into the project. The verified design can be implemented on the board or modified to meet a peripheral interface requirement. This project includes the basic DSP development and the major task was to enable the DSP to the OPB peripheral deployment.
CHAPTER 5

AMBIENT INTELLIGENCE NODE PROTOTYPING

This chapter will present the steps to configure a simple node design using the methodology and tools previously described. The first step in the overview is a list of tools in the development environment. This list is not intended to scare potential designers away from this design methodology; the main thrust of this thesis is to provide a neatly packaged, well integrated environment for them. Like most embedded designs developed in industry, this project required expertise in too many specialties to be successfully completed by one person. The resulting design environment, however, can be used to by any systems engineer with elementary C programming experience. Note that all freely downloadable content (except for i386 RedHat 6.0) is packaged in the GNATDVD [ADP08] distribution.

Components of the Ambient Intelligence Node Design Environment are:

- MATLAB® 2006a R14.1, R14.2, or R14.3 with Signal Processing Blockset™ (Commercial Product)
- Xilinx v8.1 EDK (Commercial Product)
- Xilinx v8.1 ISE Foundation with latest service pack (Free download)
- Xilinx v8.1 System Generator for DSP – integrated within the Simulink® software
- sysgen2opb.m wrapper script from WARP/Rice University (Free download)
- EDK OPB Export Tool (Free download)
- MontaVista Linux 2.4 (Free download)
One viewpoint on system design is top-down modeling, with a focus shifting from software to FPGA. This is because the FPGA industry has benefited from the same technological advances that advance the entire semiconductor industry at a breakneck pace. The resulting increase of design space in each FPGA allows (and from a marketing perspective requires) the creation of more and more sophisticated functions in each product. As a result, said designs become too complicated for a single person or even a small team to design in a reasonable time period. One solution is the concept of reusable blocks reconnected and recycled through various projects. These modules with predefined functions and interfaces can be organized in a systematic way for quick and easy re-use.

While there has been a lot of progress over the last ten years in the area of design reuse, it remains an elusive genie that no company has yet managed to bottle and sell.
Aiming to fix this technological gap, more advanced design tools appeared on the market. Xilinx and The MathWorks provided UNH with use of their Embedded Development Kit (EDK), Simulink® software together with a System Generator – along with quite a bit of technical support in getting these two tools to work together. The next step was to research the resources to craft a sample AIN Node design using as mainly off-the-shelf IP Cores. The core base for bus interconnect was Xilinx and On-Chip Peripheral Bus (OPB) developed by IBM.

As an arch has its keystone, EDK is the central and most important tool in the AIN Node design process. The IP Cores used to build the AIN system have hardware and software components, and both are managed in the relatively orderly EDK graphical environment. It is good to remember that each of these modules was created by a design team with reusability in mind. In this chapter there will be a brief introduction to the design methodologies used to develop these component parts.

The process of development of this block was not focused on speed or power, but the design flexibility. The option to include one or more DSP cores synthesized as hardware in the FPGA, gives this node design a much broader range of applications than a standard embedded computer. The efficiency and power in the design is inherent to the DSP IP cores, but a modular means of interconnecting them is required in order to meet time-to-market demands. While hardware connections among the DSP core(s) and the network port would be more efficient than software, this type of interconnect carries the disadvantage that it would have to be completely redesigned for each system. Software data-routing is thus the clear choice in reusable node design.
5.1 Node Design

The details of the node design span numerous design disciplines. The baseline configuration of FPGA hardware, Linux Kernel, and OS file system needed for a node without a MATLAB® DSP Peripheral are described in a recent publication titled, “Porting MontaVista Linux to the XUP Virtex-II Pro Development Board” [PMV06]. It was very difficult to replicate this work due to the complexity of the tools, so a different approach was taken to packaging the IP in this thesis. The FPGA portion of the Sample Node IP core is submitted as a pre-packaged, already working EDK project – see appendixes. The software portion of this sample IP core is submitted as both source and object code packaged along with a complete, configured cross-compiler environment. The hardware components block diagram is presented in the picture below.

![Node Hardware Components Block Diagram](image_url)

Figure 20: Node Hardware Components Block Diagram
The entire development and implementation process was divided into three major steps (Flowchart is presented below):

1. System Base development based on the Virtex/ PowerPC core and MontaVist Linux
2. IP block and software application creation
3. Integration of all elements within the System Base.

Both the first and the last step involved completing the System on the Chip design with the synthesis, place and route and verification.

Figure 21: Flowchart for AIN Implementation

The baseline elements of the Node:
5.2 Node Software Detail

Several automation tools were developed to streamline the process of getting Linux onto an XUPV2P board. These are all run from a Linux PC that is set up with the cross-compiler environment included in the GNATDVD [ADP08]. The first tool developed specifically for AIN Node design during the Node development was for embedding MAC addresses into the Linux Kernel. Unfortunately, the XUPV2P board does not come with a hardware MAC address; this must instead be compiled into the kernel in order for the XUP board to talk on a standard Ethernet network. The “setMAC.tcl [MAC ADDRESS]” script in the Linux Compile Server’s xupv2p directory updates the network driver to use a hard-coded value specified by the operator in the command-line parameter [MAC ADDRESS].

The kernel contains all of the hardware drivers for the board, but EDK was designed with this requirement in mind. Any time the hardware is reconfigured in EDK,
the contents of the BSP directory must be copied to the Linux server's `montaVista_2_4_devel` directory. EDK can be instructed to use a place this directory on a network file share, but we found it more practical to do this step manually, in order to prevent unexpected changes to the kernel source code. In order to put these changes into effect, the kernel must be recompiled. This can be run from the Linux Compile Server's `xupv2p` directory by calling the `updatekernel.sh` script. The finished kernel file must then be copied from "xupv2p\zImage.elf" on the Linux machine to "xupv2p\montaVista_ELF\zImage.elf" on the PC running EDK. Once again, proper use of network file shares can make this step transparent to the user, but a manual process prevents unintentional changes. While zImage.elf is the actual binary executable, it cannot run on the FPGA in this form. A script called `genACE` is run from the EDK command line to compile the kernel and the latest EDK bitstream into "xupv2p\montaVista_ACE\system.ACE". This file, once copied to the DOS partition on the CompactFlash drive, is used by the systemACE chip on the xupv2p board to load the Virtex FPGA bitstream and boot the embedded PowerPC Processor cores to the enclosed Linux kernel. The Xilinx figure below shows the actual file naming conventions used in this process:
5.3 DSP Block Design Detail

The steps in the MATLAB® DSP Peripheral design process include:

- DSP schematics design with Simulation
- Hardware model generation
- Project verification (Hardware in the Loop testing), resource estimation/tune-up
- OPB-compliant peripheral generation
- Integration of MATLAB® DSP Peripheral into EDK project

It is worth pointing out that the first step in this process is a universal design not dependent on the targeted board. Of course, the toolset provided here does add the requirements that the target hardware includes a Xilinx FPGA and embedded CPU. The first steps require knowledge of Digital Control Systems. For simplicity, the GNAT uses implements a 32-bit Multiplier DSP block.
The most important parts in the Simulink® Xilinx block-set are the “Gateway In” and “Gateway Out” blocks. These specify the bounds of the design that fall within the realm of the Xilinx System Generator. An extensive set of blocks developed by Xilinx can be used inside these bounds and implemented into the FPGA; analog environmental design tools like spectrum scopes and signal sources are ordinary Simulink® blocks that cannot be compiled into the FPGA. Once a model is designed, simulated, and operating satisfactorily in Simulink® software, the next step is to translate this model into a hardware implementation. This process requires several steps like synthesis, place & route, etc. These steps are managed and performed by Xilinx ISE, and triggered by the System Generator block. In order to optimize the output, it is important at this stage of implementation to define the hardware target. MATLAB® software ensures timely selection of a compile target by the structure of the System Generator block controls. Before System Generator will compile, all relevant fields of the following dialog must be filled in:

![System Generator dialog](image)

Figure 24: System Generator
The next logical step after software simulation is Hardware Co-Simulation. If a Board Support Package (BSP) is not loaded, then a new board can be defined by selecting → Co-Simulation → New Compilation Target. Several parameters need to be provided by the board designer:

- Core clock frequency in MHz
- Pin Allocation
- The name of the board and type
- Boundary Scan Position

The JTAG options are automatically detected from the board. Note that the board must be connected to and detected by the PC via “Xilinx JTAG Debug” USB cable before the MATLAB© software is opened. With all of these parameters defined, hardware compilation is triggered from this System Generator block menu with single click of the Generate button as shown below:

![Figure 25: System Generator](image)

The compilation steps include:

- Synthesis Options Summary
Design Verification Test (DVT) is very easy in this environment because the software and hardware implementations of the same model can be against the same MATLAB® source, and the outputs can be compared on MATLAB® scopes and spectrum analyzers.
Another very useful feature of the System Generator block is the Resource Estimator, which determines the FPGA physical size required by the model.

When the design is completed and verified, the next step is to wrap the model in an OPB peripheral structure. In order to connect the IP Block to the GNAT CPU, the block must be an OPB-bus compatible peripheral. While Xilinx’s OPB Export Tool provides fine-grain control of the OPB Bus Interface from MATLAB® software, it requires the additional purchase of The MathWorks Stateflow® software. In this design we used a MATLAB® script provided by the Rice University WARP Project to meet this
OPB structure needs, called sysgen2opb.m [WAR07]. This script is used to further abstract the MATLAB® DSP Block before it is processed by Xilinx’s OPB Export tool, eliminating both fine-grain control of the OPB Bus Interface and the need for the Stateflow® software. When the new peripheral appears in the EDK project the last hardware steps are to attach the MATLAB® DSP Peripheral to the OPB bus and generate addresses.

5.4 DSP Connectivity Detail

Several cores were found freely available from The MathWorks, Xilinx, and other sources, that come with an OPB interface. The OPB interface in the Xilinx EDK toolset is based on an IBM processor bus standard, and allows IP cores to present themselves as hardware peripherals to the processor (whether that be a hard-core PPC or a soft-core MicroBlaze). Each of these peripherals presents itself primarily as a range of addresses in memory, and includes sample application code (illustrated in the picture below).

Figure 28: Peripherals Memory Address Ranges.

The key aspect of embedded system design that makes an AIN node most useful is network connectivity to the DSP core(s). In the Node the cores are managed by the very same application that handles network communications, the gnatserver – a SOAP server that handles data-structure transmission over the internet. This application-level
code does not quite match up with the sample code provided with each IP core, however
the cores come with Real-Time Operating System (RTOS) Applications. These were
ported into the *gnatserver* application using the methods described in an article titled,
"Porting RTOS Device Drivers to Embedded Linux" [PRD04]. Unlike RTOS
environments, Linux has a memory manager, which abstracts physical memory into
multiple virtual memory address ranges. The memory manager resides in the kernel, and
maintains *exclusive* access to all physical addressing ranges. The Linux OS is
architected this way for several reasons, including security and scalability. The physical
addresses of the MATLAB® DSP Peripheral in the system enumerated “xparameters.h”
are mapped into virtual addresses in the space of the calling application by the Linux
“*mmap*” system call. Register read and write functions required to access OPB
peripherals are provided in Xilinx’ sample memory test program for the XUP board. The
relevant source files have been copied and referenced without modification into the
*gnatserver* source directory.

As is true at every other level of this system’s architecture, there are trade-offs
involved in the selection of this peripheral management structure. The application-level
support for the MATLAB® DSP Peripheral provides the developer with the greatest
possible flexibility. The kernel does not need to be recompiled, and packed into a new
ACE file to support changes in the way DSP communication is handled. Said changes
can instead be made in one C file and compiled into the top-level application. The
drawback to application-level hardware support is that it precludes the use of hardware
interrupts. The Linux architecture does not provide interrupt access to the user
application space; instead all interrupt drivers must be installed in the kernel. However,
the selected MATLAB® OPB peripheral compiler currently does not support interrupts, so a kernel-level peripheral driver is not necessary at this time.

The MATLAB® peripheral continuously operates on the input registers, independent of the application software, operating system, or CPU. Input registers are processed into a result in a fixed amount of real time, regardless of the operands – this is a feature of the MATLAB® DSP Peripheral. In this example case, the DSP is faster than the software code, so no software delays are necessary. The software reads the output register immediately after writing the input registers, and the result is ready. For longer-running or continuous processes functions, control/status registers and/or pipelining would be required to achieve desired results efficiently.
CHAPTER 6

RESULTS AND DEVELOPMENT ROADMAP

This chapter summarizes the thesis and presents the implementation bottle necks as well as directions for possible continuation of the project. The conclusion of the thesis regarding the design is presented in section 6.1 Section 6.2 contains list of identified gaps in the ambient environment as well as the future directions in the new hardware element development.

3.1 Results

The main objective of this thesis was to implement the Ambient Intelligence Node in the SOC technology using modular approach. The concept of creation of an open base node for the future development was realized. The modular approach in the System on the Chip project was presented together with a simple IP block development and integration. To actualize this work many steps were accomplished, listing the most important:

- Research of the ambient theory and the technology bases for the open design
- Base system implementation in the EDK environment
- OS implementation on top of the system base.
- The DSP block development and its integration into the main project as a OPB peripheral
All of the steps in the design were accomplished but with one significant restriction - all design steps were realized using one vendor specific environment – the Xilinx environment. Most simple explanation for using this vendor solution is that the product family made available by this vendor including the demonstration board and the development framework, is the most sophisticated solution on the market.

Some other major issues related to the development process were named during the project. First of all, there is no existing comprehensive application or a development kit that can be used for the SoC creation. The development of the SoC is relatively new discipline in hardware engineering which relies on small amounts of knowledge in a variety of topics, including development kits and IP cores. It requires a unique approach that depends on the board and the function of the system. In order to develop the system presented in this thesis, tools from different vendors had to be used.

A similar issue concerns design elements of the system. Even as the library of the DSP modules increases, the integration process varies depending on the processor type used in the architecture. Intellectual Property cores, provided by different source than the board vendors in the many cases are restrict by bus type. The variety of development kits and standards causes interoperability problems.

One of the goals of this project was to go through the SoC prototypic path, get the hands on experience and provide a universal, open solution for students and engineers to use as a base system for future applications.

3.2 Development Roadmap

There are several already defined issues which need to be addressed:
Technology standardization

The lack of standardization in the ambient technology is proportional to the number of the ambient definitions. Several ambient technology development labs created their own approaches in ambient concept interpretation. If the ambient environment is to become a global technology, interoperability interfaces need to be created.

Power

One big requirement from the system architecture is missing in this project—the power. The future function of the milliWatt node, in many cases, will be a mobile component of the ambient environment. The power issue needs to be addresses in real market products.

Communication of Sensor’s Network

The lack of bandwidth space within the communication link between microwatt elements brings a problem with signalization efficiency. The existing TCP/IP protocol approach can be “oversized” to fit into this narrow bandwidth environment.

Cross network communication.

The increase in communication networks introduces a requirement for cross-network capabilities. When two networks need to communicate with each other, cross networking interfaces and procedures need to be defined. Some ideas for this were
introduces by European designers [SON07]. These procedures should include Quality of Service, security, accounting, and management processes.
LIST OF REFERENCES


[AIP05] P. Maes. MAS 961- Ambient Intelligence, 2005,


[EIS04] Europe’s Information Society Thematic Portal, 2004,


[IPB05] Internet Pioneers Pawel Baran. Ibiblio, 2005,
http://www.ibiblio.org/pioneers/baran.html

[ANP06] Henrik Abramowicz, Ambient Networks Phase 2, D1-A1 Ambient Networks Project Description and Dissemination Plan, March 2006.


[AAS07] ARM, AMBA 2 Specification, 2007,


http://www.csee.umbc.edu/courses/undergraduate/411/spring96/dlx.html.


[WAR07] RICE University, WARP OPB Export Tool, 2007,

APPENDICES

APPENDIX A

MATLAB® DSP model (CD)

APPENDIX B

EDK Node Project (CD)

APPENDIX C

Operation System Development Components (CD)