

Assertion based debugging and monitoring of Distributed systems

Vivek Narayanan Kallankara



Department of Electrical and Computer Engineering
McGill University
Montreal, Canada

August 2010

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment
of the requirements for the degree of Master of Engineering.

© 2010 Vivek Narayanan Kallankara

Abstract

This thesis presents the steps taken in the design of a portable sleep study system. The kit enables a wireless sleep study of patients at hospitals and also covers the design issues involved for creating a PCB for the same. The intended users are researchers and students interested in a flexible and robust platform for sleep studies. A novel method to increase the monitoring and debug capabilities of wireless sensor systems is also addressed. This thesis proposes the use of assertion based methodologies for the robust, flexible design of such embedded systems that can be used to increase the debug ability of these remote systems. The assertion based debug methodology has also been used in Network-on-a-chip to increase the visibility and error tolerance of these systems.

Résumé

Ce mémoire de maîtrise présente les étapes de la conception d'un système portable conçu pour l'étude du sommeil. Ce système permet une étude du sommeil chez des patients dans un contexte hospitalier et fonctionne à l'aide d'un lien sans-fil. Les éléments de la conception sont présentés, ainsi que les contraintes liées à la fabrication du circuit imprimé. Les usagers potentiels seraient des chercheurs ou des étudiants intéressés par une plate-forme flexible et robuste leur permettant d'effectuer des études sur le sommeil. Une méthode originale pour augmenter la capacité de supervision ainsi que les capacités de débogage du réseau de capteur sans-fil est aussi présentée. Finalement, ce mémoire explique l'utilisation d'une méthodologie de conception basée sur l'utilisation d'assertions pour la conception de systèmes embarqués robustes et flexible, permettant un débogage à distance. Cette même méthodologie de débogage par assertions est aussi utilisée dans un contexte de réseau sur puce pour augmenter la visibilité et la tolérance du système aux erreurs.

Acknowledgement

First I would like to thank my parents for their support and motivation throughout my studies. Secondly, I would like to thank my supervisors Dr. Zeljko Zilic and Dr. Katarzyna Radaecka for their support and advice during my Master's degree program. Their commitment and insightful advice helped me to bring out the best in every work.

I thank my friend and colleague M.H. Neishaburi for his contribution and insightful discussions on our several collaborative works. I thank Adam Hart for his contribution and dedication on our joint project for developing the sleep apnea kit. I thank Jean Samuel for his sound advice while developing the PCB and for other collaborations. I wish to thank my colleagues Bojan Mihajlovic and Jason Tong who have always assisted me in times of need. I am grateful towards my supervisors for their financial assistance during my degree. Finally, I would like to thank all the students in the MACS (IML) lab for their stimulating conversations and enjoyable company during my period of study at McGill.

Table of contents

Chapter 1.....	1
1.1 Motivation	1
1.1.1 Overview.....	1
1.1.2 Wearable Health Monitors.....	2
1.1.3 Distributed Systems	3
1.1.4 Wireless Sensor Network.....	4
1.1.5 Network-on-a-chip.....	5
Chapter 2.....	7
2.1 Wireless Sensor Networks	7
2.2 Evaluation of wireless kits	9
2.3 ECOMOTE kit evaluation.....	9
2.3.1 Overview.....	9
2.3.2 Reading accelerometer values through serial port.....	10
2.3.3 Host program to read serial data	10
2.3.4 Limitations	10
2.3.5 Target Work group.....	11
2.3.6 Possible applications.....	11
2.4 SHIMMER kit evaluation	12
2.4.1 Overview.....	12
2.4.2 Functionalities available on Shimmer 1.....	13
2.4.3 Next Generation - Shimmer 2:.....	14
2.4.4 BioMobius Accelerometer Example.....	14

2.4.5	Limitations	15
2.4.6	Target Work group and application	15
2.5	Vital Sign Monitoring	16
2.6	Sleep Apnea.....	17
2.7	Assertions	21
2.7.1	Verification with assertions	22
2.8	Network on a chip	23
Chapter 3	27
3.1	Introduction	27
3.2	Pad Design.....	28
3.3	Interfacing with AD7746	30
3.4	Bread boarding	33
3.5	I ² C isolation.....	33
3.6	Additional features	33
3.7	Programming the dsPIC30	35
3.8	PCB design	35
3.8.1	Ground layer for pad signal layer	36
3.8.2	Placement and routing of components on board.....	37
3.9	Conclusion.....	37
Chapter 4	38
4.1	Sensor Model.....	39
4.1.1	Transaction Model	39
4.1.2	Interface	40
4.1.3	Packet Structure	41

4.1.4	Protocol	41
4.1.5	Sensor Module	42
4.1.6	Communication.....	42
4.2	Assertions	43
4.3	Results	44
4.4	Conclusions	45
Chapter 5	46
5.1	Delta encoding.....	47
5.2	Sensor-Lempel–Ziv–Welch (SLZW).....	48
5.3	Compression results	51
Chapter 6	53
6.1	Design approach.....	54
6.1.1	Communication:.....	57
6.1.2	Routing.....	57
6.1.3	Operation.....	58
6.1.4	NoC simulation performance	59
6.2	Back-flow Assertion.....	60
6.3	Acknowledge fail	62
6.4	Results and Conclusion	64
Chapter 7	66
7.1	Summary	66
7.2	Future Work	66
Chapter 8	68
8.1	Sleep apnea detection	68

8.1.1	Screen shots of schematic:	68
8.1.2	Screenshot of PCB	71
8.2	Ecomote Evaluation	71
8.2.1	Client Program	71
8.2.2	Host python program	73
8.3	Shimmer Evaluation	74
8.3.1	EyesWeb Accelerometer patch	74
8.3.2	Biomobius GUI for Accelerometer patch	75

List of Figures

General architecture of WSN.....	8
General setup of wireless health monitoring	16
A pediatric patient prepared for a polysomnogram	19
Verification with assertion.....	22
A 3×3 NoC where the cores have been arranged in a two-dimensional grid	24
Pad design	28
Patient lying on bed for sleep apnea detection.....	29
Liquid detection	30
AD7746 Pin configuration [32]	31
Register configuration for AD7746	32
Schematic of the system level design	34
Transaction model of sensor network	40
Our proposed environment	42
Breath Assertion Description.....	43
Flow chart of LZW compression	49
Example of LZW compression	50
NoC simulation model	55
Switch Architecture	56
Core architecture.....	56
Routing policy- Switch R[2,2] is faulty and hence XY-Routing will cause the packet to be stuck. YX-Routing will enable the packet to reach its destination	58
Back-flow control using assertions.....	61
Acknowledge failure	64
Snapshot of assertions fired	65
Screenshot of Power module	68
Screenshot of I ² C isolation.....	69
Screenshot of microcontroller units	69
Screenshot of Blood Leakage Detection module.....	70

Screen shot of PCB	71
Scrrenshot of accelerometer patch	74
Screenshot of Biomobius GUI	75

List of Tables

1. Number of Transactions for different simulation time periods	44
2. List of Signals and sample rates.....	47
3. Data size consideration for SLZW with 512 entry dictionary	50
4. Compression percentage for dataset 1	51
5. Compression percentage for dataset 2	51
6. Simulation model results for Data buffer size 4	59
7. Simulation model results for Data buffer size 5	60

List of Acronyms

ABV	assertion-based verification	2
ADC	analog-to-digital converter	10
CDC	capacitance to digital converter	27
CPAP	continuous positive airway pressure	20
CPU	central processing unit	3
CUD	circuit under debug	2
DMA	direct memory access	13
ECG	electrocardiography	13
ESL	electronic system level	4
FIFO	first in, first out	5
FPGA	field programmable gate arrays	1
GPIO	general purpose input/output	8
I2C	Inter-Integrated Circuit	8
IRI	inter-ring interface	25
ISM	industrial, scientific or medical	7
LZW	Lempel–Ziv–Welch	48
NoC	network-on-chip	5
OSA	obstructive sleep apnea	18
PCB	printed circuit board	6
PSG	polysomnography	18
PSL	property specification language	2
RI	ring interfaces	25
ROM	read only memory	1
RTL	register transfer level	4
SDCC	small device C compiler	10
SoC	system-on-a-chip	23
SPI	serial peripheral interface	8
SVA	systemverilog assertions	2
TLM	transaction level model	4
TMD	testing, monitoring and debugging	25
UART	universal asynchronous receiver/transmitter	8
UWB	ultrawide band	7
VLSI	very large scale integration	7
WSN	wireless sensor network	4

Chapter 1.

Introduction

1.1 Motivation

1.1.1 Overview

Embedded systems are special purpose computer systems to control or support the operation of a larger system. It performs specific pre-defined tasks and is expected to operate with no or little human interaction with its on board sensors and actuators. Embedded systems consist of hardware components and related software. The hardware complexity varies from having onboard FPGAs to simple micro-controllers to control the sensors and actuators. These devices usually have small ROM memories to store data and the software. Simple systems usually contain a single program running in a loop and responding to various inputs. In more complex cases, there might be a real time operating system to perform tasks like scheduling, synchronization, resource management etc.

Verification of temporal and functional behavior is important for high quality design of embedded systems. But in order to improve dependability, it is also important to consider the faulty management and safety measures in the early design phases. Hence for safety critical applications, common commercial off-the shelf control computers are usually unsuitable.

Moreover, bug-free design is not guaranteed by the existing verification techniques. It is now required to identify any bug as soon as the first running system becomes available [10]. It has been observed that close to 50% of the total development cycle for a new product is spent on validating the system behavior after the availability of the first silicon

[11]. Assertion-Based Verification (ABV) is acknowledged as being the instrumental and efficient pre-silicon verification technique. Armed with temporal logic and extended regular expressions, the Property Specification Language (PSL) and the SystemVerilog Assertions (SVA) can be used as languages to describe the expected behavior of a design. Resorting to the assertion modules inside the Circuit under Debug (CUD), any deviations from expected behavior will be captured; thus designers can increase both the observability and controllability of corner cases in their design which are otherwise hard to observe.

This thesis proposes the use of assertion based methodologies for the robust, flexible design of such embedded/distributed systems that can be used to increase the debug ability of these remote systems.

1.1.2 Wearable Health Monitors

There has been an increased interest in developing wearable health monitoring devices for vital sign monitoring and polysomnography. Apart from being more comfortable for patients, wearable sensors are also a cost-efficient solution in the hospital and home setting. This has made real-time patient monitoring possible in the home, work or hospital environment. Wearable health sensing and monitoring devices are now being developed for a variety of applications such as personal health, rehabilitation, and early disease detection [19].

Unobtrusive, wearable sensors mine data from the patients and report them automatically to the healthcare provider, reducing the cost and inconvenience of regular visits to the physician. Hence these devices help in providing timely healthcare to a larger population. The main activity performed by these nodes is to sample the readings and send them to a base station where the data can be analyzed or further forwarded to a central server for processing. In order to implement these systems, a number of practical obstacles must be overcome. From a patient's perspective, the device must be non-intrusive, comfortable to wear and have a user-friendly interface. From the designer's perspective, they should have efficient power consumption; have very low failure rate and high accuracy. However, we face a new set of problems when these devices are deployed on the field.

The issues related to wireless sensor network are applicable even to portable health monitoring sensors. However, most previous studies/devices related to health monitoring fail to address this issue [25][26]. Given that the device monitors health, we believe that the debug ability of these devices is of paramount importance. Nevertheless, health monitoring applications present some issues of their own. The interference and obstructions in communication faced by these sensors will be higher due to their in-building operation, which has more prevalent multi-path interference. The rate of data collection in these applications is higher than that in typical environment sensing applications. Hence efficient communication and quick response in emergency situations will be required [17].

From the above discussion, it is clear that the correct operation and the reliability of wireless health sensors are difficult issues that have not been addressed successfully by current methodologies.

1.1.3 Distributed Systems

A distributed system is defined as a group of computers doing something together [28]. They are multiple computers with individual CPU and memory with some common interconnection to enable them to communicate to each other and also maintain a shared state. This requires correct coordination among the various computers involved. Such systems could run into a variety of issues such as

- Independent failure – The system should keep functioning even if a few individuals have failed
- Unreliable communication- The communication between the computers cannot be expected to be perfect at all times. Messages may be lost or distorted.
- Insecure Communication- The channel of communication may not be secure and open to eavesdropping.
- Costly communication- The communication channel/interconnection will have lower bandwidth, higher latency and higher communication cost when compared to internal operations

For our studies of distributed systems, we consider the case of Wireless Sensor Networks and Network-on-a-chip. Even though the above mentioned issues of distributed systems are common to both these systems, they operate differently and hence require different approaches to solve their issues. We will consider these two systems separately in the chapters to follow.

1.1.4 **Wireless Sensor Network**

In the realm of wireless sensor network (WSN), it has been observed that many sensors fail to meet application requirements when deployed [14]. This may be due to the poor or hostile channel conditions, as well as hardware problems, software bugs or networking problems. These problems may not be encountered before deployment as the system behavior heavily depends on the environment. Hence the sensor networks have to be inspected and debugged on site. Resource limitations of these networks and even the environment might make in-situ inspection a hard problem. The fundamental constraint during the design of sensor networks is the energy constraint. This limits the data sensing rates, link bandwidth, node size and weight. In most cases, the radio on board the wireless sensor is the main energy consumer. The sensors have to be energy efficient and hence cannot send a lot of information about its status. Every bit of extra information costs extra energy. However without sufficient information the system cannot be debugged [1].

In recent years, the increasing complexity of digital systems along with constantly evolving specifications has forced the creation of flexible designs that can be changed rapidly. Moreover, the hardware manufacturing cycles are time consuming and expensive when compared to software based implementation [5]. Hence large digital system designs are being modeled using Electronic System level (ESL) modeling. System level design can be at different desired levels spanning from Register Transfer Level (RTL) to Transaction Level Models (TLM). At the RTL, emphasis is on the signals whereas in a TLM the details of the transaction are separated from the details of the functional units. These days, most system level designs have extensive usage of TLM [2][3]. The main advantage of this approach is that we need to model the level of data that we are interested in. Communication mechanisms like buses are modeled as abstract channels

and presented to the modules. The actual data transfer from one module to another is emphasized over the protocol that is being used [4].

After developing a high-level model, the designers face a second challenge of gathering information from their model. Most designers have to scrutinize the simulation traces and write scripts to scan these messages for simple error flagging. Hence, there is a need for a concise way to specify the desired functional and performance properties, and an efficient way to check the properties of the model [15].

1.1.5 Network-on-a-chip

In the context of distributed systems such as NoCs, it is also very useful to monitor the various data paths by including monitors that can count certain events which are meaningful to the hardware design. For example, counters for certain types of transactions (e.g. aborted flit transfers, back pressure activation, FIFO threshold levels) offer advanced on-chip tracing of transactions and can be instrumental in pinpointing corner case problems. We present a method to efficiently incorporate a large number of such monitors by leveraging high-level cover statements derived from the verification effort.

By leveraging the hardware assertion checkers built-in the circuit, the designers can obtain a comprehensive view of the system self-checking status, in real time, while reducing the design effort by using tools that automate creation and integration of the hardware checkers.

The design of a NoC architecture benefits from the use of an ABV methodology due to its complexity and high verification coverage requirements. The verification process clearly benefits from this approach by becoming more sensitive to incorrect assumptions and to incompatible protocols between modules in the design [36]. Furthermore, it was shown to considerably reducing debugging time.

1.2 Objective

In this masters' thesis we first cover the design aspects involved in developing a custom printed circuit board (PCB) for researchers and use in hospitals. The board aims to have a wire-free setup which would enable in better sleep studies. The design also aims to include safety measures and expansion units which would widen its the application and usage.

The thesis explores assertion based methodologies to increase the visibility and debug capabilities of wireless sensor networks. This requires the creation of a transaction level model of the wireless system that closely matches the functioning of the sensors. The wireless sensors are modeled on biomedical sensors. The assertion based approach aims to monitor the activity of these wireless sensors and catch faulty or critical conditions.

This thesis also studies compression techniques suitable for wireless polysomnography which further helps to reduce the power consumption on board these devices.

The Network-on-a-chip literature shows considerable promise for use of assertion based debug techniques. This thesis aims to model a NoC system and apply assertion based debug for capturing faults in the system. We propose to leverage the library of formally described properties, created as part of a design process aimed towards ABV, to translate some of the debug benefits to the final silicon.

Chapter 2.

Background

2.1 Wireless Sensor Networks

Wireless sensor network (WSN) consists of dozens of tiny battery powered devices capable of small computation. These devices monitor, sense or display information such as temperature, humidity, electrocardiogram, etc. The actuating device could cause a number of responses depending on the onboard program and available hardware. The response could involve a blinking LED, display textual information or transmit the information over the wireless channel [35].

The applications of WSNs are rapidly increasing with the miniaturization of VLSI technology and low power devices. WSNs are traditionally used in remote environments where wired connections are difficult or expensive. They have been used in commercial, industrial, environmental and healthcare applications to monitor different activities. Popular applications include aircraft monitoring, ecological habit monitoring, education and geological monitoring.

Most wireless sensor platforms have share a common architecture as shown in Figure 1 and have the following common components:

Microcontroller - These are the heart of the wireless node and provide the computational capability. These are different from traditional CPUs with respect to their power

efficiency and low cost. They are usually expected to be in sleep modes and perform bursts of computation operations. This reduces the power consumption. There a number of these processors available in the market like the Texas Instruments MSP family and the Atmel ATMEG family.

Radio - They are responsible for the data transmission and reception. The radio transceivers operate either at 315/433 and 868 MHz in Europe, 915 MHz in North America, or at 2.4 GHz worldwide. The WSNs usually operate in the license free industrial, scientific or medical (ISM) bands as they are open to all radio transmitters which follow the regulations. The most common radio choices are IEEE 802.15.4/Zigbee, Bluetooth, Ultrawide Band (UWB).

Interfaces - The sensor interfaces are required to connect the external sensors or the communication ports to the embedded processor. This is achieved by either employing SPI, I2C, UART or some GPIO.

Power - These devices usually operate on a battery or might be even connected to solar arrays in some cases.

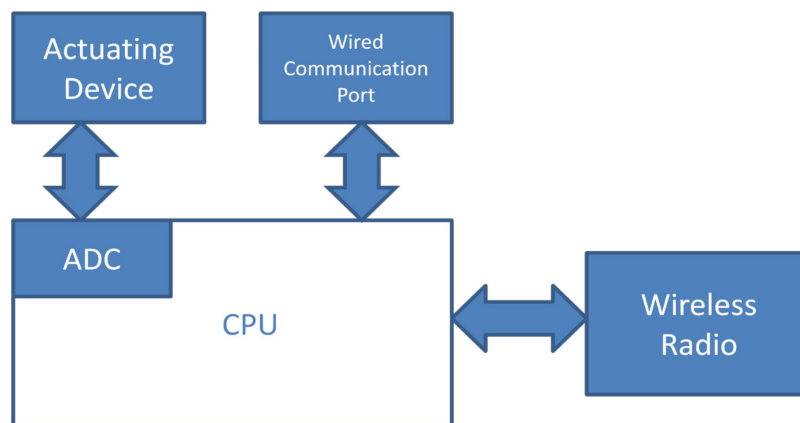


Figure 1. General architecture of WSN

A WSN shows a large degree of parallelism and is essentially a distributed system. Hence the design of these systems is challenging. These are embedded systems with constraints on energy, storage and memory resources. They need to have a robust communication protocol design to handle the unreliable communication channel. Depending on the environment they are deployed, they have to face a number of challenges like noisy radio

channels to temperature variations. Most of these WSNs are designed to operate autonomously for multiple years with limited access and supervision. Hence, there is a need to monitor the correct functioning of these devices.

2.2 Evaluation of wireless kits

There are a number of products to choose from and most of them offer similar functionalities. However use of these devices need sufficient expertise in programming microcontrollers and interfacing devices. Moreover these devices are provided for research labs and for university teaching purposes. In this section we would cover the usage and functionalities provided by two wireless sensor kits that are commercially available; namely Ecomote and Shimmer wireless kits.

2.3 ECOMOTE kit evaluation

2.3.1 Overview

Ecomote is a wireless sensor node with an integrated 8052 microcontroller on board. The mote has an nRF24E1 transceiver to enable wireless communication. The device uses its *Shockburst* mode for its wireless transmission in the 2.4GHz band. It is powered by a 40mAh rechargeable Li polymer battery. The device is also equipped with UART for serial transmission of data.

The Ecomote consumes 3mA at full power. However, the device consumes only 2 μ A when in standby mode. Hence the device should quickly return to this mode as soon as possible after waking up. The radio onboard enables it to communicate using the IEEE 802.15.4 standard.

The Eco API document [39] covers all the functions, header files and variables provided. This should be used along with the datasheet for the device. Although the authors describe the program functionality, the datasheet is required for understanding these modules better and also gives a clear picture of the microprocessor functioning. The program definitions included in this document will give insight into its intended usage.

2.3.2 Reading accelerometer values through serial port

This section contains a small program to read the accelerometer values using the ADC modules provided. The values were transmitted to the computer through serial (UART) port using the serial port modules. In order to use ADC, we have to include the ADC header file and the functions that we are using.

The program, Accel.c (Refer Section 8.2.1) first initializes the ADC with a clock ($\text{cpu clk}/32$ in our case), 10 bit resolution and an external reference voltage. The accelerometer datasheet indicates this to be approximately $V_{cc}/2$ (1.5 V). After initialization we read the x,y and z pin values. We then send these values through the serial port.

2.3.3 Host program to read serial data

In order to read serial data, a python script is required (refer Section 8.2.2). The modules provided in Universal Serial Port Python library (USPP) [41] were used for the serial data transfer. The USPP Library is a multi-platform Python module to access serial ports.

In order to run the program we need to first create the hex file and store it on the Ecomote. On switching on the Ecomote the program starts running. The host computer should also start running the host python script to capture the data through the UART connection.

2.3.4 Limitations

2.3.4.1 Linux support

The developers should look into providing drivers and applications for a Linux operating system. Currently a windows based machine is required to install and run the Ecomote applications.

2.3.4.2 Debugger

The Ecomote uses SDCC compiler which comes with a built-in debugger called SDCDB. However this debugger cannot be used for the Ecomote as there is no support for it as yet. We have to develop some novel techniques to debug the program. One way of doing it is

write some functions in order to trace the flow of the program. We could keep sending some values or characters serially to indicate the current code line or status.

It would also be a good option to write functions which enable us to read and possibly modify values in the registers. Programmers should ensure that they perform the above steps before writing bigger applications as it would be very difficult to debug later on.

2.3.5 Target Work group

The kit will be helpful for students or researchers who would like to quickly start their research work involving data sensing. The device can easily be interfaced with other sensor devices through SPI. Hence the researchers can extend its sensing capabilities by attaching their devices to the Ecomotes. The biggest advantage of the device is its small size and low power consumption.

2.3.6 Possible applications

The Ecomote can be mainly used for remote sensing applications where small data has to be transmitted intermittently. With the presence of an accelerometer, motion detection is a major application that could be exploited. It would help researchers interested in studying motions in animals, adding wireless communication/data transfer to a product, etc are some possible applications.

The Ecomotes do not have a unique address and hence when used on a network, they have to be manually assigned an address. This would be a drawback to the Ecomotes when used in sensor networks. This may not pose any major problems when we are handling only a few motes. But in applications where we need many motes in a network, each Ecomote will have to be manually assigned a unique ID while programming. This problem could be solved by adding a silicon identifier to the Ecomote.

2.4 SHIMMER kit evaluation

2.4.1 Overview

Shimmer is a wireless sensor platform mainly intended for wearable health sensing applications. Shimmer sensor has a TI MSP430 microcontroller, which is widely used in the biomedical field and also has a good reputation for energy saving applications. The device has a Roving Networks IC for Bluetooth communication and a Chipcon 2420 for 802.15.4 radio. It also has a Freescale 3-axis accelerometer and also a SD card slot (up to 2GB) for storing data. The Shimmer1 has a 250 mA battery to power its applications. The device also has numerous custom daughter boards to extend its range of functionalities.

The daughter cards included with the kit are

- ECG daughter card
- Gyro board
- An Extension board

The ECG module would be of interest to biomedical engineers and researchers who would like to capture and monitor the heart activities. The Gyro board gives the orientation of the Shimmer device (i.e. the person wearing it). The extension board helps as a breakout board. This would be useful in places where the researcher would like to connect custom sensors or devices to the Shimmer. The transfer can use SPI to communicate between Shimmer and the custom device. Shimmer developers also offer to develop custom expansion modules on request at an extra cost.

Shimmer has the advantage of a small design and is packaged robustly and tested for these environment conditions. Shimmer enables plug and play functionality for research clinics to obtain data.

Shimmer development is based on TinyOS. TinyOS is a component-based operating system and a platform targeting wireless sensor networks. It is written in the nesC (**n**etwork **e**mbdedded **s**ystems **C**) programming language. TinyOS programs are built out of software components. Components are wired together to run application.

Shimmer also has a windows based platform called BioMobius for developing applications based on the nesC programs. There are different user levels for the

Biomobius platform. A biomedical engineer would design the EyesWeb applications and develop a simple GUI for the same. There EyesWeb application is designed by TRIL and they provide a comprehensive documentation of the program. It offers an interface for complex mathematical computation and signal processing blocks.

A clinician on the other hand simply has to simply load the required programs using the bootstrap loader onto the Shimmer and run the GUI. All the data and graphs will be stored according to the EyesWeb program. The GUI can have a simple button interface like Start, Stop , Capture data, graphs.

There is also a windows bootstrap loader application. This is necessary to load the ihex program files onto the Shimmer from windows operating environment.

2.4.2 Functionalities available on Shimmer 1

The functionalities on Shimmer are ported from the TelosB node.

- Reliable TCP/IP stack for 802.15.4
- Reliable Bluetooth® SPP interface
- Telnet server (including SIP Telnet Server)
- Http server
- Real time clock module
- LED control
- Low voltage alerts
- Low-power ADC using DMA-mode
- Optimised ADC/DMA block transfer operations
- Accelerometer control
- Gyroscope daughter card control
- ECG daughter card control
- AnEx breakout board control
- Sensor control/configuration interface
- Power supply monitor operations

2.4.3 Next Generation - Shimmer 2:

Shimmer 2 has some extra functionality as listed below:

SD data bypass for faster data transfer. It also features a soft power switch. It provides Bluetooth and SD power control which lowers idle power by 50%. A Tilt-power switch is included to detect if in usage and can help trigger the sleep mode in idle state. It also features a redesigned case and a 280 mAh Lithium battery. These new features try to minimize power consumption and give longer battery periods. There is also a USB dongle called SPAN to connect the Shimmers to the host computers using 802.15.4.

2.4.4 BioMobius Accelerometer Example

2.4.4.1 Configuring and setup Bluetooth device

The Shimmer Bluetooth device has to be initially added by windows. This is documented in the *HelloWorld* application (section 3.6).

We have to first upload the program file on the shimmer mote. The device then has to be added in the Bluetooth device handler for windows. The Shimmer device is named as Firefly-XXXX by default. After adding the device, a passkey must be enabled, the default being 1234. The COM port associated with the Bluetooth device should be noted as this will be used in the EyesWeb application to communicate via Bluetooth.

2.4.4.2 Accelerometer patch:

This patch has to be made using the EyesWeb application (refer Section 8.3.1). The patch enables the user to capture the required data from the Shimmer and give access to the various ports depending on the program. Initially we need to add the Shimmer block on to the workspace. The search bar locates the required block in EyesWeb. Enable the required signals in the Shimmer by clicking on its properties menu. Add the desired bang generators (button which will start the application) and output elements or graphs. We can run the patch with the play button. The Connect button connects the Shimmer through Bluetooth. The Start button starts reading the accelerometer values and the

graphs plot these. The Stop button stops the application and the Disconnect button will disconnect the Shimmer Bluetooth connection to the windows machine.

2.4.4.3 Accelerometer GUI:

The Biombius GUI application creates GUIs (refer Section 8.3.2). These are for end like clinical technicians or even patients who do not have to interfere with the data. They are responsible only to start and stop the device and ensure the safe capture of the required information. The software enables us to draw the desired shapes for buttons. The desired patches can be imported for the GUI and linked the buttons in the GUI to the desired function in the patch. We can also add graph display and link them to the graphs present in the EyesWeb patch.

2.4.5 Limitations

- The Biomobius platform is only for clinical use. Adding functionality depends on the nesC program.
- Shimmer has limited debugging capabilities. The only way to debug is to either send data serially or to telnet to the device and dump the memory values. An inline debugger would be of greater use as we can see all the variables of the microcontroller.

2.4.6 Target Work group and application

Shimmer platform has proven to be a very versatile sensor for use in biomedical applications. The device has a number of daughter cards and expansion modules which provides expandability to its applications. However more number of Shimmer modules will be required for research purposes as the basic kit provides only 2 Shimmer nodes.

The device would be helpful for researchers in the biomedical domain but would require a programmer would sufficient expertise in TinyOS to develop applications or else would be limited to the applications provided by the manufacturer.

2.5 Vital Sign Monitoring

Vital signs are measurements of the body's most basic functions [33]. The four main vital signs routinely monitored by medical professionals and healthcare providers include:

- body temperature
- pulse rate
- respiration rate (rate of breathing)
- blood pressure (Blood pressure is not considered a vital sign, but is often measured along with the vital signs.)

Vital signs are useful in detecting or monitoring medical problems. Vital signs can be measured in a medical setting, at home, at the site of a medical emergency, or elsewhere. Figure 2 depicts a general wireless health monitoring setup in hospitals and homes.

For vital sign monitoring, we want to ensure that the patient being observed is outside any critical conditions. . For example, say the breathing rate monitor in the case of an adult could be set to ensure that it is between 12-20 breaths per minute. The moment it goes outside this range, the sensor should send a distress signal to alert the healthcare provider.

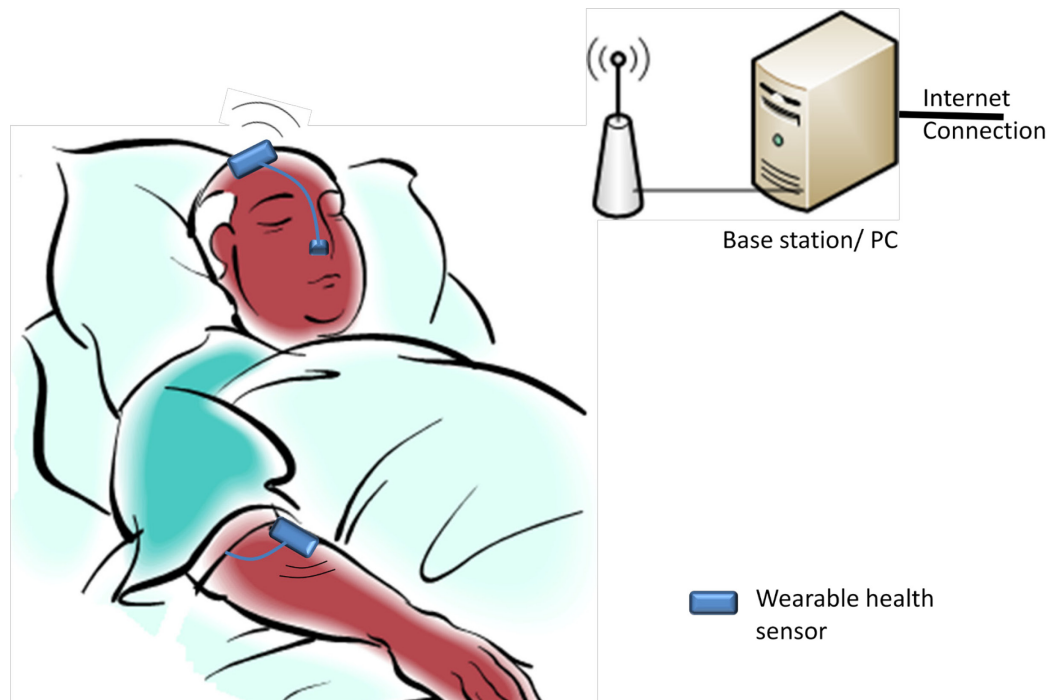


Figure 2. General setup of wireless health monitoring [55]

Galeotti et.al [25] developed a vital sign monitoring system called UPAD with a wireless connection to the computing platforms like PC, smart phones or a PDA. The UPAD has a 2-channel ECG, a respiratory signal from a chest piezo belt, a pulse oximetry signal and 3 analog inputs to connect communication devices for disabled people. The device captures the parameters as expected and has been reported to be easy to use. However the paper does not discuss any test setup of scanning for hardware or software errors. Moreover the power consumption in the worst case scenario drains the battery quickly. The power saving schemes employed is limited to reducing the sampling rate in order to conserve the transmission power. This scheme cannot be applied too often as the sampling rates in most cases are high.

Takizawa et.al [26] have performed an extensive analysis on the application of Ultra Wideband based wireless vital sign monitoring. They have categorized the vital signs into two categories namely continuous and routine vital signs. The continuous signals like ECG requires to record data constantly. The routine signals require a less frequent recording of data. Through their studies they were able to reduce delay times using designed superframe structure.

Hu et.al [19] developed a wireless healthcare device named PoSeat. The device is a smart seat cushion for chronic back pain prevention. The sensors on the PoSeat system monitor the patient's sitting behavior and also record information of the operating environment like vibration strength. This data is then sent to a PDA or mobile phone. The device then would warn the patient if the posture is not correct and also updates the patient's database on a central server.

2.6 Sleep Apnea

Sleep apnea is a sleep disorder characterized by pauses in breathing during sleep. There are three types of Sleep apnea: central, obstructive and complex. Obstructive sleep apnea is the predominant one seen among most patients. Central sleep apnea is characterized by lack of respiratory effort. In obstructive sleep apnea, there is a physical block to airflow despite respiratory effort. In complex (or "mixed") sleep apnea, there is a transition from central to obstructive features during the events themselves. Sleep apnea is rarely

detected by the patient. It is usually recognized by others witnessing the individual during such episodes or is suspected because of its effects on the body. Due to the lack of sleep in night, the patient may develop daytime sleepiness and also appear fatigued. Sleep apnea is diagnosed with an overnight sleep test called a polysomnogram or a Sleep study. Polysomnography is a comprehensive recording of the bio-physiological changes that occur during sleep. It is usually performed at night, when most people sleep. It includes measures of electroencephalography (EEG), electromyography, electrooculography, electrocardiography, ventilation and oxygenation. After recording the data overnight, Polysomnography requires skilled scoring and the corresponding interpretation by the doctor.

The American Academy of Sleep Medicine (AASM) has four classifications of diagnostic sleep equipments based on the number of clinical parameters that are recorded during the study and also the environment of study, i.e. either attended or unattended sleep study.

- A type 1 sleep study would require to be attended in a sleep centre or hospital laboratory by a sleep technician with a minimum of 7 measured parameters. These would include brain waves, eye movements, chin muscle movements, respiratory effort, oxygen saturation, airflow, heart rate, or rhythm.
- A type 2 study would be performed unattended by a portable polysomnography device at home with a minimum of 7 measured parameters of brain waves, eye movements, chin muscle movements, respiratory effort, oxygen saturation, airflow, heart rate, or rhythm.
- A type 3 study would be an unattended cardio-respiratory testing at home with a minimum of 4 clinical parameters including at least 2 respiratory variables (e.g. Respiratory movement and airflow), a cardiac variable (e.g. Heart rate or rhythm), and oxygen saturation.
- A type 4 study is an unattended single or dual bio parameter recording at home. This requires a minimum of 1 clinical parameter including oxygen saturation, airflow, or chest movement.

Polysomnography (PSG) is the gold-standard investigation used for diagnosis of Obstructive Sleep Apnea (OSA). The study is conducted overnight and the patient needs

to be continuously monitored by a sleep technician. However there are often long wait periods to meet a sleep physician. As a result more research work has focused on the development of portable sleep study devices.

In the case of polysomnography, the sensed data has to be stored for analysis at a later time. In order to ensure a completely automated and reliable study, the portable sensor should transmit the data to the base station where the data can be analyzed or even forwarded to a central server over the internet. Most related work [25][26] send a continuous stream of the raw data over the wireless channel. While this approach works, it does not do so in a power efficient manner.

Polysomnograph is useful for a variety of clinical studies and is the current preferred diagnostic methodology for sleep apnea [18].



Figure 3. A pediatric patient prepared for a polysomnogram by a respiratory therapist [31]. These wires connected could be troublesome in the sleep tests

Collop [18] follows the development and changes in portable monitors for polysomnography in her review article. The reviews of portable monitors for OSA

concluded that type 2 portable monitors were not suitable for clinical studies as there were very few studies available. Type 3 and 4 did not provide accurate apnea-hypopnea index (AHI) and were also not recommended for clinical studies. In addition, portable monitors were not recommended for split-night studies. With the development of new portable devices later on, additional studies were conducted. These studies compared the response to continuous positive airway pressure (CPAP) therapy among patients diagnosed with OSA via portable monitor with the response among patients diagnosed via PSG. The study concluded that PSG did not confer an advantage over portable monitoring with respect to CPAP usage or acceptance of therapy. This showed that portable monitors could be used to diagnose patients with OSA.

Further studies were carried out by the Tufts – New England Medical Center Evidence-based Practice for home diagnosis of OSA-hypopnea syndrome and OSA-hypopnea syndrome modeling for different diagnostic strategies. The first report concluded that type 2 and 3 portable monitors are likely to work as well as PSG in selected populations for predicting a positive response to CPAP. The second study used some mathematical simulation models for diagnosing OSA and treating with CPAP while considering different age and time criteria. The model estimated that the sleep centers would take 13.6 weeks to be diagnosed with OSA and 27.3 weeks to be offered CPAP. However, the use of portable monitors would take 2.1 weeks to diagnose and 4.8 weeks to be offered CPAP. These reports Center further reinforced the merits of portable monitors' ability to adequately diagnose OSA and provide timely diagnosis.

In 2007, the American Academy of Sleep Medicine (AASM) issued the clinical guideline on use of portable monitors in adults. These were later used to set the standards for sleep center accreditation by the AASM. They recommended that portable monitors may be used as an alternate to PSG only in patients with a high pretest probability of moderate to severe OSA. The portable monitors were recommended not to be used in patients with comorbid medical conditions or with suspected comorbid sleep disorders. They further stressed that a portable monitor must record airflow, respiratory effort, and blood oxygenation. A sleep technician is required to monitor or educate patients on the proper usage of these devices.

The author suggests that patients using these devices should be well aware of all the advantages as well as its limitations. She further adds the requirement of these devices to account for some level of data loss and must also provide test validity.

2.7 Assertions

“Assertions are statements that are added to a design to specify its correct behavior” [42]. Software assertions have been used for a long time mainly for debugging code and were sequential in nature. However, the concept of hardware assertions has been gaining usage over the past decade. In hardware, the assertions are used to specify the correct behavior of the circuit by defining corresponding properties.

Assertions are used for the following:

- As documentation or specification (designer intent) – Assertions present an unambiguous way to represent the design rules to which the implementation must adhere. They are also a compact form of documentation and specification.
- In formal verification tools for assertion checking (property checking) – This is performed by utilizing automated theorem provers and model checkers that analyze a model of the design along with its properties or theorems, and are able to formally prove their validity.
- In dynamic verification – Dynamic verification is the predominant verification approach and is mostly associated with simulations and corresponding simulators (ex: ModelSim, VCS, etc.). These simulators are usually easy to setup and do not require advanced mathematical skill required to operate model checkers and theorem provers.
- As debug aid – Assertions can also play an important part in post-fabrication silicon debug where assertion checkers are fabricated in silicon for in-chip debugging [54].
- In on-line monitoring – Complementary to debugging, the silicon fabricated assertion modules can also be harnessed to monitor the operating conditions of the device.

2.7.1 Verification with assertions

The aim of hardware verification is to ensure that a given circuit always performs according to a set of specifications. It is difficult to get complete specifications for a circuit and hence designers break them into properties which are simpler to verify. Consider the following example of ensuring that the *Ack* signal goes high within 5 clock ticks after the *Transmit* signal goes high. It is extremely difficult to manually monitor this signal and catch the error if it occurs. It is here that assertions come into play. Assertion does the monitoring or checking for us. As observed from Figure 4 the assertion is fired when the given condition fails.

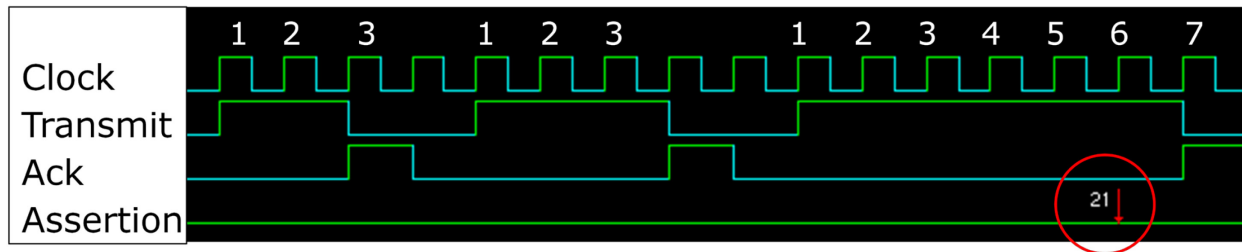


Figure 4. Verification with assertion

Assertion based verification (ABV) has been gaining in popularity for the task of verification and taping out a design. ABV utilizes the power of temporal languages like Property Specification Language (PSL) and the System Verilog Assertions (SVA) to insert checkers that monitor specific properties that a design should satisfy. Assertion based methodology for in-circuit verification requires their conversion into monitor circuits that detect assertion failures. This is done with the help of a checker generator.

From a circuit point of view, an assertion checker is a synchronous circuit that takes synchronous signals as inputs, update an internal automata based on the monitored signal transitions and report failures as a synchronous pulse on a logic output. The complexity of the automata is abstracted from the designer and its generation is fully automated by the assertion compiler. The MBAC tool is one such assertion checker generator that includes debug enhancements like hardware coverage monitors, activity tracers, assertion completion and assertion threading. These debug enhancements can be optionally

activated. Geuzebroek et.al [48] used MBAC to integrate assertion checkers such that they can use existing scan-based or debug trace infrastructures.

Boule et.al [52] present techniques to include debug capabilities using hardware assertion checkers. Their approach proposes coupling it with a software control which enhances the checker generation process so as to converge on the root cause of an assertion failure. With the embedded logic analyzers, it is possible to use dependency graphs to provide debug information. They are particularly useful for complex assertion failures. However this approach requires the assertion to be constructed correctly without having redundant signals or without the combination of sub-expressions resulting in a contradiction with respect to a subset of the input signals or parameters. The MBAC tool was also used to monitor the activity of a sequence. This helps to ensure that the input is exercising a portion of the assertion. A consequent assertion completion implementation provides further assurance of correct operation of circuit. Counting assertion failures is another feature added on the tool. This feature could also be clubbed with the assertion completion and inform of the number of successful completions and failures. The verification of pipelined circuits requires the identification of failures in separate sequences. Assertion threading achieves this goal by instantiating many copies of the assertion sequence circuits. The results of the MBAC debugging tool when compared against another checker generator called FoCs proved to use lesser hardware resources for the same coverage. The activity monitoring and assertion threading circuits were also synthesized. As expected they utilize more hardware resources for their increased debug ability. The MBAC tool proves to be very well suited for complex temporal sequences, eases debugging and offers better observability of circuits.

2.8 Network on a chip

With advances in technology we see a rapid increase in the number of transistors integrated into a single chip. Moreover we see multiple IP cores connected together on a board to deliver rich multimedia and networking services. With advancing VLSI technology, all the components can be embedded into a single chip. Thus was born the term system-on-a-chip (SoC). SoC architecture started with cores connected in an ad-hoc point-to-point approach which eventually gave way to a bus-based architecture. The bus-

based architecture eventually presented quite a few limitations like increase in parasitic capacitance, propagations delays, power consumption and arbitration times. As described by Stephan [46], the main problem with the point-to-point and bus based approaches lies in their limited scalability and flexibility. Soon the concepts of networking and parallel computing were adapted for on-chip communication and lead to the introduction of the networks-on-chip (NoC) architecture. *“An architecture that is able to accommodate such a high number of cores, satisfying the need for communication and data transfers, is the networks-on-chip architecture”* [43].

The basic and most popular NoC architecture is the two dimensional mesh networks. Figure 5 shows a 3x3 grid with 9 tiles or nodes. Each tile has an intellectual property (IP) core connected to a switch through its network interface (NI). The switches handle the routing strategy of the packets to enable communication between the different cores. For our studies we consider homogenous NoCs where each node has the same kind of processing element (PE).

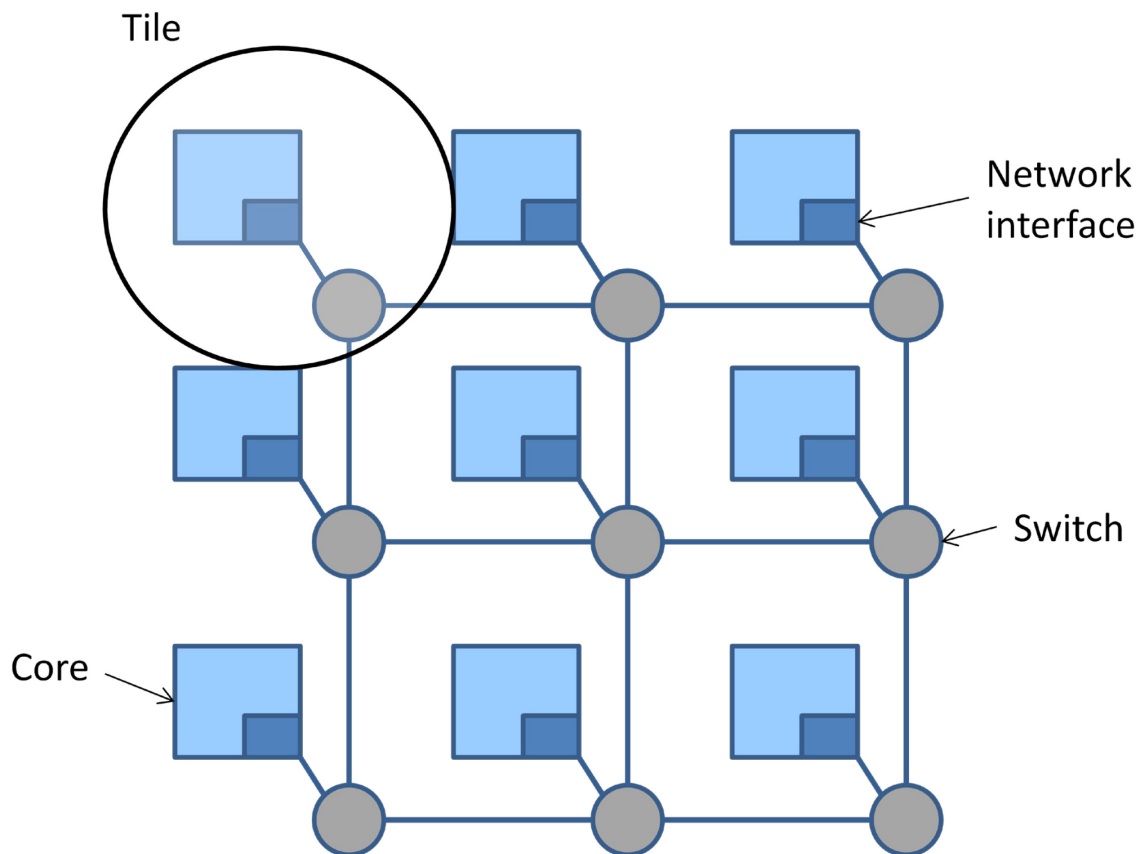


Figure 5. A 3×3 NoC where the cores have been arranged in a two-dimensional grid

As with every emerging technology, NoC also has different problems to be addressed. The biggest challenge with NoCs is due to the fact that it involves the bridging of different domains like networking, graph theory, optimization algorithms, real-time systems, synthesis, design automation, and many more. A first outlook into systematic approaches to improve NoC design quality was presented by Stephan et.al in their paper titled “A Quality-Driven Design approach for NoCs” [53]. The article presents a development platform that can estimate the quality of the design through high level modeling. The high level NoC tool can test, debug and provide CPU control from a single access point. For improved design quality the authors consider two aspects of the design: verification and the testing, monitoring and debugging (TMD) infrastructure. The authors suggest that system-level verification be included earlier on in the design phase due to the NoC’s communication centered design as opposed to a computation centered design. The TMD infrastructure incorporates hardware checker generation from PSL statements using the MBAC tool discussed earlier in section 2.7.1. The simulation model helps in design space exploration with respect to properties such as area, speed and power for different NoC configuration. Based on their model, the authors derive the relation for quality of design as a combination of verification quality (Q_V), quality of TMD infrastructure (Q_{TMD}) and quality of NoC architecture (Q_{NOC}). The quality of TMD infrastructure was further shown to be dependent on the quality scores of test, monitor and debug hardware. The cost of the design is the sum of the cost of the hardware without the TMD infrastructure and the resource requirement for the TMD infrastructure. The tool flow starts with identification of the PSL statements for verification and monitoring that are suitable for hardware translation. These are selected from the high-level model and are synthesized. This step is followed by the TMD infrastructure creation by generating monitors for the hardware assertion checkers, an interface for the PE and a network interface for monitoring assertion transitions and informing a central unit. Based on this model, the authors compare between hierarchical-ring and hyper-ring topologies. The hierarchical-ring topology consists of four local rings and one global ring. The local ring contains four ring interfaces (RI) and a inter-ring interface (IRI) to connect the local ring to the global one. The local ring of a hyper-ring has four ring interfaces (RI) and two inter-ring interface (IRI). Since the components for the architectures are similar the

quality of verification was decided to be almost the same for both the configurations. The placement of debug and monitoring hardware at the IRI bring more quality to the design and hence the hyper-ring design has a better quality of TMD infrastructure as it can route more traffic globally. The authors attribute the property of path diversity to the quality of NoC architecture. The presence of an extra IRI helps the hyper ring to obtain a better score for quality of architecture. Consequently the hyper ring also occupies slightly more area during synthesis. The authors compare the results for different operating frequencies of the two architectures. On the basis of their calculations, it was clear that the hyper-ring architecture proved to be better than the hierarchical-ring architecture. This work demonstrates a systematic approach to design with debug considerations.

Chapter 3.

Sleep apnea and liquid detection platform

In this section we present a novel approach to detect sleep apnea and develop a custom printed circuit board (PCB) which was developed as an evaluation board for researchers and in hospitals. The device measures respiration by detecting changes in capacitance due to upper and lower body movement during respiration. Our approach relieves the patient of any wires connected to the body. The pad also has an additional safety feature of liquid detection by which it detects any liquid flow on the pad. We have used Microchip's dsPIC30F6014A for controlling the pad operations and Analog device's AD7746 capacitance to digital converter (CDC).

3.1 Introduction

The respiration rate is the number of breaths a person takes per minute. The rate is usually measured when a person is at rest and simply involves counting the number of breaths for one minute by counting how many times the chest rises. Conventional measurement of respiration involves bulky apparatus with lots of wires or tubes connected to the body and the mouth. These wires and tubes pose to be a major hindrance for the patient while sleeping. Our device eliminates this problem by introducing a wire free technique to measure respiration. The pad is designed to capture the capacitive coupling between the conductors on the pad which changes due to the motion of the

lungs. A major application of measuring respiration is in detecting sleep apnea. Hence we have concentrated our work on this area.

3.2 Pad Design

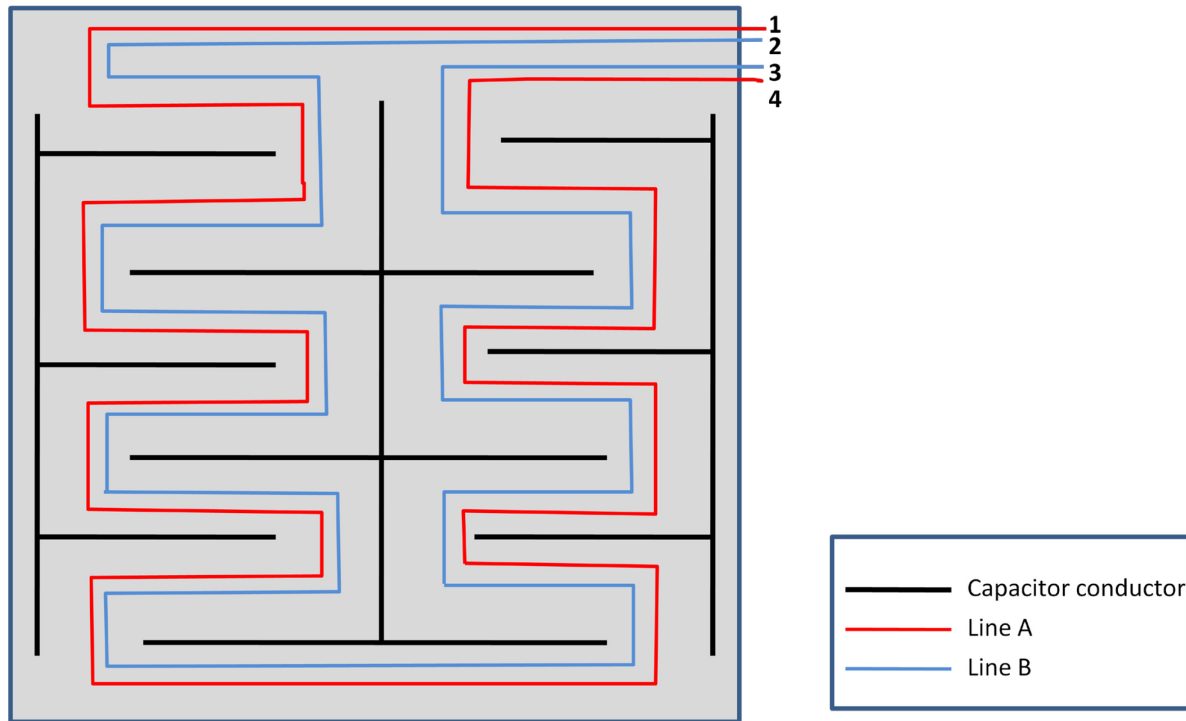


Figure 6. Pad design

The pad Figure 6 can be divided into two parts; the liquid detection and the respiration detection. We will first consider the pad design for respiration. The pad can be divided into two halves, in other words, it has two parallel plate capacitors. These capacitors measure the respiration of the upper and the lower half of the body. The connections to these capacitors have not been shown in the figure for simplicity. These capacitors are connected to the AD7746 which has two capacitive input channels. The high sensitivity of the capacitor chip to small changes in the capacitance was established using the AD7746 evaluation board. In order to measure respiration, we used the experimental setup as shown in Figure 7.

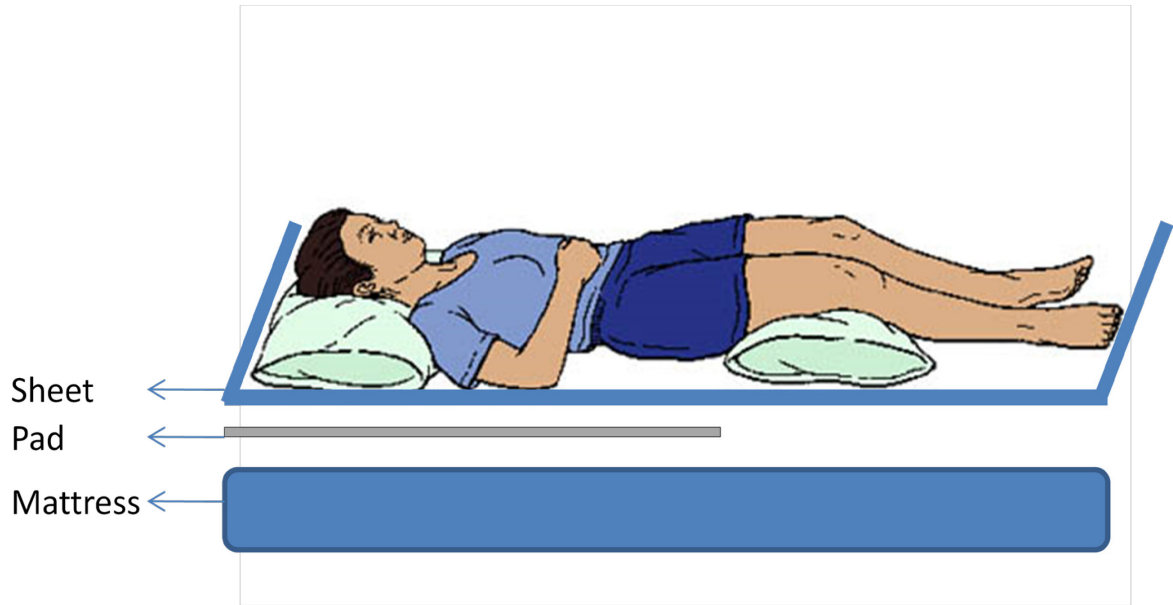


Figure 7. Patient lying on bed for sleep apnea detection

By varying the breathing speed and style, we obtained satisfactory waveforms for respiration. It was also noticed that during this procedure, the capacitance values might saturate. Hence we have to vary the input capacitance offset. This can be achieved by programming the capacitance chip accordingly.

The liquid detection on the pad has been facilitated with the help of two parallel conductors (line A and line B) running along the pad (Figure 6). The lines in the figure are spread wider for clarity. They are also more dense and closer on the pad. Under normal operation (Figure 8) a small current flows through the circuit. When there is a liquid leak, line A and line B get short-circuited. This will cause a larger current to flow through the circuit and helps detect the liquid presence.

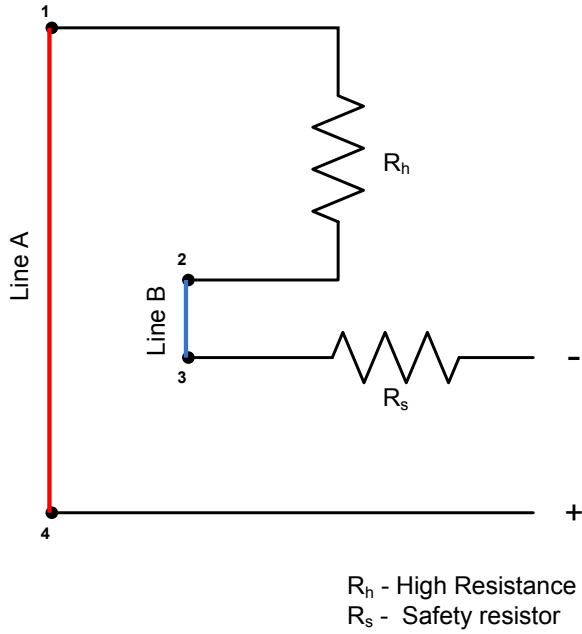


Figure 8. Liquid detection

3.3 Interfacing with AD7746

The Analog devices chip, AD7746 is a 24 bit capacitance-to-digital converter. The AD7746 supports an I²C compatible 2-wire serial interface [32]. The two wires on the I²C are called SCL (clock) and SDA (data). These wires carry all addressing, control, and data information one bit at a time over the bus to all connected peripheral devices. The CCS compiler provides the option of using software as well as hardware I²C. Since our circuit does not operate at very high frequency and also for simplicity of code, we have chosen software I²C.

The start byte address for AD7746 is 0x90 for write operation and 0x91 for a read operation. For test purposes we have used the chip in single ended capacitive input mode where in only one of the capacitive inputs is used (EXC A). For the final product, we will have to use both EXC A and EXC B pins to measure both the upper and lower chest respiration.

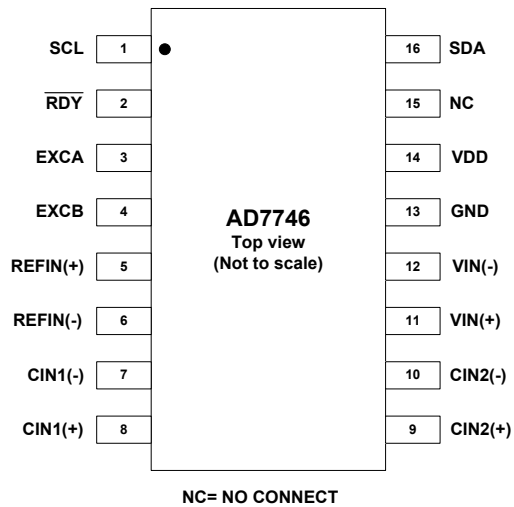


Figure 9. AD7746 Pin configuration [32]

The AD7746 has 4 registers which have to be configured before use. The configurations used for our operations are depicted in the figure below (Figure 10).

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Cap Set-up register

Register address: 0x07

Bit 7=1; Enable capacitive channel for continuous conversion

Bit 6=0; select first capacitive input

Bit 5=0; differential mode off

Bit 4-1; must 0 for proper operation

Bit 0=0; for specified capacitive chip operation

1	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

Voltage/ Temperature set-up register

Register address: 0x08

Bit 7=1; Enable voltage/temperature channel for continuous conversion

Bit 6-5=0; select internal temperature sensor

Bit 4=0; Select on chip internal reference

Bit 3-2; must 0 for proper operation

Bit 1=0; internal short switch open

Bit 0=1; sets internal chopping on the v/t channel

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

Exc Set-up register

Register address: 0x09

Bit 7=1; CLKCTRL bit set to 0 for faster conversion times

Bit 6=0; EXCON bit set to 0, Excitation present on the output only during capacitive channel conversion

Bit 5=0; disable EXCB pin as excitation output

Bit 4; disable EXCB' pin as excitation output

Bit 3=1; enable EXCA as excitation output

Bit 2=0; disable EXCA' as excitation output

Bit 0,1=11; Set Excitation voltage level

0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---

Configure register

Register address: 0x10

Bit 7,6=00; V/T channel conversion time= 20.1 ms, Update rate=49.8 Hz

Bit 5,4,3=010; Capacitive channel conversion time 20 ms, Update rate=50.0 Hz

Bit 2,1,0=001; Continuous conversion mode

Figure 10. Register configuration for AD7746

3.4 Bread boarding

In order to make sure that our design works, we bread boarded all the components. The dsPIC30 was connected to the pad and the cap chip. The test programs were run to ensure the desired operation.

3.5 I²C isolation

The inter-integrated circuit bus (I²C-bus) provides an attractive maintenance and control communication interface between parts of a system since it uses only two signal wires yet has powerful addressing and a reasonably fast, bidirectional data handling capability [34]. Medical patient-monitoring equipment needs to operate without any common physical interconnection wiring to form a safety isolation barrier to prevent any chance of electrocution.

These and similar applications need galvanic isolation, so including opto-couplers in I²C-bus signal wires is obviously attractive. Unfortunately it's not so simple to provide opto-isolation of the I²C-bus because the I²C clock and data signals are both bidirectional signals while opto-couplers can only handle unidirectional signals. The challenge to optically isolate the I²C-bus has been achieved by splitting the bidirectional I²C signals into unidirectional data streams and recombining them again.

3.6 Additional features

We also included a real time clock in our design. This was connected to the dsPIC30 through I2C.

We also wanted to incorporate certain extra features such as touch screen, Bluetooth, USB connectivity, EEPROM, and an extra SD card for extended storage capability. All these features cannot be incorporated by using the same dsPIC30 as it would be overloaded. The main purpose of the dsPIC30 is to run algorithms to detect abnormalities in breathing and hence detect sleep apnea.

Hence we decided to use another PIC to incorporate these additional features. PIC18F87J50. PIC18F87J50 family devices contain a full-speed and low-speed,

compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB host and the PIC microcontroller.

The extra SD card and EEPROM have been included so that we can store more data within the device. This gives us the option of processing the data before sending it out to the central hospital servers/computers.

For added usability to the end user, we have added a touch screen option. This is also connected to the PIC18F87J50 through I2C connection.

For both the microcontrollers we are using an external oscillator of 10 MHz. Since our board is still being developed and tested, we provided some breakout pins. This will help for easier debugging while programming.

The figure below (Figure 11) shows the various connections between the PICs.

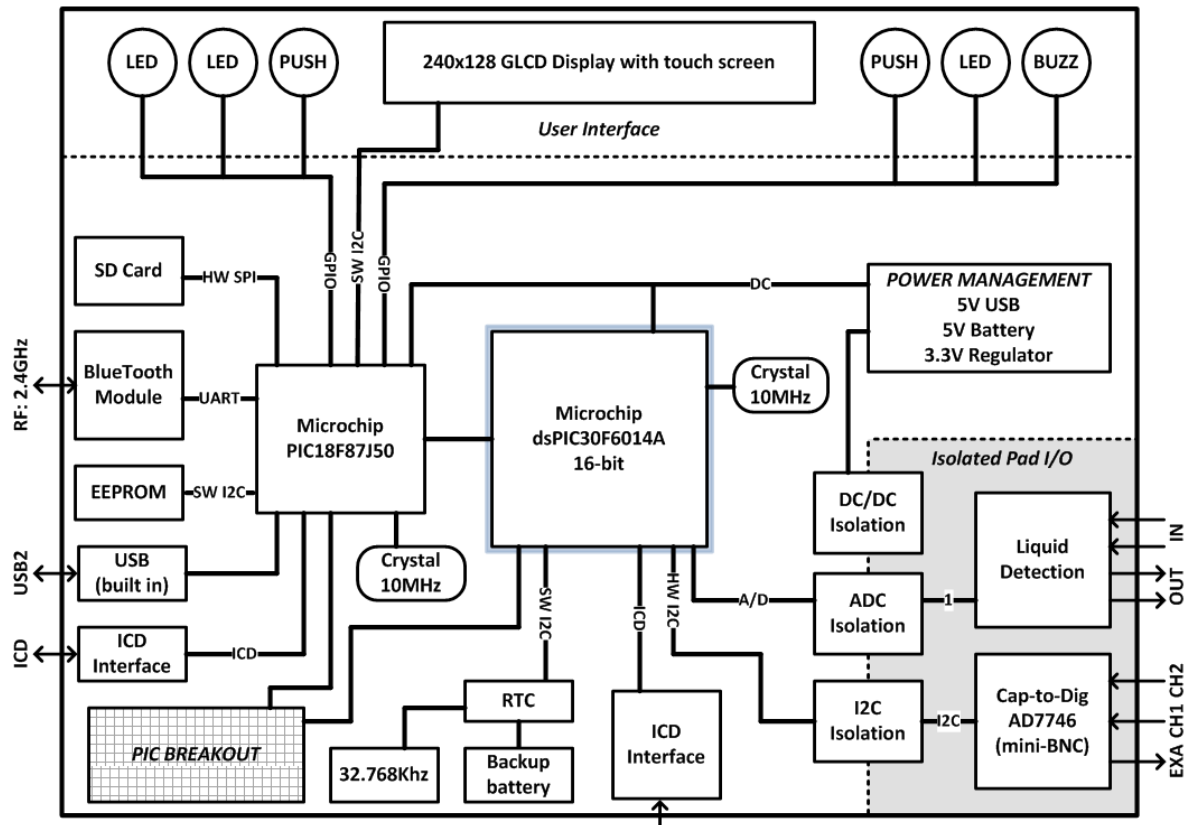


Figure 11. Schematic of the system level design

3.7 Programming the dsPIC30

The microcontroller was programmed using the PCD compiler. PCD is a C compiler for Microchip's 24bit opcode family of microcontrollers.

We shall now briefly consider the main functions in the program

Cap_init()- This The function initializes the AD7746. It sets the registers values according to the required configurations. It takes 4 parameters as input: capsetup, vtsetup, excsetup and config.

#INT_EXT3 - Interrupts/ Reading the cap values. In order to read data from the cap chip we have used the EXT3 which is connected to the RDY' pin of AD7746. When a new data value is available, the RDY' signal makes a High to Low transition. This logic gets inverted in the I2C isolation circuit. The interrupt routine reads 24 bit data from the capacitance data register and stores it in the capValue variable.

Startup()- A startup routine which toggles the LEDs and a makes the buzzer go on. This is small program for us to see the program initialization.

Buzz()- A program which takes the time input for the buzzer. The time input is in milliseconds. Once the required delay is achieved, the buzzer is turned off.

3.8 PCB design

We used CADSOFT's EAGLE software for our PCB design. The first step in making a PCB is in creating a schematic. We have created blocks of the various circuits. The components that are required are obtained from the libraries. In the case that the required components are not available, custom components needed to be created. The blocks that are present in our design are described below.

Real time clock- We are using a DS1307 surface mountable RTC by Sparkfun. It is connected through I2C to dsPIC30. We use a 32.7kHz crystal oscillator for it.

SD card - This is a socket for the SD card.

I2C isolation- Isolation circuit for the I2C interface between the capacitive chip and the dsPIC30.

Power- Provides the isolated power supply for the isolated components in the circuit.

USB and ESD- Mini USB connector which is connected to the PIC18F87J50.

XTAL's -Crystal oscillator for the PICS.

Cap chip and SMB connections- The AD7746 and its SMB connections are included in this part.

I/O- The buzzer for notification of fault conditions and LEDs for display are included in this part. We have also included some connector pins which will be useful in debugging.

dsPIC30- The dsPIC30 is included in this block with all its connections. The microcontroller would run algorithms to detect abnormalities in breathing and hence detect sleep apnea

PIC18F67J50- The PIC18F67J50 is included in this block with all its connections.

Blood leakage detection- The connections to the liquid detection conductors have to be isolated. Hence we have used an optocoupler to isolate these signals. We have seen the operation of the liquid detection earlier. Also in order to prevent any signal interference, we are using coaxial cables with BNC connectors to connect the PAD to the PCB.

Iphone Connector- Given the latest trend to run applications via the iphone, an iphone connection was also placed in the PCB design. This will further expand the range of applications and uses of our device.

Dual PIC reset switch- We have included a switch so that we can reset both the PICs on board through a single push button. When the master clear goes low, both the PICs get reset.

3.8.1 Ground layer for pad signal layer

The signals coming from the pad are very sensitive, especially the capacitance values. A slight distortion in the signal could change the values being read and hence affect the outputs of the PICs. Hence we are using coaxial cables with BNC connectors to protect these signals from distortion. However, we have to take care that these signals do not get mixed with noise due to other signals on the PCB. Hence we have provided a ground layer under the pad signals.

In order to do so, we have to restrict the routing of these signals to the upper layer alone. Eagle provides an option to restrict routing of signals to a particular desired layer.

3.8.2 Placement and routing of components on board

Once the schematic (refer section 8.1) is made, we have to place all the components on the board. Initially we placed the components such that we can minimize the size of the board. However on routing, the components had to be shifted around so that all the connections are made.

3.9 Conclusion

The sleep apnea detection platform consisting of a custom made PCB for researcher purposes was successfully developed. The device measures respiration without any wired connections to the body. The pad also has an additional safety feature of liquid detection by which it detects any liquid flow on the pad.

Chapter 4.

Monitoring and Debugging of Wireless Sensor Network using Assertions

Conventional monitoring for health studies would require a constant stream of the signals being sent to the base node to monitor this condition. As indicated earlier, the amount data that has to be transferred is very high in the case of clinical health monitoring. Also wireless sensor nodes are critically constrained by their battery life. Moreover, in order to address debug ability, the sensor needs to send additional data to ensure the proper internal functioning of the sensor node. Hence this approach of sending a continuous trace of different vital signs and internal state variables would result in quick dissipation of the battery.

It is here that assertions come into play. Assertions address the twin problems of monitoring and debug. While defining assertions, we can define assertions specific for monitoring the vital signs. These would catch any critical conditions of the patient. On detecting an abnormal condition, the assertions would fire and trigger the alarm. This would cause the sensor to send an assertion debug packet to the base station to alert those responsible. The assertions defined to debug the sensor operation would be triggered if the concerned portion fails the assertion. This would once again result in the sensor sending a debug packet containing information about the failed portion of the sensor. The debug packet gives us sufficient information to repair/replace the failing node.

4.1 Sensor Model

At the transaction level, a designer can test proper functioning of modules and the communication process separately. The designers have the freedom to later synthesize it in either hardware or software. In SystemVerilog, the interface class along with the function/task calls helps in creating systems at the transaction level. Different high level transaction strategies are used to verify the proper functionality of the sensor modules with a goal of debugging the sensor network during different failures. The failure of a node cannot be identified in a completely passive manner as passive inspection cannot determine the internal state of a sensor. The authors of [1] apply Passive Distributed Assertions (PDA) where the sensors keep sending trace messages, which are heard by a sniffer network in an online approach. After analyzing the trace messages they determine the possible failure causes. This method relies critically on the sniffer network, which cannot fail and must deliver the data constantly. We have modified this approach by implementing the assertions within the sensor itself. Our model does not send a constant trace message of data. The assertion data is transmitted only when the assertions in a sensor fail.

We have created a general model for the sensors inside our environment. Inside each model we can specify the sensing elements. Thereafter, the particular groups of assertions are generated and assigned to track the value of every sensing element. We have customized our general model to have the functionalities of biomedical sensors. A biomedical sensor contains different components for sensing different vital signs. Hence we have assigned these vital signs to the sensing elements of our model. Then related group of assertions which are in charge of checking vital signs along with other embedded assertions are created in our environment.

4.1.1 Transaction Model

As mentioned, each node in our system senses the vital signs like breathing rate, temperature, etc. These values are updated periodically and stored on the node.

4.1.2 Interface

The wireless channel was modeled with the use of SystemVerilog interface construct. By instantiating the interface only once in the top level module and subsequently passing it as the port of all sensors, we effectively model the wireless channel.

An interface allows to group different signals together. Each module that uses this interface then has a single port of the type interface instead of having multiple ports [7]. With respect to sensor networks, we can have a single communication channel (air interface), but can support different communication protocols (for example: Bluetooth and Zigbee). Our interface is based on the Zigbee protocol to send both assertion data and sensor communication; however, we can easily modify it to Bluetooth without any major changes to the sensor modules.

As shown in Figure 12, the interface supports two types of packet transmissions- debug packet and data packet. The debug packet is used to send assertion information and the data packet for sensor communications. Both of them have a header portion with source, destination and type of message fields.

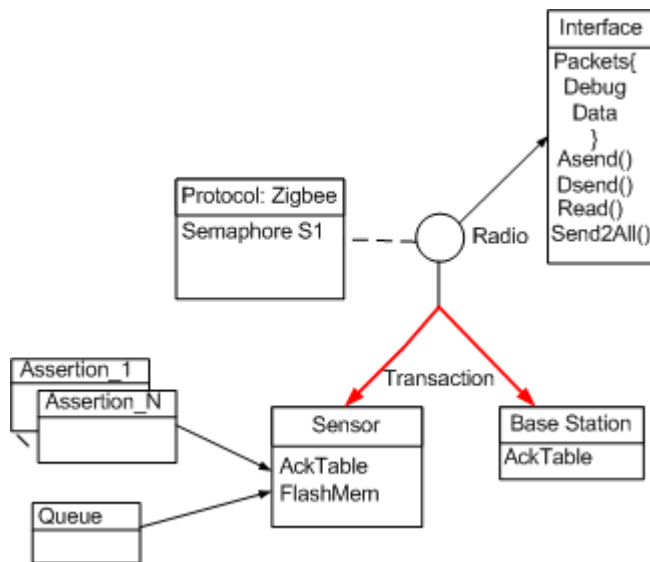


Figure 12. Transaction model of sensor network

4.1.3 Packet Structure

4.1.3.1 Debug packet

Apart from the header, the debug packet has a payload field which contains assertion id, assertion time and system status. Keeping in mind that the assertion data send should not alter the sensor network drastically, we try to keep the assertions data as small as possible and send them sporadically. Assertion id is a unique id assigned to each assertion that is being employed. Instead of sending a string of text to identify the assertion, we have used assertion ids to identify the fired assertion. Currently we are assigning the assertion ids manually. However, this can be automated by running a script to assign unique ids prior to compilation.

The assertion time is the time at which the assertion failed in the sensor. The system status field holds the value of the variable that failed. For example, say that the temperature assertion (id=1) failed in sensor2 at time 80s with a temperature value of 105F. The assertion packet to the base station would be [2,0,1] [1,80,105].

4.1.3.2 Data packet

The data packet consists of the header field followed by the payload field. The payload is an integer array of size 4.

4.1.4 Protocol

We have modeled the wireless communication based on Zigbee protocol as it is widely used for low power sensor networks. In order to emulate carrier sense collision avoidance (CSCA) in Zigbee, we have utilized the semaphore construct in SystemVerilog. As depicted in Figure1, the semaphore is part of the interface. The semaphore has been defined to have only one key; hence, we make sure that only one device can communicate at a time. The interface also contains tasks to perform operations like sending and reading the sensor packets.

4.1.5 Sensor Module

Apart from the ports, defined by the interface, the sensor module has additional members like flash memory, data queue and an acknowledge table.

As mentioned earlier, the sensor modules store the vital signs periodically in memory. Every sensor maintains its own time. However, we assumed that the sensors are time synchronized using protocols like FTSP. This however does not affect our assertions as they are fired within the sensor as opposed to the scenario with PDAs [1], where the assertions are fired at a base station after receiving the trace messages.

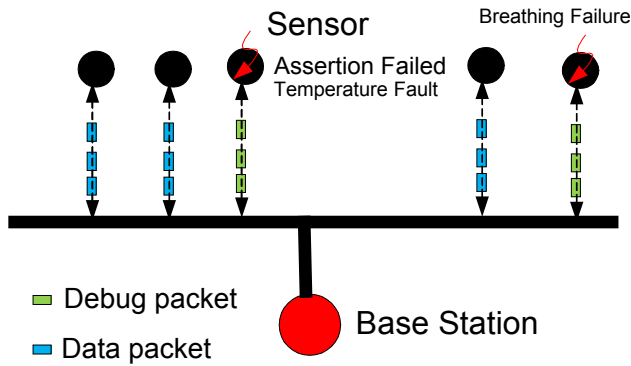


Figure 13. Our proposed environment

4.1.6 Communication

As Figure 13 shows, a sensor can send two types of messages: debug message to the base station and sensor data to other sensors or base station. The type of message can be distinguished from the information in the header field. When a message is to be sent, it is first pushed into the queue. The messages are later pushed out of the queue and send whenever the communication channel is free, i.e. whenever the semaphore key is available.

Each sensor has an acknowledge table which is used to store the pending acknowledges to be received from other nodes. The acknowledge table has two fields: destination node and time. The acknowledge assertion continuously checks that all entries in the

acknowledge table have not exceeded the wait period for a response. As it was pictured in Figure 13, if the wait period is exceeded, then the corresponding assertion fails and the node sends the debug packet to the base station informing the situation. The offending entry is also removed from the acknowledge table.

4.2 Assertions

We have employed the concurrent assertions in SystemVerilog to test the various critical conditions. The concurrent assertions have the following layers- Boolean, sequences, property, property directive. The Boolean layer is contained within the sequences to form a linear sequence. The property layer is built on top of the sequence layer. Finally the property directive checks the properties. We have used the assert directive to ensure that the defined properties holds. For example, in our model, we have defined the following property assertion to detect breathing breaks every 10 time units. This particular assertion is based on a sleeping disorder called sleep apnea which is characterized by disturbances in sleep due to pauses in breathing. Each episode, called an apnea, lasts 10 seconds or longer [6].

```

1  sequence s2();
2      @(negedge clk)
3      (breath<10 or breath>25) [*10];
4  endsequence
5  property rule2();
6      @(negedge clk)
7      disable iff(as_on==0)
8      not s2;
9  endproperty
10 breath_assert : assert property (rule2)
11     else begin
12         $display("Breath Assertion failed");
13         asend(2,t,breath);
14         storeas({2,int'(t),breath});
15     end

```

Figure 14. Breath Assertion Description

The assertion will be fired whenever the *breath* variable in line 3 (Figure 14) goes outside the range continuously for 10 time units. When the assertion fails, the information is stored in the flash memory and also transmitted to the base station. Once received by the base station, the assertion information is saved and depending on the nature of the assertion, corrective measures can be taken. In the case of sleep apnea study, this information makes it easier to detect the time points when the patient had an apneic event instead of going through the entire sensor readings and manually detecting the occurrences.

4.3 Results

To evaluate our assertions based model, we compared it with one where the assertions are turned off. On doing so, the sensor has to continuously transmit its state information to the base station. Since we are monitoring in sleep apnea the vital human health information that changes continuously, the sensor node was modeled to send this data every 10 seconds. Our assertion based sensor node sends data only when the assertion fails.

TABLE 1. Number of Transactions for different simulation time periods

Simulation time period (s)	Sensor without assertions	Sensor with assertions	Decrease in transactions (%)
100	10	2	80
500	50	20	60
1000	100	53	47
2000	200	114	43
86400(1 DAY)	8640	4224	51.1
		Average	56.22

As it was depicted in TABLE 1, our proposed debugging environment has fewer transactions. In the case of Zigbee, while sending a packet we have to consider collisions and subsequent retransmissions of the same data packet. The impact of fewer transactions is twofold; the network traffic decreases and correspondingly the probability of collisions occurring also decrease. Hence computing the assertions within the sensor, has two main advantages: i) reduce the number of packets that are transmitted ii) decrease the collision possibility.

4.4 Conclusions

Debugging of embedded wireless systems is a challenge as the amount of information available from them is limited. Moreover the number of transactions relayed to convey the state information is also critical as it interferes with the normal operation of the wireless network. A novel approach to support distributed assertions that detect critical conditions and also help reduce the network traffic is presented here. This improvement can be attributed to the elimination of trace messages as a consequence of inserting particular assertions within the sensor. A transaction level modeling for wireless systems based on SystemVerilog is also shown. The abstraction of the communication channel is achieved with the *interface* construct in SystemVerilog. This work was also presented at the 8th IEEE International NEWCAS conference in June 2010 [56].

Chapter 5.

Wireless polysomnography

The wireless radio is the most power hungry device on board of a sensor. Hence, sending a huge amount of data over the wireless channel would eat away a lot of critical battery power. In our studies we have observed that the health sensor data is repetitive over short spans of time. Based on this observation, we decide to use compression techniques on the data before transmitting it. Compression aims to reduce the amount of data to be transmitted and hence save on the energy consumed for the transmission. Conventional compression algorithms cannot be applied for small sensor devices as they do not consider the power constraint. Moreover, the memory sizes of these sensors are small and hence cannot implement memory intensive algorithms. Many previous works have touched upon compression in wireless sensor network [22][12]. However, these works do not look into efficient compression for health sensor data. It is with this motivation that we explored some lossless compression algorithms that are suitable for use with portable health sensors.

We studied the data obtained after performing a polysomnography of a patient for 3 hours. Sleep studies are categorized based on the number of physiological variables being monitored.

Sleep studies are categorized based on the number of physiological variables being monitored. The signals recorded for our study are included in Table 2 and are representative of a category 3 portable sleep study. Signal 1 measures the oxygen saturation (SPO2). Signals 2 (ABDO) and 3 (THOR) measure the thoracic and abdominal

respiratory movement. Signal 4 (EEG) measures the EEG activity and signal 5 (FLOW) measures the nasal flow.

Table 2.List of Signals and sample rates

Channel	Signal	Sample Rate
1	SPO2	200 Hz
2	ABDO	200 Hz
3	THOR	200 Hz
4	EEG	200 Hz
5	FLOW	200 Hz

5.1 Delta encoding

The simplest approach to compressing related data is by applying a differential encoding. This scheme takes advantage of the fact that there are only small changes occurring in the sensor data and hence this greatly reduces data redundancy [24]. Delta encoding is conventionally applied for lossy compression. However, we can make delta encoding lossless by including an overflow bit. The overflow bit will be set when the delta value is beyond the predefined delta range. On overflow, the overflow bit is set and is followed by the full data reading for that point. Since such fluctuations do not occur very frequently, we manage to get considerable compression.

After analyzing the data, we realized that the difference between consecutive data points in all the signals is mainly in the range of $\{-15 \ 15\}$. Hence we select a 5 bit representation for the data which is preceded by an overflow bit. In the case of an overflow, the 8 bit sampled data along with a high overflow bit will be send.

5.2 Sensor-Lempel–Ziv–Welch (SLZW)

SLZW is the adapted version of Lempel–Ziv–Welch for resource constrained sensor nodes [22]. LZW is a lossless, dictionary based algorithm that builds its dictionary as data is read from the input stream.[20][21]. The algorithm is computationally simple as illustrated in Figure 15. For example consider the input stream to be “AAAABAABCC” (Figure 16). We take the first character “A” from the input and append it to the next character to get “AA”. We find that this combination is not present in the dictionary and hence add it to the dictionary. Hence we get a new entry of 256 which represented “AA”. The output for this stage is 65 which represents “A”. We now start a new string with the last character input “A”. The next input character is “A” and so the string is “AA”. This entry exists in the dictionary and hence we add the next input character “A” to the string. The string “AAA” does not exist and is added to the dictionary. However the output is 256 and we add the next input character “B” to the string. The string “AB” is not present in the dictionary and this is added to the dictionary. The output at this stage is 65. On continuing the algorithm we get the output stream as “65 256 65 66 257 66 67 67”. The uncompressed output stream for this input would have been “65 65 65 65 66 65 65 65 66 67 67”.

The initial dictionary consists of 256 entries and the new dictionary is built on the fly. This perfectly fits our model as we are considering an 8 bit ADC sampled data. Hence we can expect 256 different data appoints. We restrict the dictionary size to 512 entries as the memory available in these embedded systems is small. The key advantage of this approach is that the dictionary does not have to be transmitted as it can be recreated at the receiver with the initial dictionary and the compressed data. While considering sensor networks, we have to consider the dictionary size, size of data to be compressed and the protocol to be followed. The size of data to be compressed depends on the entropy of the data. Hence in order to determine the best configuration of data to be compressed, we perform simulations for different sizes of data and different nature. For our evaluation of SLZW we consider the ABDO signal which varies slowly and the EEG signal which fluctuates much faster. The data to be compressed will be present in the flash memory of

the sensor. Hence the size of data to be compressed is based on the flash page size. We consider a flash page size of 512 bytes.

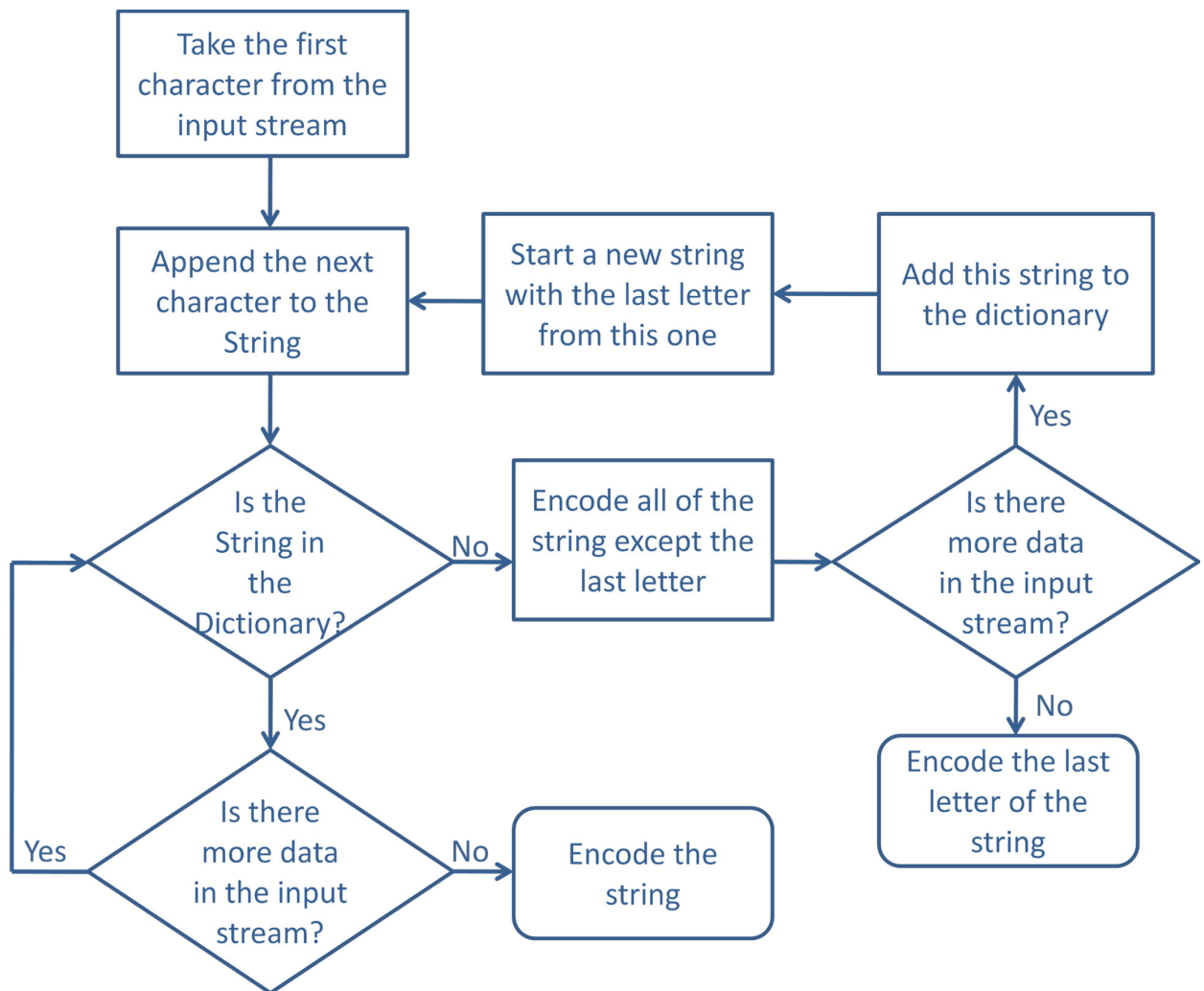


Figure 15. Flow chart of LZW compression

Input Stream: AAAABAAABCC

Encoded String	Output Stream	New Dictionary Entry
A	65	256-AA
AA	65 256	257-AAA
A	65 256 65	258-AB
B	65 256 65 66	259-BA
AAA	65 256 65 66 257	260-AAAB
B	65 256 65 66 257 66	261-BC
C	65 256 65 66 257 66 67	262-CC
C	65 256 65 66 257 66 67 67	

Figure 16. Example of LZW compression

Table 3. Data size consideration for SLZW with 512 entry dictionary

Data size per compression operation	Compression ratio	
	ABDO	EEG
1 page (512 bytes)	3.8872	2.3557
2 pages (1024 bytes)	4.3954	1.2876
4 page (2048bytes)	4.1848	1.0037
10 page (5120 bytes)	3.0125	0.9595

Table 4.Compression percentage for dataset 1

Compression Algorithm	Compression Ratio (%)		
	SPO2	ABDO	THOR
Delta encoding	1.3333	1.3333	1.3333
SLZW-512 dictionary-2 pages	20.2311	4.3954	4.6309

Table 5. Compression percentage for dataset 2

Compression Algorithm	Compression Ratio (%)	
	EEG	FLOW
Delta encoding	1.3251	1.3332
SLZW-512 dictionary-1 pages	2.3557	1.6895

5.3 Compression results

The results shown in Table 3 compare different input data sizes for the compression. The signals under consideration were ABDO and EEG from Table 2. From the table, it is clear that SLZW gives best results with an input data size of two flash pages for the case of ABDO signal. SLZW performs better with a flash page size of one for the EEG signal. This result can be attributed to the fact that the dictionary fills up optimally for ABDO when we consider an input size of 2 flash pages. Since the data varies faster for EEG signal, the dictionary also fills up faster. Hence for a larger input size, the previous

dictionary entries are not repeated and results in the transmission of the original data. We can conclude from Table 4 that SLZW gives optimal compression for a steadily varying signal with an input data size of 2 pages. Rapidly fluctuating signals compress optimally with an input page size of one.

Table 4 and 5 compare the compression results between SLZW with a 512 entry dictionary and differential encoding. Table 4 considers signals which vary steadily. Table 5 consists of signals which fluctuate rapidly. The maximum compression achievable from the delta encoding is 1.3334, i.e. when no overflow occurs. From the tables above it is visible that the compression achieved by delta encoding is nearly the best possible by it. However, SLZW provides better compression results for all the signals. The high compression ratio for the SPO2 signal can be attributed to its steady values over long periods of time.

Chapter 6.

Debugging and monitoring for Network-on-a-chip using Assertions

NoC literature shows that future designs will include networking features currently found only in dedicated network interfaces used to connect systems. The ability to fragment packets into flits, which themselves are an ordered sequences of physical transfer units (called phits) is part of the advanced features that will be supported by future designs [50]. The building blocks of NoCs are thus optimized, on-chip network switches that have the ability to dynamically route flits from one network element to another. The switches themselves are circuits, which will benefit from hardware assertion checkers as mechanisms to validate their inner operation at runtime. Complex, unanticipated network scenarios may trigger assertion failures that were not planned in the verification and that could lead to interesting bugs being discovered.

Goossens et.al describe the integration of register structures in a communication-centric infrastructure in detail [51]. The work done by our group combine some of the data abstraction and temporal abstraction by providing a state indicating violations in complex temporal protocols. Our approach proposes the creation of register files with logically grouped assertion and monitors, but the outputs of the assertion checkers and control registers could very well be integrated as part of the monitors in the InCiDE design flow [51] and their interpretation performed by a software-based debugger instead of our proposed in-system solution. The work in this thesis covers the generation of assertions for this in-system solution for monitoring and debugging.

Vermeulen et.al [47] showed the importance of standardization of the interfaces required to observe and debug large scale integrated circuits. They note that the more complex designs will integrate electronic system-level (ESL) debug strategies at the core of the circuit and will use standardized interface ports to propagate the device state and execution traces. IP assertions are part of this future integration of debug structures in their SoC debug strategy.

In most designs, especially those like NoCs that require some level of in-system monitoring (flit retry, abort counters), designers often write hardware blocks and then integrate those to the system memory map. Then device drivers are written by the software team to read those specialized registers and then extract useful performance monitoring information. PSL or SVA cover statements can alleviate much of the work involved in adding performance counters in a distributed system. In the methodology presented by [52] , the designers simply have to include cover statements destined for silicon implementation and can then leverage the automatic insertion of those performance counters in the memory map ordered by the automated tool flow.

If only the knowledge of pass/fail is required, OR-ing all the assertion checkers in a group and storing the combined checkers' output will provide this information. However, the benefits of knowing which assertion in the group has failed is likely to be very valuable despite its hardware overhead, so dedicated registers for each assertion output are the most interesting option if one also wishes to locate the root cause of a globally incorrect circuit.

6.1 Design approach

For simulating purposes, we consider a 3x3 grid with 9 cores where each core is connected to the network through a switch as shown in Figure 17. A switch has up to 5 connections: west, north, east, south and to the core with separate input and output ports for each (Figure 18). Each port has a *data-path* for transferring the data, a *valid* bit and an *acknowledge* bit. The *valid* bit is high when data has to be send through the output port. The *acknowledge* bit goes high to confirm the reception of a packet. Each input port also has two buffers- a data buffer and a debug buffer. The data buffer stores a data packet as it is received from the input port. The debug buffer stores the debug packet as it is

received from the input port. The debug buffer is given priority over the data buffer and hence packets from the debug buffer are processed faster. The switch also has an acknowledge table which keeps records the input port/buffer of the sent packets.

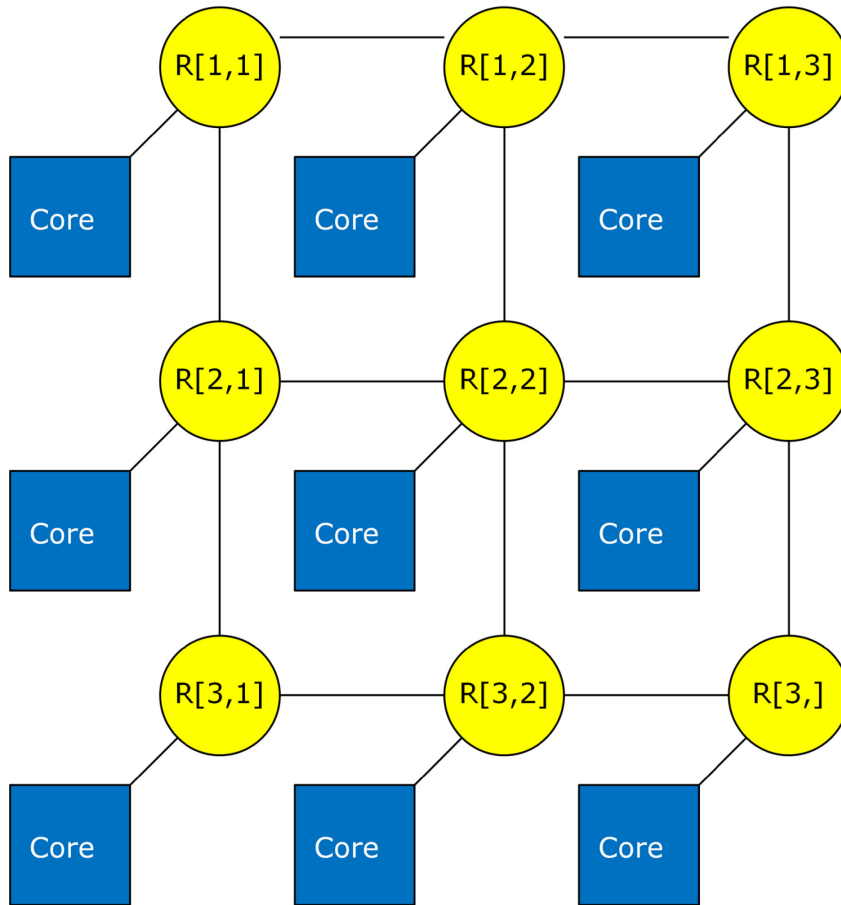


Figure 17. NoC simulation model

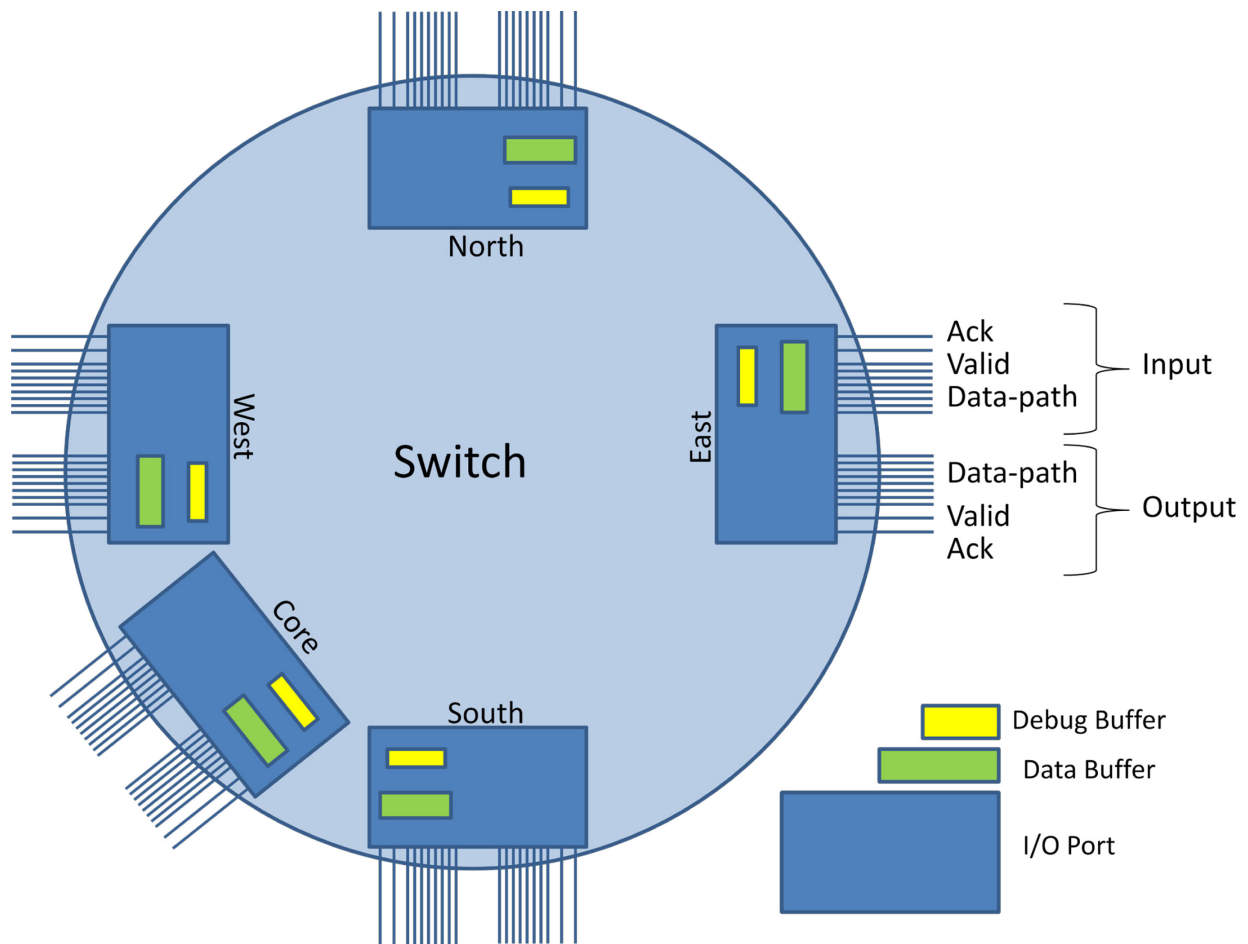


Figure 18. Switch Architecture

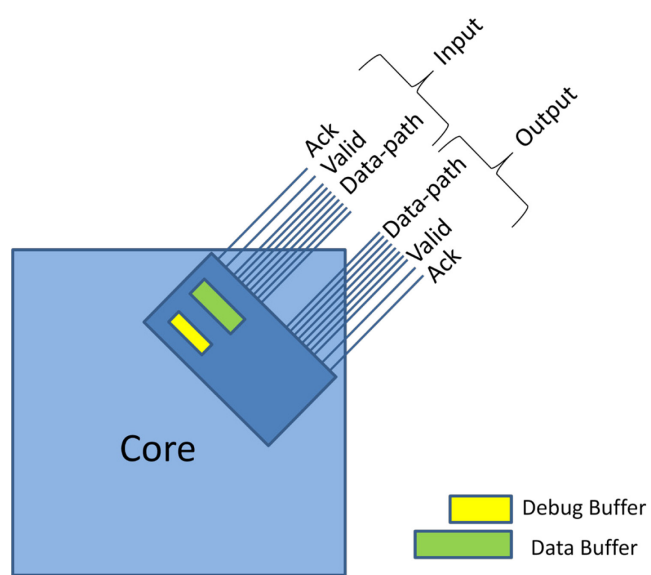


Figure 19. Core architecture

6.1.1 Communication:

The communication is packet-based. There are two types of packets: debug packet and data packet. Each packet has a header and payload field. The header field contains the following fields: source, destination, type, route and direction. The type field helps to distinguish between the debug and data packet. The route field helps to decide the type of routing. The direction field is used to store the direction the packet has to be forwarded after the routing cycle in the switch.

The debug packet is generated when a switch or core discovers a faulty condition. The payload field of the debug packet helps to identify the type of error, failing component and the time of failure. This information is then send over the network to other cores and switches which can then take corrective or fault tolerance measures.

6.1.2 Routing

We use XY-routing as the default routing policy. However in the event of failures of switches, we may have to change this routing policy to YX-routing to. This way we are able to bypass the faulty switch. Consider the example depicted in Figure 20 where the switch R[2,2] is faulty. Fault tolerance should enable the packet to reach its destination even on the failure of a single switch in the network. The switch in the routing policy ensures that fault tolerance is achieved.

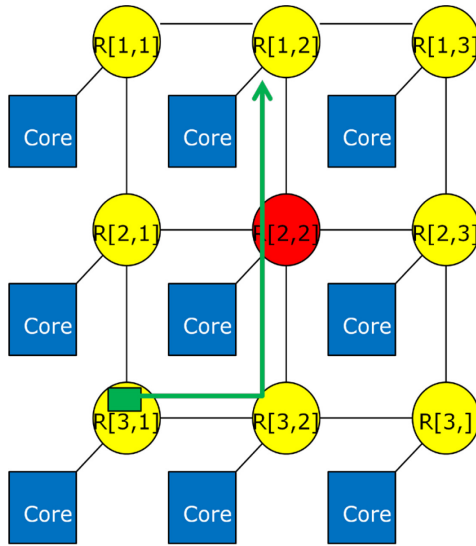


Figure 20 a. XY-Routing

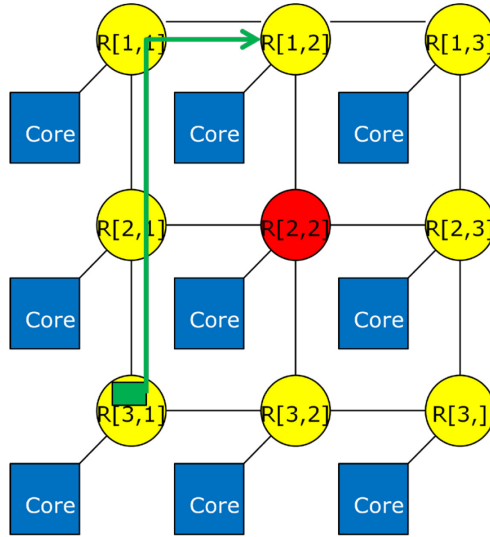


Figure 20 b. YX-Routing

Figure 20. Routing policy- Switch R[2,2] is faulty and hence XY-Routing will cause the packet to be stuck. YX-Routing will enable the packet to reach its destination

6.1.3 Operation

The switch has 3 cycles- *Buffer allocation*, *Routing*, *Switch allocation*. The *buffer allocation* occurs in the positive clock cycle. When data is received from any port they get stored in the respective buffers. If data is received then the acknowledge signal goes high on the output of the corresponding port. The switch then proceeds to the *Routing* operation of the received packet(s) in the same positive clock cycle. The routing policy decides the direction the packet has to be forwarded. This direction information is written into the *direction* field of the packet header. The ports which do not receive any packets will be in the idle state. The *switch allocation* is performed in the negative clock cycle. The switch checks that an *acknowledge* was received if the valid signal is high and deletes the corresponding entry from the acknowledge table and buffers. The assertions perform this check. The acknowledge handling is covered in detail in the section Ack-fail. The valid bit of all the ports is then lowered.

Switch allocation follows a round robin priority policy for the different ports. The debug buffers are processed before the data buffers. The packets at the top of the debug buffers of the different ports are allocated to the different ports depending upon the round robin policy. The switch then allocates the ports to the data packets if they were not previously allocated. The packets are not

deleted from the buffer until the next negative clock cycle upon reception of the acknowledge from the receiver.

6.1.4 NoC simulation performance

To assess the simulation model we inject traffic into the network. The traffic models cores' sending packets to every other core in the network. The effect of varying the injection rate (Probability of sending the packet) and varying the data buffer is studied through this simulation. From the tables below it can be seen that as the injection rate increases the latency also increases.

Table 6. Simulation model results for Data buffer size 4

Data buffer size	Total latency	Average latency	Prob of send	Core data memory size	Packets dropped
4	29258	4.06757	0.1	1000	0
4	30675	4.26042	0.3	1000	0
4	32239	4.47764	0.4	1000	0
4	34976	4.85778	0.5	1000	0
4	38967	5.41208	0.6	1000	0
4	67498	9.37472	0.7	1000	0
4	388249	53.9235	0.8	1000	0
4	783635	108.838	0.9	1000	0
4	1036274	143.927	1.0	1000	0

Table 7. Simulation model results for Data buffer size 5

Data buffer size	Total latency	Average latency	Prob of send	Core data memory size	Packets dropped
5	29088	4.0569	0.1	1000	0
5	30675	4.26042	0.3	1000	0
5	32082	4.45583	0.4	1000	0
5	34048	4.72889	0.5	1000	0
5	38376	5.33	0.6	1000	0
5	47213	6.55736	0.7	1000	0
5	273663	38.0087	0.8	1000	0
5	675770	93.8569	0.9	1000	0
5	908547	126.187	1.0	1000	0

6.2 Back-flow Assertion

For the purpose of debugging and monitoring we are utilising assertions. In this section we consider one such assertion for preventing a deadlock condition when the data buffers are full. From the simulation studies, we observe that at higher injection rates the latency increases and tends to saturate at some point. At these points, the buffers start getting full and the network would be prone to deadlocks. In order to prevent such a condition from occurring, we devised assertions which are fired when the data buffer is two-thirds full. This triggers the generation of the debug packet which is forwarded in the network to other switches and cores. On receiving this debug packet, the switch waits for a few clock cycles before sending packets in this direction. The cores also do not send packets in this direction. This way we give sufficient time for the congestion to clear in the network. Once the wait period is over, the switches and cores continue normal operation. This condition is depicted in Figure 21.

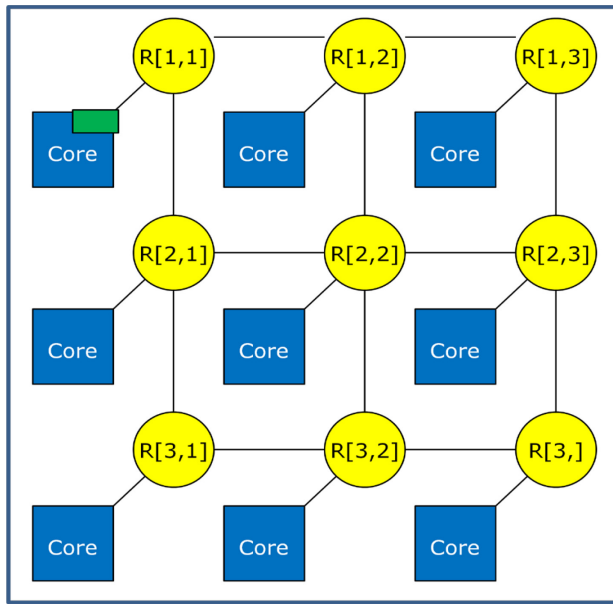


Figure 21a. Core sending packet to switch R[1,1]

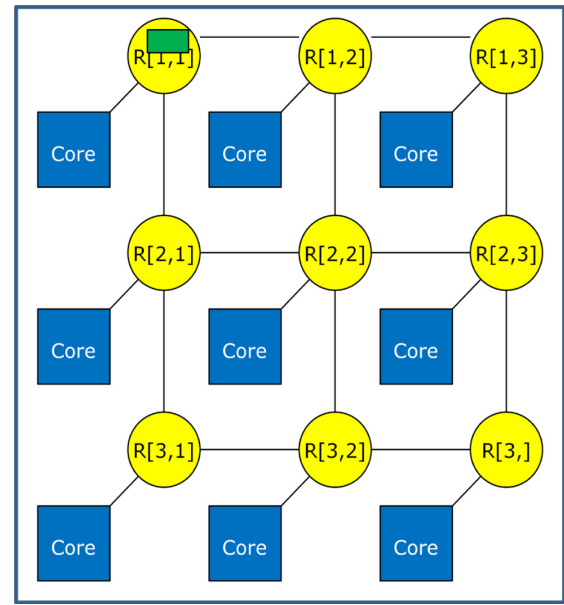


Figure 21b. Switch R[1,1] sending packet to R[1,2]

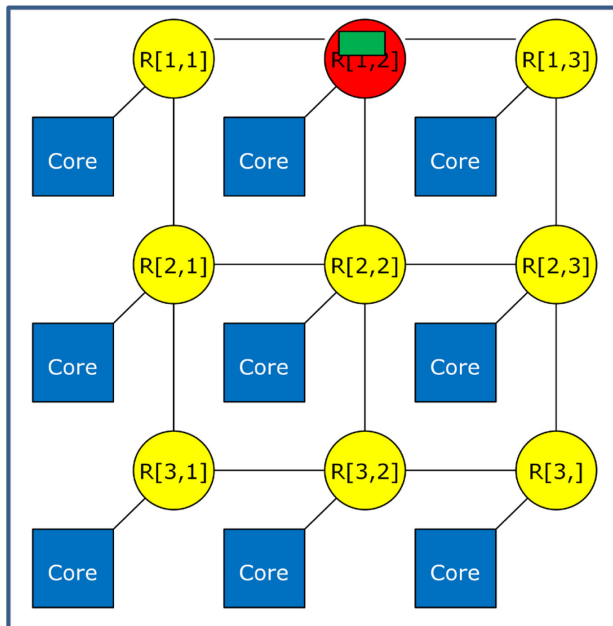


Figure 21c. Assertion fired in R[1,2] and debug packet generated

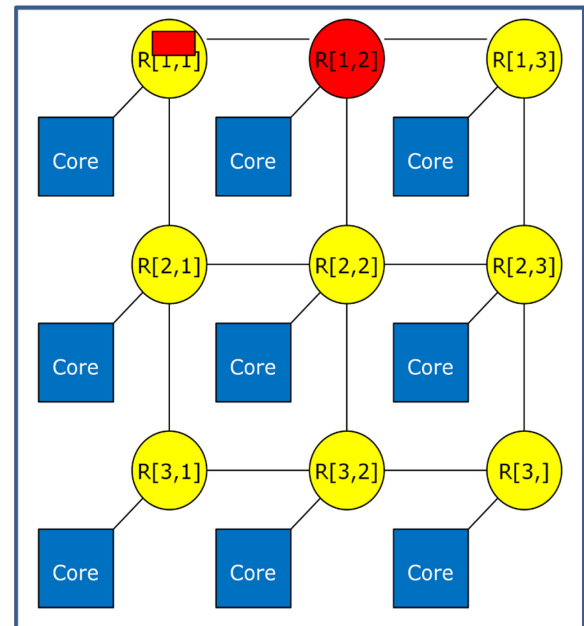



Figure 21 d. Debug packet received by R[1,1]

 Data Packet

 Debug Packet


 Buffer Overflow
in switch

Figure 21. Back-flow control using assertions

6.3 Acknowledge fail

Here we consider the case when the Core fails to send an acknowledge on reception of a data packet. The failure of the Core may be attributed due to software or hardware errors. The acknowledge table in the switch helps to map the sent packets to their corresponding input port buffers. The packets are not popped from the buffer until the switch receives an acknowledge from the receiver. When the acknowledge is received, the switch deletes the packet from the corresponding input buffer by looking up in the acknowledge table.

The acknowledge failure is depicted in Figure 22. Initially Switch R[2,2] sends a data packet to its core. The failure of the core to acknowledge causes the switch to generate a debug packet which is then propagated in the network to every other switch and core. Upon receiving the debug packet, the cores can avoid sending packets to the faulty core and continue with other operations. This way unnecessary traffic to the faulty core can be avoided. Without our assertion monitors, the situation goes undetected and leads to the overflow of the data buffer in switch R[2,2]. This would eventually cause congestion and saturation of the network.

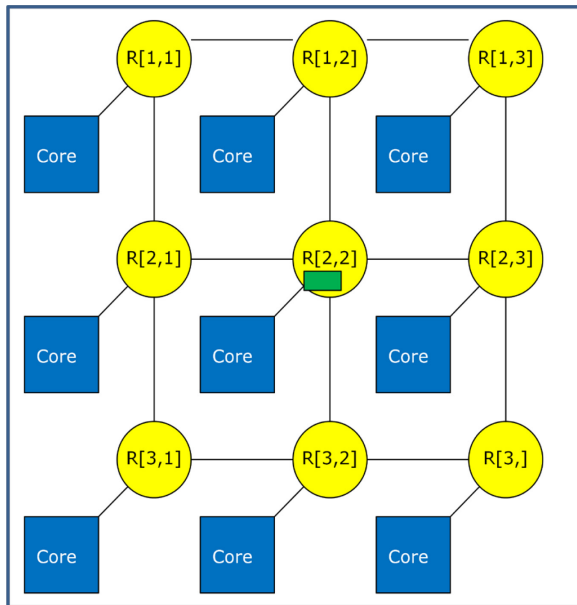


Figure 22 a. Switch R[2,2] sending packet to its Core

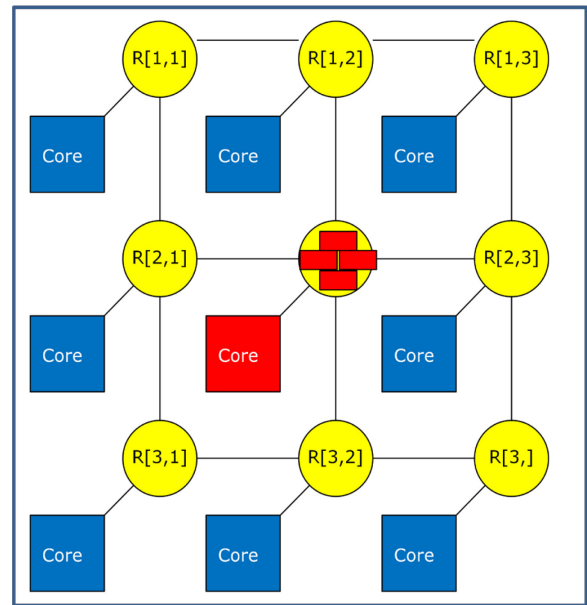


Figure 22 c. Switch R[2,2] sends debug packets to its neighbouring switches

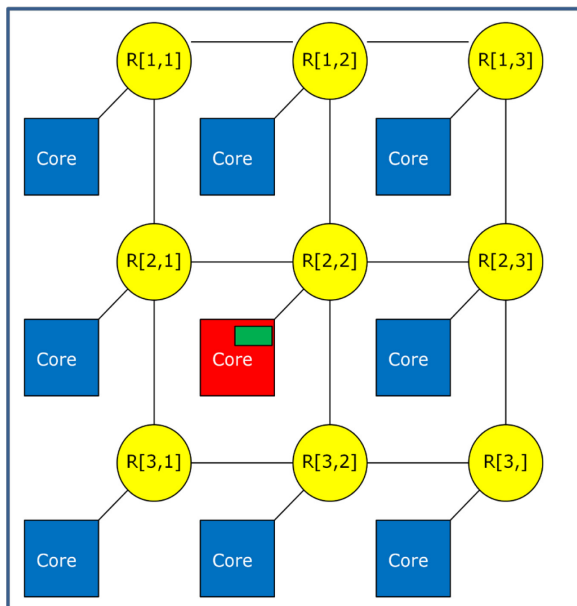


Figure 22 b. Core fails to send acknowledge to switch R[2,2]

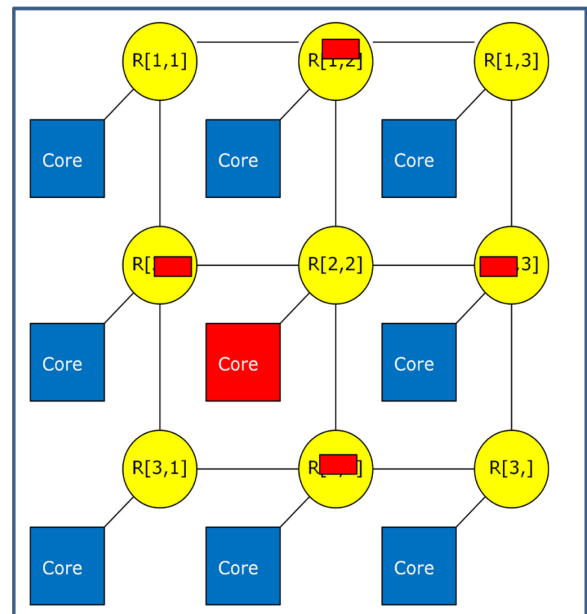


Figure 22 d. Switch receive debug packets and further disperse them

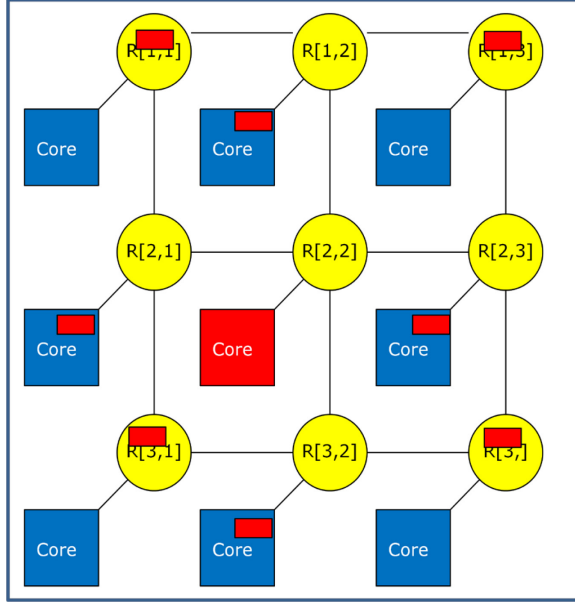


Figure 22 e. Switches and cores receive debug packets and further disperse them further

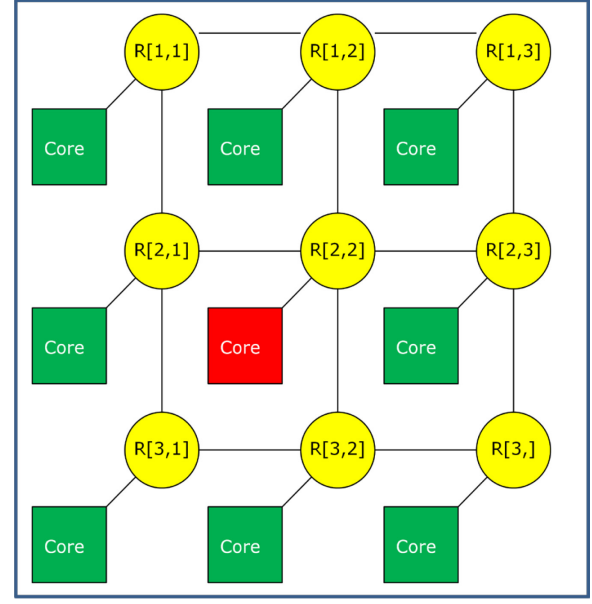


Figure 22 f. All the cores receive debug packets

Figure 22. Acknowledge failure

6.4 Results and Conclusion

The SystemVerilog Assertions for capturing acknowledge failures and buffer limit control were placed in the Switches and Cores. The simulations were compiled and run on Synopsys VCS tool. The Discovery Visualization Environment (DVE) present in it helped us to observe the different switches and cores. Figure 23 is a snapshot of the assertions in a switch. The green arrows indicate that the property is valid and the assertion has passed. The red arrows indicate the failure of the assertion. In Figure 23 we observe that the assertion has failed at two instances. At $t=1460s$ the Core_BFassert fails and at $t=1557s$ South_BFassert fails. Core_BFassert and South_BFassert represent the backflow assertion corresponding to the core and south direction respectively.

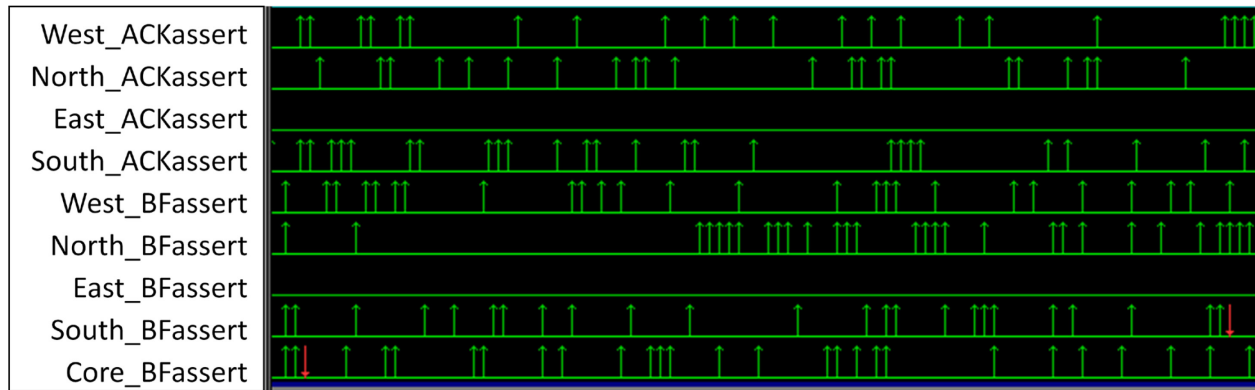


Figure 23. Snapshot of assertions fired

From our studies it is clear that assertion based approach to debug and monitor NoCs is very effective. The simulation shows that we are to pinpoint corner case problems. The designers can easily obtain a comprehensive view of the system by clubbing the assertion approach with suitable software backend tools. This methodology would help to considerably reduce the debug time.

Chapter 7.

Conclusion and Future Work

7.1 Summary

In this masters' thesis a portable sleep study kit was developed. The board has successfully shown to capture breathing measurements and detect liquid leakage detection. The kit is currently used for research purposes to study sleep apnea in hospitals. The thesis further explored the advantage of increasing the monitoring and debugs ability of wireless sensor nodes through the use of assertion modules within the sensor. It has been shown that the added hardware would enable to capture failures of nodes when deployed on field and identify the source of the error accurately. The proposed method also has an added advantage of decreasing the network traffic and help in conserving power. The Network-on-a-chip modeling and assertion based debug proved to be very valuable in capturing faults in the system. The packed based debug control also helps in providing fault tolerance to the network.

7.2 Future Work

The sleep apnea platform developed can currently only detect breathing and liquid leakage on the pad. The usage of the platform could be further increased to detect and measure vital signs. This could then be used as a complete sleep detection unit with increased application. The system

could also be equipped with assertion modules by including programmable logic on board. This would make the system more reliable and robust.

The assertion based monitoring of wireless nodes explored the working of wireless nodes with a Zigbee communication protocol. Since more and more wireless sensor networks (eg: Shimmer [8]) are now coming equipped with two radios, namely Bluetooth and 802.15.4, we would like to employ two protocols in our model also, one for sending assertions and another for sensor communication. This reduces the interference of debug packets with the sensor communication.

The Network-on-a-chip model has shown the usefulness of packing assertion modules within the switches and chips. The packet based control could be further utilized to provide a centralized monitoring of the network and its components. This could be achieved by a tool flow that packetizes the assertion checkers output for propagation in a NoC. The end result is that the NoC itself can run some assertion failure analysis by extracting the meaning of each assertion checker output signal and may react accordingly. We could also allow local CPU resources to access the assertion status information.

Chapter 8.

Appendix

8.1 Sleep apnea detection

8.1.1 Screen shots of schematic:

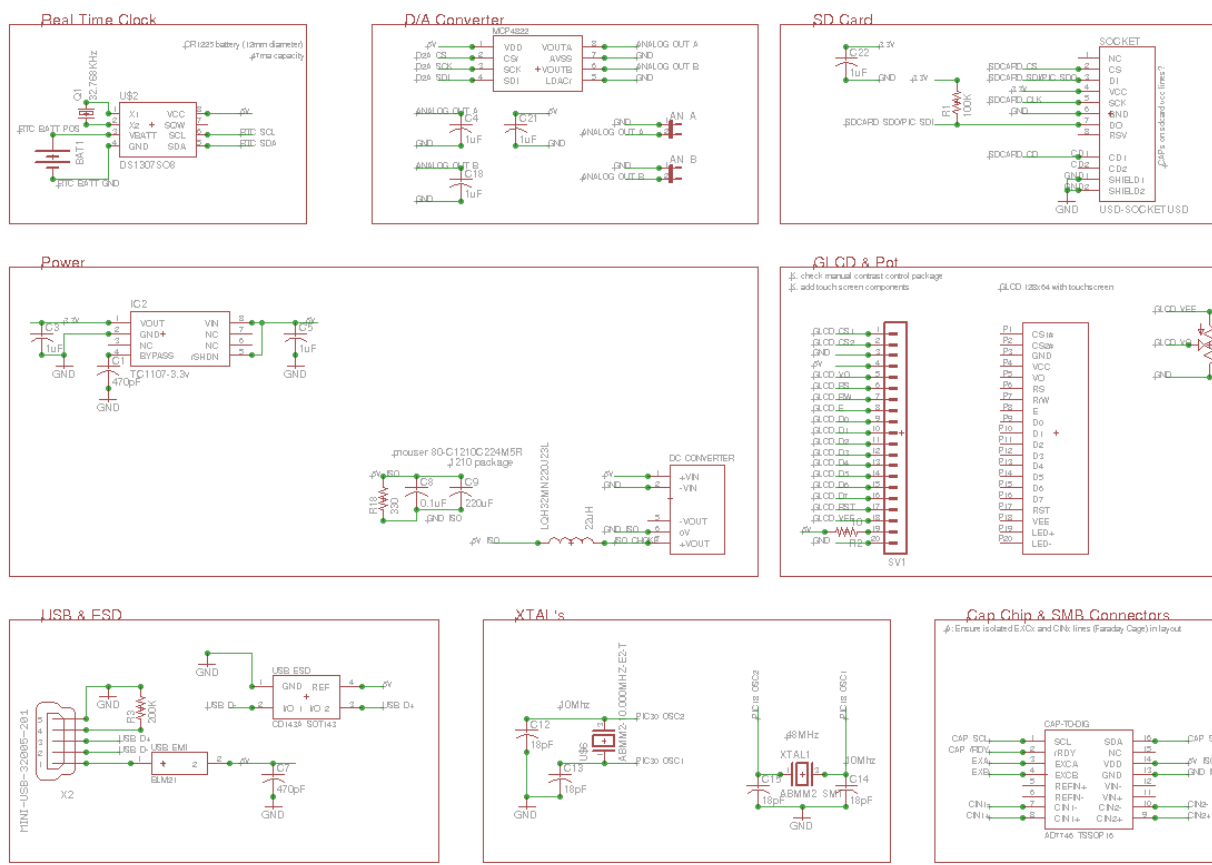


Figure 24. Screenshot of Power module

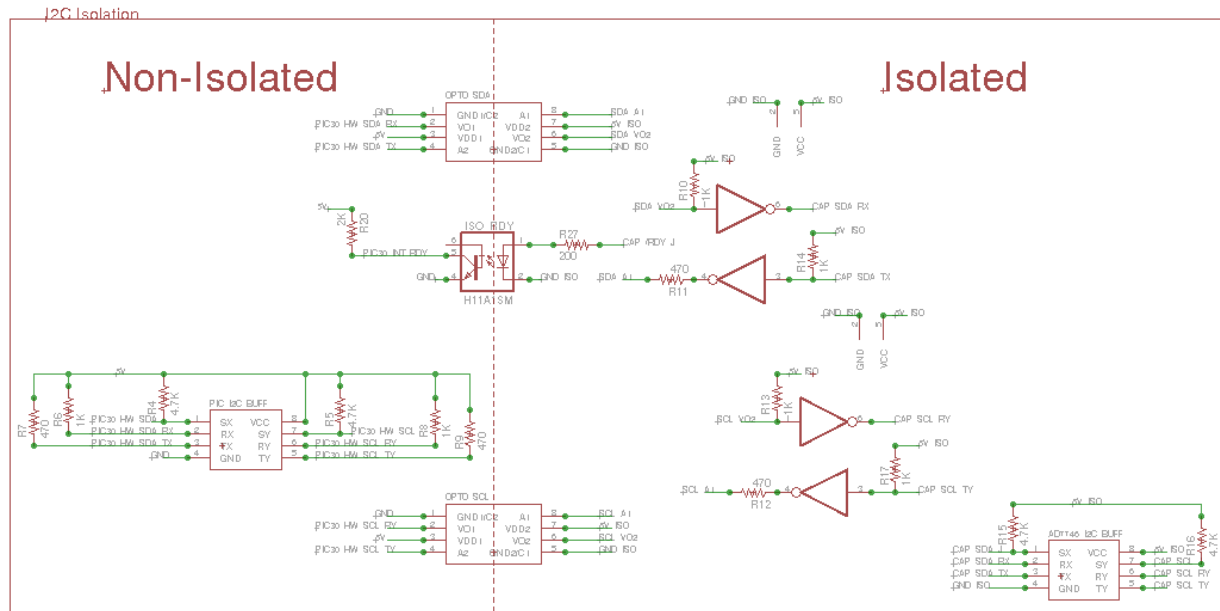


Figure 25. Screenshot of I²C isolation

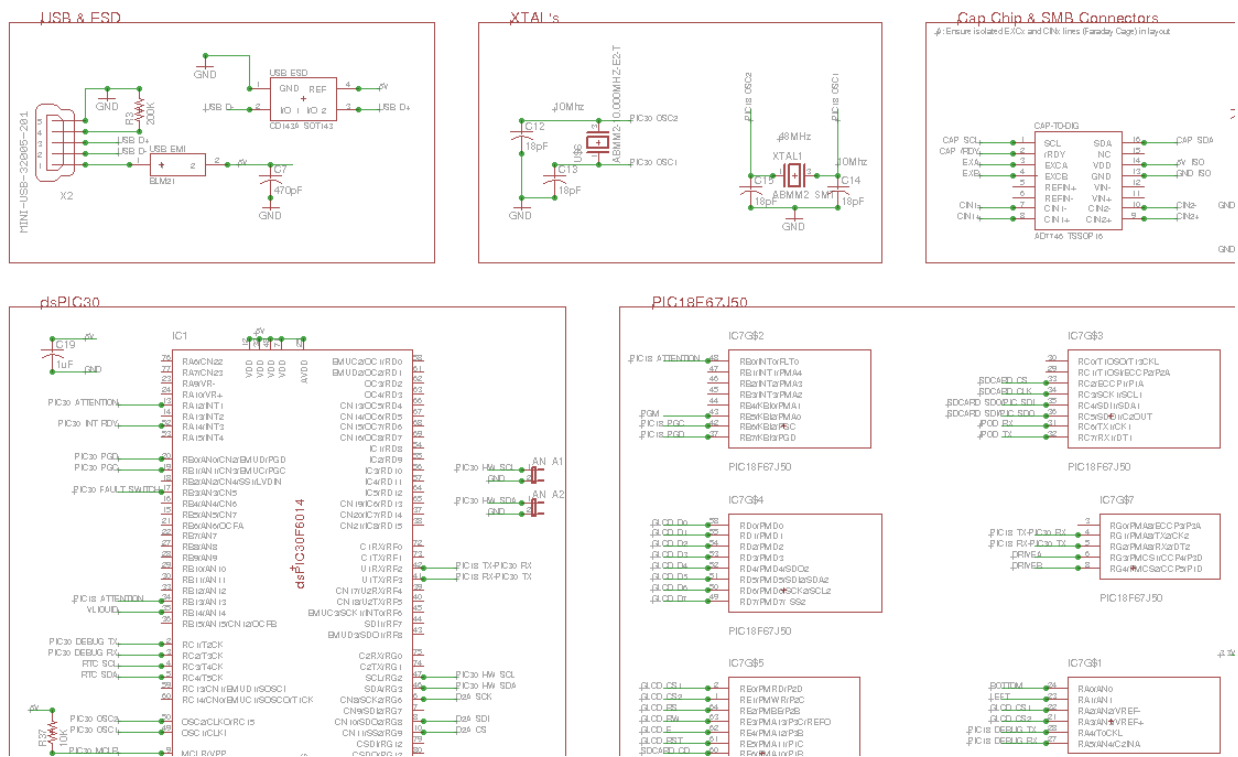


Figure 26. Screenshot of microcontroller units

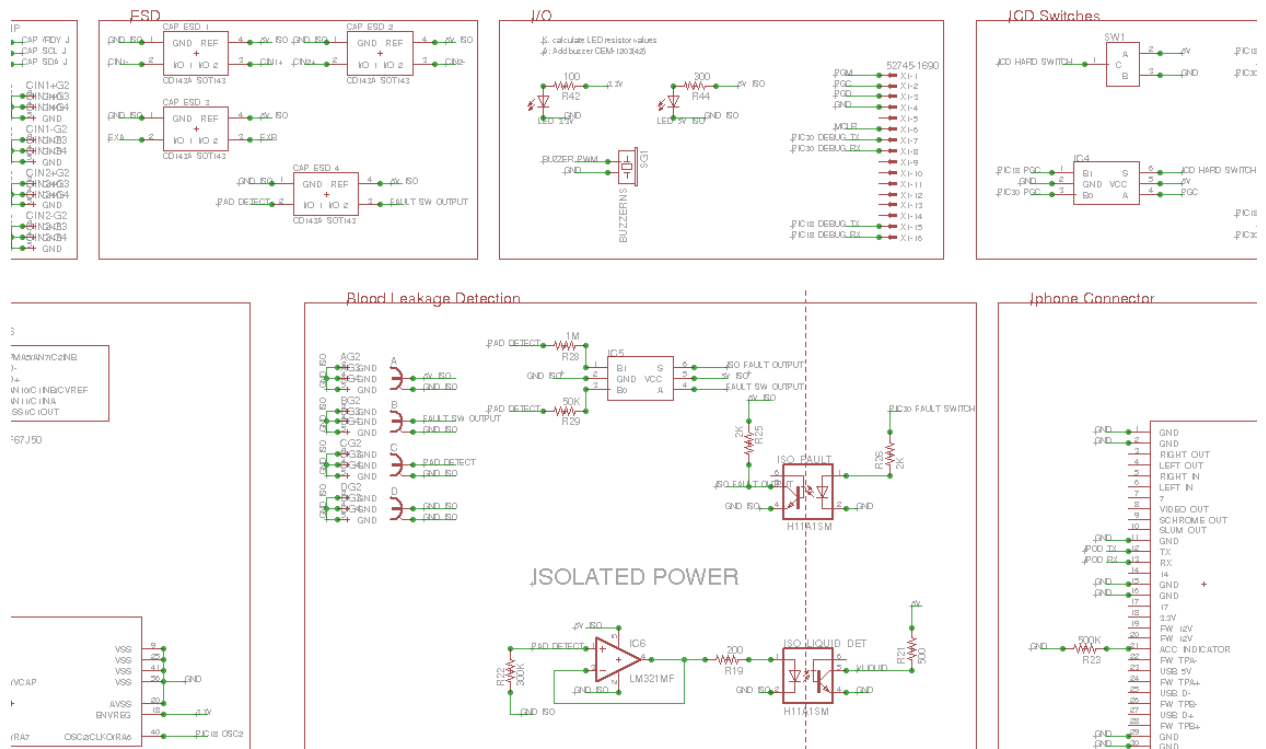


Figure 27. Screenshot of Blood Leakage Detection module

8.1.2 Screenshot of PCB

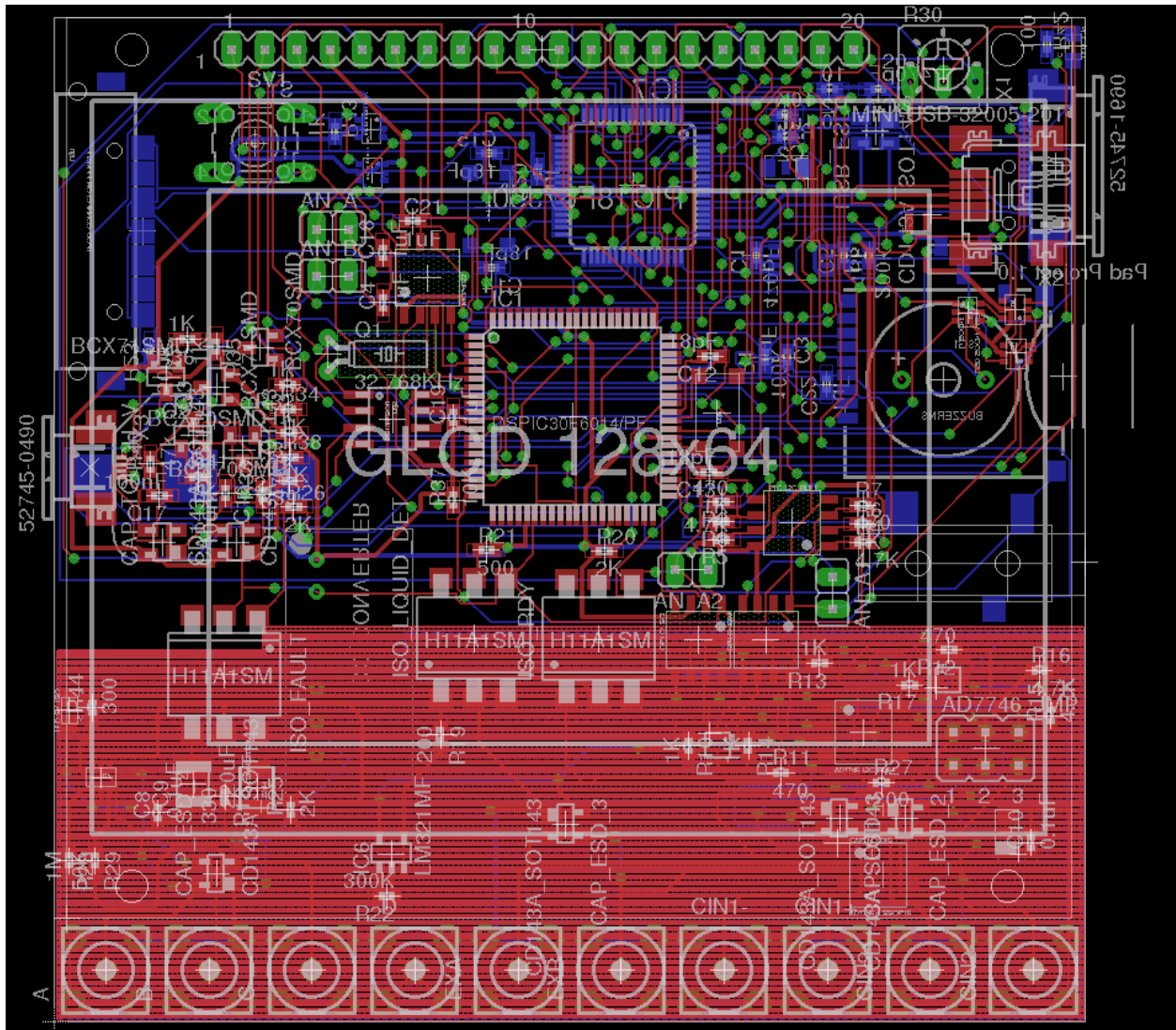


Figure 28. Screen shot of PCB

8.2 Ecomote Evaluation

8.2.1 Client Program

Accel.c – Reading Accelerometer

```
#include <eco/reg24e1.h>
#include <eco/eco_sys.h>
#include <serial/serial.h>

#include <adc/adc.h>
void store_cpu_rate(unsigned long hz);
void mdelay(unsigned int msec);
```

```

void serial_init(unsigned int baud);
char getc();
void puts(char *s);
void putc(char c);

void adc_init(char clk, char resol, char extref);
int adc_read(char in_pin);
void int_print(unsigned int val);

void main() {
    int i;

    unsigned int x_axis=0;
    unsigned int y_axis=0;
    unsigned int z_axis=0;

    store_cpu_rate(16);
    serial_init(19200); /* init serial */ // serial mode 1
    P0_DIR &= ~0x28; /* init led */
    P0_ALT &= ~0x28;

    EA = 1; /* global interrupt enable */
    ES = 1; /* serial interrupt enable */

    for(i=0;i<6;i++) { /* blink led 6 times to indicate startup*/
        blink_led();
        mdelay(500);
    }

    adc_init(ADC_CLK_D32,ADC_RES_10,EXTREF); // ADC_CLK_D32 is normally used

    while(1) { // infinite loop

        x_axis=adc_read(X_AXIS); //read x-axis;
        y_axis=adc_read(Y_AXIS); //read y-axis;
        z_axis=adc_read(Z_AXIS); //read z-axis;

        putc('x');// send 'x' before sending x-axis value
        int_print(x_axis); //sending x-axis value

        putc('y');// send 'y' before sending y-axis value
        int_print(y_axis); //sending y-axis value

        putc('z');// send 'z' before sending z-axis value
        int_print(z_axis); //sending z-axis value

        putc('e');// send 'e' to indicate end
        mdelay(500); // delay of 500ms

        blink_led(); // blink led
        mdelay(200);
    }
}

```

```
    } ;// end of while loop
```

```
    }
```

8.2.2 Host python program

Accel.py- Python script for accelerometer

```
import time
from uspp import *
tty=SerialPort("COM3", None, 19200)
# None: For blocking readings.
# params= default values for the number of bits
# per byte (8), the parity (NOPARITY) and the number of stop bits (1)
print('Displaying the accelerometer values');
print('x-axis y axis z axis');
xd=""
while 1:
    x=tty.inWaiting()
    if x>=1:
        #reading data
        xd+=tty.read(x)
        #find 'x' from the data
        c=xd.find('x')

        if c>=0:
            if c>0:
                #Remove all the characters before x
                #Incase we miss the start of transmitted data
                temp1=xd[c:]
                xd=temp1
            #To read data till we get 'e'= end of transmission
            reachedz=1;
            while reachedz==1:
                cond=1
                while cond==1:
                    w=tty.inWaiting()
                    xd+=tty.read(w)
                    if xd.find('e')>0:
                        reachedz=0
                        cond=0

            #find the values and display analog value (mV)
            posx=xd.find('x')
            posy=xd.find('y')
            xa=xd[posx+1:posy]
            gx=(int(xa)*3)/(2*2^10)

            posz=xd.find('z')
            ya=xd[posy+1:posz]
            gy=(int(ya)*3)/(2*2^10)

            pose=xd.find('e')
            za=xd[pose+1:pose]
            gz=(int(za)*3)/(2*2^10)

            print "%d %d %d"%(gx,gy,gz)
            temp2=xd[pose+1:]
            xd=temp2
```

8.3 Shimmer Evaluation

8.3.1 EyesWeb Accelerometer patch

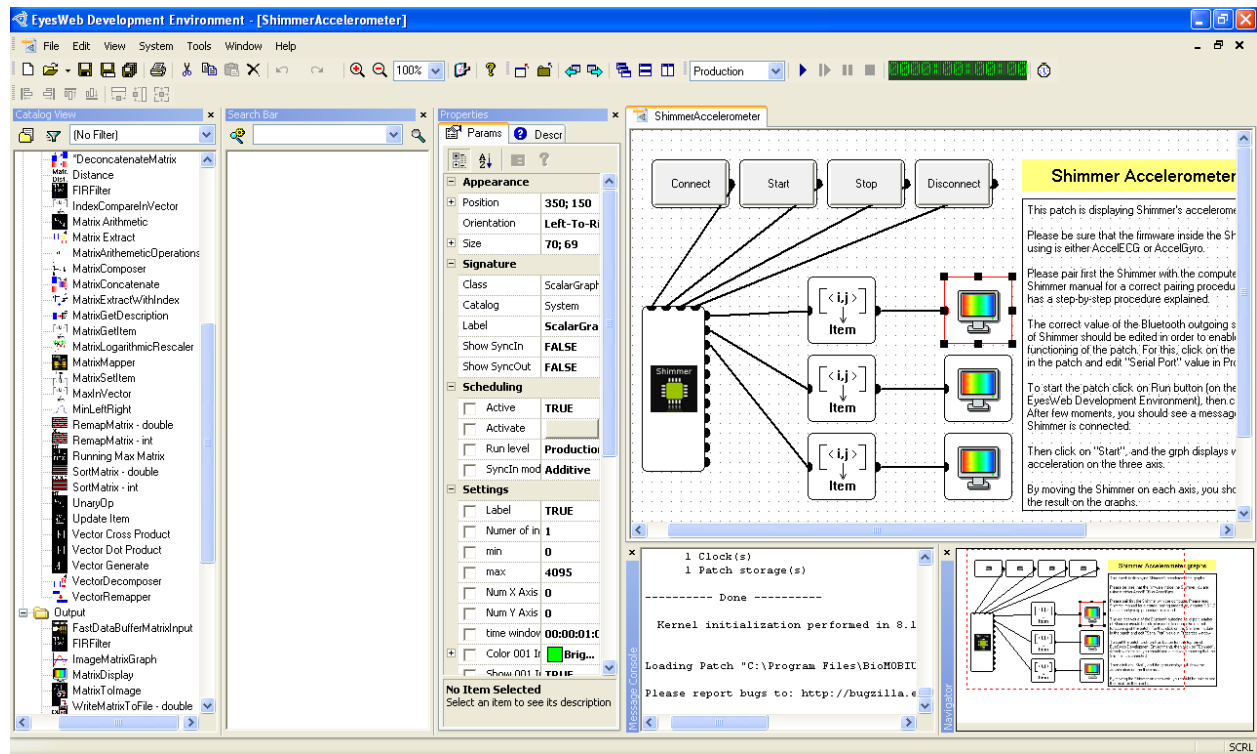


Figure 29. Screenshot of accelerometer patch

8.3.2 Biomobius GUI for Accelerometer patch

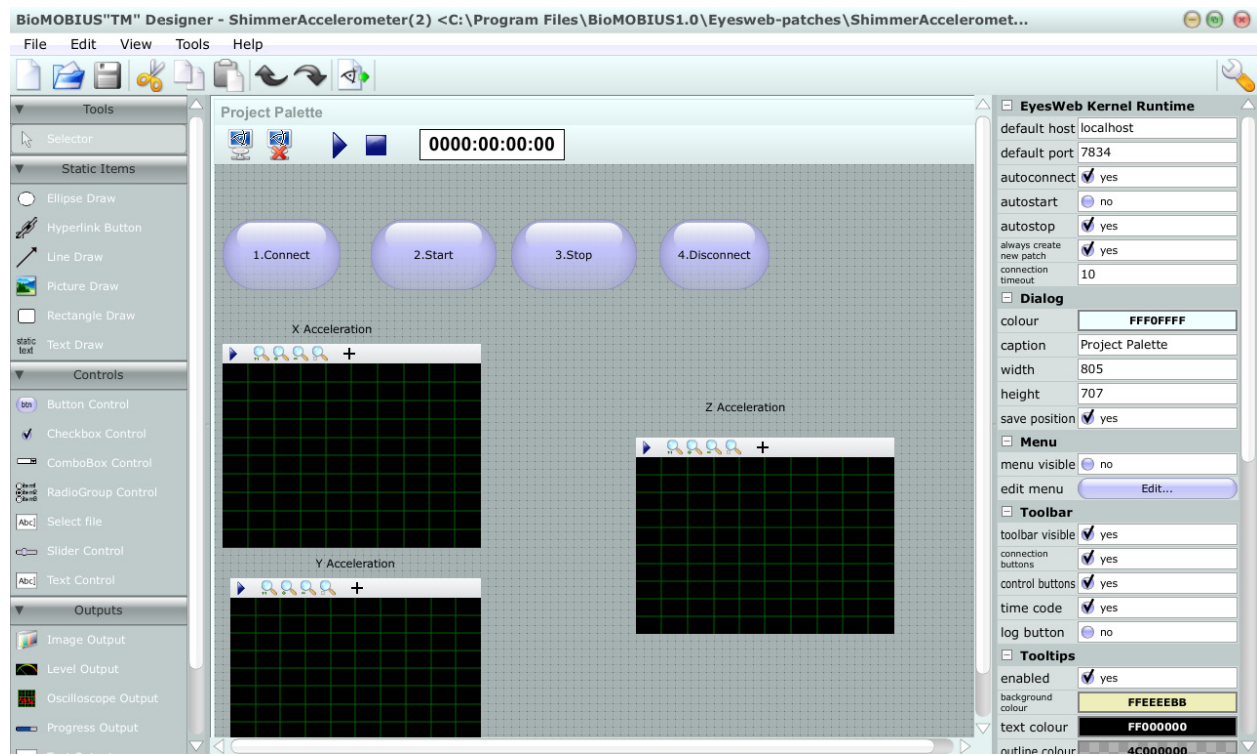


Figure 30. Screenshot of Biomobius GUI

References

- [1] K. Romer, and M. Junya, "Passive Distributed Assertions for Sensor Networks", *International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 337 – 348, April 2009
- [2] L. Yu, and S. Abdi, "Automatic SystemC TLM Generation for Custom Communication Platforms," *International Conference on Computer Design (ICCD)*, October 2007
- [3] S. Abdi, and D. Gajski, "A Formalism for Functionality Preserving SystemLevel Transformations," *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Vol. 1, pp. 139 - 144, January 2005
- [4] T. Grötter, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*, Springer, Chapter 8, pp 131-151, 2002
- [5] A. Sangiovanni-Vincentelli, "System-level design: a strategic investment for the future of the electronic industry," *International Symposium on VLSI Design, Automation and Test (VLSI-TSA)*, pp. 1-5, April 2005
- [6] S.M. Dawjee, *A Simple Appliance for the Management of Obstructive Sleep Apnea- the MEDUNSA Anti Snoring Device (MASD)*, Book Chapter in Sleep apnea syndrome research focus, edited by A.O. Lang, Nova Biomedical Books, pp. 37-39, 2007
- [7] S. Stuart, S. Davidman, and P.Flake, *SystemVerilog for Design*, Kluwer Academic Publishers, Chapter 10, pp 329-354, 2004
- [8] Shimmer Research, <http://shimmer-research.com>
- [9] G. Tolle, and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, pp. 121 – 132, Feb. 2005
- [10] M. H Neishaburi, and Z. Zilic, "Enabling Efficient Post-Silicon Debug by Clustering of Hardware-Assertions," *Proceedings of the Design Automation and Test in Europe (DATE10)*, March. 2010
- [11] J. Geuzebroek and B. Vermeulen, "Integration of Hardware Assertions in Systems-on-Chip," *Proceedings IEEE International Test Conference (ITC)*, pp. 1-10, Dec. 2008
- [12] B. Mihajlovic, Z. Zilic and K. Radecka, "Infrastructure for Testing Nodes of a Wireless Sensor Network," Book Chapter in *Handbook of Research on Developments and Trends in Wireless Sensor Networks: From Principles to Practice*, edited by H. Jin and W-B. Jiang, IGI Global 2010
- [13] M. W. Chiang, Z. Zilic, J-S. Chenard and K. Radecka, "Architectures of Increased Availability Wireless Sensor Network Nodes", *IEEE International Test Conference, ITC*, Oct. 2004

- [14] R. Zhang, Z. Zilic and K. Radecka, "Structuring Measurements for Modeling and the Deployment of Industrial Wireless Networks", *Proceedings of IEEE International Symposium on Industrial Electronics, ISIE '06*, Jul. 2006
- [15] X. Chen et.al, "Utilizing Formal Assertions for System Design of Network Processors," *Proc. Design, Automation and Test in Europe Conf. and Exhibition Designers' Forum (DATE 04)*, IEEE CS Press, 2004
- [16] E. Cotes, and N Manjikian, "A single-chip ring-based multiprocessor with region-level filtering of coherence traffic," *Workshop on Circuits and Systems, Newcas*, Aug. 2007
- [17] J. A. Stankovic et.al, "Wireless Sensor Networks for In-Home Healthcare: Potential and Challenges," in *High Confidence Medical Device Software and Systems (HCMDSS) Workshop*, Philadelphia, PA, Jun. 2-3, 2005
- [18] N. Collop, "Portable monitoring for the diagnosis of obstructive sleep apnea, *Curr Opin Pulm Med* , Volume 14, Issue 6, pp. 525–529 , Nov 2008
- [19] Y. Hu, A. Stoelting, Y. Wang; Y. Zou, M. Sarrafzadeh, "Providing a cushion for wireless healthcare application development," *Potentials, IEEE* , vol.29, no.1, pp.19-23, Jan.-Feb. 2010
- [20] T.A. Welch, "A Technique for High-Performance Data Compression," *Computer* , vol.17, no.6, pp. 8-19, June 1984
- [21] I.H. Witten, R.M. Neal, and J.G. Cleary, 1987, " Arithmetic coding for data compression, " *Communications of the ACM* , Vol. 30, No. 6, pp. 520-540, Jun. 1987
- [22] C.M. Sadler, and M. Martonosi, " Data compression algorithms for energy-constrained devices in delay tolerant networks," In *Proceedings of the 4th international Conference on Embedded Networked Sensor Systems*, pp. 265-278, Oct. 31 – Nov. 03, 2006
- [23] J.S. Vitter, "Design and analysis of dynamic Huffman coding," *26th Annual Symposium on Foundations of Computer Science*, pp. 293-302, 21-23 Oct. 1985
- [24] Mogul, J.C., Douglass, F., Feldmann, A., and Krishnamurthy, B., "Potential Benefits of Delta Encoding and Data Compression for HTTP," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 181-194, 1997
- [25] L. Galeottei, M. Paoletti, C. Marchesi, "Development of a low cost wearable prototype for long-term vital signs monitoring based on embedded integrated wireless module," *Computers in Cardiology, 2008* , vol., no., pp.905-908, 14-17 Sept. 2008
- [26] K. Takizawa, Huan-Bang, L. Kiyoshi, H.R. Kohno, "Wireless Vital Sign Monitoring using Ultra Wideband-Based Personal Area Networks," *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE* , vol., no., pp.1798-1801, 22-26 Aug. 2007
- [27] M. Colnarić, D. Verber, W.A. Halang, *Distributed Embedded Control Systems: Improving Dependability with Coherent Design*, Springer, 2008

- [28] S. Mullender ,”Distributed Systems,” Addison-Wesley Pub Co ,Chapter 1, pg 1-16, 1993
- [29] Microchip Technology Inc,” dsPIC30F Data Sheet “ , 2004
<http://ww1.microchip.com/downloads/en/devicedoc/70083g.pdf>
- [30] Microchip Technology Inc,” PIC18F67J50 Data Sheet “ , 2009
<http://ww1.microchip.com/downloads/en/DeviceDoc/39775c.pdf>
- [31] Wikipedia ,” Polysomnography – wikipedia, the free encyclopedia“, [accessed May 2009]
<http://en.wikipedia.org/wiki/Polysomnography>
- [32] Analog Devices,” AD7746 Data Sheet ” , 2005 http://www.analog.com/static/imported-files/data_sheets/AD7745_7746.pdf
- [33] Wikipedia ,” Vital Signs– wikipedia, the free encyclopedia“, [accessed April 2010]
http://en.wikipedia.org/wiki/Vital_signs
- [34] Steve Blozis ,” Opto Electrical isolation of the I2c bus”, 2004
<http://www.embedded.com/shared/printableArticle.jhtml?articleID=49901764>
- [35] Terrance J. Dishongh, Michael McGrath ,”Wireless Sensor Networks for healthcare applications “ , Chapter 2 , pg 13-39 , Artech House, 2010
- [36] H. Foster, A. Krolnik, and D. Lacey, *Assertion-based design*, Kluwer Academic Pub, 2nd edition, 2004
- [37] Nordic Semiconductor,” nRF24E1 – Product Specification“, 2006
http://www.nordicsemi.com/files/Product/data_sheet/Product_Specification_nRF24E1_1_3.pdf
- [38] Embedded Platform Lab, National Tsing Hua University , “Eco development tutorial using SDCC “ , 2009 <http://www.ecomote.net/>
http://epl.cs.nthu.edu.tw/EcoKit/download/ECO_tutorial_v1.0.pdf
- [39] Embedded Platform Lab, National Tsing Hua University, “Eco API document“, 2009
<http://epl.cs.nthu.edu.tw/EcoKit/?q=node/12>
- [40] M.L. Hetland , *Beginning Python: From Novice to Professional*, Apress, 2005
- [41] Universal Serial Port Python Library <http://ibarona.googlepages.com/uspp>
- [42] M. Boule, and Z. Zilic, *Generating Hardware Assertion Checkers*, Chapter 2, pg13-33, Springer 2008
- [43] F. Gebali, H. Elmiligi, and M.W. El-Kharashi ,*Networks-on-Chips theory and Practice*, CRC press, Chapter 1 , pg 1-28 , 2009

- [44] L. Benini, and G.D. Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer magazine* , vol.35, no.1,pp.70-78,Jan 2002
- [45] B. Cordan, "An efficient bus architecture for system-on-chip design," , *Proceedings of the IEEE 1999 Custom Integrated Circuits* ,San Diego, pp. 623–626, 16-19 May 1999
- [46] S. Bourduas , "Modeling, evaluation, and implementation of ring-based interconnects for network-on-chip," PhD. Thesis, McGill University, 2008
- [47] B. Vermeulen, N. Stollon, R. Kuhnis, G. Swoboda, and J. Rearick, " Overview of debug standardization activities," *IEEE Design & Test of Computers*, pp.258-267, May-June 2008
- [48] J. Geuzebroek and B. Vermeulen," Integration of Hardware Assertions in Systems-on-Chip", *Proceedings of the IEEE International Test Conference*, pp. 412-421, Oct. 2008
- [49] M. Boule, J.S. Chenard, and Z. Zilic, "Debug enhancements in assertion-checker generation," *Computers & Digital Techniques, IET* , vol.1, no.6, pp.669-677, Nov. 2007
- [50] G.D. Micheli and L. Benini, *Networks on Chips: Technology and Tools*, Morgan Kaufmann , 2006.
- [51] K. Goossens, B. Vermeulen, and A.B. Nejad," A high-level debug environment for communication-centric debug", in *Design, Automation and Test in Europe*, pp. 202-207, April 2009
- [52] M. Boule, J.S. Chenard, and Z. Zilic, "Adding Debug Enhancements to Assertion Checkers for Hardware Emulation and Silicon Debug," *International Conference on Computer Design, ICCD 2006*, pp.294-299, 1-4 Oct. 2007
- [53] S. Bourduas, J.S. Chenard, and Z. Zilic, Z, "A Quality-Driven Design Approach for NoCs," *IEEE Design & Test of Computers*, vol.25, no.5, pp.416-428, Sept.-Oct. 2008
- [54] M. Boule, J.S. Chenard, and Z. Zilic, "Assertion Checkers in Verification, Silicon Debug and In-Field Diagnosis,". *8th International Symposium on Quality Electronic Design, ISQED '07* , pp.613-620, 26-28 March 2007
- [55] Image obtained from http://www.tucos.com/ai/Style0531_3/html/image59.htm
- [56] V. Kallankara, M.H. Neishaburi, K. Radecka and Z.Zilic , "Using Assertions for Wireless System Monitoring and Debugging," " *8th IEEE International NEWCAS conference*, 20-23 June 2010, in press