2-18-2005

System Level Energy Optimization for Location Aware Computing

Hariharan Sankaran
University of South Florida

System Level Energy Optimization for Location Aware Computing

by

Hariharan Sankaran

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Date of Approval:
February 18, 2005

Keywords: GPS, Power dissipation, Codec, Dynamic power management, System modeling

DEDICATION

To Krishna Chaitanya, Nityananda, Sri Advaita, Gadadhara, and Srivasa.

ACKNOWLEDGEMENTS

I would like to thank Dr. Srinivas Katkoori for his continuous support and encouragement throughout the course of the research. I would like to thank my committee members Dr. Nagarajan Ranganathan and Dr. Soonate Kim. I also convey my thanks to Daniel Prieto and his technical support staff for extending their helping hands whenever needed. I thank my parents and my sister for helping me in my quest to pursue Master's degree. I thank all my friends in the VCAPP group and outside, who have encouraged and provided help in accomplishing this task.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYSTEM LEVEL ENERGY OPTIMIZATION FOR LOCATION AWARE COMPUTING

Hariharan Sankaran

ABSTRACT

We present an energy conscious *location-aware* computing system that provides relevant information about the user's current location. The location-aware computing system is initialized with a map (in the form of a graph) as well as audio files associated with several locations in the map. The system consists of: GPS receiver module, Serial port, Compact flash module, Stereo codec, Power manager module implementing three sub modules namely, GPS-to-real-world position conversion module (implements algorithm to convert GPS co-ordinates to graph nodes), Nearest-location-search module (implements modified Dijkstra's algorithm), and User speed estimation module. The *location-aware* computing system receives the GPS co-ordinates for the current location from GPS receiver through the serial port. The system converts the GPS co-ordinates to map co-ordinates stored in the Compact Flash card. If the current location matches the landmarks of interest in the site, then the relevant audio details of the current location is played out to the user. The power manager sets the GPS co-ordinates update frequency to avoid keeping the system component ”on” throughout the entire course of travel.

The power manager implements an algorithm that works as follows: at any given location, the algorithm predicts the user speed by exponential average approach. The attenuation factor of this approach can be varied to account for the user speed history. The estimated speed is used to predict the time (say T) required to reach the next nearest location determined by Nearest-location-search module implementing modified Dijkstra's algorithm. The subsystems are shut-down or switched to low-power mode for time T. After time T, the system will wake up and re-execute the algorithm.

CHAPTER 1

INTRODUCTION

Ubiquitous computing technology is a paradigm shift from personal desktop computing era to a technology that is embedded in the user environment. Mark Weiser [1], considered as father of ubiquitous computing defines ubiquitous computing as a technology opposite to virtual reality. Virtual reality technique creates an environment around the user and forces the user to live in that environment. On the other hand ubiquitous computing forces the computer to live in the user's natural habitat. Ubiquitous computing is a natural progression of mobile computing research due to the advancements in the field of mobile communications and computing power. The communication allows the system to share information among different portable devices like sharing the status, sensing user location, environment etc., to reveal the context in which they are operating.

Context-awareness is the force that drives the research of ubiquitous computing. Context can be broadly classified into four categories [2][3]:

1. Computing context, based on communication cost, resource availability, and network availability.

2. User context, based on user location, user profile, and people nearby.

3. Physical context, based on temperature, noise level, and lighting intensity.

4. Time context, based on time of the day, week, month or year.

Several authors in literature have proposed different definitions for context. Dey [4] defined context as information that characterizes the situation of the entity (person, location, or object). Schmidt [5] defined context as the user and device states, like surroundings, situation etc., Ubiquitous computing applications adapts itself to different contexts providing tailored functionality based on the context. For example, a context-aware application built in a cell-phone can turn itself to vibrate mode on reaching a movie theatre without any user intervention, or remind the user about the list of books to be picked up when he/she is in a library. An important distinction between a mobile device such as PDA, laptop etc., and a ubiquitous computing device is, a mobile device require an user to operate them in order to extract service from them. In these traditional mobile devices, the devices are the center of focus instead of tasks that needs to be accomplished. On the other hand, ubiquitous computing does not require human intervention and the application changes it's charactersitics based on the current context. The ubiquitous computing devices blends itself with the user environment similar to the eye glasses, which acts as an eye to a person.

1.1 Location-aware Computing

Advancements in the field of mobile computing, location sensing, and wireless networking has created a new class of computing called location-aware computing. Location-aware computing devices belongs to the class of context-aware computing based on the "primary" context, the user location. The devices of this class may influence the behavior of the application as the location changes or just display the change in context to the user. Chen et al [2] calls the former as active context-aware devices and latter as passive context-aware devices.

Location-aware devices requires location sensing technique to determine and adapt itself to changing locations. Location sensing techniques may give absolute co-ordinates of locations such as GPS (Global Positioning System), GLONASS (Global Navigation and Surveillance System), and GSM (Global System for Mobile communications) or relative co-ordinates such as cell based wireless communication broadcasting the location information. Location sensing element can be broadly classified into two categories: indoor position tracking elements and outdoor position tracking elements. Indoor position tracking elements may include technologies such as active badge, infra red communication, RFID (radio frequency identification) technology, etc., While outdoor position tracking elements include GPS, GLONASS, and cell based wireless communication.

1.2 Challenges in Design of Location-aware Computing System

The fundamental issues in the design and implementation of *location-aware* computing system are [6]:

1. Hardware: mobile elements are resource poor compared to stationary dektops due to size, power, and weight.

2. Security: mobile communications are fraught with dangers of security violations due to communication through open (public) channels.

3. Network: wireless connectivity is highly variable in reliability and performance.

4. Energy Constraints: mobile elements have to operate under limited energy budget compared to stationary desktop applications.

Research on reducing hardware constraints focuses on making use of stationary hardware resources that are available nearby to the user location. For example, a location-aware device could outsource computation intensive jobs to the nearest server through wireless communication. This allows the location-aware devices to be simple in design, rich in resources, less in weight, and low power consuming. But the location-aware devices becomes dependent on stationary devices limiting the range of its use. Another hardware resource that places constraints on location-aware computing system is location sensing elements. The use of different location sensing elements has implications on factors such as [7]: accuracy, sensing modalities, energy, orientation, and dynamic environments.

Research on security focusses on developing protocols and mechanisms to audit the release of location information to guard against the privacy lawsuits. In a location-aware computing system, there is an inherent friction between privacy and location information requiring system design techniques to control the exposure of location information, efficient access control protocol to monitor exposure of location information to right party, and user interface techniques to inform user about the locations being monitored and precautionary measures to enhance trustworthiness of monitoring process [6]. Wireless communications is the field which has seen explosive growth in recent years, due to the wide spread use of mobile computing devices and enormous commercial oppurtunity associated with ubiquitous computing. The main factors inherent in networks (wired or wireless) that affects the advancement are: frequency, bandwidth, range, and power.

The energy constraint determines the success and longetivity of mobile computing system. As it can be seen from the challenges described above energy is the common thread that gets stretched to accomodate the need to meet desired results. Addition of hardware resources to a mobile computing device increases the energy demand of the device: increasing the range of communication capabilities requires more transmission power, tracking of user location requires frequent updates from location sensing elements increasing the bandwidth consumption and location-update processing cost are some examples illustrating the need for policies to tackle the energy budget problem, which co-exist in mobile computing devices.

In this thesis, we have developed an initial prototype of *location-aware* computing system, NAVIFIND. NAVIFIND aids the tourist to find the landmarks of interest of a given site via audio output. The NAVIFIND uses GPS as the location tracking element and can be used anywhere around the globe. In NAVIFIND, we address the problem of energy consumption in location-aware computing system. We have proposed a system level energy optimization technique for reducing the energy consumption. As described earlier, a location-aware computing system provides tailored functionality only on reaching certain predetermined or dynamic environments. Our policy predicts the time (say T) the user might take to reach these environments. The system components are switched OFF or to a low-power mode for time T, where the execution of the application is not needed hence reducing the energy consumption.

1.3 Power Optimization: System Level vs Traditional Approach

Before going into the details of device-level (traditional approach) and system-level power optimization policies it is neccessary to analyze the sources of power dissipation in CMOS circuits. The sources of power dissipation in CMOS circuits are:

1. Static Power Dissipation

2. Dyanamic Power Dissipation

Static Power Dissipation is due to leakage current between power supply and ground. The main sources of leakage currents are (i) Reverse-biased p-n junction current; (ii) Subthreshold leakage currents; (iii) Punch-through; (iv) gate-induced drain leakage; (v) Gate-tunneling. Aggressive scaling of technology and reduction of threshold voltage are the causes for increase in leakage current and static power dissipation.

Dynamic power is consumed whenever the circuit switches and an average dynamic power is given by

$$P_{dynamic} = C_L \cdot N_{SW} \cdot V_{DD}^2 \cdot f \tag{1.1}$$

where C_L is the load capacitance, N_{SW} is the switching activity, V_{DD} is the supply voltage, and f is the switching frequency. The researchers have attacked each of the individual term in dynamic power equation to reduce the dynamic power dissipation. Techniques proposed in the literature includes reducing capacitive load C_L by transistor sizing and optimal placement and routing techniques. The switching activity is reduced by techniques such as invert bus encoding [8] and employing gray code [9]. Scaling down the supply voltage leads to significant power savings. The equation 1.1 shows quadratic dependance of power on V_{DD}. As process technology shrinks more transistors are packed in the same die area, complex design styles are employed, and higher clock frequency resulting in significant power consumption.

1.3.1 Traditional Approach to Power Optimization

The traditional approach concentrates on power analysis and optimization at RTL or gate level. The low power design flow for traditional approach is shown in Figure 1.1. Until the recent past VLSI design community has spent most of the time evolving new power optimization methodology at logic and RTL level which includes works such as Guarded evaluation [10], Precomputation [11], and retiming [12].

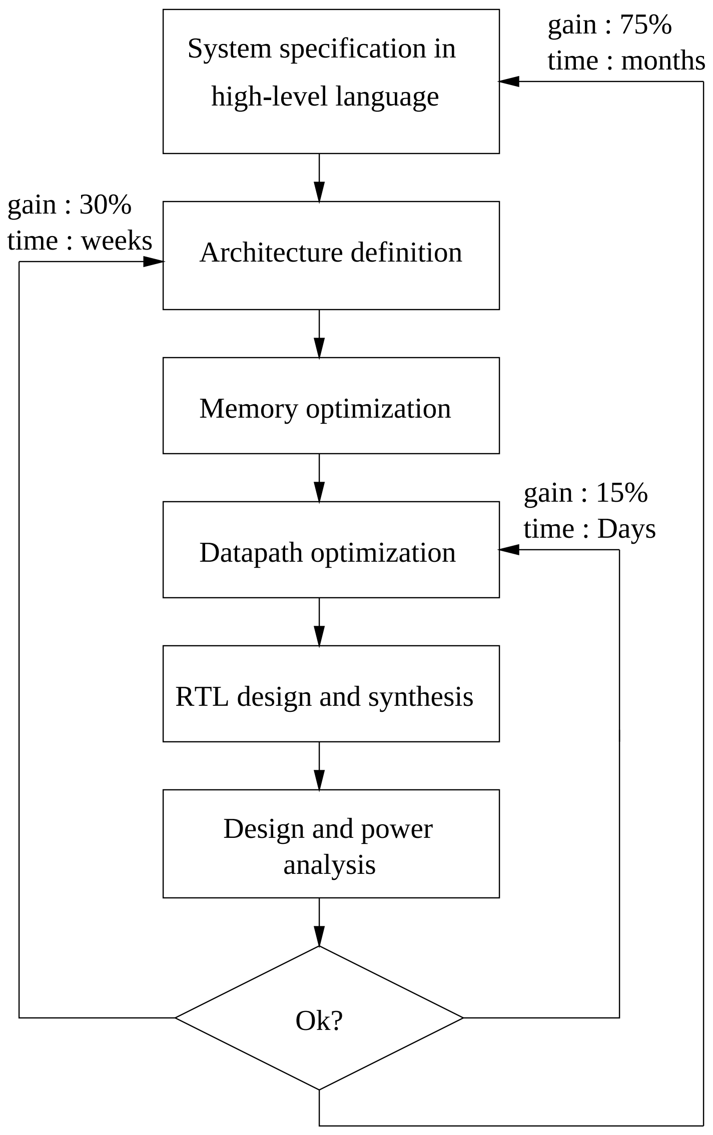

Figure 1.1 Device-level Low Power Design Flow [13]

The power optimization techniques at the RTL or logic level optimizes a particular module. But system is a collection of various heterogenous units working together to achieve the desired functionality. A power aware design decision at one functional unit may affect the power consumption in another unit which is not considered at the device level. Design decisions such as algorithm selection and mapping of the selected algorithm to an architecture are done at higher level of abstraction. Performing power analysis and implementing power optimization features at lower levels of abstraction might not always yield an optimal power gain. And it can be easily seen from the flow diagram for traditional approach in Figure 1.1 that design time is considerably increased to months in some cases if the power constraint is not met. If architecture or algorithm is modified then the entire sequence of steps from architecture definition in Figure 1.1 have to be executed increasing the design time to months [13].

1.3.2 System-Level Approach to Power Optimization

The System-level approach requires mapping of system specification usually written in a high level language such as C or VHDL onto an architecture. Figure 1.2 shows the system-level design flow [13]. At this level various algorithms and architectures satisfying the functionality are analysed before the actual netlist synthesis therby saving considerable design time as well as achieving desired power and performance metrics. There are actually three stages in a system design flow:

1. System Modeling

2. Design & implementation

3. System Management

System Modeling is the process of developing an abstract view of the system specified as an executable or non-executable model depending on the system complexity. In this stage where key algorithmic decisions are made before mapping the algorithm onto the architecture. The algorithmic decision may include deciding on the bandwidth requirements for different units in the system and level of accuracy to be supported to achieve desired functionality. It also gives a hardware/software partition and macro-architectural-template [14].

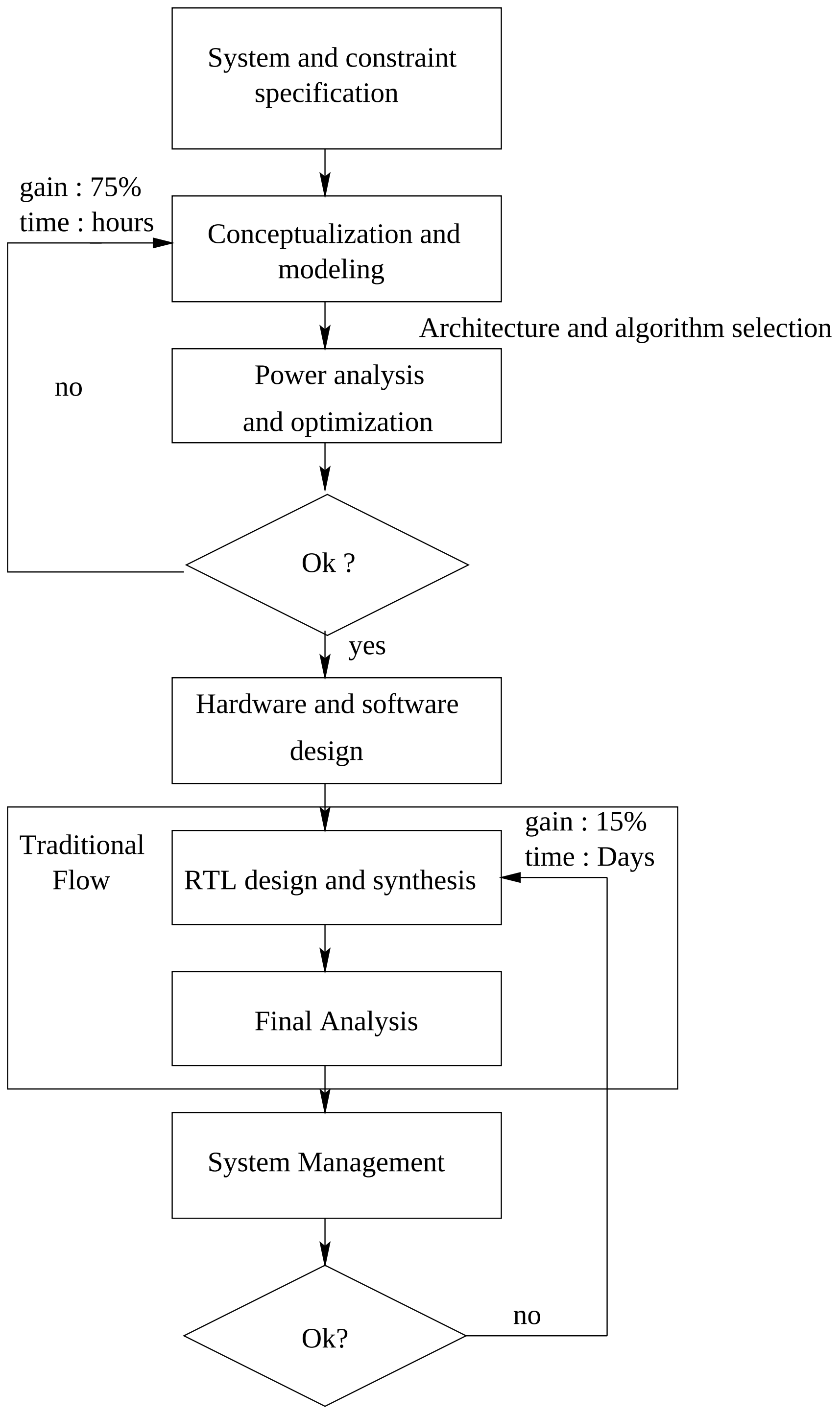
System and constraint specification
gain : 75%
time : hours
Conceptualization and modeling
Architecture and algorithm selection
no
Power analysis and optimization
Ok ?
yes
Hardware and software design
Traditional Flow
gain : 15%
time : Days
RTL design and synthesis
Final Analysis
System Management
no
Ok?

Figure 1.2 System-level Low Power Design Flow [13]

System Design phase deals with actual implementation and refining of hardware architecture from system model into computational, memory, and communication units. The decisions such as implementing hardware architectural template as an ASIC or as programmable cores are taken in this phase and various power optimization technologies such as dynamic frequency scaling, multiple voltage supply on a single chip are analyzed in this stage. *System Management* phase deals with design of energy efficient system level software such as operating system to provide efficient runtime support system.

1.4 Proposed Energy Conscious Location-aware Computing System

In this thesis, we propose a portable location-aware computing system, NAVIFIND that provides relevant information about the user current location. NAVIFIND aids the tourists to learn the landmarks of interest of a given site by providing audio output to the user. The system is initialized with a map of the current site (in the form of a graph) and audio data pertaining to landmarks of interest in the site. NAVIFIND system consists of the following sub-systems: GPS receiver module, serial port, compact flash module, stereo codec, and power manager with three sub-modules namely, GPS-to-real-world position conversion module (implements algorithm to convert GPS co-ordinates to graph nodes), nearest-location-search module (implements modified Dijkstra's algorithm), and user speed estimation module.

The power manager implements an algorithm that works as follows: at any given location, the GPS co-ordinates of the location is converted to graph nodes, based on which the algorithm predicts the user speed using exponential average approach. The attenuation factor of this approach can be varied to account for the user speed history. The estimated speed is used to predict the time (say T) reqired to reach the next nearest location The subsystems are shutdown or switched to low-power mode for time T. After time T, the system will wake up and re-execute the algorithm.

CHAPTER 2

RELATED WORK

The development in the field of mobile computing, wireless communication, location sensing, and augmented display has spawned explosive growth in the field of ubiquitous/pervasive computing. The growth of these mobile devices has emphasised the need for reducing energy consumption at all levels of system design. A number of location-aware computing systems have been proposed in the literature. Section 2.1 briefly discusses some interesting location-aware computing systems. We also present a section on system level power optimization techniques proposed in the literature. Some of the system level power optimization techniques proposed are simple and can be implemented as a hardware component.

2.1 Location-aware Mobile Computing Machines

This section describes about different location-aware computing systems developed in various research laboratories around the world. The section also provides insight on various location sensing techniques used to track user position and different contexts based on which mobile units model their applications.

2.1.1 Cyberguide: A Mobile Context Aware Tour Guide

Cyberguide [15] is a mobile tourist guide, providing directions and information about nearest places of interest to the user based on the user location. The prototypes of cyberguide has four major components: map component, information component, positioning component, and communication component.

- Map component, contains maps of the location as bitmaps or vector-based maps. This component is used to track the user location relative to user physical surroudings.
- Information component, contains information about the sites of interest in the maps as Newton books (Newton platform documentation viewer).
- Positioning component, tracks the postion and orientation of user using GPS as the location sensing component for outdoors and IR technology for indoor use.
- Communication component, implements both wired and wireless communication to communicate between the mobile units; between mobile units and network; and to broadcast the location update information.

Indoor Cyberguide, one of the member of the Cyberguide family provides directions to user to various demonstrations exhibited; displaying details about the exhibits, and displays the map of the surrounding area where user resides. The postioning component used to obtain user position is a set of TV remote control units acting as active beacons and a special IR receiver tuned to the carrier frequency of the active beacons. Outdoor cyberguide guides the user based on the latitude and longitude information obtained from the GPS unit.

2.1.2 Context-aware Electronic City Tour Guide

City tour guide [16] is designed to eliminate the tourists need to stick to the fixed start and fixed durations of a group-based tours and allows the user to interact with the electronic guide to prepare a suitable schedule of various places to visit according to his/her personal interests. The guide uses a Fujitsu teampad with pentium 166 MMX processor; and user interface is through a modified browser template. The user interface allows the user to enter or retrieve following information:

- User personal preferences like historical or architectral and prefered language to display information.
- Navigation of the city using a map.
- Creation and modification of a tour.

- Communication with other mobile units or tourist information center by sending text messages.
- Booking accomodations remotely.

The city guide is also capable of providing dynamic information through cell-based wireless infrastructure. The position information is obtained through base stations estabilished for cell-based wireless communication. This approach of finding position using strategically located base stations has the advantage of having no extra hardware requirement. But, may result in low resolution of positional information. The city guide caches large parts of information model locally to tackle situations where city guide does not have network connectivity due to cell-based wireless communication infrastructure. Other network related activities like location information, booking accomodation, and communication through text messaging are affected due to the loss of network connectivity. The city guide has a battery lifetime of approximately two hours.

2.1.3 Metronaut: A Wearable Computer

Metronaut [17] is a wearable computer supporting applications like navigation, messaging, and scheduling. The positional information of a visitor is obtained using a bar code reader. The user can use the bar code reader to scan the bar code stickers (stickernet) containing the location information around the campus. On finding the location information the metronaut provides directions on a LCD screen to the user. The four primary components of metronaut are: Metronaut, ground-based network system, SkyTel paging network, and SkyTel network interface.

Metronaut allows scheduling and messaging through a two-way pager. The communication proceeds from the mobile unit (Metronaut) via the pager to the SkyTel paging network, to the SkyTel network interface. The SkyTel network interface will then the forward the information to the server. The server processes the information on behalf of Metronaut and replies back to the mobile unit. Metronaut consumes less than one watt of power due to the absence of power hungry location sensing sytems like GPS (about 400mW). But, the downside of having bar codes and bar code readers as the location sensing technique is, the scalability of Metronaut is limited compared to mobile units with GPS or GLONASS as position tracking element.

2.1.4 Active Badge Location System

Active Badge Location System [18] is designed primarily to help telephone receptionist to locate a person working in a large organisation precisely and to forward a call to his nearest telephone destination. The system consists of an active badge that emits a unique code, which is nothing but a pulse-width modulated infrared signals for approximately a tenth of a second every 15 seconds and this signal is picked up by a network of sensors placed throughout the building relaying the data to the central server. The server is designed as a four layer system consisting of:

- Network Control, responsible for polling all the sensors in the netwok.
- Representation, responsible for extracting valid data from network, time-stamp the data, and store the data containing badge ID, location, and time-stamp in the data structure.
- Data Processing, to process the data collected across active badge network and to compress the data to avoid network congestion problem.
- Display interface, displays user location information, change of locations and other relevant details pertaining to the active badge as text or graphical display.

2.1.5 Wearable Remembrance Agent

Wearable Remembrance Agent (RA) [19] is designed to provide the user with relevant information gathered in the past depending on the user location. For example if a person enters a specific class the notes he has taken in the past attending the same class at the same time of the day will be displayed using heads-up display. The wearable remembrance agent uses five context cues to provide relevent information:

- Wearer's physical location, provided by GPS when used outdoors or an indoor location tracking element like active badge or IR technology or location entered explicitly by the user.

- People who are currently around, provided by another person's wearable computer or active badge system or entered by the wearer.
- Subject field, provided by the user as an extra tag or information extracted from the header fields like subject line in email.
- Date and time-stamp, provided by the system clock.
- The original notes, the body of the note converted to word-vector for later keyword analysis.

The RA is implemented using lisp and C running under linux on a wearable 100MHz 486 based processor with a keyboard and monochrome heads-up display.

2.1.6 Location-aware Information Delivery with comMotion

comMotion [20] is a location learning agent capable of associating future events to the learned location. The Position component used is a GPS receiver. comMotiom also provides mobile access to location-based information from the Web. The architecture for comMotion includes a portable PC, a GPS receiver, and a CDPD modem. comMotion supports both speech and graphical user interfaces. The speech interface includes speech recognition and text-to-speech synthesis supporting speaker-independence and continous speech recognition.

comMotion also acts as a remembrance agent by alerting the user on reaching a particular destination. For example comMotion alerts the user to buy milk when user passes through the grocery store. As said earlier comMotion is a location learning agent it accomplishes this by querying the user on reaching a new location, the user may ignore or enter details about the new location. The comMotion has a map module to display the current user location together with neighbourhood locales, such as banks, grocery stores, or schools. comMotion supports communication with other user mobile devices and allows the user to subscribe to information services like headline news, weather reports, and movie listings from information sources on the Web.

2.1.7 Drishti: Navigation System for Visually Impaired and Disabled

Drishti is a wireless pedestrian navigation system designed to improve the pedestrian experience of a visually impaired person. The applications provided by Drishti includes optimized route planning based on user preference, temporal constriants such as traffic congestion, and dynamic obstacles like ongoing road work and road blockade for special events, providing environmental conditions and landmark information queried from a spatial database along the user route through voice cues.

The major components (hardware/software) used in Drishti includes wearable computer, voice recognition and synthesis, wireless networks, Geographic Information System (GIS), and GPS. The hardware component includes a Xybernaut MA IV wearable computer with pentium 200MHz processor, full duplex sound card, VGA Head mounted display, DGPS receivers, and 802.11b wireless LAN technology. The sofware portion of Drishti includes a spatial database engine, C API for route store, Mapserver to serve GIS datasets over the internet, and voice recognition and synthesis software. Drishti guides the visually impaired user based on static and dynamic data. The dynamic data is provided through wireless communication between the mobile unit (Drishti) and a central server. Drishti also allows visually impaired user to enter comments about difficulties faced along the route planned by Drishti, which could be used as a pointer for future optimized route plan.

2.2 System Level Power Management Policies

Location-aware computing system are portable units working under limited energy source provided by the battery. The principle behind any system level power management policy is to switch the power manageable components to low power modes under performance constraints as soon as they become idle. The key to this decision making process is to know when to switch the system to low power modes and to which mode in case of multiple power modes available in a power manageable system component. To make things worse in most real-time systems the workload is non-uniform, depending on the user behavior and is event driven. It could be argued that today's electronic components have more efficient low power features at device level, but the fact is, these components are going to interact with one another as a part of a system. A power policy at component level may force the component to a low power state when the required constraints are met, without knowing it may be reactivated soon without any energy gain and thus incurring penalty for misprediction. A system level power policy implemented in hardware or software will have complete working knowledge of the system and has finer control over the operations of the system thereby providing significant savings and less performance/power penalty.

Based on the ability to predict the idle time the system level power management policies are classified into three broad categories:

1. Timeout (Static and Dynamic)
2. Predictive (Static and Dynamic)
3. Stochastic

A key criterion to be satisfied by all power management policies is, a system component should be switched to low-power state only if the saved power exceeds the overhead involved to switch states. The overhead here implies number of housekeeping procedures like backing-data and storing system status. An excellant example to illustrate this fact is shown in the Figure 2.1.

Figure 2.1 shows the effect of applying a simple shutdown policy to two different scenarios. The assumption here is a system component has only two states to switch to, running or sleep state. Let I be the system idle time, E delay overhead of entering the sleep state from running state, S sleeping time, and W delay overhead for resuming from sleep state to running state. P_R and P_S are the power consumption values in running and sleep states. P_{EW} is the average power-dissipation overhead of entering the sleep state from running state and viceversa. EG represents energy gain and $P_R \geq P_{EW} \geq P_S$.

In Figure 2.1(a) the idle time $I \geq E$ so energy gain is $P_R \cdot I - (E + W) \cdot P_{EW} - P_S \cdot S$ with a delay overhead of W. Consider Figure 2.1(b) $I \leq E$ the energy gain is $P_R \cdot I - (E + W) \cdot P_{EW}$ which is negative and a large delay overhead $delay = W + (E - I)$ justifying the fact that power management policies has to determine a threshold value above which the power manager can switch the system component to sleep state. And this value is device dependent.

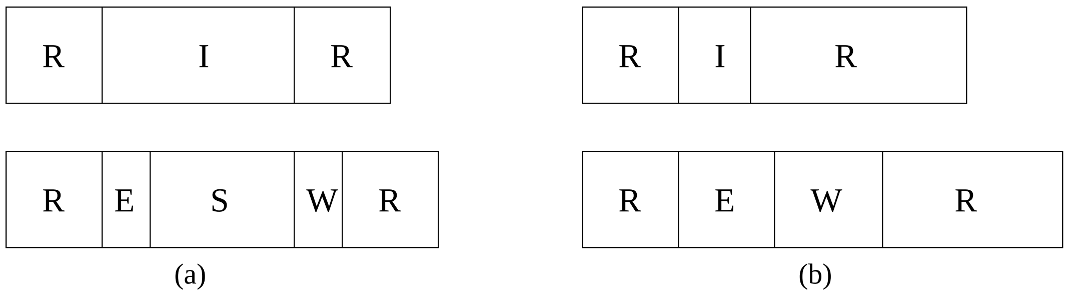

Figure 2.1 Two Contrasting Scenarios in Shutdown Prediction (a) $I \geq E$ (b) $I \leq E$

2.2.1 Time-out

A time-out policy sets a time-out value τ after which the device is shutdown or switched to low power state. The time-out policy may be further classified as static or dynamic, depending on their ability to adapt at run time.

2.2.1.1 Static Time-out Policy

The static time out policy is the most simple policy based on fixed time-out interval. The system component or device waits for the pre-defined time-out value τ to expire. If the component idle time is less than the time-out value τ, the sytem component is kept active and viceversa. In windows operating sytem user can set the timeout value for display or hard disk through advanced configuration and power interface (ACPI). On expiration the components are switched to low power states. The success of this policy depends on the proper choice of time-out value and it may be device dependent. Short time-out value is good but repeated shutting down and reviving of system components incurs performance as well as energy costs. A large time-out interval means the system is active for a longer time during idle periods therby wasting power. A best time-out policy is one where the time-out interval is zero for long idle time periods and a long time-out value for short idle periods [21]. But realistically a best time-out interval is not possible due to the need for future information which cannot be predicted accurately or available before hand. So best time-out is only useful for analyzing the performance and power costs incurred in other algorithms.

A subtle variation of fixed time-out policy is Device dependent time-out policy. Device dependent time-out (DDT) policy considers the hardware parameters of the devices to set the time-out values. The time-out values are selected based on minimum required threshold value the device under consideration needs to remain idle to achieve power savings.

An obvious drawback in static time-out policy is it doesnot consider workload statistics which are non-uniform and are mostly event driven. Consider a situation when there is a close call, such that once the time-out value expires the component goes to sleep state but almost immediately an event triggers the component to wake up from sleep state then there is a power penalty involved in waking up the component and as well as no significant power is saved in sleep state, there is no way the fixed time-out policy is going to adjust to this situation the more frequent close calls occur the more power penalty this policy is going to suffer.

2.2.1.2 Adaptive Time-out Policy

To eradicate the above mentioned close call scenario the time-out policies needs to be adaptive. In ATO [22] the time-out value is adjusted dynamically. Various situations demanding updation of time-out values are

1. When system revival time is too long and unacceptable while time-out is too short and must be increased.

2. When system revival time is acceptable and time-out is long enough and can be shortened without increasing repeated system revivals.

3. In close call situations, Where the time-out value may be just lower than the idle time where the system is shutdown and revived immediately prompting increase of time-out value.

Another important criterion that has to be decided in ATO is the rate of adjustment of time-out value. The human psychology gives the clue to decide about the rate of adjustment, humans tend to have short term memories a delay is acceptable every couple of hours if the system goes idle for sufficiently long time rather than delay encountered every five minutes. Considering the fact the time-out may be decreased by a small factor if multiple acceptable system revival delays occurs. In case of unacceptable delays the time-out might be increased by a considerable amount to exploit the human tendency.

The rate of adjustment is given by the ratio τ and the previous idle period. If the ratio is too small the policy increases τ. If small decreases τ. One good example of this policy is in hard-disk where spin-up after spin-down takes a considerable time and user may not be quite satisfied with this delay. In such cases a trade-off between performance and power savings is required. For the above described scenario the time-out value τ is adjusted by adding two different values α and β to τ when undesirable or accepatable spin-ups occur. Normally the α and β values are set as $\alpha > 0$, $\beta < 0$ and $\alpha > \beta$ or another method can be to multiply τ by α and β in this case the values for α and β are $\alpha > 1$ and $1 > \beta \geq 1/\alpha$. In other words, when a unacceptable spin-up occurs the spin-down time-out should be increased by enough to avoid unacceptable spin-up called as bumps in future. When an acceptable spin-up occurs, the spin-down time-out can be decreased, but more gradually. The rate of adjustment of τ is done only within certain range to avoid abnormal behaviour like increasing time-out indefinitely or decreasing the time-out to lower values thereby incresing bumps.

An example illustrating the effects of the above described spin-down policy on a windows trace is shown in Figure 2.2. The Figure 2.2 also depicts the energy consumption and bumps encountered in fixed time-out versus adaptive time-out policy. As it can be seen small fixed time-out results in least energy consumption but increase in bumps and increase in fixed time-out or using adaptive policy results in decrease in number of bumps but increase in energy consumption. The Figure 2.2 also shows the minimum and maximum value within which the adaptive policy updates the time-out value τ.

2.2.2 Predictive Policy

The main drawback in the time-out policy is the need to wait for the time-out to expire resulting in waste of energy during the time-out period and there is always a performance penalty on wakeup. In predictive policy this shortcoming is rectified by predicting the idle period before hand and the components are switched to low power states depending on the predicted value and the second problem is addressed by a pre-wakeup procedure.The Predictive policy may be static or dynamic.

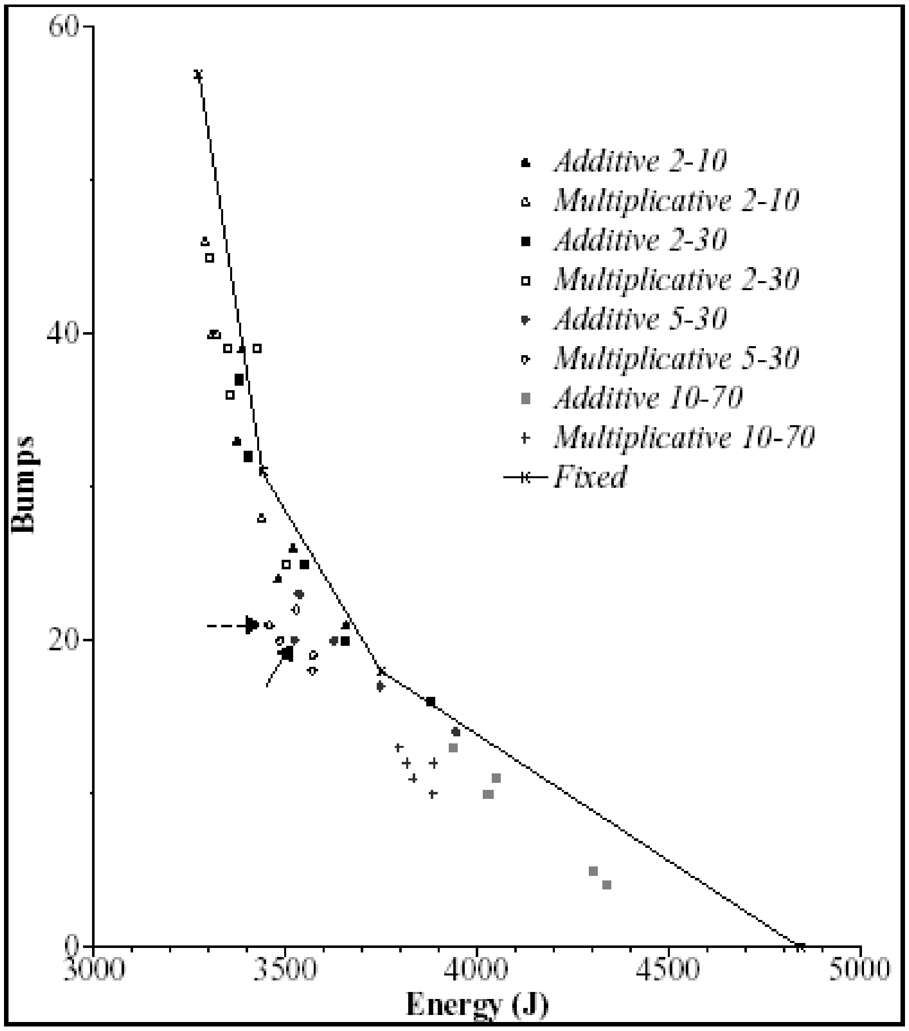

Figure 2.2 Simulation Results Comparing Energy Consumption and Bumps, for Adaptive Time-out and Fixed Time-out [22]

2.2.2.1 Prediction by Regression Analysis and L-shaped Policy

Srivastava et al. [23] have proposed two predictive shutdown schemes based on offline analyses of sample traces, first one is based on regression analysis and the next one is based on on-off activity of the sample trace. The idea behind their approach is a simple heuristic rule that uses computation history to make the prediction. The workload trace they used is the on-off activity of X-server running on a dedicated processor.

In the first method Srivastava et al. obtained a nonlinear regression equation from the past history.

$$T_{pred} = \phi(T^{n}_{active}, T^{n-1}_{idle}, \cdots, T^{n-k}_{active}, T^{n-k-1}_{idle}) \tag{2.1}$$

The predicted value T_{pred} depends on the past sequence of idle and active periods. If the predicted value $T_{pred} \geq T_{cost}$, decision to shutdown or switch to low power state is made. T_{cost} is the break-even time, the time that makes the energy consumption equal in a device that is kept in working state and the device that is shutdown and revived [24]. To acheive power savings the break-even time should include both the transition delays and the minimum time the device has to spend in sleep state.

The disadvantage of this approach is there is no general way to decide the type of regression equation and extensive data collection and analysis needs to be carried out to fit the regression model [25].

The second approach is based on intuition observed from the L-shaped plot in Figure 2.3, T^{n}_{active} versus T^{n}_{idle}. This suggests that a large value of T^{n}_{active} is followed by a small value of T^{n}_{idle} and viceversa, suggesting to shutdown or switch to low power states if the active period is short since it will be followed by a long idle period. But this approach runs into trouble when short busy period is followed by a short idle period, the plot in which the horizontal and vertical axes meet. This approach also fails to optimally predict if the scatter plot is not L-shaped.

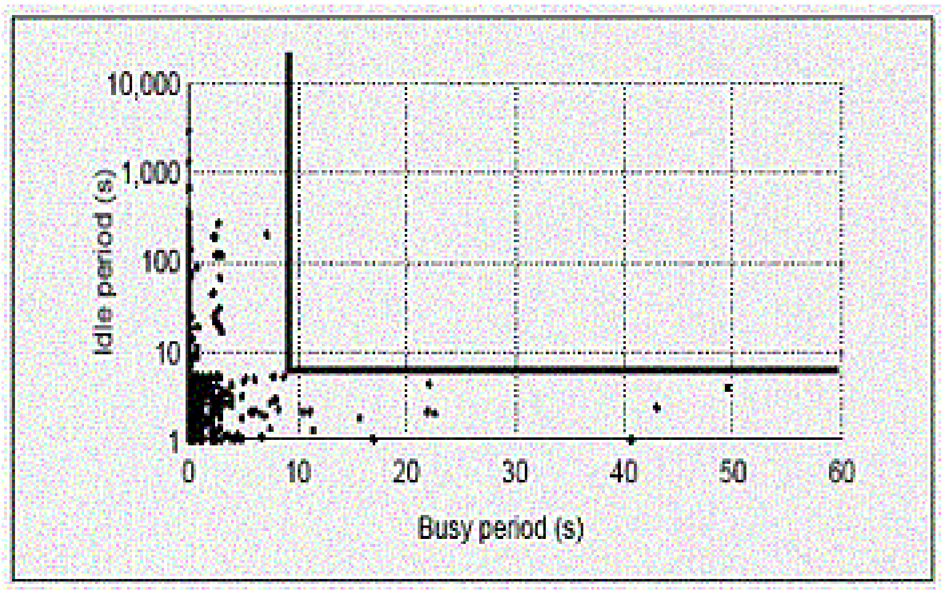

Figure 2.3 L-shaped Scatter Plot for T_{active} versus T_{idle} [23]

The major disadvantage of both these approaches is it is based on a sample traces of particular X-server application and there is no guarantee the same policy will hold good other workload traces and it involves extensive data collection and analysis.

2.2.2.2 Prediction by Exponential-average Approach

Hwang et al. [26] proposed a exponential-average approach generally used in CPU scheduling problem for predicting the idle period. The EA approach predicts the upcoming idle period as accumulative average of previous idle periods. The recursive prediction formula is given as

$$I_{n+1} = a \cdot i_n + (1 - a) \cdot I_n \tag{2.2}$$

In the formula I_n, the last predicted value is the force that opposes any change so it is called as inertia, i_n, the latest idle period is the force that pushes the predicted idle period towards the actual idle period, I_{n+1} is the new predicted value and a is the constant attenuation factor, $0 < a < 1$. The Eqn. 2.2 predicts the upcoming idle period as a function of the latest idle period i_n and previous predicted value I_n controlled by the relative weight of the recent and past history which is given by the parameter a. If $a = 0$ then $I_{n+1} = I_n$, recent history has no significance in prediction. On the other hand, if $a = 1$ then $I_{n+1} = i_n$, signifies past history has no effect. Hwang et al. [26] have set the value to be 0.5 giving equal weightage to the past and present history. The Eqn. 2.2 when expanded shows how the earlier idle periods loses weightage while predicting new value.

$$I_{n+1} = a \cdot I_n + a(1-a)i_{n-1} + \cdots + a(1-a)^n i_0 + (1-a)^{n+1} I_0 \tag{2.3}$$

Two major features Hwang et al. [26] have added to this policy are Prediction miss correction and Pre-Wakeup. As it has been stated earlier the quality of the prediction depends on two key parameters: safety and efficiency. These two features are aimed to enchance the quality of prediction when different scenarios occur.

2.2.2.3 Prediction Miss Correction

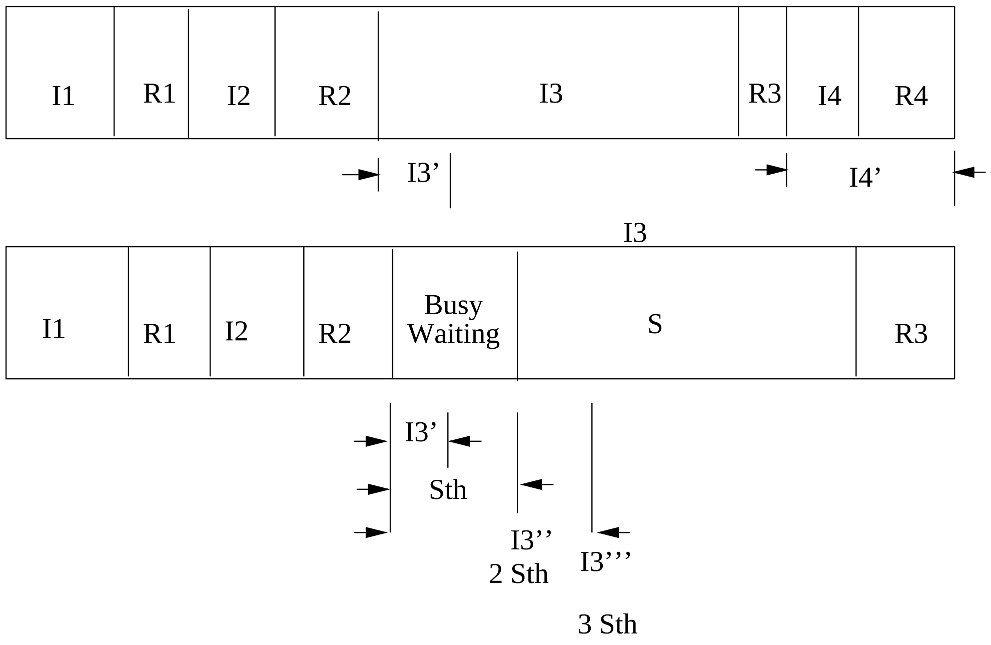

Figure 2.4 Prediction Miss Correction Using Watch Dog Scheme

Figure 2.4 shows the scenario in which a long idle period occur after continous, uniform idle periods in such as cases the normal EA method will under predict the idle time by correlating past history of idle times resulting in less power savings. Consider another scenario shown in Figure 2.4

a idle time I_4 follows the long idle time I_3 in such cases the EA policy will overpredict the idle time making the system to suffer performance penalty.

Hwang et al. [26] have proposed a watchdog scheme shown in Figure 2.4 to solve the under prediction problem. Acccording to this scheme if the predicted value is less than the threshold value, where threshold value is the minimum idle time required to obtain power savings then the system component remains in the busy waiting state and starts a timer to trace the actual idle period and the system performs another prediction once the threshold value expires. If the new predicted value is greater than the threshold then system goes to sleep state or else stays as busy waiting.

The second problem of overprediction is solved by adding a saturation condition to the equation 2.2 as below

$$if(ai_n + (1-a)I_n > cI_n)I_{n+1} = cI_n \tag{2.4}$$

where c is a constant, limiting the growth rate of I to c times per update.

2.2.2.4 Pre-wakeup

The system suffers from a delay penalty on transition from sleep state to running state as it has to perform recovery procedure.To avoid such an undesirable delay, the arrival of the next event needs to be predicted so that the system can be revived before the arrival of the event. Figure 2.5 shows two possible scenarios when applying pre-wakeup scheme.

Let I be the actual idle period, I_{pred} the predicted idle period, W the system revival time, and $D = |I_{pred} - I|$ the error of prediction. The first scenario in Figure 2.5 shows the overprediction of idle time, $I_{pred} > I$ and $D \leq W$. In such a case the system resumes its operation $I_{pred} - W$ time ahead thereby reducing the undesirable wakeup delay by $W - D$. If $D \geq W$ then the system will wakup once the original idle time expires and the pre-wakeup scheme has no effect in such a scenario with a delay penalty W. The second scenario depicts the under estimation of predicted idle time, $I_{pred} < I$. In this case there will be no delay penalty but energy gain decreases due to pre-wakeup.

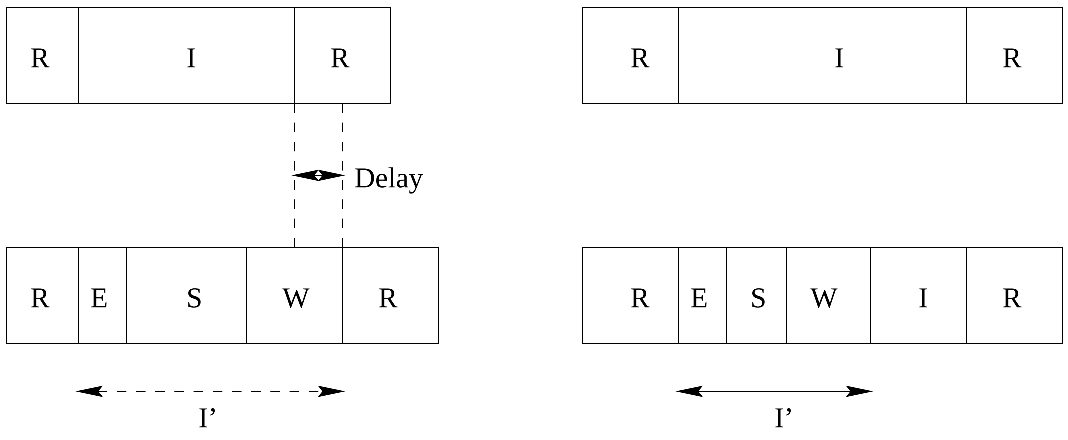

Figure 2.5 Pre-wakeup Scheme

2.2.2.5 Prediction by Adaptive Learning Tree

Chung et al. [27] have proposed the idea of idle period clustering and adaptive learning for predicting the idle time similar to the branch-prediction schemes in microprocessors. The idle period clustering technique involves computation of as many threshold values as power states in the system so each power state can be bounded by thresholds. Thus a sequence of idle periods can be transformed into a sequence of integers, each integer representing the best power state for the predicted idle time.

$$IG(t_{idle}) = \begin{cases} 0, & \text{if} \quad t_{idle} < I_0; \\ i+1, & \text{if} \quad I_i < t_{idle} < I_{i+1} \quad \text{for} \quad 0 \leq i < n; \\ n, & \text{if} \quad I_n < t_{idle}. \end{cases} \tag{2.5}$$

where t_{idle} is the predicted idle period and $I_0 ... I_n$ is the threshold values for n power states.

Chung et al. deals with the next problem which involves prediction of idle period by constructing a tree structure in which the past idle periods are encoded as tree nodes along with the predicted idle periods with the prediction confidence level. A decision to pick out a predicted value from the tree is based on path matching procedure which in turn depends on history encoded on the nodes and also on the confidence level of the predicted node. In case of correct prediction the confidence level of the predicted node is increased and viceversa. The drawback in this approach is memory resource is not constrained, there is no measure to restrict the growth of the tree requiring more memory in case tree grows arbitrarily long.

2.2.3 Stochastic Control Policy

Policy optimization is an optimization problem under uncertainity. The key features of stochastic policy are generality, high level of abstraction, and non-determinism. Benini et al. [28] have proposed a stochastic policy in which they have modeled the arrival of requests, power-state changes of the service provider under performance constraints as stationary discrete-time Markov process. The proposed policy has the following key characteristics: (i) models uncertainity in the system power consumption and transition times (ii) model complex systems with multiple power states (iii) computes globally optimum power management policies (iv) explores power and performace trade-off in a controlled fashion. The components of the proposed Markov model are

Service Requestor, is modeled as a Markov chain with state set R, where the observed variable is the nunber of requestss_r sent to the service provider (SP) during time interval t_n.

Service Provider, providing service to the incoming requests is modeled as a controlled Markov chain with S states. During each interval it can be in only one state characterized by performance and power constarints. The transistions between different states are controlled by the command issued by the power manager (PM). The transitions between different power states are probabilistic and probabilities are controlled by the command issued by the power manager (PM).

Queue, buffer to hold the requests arriving during a period. The requests are serviced in the same period with a probability depending on the power state of the system thus modeling the non-deterministic service time of a request.

Power Manager, sets the state of the service provider at the begining of each period, by issuing commands from a finite list. PM implemets a function $f : S\ X\ R\ X\ Q -> \ A$ from the state set of SP, SR, and Q to the set of possible commands A. The function is an abstarct representation of a decision process.

Cost Metrics, It is a function of both the state S and δ_S, the decision taken during state S. It is generally association of power and performance values when SP is in state S and a command is issued.

Policy optimization is performed by composing global controlled Markov chain from the Markov chains of SP and SR and the problem of finding optimal power policy under performance constraints are cast as linear program problem, producing a stationary randomized policy.

The advantages of this approach are flexibility, generality, and global optimality. But the drawbacks are policy optimization requires SP and SR to be Markov models, for non-Markov process the optimality is just approximate solutions and implemeting a randomized policy is usually more complex compared to predictive or time-out policies making it unsuitable to be implemented as hardware PM.

2.2.4 Share Algorithm

The share algorithm [21] is a member of multiplicative-weight algorithmic family, receives as input a set of ”experts”, other algorithms which makes predictions. The goal of this algorithm is to combine the predictions of various exprets to reduce error and to arrive at common consensus. The algorithm assigns one weight per expert, representing the quality of that expert predictions and predict with a weighted average of the experts prediction. After each prediction the weights of the experts are updated, the weights are slashed drastically if a prediction is misleading. The weights of good experts are kept untouched. The share algorithm has been applied to disk spin-down problem, which is nothing but deciding when to spin down the hard disk to save energy. For example different experts may be different fixed time-out values.

The share algorithm uses two parameters the learning rate, η, a real number greater than one and controls how rapidly weights of misleading experts are slashed and the share parameter, α, is a real number between zero and one, controls how rapidly poor performing experts recovers when that expert begins predicting well. On each trial the algorithm:

1. Uses a time-out equal to the weighted average of the experts:

$$time - out = \frac{\sum_{i=0}^{n} w_i x_i}{\sum_{i=0}^{n} w_i} \tag{2.6}$$

2. Slashes the weights of poorly performing experts

$$w_i^l = w_i e^{-\eta Loss(x_i)} \tag{2.7}$$

3. Shares some of the remaining weights

$$pool = \sum_{i=1}^{n} w_i^l (1 - (1 - \alpha)^{Loss(x_i)}) \tag{2.8}$$

$$w_i^n = (1 - \alpha)^{Loss(x_i)} w_i^l + \frac{1}{n} pool \tag{2.9}$$

The new w_i^n weights are used in next trial.

CHAPTER 3
LOCATION-AWARE COMPUTING SYSTEM - NAVIFIND

Location-aware computing systems are a class of wearable computers providing tailored functionality with respect to user location. Location-aware computing takes mobile computing to next paradigm by embedding computing into the context of our living activities, with minimal effort on the part of the user. Location-aware computing system is designed to be an intellegient system similar to human brain, which takes appropriate action based on the situation. Section 3.1 gives an overview of a location-aware computing system (NAVIFIND) proposed as a part of the thesis and describes the components used to build the location-aware computing system, NAVIFIND. Section 3.2 describes the system level modelling of proposed location-aware computing system.

3.1 Overview of Proposed Location Aware Computing System

NAVIFIND is the initial prototype modelled to aid the user to learn landmarks of interest of a given site. Figure 3.1 shows the block diagram of the proposed location-aware computing system and also shows the control and data flow between different components in the system. The proposed navigation system consists of following components: (1) GPS receiver (RF 8000); (2) Compact flash memory card; (3) Stereo codec; and (4) ASIC-NAVIFIND.

The RFMD GPS receiver [29] tracks the user position and provides absolute co-ordinates of the user position. The compact flash memory card is used to store the map (in the form of a graph) and audio files associated with several locations in the map. The map contains the latitude and longitude co-ordinates of all the buildings and intersections and also contains the adjacency lists for each building and intersection. The memory card is formatted to FAT16 file system and files are stored in flash card through external card reader. Stereo codec provides audio ouput to an earphone.

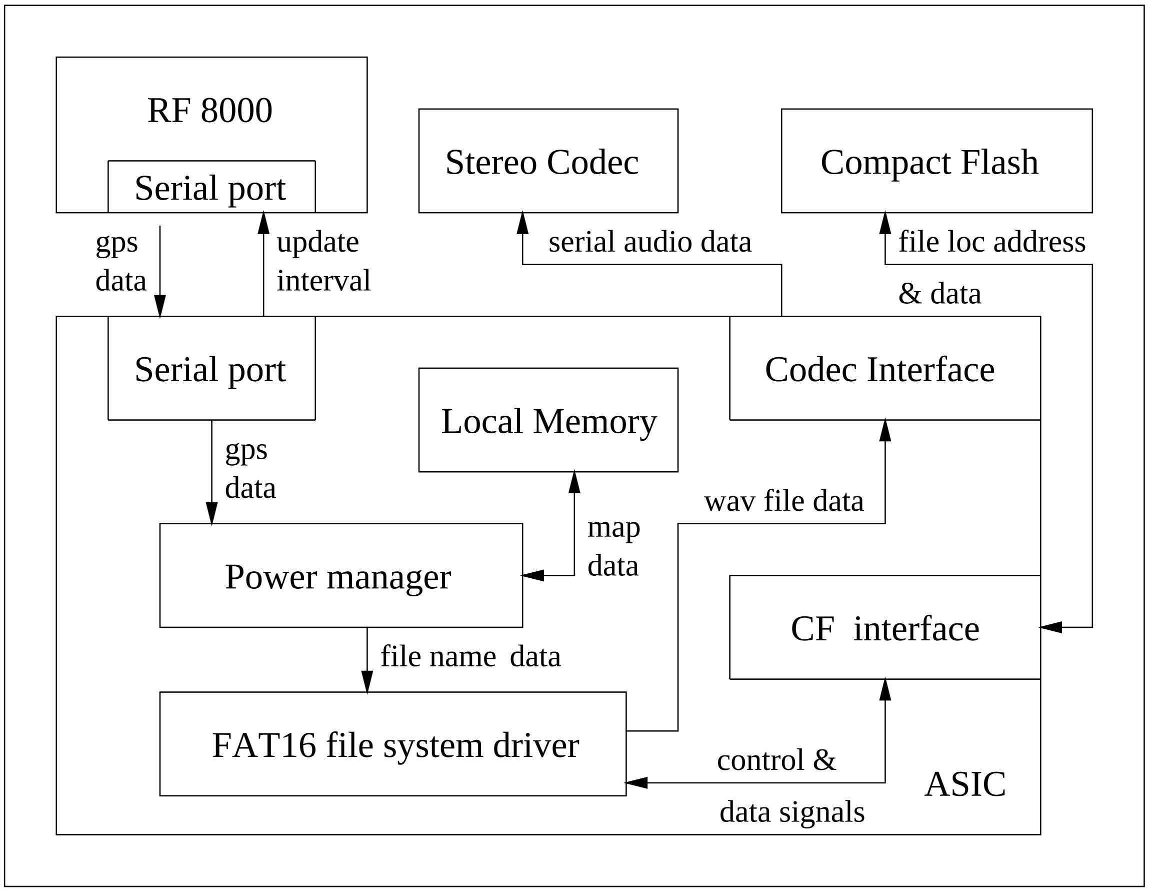

Figure 3.1 Control and Data Flow in the Proposed Location-aware Computing System

The ASIC-NAVIFIND designed has five components: (1) Serial port - UART; (2) local RAM - 2KB; (3) FAT16 file system driver; (4) Compact flash interface module; and (5) Power manager. A serial port is designed to communicate with the primary serial port of GPS receiver. The communication protocol includes sending and receiving binary messages at a BAUD rate of 19200 bps. The ASIC has a small local memory for caching the map originally stored in the compact flash card. The map is locally cached to provide higher performance and the local cache allows byte addressability which is not possible with compact flash card formatted to FAT16 file system. The FAT16 file system driver determines the sector addresses for files to be read from compact flash card by reading the boot record, FAT, and the root directory structure. The compact flash interface module handles read and write operation between the host (ASIC) and compact flash card [30]. Power manager module and the algorithm implementation is explained in section 3.3

The overall working of NAVIFIND involves guiding the tourist to learn the landmarks in a given site by giving relevant audio details about the landmarks in the site. The GPS receiver tracks the user position, which is communicated through a serial port in the form of binary messages. The ASIC checks to see whether the current user position matches with any of the landmarks stored in the map. In case of a match the ASIC activates the FAT16 file system driver module which determines the location of the corresponding audio file in a compact flash card. The compact flash interface module communicates with the compact flash card controller (internal to flash card) by generating appropriate timing signals to access the internal compact flash card registers to retrieve the audio data stored in WAV format. The stereo codec does the digital to analog transformation and plays out the audio messages through an earphone.

3.1.1 GPS Receiver

The RFMD GPS receiver (RF8000) [29] is used as a location sensing component in the proposed location-aware computing system. The GPS receiver has 12 parallel channels to track satellites and is designed for OEM use in automotive navigation, marine navigation, telematics, and asset tracking. The receiver supports 3D and 2D navigation modes. The receiver enters 3D navigation mode when four or more satellites are available with good geometry. On the other hand if fewer than four GPS satellites are available or when a fixed altitude can be used to produce acceptable result, the GPS enters 2D navigation mode. The RFMD GPS receiver is designed to operate under harsh conditions and performs robustly in situations where high signal blocake are concerns. The receiver provides a navigational accuracy of 5.8 meters in horizontal direction and 9.7 meters in vertical direction. The receiver also supports DGPS mode with accuracy of less than 1 meter.

The GPS receiver supports three power modes: Off mode, Operate mode, and Battery back-up mode. In off mode the receiver is completely inactive without any power supply. The receiver enters operate mode when an external DC supply (3.3V) is connected to the receiver primary input terminal. The battery back-up mode is used to store critical satellite data to achieve rapid TTFF (time to fast fix) when external power supply is disconnected.

Figure 3.2 shows the block diagram of RFMD GPS receiver module. GPS receiver supports four signal aquistion modes and TTFF variation is based on time to collect full ephimeris data:

- Cold start, the receiver enters the mode on start-up. The time to first fix (TTTF) in this mode is 44 seconds

- Warm start, the receiver enters the mode when there is a long power-off and battery back-up power is maintained. The receiver has valid data like position, time, almanac, and frequency parameters in memory and TTTF is 40 seconds.

- Hot start, the receiver enters the mode on software reset or short powe-off cycles when battery back-up is maintained. The receiver will have position, velocity, time, ephemeris, almanac, and frequency parameters in memory. TTTF is 10 seconds.

- Reacquistion, the receiver enters the mode on signal blockage preceded by a period of continous navigation. TTTF is 1 second.

Table 3.1 shows the pin configuration for the RFMD GPS receiver. The receiver supports communication protocols which includes sending and receiving binary messages (Proprietary communication protocol) or NMEA (National marine electronics association) messages, a standard communication protocol. The BAUD rate setting for binary messages is 19200 bps and NMEA standard

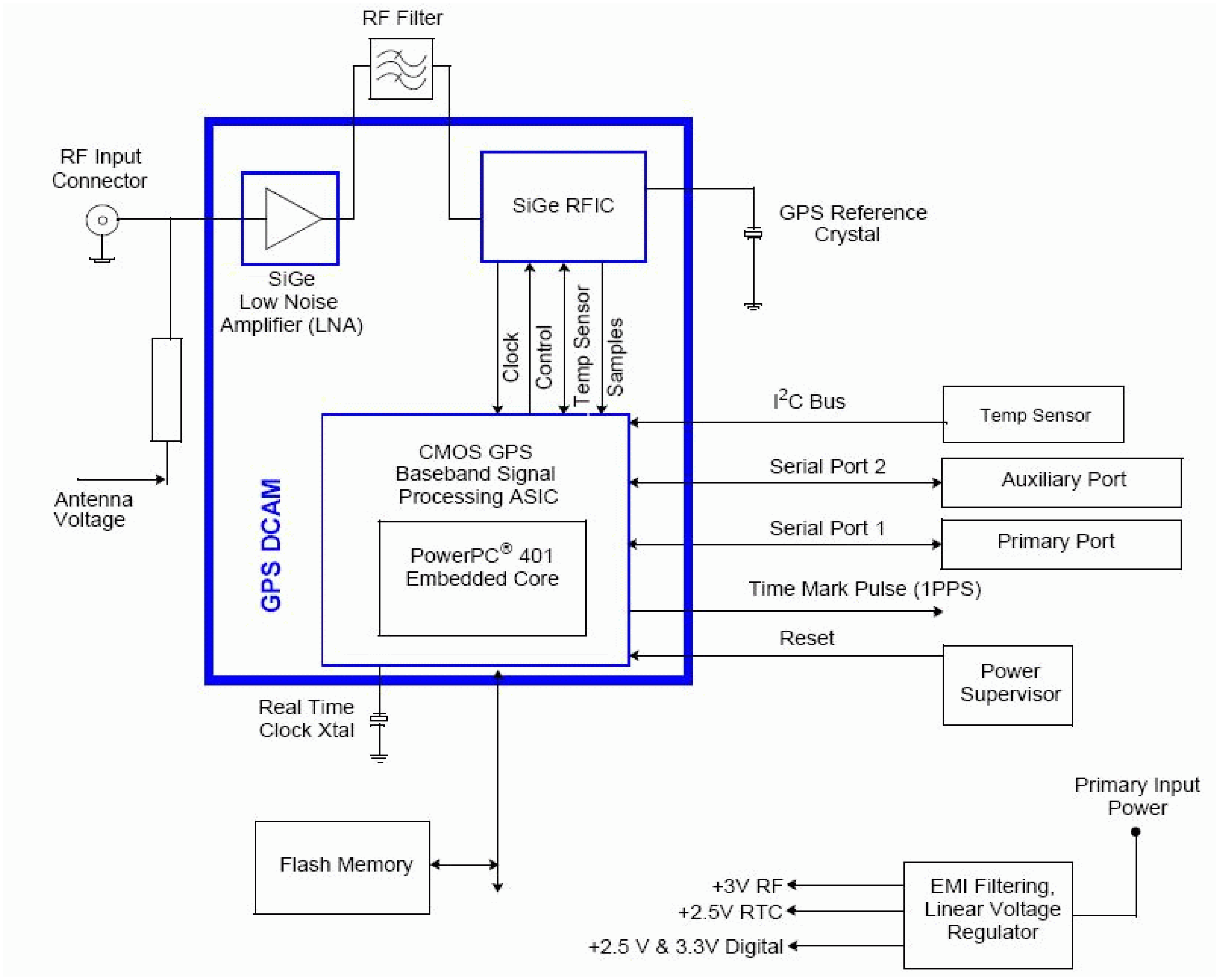

Figure 3.2 RFMD GPS Receiver Block Diagram [29]

Table 3.1 GPS Receiver Pin Configuration

Signal	Description
V3_3P	Main power input to the receiver.
GND	DC ground to the receiver.
TX1	Primary serial port transmit line.
RX1	Primary serial port receive line.
TMARK	UTC time-mark pulse, one pulse per second.
V_ANT	Provides power connection to the GPS antenna.
V2_5BU	Provides back-up power for receiver’s real-time clock.
RX2	Auxillary serial port receive port for DGPS communications.

communication setting is 4800 bps. GPS receiver manual [29] provides a detailed description of varous input/ouput messages supported in binary and NMEA communication protocol. The binary communication protocol supports an input command to set the message frequency rate. The proposed system level power management policy utilizes this input message to set the frequency update rate to minimize the energy consumption in the proposed location-aware computing system. Table 3.2 shows the power consumption values for different frequency updates.

Table 3.2 Power Consumption Information of RFMD GPS Receiver

Update frequency	Power Consumption
1 sec	400 mW
1 min	135 mW
5 mins	30 mW

3.1.2 Compact Flash Card

The Compact flash memory card uses sandisk flash technology to provide mass storage. The compact flash card has an intelligent oncard controller for data storage and retrieval, to manage interface protocols, supports Error correction code (ECC) for data protection, defect handling and management, power management and clock control. The communication between the host system and compact flash card is through the oncard controller. The compact flash card supports PCMCIA ATA (Personal computer memory card international association ATA) and true IDE standards. The compact flash card is a dual volatage product supporting 3.3V or 5V.

The proposed location-aware computing system accesses compact flash card in true IDE mode. The compact flash card supports standard ATA register and command set. Table 3.3 gives a brief description of registers available in compact flash card. The compact flash card supports various ATA commands, our proposed system uses only the read sector command to access files stored in compact flash card. Table 3.4 shows the pin description of a compact flash card used in true IDE mode. Compact flash card product description manual has a detailed pin description for other modes [30]. Table 3.5 shows the I/O decoding to access the ATA register set, which includes task file register, alternate status register and device control register.

Table 3.3 ATA Register Set

Register name	Description
Data register	16-bit read/write register for transferring data blocks between compact flash card data buffer and the host.
Error register	8-bit read only register with each bit showing the source of error.
Feature register	8-bit register write only register. It provides information about the features of the compact flash card utilized by the host.
Sector count register	8-bit register specifying number of sectors to be transferred to or from compact flash card during write or read operation.
Sector number register	8-bit register contains the starting sector number or bits 7-0 of logic block address (LBA).
Cylinder low register	8-bit register contains low order 8 bits of cylinder address or bits 15-8 of LBA.
Cylinder high register	8-bit register contains high order 8 bits of cylinder address or bits 23-16 of LBA.
Head/Drive register	8-bit register to select drive and head or bits 27-24 of LBA. The register is also used to select CHS or LBA addressing mode.
Status & Alternate status register	8-bit register to return the status of compact flash card.
Device control register	8-bit register to control interrupt request and to issue soft ATA reset.

Table 3.4 Compact Flash Card Pin Description

Signal name	Description of signal interface in true IDE mode
A0-A2	In true IDE mode address lines A0-A2 are used to select the task file registers.
A10-A3	Grounded in true IDE mode.
-PDIAG	Pass Diagnostic signal in the master/slave handshake protocol.
-DASP	Disk Active/Slave Present signal in the master/slave handshake protocol.
-CD1, -CD2	Card Detect pins used to determine if the card is fully inserted into the socket
-CS0, -CS1	-CS0 is chip select for task file registers (data to head/drive) and -CS1 is chip select for alternate status & device control register.
-CSEL	To configure the device as master or slave. when the pin is grounded then master. If open then the device is configured as slave.
D15-D00	Data signals. All task file operations occur in byte mode (D00-D7) and data transfers are 16 bits using D00-D15.
-INPACK	Not connected to host.
-IORD	Input/ouput read signal generated by thehost.
-IOWR	Input/output write signal.
-ATA SEL	The pin should be grounded to enable true IDE mode.
INTRQ	Active high interrupt request signal to the host.
-REG	Should be connected to VCC by the host.
-RESET	Active low hardware reset from the host.
VCC	5 V, 3.3 V power.
-VS1, -VS2	Voltage sense signals. -VS1 grounded to read compact flash card card information structure (CIS) at 3.3 V. -VS2 is left open.
-WE	Not used and should be connected to VCC by the host.
-IOCS16	the signal is asserted low to indicate word transfer cycle.

Table 3.5 Signal Configuration for Register Access in True IDE Mode

-CE2	-CE1	A2	A1	A0	-IORD=0	-IOWR=0
1	0	0	0	0	Data register	Data register
1	0	0	0	1	Error register	Feature register
1	0	0	1	0	Sector count register	Sector count register
1	0	0	1	1	Sector number register	Sector number register
1	0	1	0	0	Cylinder low register	Cylinder low register
1	0	1	0	1	Cylinder high register	Cylinder high register
1	0	1	1	0	Head/Drive register	Head/Drive register
1	0	1	1	1	Status register	Command register
0	1	1	1	0	Alternate status register	Device control register
1	0	1	1	1	Drive address	Reserved register

3.2 System Level Modeling of ASIC - NAVIFIND

The ASIC NAVIFIND has been modeled as a procedural executable model using a hardware desription language (VHDL). The ASIC-NAVIFIND has five components:

1. Serial port
2. local memory
3. FAT16 file system driver
4. Compact flash interface module
5. Power manager

3.2.1 Serial Port - UART

The GPS receiver outputs spatial data through the primary serial port transmit line (TX1) at 19200 bps and can receive commands as binary messages from the host through the primary serial port receive line (RX1). An Universal asynchronous receiver-transmitter (UART) in ASIC-NAVIFIND handles the communication between the RFMD GPS receiver and the host (proposed location-aware computing system). The three main components of UART are:

1. BAUD rate generator, divides the master clock to provide the bit clock (Bclk) with a period equal to one bit time to achieve 19200 bps for binary messages or 4800 bps for NMEA messages. It also provides a clock eight times the frequency of bit clock (BclkX8) to receive the serial data from GPS receiver transmit line (TX1).
2. UART receiver, implements receiver control.
3. UART tranmitter, implements transmitter control.

The serial data format for the GPS reciever input and output binary messages are:

- BAUD - 19200 bps

- Data bits - 8 bits
- Parity - None
- Stop bits - 1

The data format for NMEA messages remains the same except for BAUD rate settings which is set at 4800 bps.

UART transmitter is designed to communicate with the primary serial port receive line (RX1) of RFMD GPS receiver. The UART transmitter transmits the data stored in the transmit data register (TDR) in a standard serial data format. The host on loading the TDR register sets the transmit flag (Tflag = '1'). The UART transmitter waits for the rising edge of the bit clock ($Bclk \uparrow$) and then transmits a logic '0' as the start bit. For the next eight clock cycles of Bclk the transmitter control sends the data bits on detection of rising edge of bit clock ($BCLK \uparrow$). The data bits are followed by a logic '1' stop bit and tranmitter control clears the transmit flag allowing the host to write next data into the TDR register. The tranmitter control goes to an idle mode if Tflag is not set (Tflag = '0') indicating absence of data for tranmission or resumes data transmission if Tflag is set (Tflag = '1').

UART receiver interfaces with the primary serial port transmit line (TX1) of RFMD gps receiver. The operation of the UART receiver is as follows:

1. On detection of start bit the UART receiver reads eight data bits transmitted at Bclk frequency and shifts it into the shift register.
2. Receiver control checks the logic value of the stop bit. If it is found to be '1' then the transmission is successful.
3. On successful transmission the contents of shift register are loaded onto host register for the host to read and shift register could be used for shifting next transmitted data.

The receiver controller has a clock (BclkX8) input eight times the frequency of transmitter control clock (Bclk) to avoid setup and hold times problems. The received data bits are sampled eight times during each bit time. The serial data transmitted is asynchronous and transmitting device clock (in our case the transmitting device is RFMD GPS reciever) may not be in complete synchronization with the receiver clock. So reading the transmitted data at the rising edge of Bclk could lead to problems like reading wrong data or reading spurious data etc., to avoid this problem the receiver control samples the received bit at eight times the frequency of transmitted bit and reading it in the middle of each bit time thereby providing maximum reliability. Figure 3.3 shows how the sampling is done on the received bit.

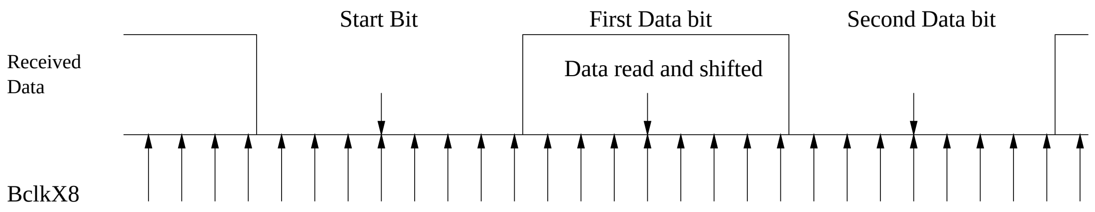

Figure 3.3 Sampling in UART Receiver Control

UART BAUD rate generator is used to generate the clock inputs for the UART transmitter and receiver control. Figure 3.4 shows the block diagram of a BAUD rate generator. The BAUD rate generator divides the system clock to provide Bclk and BclkX8 to the transmitter and receiver control. The BAUD rate generator can be configured to select different BAUD rate settings. For example the system clock will be divided in such a way to provide 19200 bps for binary message protocol or 4800 bps for NMEA protocol. For example if the system clock is 8 MHz and we want BAUD rates 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400. The maximum clock frequency required is 38400 x 8 = 307200 for BclkX8. To achieve this the system clock (8 MHz) should be divided by 26 for 38400 bps and division by 52 for 19200 bps and so on.

3.2.1.1 Clock Generation Problems in BAUD Rate Generator

Due to the asynchronous nature of data tranmission the primary goal of an UART design is to provide synchronization mechanism between the transmitting and receiving device. As explained in order to achieve a maximum frequency of 307200 (38400 x 8) for 38400 bps setting the BAUD rate generator needs to divide the system clock (for example 8 MHz) by 26. But in reality to get the required maximum frequency the system clock (8 MHz) has to be divided by 26.04. Since integer

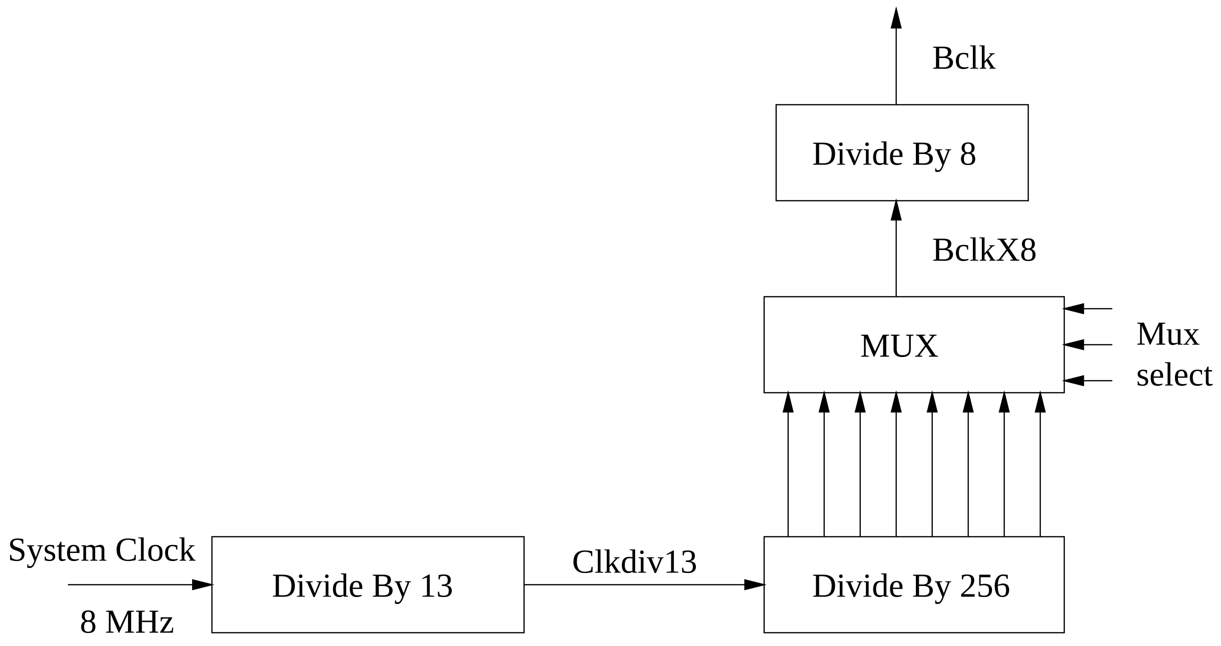

Figure 3.4 BAUD Rate Generator

division is the only possible way we have to either accept a small error or adjust the system clock frequency to 7.9877 MHz. Table 3.6 shows the scenario if UART operates with a small margin of error arising out of integer division. The arithmetic error arising out of clock division is one of the important source of error in UART communication [31].

Table 3.6 Clock Generation Problems Affecting the BAUD Rate Settings

Mux Select input	BAUD rate
000	38462
001	19231
010	9615
011	4808
100	2404
101	1202
110	601
111	300.5

Table 3.6 clearly shows that the transmitting device will be sending more data bits per second than the required BAUD rate setting. The receiver on another host will be operating on a clock according the BAUD rate setting. In this case receiver clock in another host will have a clock fre-

quency lesser compared to transmitting device clock frequency. It can be argued that the receiver will always read the data in the middle of the bit clock compensating for the error due to slight frequency variations. As we can see from the table 3.6 as we move from one setting to another setting the number of extra bits transmitted gets doubled. The effect is more pronounced for larger BAUD rate settings. For example for 38400 bps 62 extra bits are transmitted affecting the synchronization between the transmitting and receiving device. Over the long run the receiving device will read the data not in the middle of the bit clock but at a later period, which will eventually lead to decoding wrong messsages or skipping start or stop bit causing transmission failure. The same error occurs in the opposite direction if the host receiver operates at a higher frequency than the BAUD rate setting of a communication protocol.

3.2.1.2 Resynchronization of Transmitter and Receiver Clocks to Achieve Ideal Baud Rate Setting

To compensate for the error due to integer division of system clock in BAUD rate generator the transmitter (Bclk) and receiver (BclkX8) is resynchronized at regular intervals. In our proposed location aware system serial port transmission line is interfaced with Receive (RX1) port of GPS receiver. The serial port transmission line in ASIC-NAVIFIND will be transmitting 19231 bits per second compared to a ideal setting of 19200 bps. For 19200 bps each bit will have a clock cycle period of 52083.33ns but in our case for 19231 bps each bit transmitted will have a clock cycle period of 52000ns. Each bit is transmitted 83.33ns earlier than required. If this error is allowed to accumulate then 19200 bits will be transmitted in 0.9984 seconds rather than in 1 second leading to synchronization problem with the receiver in another device. In order to avoid accumulation of error over one second period we introduce delay in the clock at regular intervals which compensates for error accumulated till that clock period. A delay of 2500ns (or 20 system clock cycles) is introduced after the transmission of every 30 bits of data on rising edge of bit clock ($BCLK \uparrow$). As it can be seen on every 30 bits of transmission the error accumulated compared to ideal bps setting (19200 bps) is 2500ns i.e., the transmitter in ASIC-NAVIFIND would have transmitted data 2500ns earlier compared to ideal setting. As explained above Bclk and BclkX8 are generated by dividing the system clock. So 2500ns corresponds to 20 system clock cycles and by resetting the counter for 20 clock period cycles introduces a delay of 2500ns on the bit clock and synchronizes the transmitter clock with the receiver clock in another host. It could be verified by dividing 19200 bits into groups of 30 bits will give 640 groups and introducing a delay for each group will result in introduction of delay of 0.0016 seconds (640 x 2500 ns) for 640 groups. By adding the introduced delay of 0.0016 second with the transmission time of 0.9984 second (19200 * 52000) will provide us an ideal setting of 19200 bits per second. The delay insertion method also does the job of synchronization along with mid bit synchronization in receiver control in UART.

3.2.2 FAT16 File System Driver

ASIC-NAVIFIND implements the FAT16 file system driver to access the files stored in the compact flash card. Addition, deletion or modification of files are carried out externally through flash card reader. The FAT16 file system provides a simpler way to access and store the files and frees the location-aware system from handling file management related issues. Navigation related files like spatial database (for example graph of a location) and audio files coresponding to landmarks of interest can be stored in the compact flash card inserted into the flash card reader, which can be connected to PC or laptops supporting ports to interface with the flash card reader. The file storage requires just a drag and drop once the flash card is recognised by the PC. The FAT16 file system driver in the PC will write the relevant details about the file in the FAT16 file data structures which includes boot record, file allocation tables, and directory structure. The ASIC-NAVIFIND has a FAT16 file system driver to access the files while the user is on the move and to provide details about the current location to the user.

There are four logical parts in a FAT16 file system they are:

1. Boot Sector

2. File Allocation Table

3. Directory Structure

4. Data Space

3.2.2.1 The Boot Sector

The boot sector contains the boot parameters. The boot parameters contains information about the way compact flash is organised like number of sectors per cluster, number of reserved sectors etc., Table 3.7 shows the contents found in a boot sector.

Table 3.7 Structure of Boot Record

Offset	Description	Length in bytes
00h	Jump instruction	3
03h	OEM name	8
0Bh	Bytes per sector	2
0Dh	Sectors per cluster	1
0Eh	Reserved sectors	2
10h	Number of FAT copies	1
11h	Maximum Root directory entries	2
13h	Total sectors	2
15h	Media Descriptor	1
16h	Sectors per FAT	2
18h	Sectors per track	2
1Ah	Number of heads	2
1Ch	Number of hidden sectors	4
20h	Huge sectors	4
24h	BIOS drive number	2
26h	Boot signature	1
27h	Volume ID	4
2Bh	Volume name	11
36h	File system name	8

The boot sector is the first sector in FAT16 file system and the parameters found in the boot sector are used for calculating the address of other three areas in the FAT file structure. The formula for calculating the addresses of other three areas of FAT16 file structure based on the boot parameters are:

$Boot\ sector\ address = First\ sector$

$FAT\ address = Boot\ sector\ address + number\ of\ reserved\ sectors$

$Root\ directory\ address = FAT\ address\ +\ number\ of\ sectors\ per\ FAT\ (16h)$
$*\ Number\ of\ FAT\ copies\ (10h)$

$Data\ space\ address\ = Root\ directory\ address + (Maximum\ Root\ directory\ entries\ (11h) *$
$32)/Bytes\ per\ sector\ (0Bh)$

The parameter Maximum Root directory entries indicate how many maximum files can be created in the root directory of the disk. If there are subdirectories then there can be several more files and sub-sub-directories within it.

3.2.2.2 Root Directory Structure

The root directory structure contains a 32 byte entries for various files and subdirectories in the root directory. The root directory structure contains the starting cluster number for each file. Table 3.8 shows the directory entry parameters. There are four types of directory entries they are:

- File
- Subdirectory
- Volume labels
- Long file name specification

Figure 3.5 shows the parameters of an attribute byte associated with each directory entry. If the volume and sub-directory bit are set then it is a long file name specification. If the voulme bit is set then it is a volume label and if directory bit is set then it is a subdirectory. If both volume and subdirectory are reset then it is a file.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Unused	Unused	Archive	Sub directory entry	Volume label entry	System	Hidden	Read Only

Figure 3.5 Structure of an Attribute Byte

Table 3.8 32 Byte Directory Entry Structure

Description	Size in bytes
Filename	8
Extension	3
Attributes	1
Reserved	1
Creation time (millisecond portion)	1
Creation time	2
Creation date	2
Last access date	2
High word of starting cluster number	2
Last modification time	2
Last modification date	2
Starting cluster number of file/subdirectory	2
File size	4

3.2.2.3 File Allocation Table

The File allocation table contains information about the clusters used by each file, unused disk space, and space that cannot be used because of defects. The FAT facilitates the freedom to allocate any disk space available to files. i.e., the clusters allocated to a file need not be contiguous. The starting cluster of a file is obtained by reading the file directory entry and using the starting cluster as an index the file remaining cluster if allocated can be tracked. The FAT keeps track of the following:

- Unused cluster indicated by 0000h.
- Reserved cluster indicated by FFFh-FFF6h.
- Bad cluster indicated by FFF7h.
- Last cluster of file indicated by FFF8h-FFFFh.
- Next cluster of file indicated by any other number.

For example if a file has more than one cluster allocated to it then the directory entry for the file will give the starting cluster number. By indexing the FAT with starting cluster number and reading the data value from the corresponding FAT location the address of the next cluster allocated to the file can be found. On end of file the last cluster allocated to file in FAT will contain the entry FFFFh. The first two FAT entries are unused indicating cluster number 0 and cluster number 1 cannot be used for data storage. The formula shown below shows how to calculate the actual physical address for the data section of a file from the cluster number in LBA (Logic Block Addressing) mode.

$File\ starting\ LBA\ address = Data\ space\ address + (starting\ cluster\ number - 2) * sectors\ per\ cluster$

3.2.2.4 Summary of Operations to Access a File in Compact Flash Card

The following procedure provides a brief overview of various steps involved in accessing a file stored in a compact flash card formatted to FAT16 file system.

1. Reading the boot sector to access the boot sector parameters. Boot sector is usually the first sector in FAT16 file system.

2. Calculation of FAT, Directory root structure and Data space address from boot sector parameters.

3. Searching through the Root directory structure by reading the directory entry to find a match for the required file using file name as the search attribute.

4. On match the starting cluster number is converted to LBA address to access file data section.

5. Reading the data stored in the file data section coresponding to starting cluster number.

6. Indexing the FAT using the starting cluster number to determine if the file has more clusters allocated to it. FFFFh in FAT entry for the corresponding cluster entry indicates end of file. If file has more clusters then steps 4 and 5 are repeated until end of file.

3.2.3 Compact Flash Interface and Local Memory

The compact flash interface handles the read/write timing requirements for accessing the compact flash card by the host ASIC-NAVIFIND. The compact flash interface is the controller through which the FAT16 file system driver can access compact flash card. It provides mechanisms to read and write the task file registers. It also enables the LBA addressing mode in compact flash card by invoking enable LBA mode command in compact flash card (writing a value 11100000 to drive/head register). Section 3.1.2 provides a detailed explanation on signal interface and pin configuration between the ASIC-NAVIFIND and compact flash card in true IDE mode.

A local memory is used to cache the files accessed from the compact flash card. It is also used by the power manager to store temporary results. A location-aware system requires frequent access to map database. By accessing the map from the compact flash frequently will affect the performance of the system. The reasons for it are two fold (1) The read/write timing requirements for the compact flash card are more compared to an SRAM. Since accessing compact flash card requires lot of pre data access operations like checking the status register to determine the status of the card (busy or ready), writing sector count register, sector number register, cylinder high and low register, drive/head register and command register. and (2) The compact flash card doesnot allow byte accessibility and therfore cannot be used for temporary storage.

3.3 Power Manager

The proposed *location-aware* system is designed to be power-aware to reduce the energy consumption and to enhance the portability of the system. The *location-aware* systems provides tailored functionality with respect to user location. In our proposed system, the tailored functionality is providing audio information about the locations of interest in the map. The proposed algorithm predicts the time the user might take to reach the locations of interest from current location and switch the system to low-power mode for the predicted time interval.

3.3.1 Overview of Proposed Power Management Algorithm

The conventional *location-aware* system without any power policy will keep the position tracking component namely a GPS receiver active continously wasting significant amount of power and energy. Another drawback is the frequent search of map to determine locations of interest to execute tailored functionality. A simple and no overhead solution is to have a fixed time-out policy.

The drawbacks of fixed time-out policies are : (1) unwanted power consumption when waiting for time-out to expire; and (2) difficulty in selecting an optimal time-out interval.

The proposed algorithm belongs to the category of adaptive predictive policy. The proposed power policy overcomes the above mentioned shortcomings by predicting the time-out interval beforehand and dynamically varying the time-out interval to maximize the energy savings. A *location-aware* system executes only on reaching predetermined locations or learned locations. On the same lines our proposed *location-aware* system plays out details of location only on reaching the predetermined set of locations in the map. The proposed power management algorithm conserves power by predicting the time user might take to reach the nearest location and setting GPS update frequency accordingly. Table 1 shows the power consumption values for different GPS update frequencies. Another advantage of this policy is that since activities of most of the components in the system are location dependent all the dependent components can be switched to low-power state or can be shutdown to save power and energy.

Figure 3.6 shows the block-level view of the power manager. The proposed power manager has three modules: GPS-to-real-world position conversion module, nearest-location-search module, and user speed estimation module. The input to the algorithm is a map represented as graph with buildings (or landmarks) and intersections as nodes and path between them as edges. Each node has a unique latitude and longitude co-ordinates. The graph also has details of orientation of each edge, running North-South or East-West. The current version of algorithm handles only manhattan routes.

Power manager is the top level module that co-ordinates the operation of GPS-to-real-world position conversion module, Nearest-location-search module and User speed estimation module. When the system is switched ”on” the power manager gets the latitude and longitude data of current location from GPS receiver and loads it into GPS-to-real-world position conversion module. Since the the system was in off state there wouldn’t be any prior history. With current location as reference power manager activates the Nearest-location-search module. Due to the absence of prior history the user travel speed is set to maximum speed i.e., in our implementation it is 2 m/s. The Nearest-location-search module calculates all the positions that could be reached with maximum

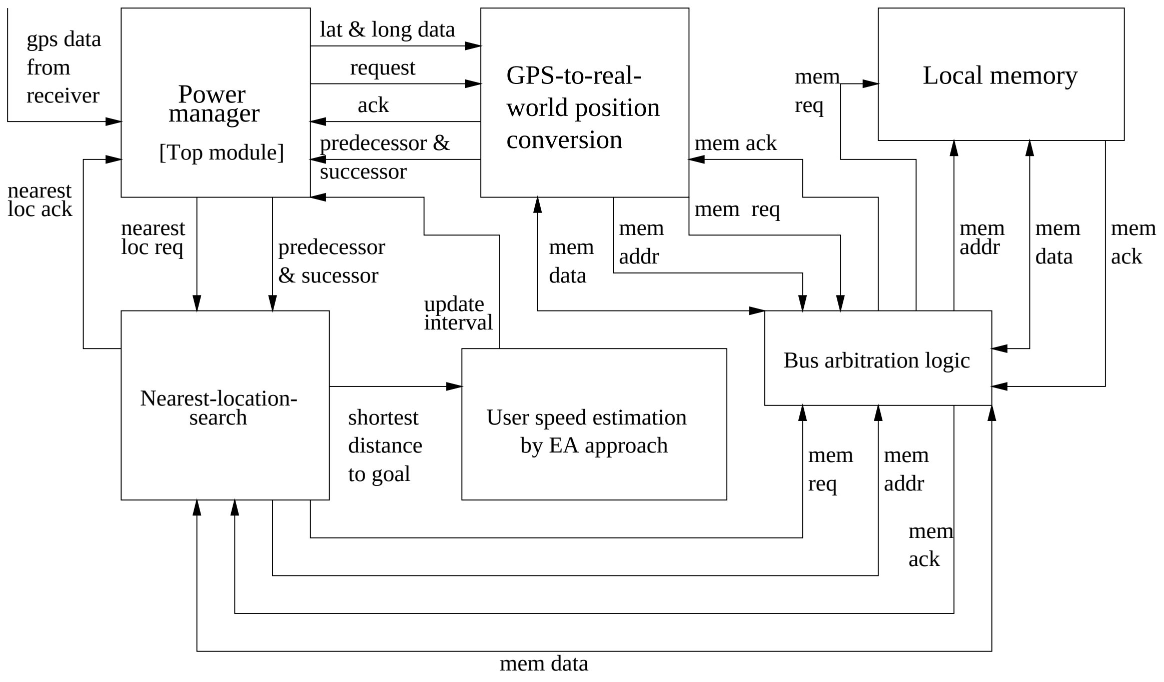

Figure 3.6 Block-level View of Power Manager

speed depending on the routes in the map from current location. After one minute a new GPS update is obtained and new position in graph is computed. The user's actual speed is calculated based on distance travelled from previous position to current position. The initial GPS update of one minute is set to calculate actual user speed and avoid the use of arbitary value of 2 m/s as user speed. With the currently computed user speed the Nearest-location-search module computes the distance and time to reach the nearest location or intersection that may be reached through more than one route i.e., a cycle. The time to reach the nearest location or intersection that results in a cycle is used as next GPS update time interval. Cycles are considered to avoid under or over estimation of actual user speed. A major drawback of using recently computed actual user speed for predicting the future GPS update interval is that the user might travel at different speeds during different time periods. There is no guarantee the user will travel at the same speed he/she has travelled recently. To overcome this problem the user speed to be used for setting future GPS updates is estimated by exponential-average approach used earlier in the CPU scheduling problem [32] and more recently for predictive system shutdown in event-driven applications [26]. The exponential-average approach estimates the user velocity as a cumulative average of previously calculated user velocities.

3.3.1.1 GPS-to-real-world Position Conversion Module

Figure 3.7 shows the pseudo code for algorithm to convert GPS latitude and longitude data to graph nodes in the map. The algorithm receives as input latitude and longitude data of current position. Lines 4-16 in Figure 3.7 determines the graph nodes that matches the latitude and longitude data of current location. In case of a perfect match i.e., latitude and longitude of current location exactly matches latitude and longitude of a graph node (line 5). In other words the user is exactly on or near a building or intersection. Lines 6-12 checks whether the exactly matched graph node is an intersection or building (or landmark). If the match is a building then the system plays out the details of the location. If the match is an intersection, then *predecessor* of matched intersection is found and distance to reach the current position from previous update position is calculated (lines 8-10). The actual user speed is calculated by dividing distance travelled from last update position to current position by update time set at last update position (line 11). In the absence of a perfect match, i.e., when user is between two nodes, Lines (13-15) computes all the graph nodes having same latitude or longitude as the current position's latitude or longitude.

Figure 3.8 shows an example of user current location between two nodes. In Figure 3.8, U represents user current position. The intersections 3 and 4 will have the same latitude as the user current position but the intersections longitude varies from current position's longitude. So the graph nodes 3 and 4 are pushed into a Stack S (line 14). Lines 18-33 performs boundary checks to find the nodes between which the user current position lies. One of the end nodes will be a node in the Stack S. For each node in the Stack S it's adjacent nodes and orientation of adjacent nodes are checked to determine the nodes between which the user is currently located. On finding the end nodes the path of user travel is tracked to determine the distance travelled by the user. The predecessor function determines which one of the end nodes is a *predecessor*. Lines 26-30 sets other end node as successor. The *predecessor* and *successor* nodes are used for predicting the route the user might take from current location.

```
1  Algorithm GPS-2-real-world-position-data
2  Input ← current_gps_lat_data and current_gps_long_data
3  begin
4  foreach graph_node ∈ G
5      if (current_gps_lat_data = graph_node_lat_data  &&
             current_gps_long_data = graph_node_long_data) then
6          if (graph_node = landmark) then
7              Audio message about current location
8          else if (graph_node = intersection) then
9              predecessor_node ← find_predecessor(graph_node,NULL)
10             distance_travelled ← track_path()
11             actual_user_speed ← distance_travelled/update_time
12         end if
13     else if (current_gps_lat_data = graph_node_lat_data  or
               current_gps_long_data = graph_node_long_data) then
14             S ← Push(graph_node)
15     end if
16 endfor
17 pred_n_succ_found ← FALSE
18 while pred_n_succ_found ≠ TRUE
19     single_match_node ← Pop(S)
20     foreach adjacent_node ∈ single_match_node
21         if (current_gps_lat_data,current_gps_long_data is between
               adjacent_node_lat_data,adjacent_node_long_data  &&
               single_match_node_lat_data,single_match_node_long_data) then
22             predecessor_node ← find_predecessor(single_match_node,adjacent_node)
23             distance_travelled ← track_path()
24             actual_user_speed ← distance_travelled/update_time
25             pred_n_succ_found ← TRUE
26             if (single_match_node = predecessor_node) then
27                 successor_node ← adjacent_node
28             else
29                 successor_node ← single_match_node
30             end if
31         end if
32     endfor
33 end while
34 return predecessor_node, successor_node
35 end Algorithm
```

Figure 3.7 Pseudocode for GPS-to-real-world Position Conversion Algorithm

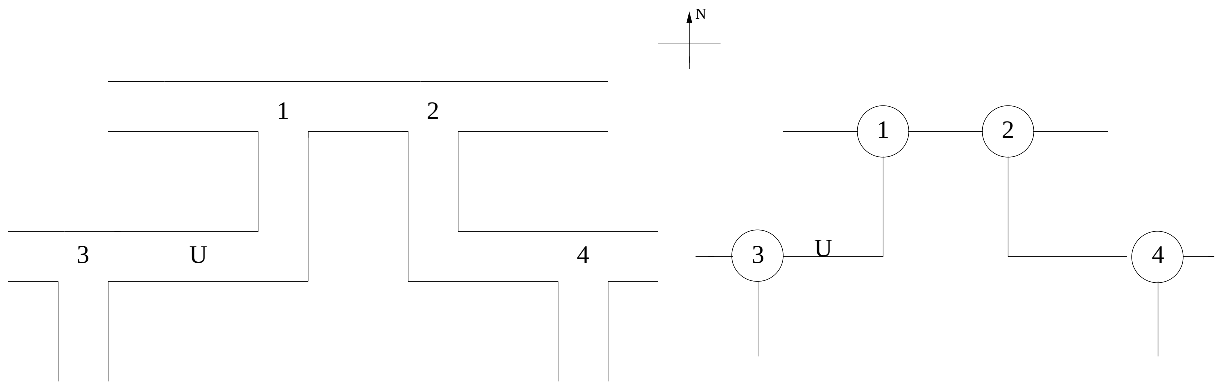

Figure 3.8 Example for Converting GPS Data to Real-world Data (a) Original Map (b) Equivalent Graph Representation

```
Algorithm Nearest-loction-search
Input ← predecessor_node, successor_node,G
goal = landmark or cycle_node
current_best = successor_node
Best = Best ∪ current_best
goal_found = FALSE
while (goal_found)
      foreach vertex v ∈ adj_lists[current_best]
            if v ∈ Adjacent_arr then
                  Cycle_arr = Cycle_arr ∪ v
                  Adjacent_arr = Adjacent_arr ∪ v
                  Distance_arr = Distance_arr ∪ dist[v]
            else
                  Adjacent_arr = Adjacent_arr ∪ v
                  Distance_arr = Distance_arr ∪ dist[v]
            end if
      endfor
      current_best = Extract-Min()
      Best = Best ∪ current_best
      if (current_best ∈ Cycle_arr or landmark) then
            goal_found ← TRUE
      end if
end while
return Distance_to_nearest_node
end Algorithm
```

Figure 3.9 Pseudocode for Nearest-location-search Algorithm

3.3.1.2 Nearest-location-search Module

The Nearest-location-search module is a modified Dijkstra's single source shortest path algorithm to handle cycles in the graph. Figure 3.9 shows the pseudo code for Dijkstra's single source shortest path algorithm to find the nearest landmark. Each node in a graph can only have a maximum of four adjacent nodes, one each in all directions due to manhattan route considerations. The data structures used for implementing Dijkstra's algorithm are Adjacency array, Best array, Distance array, and Cycle array. The Adjacency array contains list of adjacent nodes for the best node selected, Distance array contains the list of distances from adjacent nodes to best node selected, Best array maintains a list of best nodes selected, and Cycle array contains the nodes that can be reached by more than one route from source node. The goal or destination node is the nearest location or a node that forms a cycle, whichever is earlier. The node that forms a cycle is considered to be a intermediate goal due to the fact Dijkstra's algorithm doesnot handle cycles in a graph. The implemented algorithm avoids this problem by setting next update lesser than time to reach the node that forms cycle, so there will be only one path from source to destination.

The inputs for the algorithm are predecessor and successor nodes determined from GPS-to-real-world position conversion module and the map G. The initial best node is the successor node, since predecessor is the node visited earlier by the user (line 4). Lines 7-23 computes the shortest distance to reach nearest location or node that forms a cycle. In lines 8-17 the algorithm finds the nodes adjacent to current best node and adds those nodes to Adjacency array and corresponding distances into the Distance array. Line 9 checks for the prior occurrence of adjacent node in the Adjacency array. If found then the adjacent node can be reached through two different paths so it is added to the Cycle array. The Extract_Min function finds the next best node not present in the Best array. Line 20 checks whether the best node selected is a landmark node or it belongs to Cycle array. If it is the case the next update is set to time (calculated using estimated speed by exponential-average appraoch) to reach the landmark node or set to marginally less value in case of node in Cycle array. The reason for setting next update to a marginally lesser value is (1) the node in the Cycle array can be reached by more than one path, and (2) estimated user speed by exponential approach may have underpredicted the user speed. By setting to a lesser value the algorithm makes sure the user might not travel past the node in the Cycle array.

3.3.1.3 User Speed Estimation Using EA Approach

The speed of user travel can be determined by the general formula, dividing distance travelled by time taken but the problem with this approach is that the user is not bound to travel at same speed at all the time during the course of the distance or one cannot be sure the user will travel at same speed in future. So taking current speed as a base value for setting the next update might be misleading and will hamper the real-time performance of the system for example missing a location etc. To avoid these problems we propose a method which was previously used in CPU scheduling problems [32] and as a system shutdown technique for energy savings of event-driven computations proposed by Hwang et al., [26], the exponential-average approach.

In this approach the predicted user velocity will be a cumulative average of previous user velocities. The Hwang et al approach predicts the length of the upcoming idle period by the cumulative average of previous idle periods [26]. Similarly, we can predict the next user velocity as the cumulative average of previous user velocities. The user velocity estimation formula is

$$V_{n+1} = a \cdot v_n + (1 - a) \cdot V_n \tag{3.1}$$

V_{n+1} is the new predicted user velocity, V_n is the last predicted user velocity, v_n is the actual user velocity measured by dividing distance travelled by time taken, and a is the attenuation factor in the range between 0 and 1. In our implementation the attenuation factor a is set to 0.5. The attenuation factor determines the weightage of the recent estimates versus past estimates.

CHAPTER 4

EXPERIMENTAL RESULTS

This section presents experimental results obtained for proposed location-aware computing system and the efficiency of the proposed system level energy optimization policy. The results presented for proposed location-aware computing system and power policy includes:

- Resynchronization of local transmitter and receiver clock to acheive ideal Baud rate setting in UART.
- Energy savings obtained in proposed system level energy optimization policy.

4.1 Location Aware Computing System - NAVIFIND

The proposed location-aware computing system has been designed to communicate with external components like RFMD GPS receiver, and Compact flash card. The location-aware system has a central component implemented as an ASIC (NAVIFIND), which acts as a system manager co-ordinating overall operations of the system. The system manager is implemented in a hardware description language, VHDL. The major components of the system manager includes: (1) Serial Port - UART; (2) FAT16 file system driver; (3) Compact flash interface; (4) Stereo codec interface; and (5) Power manager. The Serial port serves as a interface to communicate with RFMD GPS reciever and Compact flash interface is designed to communicate with compact flash card. The external components like GPS receiver and compact flash card are also modelled in VHDL for simulation purposes. Each component of system manager has been tested individually for their correct operation. The overall operation of the system with all the components integrated are tested for different testbench settings. The testbench settings includes:

1. Testing communication protocol in asynchronous data transmission.

2. Reading files stored in FAT16 file system format in a compact flash card.

3. Random generation of maps.

4. Random generation of user travel pattern in a map.

5. Randomization of user walking speed over the course of travel.

The First testbench setting is used to test the validity of data transmission or reception of UART. It tests the communication protocol between the RFMD GPS receiver and the serial port in the ASIC-NAVIFIND. The next setting generates different file names which may or may not be stored in compact flash card. The setting is used to test the validity of FAT16 file system driver. If the file name is valid the file system driver must determine the LBA address of the file stored in compact flash card by accessing the data structures (Boot record, FAT, Root directory structure) of FAT16 file system. The remaining three sections are used to estimate the efficiency of the proposed power management policy compared to a system with standard time-out policy and a system with no power policy.

4.2 Synchronization Protocol in Serial Port

The serial communication between RFMD GPS receiver's serial port and the ASIC-NAVIFIND serial port are carried out asynchronously. To establish synchronization the data tranmission between RFMD GPS receiver and ASIC-NAVIFIND must be synchronized. In asynchronous serial transmission the sender and the receiver have a common timing parameter i.e., baud rate setting. Chapter 3 gives a detailed description about the problems that will arise in establishing synchronization in asynchronous serial transmission. One of the important source of error in UART communication is the artithmetic error in baud rate setting. The local clocks for the UART are generally generated by integer division of system clock. But standard UBRS (UART Baud Rate Setting) requires division of system clock by a floating point number. For example, a system with 8 MHz system clock requires division by 52.08 to obtain 19200 UBRS. So the value 52.08 is rounded off to

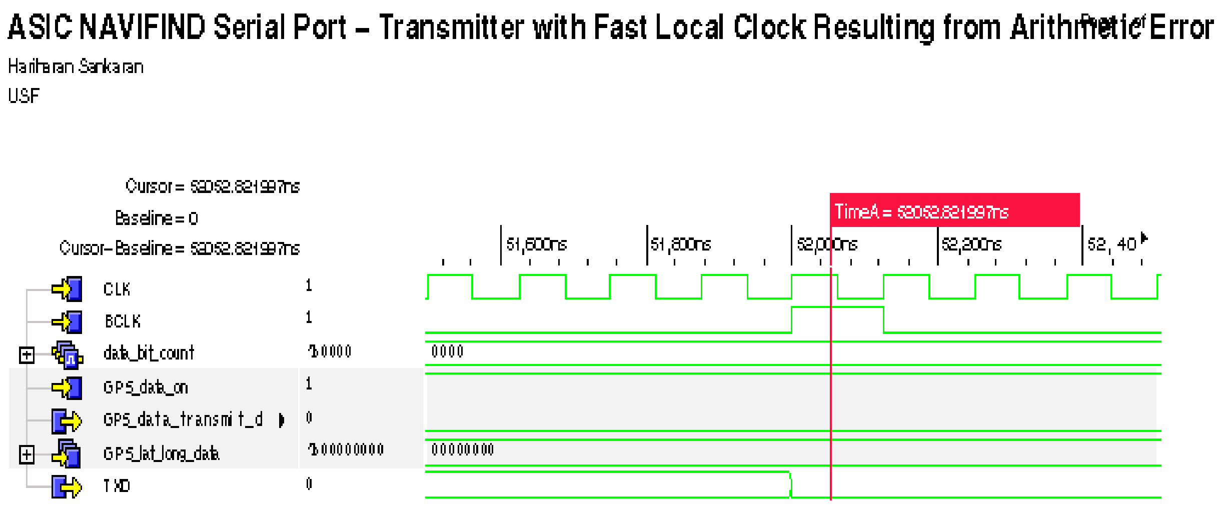

Figure 4.1 Arithmetic Error Affecting the Baud Rate Setting Between ASIC-NAVIFIND Transmitter and RFMD GPS Receiver

nearest integer value of 52. The arithmetic error due to this rounding off increases the UART Baud Rate Setting (UBRS) to 19231.

The arithmetic error causes two problems. The worst case is a slow receiver trying to listen to a sender transmitting too fast. The other extreme is fast receiver listening to a slow sender. To avoid the difference in baud rate setting between the transmitter and receiver, the local clocks has to be resynchronized at regular intervals to avoid propogation of error resulting in receiver decoding messages incorrectly. Figure 4.1 shows the ASIC-NAVIFIND transmitter transmitting the start bit at 19.23 KHz (52000 ns) instead of 19.20 KHz (52083.33 ns) required for 19200 UART Baud Rate Setting. The arithmetic error is due to the above mentioned integer division by 52. Similarly Figure 4.2 shows a fast receiver with 8 samples per bit cell operating at a frequency of 153.84 KHz compared to a ideal Baud rate requirement of 153.60 KHz.

Figure 4.3 shows the resynchronization of local transmitter clock to achieve ideal Baud rate setting between a fast transmitter and a slow receiver. To avoid the propogation of error due to fast transmitter a delay of 2500 ns is introduced to the local clock of ASIC-NAVIFIND tranmsitter (Bclk). The delay will be introduced for every 30 clock cycles of transmitter clock (Bclk). The delay introduction compensates for arithmetic error accumulated over a period of 30 clock cycles.

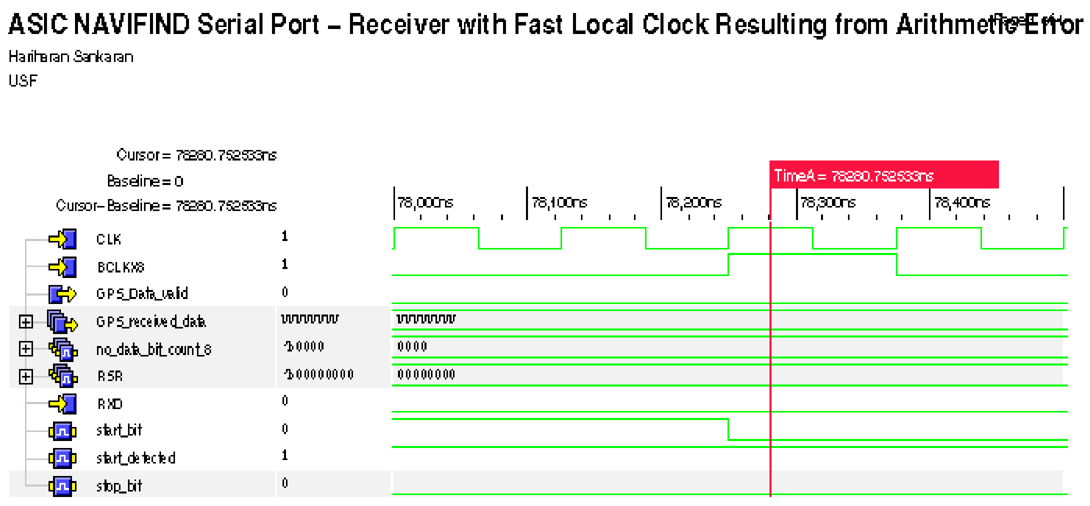

Figure 4.2 Arithmetic Error Affecting the Baud Rate Setting Between ASIC-NAVIFIND Receiver and RFMD GPS Transmitter

The delay introduction resychronizes the transmitter to transmsit at ideal baud rate of 19200 bps. Figure 4.3 shows the clock pulse number 31 of transmitter clock (Bclk). For a clock cycle of 52000 ns (or 19.23 KHz) the local clock of transmitter ($Bclk \uparrow$) should rise at 1612000 ns (31 x 52000) instead with the introduction of delay the transmitter clock rises at 1614500 ns ($30 * 52000 + 2500 + 52000 \Longrightarrow 30 * 52083.33 + 52000$). For every 30 clock cycles the local clock resynchronizes itself to componensate for the arithmetic error and allows transmission of messages at ideal Baud rate (19200 bps). Similarly Figure 4.4 shows ASIC-NAVIFIND serial port receiver local clock (BclkX8) resynchronizing to receive at ideal Baud rate of 19200 bps.

4.3 System Level Energy Optimization Policy

The experiments for the proposed policy were conducted on three random graphs generated by a random graph generator (C code). The random graph generator takes number of buildings (or landmarks) as input to generate the graph. Table 4.1 shows the details of the graph generated by random graph generator. A random path selector takes the generated random graph as input to select the source node, destination node, and a valid random path from source node to destination node.

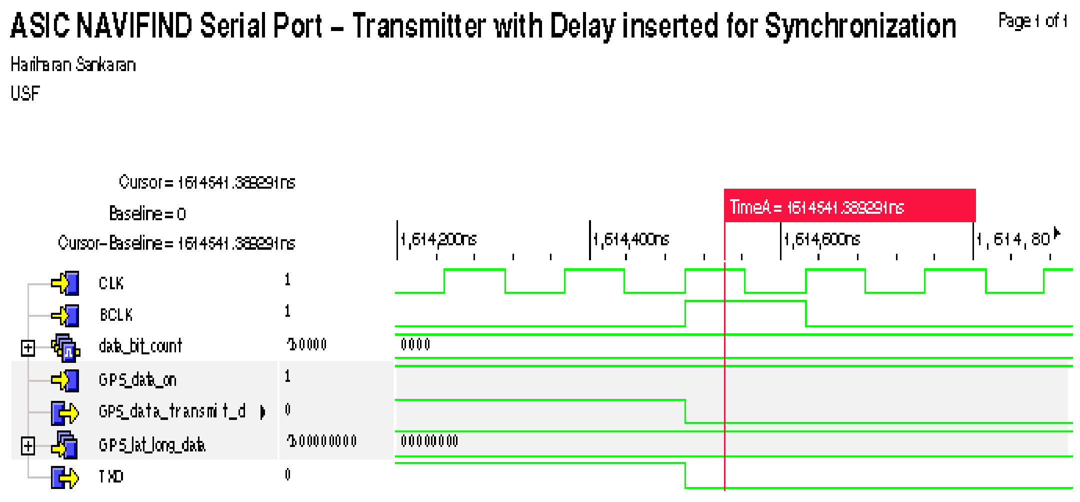
ASIC NAVIFIND Serial Port – Transmitter with Delay inserted for Synchronization
Page 1 of 1
Hariharan Sankaran
USF
Cursor = 1614541.389291ns
Baseline = 0
Cursor-Baseline = 1614541.389291ns
TimeA = 1614541.389291ns
1,614,200ns
1,614,400ns
1,614,600ns
1,614,80
CLK 1
BCLK 1
data_bit_count 'b0000 0000
GPS_data_on 1
GPS_data_transmit_d 0
GPS_lat_long_data 'b00000000 00000000
TXD 0

Figure 4.3 Resynchronization of Local Trasnmitter Clock to Compensate for Baud Rate Variation due to Arithmetic Error

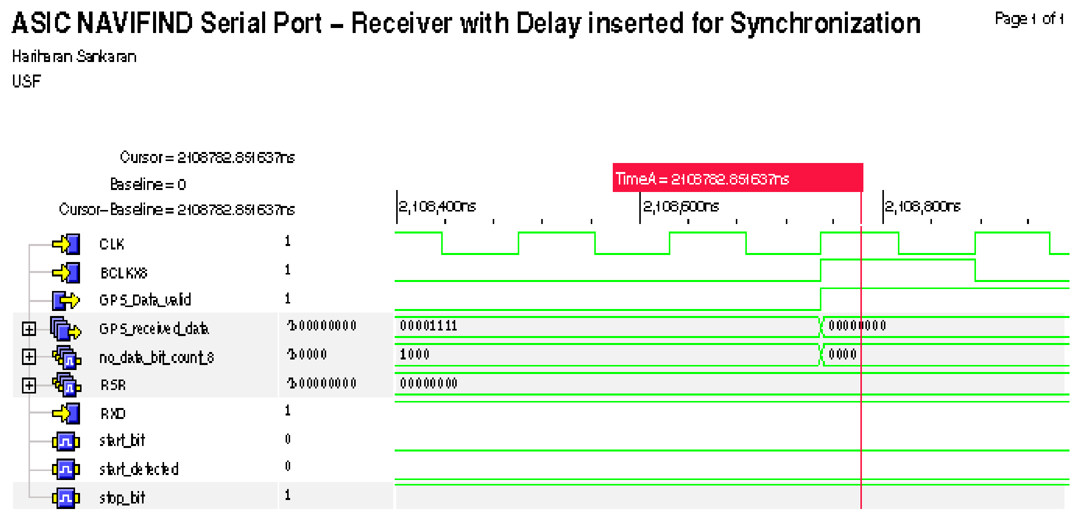
ASIC NAVIFIND Serial Port – Receiver with Delay inserted for Synchronization
Page 1 of 1
Hariharan Sankaran
USF
Cursor = 2108782.851637ns
Baseline = 0
Cursor-Baseline = 2108782.851637ns
TimeA = 2108782.851637ns
2,108,400ns
2,108,600ns
2,108,800ns
CLK 1
BCLKX8 1
GPS_Data_valid 1
GPS_received_data 'b00000000 00001111 00000000
no_data_bit_count_8 'b0000 1000 0000
RSR 'b00000000 00000000
RXD 1
start_bit 0
start_detected 0
stop_bit 1

Figure 4.4 Resynchronization of Local Receiver Clock to Compensate for Baud Rate Variation due to Arithmetic Error

Table 4.1 Random Graph Information

No.	Random graphs	No. of buildings (or landmarks)	No. of intersections generated
1	Graph-1	7	25
2	Graph-2	7	27
3	Graph-3	12	34

For each graph, a random path selector was used to select two different source and destination nodes and a random path, which must be a valid path in the graph. We have conducted two experiments: (1) user speed varied over period of time while user is travelling through the path, and (2) having constant speed throughout the entire path. The proposed *location-aware* system with power manager was impemented at system-level (VHDL and C) and the simulations were carried out using Cadence NCLaunch suite.

Table 4.2 shows the results for the first experiment. We compare the energy consumption for the proposed *location-aware* system with fixed one minute time-out policy and the proposed policy for variable user speed. A key point to note is better energy savings results from better user speed estimates. But on the other hand accurate user speed estimation (or prediction) is not possible. Since different users travel at different speeds, even the same user travels at different speeds during different time periods. Table 4.2 clearly proves the above mentioned point. As we can see in some cases energy savings for average user speed = 1.6 m/s are better and in others average user speed = 1.35 m/s provides better energy savings. But intuitively we could see the slower the user travels more the energy consumption for fixed time-out policy compared to the proposed policy. Another factor which might reduce the energy savings achievable in the proposed policy is the topology of the location and the path travelled by the user. If there are more intersections having more than one path from current update position before any landmark (or building) is found then the proposed policy will set the intersection as the intermediate goal instead of building (or landmark). We also found that in some cases for the same path with same average user speed the proposed algorithm consumes less energy. The reason for having different energy consumption for same average speed in the same path is due to varying user speed during different parts of the route coupled with better speed estimates helps the proposed policy to avoid intermediate goals. For example the user might have travelled a part of the route where there are chances of more intermediate goals at a fast pace or exponential average might have generated a better user speed estimate i.e., closer to the actual user speed. or combination of both might have helped the proposed policy to avoid local minima.

Table 4.3 shows the results for another experiment with user speed as constant. Table 4.3 compares the energy savings for the proposed policy and the fixed one minute time-out policy. On comparison of results obtained for variable user speed and constant user speed we could conclude that results obtained for constant user speed are better on most occasions. This due to the fact with every estimate made by exponential-average the estimated (or predicted) speed inches towards original user speed, while for variable speed the estimated speed may or may not be closer to the actual user speed. But the main drawback is we cannot assume that user is always going to travel at same speed. Table 4.4 and Table 4.5 shows the effectiveness of proposed policy over a system which is kept active continously for variable and fixed speed.

Table 4.2 Comparison of Energy Savings for Proposed Algorithm and One Minute Time-out Policy for Variable User Speed

Graph or map used	Distance between source destination (m)	Average user Speed (m/s)	Proposed Algorithm (mW)	One minute time-out policy (mW)	Energy savings (%)
Graph-2	1042	1.35	570	1755	67.5
		1.60	675	1485	54.5
Graph-2	1127	1.35	570	1890	69.8
		1.60	705	1620	56.5
Graph-1	1252	1.35	660	2160	69.4
		1.60	525	1755	70.1
Graph-3	2098	1.35	330	3510	90.6
		1.60	330	2970	88.9
Graph-3	2179	1.35	465	3645	87.2
		1.60	465	3105	85.0
Graph-1	2788	1.35	660	4590	85.6
		1.60	525	3915	86.6

Table 4.3 Comparison of Energy Savings for Proposed Algorithm and One Minute Time-out Policy for Constant User Speed

Graph or map used	Distance between source destination (m)	Constant user Speed (m/s)	Proposed Algorithm (mW)	One minute time-out policy (mW)	Energy savings (%)
Graph-2	1042	1.1	465	2160	78.5
		1.5	540	1620	66.7
		1.8	540	1350	60.0
Graph-2	1127	1.1	465	2295	79.7
		1.5	675	1755	61.7
		1.8	435	1485	70.7
Graph-1	1252	1.1	465	2565	81.8
		1.5	570	1890	69.8
		1.8	435	1620	66.6
Graph-3	2098	1.1	465	4455	89.5
		1.5	330	3240	89.8
		1.8	195	2565	92.3
Graph-3	2179	1.1	465	4455	89.5
		1.5	465	3105	85.0
		1.8	330	2700	87.8
Graph-1	2788	1.1	660	5670	88.3
		1.5	525	4050	87.0
		1.8	735	3510	79.0

Table 4.4 Comparison of Energy Savings for Proposed Algorithm and One Second Updates With No Power Policy for Variable User Speed

Graph or map used	Distance between source destination (m)	Average user Speed (m/s)	Proposed Algorithm (mW)	one sec. update with no power policy (mW)	Energy savings (%)
Graph-2	1042	1.35	570	308800	99.8
		1.60	675	260400	99.7
Graph-2	1127	1.35	570	334000	99.8
		1.60	705	281600	99.7
Graph-1	1252	1.35	660	370800	99.8
		1.60	525	313200	99.7
Graph-3	2098	1.35	330	621600	99.8
		1.60	330	524400	99.8
Graph-3	2179	1.35	465	645600	99.9
		1.60	465	544800	99.9
Graph-1	2788	1.35	660	826000	99.9
		1.60	525	697200	99.9

Table 4.5 Comparison of Energy Savings for Proposed Algorithm versus System With No Power Policy for Constant User Speed

Graph or map used	Distance between source destination (m)	Constant user Speed (m/s)	Proposed Algorithm (mW)	One sec update with no power policy (mW)	Energy savings (%)
Graph-2	1042	1.1	465	378800	99.8
		1.5	540	278000	99.8
		1.8	540	231600	99.7
Graph-2	1127	1.1	465	410000	99.8
		1.5	675	300400	99.7
		1.8	435	250400	99.8
Graph-1	1252	1.1	465	455200	99.9
		1.5	570	334000	99.8
		1.8	435	313200	99.8
Graph-3	2098	1.1	465	762800	99.9
		1.5	330	559600	99.9
		1.8	195	466400	99.9
Graph-3	2179	1.1	465	792400	99.9
		1.5	465	581200	99.9
		1.8	330	484400	99.9
Graph-1	2788	1.1	660	1014000	99.9
		1.5	525	743600	99.9
		1.8	735	619600	99.8

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

We have proposed a *location-aware* computing system that provides relevant information about the user's current location. The proposed system has GPS receiver as the position tracking component, a removable storage in the form of compact flash card for storing maps and audio files, Stereo codec to provide audio output and in house power manager. We have presented a predictive power policy to switch the system components to low-power state or to shutdown the components when they are no longer needed. We have conducted experiments on random graphs for different variable and constant user speed and achieved energy savings in the range of 55-99%. In an effort to implement the entire system in a single chip we have designed a standard cell library comprising of all basic gates and arithmetic and logic units. The system is implemeted as a system-level model in VHDL and C. The system level model can be synthesised by using an High Level Synthesis tool (HLS) like AUDI (Automatic Design Instantiation) , developed indigenously by Dr. Katkoori and his Research group at USF. To automate the physical design flow we have developed a back end tool in conjunction with the HLS tool.

REFERENCES

[1] M. Weiser, “Hot topics-ubiquitous computing,” *Computer, IEEE*, vol. 26, no. 10, pp. 71–72, October 1993.

[2] G. Chen and D. Kotz, “A Survey of Context-aware Mobile Computing Research,” in *Technical Report TR2000-381*. Department of Computer Science, Dartmouth College, November 2000.

[3] B. Schilit, N. Adams, and R. Want, “Context-aware Computing Applications,” in *Proceedings of IEEE Worshop on Mobile Computing Systems and Applications*, December 1994, pp. 85–90.

[4] A. K. Dey and G. D. Abowd, “Towards a Better Understanding of Context and Context-Awareness,” in *Technical Report GIT-GVU-99-22*. Georgia Institute of technology, College of Computing, June 1999.

[5] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde, “Advanced Interaction in Context,” in *In Proceedings of First International Symposium on Handheld and Ubiquitous Computing*, September 1999, pp. 89–101.

[6] C. A. Patterson, R. R. Muntz, and C. M. Pancake, “Challenges in Location-Aware Computing,” *Pervasive Computing, IEEE*, vol. 2, no. 2, pp. 80–89, April-June 2003.

[7] N. Bulusu, D. Estrin, and J. Heidemann, “Tradeoffs in Location Support Systems: The Case for Quality-Expressive Location Models for Applications,” in *Proceedings of the Ubicomp 2001 Workshop on Location Modeling*, October 2001.

[8] M. R. Stan and W. P. Burleson, “Bus-invert Coding for Low-power I/O ,” *IEEE Transactions on VLSI Systems*, vol. 3, no. 5, pp. 49–58.

[9] R. Hakenes and Y. Manoli, “Improving Microcontroller Power Consumption Through a Segemented Gray Code Program Counter,” in *Proceedings of the IEEE International Conference on Computer Design*, October 1999, pp. 277–278.

[10] V. Tiwari, S. Malik, and P. Ashar, “Guarded evaluation: Pushing power management to logic/synthesis/design,” in *Proceedings of the 1995 International Symposium on Low Power Design*, April 1995, pp. 221–226.

[11] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papafthymiou, “Precomputation-based sequential logic optimization for low power ,” in *Proceedings of the 1994 International workshop on Low Power Design*, 1994, pp. 57–62.

[12] J. Monteiro, S. Devadas, and A. Ghosh, “Retiming sequential circuits for low power,” in *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1993, pp. 398–402.

[13] W. Nabel and L. Kabous, “A System-level Methodology for Low Power Design,” EETimes, May 2003.

[14] L. Benini and G. De Micheli, “System-Level Power Optimization:Techniques and Tools,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 115–192, April 2000.

[15] G. D. Abowd et al., “Cyberguide: A Mobile Context-Aware Tour Guide,” *Wireless Networks, ACM*, vol. 3, no. 5, pp. 421–433, 1997.

[16] K. Cheverest, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, “Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[17] A. Smailagic and R. Martin, “Metronaut: A Wearable Computer with Sensing and Global Communication Capabilities,” in *Proceedings of the International Symposium on Wearable Computing*. IEEE, 1997.

[18] R. Want, A. Hopper, V. Falcao, and J. Gibbons, “The Active Badge Location System ,” *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, January 1992.

[19] B. J. Rhodes, “The Wearable Remembrance Agent: a system for augmented memory ,” in *Proceedings of the International Symposium on Wearable Computing*. IEEE, 1997.

[20] N. Marmasse and C. Schmandt, “Location-aware information delivery with comMotion ,” in *Proceedings of the 2nd Internation Symposium on Handheld and Ubiquitous Computing*, 2000, pp. 157–171.

[21] D. P. Helmbold, D. D. E. Long, T. L. Sconyers, and B. Sherrod, “Adaptive disk spin-down for mobile computers ,” *Mobile networks and Applications*, vol. 5, no. 4, pp. 285–297, December 2000.

[22] F. Douglis, P. Krishnan, and B. Bershad, “Adaptive Disk Spin-down Policies for Mobile Computers,” in *Proceedings 2nd USENIX Symposium on Mobile and Location-independent Computing*, 1995, pp. 381–413.

[23] M. B. Srivastava et al., “Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmbale Computation,” *IEEE Trans VLSI Systems*, vol. 4, no. 1, pp. 42–55, Mar 1996.

[24] Y. Lu and G. De Micheli, “Comparing System-Level Power Management Policies,” *Design and Test of Computers, IEEE*, vol. 18, no. 2, pp. 10–19, March-April 2001.

[25] L. Benini et al., “A Survey of Design Techniques for System-Level Power Management,” *IEEE Transactions on VLSI Systems*, vol. 8, no. 3, pp. 299–316, June 2000.

[26] C. Hwang and A. C. Wu, “A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation,” in *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1997, pp. 28–32.

[27] E. Chung et al., “Dynamic Power Management Adaptive Learning Tree,” in *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1999, pp. 274–279.

[28] L. Benini et al., “System-Levl Dynamic Power Management,” in *Proceedings of the 1999 workshop on Low Power Design*, March 1999, pp. 23–31.

[29] *RFMD GPS Receiver Evaluation Kit Manual*, 2002.

[30] *SanDisk Compact Flash Product Manual*, 2004.

[31] W. Elmenreich and M. Delvai, “Time-Triggered Communication with UARTs,” in *Proceedings of the 4th IEEE International Workshop on Factory Communication Systems*, August 2002, pp. 97–104.

[32] J. L. Peterson and A. Silberschatz, *Operating System Concepts*.