5th May, 2008

Micro-controller for Sensor Networks

Design of a micro-controller geared toward sensor network applications

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology (Hons.)

By

Digvijay Singh

03CS3004

Under the guidance of

Prof. A. Pal

" YOGA KARMASU KAUSALAM "

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

CERTIFICATE

This is to certify that the thesis titled “**Micro-controller for Sensor Networks**” submitted by **Digvijay Singh** (03CS3004) to the Department of Computer Science and Engineering, is a bonafide record of work carried out under my supervision and guidance .The thesis has fulfilled all the requirements as per the regulations of this institute and is valid for submission and evaluation purposes.

Prof. A. Pal

Department of Computer Science and Engineering

Indian Institute of Technology

Kharagpur

TABLE OF CONTENTS

ABSTRACT

In a wireless sensor network (WSN), reducing power consumption is of utmost importance. Battery operated sensor nodes need low-power components designed to prolong their lifetime in the field. This gives rise to a need for WSN specific hardware which is designed to prolong the life-time of the sensor nodes.

We explore this aspect of hardware design by focusing on a node’s micro-controller / processor. In an attempt to develop a WSN specific micro-controller, we survey various commonly used micro-controllers and the characteristics of the typical WSN applications.

We conclude with a design proposal and power simulations to compare our design with the commercial micro-controllers being used in sensor nodes today.

Chapter A

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude, respect and regards for **Prof. A. Pal** under whose guidance, constant encouragement, patience and trust, I have worked on this project. He exposed me to the research topic through proper counsel rigorous discussion and always showed great interest in providing timely support and suitable suggestions. I am thankful to him for the constructive criticism and suggestions for improvements in various stages of this work.

Thanks also to **Prof. A. Gupta** who pointed out (in the last project evaluation) that I focus on a narrower aim for this semester to achieve at least a working design – this was very crucial to success as I have realized now.

I would like to thank all the faculty members and laboratory staff for their cooperation and support.

I also owe regards to my friends **Sujan Kundu** and **Sai Prashanth** for their help throughout the project and the nice time we had working together. I am also thankful to my classmates for their company for five years and all friends who have directly or indirectly assisted me in my endeavors.

Finally, I'm indebted to my parents and my brother for their love, support and inspiration throughout my life.

Digvijay Singh

Chapter B

INTRODUCTION

A sensor network is a term referring, in general, to a collection of networked embedded systems. Each of the systems constituting the network is called a sensor node or just a node [1].

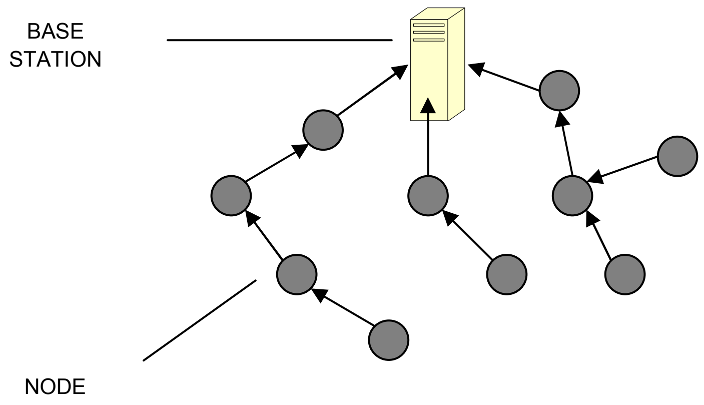


Fig. A SENSOR NETWORK

Each sensor node has the following basic functions:

- **Data Sampling**
    - Gather data from the environment.
- **Data Processing**
    - Process the data using the node's processing capabilities.
- **Data Communication**
    - Relay data to other nodes through the network.

Although computer-based instrumentation has existed for a long time, the density of instrumentation made possible by a shift to mass-produced intelligent sensors and the use of pervasive networking technology gives sensor networks a new kind of scope that can be applied to a wide range of uses. These can be roughly differentiated into:

- **Monitoring space**
    - Environmental monitoring.

    - Precision Agriculture.
    - Alarm and Security Systems.
    - Climate control and surveillance.
- **Monitoring objects**
    - Structural monitoring.
    - Motion detection.
    - Healthcare.
    - Automated Manufacturing.
- **Monitoring the interactions of objects with each other**
    - Wildlife monitoring.
    - Homeland security.

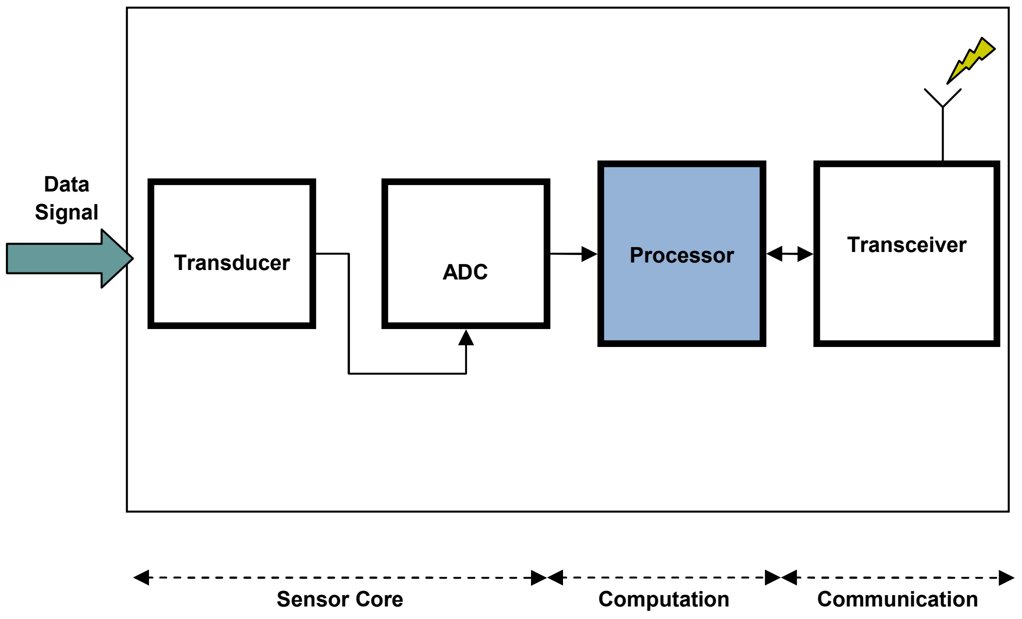


Fig. BASIC ARCHITECTURE OF A SENSOR NODE [1]

Sensor nodes can use many communication media to relay information through the network, but there is an increasing trend towards the medium being wireless because for large networks, laying cables is a daunting task [2]. Thus, wireless sensors networks (WSNs) are becoming increasingly popular over wired sensor networks.

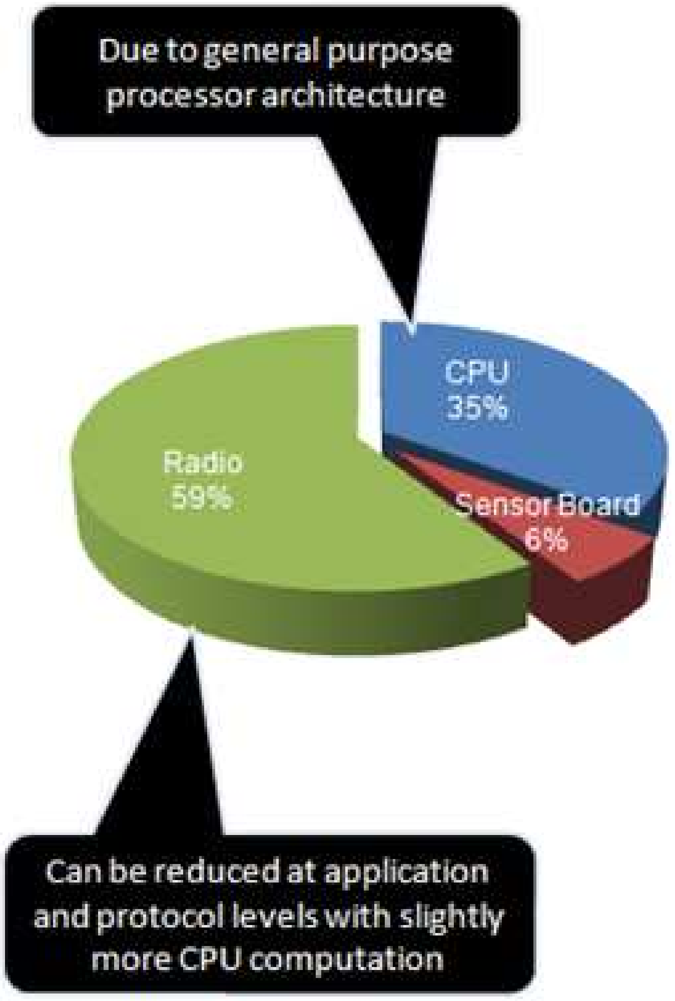


Fig. POWER CONSUMING COMPONENTS OF A SENSOR NODE [2]

In this project we actually look at the processor part of a sensor node. Our basic objective is to propose an architecture that is better suited to sensor network applications than commonly available off-the-shelf micro-controllers [3] that are currently in use.

Chapter C

MOTIVATION

In their survey of processor choices for sensor networks [3], Lynch and O' Reilly show us a handful of commercial processors that can be used today for sensor network applications. The paper shows a wide disparity in the features and instruction sets of these processors. It concludes by saying that the MSP430 is the best suited of the current commercial processors.

The paper clearly shows the paucity of processors specifically targeted towards the sensor network domain, rather all the processors in use by sensor network deployment are general RISC processors that target a large variety of embedded systems. This in-turn causes their designs to be laced with too many generalities as can be seen from the wide variety and disparity of different processor's features explored in [2] and [3]. These generalities in-effect make the design inefficient for specific sensor network applications. Generally, these processors have hardware which is not required by the sensor node at all and just wastes power.

We instead try to take a different approach to processor design and attempt to come up with a micro-controller tailored to the common application needs of the sensor network domain, while stripping off useless features to increase efficiency.

There have been other forays into this area like the SNAP/LE [4]. The only problem with these designs is that they are asynchronous and remain theoretical with no hardware simulation or actual fabrication data is available to support their claims. These designs are promising but still in the future.

We instead will follow a conventional synchronous design, not with the view of proposing a new design idea or paradigm but with the view of making conventional designs more tailored to sensor network applications. This approach will also give us the opportunity to fabricate and conduct hardware simulation of our micro-controller unit (MCU).

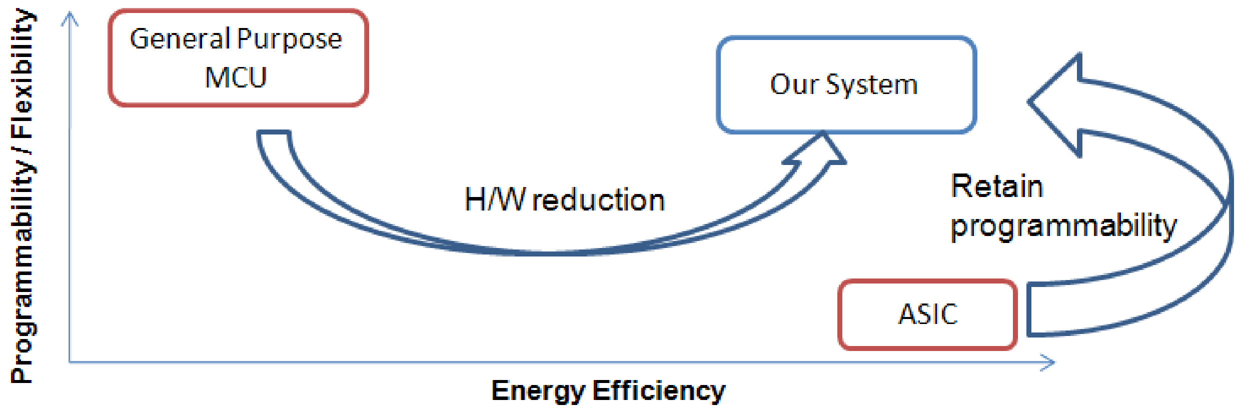


Fig. ADVANTAGE OF APPLICATION SPECIFIC MCUs

To recapitulate, our motivations for this processor design are:

- Application specific hardware leads to reduced power due to discard of excess features.
- Sensor network applications have been studied in detail and their characteristics are available.
- Dearth of sensor network specific processors – most sensor nodes use off-the-shelf components like the ATMega128L [6].
- The remaining designs are theoretical and fabrication is way in the future due to use of unconventional design techniques. This causes a lack of actual fabrication or hardware simulation of the design.

Chapter D

DESIGN BACKGROUND

This section explores the various requirements and issues of general sensor node processors/ controllers. Studies of the basic features required by common sensor network applications are explored to give us a background for our design.

Sensor network processors introduce an unprecedented level of compact and portable computing. These small processing systems reside in the environment which they monitor, combining sensing, computation, storage, communication, and power supplies into small form factors. Sensor processors have a wide variety of applications in medical monitoring, environmental sensing, industrial inspection, and military surveillance. Despite efforts to design suitable processors for these systems [4], there is no well-defined method to evaluate their performance and energy consumption. The historically used MIPS (millions of instructions per second) and EPI (energy per instruction) metrics cannot provide an accurate comparison because of their dependence on the nature of instructions, which differ across instruction set architectures.

To properly evaluate architectures and get an idea of the kind of benchmark algorithms we must gear our design towards, we use the ideas on WiSeNBench [8] and SenseBench [9]. These benchmarks help us come with a basic level of requirements to help in design of our processor. This, along with a survey of commonly used features of current sensor network micro-controllers, like the MSP430 [7] and ATMega128 [6], in sensor network applications forms the basis for our design features.

We use conventional low-power micro-controller design techniques suited to the features which best fit the sensor network applications. This will help us come up with efficient, easily usable micro-controller that can be manufactured using the current fabrication technology.

D.1. DESIGN NEEDS

The main findings from the test-benches, benchmarks and literature are summarized as follows:

- Power is the key objective in sensor network processors i.e. low-power modes are useful.
- RISC-based processors work best as complicated instructions like multiplication and DSP abilities are rarely required by sensor nodes applications which are meant to be lightweight [8].
- Sensor nodes are typically interrupt-driven systems and so need a good interrupt structure.

D.2. TYPICAL MICRO-CONTROLLERS

This section explores some of the most commonly used micro-controllers in sensor networks. Their relevant features are mentioned in the following sub-sections.

D.2.1. PIC Series

The PIC series of micro-controllers [5] is manufactured by Microchip. They are widely used in embedded systems today. The features of the micro-controller are as follows.

- **Context Switching Features**:
  - The PIC has no special features that help context switching but instead uses the common approach of saving the context i.e. registers, flags etc. each time the context is switched.
- **Interrupt Structure**:
  - Interrupt structure is simple but powerful.
  - Priority levels can be set for the interrupts (HIGH or LOW).
  - The robust interrupt structure is suited to real-time interrupt driven applications.

- **ISA**:
  - PIC instructions vary in number from about 35 instructions for the low-end PICs to about 70 instructions for the high-end PICs.
  - PIC instruction size varies from 12 bits in the PIC12 series to 30 bits in the PIC30 series. All operands are 8 bit and so the PIC is called an 8-bit micro-controller.
  - Most instructions are single cycle execution (4 clock cycles), with single delay cycles upon branches and skips.
  - ISA is RISC-like but not exactly RISC as not all instructions are of fixed length and load-store architecture isn't followed.
- **Memory Architecture**:
  - The PIC micro-controllers follow the Harvard architecture i.e. separate code and data space.
  - PICs have a set of register files that function as general purpose RAM, special purpose control registers for on-chip hardware resources are also mapped into the data space.
  - The data memory is divided into banks. The current bank is accessed in one instruction; otherwise the bank must be switched.
  - The addressability of memory varies depending on device series, and all PIC devices have some banking mechanism to extend the addressing to additional memory. Later series of devices feature move instructions which can cover the whole addressable space, independent of the selected bank.
- **I/O Features**:
  - The PIC series of devices have I/O ports which can be used for transceiver and transducer/ADC interfacing.

    - Some of the PIC devices even have on-chip ADCs that can be used to directly sample analog data from the transducers.
    - The PIC series also have a UART though these aren't required in the basic sensor node architecture, they can be used to interface to a PC's serial port on the nodes which behave as wired "base stations".
    - The PIC18 also has hardware which can carry out I2C or SPI.

- **Applicability of Low-power Techniques**:
    - The PIC micro-controllers can operate over a wide frequency and voltage range which makes frequency scaling possible.
    - The PIC12 and PIC16 series don't explicitly support frequency scaling but external hardware can be used for this purpose.
    - The PIC18 series and above explicitly allows for frequency scaling by providing an internal RC oscillator that can be used as the clock. The RC oscillator's frequency can be scaled.
    - The PIC18 series also allows switching between external clock and the internal oscillator. Clock switching takes time and results in a delay of two old clock cycles and three new clock cycles.
- **Low-power sleep modes**:
    - The sleep modes in the PIC series are extremely simplistic compared to some of the other processors (like TI's MSP430). As a matter of fact, the lower end PICs don't even provide sleep modes.
    - The PIC16 series has only one sleep mode wherein the processor core and all other peripherals except the asynchronous timer is shut-off.
    - The PIC18 series has two sleep modes, one in which only the core is shut-off and all clocks to peripherals keep running and the other in which all peripherals and the core is shut-off.

- **Pipelining**:
    - PIC instructions generally take one machine/instruction cycle to execute i.e. 4 clock cycles.
    - The execution of an instruction takes place in two phases: 1) Fetch and 2) Execute. This allows the PIC architecture to be pipelined.
    - The pipeline is a two stage pipeline.
- **Shortcomings**:
    - PIC microcontrollers have a very small set of instructions, leading some to consider them RISC devices. However, the PIC architecture does not reflect many of the advantages of RISC design. For example:
        - PIC does not have a load-store architecture, as memory is directly referenced in arithmetic and logic operations
        - it has a single working register, while RISC designs typically include 16 or more general purpose registers
        - its addressing modes are not orthogonal, since some instructions can address RAM or immediate data, while others can only use the working register
        - bank switching is required to access the entire RAM of many PIC devices, making the development of libraries of position-independent code complex and inefficient
        - a stack cannot be implemented efficiently, so it is difficult to generate reentrant code

D.2.2. AVR Series

The AVR series of micro-controllers [6] follows a Harvard architecture single with an 8-bit RISC core running single cycle instructions. Particularly, the ATMEGA128L is a widely used microcontroller in sensor nodes and is featured by several motes, including Berkeley Motes and Mica2/MicaZ [11].

- **Broad Device Classification**:
    - tinyAVRs (e.g. all the ATTiny series)
        - 1-8 KB program memory
        - 8-20 pin package
        - Limited peripheral set
    - megaAVRs (e.g. all the ATmega series, including ATmega128L)
        - 4-256 KB program memory
        - 28-100 pin package
        - Extended instruction set (instructions for multiply and handling larger program memory)
        - Extensive peripheral set
    - Application Specific AVRs
        - AVRs with special features like LCD/USB controller, etc.
- **Interrupt Structure**:
    - Powerful interrupt structure.
    - The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles, the program vector address for the actual interrupt handling routine -is executed.
- **ISA**:
    - The AVR ISA is more compact than most other 8-bit microcontrollers. The ATmega128 offers 133 powerful instructions
    - Each instruction takes one or two 16-bit words
    - Arithmetic operations work on registers R0-R31, but not directly on the RAM and take one clock cycle, excepting multiplication and word-wide addition which take two cycles

  - RAM and I/O space can be accessed only by copying to or from registers. Indirect access (including optional post-increment, pre-decrement or constant displacement) is possible through registers X, Y, and Z (pointer registers).
  - All accesses to RAM takes two clock cycles. Moving between registers and I/O is one cycle. Moving eight-bit or sixteen-between registers or constant to register is also one cycle. Reading program memory (LPM) takes three cycles.
  - A few AVR microcontrollers lack certain non-basic instructions like multiplication, extended loads/jumps/calls, long jumps and power control.
- **Memory Architecture**:
  - The AVR is Harvard architecture based with programs and data stored separately for performance and parallelism.
  - Flash, EEPROM and SRAM are all integrated on single chip, removing need for external memory
  - The non-volatile Self-Programmable instruction flash memory (up to 256K) is used predominantly for storing program.
  - The data address space consists of the register file, I/O registers and the SRAM (up to 8K). The AVRs have 32 single-byte registers
  - The AVR has memory-mapped I/O registers. The working registers occupy the first 32 memory addresses (0000_{16} – $001F_{16}$) followed by 64 I/O registers (0020_{16} – $005F_{16}$). Actual SRAM for data storage starts after the register section.
- **I/O Features**:
  - Bi-directional General Purpose I/O ports.
  - The AVRs have a built-in ADC and Analog Comparators
  - On chip debugging (OCD) support through JTAG or debugWIRE on most devices. JTAG signals are multiplexed on GPIOs
  - Serial Peripheral Interface and a Two-Wire Serial Interface for flexible communication
  - Analog Comparators
  - 10-Bit A/D Converters, with multiplex of up to 16 channels

- **Applicability of Low Power Techniques**:
    - Low-voltage Devices Operating Down to 1.8V i.e. variable operating voltage.
    - Software Selectable Clock Frequency i.e. frequency scaling possible.
- **Low Power Sleep Modes**
    - Six Power-Saving Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby. User can tailor power consumption to application's requirement
- **Pipelining**:
    - Each instruction takes one or two cycles and consists of the Fetch and the Execute cycles.
    - Amtel's AVRs have a single level pipeline design, i.e. next instruction is fetched as current one is executing
- **A Comparison between AVR and PIC**
    - AVRs have non-banked access to data memory, whereas PICs require setting bank registers to access beyond 256 bytes of memory. Also, some AVRs support hooking up external SRAM in a way that allows the MCU to use it natively (rather than going through a series of port accesses).
    - AVRs have 32 general purpose registers, the PIC only has one.
    - In AVRs that have SRAM (most of them), the stack is contained within SRAM instead of being limited to a built-in hardware stack. Conversely, with PIC, this is one less thing to worry about.
    - The ATmega and PIC18F have hardware multipliers, ATTiny and PIC16F do not.
    - The AVRs support a more generalized interrupt system, as opposed to the PIC high/low priority interrupt vectors.
    - Although PIC's clock speeds appear higher, the clock speed is divided by four to give the actual instruction rate.

D.2.3. MSP430 Series

The MSP430 micro-controller [7] by Texas Instruments is one of lowest power consuming processors on the market currently. It is ideal for wireless applications and embedded systems. We illustrate the features and design of this micro-controller in the following sections.

- **Context Switching Features**:
    - Dedicated stack and stack instructions are present for saving context and fast context switching in case of interrupts and other context changes.
    - A variety of addressing modes are available even for the stack operations.
- **Interrupt Structure**:
    - The MSP430 provides two timers and a watchdog timer.
    - Powerful interrupt structure with ability to mask interrupts.
    - Lack of ability to set interrupt priorities but the interrupts have a fixed priority decided by ordering (daisy chaining).
- **ISA**:
    - The MSP430 is a 16 bit RISC processor and has most of the RISC features in its ISA.
    - There are 27 core instructions and 7 addressing modes available. Instructions are 16 bits, followed by up to two 16-bit extension words.
    - There are three core instruction formats: 1 operand, 2 operands or a jump.
    - Instructions generally take 1 cycle per word fetched or stored, so instruction times range from 1 cycle for a simple register-register instruction to 6 cycles for an instruction with both source and destination indexed.
    - Rich ISA with Boolean and arithmetic instructions; dedicated multiplication hardware.

    - Dedicated stack and stack instructions are available.

- **Memory Architecture**:
    - The micro-controller uses the Von Neumann architecture i.e. same space for code and data.
    - A single 16-bit pointer is used to address the whole ROM and RAM space. Memory is byte-addressed, and pairs of bytes are combined little-endian to make 16-bit words.
    - The processor contains 16 16-bit registers. R0 is the program counter, R1 is the stack pointer, R2 is the status register, and R3 is a special register called the *constant generator*, providing access to 6 commonly used constant values without requiring an additional operand. R4 through R15 are available for general use.
- **I/O Features**:
    - The MSP430 has a host of I/O features built in it.
    - First and foremost it has 10 I/O ports for interfacing devices like radio and transducers.
    - Some models also come with in-built ADC which is useful in sampling analog transducer data.
    - Full UART and SPI support is available on all the models.
- **Applicability of Low Power Techniques**:
    - The MSP430 is ideal for low-power applications and as such is built with this in mind.
    - It has the ability to sample input voltage using its "supply voltage supervisor" and set a flag each time it falls below a software programmable threshold. This can be used for battery-aware scheduling.

  - Other features are the variable operating voltage and scalable clock frequency.
  - The clock system is designed specifically for battery-powered applications. A low-frequency auxiliary clock (ACLK) is driven directly from a common 32-kHz watch crystal. The ACLK can be used for a background real-time clock self wake-up function. An integrated high-speed digitally controlled oscillator (DCO) can source the master clock (MCLK) used by the CPU and high-speed peripherals. By design, the DCO is active and stable in less than 6 µs. MSP430-based solutions effectively use the high-performance 16-bit RISC CPU in very short bursts.
    - Low-frequency auxiliary clock = Ultralow-power stand-by mode
    - High-speed master clock = High performance processing
- **Low Power Sleep Modes**:
  - The MSP430 has six different power modes, ranging from fully active, to not clocking the core, to keeping the digital oscillator running to generate the clock but disabling the loop control to save power to fully powered down (with peripherals separately enabled or disabled).
  - Due to the use of the digital oscillator, wakeup time can be as low as 6µs.
- **Pipelining**:
  - The MSP340 is not a pipelined architecture.
  - Due to variable number of cycles in the instructions (1 to 6) there is no pipelining, but most instructions run in just one cycle so this doesn't slow the processor down much.
  - But because of the lack of pipelining and division of the instruction cycle into smaller cycles, the maximum clock frequency (8Mhz) is slower than other similar processors but gets the job done as the instructions are just one cycle.

Chapter E

CHARACTERISTICS OF WSN APPLICATIONS

Wireless sensor network (WSN) applications have certain common characteristics as is explored in this chapter. We look at all the application characteristics from a processor point-of-view as they'll impact our initial design proposal.

From a node's processor's perspective the applications can be divided into:

- Data Sampling
- Data Processing
- Data Communication

Each of these applications together forms the full functionality in which a node's processor must be involved. The diagram below summarizes the characteristics of a typical sensor node:

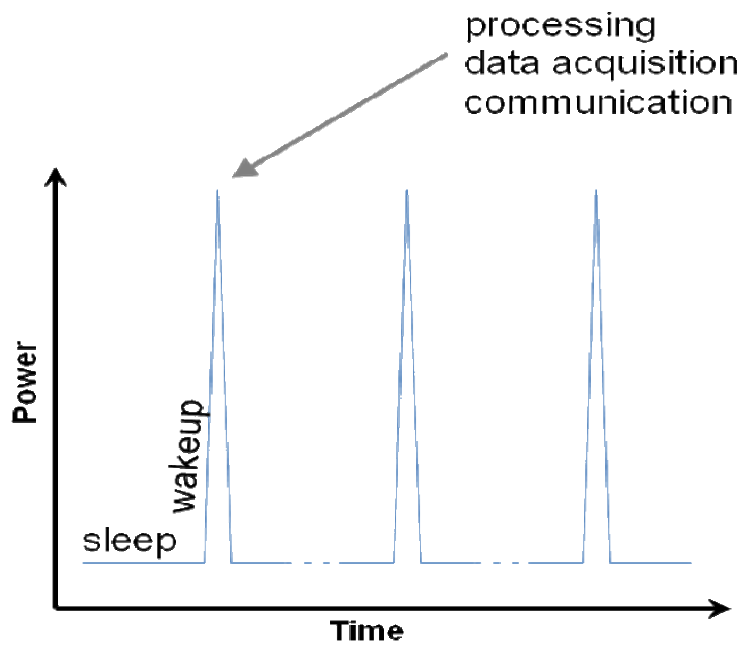


Fig. TYPICAL WSN NODE'S APPLICATION

E.1. DATA SAMPLING

The following table summarizes the list of typical periodic data sources in WSN applications [8]:

Phenomenon	Sampling (Hz)	Resolution (bits)
Atmospheric temperature	0.017 - 1	8
Barometric pressure	0.017 - 1	8
Body temperature	0.1 - 1	8
Seismic vibration	0.2 - 100	8 - 12
Blood pressure	50 - 100	8 - 10
Engine temp / pressure	100 - 150	8 - 10
ECG (electro-cardiogram)	100 - 250	8 - 16

For non-periodic events, like detection and notification, the events are interrupt-triggered. They occur infrequently [1] but must be reported quickly and reliably.

Thus, from the above data we can conclude that for data sampling we need:

- Low-duty cycle (<1%) i.e. sleep capabilities needed.
- Powerful interrupt structure to easily handle interrupt-driven events.
- Long life-time necessary as the duty cycle is low and so infrequent events may be too few and too far in between.

E.2. DATA PROCESSING AND COMMUNICATION

We've compiled a list of the typically used data processing and communication applications/algorithms in most WSN nodes from tinyOS [12], the most common OS for sensor nodes, and two test-benches, SenseBench [9] and WiSeNBench [8], which are the de-facto standard for evaluating WSN processors.

- Crypto / Security:
    - CRC8
    - CRC16
    - Hash Algorithms
    - SPINS
    - RC5
    - TEA – Tiny encryption algorithm
- Basic / Core:
    - Sorting
    - Fibonacci
    - Binary Search
    - Min-max finder
    - Sum-array
    - Majority
    - Top10
- Compression:
    - RLE (run-length encoding)
    - Wavelet encoding
    - Compression with nibble differences
- Routing:
    - Ad-hoc routing.
    - Multi-hop routing.

    - Level-hop routing.
    - EnergyEff routing.
    - SMAC (medium access control)
- Radio:
    - Manchester Encoding.
    - NRZI Encoding.

E.2.1. Results From Literature

Results for the above applications/algorithms for common architectures ATMega128, PIC16 and MSP430 were compiled from literature [3, 5-9]:

- Code size
    - Most algorithms have code sizes under 500 bytes for 16-bit architectures.
    - For the 8-bit RISC architectures, size was under 700 bytes.
- Memory access
    - All algorithms have 40 to 60% instructions as memory accesses for both the 8-bit and 16-bit architectures.
- Frequent instructions
    - Load, Store, Add, Sub, R-shift, L-Shift, Mov, Xor. All other instructions occur less than 1% of the time.

The above requirement guidelines along with the literature survey of common MSP430 [7], ATMega128 [6] and PIC [5] series of micro-controllers help us propose our initial architecture, instruction set and basic I/O features.

Chapter F

THE INITIAL ARCHITECTURE

Using the requirement guidelines we give a summary of the initial architecture and instruction-set in this section. This is just the initial architecture and will be tweaked for improvement once initial simulation results are presented.

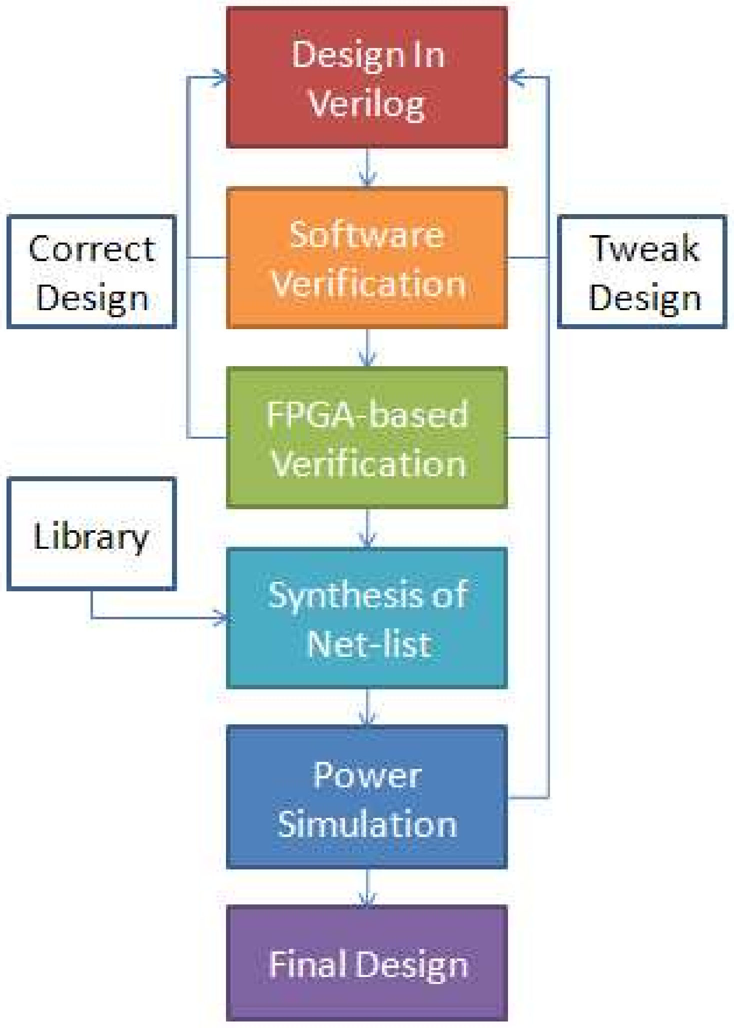
F.1. FEATURES

Architectural Features

- **8-bit synchronous accumulator-based processor**
  - Harvard memory architecture
  - Data Memory of 256 Bytes and Program Memory of 4 KB.
    - Initially taken as register arrays – to be fixed through simulation results.
  - RISC-based Load-Store ISA
    - 1 or 2 byte instructions with 1 or 2 cycle execution.
  - 16 X 8-bit GPRs
  - Stack of 8 levels
    - Fast context-switching (jumps, calls, interrupts etc.)
- **Pipelining**
  - Two-stage pipeline with Fetch-Execute.
- **I/O Features**
  - 1 GPIO (8 bit) and UART (for PC interfacing).
- **Interrupt Structure**
  - Two general purpose interrupts with priority levels and mask-able.
- **Timers**
  - Watchdog timer and 16-bit timer.
- **Low-power Modes:** Idle and Sleep.

F.2. DESIGN FLOW

We follow the design flow given below to arrive at the needed simulation results for tweaking our initial architecture to make it efficient to WSN applications compared to current processors.



The above design flow has the following details:

- It is iterative and so allows progressive improvement of the Verilog [15] design.
- The FPGA verification is done using the Altera Cyclone II FPGA kit [13].
- The national semiconductor 180nm CMOS library is used for synthesis.
- Synopsys Design Vision [14] is used as the synthesis and simulation tool for power.

F.3. VERIFICATION

After coding the design in Verilog HDL [15], functional verification of the design is done using two basic tools:

1. Verilog Simulator [17]:
    a. Does functional simulation in software.
    b. Timing waveforms may be generated for proper synchronous design.
2. Altera Cyclone II FPGA [13]:
    a. Does functional simulation is hardware.
    b. Gives a more realistic view of design problems like clock skew.
    c. Can be interfaced to real hardware like PCs and ADCs.

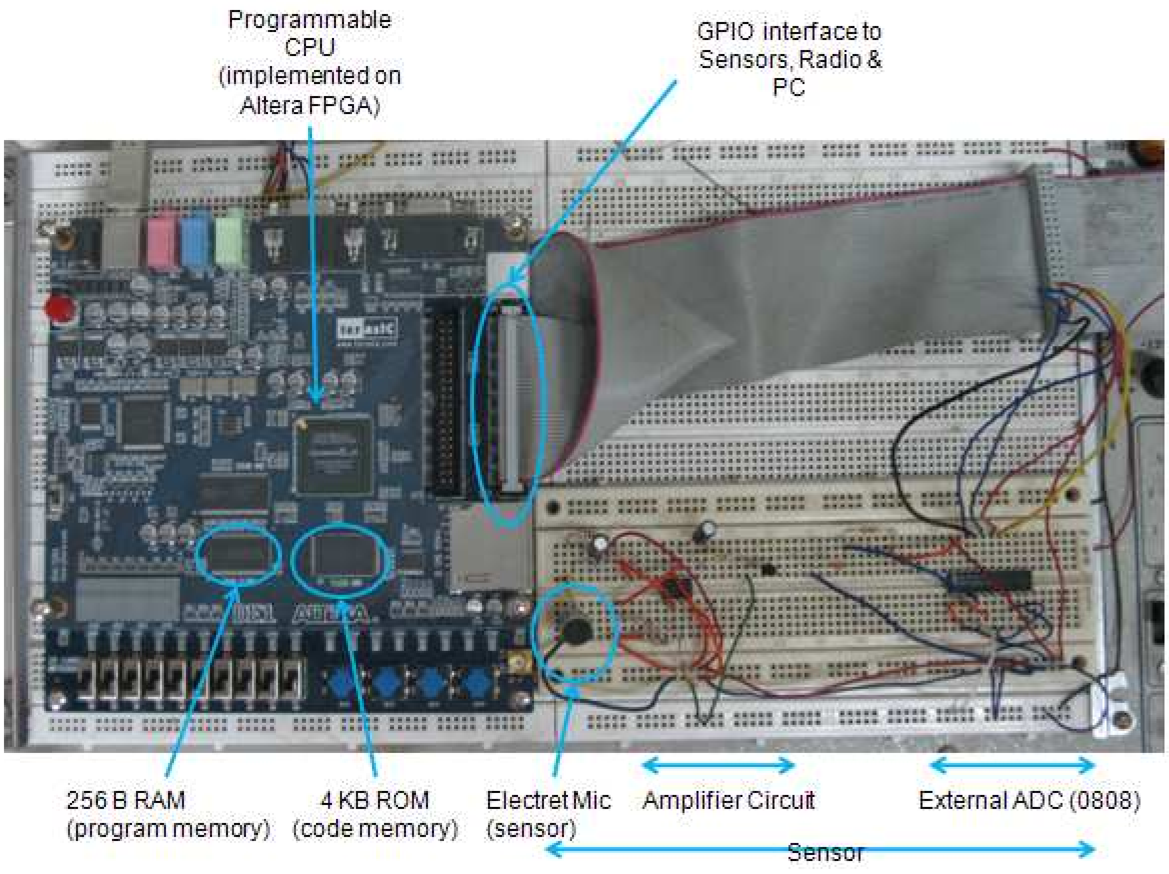

Programmable CPU (implemented on Altera FPGA)
GPIO interface to Sensors, Radio & PC
256 B RAM (program memory)
4 KB ROM (code memory)
Electret Mic (sensor)
Amplifier Circuit
External ADC (0808)
Sensor

Fig. ALTERA CYCLONE II FPGA SETUP

We tested the following basic programs in the ROM to verify the features' functionality:

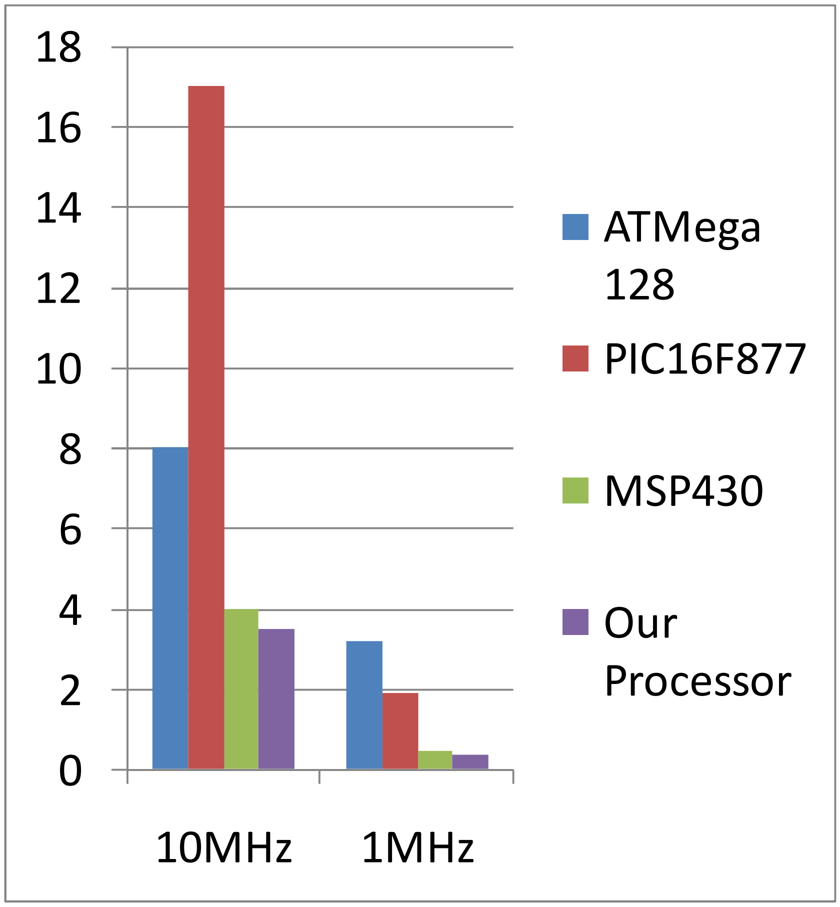
i. Fibonacci number generation – tests ALU, conditional jumps, registers and RAM's operations.
ii. Interrupt-based CRC8 of input on GPIO – checks ALU, interrupt and GPIO.
iii. Recursive Quick-sorting – tests RAM's operations, stack due to recursion and calls/jumps.
iv. Periodic sleep-wakeup cycle and display count of cycles – tests GPIO, sleep modes and power-up timer.
v. Interface to PC using UART – tests use of UART.
vi. Periodic Sampling of ADC data – tests timer and GPIO.

After verifying all the functionality and correcting any errors In the design, we moved on to synthesis and simulation of the design using Synopsys Design Vision [14] and the National Semiconductor 180nm library.

F.4. SIMULATION RESULTS

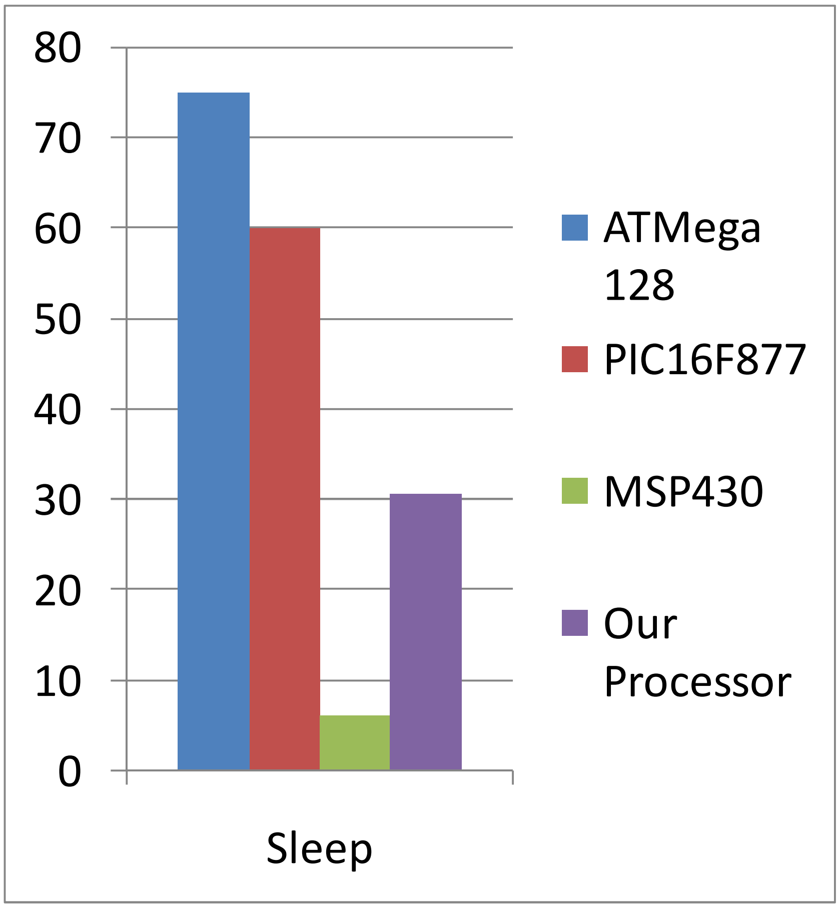
This section summarizes the power simulation results obtained for the initial architecture compared to the other commonly used MCUs.

1. **Active Power (mW)**



We are performing better than all other MCUs as far as dynamic power is concerned, so our initial design features were a good start. As can be seen we have the lowest active power consumption in the 10Mhz (3.49mW) and the 1MHz (0.35mW). The only MCU even coming close is the MSP430 with 4mW at 10Mhz and 0.42mw at 1MHz.

2. Sleep Power (µW)



It can be clearly seen that we are lagging to the MSP430 in this area. So what went wrong?

Let's look at the CMOS power dissipation sources [16]:

- **Dynamic and Short-circuit**:
  - Mostly comes into play during active mode of operation i.e. during switching or clock operation.
- **Leakage**:
  - Is a static source of power dissipation and becomes dominant in deep sub-micron technologies especially during sleep modes where clock frequency is reduced and so dynamic power has little bearing.

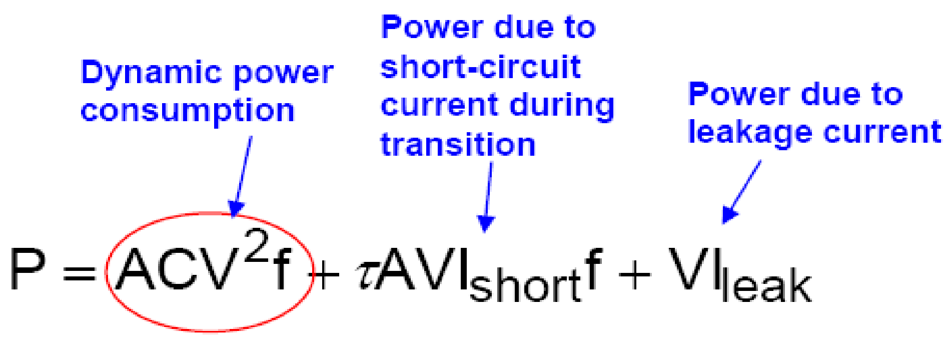

Dynamic power consumption
Power due to short-circuit current during transition
Power due to leakage current
$P = ACV^2f + \tau AVI_{short}f + VI_{leak}$

Fig. POWER DISSIPATION COMPONENTS IN CMOS

After some survey we found that it was RAM and ROM that were creating too much leakage power (>25µW) which was causing us to use a lot of power in our sleep mode, even though dynamic power usage was negligible. This was because in our initial design we had used a simplistic memory model for RAM and ROM by modeling them as register files in Verliog. This was leading to too much leakage current.

So, based on these simulation results, we tweaked our design to use 256B SRAM for our RAM and 4KB Flash for our ROM from the 180nm library we were using for our synthesis. The results of these tweaks and their simulation values are shown in the next chapter.

Chapter G

THE FINAL ARCHITECTURE

Using the initial architecture's simulation results we propose our final tweaked architecture.

G.1. FEATURES

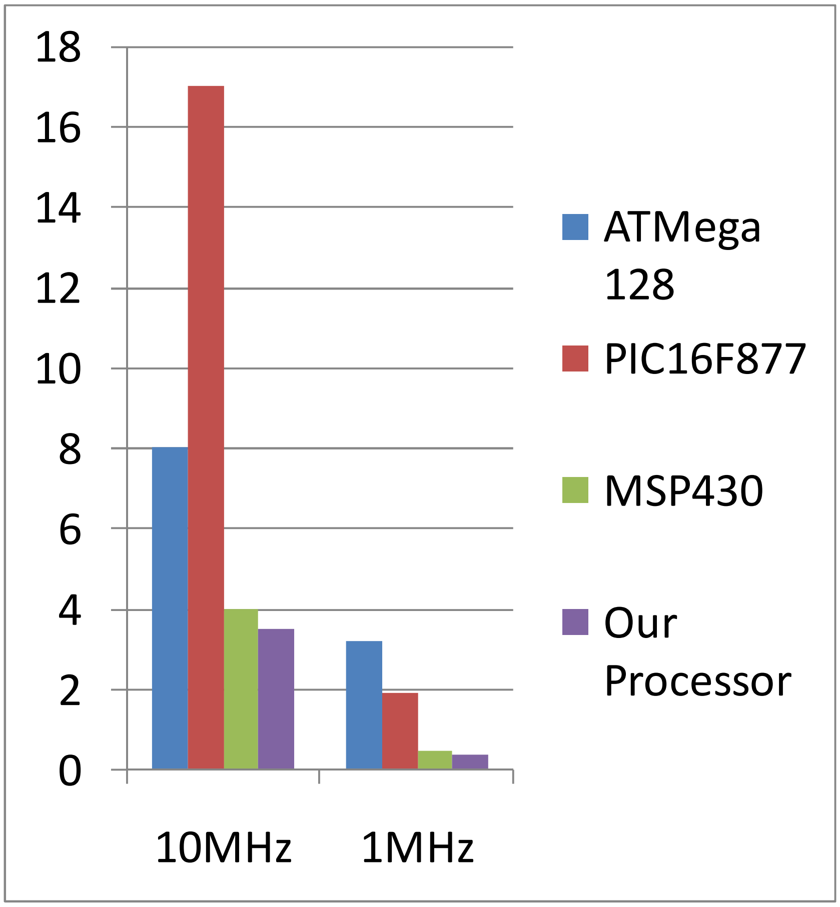
Architectural Features

- **8-bit synchronous accumulator-based processor**
  - Harvard memory architecture
  - Data Memory of 256 Bytes SRAM.
  - Program Memory of 4 KB FLASH.
  - RISC-based Load-Store ISA
    - 1 or 2 byte instructions with 1 or 2 cycle execution.
  - 16 X 8-bit GPRs
  - Stack of 8 levels
    - Fast context-switching (jumps, calls, interrupts etc.)
- **Pipelining**
  - Two-stage pipeline with Fetch-Execute.
- **I/O Features**
  - 1 GPIO (8 bit) and UART (for PC interfacing).
- **Interrupt Structure**
  - Two general purpose interrupts with priority levels and mask-able.
- **Timers**
  - Watchdog timer, 16-bit timer and power-up timer.
- **Low-power Modes**
  - Idle: reduce clock frequency to reduce dynamic power [8, 16] – faster wakeup.
  - Sleep: shut-off clock to all components except interrupt controller through clock gating [8] – slower wakeup.

G.2. SIMULATION RESULTS

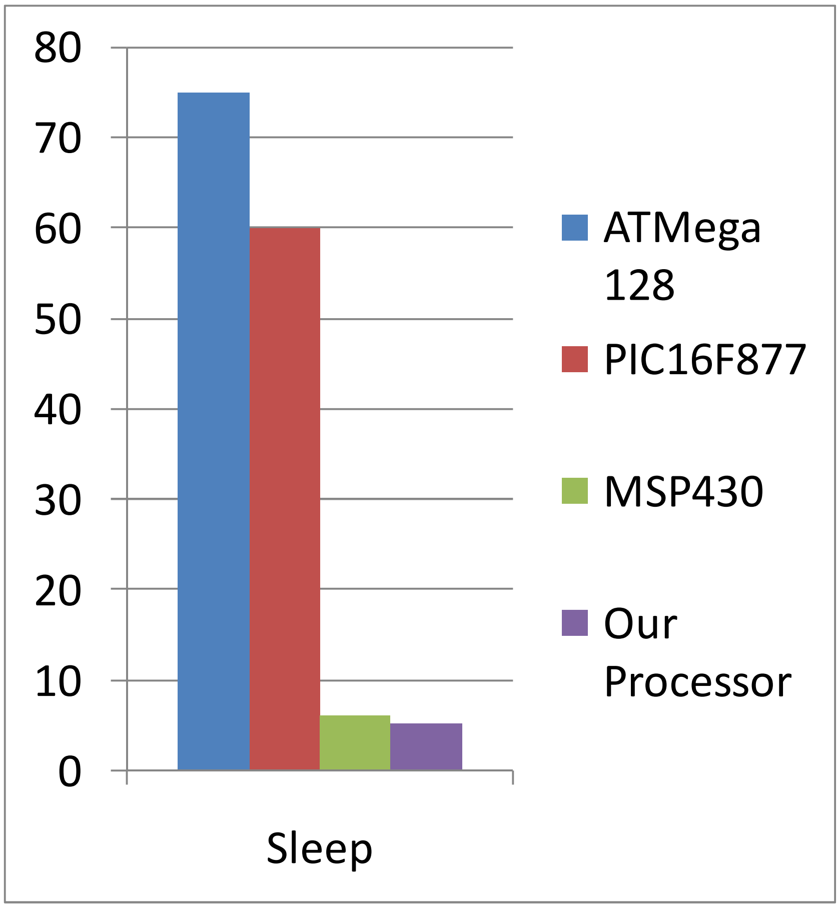
This section summarizes the power simulation results obtained for the final architecture after the whole design flow in the previous chapter was followed.

1. **Active Power (mW)**



We are performing better than all other MCUs as far as dynamic power is concerned, so our initial design features were a good start. As can be seen we have the lowest active power consumption in the 10MHz (3.49mW) and the 1MHz (0.35mW). The only MCU even coming close is the MSP430 with 4mW at 10MHz and 0.42mw at 1MHz.

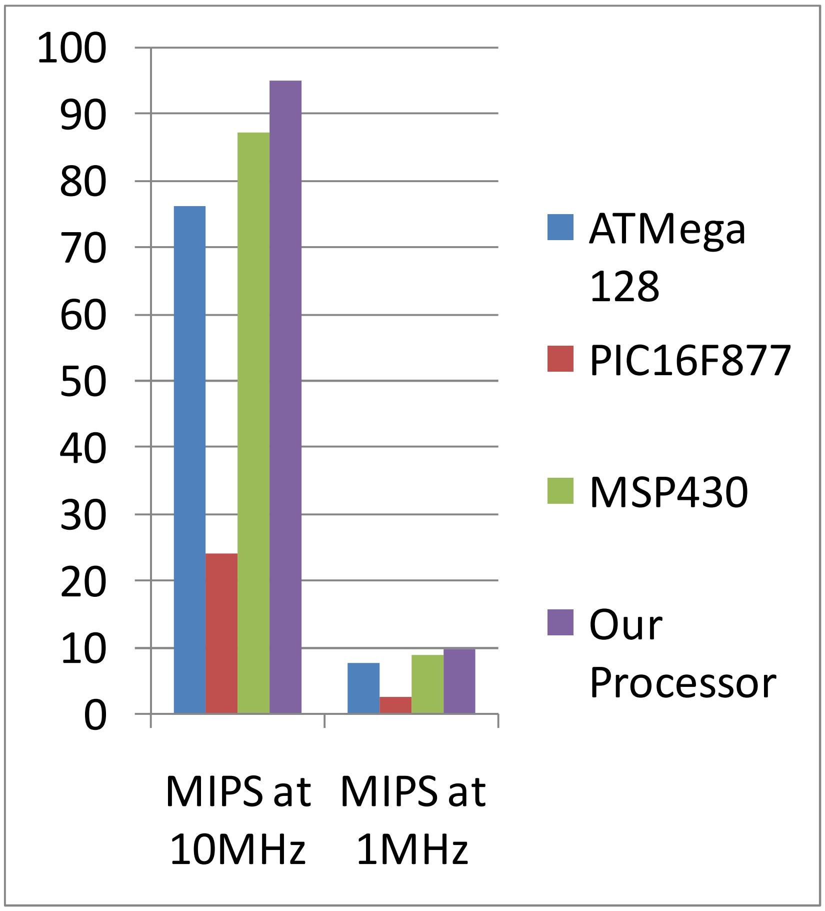
2. **Sleep Power (µW)**



It can be clearly seen that we have beaten all other MCUs in sleep power also. The replacing of the register files with SRAM and Flash have considerably reduced leakage power (~5µW). This gives us an edge compared to all other MCUs with only MSP430 coming close (~6µW). Sleep power is paramount in node design as most nodes have very low duty cycle (<1%) and are asleep most of the time [1]. Thus, to sum up our final architecture:

- Beats all the MCUs in the market in terms of power consumption.
- Doesn't cause any major loss in functionality i.e. retains all necessary features.
- Does retain programmability to help it run a multitude of WSN applications.
- Uses only conventional synchronous design techniques making fabrication easy.

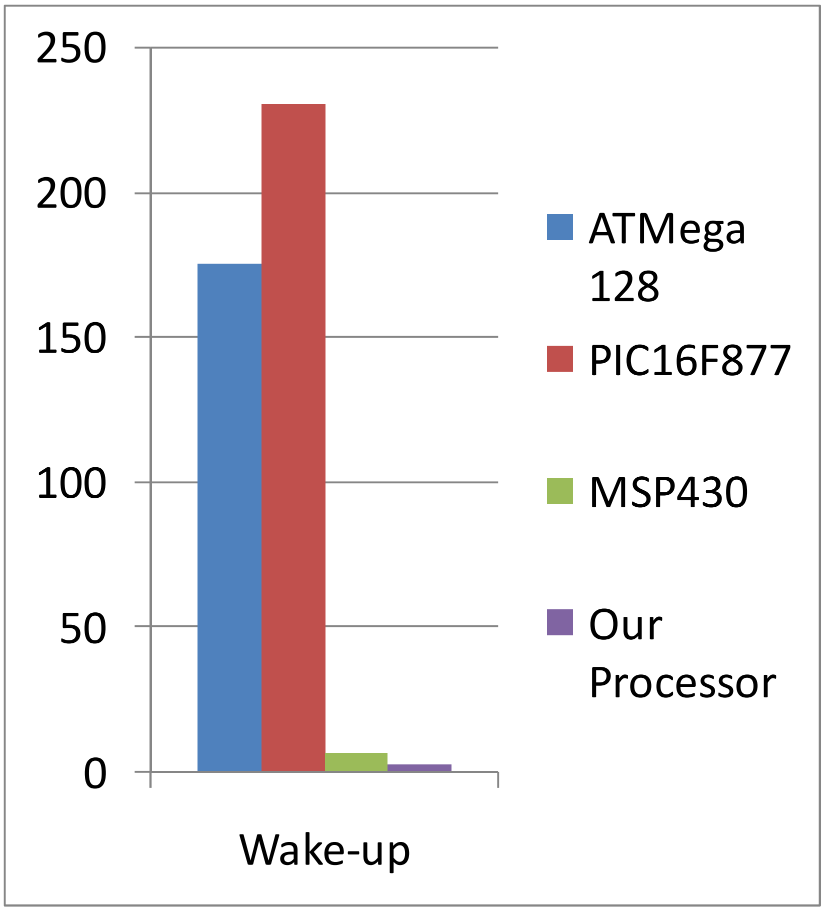
3. MIPS (Million-instructions-per-second)



As can be seen we surpass all the other MCUs in the MIPS at both 10MHz and 1MHz clock frequencies, showing that we have a good speed of operation for our design. This is necessary as duty-cycle is low (<1%) and so the processor must wake-up and 'quickly' finish its processing and sleep to save dynamic power.

The MIPS metric gives a reasonable gauge of speed of operation among similar architectures [8]. Varying instruction-sets cause trouble in using MIPS as a proper metric, but in our case all the processors in consideration are basically RISC-based. Even though our instruction-set is reduced compared to the others, we still have the same basic instructions and these are the ones that are frequently used. So, MIPS still gives a reasonable gauge of speed of operation.

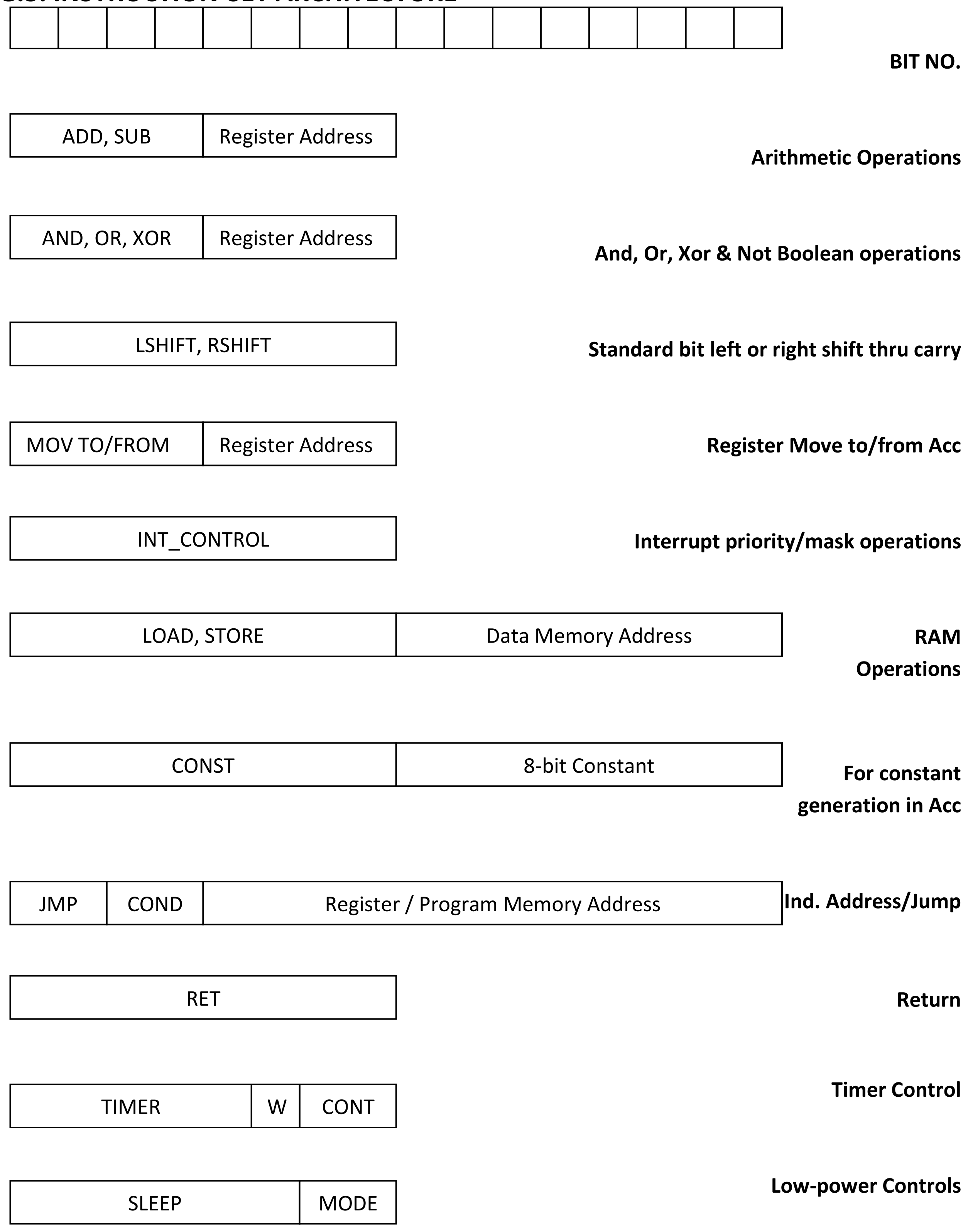
4. Wake-up Time (μs)



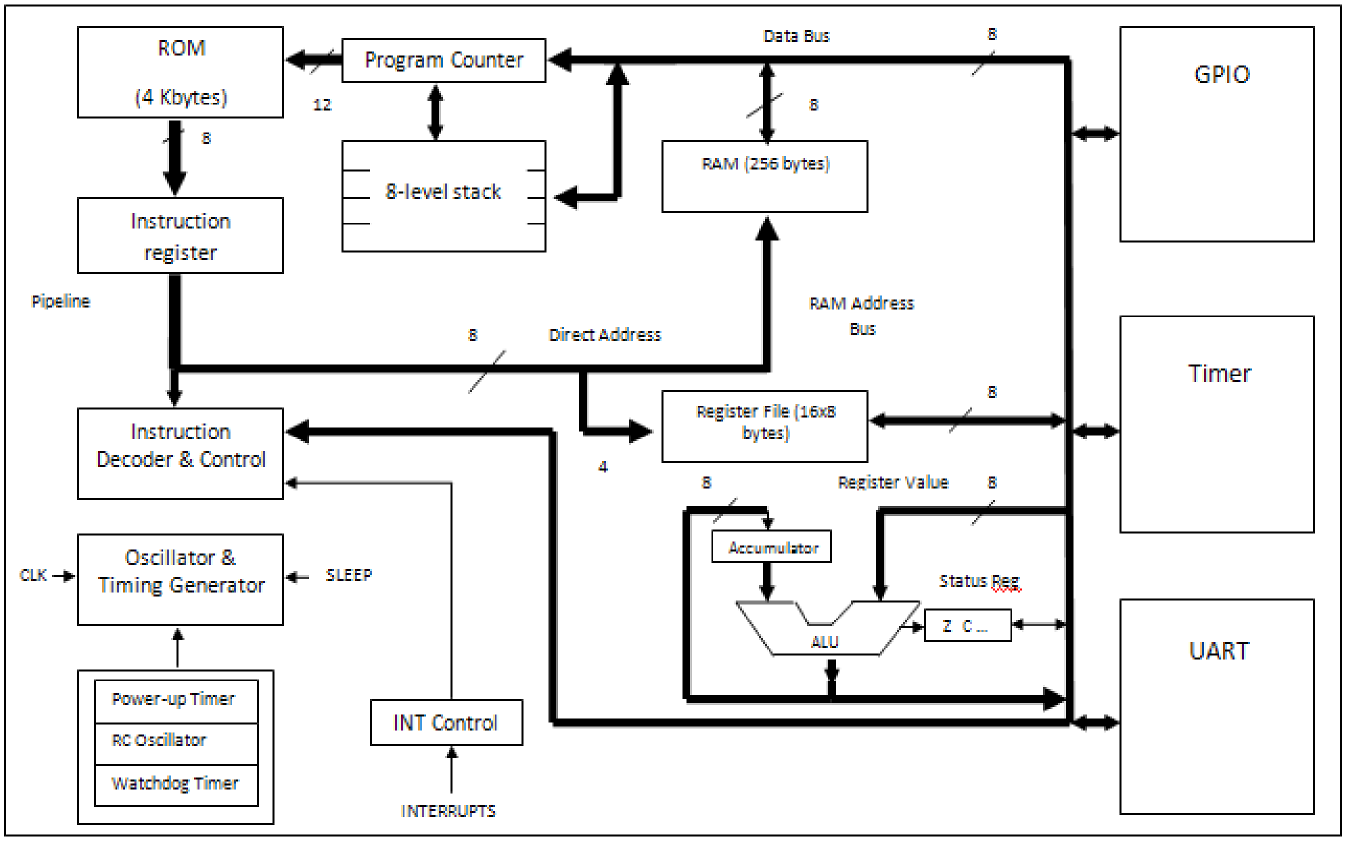
As the results how, our processor design has the lowest wake-up time among the given MCUs, with only the MSP430 coming close (~6μs).

Wake-up time is also an important parameter in sensor node processor design. As stated earlier, sensor nodes have low duty-cycle and so need to sleep and wakeup – if we have high wakeup time, we will miss the events / data samples because the events are real-time in nature with hard deadlines [2]. Thus, low wakeup times in the processor are an advantage in helping application designers easily meet event monitoring / data sampling deadlines even though the node may be asleep most of the time.

G.3. INSTRUCTION-SET ARCHITECTURE



G.4. BLOCK DIAGRAM



The above block diagram also sums up our high-level design of the processor. As can be seen, the design uses no unconventional patterns, but achieves superior efficiency through application specific features and use of common low-power techniques like clock gating for its sleep mode [8].

For more details or a soft-copy of the code, email me: digvijay.in@gmail.com

Chapter H

REFERENCES

[1] Lewis F. L., **Wireless sensor networks**. In *Smart Environments: Technologies, Protocols, and Applications*, 2004.

[2] Pottie G. J. and Kaiser W. J., **Wireless Integrated Network Sensors.** In *Communications of the ACM*, 2000.

[3] Lynch C. and O' Reilly F., **Processor Choice for Sensor Networks.** In *Workshop on Real-World Wireless **Sensor** Networks*, 2005.

[4] Ekanayake V., Kelly IV C. and Manohar R., **An Ultra-low-power Processor for Sensor Networks.** In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2004.

[5] **Microchip Technology Ltd.** www.microchip.com.

[6] **Atmel Corporation** www.atmel.com.

[7] **Texas Instruments** www.ti.com.

[8] Mysore S., Agrawal B., Chong F. T. and Sherwood T., **Exploring the Processor and ISA Design for Wireless Sensor Network Applications.** In *Proceedings of the 21st International Conference on VLSI Design,* 2008.

[9] Nazhandali L. and Minuth M., **SenseBench: Toward an Accurate Evaluation of Sensor Network Processors.** In *Workload Characterization Symposium,* 2005.

[10] Piguet C., Masgonty J. M., Arm C., Durand S., Schneider T., Rampogna F., Scarnera C., Iseli C., Bardyn J. P., Pache R. and Dijkstra E., **Low-power design of 8-b embedded CoolRisc microcontroller cores.** In *IEEE Journal of Solid-State Circuits,* 1997.

[11] **Crossbow Inc.** www.xbow.com

[12] **tinyOS** www.tinyos.net

[13] **Altera Inc.** www.altera.com

[14] **Synopsys Inc.** www.synopsys.com

[15] **Verilog HDL** www.verilog.com

[16] **CMOS** http://en.wikipedia.org/wiki/CMOS

[17] **SimVision** www.cadence.com