Custom Single-Purpose Event Processor Design: For Low Power WSN Node

KSHITIJ SHINGHAL[1], ARTI NOOR[2], NEELAM SRIVASTAVA[3], RAGHUVIR SINGH[4]
[1]Department of E&C Engg., M.I.T, Moradabad, India.
[2]Department of M. Tech. VLSI Design Group, C-DAC, Noida, India.
[3]Department of E&C Engg., I.E.T, Lucknow, India.
[4]Academic Advisor, Shobhit University, Merrut, India.

ABSTRACT

A basic processor consists of a controller and a data path. The datapath stores and manipulates a system's data. The data in the case of WSN node is the data acquired by the sensors of node after measuring the ambient conditions to be stored, processed and compressed for transmission; controller is capable of moving data through datapath. The combinational and sequential logic design techniques are applied to build a controller and datapath for a custom designed single-purpose event processor for WSN Node. Such custom designed single-purpose event processor results in several benefits in terms of faster performance, smaller size lower power consumption. WSN Node consists of sensors, a radio circuit a processor/microcontroller and some sort of power source. Event processor is custom designed, simulated, tested and implemented for single purpose application i.e. WSN Node using VHDL. Active HDL is used for simulation; to examine the various trade offs of using general purpose versus single purpose processor to implement necessary WSN node functionality.

KEYWORDS: *Datapath, VHDL, WSN Node, Active HDL, Custom Single purpose event processor.*

I. INTRODUCTION

Wireless Sensor Networks (WSN) is rapidly becoming popular and finding use in many areas of day to day life. The sensor nodes are able to monitor a wide variety of ambient conditions that include the following: flow, temperature, pressure, humidity, moisture, noise levels, mechanical stress, speed, etc. There are numerous areas for the deployment of wireless sensors: physiological monitoring, habitat monitoring, precision agriculture, forest fire detection, nuclear, chemical, biological attack detection, military, transportation, disaster relief, and environmental monitoring (air, water, and soil chemistry) etc. The basic components of a sensor node are shown in figure 1.

A typical sensor node consist of a sensing unit having sensor for sensing ambient conditions, a processing unit having a CPU for processing collected data and a transmission unit having transceiver for communication with base. Other units depending upon application are position finding system and a mobilizer all these units powered by a power unit sometimes connected to a power generator. Since maximum network life is desired and the amount of energy available for each node is finite (nodes are typically battery powered), low power implementations are a required for longer life of WSN node. In the proposed design a custom single purpose event processor is designed such as to meet the specific requirements of the processing unit of atypical processor node. Designing is done by determining the systems architecture for WSN Node, and then mapping the functionality through behavioral and structural description to that architecture. Designing a custom designed single-purpose event processor may result in increase in NRE cost and time to market as well as it will result in reduction in power consumption, number of transistor on chip and number of clock cycles (increasing speed).

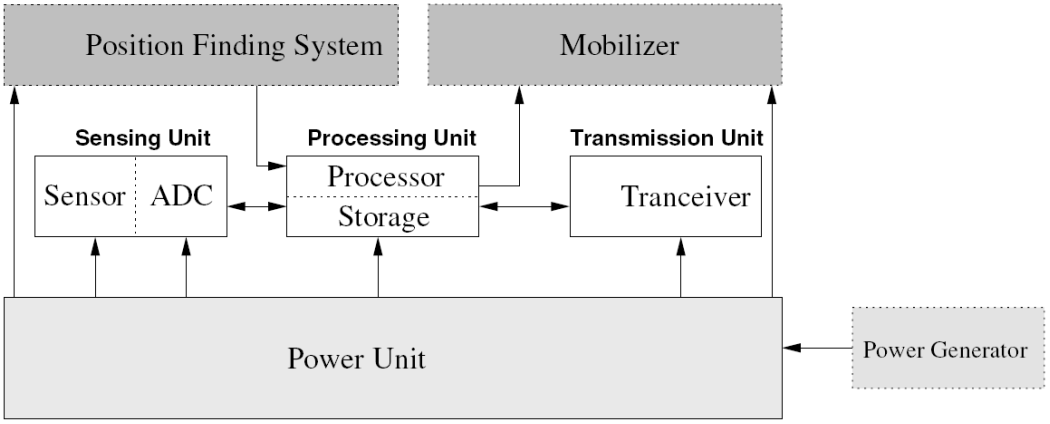

Figure 1 Components of a Sensor Node

II. MOTIVATION AND GOALS

Proposed architecture replaces most of the functionality of a general purpose microcontroller with an event-driven system specifically optimized for monitoring applications in an agricultural farm. A summary of design goals for the system architecture are presented below and detailed discussions of goals and how the architecture meets these goals follows in the following subsection.

1. **Event-Driven Computation:** Eliminate unnecessary event-processing overhead with event-driven hardware system architecture.
2. **Hardware Acceleration to Improve Performance and Power:** Build a system composed of several components that are optimized for specific tasks.
3. **Exploiting Regularity of Operations within an Application:** Optimize the common-case behavior within an application ie special purpose deign for single application (WSN node).
4. **Optimization for a Particular Class of Applications***:* Optimize the common-case behavior of monitoring applications to reduce power, while still providing general-purpose processing capability to enable broad functionality.
5. **Modularity:** Provide an easily extensible system architecture that allows different sets of hardware components to be combined into a larger system targeting a particular application.
6. **Fine-grain Power Management Based on Computational Requirements***:* Provide explicit programmer- accessible commands for fine-grain resource and power control.

III. ARCHITECTURE DESCRIPTION

To fulfill design goals, the basic functionality of a general-purpose microcontroller is replaced with a modularized, event-driven system. The system architecture is illustrated in Figure 2.
There are two distinct divisions within the system in terms of the positions of the components with respect to the system bus. We refer to the components to the right of the bus as slave components and those to the left as master components except the memory, which is a slave. The system bus has three divisions – data, interrupt, and power control. The slaves compete for the interrupt bus using centralized arbitration if more than one slave has an interrupt to signal. The slaves also respond to read or write requests from the master side on the data bus, thus allowing the masters to read their information content and control their execution.

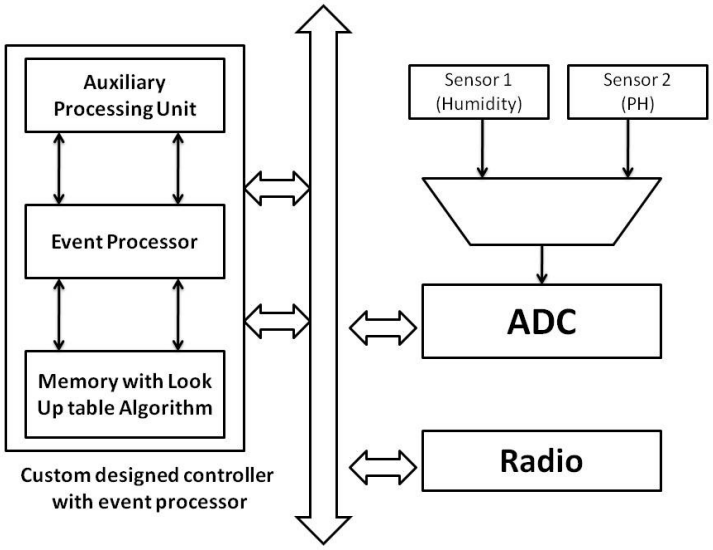

Figure 2 The system architecture

3.1 Event-Driven System We propose an event-driven system in which all of the master components are involved with event handling, and the slaves assist the master components in their tasks and signal the occurrence of events to trigger the master components. All external events, such as the beginning of radio packet reception, are expressed as interrupts by an appropriate slave component. The slave components also raise interrupts for their internal events, such as completion of an assigned task. To the master components, there is no distinction between external and internal events. Also, since the occurrence of all events is signaled by interrupts, we will use the terms event and interrupt interchangeably. The system idles until one of the slaves signals the presence of an event, and when all outstanding events have been processed, the system returns to its idle mode. Since all the system does is respond to events, there is no software overhead for interrupt handling.

3.2 Hardware Acceleration to Improve Performance and Power There is a general-purpose microcontroller in system designed. However, unlike other sensor network device architectures, the intent of design is for the microcontroller to be the last resort for any computation, i.e. the microcontroller should be called upon to perform a task only if the rest of the system does not have the requisite functionality. Specific tasks that are considered common to a wide variety of application are offloaded to hardware accelerators, which can be more power and cycle efficient than the general-purpose microcontroller. Hence, the microcontroller can usually be powered down by gating the supply voltage. This not only reduces active power but also leakage, which can be a very significant source of power consumption for low duty cycle operation. In the absence of a hardware timer, a software timer would have to be implemented in the microcontroller, requiring the microcontroller to always be active. There is a generic filter slave for basic data processing. In our architecture, this block is a simple threshold filter with a programmable threshold. We also implement a message processor block to offload packet processing and avoid waking up the microcontroller for common events such as packet forwarding and transmitting packets of collected samples. The slave components also include essential sensor network device components such as the radio, and a block of sensors and Analog-to- Digital Converters (ADCs).

3.3 Exploiting Regularity of Operations within an Application All immediate interrupt handling is offloaded to the event processor block while the microcontroller is powered down. The event processor is a simple state machine that can be programmed to handle an interrupt by transferring data blocks between the slave devices and setting up control information for these devices to complete their tasks.

The event processor can also be programmed to wake up the microcontroller if the requisite functionality for processing the interrupt is not otherwise available. To some extent, the event

processor can be perceived as an intelligent DMA controller. Thus, there are two levels of interrupt service routines (ISRs) to handle an interrupt: at the event processor level and at the microcontroller level. ISRs for both the event processor and the microcontroller are stored in the main memory, which is a unified instruction and data memory connected to the bus.

A regular event is one that can be processed wholly by the event processor and the slave components. An irregular event is one that requires the microcontroller. One of the tasks involved in mapping an application to our system is to determine the partitioning of events into regular and irregular events. The regularity of an event is determined by the functionality present in the slaves and the event processor. For a typical application, events such as sampling, transmitting samples, and forwarding packets would ideally be regular while application or network re-configurations would often be classified as irregular.

3.4 Optimization for Monitoring Applications The system can handle only one outstanding interrupt at a time. As a result, slave devices may continue to write their interrupts to the interrupt bus. However, if the system begins to be overloaded, events will simply be dropped. Outstanding events cannot preempt either the microcontroller or the event processor. The system bus, the microcontroller, and the event processor are all non-pipelined. All of these simplifications give rise to a light-weight system that is well suited to handle monitoring applications while consuming very little power.

3.5 Modularity The entire slave devices are attached to the system bus and are memory mapped. Both control and data are communicated to and from the slaves by simply reading from and writing to appropriate addresses in the memory. This memory mapped interface allows the system design to be extremely modular and new components and hence new functionality can be added on to the system bus without modification of the event processor or the microcontroller.

3.6 Fine-grain Power Management Based on Computational Requirements Since the master components are triggered by interrupts, the ISRs for each interrupt can configure the system according to its computational requirements for handling the interrupt. To sufficiently curb leakage power, special instructions within the event processor are used to gate the supply voltages of system components. Note that the system does not infer the resource usage for an event; rather, the ISR programmer selects the components to turn on depending on the needs of the application. Individual power enable lines are required for each component under direct control. V_{dd}-gating and power down implementations will vary depending on the circuit-level design of the individual slave components. Such power control may not only be exercised over the slave components, but also over segments within the main memory that contain temporary data, such as application scratch space. The event-driven programmable resource usage is one of the most significant innovations of the system design. It allows configuration of system power consumption with very little logic over- head, as opposed to a technique that attempts to infer resource usage. Also, it allows the addition of several specific components to the system as slaves that can be used in varying combinations to provide the functionality required by an application. Any component unused in an application can be turned off (i.e., supply voltage gated) and is nearly invisible during the entire lifetime of the application. Therefore, the system can satisfy the general-purpose requirements of applications by providing a broad range of slave components, enabling an on-demand functionality that imposes negligible overhead when a component is not required.

IV. SYSTEM COMPONENTS

Details of other system components are described in following section:

4.1 System Bus As discussed earlier, the system bus comprises the data bus, the interrupt bus, and power control lines. The data bus has address, data, and control signals indicating read and write operations. In current implementation the address bus has 16 lines, the data bus has 8 lines, and there is one control signal each for read and write operations. The address space for memory-mapped architecture is therefore 64K. The address and control lines can be driven only by the event processor and the microcontroller in mutual exclusion as determined by the bus arbiter, which is currently just a mux. The data lines are driven by the slave that determines that the current request lies in its address range, and are demultiplexed to the originator of the request, i.e., the event processor or the microcontroller.

The interrupt bus has 6 address lines and control signals for arbitrating the writing of interrupts by slaves. The system is therefore capable of handling 64 interrupts in the current model. The event processor has control signals in the interrupt bus to indicate when it has read the current interrupt address.

The power control lines are handshake pairs for each slave or memory segment controlled. The handshake is relevant only when a component is turned on, to determine the time when the component can be used. The system currently makes no assumptions about the time taken to wake up for the components over which explicit power control is exercised.

4.2 Microcontroller The microcontroller is used to handle irregular events, as discussed in previous sections, such as system initialization and reprogramming. The microcontroller is a simple nonpipelined microcontroller. It implements an 8-bit Instruction Set Architecture (ISA). The system uses available computational cores with necessary modifications for low-power features required for the system.

4.3 Event Processor The event processor is essentially a programmable state machine designed to perform the repetitive task of interrupt handling. Figure 3 illustrates a simplified version of the actual state machine within the event processor

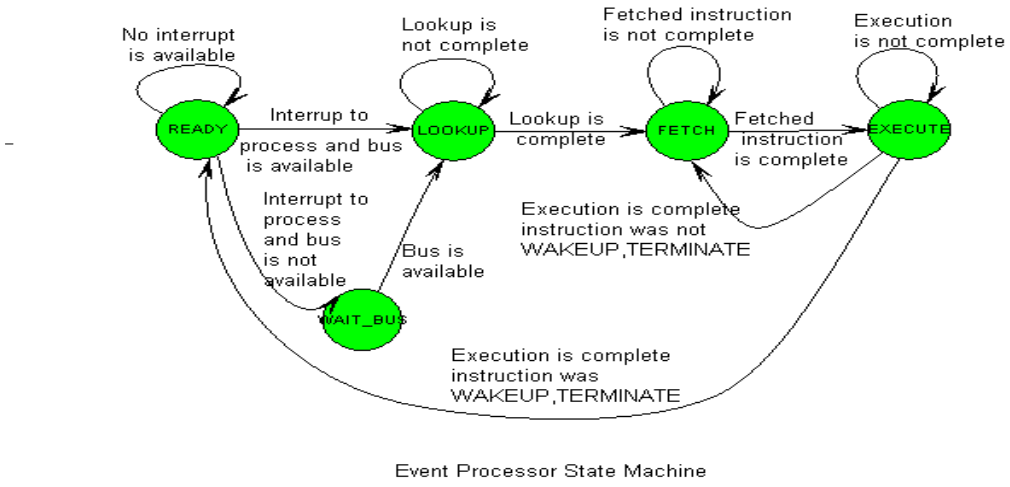

Figure 3 Event processor state machine diagram

The event processor idles in the READY state until there is an interrupt to process. When an interrupt is signaled, the event processor transitions to the LOOKUP state if the data bus is available, i.e., the microcontroller is not awake. If not, the event processor transitions to the WAIT BUS state and waits until the microcontroller relinquishes the data bus. In the LOOKUP state, the event processor looks up the ISR address corresponding to the interrupt. The lookup table is stored in memory, and the starting location of the table, offset by an amount proportional to the interrupt address, contains the address of the event processor ISR. When the lookup is complete, the event processor transitions to the FETCH state, in which the first instruction at the ISR address discovered in the LOOKUP state is fetched. The event processor stays in the FETCH state until all the words of the current instruction have been fetched, and then it transitions to the EXECUTE state.

The instructions within an event processor ISR can be one of the following – SWITCHON, SWITCHOFF, READ, WRITE, WRITEI, TRANSFER, TERMINATE, or WAKEUP. Table 1 provides a summary of the operations corresponding to the instructions. The event processor has one register used to store temporary data. The op- codes are each 3 bits and the instructions vary in the number of words they span.

Table 1 instruction Set

Instruction	Size	Description
SWITCH ON	One Byte	Turn on a component & wait for acknowledgement that the component is ready to proceed.
SWITCH OFF	One Byte	Turn off a component.
READ	Three Bytes	Read a location in address space & store to the register.
WRITE	Three Bytes	Write a location in address space & store to the register.
WRITEI	Three Bytes	Write an immediate value to a location in address space.

TRANSFER	Five Bytes	Transfer a block of data within the address space.
TERMINATE	One Byte	Terminate the ISR without waking up the microcontroller.
WAKEUP	Two Bytes	Terminate the ISR & wake up the microcontroller at a microcontroller ISR address.

The EXECUTE state holds until the instruction has been completely executed, e.g., the complete transfer has been completed for a TRANSFER instruction. A component is completely powered on for the SWITCHON instruction. If the instruction is not a WAKEUP or TERMINATES instruction, the event processor returns to the FETCH state and fetches the next instruction in the ISR for execution. For WAKEUP or TERMINATE instructions, the event processor returns to the READY state and waits for the next interrupt regular message processing tasks, including message preparation and routing. Simple tasks such as table lookup and check-sum calculations can be sped up using hardware implementations (with low power overhead).

Currently, the message processor interface has two memory blocks for each message as well as memory-mapped control words. Data is transferred to the message processor from sensor devices and once the message has been prepared the message processor fires an interrupt and the message are sent to the radio. All incoming messages are transferred from the radio to the message processor. If the message is a regular message, the message processor looks up whether the message should be forwarded. If the message is an irregular message, then an interrupt is fired and the event processor wakes up the microcontroller.

V. SIMULATION SETUP

A synthesizable implementation of all the blocks of the figure 3 will be captured at the register transfer level (RTL) and written in VHDL, and integrated in to a simple system. The controller fetches instruction from its read only program memory and decodes them using the decoder components. The ALU component is used to actually execute arithmetic operation. The source and destination of these operations are registers that reside in the internal RAM of the processor. A VHDL simulator / compiler Active HDL is used to compile VHDL program. The ROM is generated using the special in build module in Active HDL which outputs the VHDL description of the desired specification of the ROM. Now all the components of the system are ready to be connected together to make WSN Node. Now all the software modules are compiled and linked to obtain VHDL executable program, which are simulated, tested and finally converted into gate level netlist. The simulation is done using Active HDL, VHDL simulator software. A design flow diagram of Active HDL is shown in figure 4.

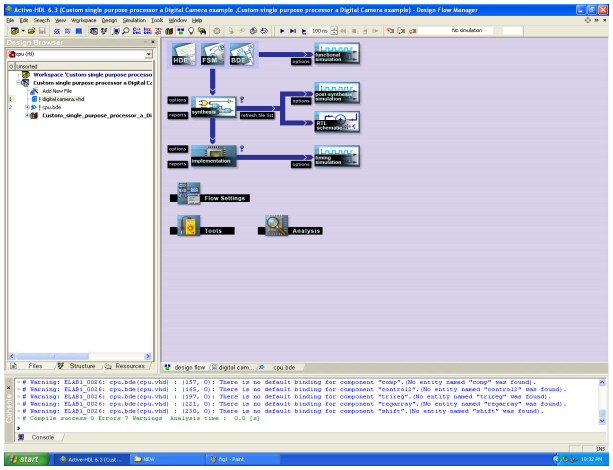

Fig.4 : Design flow diagram for Active HDL

Active HDL takes as input the VHDL files, making the system, and functionally simulates the execution of the final design but interpreting the descriptions. The event processor block diagram generated through Active HDL is shown in figure 5.

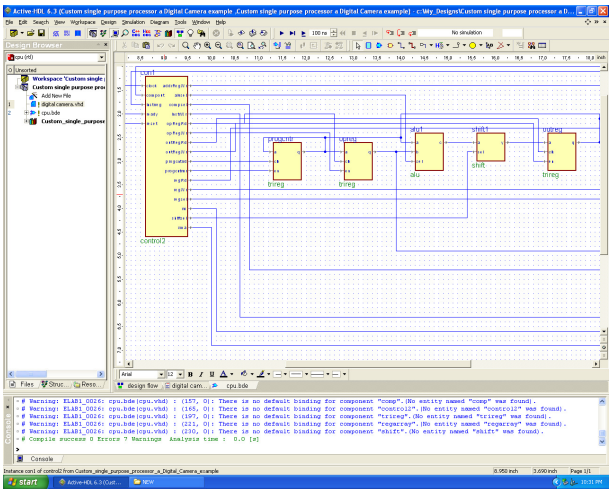

Figure 5 The event processor block diagram generated through Active HDL

With the help of simulation it is determined whether the design is functioning correctly. Moreover the amount of time or clock cycles can be measured, that the custom single purpose WSN Node system takes to processor a single image. This is the first metric of interest that is performance. Figure shows how after simulating the VHDL models, we obtain the execution time. Then gate level model is stimulated to obtain the immediate data necessary to compute the power consumption of the circuit.

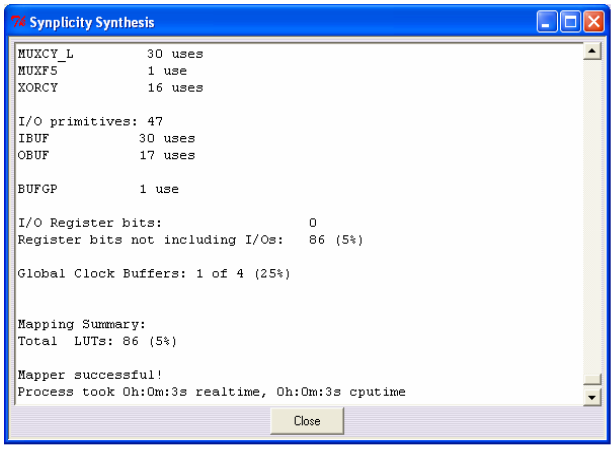

Figure 6 Synthesis report generation through Synplify pro

One of the designs is tested for the correct functionality, the synthesis tool Synplify pro is used to translate the VHDL files down to an interconnection of logic gates. A synthesis tool is like a compiler for single purpose processors. It reads a VHDL file and translates it into a corresponding gate level description. Here Synplify Pro Version- 7.5 is used for the synthesis of WSN Node. A synthesis report generation through Synpilfy pro is shown in figure 6. Now we also obtain the total number of gates to implement design area required to implement WSN Node, total power consumption by keeping track how many times the gate switches from zero to one and from one to zero since the power consumption can be approximated in the amount of switching that takes place in the circuit is known.

VI. POWER ESTIMATION METHODOLOGY

6.1 Methodology Since the power consumption of our system can be accurately characterized only after a fabricated chip has been measured, we restrict ourselves to obtaining a conservative estimate based on the active power consumption of the system for its most frequent activities, i.e. the processing of regular events. Again, since we expect to use commodity radio and sensor components,

we do not consider these components in our estimates.
The event processor is the largest power consumer in the system since this is a component that must always be powered. Moreover, this block has the most complex micro-architecture of all of the components involved with regular event processing. Hence, for this block, we have obtained conservative estimates by going through the complete process of synthesizing a VHDL model, performing placement, routing, and simulating the final netlist. For the other components, we have broken them down into common substructures such as incrementers, comparators, buffers, etc., and estimated the power consumption numbers for all of these components by simulating netlists synthesized for these sub-structures and combining the results.

6.2 Power Estimates The power estimates for the main components of the system are presented in Table 2. The power numbers are shown for active and idle modes (gated clock) of each component at a supply voltage of 5V and a clock frequency of 100KHz. The active power consumption corresponds to a situation in which the event processor always has an outstanding interrupt to handle. Therefore, the event processor is always switching in this mode because it begins to process a new interrupt the moment it gets done with the current one. The idle power corresponds to a duration in which the event processor is not provided an interrupt to handle. Both of these situations are extreme cases that we do not expect in normal situations.

Table 2. Power Estimates for Regular Event Processing in the System

	Idle/Active	Vdd 5(V)
Event Processor	Active	14.25 µW
	Idle	0.018 µW
Timer	Active	5.68 µW
	Idle	0.024 µW
Threshold Filter	Active	0.42 µW
	Idle	~0.0 µW
Memory	Active	2.07 µW
	Idle	0.003 µW
System	Active	24.99 µW
	Idle	0.070 µW

VII. SIMULATION AND RESULTS

After analyzing the final implementation using the approach outlined in figure through simulation following points were observed

- The power consumption was measured to be 24.99 µW.
- This means that the batteries will last 12 times longer than the design having general-purpose processors.
- The area is measured to be 128000 gates.
- The size of the chip has increased significantly (since all the components are now fabricated on a single chip).
- There is significant increase in NRE cost and time to market.

The results are summarized in table3. Each of the design metrics is shown through graphs in figures 7 & 8 respectively.

Table 3 summary of design metrics

	WSN Node I*	**WSN Node II****
Power(Watt)	36.78	24.99
Size(gate)	90,000	128,000

* WSN Node with separate microcontroller
** WSN Node with event processor.

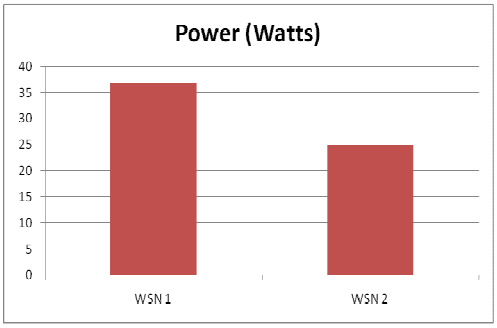

Figure 7 power consumption for WSN nodes

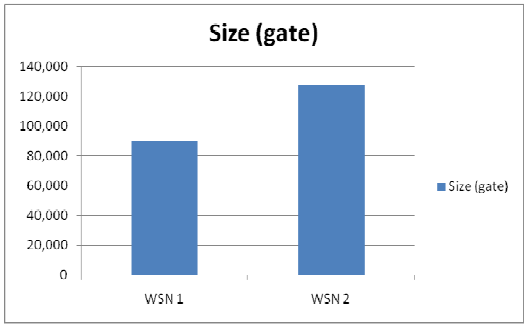

Figure 8 Size of WSN node per gate

VIII. CONCLUSION

This work presents an event-based processor for WSN node. The event-based processor has allowed a considerable decrease in the number of changes in the control action and made possible the compromise between quantity of transmission and performance. Event-based processor for WSN node will not only reduce post processing latency but will also reduce the overall power consumption in a wireless sensor network node. This is achieved by reducing the amount of data transmitted by the post processing unit which shares a large amount of total power consumed by a WSN node there by resulting in a low power WSN node. WSN node using customizable low power processor can find a wide application in different areas, especially where it is needed to monitor and save bulk data for different agricultural and non-agricultural practices.

REFERENCES

[1] Beutel, Jan. Geolocation in a PicoRadio Environment. Masters Thesis. 2000, 9.
[2] Cetintemel, Ugar, Flinders, Andrew, and Sun, Ye. Power-Efficient Data Dissemination in Wireless Sensor Networks. International Workshop on Data Engineering for Wireless and Mobile Access Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access. 2003, 1-8.
[3] Chen, Wei-Peng and Sha, Lui. An Energy –Aware Data-Centric Generic Utility Based Approach in Wireless Sensor Networks. Information Processing In Sensor Networks Proceedings of the third international symposium on Information processing in sensor networks. 2004, 215-224.
[4] Dousse, Olvier, Mannersalo, Petteri, and Thiran, Patrick. Latency of Wireless Sensor Networks with Uncoordinated Power Saving Mechanisms. International Symposium on Mobile Ad Hoc Networking & Computing Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing. 2004, 109-1200.
[5] Hill, Jason, Horton, Mike, Kling, Ralph, and Krishnamurthy, Lakshman. The Platforms Enabling Wireless Sensor Networks. Communications of the ACM June 2004/ Vol47. No. 6. ,41-46.
[6] Polastre, Joseph, Hill, Jason, and Culler, David. Versatile Low Power Media Access for Wireless Sensor Networks. Conference On Embedded Networked Sensor Systems Proceedings of the 2nd international conference on Embedded networked sensor systems. 2004, 95-107.
[7] Rabaey, Jan M., Ammer, M. Josie, Silva, Julio L. da, Jr., Patel, Danny, Roundry, Shad. PicoRadio Supports AdHoc Ultra-Low Power Wireless Networking. Computer Magazine (July 2000), 42-44.
[8] Savarese, Chris. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. Masters Thesis. 2002, entire thesis.
[9] Savarese, Chris, Rabaey, Jan M., Beutel, Jan. Locationing in Distributed Ad-hoc Wireless Sensor Networks. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2001, 2037-2040.
[10] Seada, Karim, Zuniga, Marco, Helmy, Ahmed, and Krishnamachari, Bhaskar. Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks. Conference On Embedded Networked Sensor Systems. Proceedings of the 2nd international conference on Embedded networked sensor systems. 2004, 108-121.
[12] Shih, Eugene, Cho, Seong-Hwan, Ickes, Nathan, Min, Rex, Sinha, Amit, Wang, Alice, and Chandraskasan, Anantha. Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks. International Conference on Mobile Computing and Networking. Proceedings of the 7th annual international conference on Mobile computing and networking. 2001, 272-287.

[13] Stemm, Mark, and Katz, Randy H, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, vol. E80-B, no. 8. Aug 1997, 1125–1131.
[14] Wei Ye; Heidemann, J.; Estrin, D. An energy-efficient MAC protocol for wireless sensor networks**;** INFOCOM 2002. Twenty**-**First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume 3, 23-27 June 2002, 1567 - 1576 vol.3.
[15] Wei Ye; Heidemann, J.; Estrin, D. Medium Access Control With Coordinate Adaptive Sleeping for Wireless Sensor Networks. IEEE/ACM Transactions on Networking, Vol., 12, No. 3, June 2004, 493-506.

Authors Biographies

Kshitij Shinghal has 11 Years of experience in the field of Academic and is actively involved in research & development activities. He obtained his Masters degree (Digital Communication) in 2006 from UPTU, Lucknow. He started his career from MIT, Moradabad. Presently he is working as an Associate Professor & Head, Deptt of E&C Engg., at MIT Moradabad. He has published number of papers in national journals, conferences and seminars. He has guided two Masters, more than sixty students of B. Tech, and guiding three M. Tech. theses. He is an active Member of Various Professional Societies such as ISTE, IACSIT, IAENG etc.

Arti Noor has 17 Years of R&D experience in the field of VLSI design & technology characterization, VHDL & computer programming, and speech synthesis. She obtained her Ph.D. (Electronics and communication Engineering) in 1990 from Banaras Hindu University, Banaras. She started her career from CEERI, Pilani and then joined the CDAC, Noida as Scientist EI. Presently she is working as an Associate Professor (Scientist-E) & HoD, M. Tech VLSI Division at CDAC Noida. She was also involved in various research development activities in CEERI, Pilani and in CDAC, Noida. She worked on many consultancy projects for ISRO Bangalore, IIT Delhi, ISAC Bangalore, and VSSC Trivandram. She has guided Two Ph.D in the area of Microelectronics and VLSI Design, 50 students of B.Tech/M.Sc./ M. Tech/ ME, more than 15 M.Tech. theses.

Neelam Srivastava has 22 Years of experience in the field of Wireless Sensor Network design. She obtained her Ph.D. (Electronics and communication Engineering) in 2004 from Lucknow University, Lucknow. She started her career from IET, Lucknow. Presently she is working as an Associate Professor, Deptt of E&C Engg., at IET Lucknow. She is involved in various research & development activities. She is guiding two Ph.D. in the area of Wireless Sensor Networks, 50 students of B. Tech, more than 15 M. Tech. theses.

Raghuvir Singh has experience in Research, Development, Teaching and Administration for more than 40 years. He obtained his B.Sc., B.E. (Telecommunication), M.E. (Electronics) and Ph.D. (Electronics and communication Engineering) Degrees in 1958, 1962 & 1970 respectively. He worked in CEERI, Pilani and retired as Head of Electronics & Communication Engineering Department of University of Roorkee (presently IIT Roorkee). He was awarded the IETE award in 1965, Khosla Research Award in 1970 and Anna University National Award for his outstanding career and contribution to Engineering and Technology in 1994.His name was recommended by IETE Award Committee for the FICCI Award in 1999. He has supervised 5 Ph.D. theses, 45 M.E. dissertations and has more than 50 publications in National and International Journals and conferences to his credit.