Energy Efficient Task Allocation and Scheduling in Distributed Homogeneous Multiprocessor Systems

ANJU S. PILLAI, T.B. ISHA
Department of Electrical and Electronics Engineering
Amrita Vishwa Vidyapeetham,
Amrita School of Engineering, Coimbatore
INDIA
s_anju@cb.amrita.edu, tb_isha@cb.amrita.edu

Abstract: - With the advent of semi conductor technology, the development of more complex embedded real time applications is made possible today. This accelerates the development and support for multiprocessor based systems. The paper presents the development of "*a power-aware real time embedded system for temperature monitoring and control in safety critical applications*". The main objective of the work is to perform a hardware implementation of task allocation for multiprocessor systems based on task dependencies and precedence relations and schedule the real time tasks. The proposed work also provides an energy-aware solution by integrating Dynamic Voltage Frequency Scaling (DVFS) technique and shutting off the unused peripherals of the processor. A laboratory model of multiprocessor based experimental setup was developed to validate the functioning of algorithms using ARM7 LPC2148 microcontrollers.

Key-Words: - Multiprocessor systems, Energy minimization, Embedded Systems, Microcontrollers, Scheduling, Task-processor allocation, Dynamic voltage frequency scaling

1 Introduction

The wide spread use of multiprocessor based systems are due to the extended performance capabilities, balanced work load distribution, increased flexibility with good responsiveness etc. But, these advantages comes with the cost of increased complexity, additional task-processor allocation strategies, inter processor task communication and synchronization. Thus, implementation of multiprocessor based systems is highly complex and laborious. Most of the recent researches in the field of real time embedded domain are pertaining to development of multiprocessor system for various application developments. Most of research works are carried out on the software side, focusing on development of algorithms and protocols, addressing performance issues, energy issues, memory conflicts, meeting temporal constraints etc. Also, multiprocessor systems are used in many applications including automobiles, industrial control applications, embedded application development, fault tolerant systems, motor control applications and many more. The features of multiprocessor system may vary from application to application.

One of the critical domains where most of researches are concentrated is exploring the cause of power consumption in various embedded processors and devising different techniques and strategies for its reduction. There is an ample amount of methods proposed in the field for power consumption reduction of processor for both uniprocessor and multiprocessor systems. These techniques can be broadly classified as: software techniques, hardware techniques and power-aware techniques [1], [2]. The power-aware techniques are relatively new in the field, which combines both hardware and software techniques for energy saving. Hardware implementation of different techniques on real time applications is rarely attempted due to the difficulty in realizing these techniques on practical systems [3]. This paper presents the hardware implementation of an energy-aware task-processor allocation algorithm for multiprocessor systems based on task dependencies and precedence relations and scheduling of real time tasks for temperature monitoring and control in safety critical application.

The major contributions of the work include:

- Software simulation of a novel task dependency based allocation algorithm and scheduling of real time tasks for the application of temperature monitoring and control in safety critical systems
- Experimental testing and validation of job allocation and scheduling in multiprocessor system using ARM7 LPC2148 microcontrollers

The rest of the paper is organized as follows: section 2 describes the related work carried out in the field followed by system overview in section 3. The problem statement is given in section 4 and solution steps and methodology are presented in section 5. Section 6 briefs the hardware description and simulation results are presented in section 7, followed by experimental validation procedures in section 8. In section 9, hardware experimental results are furnished and finally the paper is concluded in section 10.

2 Related Work

Need for power consumption reduction in embedded processors and systems are well recognized in the field, especially for portable embedded systems [3], [4]. Energy issues being the most crucial one, needs prime attention and solution. But, this being complex and challenging, yet there are a wide scope of exploring newer techniques for the same. There is a strong literature support to validate this [5], [6], [7]. The different factors which influence the processor functioning to cause power consumption are examined [8], [9]. As software means of power consumption reduction are the most prevailing in the field, consists of various methods and techniques viz. development of algorithms and protocols for energy saving, profiling by software [2], [10]. Compiler optimization techniques for energy saving is implemented by identifying power-aware instructions in [11]. As software means of energy consumption reduction does not alter the underlying hardware of any systems, are usually preferred. But, these techniques do not consider factors like heat loss, variations in atmospheric conditions, etc. thereby imposing a restriction on using such simulation based techniques.

There is sufficient literature support for energy consumption reduction attempted by hardware means like: DVFS, actual energy consumption measurement on processors etc. [12], [13]. Most of these works perform the measurement of actual current consumed by the processor while executing different tasks. Use of hardware performance counters to generate CPU signals at different intervals of processor execution is also attempted in the field [14]. Even though this method gives accurate results, as tasks becomes more application dependant, the current measurement becomes just an approximation of current consumed by the processor core alone. Approaches for reduction of power hunger off-chip memory access are always carried out by various means like: holding the array values of next few instructions to be executed within the processor registers etc. But, this introduces a limitation by adding more instructions in the execution loop to perform this, thereby increasing the execution time of the task code [15], [16]. Combining DVFS with power saving modes, sleeping modes etc. are also common in the field, for energy saving of processors [17]. Use Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) for performance and power analysis for multiprocessor systems are attempted in [18].

This paper presents software simulation and hardware implementation of job allocation and scheduling of real time tasks in multiprocessor application. The objective of the work is to *develop a power efficient temperature monitoring and control system in distributed network for a multi processor platform.* For the implementation, a task dependency based assignment algorithm is proposed and Rate Monotonic (RM) scheduling policy is used for task scheduling. The multiprocessor system is implemented using *Client-Server* model. The software simulation as well as hardware implementation of the application is done using ARM7 LPC2148 microcontrollers.

3 System Overview

The system under consideration consists of *P* homogeneous processors. There are a set of *n* independent and *m* dependent tasks which are to be scheduled on *P* processors in the system. Each task in the system is periodic in nature. The optimal number of processors required to execute the given number of tasks are found out by computing the total utilization of the task set. In the present system, a shared memory multiprocessor system based on *Client-Server model* is implemented. The server node acts as the shared memory and maintains the information regarding every task in the system. The problem of job allocation is established using a global scheduling approach where, a main queue is maintained in the server node. All the tasks in the system are assigned a static priority based on the

well known RM scheduling policy. The different tasks in the system are ordered in the main queue, based on task dependencies and precedence relations. Upon the identification of dependent tasks, these *m* tasks are grouped and assigned to the available processors in the system along with the independent *n* tasks. The scheduling decisions are made by individual processors only.

This paper investigates power-aware embedded system for temperature monitoring and control in safety critical applications. The basic objective of the work is to provide a power-aware solution for a real time multiprocessor system which is employed to monitor and control the temperature of a system. Typically, the application could be temperature monitoring and control of a room where a patient is admitted like, an Intensive Care Unit (ICU). The embedded device should control the temperature when required without incurring any problem to the patient's health condition and also generate an alarm for warning when the temperature changes beyond the threshold values. Such systems need to operate throughout the day so, incorporating methods to provide a power-aware solution will be highly demanded.

3.1 Task Model

The system consists of a set of *n* independent and *m* dependent and periodic tasks $\Gamma = \{\tau_1, \tau_{2,\ldots.}\tau_n\}$. The attributes of each τ_i task is represented by (C_i, T_i, d_i) where, C_i the worst case execution time, T_i the time period and d_i the dependency of the task τ_i. A task τ_i is said to be dependent on another task τ_j due to data dependency, need for sharing of common resources, the features of application to be developed or it could be due to control/conditional dependency. In RM scheduling, a task τ_i is assumed to have deadline $D_i = T_i$. Being periodic in nature, every task appears once in its time period. Once a task starts executing on a processor, it either completes execution or can be preempted by a high priority task and resume its execution at a later time. Otherwise, no task is allowed to suspend its execution by itself. From the total *n* independent and *m* dependent tasks, using the assignment policy a set of *k* tasks is assigned to P_i processor. In any case the total utilization of the task set U_i should be within the available processor capacity of the system.

If the available processor capacity is *c*, then

$$U_i = \sum_{i=1}^{n+m} \frac{C_i}{T_i} \leq c \quad (1)$$

The proposed model adopts a partitioned approach for multiprocessor task allocation and scheduling where; tasks are first allocated to the available processors and then scheduled in the local client nodes [19]. Once the tasks are allocated to individual processors, tasks are assigned priorities by RM policy. The RM priority assignment rule is *shorter the time period, higher the priority*. The assigned priorities remain static throughout the task execution period.

3.2 Processor Model

A multiprocessor system of *P* homogeneous *q* processors P_1, P_2,....P_q are considered. The homogeneous processors are meant to have similar frequency and voltage ratings and exhibit similar computational capacities. Each P_i processor is a DVFS processor, which can dynamically change its supply voltage and clock frequency at run time to conserve power when required. In the proposed system, Dynamic Frequency Scaling (DFS) technique is implemented in each processor for energy consumption reduction. The processor frequencies can be varied within the range of f_{min} to f_{max}. It is assumed that when the processor is made to run at a lower frequency, which will enable the processor to run with reduced energy consumption with a corresponding reduction in speed. But, at no case a processor is made to run at a lower frequency, which may result in deadline miss for any of the tasks in the system. The current system uses ARM7 LPC2148 microcontroller for implementation. The operating points of ARM7 microcontroller is as shown in Table I.

Table I: Operating points of ARM7 LPC2148 microcontroller

Operating points	Clock frequency (MHz.)	Supply voltage (V)
O_1	12	2.8
O_2	24	3.0
O_3	48	3.1
O_4	60	3.3

3.2 Energy Model

Energy consumed by any embedded processor can be represented as:

$$Energy, E = Power * Time \quad (2)$$

Power dissipated or consumed by a processor which is operated at a supply voltage of v volts and clock frequency f Hz is given by:

$$Power\ Dissipation = C_{eff} * v^2 * f \quad (3)$$

where: C_{eff} the effective load capacitance, v the supply voltage and f the clock frequency. Thus, the energy consumption is given by:

$$E = C_{eff} * v^2 * f * t \quad (4)$$

where: t is the time duration in which the processor executes the tasks.

The total energy consumption of the system or the energy consumed by all the q processors in the system is given by:

$$E_{system} = C_{eff} \sum_{i=1}^{q} v_i^2 * f_i * t \quad (5)$$

In the present model, the total energy consumed by the system is the sum of energy consumed by the server node and the client nodes. Thus,

$$E_{system} = E_{server} + E_{clients}$$

$$= E_{server} + C_{eff} \sum_{k=1}^{l} v_k^2 * f_k * t \quad (6)$$

For energy saving, the processor is operated at reduced clock frequencies, without compromising the timeliness. i.e. without missing the task deadlines. The energy saving is calculated by comparing the energy consumption for a fixed period of time (Δt) when operated at different frequencies. For a processor, when operated at rated voltage and rated frequency, the energy consumption is given by:

$$E_{rated} = C_{eff} * v_{rated}^2 * f_{rated} * \Delta t \quad (7)$$

The energy consumption of the processor when operated at a lower frequency f_1, than rated value ($f_{rated} > f_1$) is given by:

$$E_1 = C_{eff} * v_{rated}^2 * f_1 * \Delta t \quad (8)$$

The energy saving is thus calculated as:

$$\% E_{saving}(SFS) = \frac{E_{rated} - E_1}{E_{rated}} * 100 \quad (9)$$

This is actually the energy saving obtained by Static Frequency Scaling (SFS), where each task instance is operated at a fixed lower voltage and frequency value. When task instances are operated at different clock frequencies as required, it becomes Dynamic Frequency Scaling (DFS). The corresponding energy saving is represented as:

$$\% E_{saving}(DFS) = \frac{E_{rated} - E_{variable}}{E_{rated}} * 100 \quad (10)$$

where: $E_{variable}$ is the energy consumed by the processor when different task instances are operated at different frequencies, as demanded for energy conservation.

4 Problem Statement

For a set Γ with n independent and m dependent periodic tasks and P homogeneous processors in the system, the goal is to implement a task allocation and scheduling in a multiprocessor system for energy minimization subject to the condition:

- The total utilization of the task set is less than or equal to available processor capacity
- Individual processor assignment is done without violating the schedulability by RM policy
- All the processors in the system are DVFS processors
- Energy saving is accomplished without compromising the temporal requirement of the tasks i.e. without missing the task deadlines

5 Methodology

The solution steps include details about the task-processor assignment algorithm and scheduling policy. These details are described in the following sub sections.

5.1 Task Processor Allocation Algorithm

Once the number of tasks and the task attributes are known, this information is stored in the central server, which acts as the shared memory. It is the responsibility of the server to allocate the tasks to the available processors in the system. The main objective of the work is to minimize the power consumption of the embedded multiprocessor system. For this, a task assignment strategy is devised to minimize the power consumption by grouping the tasks based on task dependencies and precedence relations. The task allocation strategy is described in the following steps:

- Fix the number of processors required to run all the tasks in the system by calculating the task set utilization
- Number of processors required

$$= \left\lceil \sum_{i=1}^{n+m} \frac{C_i}{T_i} \right\rceil \quad (11)$$

- Check the task dependencies and pack the dependent tasks together
- Assign the dependent tasks to the same processor, without violating the schedulability by RM policy
- Upon the completion of dependent task allocation, next step is allocation of independent tasks
- Tasks which does not share any common resources are allocated to the available processors to balance the work load on various processors

5.2 Scheduling of Tasks

After task allocation, scheduling happens in client nodes. The prime objective of scheduling is to make sure that every task in the system is able to run successfully meeting both functional and temporal requirements. The client node takes the scheduling decisions to assign priorities to individual tasks. The priority assignment is by RM policy, which is a static priority assignment algorithm. In RM scheduling, tasks with shorter time period is considered critical, as it is the frequently occurring task in the system. The assigned priorities remain constant throughout the execution period.

6 Hardware Details

The following sub section presents details about the application and hardware setup.

6.1 Application Details

The system under consideration is a multiprocessor system, which consists of 4 processors implemented using four ARM7 LPC2148 microcontroller. The system is implemented using shared memory architecture using a *Client-Server* model. The basic system model is shown in Figure 1.

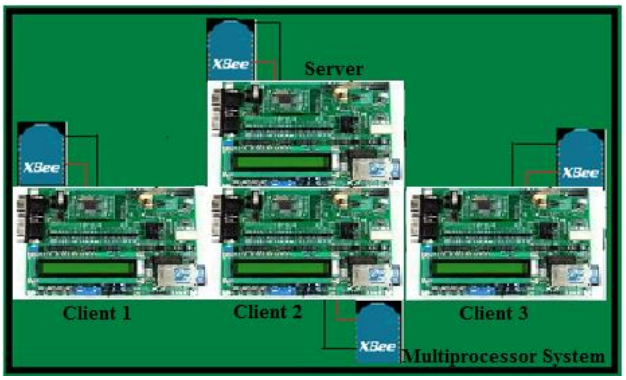

Figure 1: System Model

The server node acts as the shared memory and the other nodes are the clients, which communicate through the common memory. Direct client to client communication is not possible in the present architecture. The objective of the work is to develop a feasible algorithm to schedule the real-time tasks in multiprocessor system so as to meet both the functional and temporal requirements of the system. In addition to this, a power-aware solution for scheduling is also implemented.

The real time application of temperature monitoring and control for safety critical application consists of 6 real time tasks which are essential for the application implementation. All the real time task information is available only in the server node. The server node runs the task allocation algorithm and finds an optimal assignment to reduce the number of processors and energy consumption. The client-server communication is established via. wireless mode using Zigbee. Once the tasks are assigned to an optimal number of processors, each client node schedules the assigned job so as to meet the functional and temporal constraints. The scheduling decisions are made based on RM policy. Upon task assignment, the energy consumed to schedule these tasks on individual processors is calculated, when operated at rated voltage and clock frequency. Also the energy consumption of the server node is computed. The cumulative energy consumption of the entire system including both server and client nodes is thus calculated. Finally, energy consumption reduction by DFS is implemented by running the possible tasks at lower frequencies without missing the task deadlines and

switching of the unused peripherals of the processors. The multiply and divide ratio of the Phase Locked Loop (PLL) control registers are configured to vary the switching frequency of the LPC2148 microcontroller. Thus, a power efficient embedded solution for multiprocessor system is implemented using ARM7 microcontrollers. The different steps involved in the application development and implementation include:

- Identification of the required number of tasks to implement the application
- Writing the task codes and finding the real time aspects of the tasks viz. execution time, deadline and time periods
- The execution time of tasks are computed by analytical approach by counting the number of clock cycles required to run the task code
- Execution time of a task code = No. clock cycles to execute the code * $\left\{\frac{1}{f}\right\}$
- After computing the task execution time, the deadline and time period of each tasks are appropriately fixed after understanding the application nature
- Development of task-processor assignment algorithm based on task dependency and precedence relations
- Development of scheduling policy based on RM
- Implementation of wireless communication between *Client* - *Server* nodes
- Entire system networking and synchronization

The hardware setup of the system is shown in Figure 2 and Figure 3.

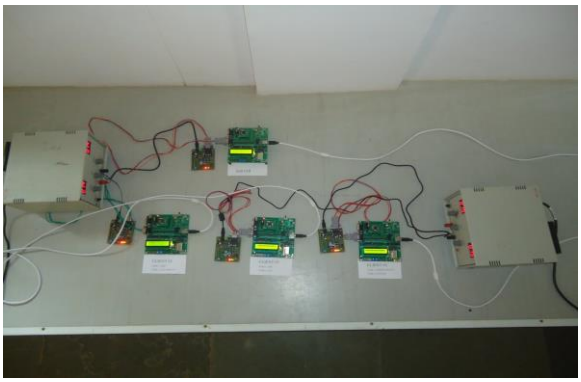

Figure 2: Hardware Setup

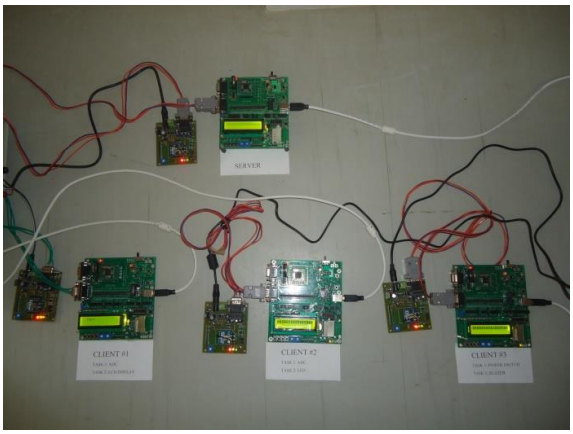

Figure 3: A close view of LPC2148 Microcontroller based Multiprocessor System

The different steps of application development are described in detail in the following sections.

6.2 Real Time Tasks to Implement the Application

To implement "*Power-aware embedded system for temperature monitoring and control in safety critical applications*", the different real time tasks identified are:

- Temperature sensing task to measure the ambient temperature, implemented using an Analog to Digital Converter (ADC-I).
- Motor control task to adjust the temperature of the room, implemented through a second ADC (ADC-II).
- LED ON/OFF task to depict the power availability
- LCD display task to show the temperature level
- Power switch ON/OFF to trigger a Buzzer
- Buzzer to generate a warning signal

The coding for each of these tasks are done using Embedded C and tested for functionality. Upon compilation, each task computation time is

calculated by analytical approach by counting the number of cycles. After getting the execution time and by understanding the application task nature, each real time task's deadline and time period are fixed. Thus, the attributes of the real time tasks are computed when the controller operates at the rated supply voltage of 3.3V and 60MHz. clock frequency as shown in Table II.

Table II. Real time task parameters

Task	Execution time (C_i)	Time period (T_i)
ADC-I	10.81msec.	25msec.
ADC-II	23.56msec.	50msec.
LED	6.4µsec.	30 µsec.
LCD	46.5µec.	70 µsec.
Power switch	3.5msec.	10 µsec.
Buzzer	28.12µsec.	50 µsec.

6.3 Communication

The server node communicates to the client nodes via wirelessly through Zigbee. The frame format is as shown below.

HEADER (1byte)	DATA (n Byte)	EOF (1 Byte)

The first field in message format is the HEADER field, which indicates the address/ID of the client node. It is 1byte long. The second is DATA field, consists of *1to n* bytes information regarding the assigned task. i.e. this field indicates the task numbers assigned to the processor, depending on the decision of task allocation algorithm . Third is the End Of Frame (EOF) packet. Upon the reception of a message from the server node, the client sends back an acknowledgement to indicate a successful/failure reception. In wireless communication, there are two feedback generation mechanisms: one is using *timers* and the other one is acknowledgement ***ACK*** *packets*. In the present system, ***ACK*** *packets* are used as feedback signals. A positive ***ACK*** is transmitted from the client to the server, when desired/correct packets are received by the client node otherwise a negative ***ACK.*** The server will continue retransmission of the packets, until it receives a positive ***ACK*** from the client node as in Figure 4.

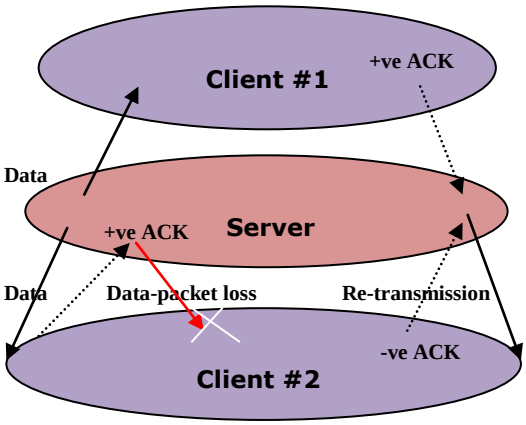

Figure 4: Sample data exchange and acknowledgement signals between two client nodes and server

7 Simulation Results

Performance evaluation is carried out using simulations run by C. The energy consumption of various processors in the system with varying voltage and frequency operation is computed and shown in Figure 5.

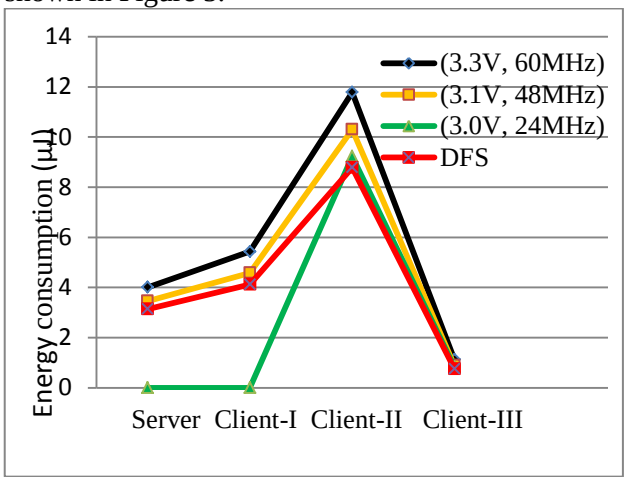

Figure 5: Variation in energy consumption with varying voltage and frequency operation

Energy saving obtained by SFS and DFS by simulation approach is computed and is tabulated in Table III.

Table III: Energy saving obtained by simulation

	Server	Client-I	Client-II	Client-III
(3.1V, 48MHz)	13.71%	15.65%	12.56%	21.24%
(3.0V, 24MHz)	NS	NS	21.64%	26.55%
(2.8V, 12MHz)	NS	NS	NS	NS
DFS	21.69%	24.12%	26.55%	32.74%

NS – Not Schedulable

8 Experimental Validation

A laboratory model of the multiprocessor system is implemented using four ARM7 LPC2148 microcontrollers. A shared memory multiprocessor system based on *Client-Server model* is implemented, where the server node acts as the shared memory and the clients communicate through the server. The objective of the work is to *develop a power efficient real time embedded control in a distributed multiprocessor system.* As mentioned earlier, six real time tasks are realized for the application and the task parameters are computed as in Table III. The task deadline is assumed equal to the task time period ($D_i = T_i$). The total task set utilization is calculated using equation 1 and is equal to 2.576. Therefore, a total of three processors are required to schedule the above six real time tasks. The task information is available in the server node. In the server node task allocation algorithm is executed and the resulting task allotment is shown in Table IV.

Table IV. Task Assignment

Client node	Tasks	Processor utilization
1	ADC-I, LCD display	0.979
2	ADC-II, LED task	0.685
3	Power switch, Buzzer	0.912

The different fields in the frame format for each of the client node are as shown in Table V and VI.

Table V: Task ID assigned to processors

Task Name	Task Address/ID
ADC-I	1
ADC-II	2
LED	3
LCD	4
Power switch	5
Buzzer	6

Table VI: Different fields in message frame format

Client node	EOF	Acknowledgement
1	$	X
2	*	Y
3	#	Z

Once the tasks are assigned to the processors, scheduling is done in the client nodes by RM policy. The energy consumed by the client and server nodes to allocate and schedule the above six real time tasks is computed using equation 5 and 6 and is shown in Table VII.

Table VII. Energy consumption of various processors @rated voltage and frequency (3.3V, 60MHz.)

Node	Energy consumed for scheduling the tasks once in the processor (µJ)
Server	5.86
Client #1	7.094
Client #2	15.39
Client #3	2.305

For conserving energy, while implementing the above application, the task instance execution speed is slowed down by operating at lower voltage and frequencies, as possible. The energy consumption reduction is achieved through two approaches SFS and DFS.

The processor when operated at a fixed operating point of lower voltage and frequency is SFS and changing the operating voltage and frequency at run time to facilitate different instances of the same task to run with different voltages and frequencies are DFS. Tasks are run at reduced voltage and frequencies by both SFS and DFS techniques, without compromising the temporal constraints.

9 Experimental Results

The energy consumption of the various processors while implementing the application of "Power-aware embedded system for temperature monitoring and control in safety critical applications", without any techniques (i.e., operating at rated voltage and frequency) and with SFS and DFS and by configuring the registers of LPC2148 microcontroller to cut off the clock frequency of unused peripherals are depicted in Figure 6 to 9.

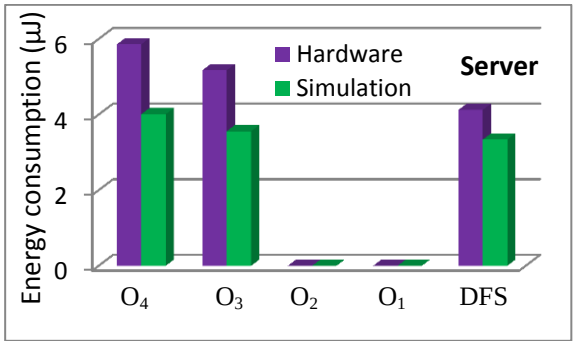

Figure 6: Energy consumption of Server node

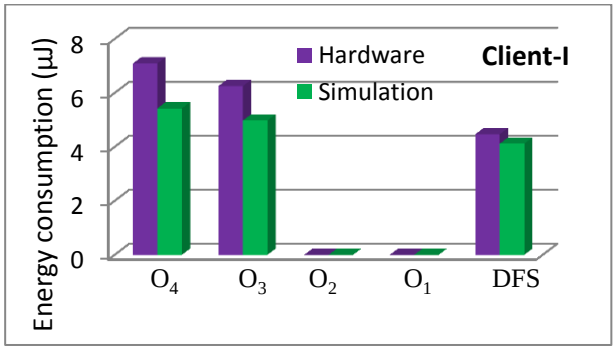

Figure 7: Energy consumption of Client-I node

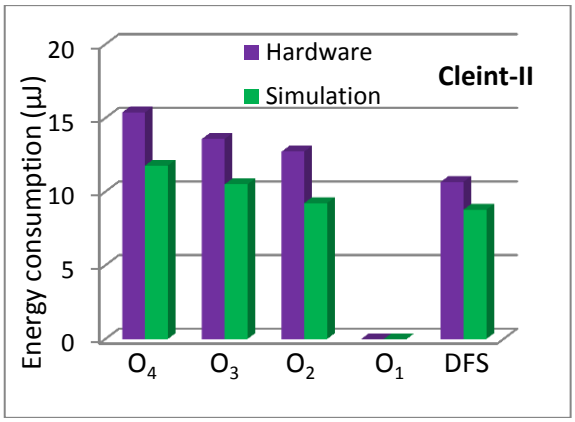

Figure 8: Energy consumption of Client-II node

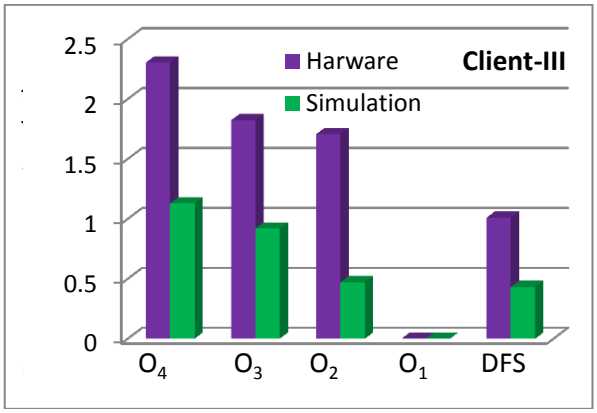

Figure 9: Energy consumption of Client-III node

The energy saving obtained by SFS and DFS by hardware experiment is computed using equation 9 and 10 and tabulated in Table VIII.

Table VIII: Energy saving obtained by hardware

	Server	Client-I	Client-II	Client-III
(3.1V, 48MHz)	11.76%	13.36%	11.71%	20.91%
(3.0V, 24MHz)	NS	NS	17.32%	25.94%
(2.8V, 12MHz)	NS	NS	NS	NS
DFS	19.35%	22.88%	24.15%	26.68%

NS – Not Schedulable

From the above results it is verified that energy consumption of various processors in the

multiprocessor system can be reduced to a considerable amount by switching the operating voltage and frequency of the processor to a lower value, without compromising the temporal constraints and by switching off the unused peripherals of the processor. Energy saving in the order of µJ is also crucial for a single processor core, as big and complex networked embedded systems consist of thousands of processors/ controllers. Thus, the cumulative energy saving of the entire system will be definitely a considerable amount and is requisite for embedded systems, especially portable embedded devices.

From Table III and Table VIII, it could be seen that there is a difference in the amount of energy saving obtained by software simulation and hardware validation. A graph showing the comparison of both approaches is depicted in Figure 10 to 13. This difference is due to the absence of environmental factors like: temperature effects, overhead associated with the latency, communication and synchronization and non inclusion of enormous peripherals in the controller board and many other physical parameters in the simulation analysis.

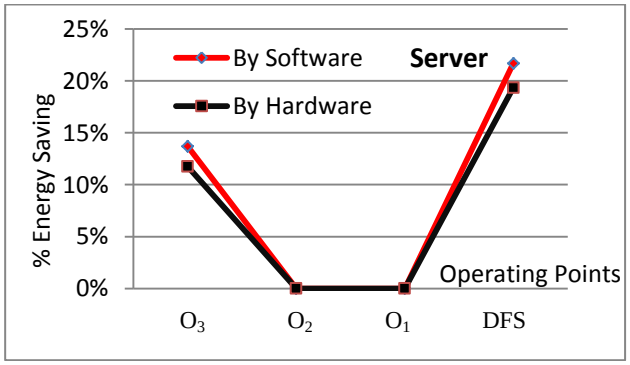

Figure 10: Comparison of energy saving in Server by software simulation and hardware validation

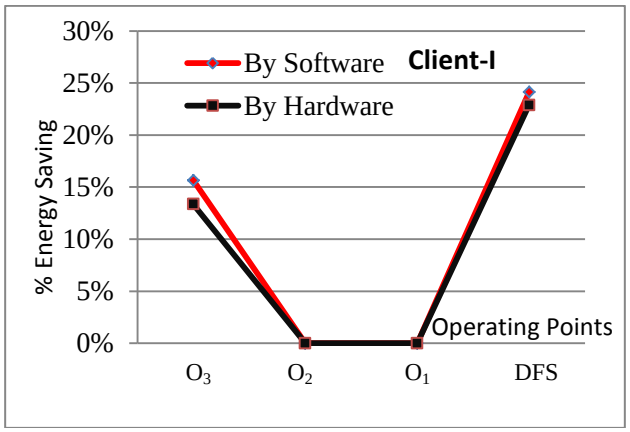

Figure 11: Comparison of energy saving in Client-I by software simulation and hardware validation

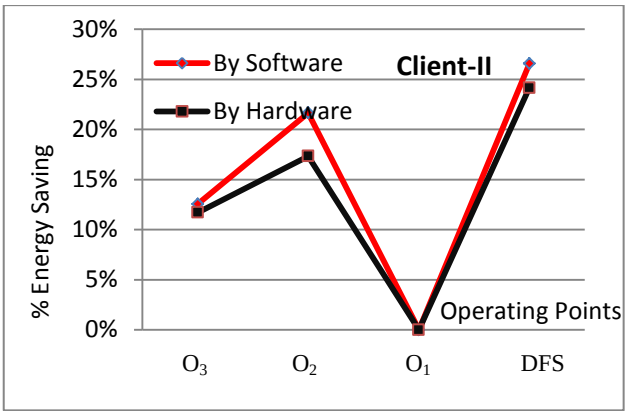

Figure 12: Comparison of energy saving in Client-II by software simulation and hardware validation

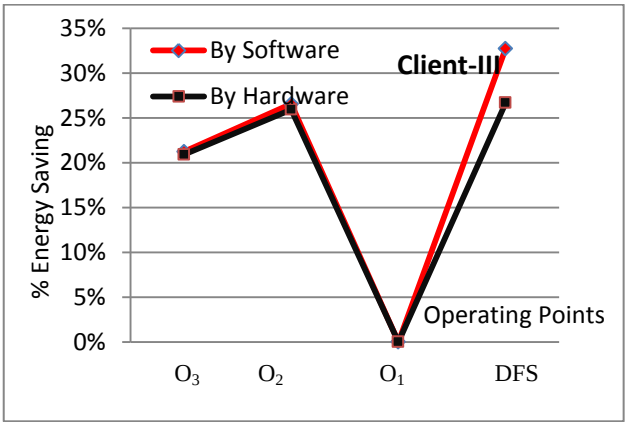

Figure 13: Comparison of energy saving in Client-III by software simulation and hardware validation

10 Conclusion

The widespread use of multiprocessor system for application development claims for the need of energy constrained system. The energy issues being the most crucial one, demand more attention while implementing portable systems. This paper presents a software simulation and an experimental validation of energy consumption reduction in a distributed networked multiprocessor system. The chosen application of "*Power-aware embedded system for temperature monitoring and control in safety critical applications*" is implemented in hardware using four ARM7 LPC2148 microcontrollers. A shared memory architecture using *Client-Server model* is followed for the implementation. Energy consumption reduction is achieved by the use of energy aware task-processor allocation algorithm for multiprocessor systems based on task dependencies and precedence relations and by dynamic voltage and frequency scaling technique along with cutting off the supply to the unused peripherals of the controller by

configuring the registers of the LPC2148 micro controller. The results show an evidence of substantial energy saving viable by DVFS technique in distributed networked multi processor systems.

On-going efforts include optimization of multiprocessor system for energy minimization by different approaches like: GA, PSO etc. Also, efforts to solve multi-objective function optimization will be taken up in future.

References:

[1] K. Roy, M.C. Johnson "Software Design for Low Power," *NATO Advanced Study Institute on Low Power Design in Deep Submicron Electronics,* Aug. 1996. http://www.ece.purdue.edu/~ vlsi/papers/mark/lpsw chap.ps

[2] R. Graybill, R. Melhem (Eds.), "Power Aware Computing", *Kluwer Academic/Plenum Publishers*, ISBN 0-306-46786-0, May 2002.

[3] Pedram, M., "Power optimization and management in embedded systems," *Design Automation Conference, 2001.* Proceedings of the ASP-DAC 2001. Asia and South Pacific , pp.239-244, 2001. doi: 10.1109/ASPDAC.2001.913312

[4] T. Simunic, L. Benini, G. De Micheli, M. Hans, "Energy Efficient Design of Battery-Powered Embedded Systems," *International Symposium on Low Power Electronics and Design*, pp.212-217, 1999.

[5] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step toward software power minimization," *IEEE Trans. VLSI Syst.,* vol.2, pp.437-445, Dec. 1994.

[6] Sung I. Park, "The Design of Power Aware Embedded Systems," PhD Thesis, University of California, Los Angeleles, 2003.

[7] Gary K. Yeap, "*Practical Low Power Digital VLSI Design*," Springer, London, 1998.

[8] A. S. Pillai, T.B. Isha, "Factors Causing Power Consumption in an Embedded Processor - A Study", *Int. J. of Application or Innovation in Engineering & Management,* Vol. 2, Issue. 7, pp. 300-306, July 2013.

[9] Chinnery D., Keutzer K., "Closing the Power Gap between ASIC & Custom Tools and Techniques for Low Power Design," *Springer*, 2007.

[10] Schubert, S., Kostic, D., Zwaenepoel, W., Shin, K.G., "Profiling Software for Energy Consumption," *IEEE International Conference on Green Computing and Communications (GreenCom)*, pp.515-522, Nov. 2012.

[11] L. Wehmeyer, M. K. Jain, S. Steinke, P. Marwedel, and M. Balakrishnan, "Analysis of the Influence of Register File Size on Energy Consumption Code Size, and Execution Time," *IEEE transactions on Computer Aided Design of Integrated circuits and systems,* Vol.20, No.11, Nov. 2001.

[12] Ravindra Jejurikar, Rajesh Gupta, "Dynamic Voltage Scaling for System-wide Energy Minimization in Real-Time Embedded Systems," *Proceedings of the 2004 International Symposium on Low Power Electronics and Design,*Vol.11, No.11, pp.78–81, Aug. 2004.

[13] R. Narayanan, A.S. Pillai, "Measurement of Current and Power Consumption in the Order of Milli-Watt Resolution in Processor Core Level, "*Proceedings of the Int. Conference on Circuit, Power and Computing Technologies (ICCPCT),* Noorul Islam College of Engineering, Kanyakumari, Mar. 2014.

[14] Russ Joseph and Margaret Martonosi, "Runtime Power Estimation in High Performance Microprocessors," *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, Vol. 11, No.11, pp.35–140, Aug. 2001.

[15] Jeffry T. Russell, Margarida F Jacome, "Soft ware Power Estimation and Optimization for High Performance, 32-bit Embedded Processors," *IEEE Proceedings of ICCD'98,* Oct.1999.

[16] Theodore Laopoulos, Periklis Neofotistos, C. A. Kosmatopoulos, and Spiridon Nikolaidis, "Measurement of Current Variations for the Estimation of Software Related Power Consumption," *IEEE Transactions on Instrumentation and Measurement,* Vol.52, No.4, Aug. 2003.

[17] P. Pillai, K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems," *IEEE Symposium on Operating Systems Principles,* pp.89–102, 2001.

[18] K. Thanushkodi, K. Deeba, "On Performance Analysis of Hybrid Intelligent Algorithms with GA, PSO for multiprocessor Job Allocation", *WSEAS Transactions on Computers*, Vol.11, No.1, pp.131-147, 2012.

[19] M. Bambagini, G. Buttazo, and S. Hendseth, "Exploiting Uni-Processor Schedulability Analysis for Partitioned Task Allocation o Multi-Processors with Precedence Constraints," *RTSOPS* 2012, p. 17, 2012.