IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 2 (Mar. - Apr. 2013), PP 50-59
www.iosrjournals.org

Section Code Task Model for Heterogenous Processor In Real Time System

[1]Jestin Rajamony, [2]S. Gladson Oliver
[1,2]*Department of Information Technology,CSI Institute of Technology, Thovalai.*

Abstract: *Even thou there are many schedulers available, almost every scheduler are not giving the maximum performance when used in real time system when there is an overload. To overcome this, an innovative approach has been used in this paper which gives the maximum utility when compared to the other related schedulers available. The program is split and grouped together in the initial stage, and a section code is tested for the entire cluster of code and using the feedback loop the miss ratio is identified. The major work of the paper is selecting the desired algorithm and fixing to the desired core in the heterogeneous multicore processor. So if a deadline is not met by any one of the core, then the scheduler submits it to the next core of high or low end speed using the feedback control framework This gives a wider spectrum when double check is done for each code before it is used in the real time system avoiding the miss ratio and obtaining the maximum utility. The paper uses operating system scheduler algorithm over the control system methodology for its design and scheduling.*

Key-Words: - *Section code, loop cloud, MAPS.*

I. Motivation & Introduction

From the single core to dual core and today multicore CPU have become a commodity items in major researches. The expectation in the next decades is that the number of cores in a CPU will increase to as many as hundreds. All cores in a homogenous multiprocessor have same performance with a same instruction set, but however they differ in terms of performance characteristics with changes in clock frequency, cache size etc. Recent researches have advocated the need for a class of heterogeneous multicore processors. Here even with same instruction set, there is a possibility of lot of changes in the performance characteristics. The architecture available for performance symmetric (homogenous) multi core processor is effective compared to the performance asymmetric (heterogeneous) multicore processor [1]. For example, when workload characteristics are matched to heterogeneous cores, performance gains up to 40% are observed.

Process alternate between two states like CPU burst and Input Output burst and so on. The duration of the CPU burst has to be measured. Whenever the CPU becomes idle the operating must select one of the processes in the ready queues to be executed. The selection process is carried out by the short term scheduler or the CPU scheduler. Many schedulers are available like EDF, AED, spring scheduler etc that are designed for the real time system. But when the system gets overloaded then the Earliest Deadline First (EDF) algorithm degrades rapidly than any other schedulers. This is due to the fact that it gives the highest priority to transaction that is close to the deadline misses. So Adaptive earliest deadline (AED), is used that detects overload condition and then modifies transaction with priority [2] using the feedback control mechanism. In an unpredictable environment, it is very difficult for the real time system designed to meet the required deadline. Spring scheduling algorithm [3] using the online admission control algorithm can guarantee partly in resource insufficient environment that is unpredictable. There are many other scheduling algorithms that support the real time scheduling environment with sufficient resources. Despite many real time algorithms available, none of the algorithm supports fully the real world problems. And in the heterogeneous multicore processor there is no scheduler that makes it to be maximum utilized and avoiding the overload.

The research of this paper is towards soft real time application in the web server of online banking system. It uses the feedback control theory and its framework in an unpredictable real time system. Failure to meet these leads to deadly problems and finally loss of customer, financial damages, reputation etc. Almost every research papers related to this is concerned with deadline misses and overload, but they are not fulfilling even the deadline misses and the overload problems. Here a priority event triggered sampling has been proposed with an idea of sampling, communicating and controlling only if something significant has occurred in the system. The effect of control system performance degrades when a periodic task is implemented. This generates the problem in sampling and produces latency jitter. Two major problems have been identified in the control application that reduces the performance of the system.

- ❖Allocation of resources to control applications in order to maximize control performance.
- ❖Novel & unknown computational models for implementing control algorithms using real time technology.

The major work of the paper is selecting the desired algorithm and fixing to the desired core in the heterogeneous multicore processor. So if a deadline is not met by any one of the core, then the scheduler submits it to the next core of high or low end speed using the feedback control framework. The aim of this paper is based on the deadline based metrics from the worst case to the best case, where there is a major shift of total load in the system. The system CPU should be fully utilized when there is job waiting in the queue. Also the processor utilization can be dynamically obtained by assigning priorities on the basis of the current deadlines [4].

Most of today's job works with the threads, where a processor holds the specified resources. The thread are given a specified percentage of CPU cycle over a period of time and uses a feedback back scheduler to assign automatically both proportion and periods . Based on this a QoS optimization algorithm and a communication subsystem architecture was developed. The actuator depends on the QoS algorithm that meets both predictability and graceful degradation requirements. The QoS negotiation guarantees the required QoS and rejects the service request, by outperforming binary admission control schemes. The scheduler architecture includes the following elements

- Feed back control schedulers architecture that maps the feedback control structure.
- Multicore asymmetric processor with a loop cloud with a section code identifier from the task model.
- A set of performance metrics and a speed analyzer for the digital controller.

In contrast, the frame work enables system designers to systematically design adaptive real time systems with established analytical methods to achieve desired performance guarantee in an unpredictable environment.

II. Modelling Multicore Heterogeneous Processor With Control System In Real Time Environment.

The control theory methodology is used to establish an analytical model to approximate the controlled system in the closed loop architecture with the multicore asymmetric processor. The control input to the controlled system is the change in the total estimated utilization. The output of the controlled system includes the controlled variables, miss ratio M (k) and utilization U (k).

According to the control input $D_B(k)$ the QoS actuator changes the total estimated utilization B (k+1) for every sampling period at every sampling instant k,

$$A(k) = G_A B(k) \quad (1)$$

Utilization ratio in the sense of this paper means that the workload extension in terms of the total requested utilization of the unknown task. The section code is taken as a sample from a group of cluster of task and executed and made to be a known task by the first analysis, which is the key challenging goal of the paper. Then using the MAPS the speed of the processor and the core utilization is found out from the loop cloud using the priority event triggering with the closed loop system. The speed is monitored by the speed analyzer [5] that finds the speed of the processor and the digital controller.

2.1 Tasks guided to the processor core assignment

Every program exhibits phase behaviour while the job is in execution as it goes through various phases of execution [6]. These phases show similar runtime characteristics compared to the other phases of execution. Therefore it better to group the similar ones into the code section [7]. The program is divided into different sections and the section code is classified into one or more phases. Theses sections are clustered into similar groups. These groups are then given an identity for the priority since it is a real time system the task with more priority is considered with more importance. Since each process has to link to another process a flow of control is identified from one phase to a different phase type. Provide an identity mark with the phase known as the phase mark that includes the current section, dynamic performance analysis and the core switching decision. A small sample of this section is executed from the cluster which is a sole representative of the entire section.

2.2 Control Flow determination

The aim of this area is to determine the points in the control flow where the phase is likely to change from one state to other. To do that first the entire program is divided into procedures (P) and each procedure $p \varepsilon P$ into a basic block. The blocks that are related are then grouped together into a cluster. The basic block (B) refers to that it has a section of code that has one entry point and one exit point with no jumps in between them. From these, attributed intra-procedural control-flow graphs for procedures in the program are created. Using the standard algorithms, the attributed control flow graph of a procedure, is then partitioned into a unique set of intervals (I).

$$(i(\eta) \varepsilon I) (\eta \varepsilon N) \quad (2)$$

where η is the entry node in a closed path and i is the interval.

Once the phase transition analysis is complete then insert the phase marks in the binary to produce phase information and the dynamic code fragments. Consider a section of basic block and place in it a set of related instructions. The advantage of using basic block is that execution of a single instruction in the block implies that all instruction in the block gets executed. Also the number of switching from block to another is reduced.
When the control input $D_B(k) \neq 0$ the control variable is constant. So the utilization U (k) is outside the saturation zone and equals A(k), when the multicore heterogeneous processor is not utilized or underutilized ($A(k) \leq 1$) and U(k) saturates at 1 because it can never exceed 100% when the CPU is overloaded. Each core in the multicore heterogeneous processor releases an output in cycles of seconds. This becomes a cloud of signal in the output of the processor. It takes a lot of time to review each loop by the closed loop scheduler and give the output. So a loop clouding is used where a priority event triggering is been taken.

2.3 Priority event triggered over a shared network loop.

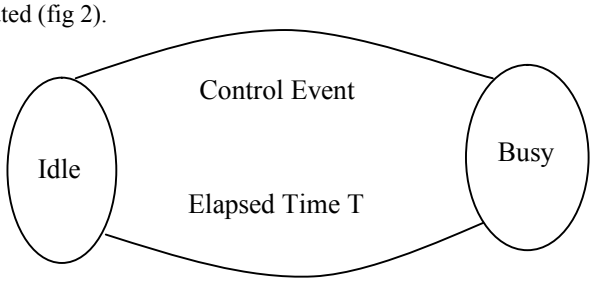
When N control loops are closed over for the multi-core heterogeneous process, where each core in the processor i ε 1…N is given by the first order stochastic differential equation

$$dx_i(t) = a_i x_i(t)dt + u_i(t)dt + \sigma dw_i(t), \qquad x_i(0) = 0 \quad (3)$$

Where x_i is the state, a_i is the process pole, u_i is the control signal, w_i is the Wiener process with unit incremental variance and $\sigma_i > o$ is the intensity of noise. The outputs of the processors are checked by taking a sample at certain discrete time interval $\{ t_{i,k} \}_{k=0}^{\infty}$.

$$x_{i,k} = x_i(t_{i,k}) \quad (4)$$

Depending on the medium access scheme the sampling can be a event triggered, so this becomes like a virtual network that is however a shared form of resource, so that only one control loop may access at a time.
If two or more processor cores in the multicore heterogeneous process attempt to transmit data at the same time, a priority event triggering is used to gain the access of the network loop. The other nodes are kept for *s* seconds and then simply discarded. Once an access is gained, the network loop stays occupied for T seconds, corresponding to the transmission delay from the processor core to the actuator. During this interval, no new control events may be generated (fig 2).



(Fig 2)

The controller for each loop may be collocated with the processor core or actuator, where the control signal delay or network loop delay is assumed to be unknown, so that is rectified in the priority event triggering.

2.4 Priority event triggering.

Control algorithms can be implemented considering the periodic actuation. Digital control system works with sampling period, control execution time, jitter, and complexity of the execution time. Instead of computing the utilization with the processor state z_k, which is time consuming and outdated, an estimated utilization B(k) is used. Therefore the process utilization $u(k)$ is computed with the estimated utilization at time t_{k+f} i.e. z_{k+f}.
Therefore,

$$z_{k+f} = (t_k \in (t_{k-1+f}, t_{k+r}))\ z_k \quad (5)$$

where x_k is the input of the processor, $_k$ identifies the job within a sequence of the jobs and ϕ & Γ are the input matrices. With this the controller will compute the change in utilization $D_U(k)$,

$$D_U(k) = G\, z_{k+f} \qquad (6)$$

where G is the controller gain i.e. tunable parameter, $D_U(k)$ is the control input and z_{k+f} is the error ratio identified. With this there is no delay in the closed loop model.

The goal of the QoS actuator is to enforce a new total requested utilization $B(k+1) = B(k) + D_B(k)$. So the control
signal $D_B(k+1)$ is generated by the QoS actuator q is given by the step signal,

$$u_i(t) = \sum_{k=0}^{\infty} \delta(t - t_{i,k} - T)\, x_{i,k} \qquad (7)$$

Under the utilization constraint of *B (k+1)*, the QoS actuator calls a QoS optimization algorithm to give the maximum value. The QoS actuator is invoked at the execution of each task from the loop cloud using priority event triggering mentioned above. Previous studies show the system without arriving time.

III. Scheduling Algorithms Design.

Based on the control theory and the mathematical analysis the design of the closed loop system is applied for the real time system. The controller computes the control input $D_U(k)$, with the change in the total estimated requested utilization based on the miss ratio error and the CPU utilization error at each sampling function *k*.

For the closed loop architecture, the simple proportional P control function is used to compute the input. Therefore the P control function is,

$$D_B(k) = K_p\, E(k) \qquad (8)$$

where *E (k)* are the miss ratio reference and the CPU utilization reference.

The key idea of this design is to assign a section of code to the core, by which the in-depth use of the code in section can be known. Also the resources required by the code and the resources available for the request should be known. Since it is a real time system in an unpredictable environment, the arrival tasks need is not known priory. Therefore the arriving program is split into procedures and each procedure in to basic blocks; so that a basic block will have a section of code that have one entry point and one exit point. Then the blocks that are similar are connected together as shown in fig 3. A small code is also included in the entry point and another code at the exit point making it connect to the right code in the cluster of blocks.

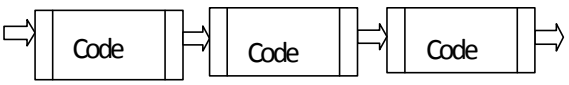


Fig 3

The blocks are connected by means of the principles of the graph.

$$G = (V, E) \qquad (9)$$

where E represents the edges, *V* represents set of blocks, and each block is represented as *v* and each block is linked together by lines or connecting arcs. An edge is represented by the pair of vertices (*u, v*).

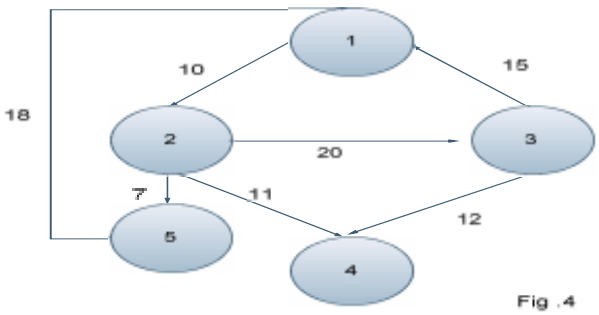


Fig .4

Fig 4 graph (control flow) that marks the vertices and the edges.

Fig 4 shows the graph (control flow) that marks the vertices and the edges. Since the blocks are all having instructions they have to be linked to the respective links. This is a very tedious process when considering with the automatic dissection of the program into different blocks. Also since the arrival job is unpredictable in the real time environment, it is also very difficult to group the task. So they are enclosed in the graph as node with edges and the connection direction is also mentioned. From the digraph (fig 4),

$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{(1,2), (2,3), (2,4), (2,5), (3,1), (3,5), (4,5)\} \quad (10)$$

At the edges the graph (control flow) is mentioned with the size of the task in the block. These sizes may be same or different but here the grouping of the block is by means of similar task. A matrix representation of the control flow depends on the ordering of the nodes. For the weighted graph, the elements of the adjacency matrix M of the order of $n \times n$ is then,

$M[i][j] = W_{ij}$ If $(V_i, V_j) \in E$ and $W_{ij} = (V_i, V_j)$ (11)

= 0, otherwise.

The elements in this adjacency matrix are a Boolean number 0 or 1. The adjacency matrix for the above digraph (fig 4) is given by,

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	1	1
3	1	0	0	0	1
4	1	0	0	0	0
5	0	0	0	0	0

The connection of the blocks is done by using an array of link list which is called by adjacency list. The adjacency list represents by means of adjacent vertex number and the weight (i.e. size of the instruction) of the edges. Each block has a phase mark. A phase mark is a small code fragment that indicated the phase type for the current section. Each block has three sections, an entry section, instruction section and the exit section. The connection established by the core switching technique is mentioned below in the fig 5.

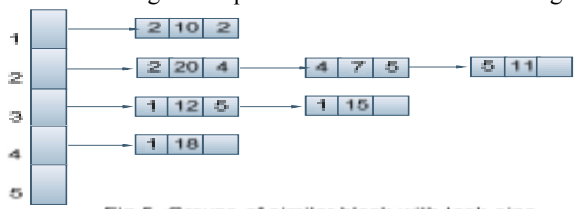


Fig 5. Groups of similar block with task size

Once the blocks are assigned in the scheduler it gives the number of blocks B_n in the group. So in the closed loop scheduler if block is less than the number of block, the controller indicates the error in the control variable by means of miss ratio M (k). Then the manipulated variable is corrected with the missing block M_s to that of the reference utilization U_s, that gives the actual utilization to the scheduler.

The problem is that allocation of the resource right to the respective core of the processor. The problem is fixed here by the allocation of the block to the core of the processor with the shortest path first method. The path between two vertices *u* and *v* of a graph *Ġ* with minimum number of edges in the path is the shortest path. For a weighted graph it is the path between *u* and *v* for which the sum of weights in the path is the minimum.

Consider a path $p = v_0, v_1, v_3, v_4 \ldots v_k$ i.e. $p = \{(v_0, v_1), (v_1, v_2), (v_2, v_3), \ldots (v_{k-1}, v_k)\}$

Then the path can be found as

$$W t(p) = \sum_{i=0}^{k} W t(v_{i-1}, v_i) \quad (12)$$

Based on the Dijkstra algorithm and with the equation 12 the shortest path is found to link the blocks in the group. Each program is divided into procedures and the procedures are divided and classified into similar blocks. So it resembles as a tree in the forest as shown in the figure 6.

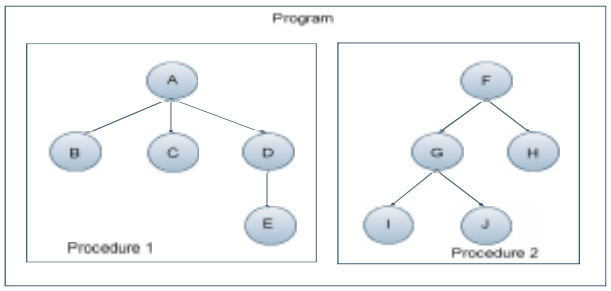


Fig 6 Forest(Program) of Trees(Procedures) with leaves (Blocks)

Fig 6: Forest (Program) of Trees(Procedures) with leaves (Blocks)

A, B, C, D and E are all blocks related to a single procedure. These individual blocks are split into sections and then the similar blocks are grouped together. All the procedures are then grouped to form a program. The binary representations of the sections of the block are classified into three parts, the entry section i.e. current section, the code section i.e. the dynamic performance area and the exit section i.e. core switching decision area.
The binary representation of the block is shown in the fig 7.

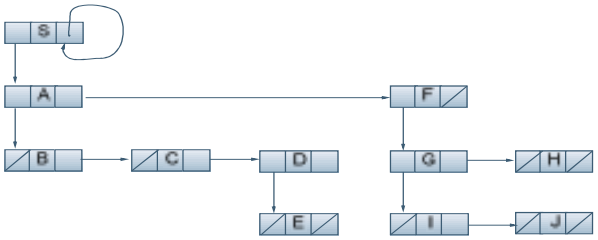


Fig 7. Binary Representation of the blocks

The conversion from the forest to tree then to the binary representation is a natural correspondence. The two procedures can be specified in the following manner to represent their levels (Table 1).

Procedure 1		Procedure 2	
Levels	Blocks	Levels	Blocks
1	A	1	F
2	B	2	G
2	C	3	I
2	D	3	J
3	E	2	H

Table 1

Using a simple algorithm the conversion from one level to the next level can be implemented for the notion of the natural correspondence.
Once the section of the block of code is executed, the control signal is in the loop cloud. The section of code comes from different cores *l* of the heterogeneous processor, so there would be a maximum of *l* numbers of control signals in a single periodic cycle. Since it is real time system, it is very necessary to select from this cloud the very important signal. So in the block there is a priority assignment that alerts the priority event trigger. From this single section of block execution the other related block can be easily executed based on the rectification made in the control variable. So from the clouds of signal an event is triggered [8][9] based on the priority the signal goes into the loop leaving the other signals for T s.

```
while (Ev = E(t)) {
        if (Mg > Th) then
            {
              PriorityEventTrigger Method();
              LoopCloud = T;
            }
        Endif
    }
PriorityEventTrigger Method(){
   switch (Pe.Ie) {
case 1,1 : EM = Ms − M(k);
          EU = Us − U(k);
          B(k+1) = B(k) + DB(k);
case 1,2 : Dc(k+1) = B(k+1) ÷ DB(k);
case 2 : Not valid
}
```

Algorithm 1

A section of the algorithm used in the experiment is been given in the algorithm 1, where event is E_v and the triggered event at a cycle is E(t), M_g is the magnitude , *Th* is the threshold, *T* is the waiting seconds, E_M is the miss ratio error, M_s is the desired miss ratio, *M(k)* is the deadline miss ratio, E_U is the CPU utilization error, U_S is the Desired *CPU* Utilization, *U(k)*is the CPU utilization, *B(k+1)* is the total estimated utilization, *B(k)* is the estimated utilization ,$D_B(k)$ is change in control input, and $D_c(k+1)$ is the derived utilization ratio. The blocks priority and level is checked first and during the initial section of code in the block the total estimated utilization is found out. Next during the second execution the block the derived utilization is obtain that is fixed for the rest of the related blocks in the group, with this the unpredictable task is made to be a predictable task in the real time environment.

IV. Experimental Setup.

The first step for the experiment is the heterogeneous multicore asymmetric closed loop system processor simulator for the real time systems. This is done by evaluation of the hardware setup and the software components.

4.1 Hardware setup

Here the experimental setup consists of a 4 cores setup with the asymmetric multicore processor. The clock frequencies of the cores are set at 2.4 GHz, 2.2 Ghz, 1.8 Ghz and 1.6 Ghz respectively. The complete set up is set on Intel dual core processor on two systems for the experiment purpose. Since the experiment is tried on the simulator the number of system is made to two.

4.2 Software setup

The experiment is using the kernel of the simulator true time [10]. There can be two kernels in a system with the clock frequency of 2.4 GHz and 1.8 Ghz and in the other system the kernel clock frequency has been set to 2.2 GHz and 1.6 Ghz.

4.3 HMP Simulator

The simulator is divided into two sections of research work, one section deals with the operating system scheduler [11] and the other section is towards the control system for the fine tuning. Since the signal output is from different cores of the processor there form a number of loops, called loop cloud. Here in the loop cloud only one of control signal may be accessed at a time. The loop cloud takes the signal by priority event triggering. Event triggering occurs at every h s i.e. at each cycle per second set and selects one of the triggered event according to their priority. This occurs if the magnitude is greater than the threshold then the priority set is chosen. A speed analyzer is also used to check the scheduled task parameters interaction and control loop performance that matches with that of the digital controller.

V. Simulation Result

5.1Comparison with existing works

The simulated results are compared with the existing works and their evaluation includes the naif task model [12], one-sample task model [13], switching task model [14], the split task model [15] and one shot model. Several works have already been done for the maximum utilization of the CPU. Most papers are using feedback loop for taking the controlled variable and trying to make the manipulated variable and again sending it back to the CPU. This extracts extra time for again reengineering the whole set up. In theses papers the works were towards the improvement of the digital control system performance [16].

All these works are directly using the control system framework and the operating system is not much covered. Here the paper is on the operating system scheduler where the program it self is split with an algorithm and made ready with full details for the execution. The control system theory is been used in the hardware computing of the control signal. This gives a more efficient way of utilizing the cores of the processor without any misses.

No unified framework exists to the date for designing a real time system that has full performance for the desired dynamic responses. But here the output is monitored and then checked for its status if the status is ok then it is passed out else it is sent for reengineering. The key advantage of this paper is to take a section of related code and execute it then the entire blocks in the group are executed. So this gives the maximum utilities of the cores in the CPU.

5.2 Work Load

In the online banking web server the workload is periodically checked for any recent updates in the bank. Also the queries related to the transaction are been asked a periodically. So it is not clear at what particular circumstance a particular instant will occur. And failure to meet this will lead to financial loss, loss of customer, spoilage of reputation and finally some times the whole mission loss. Also another key feature is timing constraint to avoid all the losses for the online banking area.

Task Set	Composition	Basic Scheduling Policy
Periodic/A	Periodic 70%	Extended Deadline Monotonic
Periodic	A Periodic 30 %	Earliest Deadline First

Table 2: Testing Configuration

In the online banking queries will be less when compared to the regular transaction. So the periodic composition is to be more when compared to aperiodic composition (Table 2). Since it is real time application it is necessary to have the deadline to be adjusted dynamically also to a limit.
For the analysis of the block that is in the loop cloud the different model with the specific timing for the design of the controller are evaluated as shown in table 3.

Task Models	Sampling period (ms)	Timing Delay (ms)
Naif	20	6
One Sample	20	20
Switching Controller	22	8
Spilt	20	2
One Shot	20	2
Priority Event Trigger	20	0

Table 3 Comparison

Here the work load is assigned for the priority event trigger with the sampling period of 20 ms and that of the timing delay 0 ms since one of the prior event will trigger at the loop cloud so there cannot be any time delay. The timing of the task in the computer control is mainly important for the simulation. The tasks are associated with the events that include the timer event, termination of the block, the data that are ready to be executed etc. The tasks usually share the data but can be executed in different core of the processor. The main problem is how the blocks will be classified and then grouped together which is the major work of this paper. A timing assumption is made for the control loop system.

1. Sampling is performed by the time period Te of the ideal system with some variations T_e.
2. The actuation is performed instantly when the control signal $u(k) \leftarrow T_e$ is received.
3. With the correctly identified blocks with out the disturbances the scheduler algorithm is computed.
4. If there is an internal or external disturbance then the sampling period can be estimated as :

$$T_e \in \left[T_e^l , T_e^n \right] \qquad (13)$$

where T_e^n is the time period for the ideal control systems.

5.3 Task Model

Each task has several QoS levels. Here the total task T_i has N>3 QoS levels j with the following attributes (0<j<N-1). Three QoS levels (0-rejection, 1-acceptance and 2-Wait) are taken into consideration. The key feature of the task model is that it makes the task to be known in unpredictable environment where tasks actual CPU utilization is time varying and unknown to the scheduler.

The single task T_i is set inside the single basic block B_i with certain attributes that are relatively common: A block identification number, a list of execution code segments, a time period *h*, a relative deadline R_D, a release time and the remaining approximate execution time $RE_i[j]$. When a section of the code is executed the release time and the remaining execution time can be determined. The time period and the relative dead line can be change after the manipulated variable from the controller to the actuator for the basic scheduler. So the blocks can now be rearranged and assigned together into a similar group.

5.4 Simulation approach

The Simulink and the Matlab platform with the True time kernel [17] was used for the simulation process to implement four cores of the processor with different clock frequency.
The aim of the simulation work is to make the core of the processor to be maximum utilized U (k) = 1 and to avoid the miss ratio M (k) =0. At any time period either one of the control variable will be active. Since each core here is considered as a kernel the clock frequency of the kernel is set to be different as 2.4 GHz, 1.8 GHz, 2.2 GHz, and 1.6 GHz.

The first step is comparison of code task model that are used in real time and models currently used in real time control system. The simulation is done with true time scheduler and Matlab tools. The tasks are grouped as T_i, and the period of the task is h, D is the relative deadline. With the square wave the input is set and the sampling period and the time delay are determined with the controller gain. The simulation evaluates the tracking of the set point change from 0 to 5v at the time interval of every 5ms.

5.5 Experimental Results

Nearly 25 jobs are send in the given arrival order, where the jobs of related works are split up like computation, iteration, external execution etc. A section of the code from the task in the group is send to one of the idle core of the multi core heterogeneous processor. This initiates the speed of execution of the task in the

group. So the jobs in the cluster are to be in the same format with almost the same execution speed. The simulation result is based on the processor for the real time control of the online banking web server system. The delay time and the computation time are also been considered, so the discrete time is set to 0 whenever it is required.

With the feedback loop, the sample of the section code is sent for the total estimated utilization where the controller calls the optimized algorithm and assigns the QoS Level.

From the result in the figure 10, the high level line means jobs in the execution, the middle level line shows the job is pre-empted due to the execution of the jobs of T_1. The job settles down after 0.07ms and compared to the reference line.

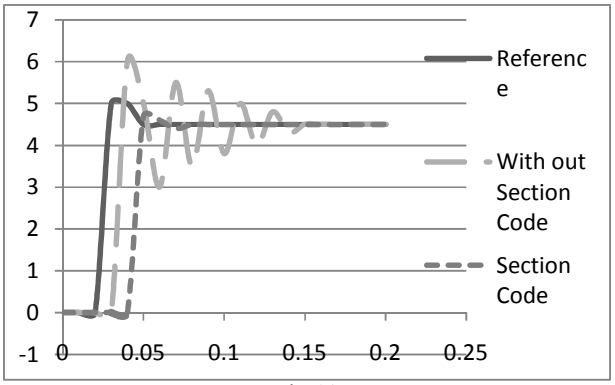
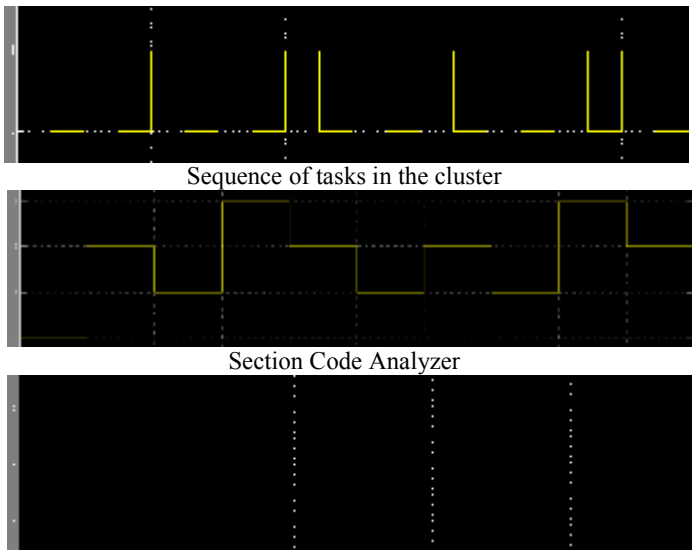


Fig 10

From the output of the above figure, the section code model rises from the 0 to 5v and remains stable with the reference input. With the pulse generator a disturbance is created that creates a overloading and neutralizes after 0.08ms. So during the next cycle the overloading is detected and then rectified. Therefore the lower the curve, the performance is better. So the problems in the other models, like jitters, gain degradation, overloading is completely removed since by the first check itself the problems are identified and then analyzed.

The simulation process was conducted over matlab using true time kernel for the implementation in the real time multicore heterogeneous processor. Four kernels are set at different execution speed and a feedback loop is set to find out the miss ratio. All the four kernels are started at the same time and during the execution period certain task is affected by an external disturbance. This disturbance is generated by the pulse generator. The sampling period ranges from 20 to 40ms, delay time is 10ms and the simulation time is 3s.



Sequence of tasks in the cluster

Section Code Analyzer

Fig 11 Experimental result

The performance output is given for the section code model as shown in figure 11. Without detection, the signal goes out in a disorderly way. But when a section code is detected from the group of task, the exact time limit is calculated for the job. So the rest of the tasks are sent to the specified core since it is almost the same. So using the section code model the full utilization of the core is obtained without any miss ratio. The set point changes

here at time 1ms, the next job changes at 6ms and so on. So there is a small gap between the first set point and the second set point. But when it proceeds on the core keeps on getting the job that is ready in the queue. A better performance of maximum utilization of nearly 57% is obtained when using EDF and 71% when using AED.

The performance evaluation is based on the utilization of the cores in cumulative. In the previous figure, the core gives a performance of the steady state after the initial rise. The input pulses and the utilization core curve becomes stable (i.e.) the utilization is full after an initial state. Therefore, a minimum loss is occurred compared to the other task models.

VI. CONCLUSION.

With the advent of the heterogeneous multicore processor, this algorithm could give a maximum utility and a minimum miss ratio while used in the real time system. The existing schedulers that are available are compared and their performance is shown. Also the program was split into procedures and grouped together in the initial stage. Then a section code was tested for the entire cluster of code and using the feedback loop the miss ratio was identified. This gave wider spectrum for each code before it was used in the real time system avoiding the miss ratio an obtaining the maximum utility. Since operating system scheduler algorithm was used over the control system methodology for its design and scheduling the entire performance is better when compared with the existing work. In the future work it is better to avoid the underloading after a period of time.

References:

[1] R.Kumar et al, "Single ias hetrogenous multicore architecture for multithreaded workload performance", *ISCA*, Page 64, 2004.

[2] J. R. Haritsa, M. Livny and M. J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems", *IEEE RTSS*, 1991.

[3] W.Zhao, K. Ramamritham and J.A.Stankovic, "Pre-emptive Scheduling Under Time and Resource Constraints", *IEEE Transactions on Computers*, No.36 (8), 1987.

[4] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of ACM*, Vol. 20, No. 1, pp. 46-61, 1973.

[5] Pawet Piqtek, Wojciech Greja, "Speed Analysis of a Digital Controller In the Critical Application", *Journal of Automation, Mobile Robotics & Intelligent Systems*, Vol 3, 2009.

[6] A.Georges et al, "Method-level phase behavior in java workloads", OOPSLA, 2004.

[7] P.Marti and M.Vekasco, "Toward flexible scheduling real time control tasks: Reviewing basic control models", Proc. Hybrid systems, *Computation and control*, LNCS.

[8] E.Johannesson, T.Henningson and A.Cervin, "Sporadic control of first order linear stochastic systems," *ICHS*, Computation and control, 2007.

[9] M.Rabi, "Packet based inference and control," PhD thesis, *Institute for systems research*, University of Maryland, 2006.

[10] Martin Ohlin,Dan Henriksson, Anton Cervin, *Truetime*, Department of Automatic Control, Lund University, 2007.

[11] J.W.S. Liu, et. Al., "Algorithms for Scheduling Imprecise Computations", IEEE Computer, Vol. 24, No. 5, 1991.

[12] K.E.Arzen, A.Cervin, J.Eker and L.Sha, "An introduction to control and scheduling co-design", in Proc. 39th *IEEE Conf. Decision and control*, 2000.

[13] T.A.Henzinger, B.Horowitz and C.M.Kirsch, "Giotto: a Time triggered language for embedded programming," in *Proc.workshop embedded software*, 2001.

[14] P.Marti, G.Fohler, K.Ramamritham, and J.M.Fuertes,"Jitter compensation for real time control systems", Proc 22rd *IEEE Real time system symposium*, 2001.

[15] P.Balbastre, I.Ripoll, J.Vidal and A.Crespo, "A task model to reduce control delays," in Proc 39th *IEEE conf. Decision and control*, 2000.

[16] Chenyang Lu, T.F. Abdelzaher, John A. Stankovic, Sang H. Son, "Feed back Control Approach for Guaranteeing Relative Delays in Web Servers," *IEEE Real-Time Technology and Applications Symposium (RTAS'01)*, June 2001.

[17] Anton Cervin, "The real time Control Systems Simulator," Reference Manual, Department of Automatic Control, Lund Institute of Technology, 2000.