A Survey: To achieve energy efficiency using scheduling techniques for task mapping on multicore heterogeneous systems.

Sahar Arooj[#1]
Department of Computing
Riphah International University,
Lahore,Pakistan.
Sahar.arooj@riphah.edu.pk

Sonaina Ilyas[#2]
Gujrat,Pakistan
sonainailyas005@gmail.com

Sibtain Ihsan [#3]
Lahore,Pakistan
sibtain94.se@gmail.com

Abstract: **Energy consumption is a big barrier in computer systems. Traditional technologies are used in various DDSs and modern architects to achieve energy conservation. DPM strategy for management is based towards the job performance. DVFS is used together with slack time to achieve efficiency. This work focuses on surveying achievement of energy efficiency using multiple algorithms for scheduling. It also focuses on the techniques classification which are used in the research.The aim of this survey is to provide researchers with knowledge of the power management by scheduling.**

Keywords: **Graphical Processing Unit (GPU), Dynamic voltage frequency scheduling (DVFS), Dynamic Power management (DPM), Directed acyclic graph (DAG), Mixed Integer linear programming (MILP).**

I. Introduction:

A multi-core processor is a single computing element with 2 or more independent process units referred to as cores that scan and execute program directions. A multi-core processor actualizes parallel processing during a solitary physical bundle. Heterogeneous computing refers to systems that contain multiple process units – central process units (CPUs), graphics process units (GPUs), digital signal processors (DSPs), or any variety of application specific integrated circuits (ASICs). The system design permits any accelerator, as an example a graphics processor, to work at the identical process level because the system's CPU. What is a scheduler? A scheduler may also be known as a job scheduler which manages the job queue for a computer cluster. It is one of the main components of IT infrastructure. Scheduler is the process which decides what task and what process should be accessed and run at what time by the resources of the system. It is required to maintain the multitasking capabilities of a system and to keep its performance at the highest level, it help in optimization of system's maximum performance. Real-Time Scheduling Algorithms. For many period systems, the work-load changes from moment to moment, supported external events that need dynamic planning. For this purpose, there are 2 key questions: however they select the subsequent (ready) task to run? Shortest job initial, static priority to highest priority prepared task, earliest start-time point in time initial (ASAP) and earliest completion time point in time initial (slack time). Liu and Layland (1973) demonstrated that for an arrangement of n periodic tasks with extraordinary periods, an achievable calendar that will constantly meet due dates exists if the CPU usage is beneath an explicit bound (contingent upon the quantity of undertakings). The schedulability test for RMS is:

$$U = \sum_{i=1}^{n} \frac{Ci}{} \leq n\ 2^{\bar{n}} - 1_1$$

Ti rate-monotonic scheduling (RMS) rule may be a priority-assignment rule that is employed in time period in operation systems (RTOS) with a staticpriority programming category. Earliest deadline 1st (EDF) or least time to travel may be a dynamic priority scheduling algorithmic rule employed in period of time operative systems to put processes in a very priority queue. the schedulability take a look at for EDF is:

$$n$$

$$U = \sum_{i=1} \frac{Ci}{Ti} \leq 1$$

Energy efficiency is characterized as the mix of enhancing execution while keeping up or decreasing power utilization. Generally, upgrades in energy productivity up to a great extent come as a result of Moore's law, the multiplying of the quantity of transistors on a chip about like clockwork through ever-littler hardware. By and large, more transistors on a solitary PC chip and less physical separation between them prompts better execution and enhanced energy effectiveness. In any case, the energy related advantages that outcome from Moore's law are backing off, possibly compromising future advances in figuring. Utilizing a pattern might really utilize a lot of energy than not utilizing one, and also the closure component might not work on the off probability that you simply have a pattern initiated. Indeed, gift day liquid crystal display shading screens needn't hassle with screen savers by any means that. Power Mode Management: There are 3 styles of modes within the system that are i) idle state, ii) sleep mode iii) active mode. Once a processor doesn't execute any task and its consumption of power is set by the idle power. Dynamic voltage could be a power management technique that is employed in ADPS design, with in which voltage employed by element/job is increased or remittent the circumstances. It essentially uses 2 power saving techniques that are dynamic frequency scaling and dynamic voltage scaling that are utilized in embedded systems to save lots of power. Power will be reduced by lowering frequency or voltage of the hardware. The mixture of offer voltage and frequency features a blocky impact on total power dissipation as a result of dynamic power consumption features a quadratic dependence on voltage and a linear dependence on frequency. Mixed integer linear programming:

An integral programming downside may be a mathematical optimization or practicability program within which some or all of the variables are restricted to be integers. In several settings the term refers to number applied mathematics (ILP), within which the target operate and also the constraints (other than the number constraints) are linear. Dynamic Power Management (DPM) laptop systems are designed to deliver peak performance, however are typically idle or wont to perform tasks that don't need such performance. For a given technology/architecture, there's an in depth relation between the performance provided by a system and also the power it consumes. Dynamic power management (DPM) may be a management strategy geared toward adapting the power/performance of a system to its employment. A fault-tolerant style permits a system to continue the operation it's supposed for, probably at a reduced level, instead of failing fully, once some a part of the system fail. There are two types of faults Transient Faults and hardware faults.

II. Power/Energy Model:

The basic energy model which we will be focusing on is proposed by [1] , [2] and stated that microcontroller energy can be communicated as:

$$E = E_d + E_s + E_e + E_w$$

Further [2] Proposed total energy consumption in 1 hyper period:

$$E_t(S) = E_d(S) + E_i(S) + E_s(S) + E_{m-ov}(S) + E_{v-ov}(S)$$

(S) & (S) is the energy consumed total while the processor is in sleep and idle mode, whereas $E_{m-ov}(S) + E_{v-ov}(S)$ is the total of switching of mode and voltage respectively.

That is why, a static schedule S, they calculated the total energy consumed by the system : Etotal(S)=Eactive(S)+Eidle(S)+Esleep(S)+Ecov (S) where Ecov(S) is the consumption of energy due to the processor overhead by mode switches.

The failure rate at a frequency level can be modeled as : $\lambda(f) = \lambda 0 \times 10d(fmax-f)/(fmax-flow)$, $\lambda 0$ represents average fault rate equivalent to the frequency maximum fmax, and d(> 0) is a constant, which represents the sensitivity of failure rates caused by transient faults. Along these lines, the unwavering quality of an undertaking Ti on processor um ∈ U with WCET wi,m at recurrence fl is communicated as:

$$Ri = e^{-\lambda(fl) \times wi,m \times fmax/f}$$

Systems reliability can be defined as:

$$R(G) = \prod_{Ni=-01} Ri$$

This model was used by [3] [4] [5] [6] [7] [8] [9] [10] with some variations in it.

The basic model can be used for minimizing energy consumption in both multicore and uniprocessor real time systems. In literature various different forms and solutions were given to solve energy consumption problem.

Huang used tool: TGFF Proposed solution:

As energy consumption of the system has become an important issue because of the energy being waisted when the system is idle or in sleep mode or the system is performing a single task while the other ones are waiting we will be focusing on these points for the survey and how to optimize energy utilizations via task scheduling. When doing literature review on the solution for energy optimization is that [8] Scheduling statically for the recovery of tasks which are shared keeps a slack time which is might be needed during runtime. The slack which is saved/specified for recovery that is shared is specified to be as bigest task scaled on the processor working on it under the settings of DVFS. Not at all like unwavering quality mindful DVFS planning, PMM is acquainted with recover the dependability misfortune due to DVFS,and DVFS and PMM are consolidated to diminish framework add up to energy utilization comprehensively. In the scheme EFS, recovery of task combined with optimization of energy techniques are used to provide reliable energyefficient scheduling. But, a main issue is that DVFS and PMM focus to minimize total energy consumed that fills systems slacks that are available, while the shared-recovery needs redundancy of the time that increases consumption of energy. These procedures basically go after the accessible framework slacks, in this way, the objective of saving energy of the system must be cautiously weighted with the objective of upgrading framework unwavering quality.

In addition, the heterogeneity of the framework builds the multifaceted nature of planning with the EFS plot. In order to achieve best scheduling solutions which are efficient for energy a new approach named MLP was introduced including EFS.

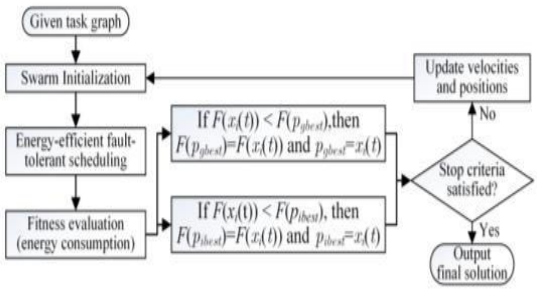


Fig: 1 scheduling algorithm

In view of the rundown booking strategy, he introduced processor allocation to the task stochaistically. After jobs are created the algorithm works on performing energy efficiently schedule under EFS algorithm than every task, position which is individually best for it is refreshed when a superior task comes. In every emphasis, every molecule travels through the optimization space as a function of its velocity. The evolution procedure is rehashed until the ceasing basis is satisfied. Another optimized version of (DVS) was presented [7], in which diverse supply voltages resulted in multiples frequencies for execution. Exhibiting new ways to deal with limit the normal energy utilization for ongoing assignments with discrete likelihood density elements of their remaining burden data when there exist just frequencies which are discrete accessible. Intra-task and between task scheduling situations are investigated, in which the previous thinks about frequency task amid the execution of an undertaking task and the last investigates frequency task among tasks. For intra-task scheduling with frequency, he build up a productive algorithm to determine ideal frequency scheduling. For a solitary task with the goal that the normal energy utilization is limited. The calculation is additionally reached out to adapt to *real time periodic* tasks that may have distinctive power utilization qualities due to various activities of switching.

[11] Utilized slack time to propose another scheduling algorithm for multicore systems. To understand this execution, an assignment isn't executed following it arrives however put off until the point when a few tasks arrive. At that point, these prepared tasks are executed constantly on core Clow. On the off chance that the deadly infringement is anticipated amid execution, the working core can be changed into set of cores Chigh. Sooner or later, on the off chance that the lumped execution is unimaginable on Chigh because of absence of prepared assignments, at that point core set is changed to Clow. As referenced in segment 1, to understand this undertaking planning, they focued around a slack time, which is the aggregate of the distinction between present time and the soonest due dates of each activity. At last it comes down to an issue that when the execution begins and what number of assignments on each core are executed Chigh and Clow. FCFS is used by each core. To acknowledge constant execution with two distinctive execution set of cores, exchanging of cores periodically is essential. To limit energy utilization, the exchanging interim ought to be amplified. To begin with, we begin from S0, which is control off state. Presently, we disclose how to get begin. At the point when L first tasks get execution on Slow, if any undertaking won't be prepared, the execution ought not to be begun yet to keep away from

superfluous idle time. Next, discussing when core execution should increment from Slow to Shigh. At the point when the following assignment is executed on Slow and after that next L errands are executed on Shigh, if there is any chance of deadline occurrence infringement happens, the state of cores working must be changed to Shigh before executing the following undertaking to meet due date limitation.

Table: 1 and 2

TABLE 1. EVALUATION SETTINGS FOR JOBS

Parameters	Job A, B, C	Job D, E, F
Number of jobs	3	1 to 3
Job ID j	A, B, C	D, E, F
Input interval $I(j)$	100ms	40ms
Deadline period $d(j)$	250ms	

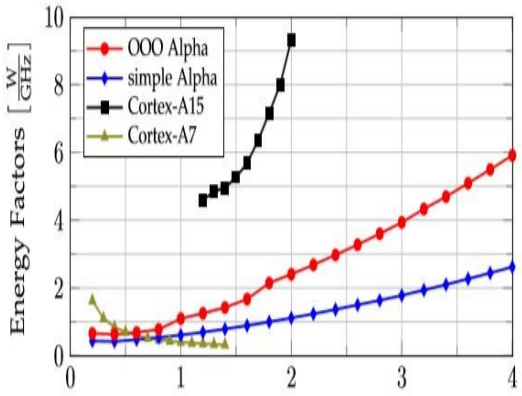
TABLE 2. EVALUATION SETTINGS FOR THE HARDWARE PLATFORM

	MCU1	MCU2	MCU3	MCU4
core ID c	c_1	c_2	c_3	c_4
Relative Performance	1.0	2.0	3.0	4.0
Power Consumption in				
Active state [mW]	15.4	36.8	90.9	231
Sleep state [μW]	0.69	2.07	6.20	18.6
Energy Overhead [μJ]	51.0	124	253	430
Exec time of				
job A, B, C [ms]	2.4	1.2	0.8	0.6
job D, E, F [ms]	49.8	24.9	16.6	12.45

executed on Shigh. This technique is over and over connected until any due date infringement is normal. In case of its normality, the state which is held on currently gets changed to Shigh. This kind of process is applied continuously until we don't have any task which is ready to run. Respectively when a new task is expected, the state which was currently held gets changed to S_{low}.

[5] Addresses the problem of minimizing emerging for the RT occasional tasks using DVS on multiple processors when and used scheduling for partitioned tasks. Considering the minimzing of energy for tasks which are preemptive hard and periodic and are scheduled on an identical **multiprocessor platform with DVS capability** [23].

[3] proposed energy-sub-optimal-federated scheduling for DAG tasks. For multicore scheduling, an individual task is restricted or can be executed on a single processor which is same as partitioned scheduling or it has a special access to all the processor present in the system on which it can be executed. As each processor is specified to a single DAG task they tried to minimize the energy consumed of that task. For the DAG first tasks are decomposed into a set of task that are sequential and can execute sequentially. To substitute variables the minimization of energy problem is transformed into optimization convex problem. **Processor sharing:** The major cost is dependent upon the energy consumption which is leaked. The scheduling solution is not that optimal for that they introduced a new solution. They combine processors that have been allocated to a similar DAG undertaking. In this progression, every DAG undertaking is dealt with independently and the subsequent processorhub/DAG task stays in the combined planning system. In any case, in this subsection, we have expected that every processor to be consolidated just once.



Frequency GHz [12] Load distribution:

[9] Proposed that the ideal load dissemination is determined if we have a taskset and a stage with discretionary number of centers and power show parameters. The load to be appropriated is the aggregate calculation request of the taskset, which can be communicated as a solitary amount just under an admired supposition that the equivalent cycle's number on any PE is requirement of the tasks. A taskset with aggregate calculation request of U = ten cycles for each second is allotted onto two unique stages, Πa and Πb. The two stages with a similar power display parameters κj and βj. Be that as it may, platform Πa is homogeneous

with α1 = α2 = 2, though stage Πb is heterogeneous with α1 = 2 andα2 = 2.75. To deal with stages on which the variety in execution time crosswise over PE types is huge, we sum up the streamlining issue utilizing the accompanying knowledge. The aggregate load is reasonably part not just into the same number of sections as there are centers yet every one of those sections is then part into the same number of sub-pieces as there are errands. As a result, each part can join execution

contrasts crosswise over heterogeneous centers and undertakings.

For marginal power heuristics he proposed a rule for assignment of tasks which is same as the process for optimization end of required distribution of load. Tasks get sorted in task to a PE just while doing as such does not surpass the pre-determined wanted load. Setting the powerful limits of the cores to their ideal burdens toward the start of the allotment methodology anticipates early choices which change portion which is far different from the one which is ideal.

Xing Liu, 2018) propose solution stated that the reason for not mapping CFG single core technique to the multi-core system is due to the concurrent task execution in the multi-core system. For fault detection he proposed that by the comparison of consistency within runtime signatures comparing the consistence between the run-time signatures as well as the formal signatures which are precomputed, the controlflow faults occurred in the multi-core system can be detected. Furthermore for fault recovery by check pointing Xing added two two general registers to reduce recovery time. At checkpoint while doing context switching registers values are moved to the other one so when a fault occurs it only requires shifting of processor bus for rollback recovery.

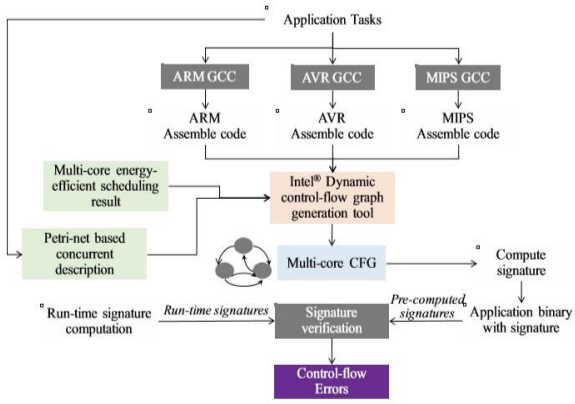


Figure 2. Elmentary diagram of multi-core fault detection process.

Fig 2: Elementary diagam of multi-core fault detection process

Connection analysis was based on the required performance level as per the analysis request. By taking the number of levels of DVFS for each cover, having huge number of considerations to think through the level and usage level of each cluster to succeed. For example, if ours are divided into two types of cluster NB and NL, each basic type has FB and FL frequency levels and the use of the work is the level L. Then the num of councils is:

C =
Nb∗ ∗ ∗ ∗ ∗NL Fb FL Lb LL+Nb∗ ∗ Fb Lb+NL∗ ∗FL LL

Believing that the DVFS applied to the cluster level and the use of all the cash-in-work work is the same level of load.

Joint Optimal DVFS Task Mapping**:**

They proposed an ideal calculation, i.e., OJTM, to fathom PP1 dependent on its exceptional attributes, i.e., the twofold and ceaseless factors are combined with one another directly. The basic thought of OJTM originates from Benders disintegration. Rather than considering every one of the factors and the requirements of PP1 all the while, OJTM breaks down the PP1 into a Master Problem (MP) and a Slave Problem (SP), which are fathomed iteratively through an input circle. Thusly, the computational multifaceted nature of the arrangement is significantly diminished [23], [30], regardless of whether the PP1 is noncurved. The structure of OJTM is appeared in Fig. 1. The MP represents all the twofold factors (errand to-processor portion and recurrence toundertaking task) and the comparing some portion of the target work and the requirements of PP1. It additionally incorporates a lot of limitations called Benders cuts. The SP incorporates all the consistent factors (assignment modification) and the related limitations of PP1. The SP arrangement gives extra requirement, which is consolidated in the MP through the Benders cuts.

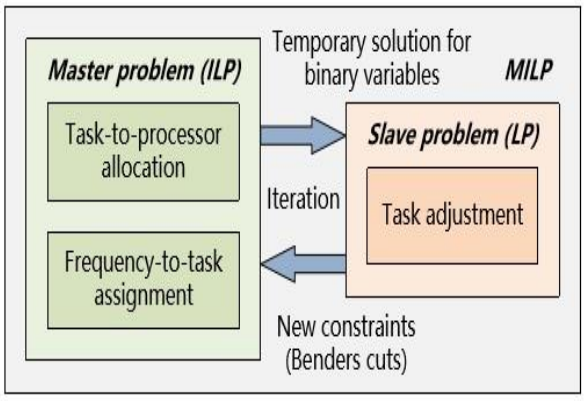


Figure 3. The structure of OJTM algorithm

III. Joint DVFS QoSaware

Task mapping (JDQT) calculation, to efficiently settle the PP1 (20). The structure of JDQT is appeared in Fig. To settle the PP1, finding the ideal errand to-processor distribution and recurrence to-undertaking task choices is the most critical advance. This is in such a case that parallel factors c and q are resolved, the PP1 will decrease to a LP issue, which has a more straightforward structure and is a lot less demanding to tackle. Drinking sprees disintegration [26], [27] is a powerful strategy for fathoming MILP with ensured worldwide optimality. Rather than considering every one of the factors and the limitations of PP1 all the while, Benders disintegration partitions the in two small problems with lesser factors i.e Master as well as slave problem. At this point it iteratively runs the the MP and SP that is one is run in the other.

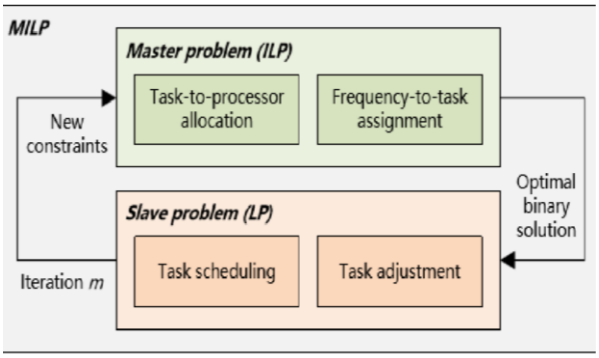


Fig4. The structure of JDQT algorithm.

Mp has all the variables which are binary and also functions which are it's part correspondingly. It likewise contains one auxilary (persistent) variable ˆ Φ that is acquainted with encourage the connection between the MP and the SP. Then again, the SP consolidates all the consistent factors {o,ts,h} and the related requirements of PP1. At first, we comprehend the MP and acquire a lower destined for PP1's ideal target work an incentive alongside a lot of esteems for the double factors. At that point substitute these qualities into the SP and understand the double of the SP (DSP) to acquire an upper headed for PP1's ideal target work esteem. In light of the arrangement of the DSP, another limitation called Benders cut is created. This requirement is included into the MP, and another emphasis is performed to understand the refreshed MP and SP.

*Accelerated Solution Method: AJDQT***:** In spite of the fact that the arrangement given by JDQT is ideal, this strategy can't be utilized to efficiently tackle huge issue sizes. This is because of the accompanying reasons: I) as the MP (21) is an ILP, this issue is still hard to unravel specifically, contrasted and the SP, and ii) at every emphasis, another achievability imperative (31) or infeasibility limitation (34) is included into the MP. With an expanding number of emphases, the computational unpredictability and the span of MP both increment. So as to evade these difficulties, they proposed a quickened JDQT (AJDQT) calculation to additionally diminish the computational multifaceted nature of JDQT. This calculation contains two sections.

1 Sched_ algorithm for deadline

To apply our approach to consider the use of a work on our specific strategy, we need to be assigned to target at the specific level, to execute the implementation process. Stay tuned during. There is no direct way to implement such obstacles in common Linux schedules. The Linux mainstream has introduced a different scheduler framework. It clearly offers the possibility of expressing the use of a work. For each work, the Shaded Dood line line requires three parameters: Work runtime, its duration and last time. Therefore, we can get the same usage usage level while assigning a specific job timetime value.

2 Model validation

To evaluate our methodology, we are considering an app that can describe diverse parallel levels and the date set as well. We use Black Scholes applications from a Paris benchmark. This app computes European Options Portfolio prices, based on partial sectarian formulas. The Black Scholes Benchmark symbolizes a comprehensive series of divine partial equality (DPE), DPE is used in final examination. Plans applications with input options as input portfolio. After that, the portfolio distributed between the themes is processed. The application mainly contains FL heat point processes & CPU intensity. The app is mostly expanding and it uses a small parallel. The app applies as follows: Initially the main thread option portfolio starts, then spy the worker's topics that make in parallel data part. Blk Schls EqEuro NoDiv purpose runs the function to calculate the cost of each thread.

During experiments, we use 8 themes in a parallel region and only estimate the region behaviour, called regional interest (ROI). Parallel model diagram in fig 2 shown.

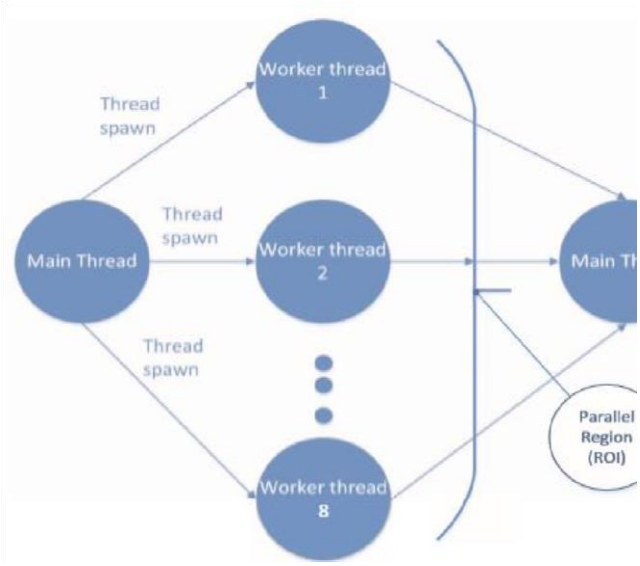


Fig: 5 benchmark Parallelism Model

We offer some effects that have been described for a while, according to its statement. Find the utility of the model utility model regarding the workshop process. Then, via artificial quality, we value the effectiveness of the use of control on energy utility types. Finally, compare 2 architecture and possible explanations for their differences about energy efficiency will be discussed.

Ans to Q1. Effect of the type of work on relative energy utility of partner points.

We select the selected platforms from the core cork prochin on parallel multiple degrees offered from the selected platforms. Executed with 1000 Workloads repetitions and amount work metrics as per the workshop per second throughput. Average power consumption is dignified during the time. In Figure 1 we selected the energy ef fiiiency world's workload for platform of selection con points. There is usually no scientific alteration in the utility in case by the C3 and C1 provided adaptation points for three workshops. The difference between the controls of these adaptation points is control control we remind that only, which is implemented up to 60% in C3, while the core in C1 reaches 100 percent. Both sides have set their intermediate frequency (600 MHz and 1GHz for A15) for an A1. Although a comfortable procession strategy is selected or otherwise applied to the passive methodology, where the process is set at the lowest possible speed, performance requirements, although with the difference in energy utility C2 and C4 It remains more in which the race is inactive, inactive, on fastest speed running. There is no energy utility on high-speed speeds.

Clearly during the process of cryptographic function, our relation to platform opportunities points is of relative energy different results. Here, the C2 and C1 displays enhanced than the C3 and C4 better. This behavior model is against predictions. Autel in the data results are presented. Same behavior we observe in terms of ARM, with a improved num of energy providing. Still, the best energy of this combination is achieved by the success C1 in a safe bet application. The original model in highest energy selection is not able and utility approach in terms of the highest fully invented in the workshop instructions.

Ans to Q2. Efficiency with Load control impact
using control impacts utility, and make sure that its effects are more important than others, comparison with others. Acquisition Beach used Multi-core Energy Standards to produce load-based work-based work [11]. The burden of the work is suspended by the set of preliminary prediction points (C1-C4) sets. Performance statistics is logged by external data. Results 3, and 4 are presented. 4.

In the case of workloads included by enters extra additions and conditions, we cannot get identification from the SHA256 workshop. For Intel Platform (ID 4), according to C3 and C3, still offers better energy than C2 and C4. Combining the same level, as well as the army is repeatedly repeated through the model used. It is obvious that no differe in the level of energy utility achieved by the for different types of C1 and C3 instruction for both architectures.

Ans to Q3. In which better energy is provided utility Arm vs Intel?

We discussed in the circles of educational and industrial communities. In our experiments we deal with problem with a two scene. On one hand we use real applications to measure energy utility levels to get both architects

from used domains, on the other hand, we use architectural tests to test architects that work with us. Enables samples to do. Intel got more energy. In comparison to ARM, in every workshop. Even the XML shows the lowest score for the energy utility in the. For low score One possible explanation is that the application is a poor IPP, combined with SHA256 applications, most of the workloads combined with ILP, which show the biggest result of energy utility. The power consumption during the load of the liner algebra work process.

During execution, Exynos chip is advanced than the temperature atom's patient atomic temperature. It proves that the heat generated does not get rid of rapidly, which increases the temperature of the fast. This observation gives us the result that there is more static power in powerful data, even though the army displays better performances (more frequent chances), yet the power utility score is in favor of Intelom. As a competition we done experiments same on architecture of intel, in which low-static electricity consumption.

In the second case the overall picture is obtained, as well as various function models used in Apple Beach synthetic quality. Army architecture provides better energy models alomost all of them used utility, branches, in Intel shows better results. It can be a better branch offering in processor of Atom. This integration is done with the epEBench standard where we emphasize both chips with the instructions of the 200G branch and there is a decrease of 15% lower branch atomic Exynos chip.

In addition, there is no difference in the difference between Intel and a point and double point operation, while ARM is a single-precision operation with maximum energy efficiency than the double energy. This is due to the presence of emigration in the Army architecture we believe on this, with which the A7 houses provide an increasing level of utility. In addition, the energy utility achieved in the load load consisting of the SIMD instructions is more in the army compared to Intel (we are very little because they are very low because they are very low), we understand if we have a neon hanging engine in Exynos Chip. Consider the presence.

Ans to Q4. Thermal feature on the results of energy utility.

Temperature experiences held in the previous submission, we can see the presence of increasing power consumption in Army Chip compared to Intel, due to the difference in the energy efficiency score. Previous experiences were organized in a highly refrigerated environment, which resulted in increased control over the chip temperature. In another experience, we remove the board from the controlled environment and try to re-process the liner workshop with 1000 repetition and with only active fan as a cooling system.

With oscilloscope during execution Current surveillance used. During process of multi-repeat work load, we start gradually slowly in consumption of current, the initial repetition of the load. For the first time, in power consumption increase is due to the stagnant power. The same experience is held in Atom Intel which provides better control over thermal effects. On the other hand, the function of artificial load of static power produces different levels. We understand that only specific types of instructions, such as instructions of memory, while processing inside the loop creates more stable power in comparison with others.

Reinforcement learning algorithm to search for the optimal solution: RL is a sort of machine learning calculation. It can learn iteratively by cooperating with nature, in order to boost a given thought of total reward. As Q-learning is a RL system utilized in machine learning, utilize the Q-learning calculation to settle the multi-core energy productive scheduling issue in this segment. The learning procedure of Learning is as per the following: Before learning starts, Q is instated to a potentially self-assertive settled esteem. At that point, at each time t, the operator chooses an activity at, watches a reward, enters another state st+1 (that may rely upon both the past state set and the chose activity), and Q is refreshed.

As it is appeared in Fig4 structure. To settle the PP1, finding the ideal errand to-processor distribution and recurrence to-undertaking task choices is the most critical advance. This is in such a case that parallel factors c and q are resolved, the PP1 will decrease to a LP issue, which has a more straightforward structure, is a lot less demanding to tackle. Drinking sprees disintegration [26], [27] is a powerful strategy for fathoming MILP with ensured worldwide optimality. Rather than considering every one of the factors and the limitations of PP1 all the while, Benders disintegration partitions the PP1 into two littler sub-problems having less factors and imperatives (i.e. Ace Problem (MP) and Slave Problem (SP)). At that point, it fathoms the

iteratively the subproblems by using the arrangement of them.

All-encompassing calculation, which is known as Fixed Frequency Island-and Task-Aware Largest Task First(FIT-LTF), appoints each assignment to the slightest stacked core of the island with the most reduced factor of energy of core type for the comparing undertaking task, to such an extent that all tasks stay achievable. FIT-LTF is introduced in Algorithm 3, which has some similarity to Algorithm 1, with the principle contrast that in FITLTF there is no worldwide request of the core kinds regarding their energy factors. In this way, the second circle (lines 3 to 10) isn't hindered and consequently the calculation tests all Q center sorts and all islands of each core type, with a period multifaceted nature of O(V). Subsequent to finding the core of type j with littlest cycle usage (line 4), i.e., Ck,i, in the event that τn can be plausibly doled out to Ck,i (line 5), records g and y monitors islands, cores which are best efficient in energy, separately, in which to appoint τn (lines 6 to 8). On the off chance that there was an attainable task (i.e., on the off chance that g > 0 in line

11), τn is alloted to core Cg,y, with a period intricacy of O(logMg), or of O(logM_k^{max}) in the most pessimistic scenario among all islands. Hence, the aggregate most pessimistic scenario time multifaceted nature of FIT-LTF is O(N (V + log M_k^{max})).

IV. Experiment Results

This section has compared comparisons with our Algorithm three suggested algorithms, including RDVF [25], MVFS-DVFS [27], MMF-DVFS [26].

Use two-way method for these algorithms and 2 setup graphics: 1st phase to get both the time-based Hurstestic Base and line base line. After this, HEFT without performing without performing performance in terms of energy utility around four of Algorithms. **A. Setup for Experiment**

The same voltage [16] used in processors is used for frequency scanning experiences and built-in HCC in the tab. II Enters the frequency of these processors and related power consumption.

V. PARAMETERS EXPERIMENT

Parameter Value The num of processors {2, 4, 8, 16, 32} cycles of Task execution [5×109, 50×109] CCR {0.2, 0.5, 0.8, 1}.The work of the applications of the application is presented by the process of execution cycles that have uniform distribution for random values generated at intervals [5 × 109, 50 × 109]. Random graphics vary depending on work, costs of communication and work distribution, while real-world structure graphics is presented first. Experiments are set up by different various processors and tasks. Consumption of energy each email finder is considered as the baseline result of the algorithms similar applications are routine by the value of HEFT on graphs. We resolve the lpv5.5, [21], solar based on the revised cussed method to solve the linear problem we have. In the results, the average number of data obtained in each of the data points is 100 experimentations.

B. Application Graphs Randomly Generated

A random DG generator that in related tasks used [24], to create heavy application graphics with different features. The number of jobs is selected from {10, 20, 50, 80, 100, and 200} & quantity of processors in the HCC is selected in the tab. III. No. 3 describes the comparison of RVFS, MMFDVFS, MVFS-DVFS, and LP-DVFS compared to different types of functions and energy-saving ratio with processors. This indicates that

In additionally, we understand that the lack of promotion of energy of four sla rehabilitation algorithm is high when the num of work growths. This incident can indicate that additional tasks are flown

LPDVS has eliminated in all cases other algorithms.

into some degree of processors, such as decrease in slow-time reduction. As a result, this can be done which increases the number of processors when it displays more energy savings.

C. Application Graphs Real-world 1)

Gaussian Elimination:

Given ending is an algorithm to solve the linear equality system in linear equations, which is also performed on the relevant matrix of the protective shows the detailed explanation of the gas alleviation algorithm. Each TK, K indicates the operation of a pilot column and represents each TK, a refreshing operation. FY Oraga graphics, which has been approved. The image displays the graph for a special

It can be seen that LPD is better than other algorithms in energy efficiency and performance the values stabilize when the size of the matrix gets more than 10. In this way, the image shows the results of comparisons on processors

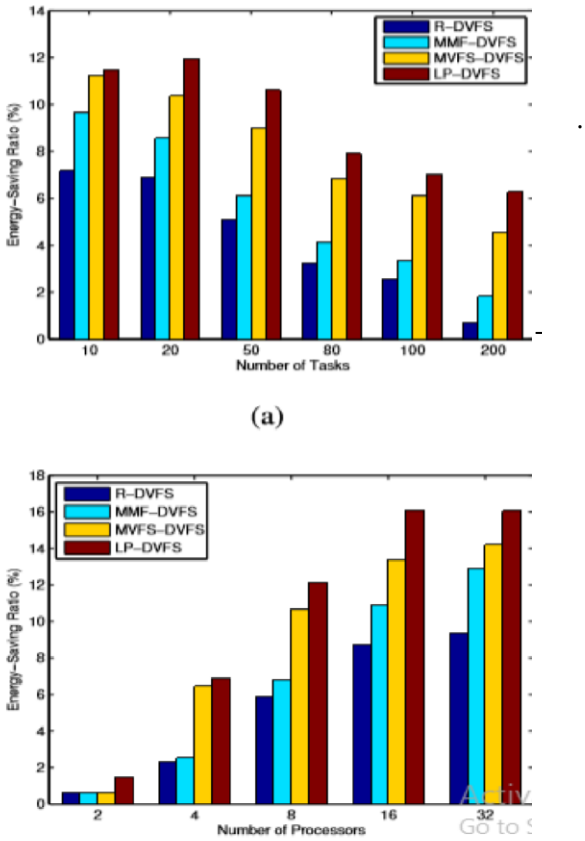


Fig 6. Comparion with performance of the tasks

at different processors. Energy deficiency estimates at the maximum cost when the processors are near 8. In other words, the process of increase in the processor may increase the decrease in energy due to gussasion application of elimination.

2) Transform Of Fast Fourier:

The (FFT) counts a configurable instant, or its detailed dimension of a dimensional FFT algorithms and a graphic, the number of tasks, the number is used. The FFT algorithms contain two parts: Reverser Call and Butter fly Operation. The total num is get by FFT v point where 2 × (v-1) + 1 re-cursive call task and Vlog2v butterfly are employed by the UV From 4 to 64, the amount of input points for the graphics points gets the power-saving ratio of each algorithm. LPD DVFS and MMF-DVFS more energy algorithms perform utility schedules in all cases. Specifically, the higher performance of 2 algorithms becomes more important. The quantity of processors cannot increase in the way of save more energy, in case of prolonged demand.

A. Table Of Energy Efficiency

Proof of the concept of the view, they used platform adaptation points that provide utility which is of high level to perform as an actor of an application which matches stereo. To promote the highdefinition of African platform, we start with the actual measurements made with Exynos 5422 chip, such as [26]. With the models we receive and related performance values, we can list all platforms opportunity lists in utility energy terms, power and performance. The list has been configured as a folklore table, which is counted as platform points.

The compatibility point of each platform contains:

- LB: The use of cores in the larger cluster
- FB: frequency cluster of large cluster
- NB: A15 runninG number of cores
- Li: Cores usage rate in small cluster
- Fi: Frequency point of small cluster
- NL: A7-core number of running

Relevant performance and energy utility has been shown in the energy utility table. By looking at the table, we can target platforms of points of interest points. A group is selected for the necessary performance level, and inside different energy level exist.Then we can choose the best energy viewpoint. Assume that we want to deal with 35 kg / one act, while consulting the energy utility table, we have realized that we have various platform opportunities points that promote this performance. In order to make the right decisions in Figure 2, to create opportunities and ideas, 10 performance we plot in groups from the table of energy efficiency. We can use 1,15 GHz at 1A15 tomorrow, which consists of 20% + 1 A7, which was processed at 1,2 gts at 90% rate, or with 10% and 4 A7 Could have a 5GHZ on 4 A15. 70% with 200 MHz. The second introduction gives improved energy utility 5 times.

From table I received, we choose the opportunity point of the highest energy utility platform, which is highlighted at a greater parallel level (which is due to 8 for this architecture). This Platform's Opportunity Approach 4% on the 60% with a rate of usage on AGH-core 1.1GHz 60% A7 cores 4 with 200MHz rate. According to the table, I provide this organization performance of 137kg / s (performance per second). This is the platform opportunity point (COI) for interest.

Therefore, within the PREESM device, we have set a schedule + application map. We focus on a specific area of implementation where committees are running, as it is advanced on Figure 3. We call this process of execution as a whole (ROI). They used COI utilizing Linux driver such as cpuidle and cpufreq drivers for the frequency and use of cores configuration. Experimentally, to verify our viewpoint, we need to overcome that on which a specific task is implemented. The usage rate is part of the platform adaptation point. In order to control the use of cluster usage rates, we have used the Shade timeline [2], which has a new schedule in which the CPU bandwidth may be able to control.

Using Digital Oscosecope, we evaluate the current consumption of the board so that power consumption measures during the application process. In Identity 4 we show the Oscoscope scene of applications implementation without having to overcome the rate of use, and where it uses the square clock clock frequency. Current consumption in the same area with a rate of 60% when this area is used to schedule when seeing current consumption. A wrapper we use to implement the actor's mode of version to enable its COI for this specific area. Wrapper pseudo code is showed below.

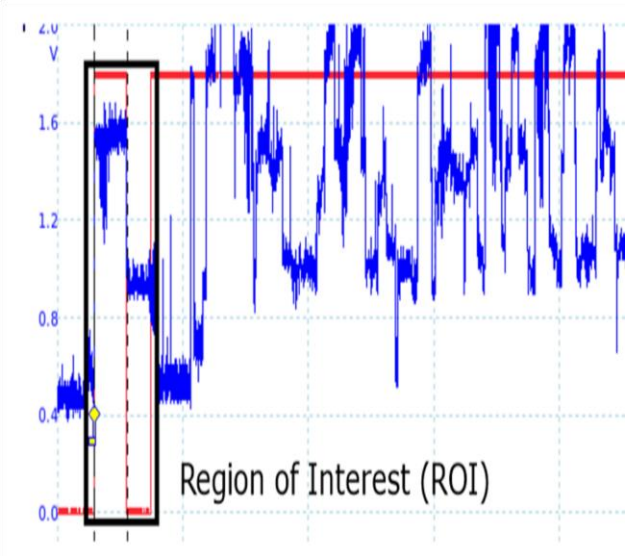


Fig7. ROI normal schedule

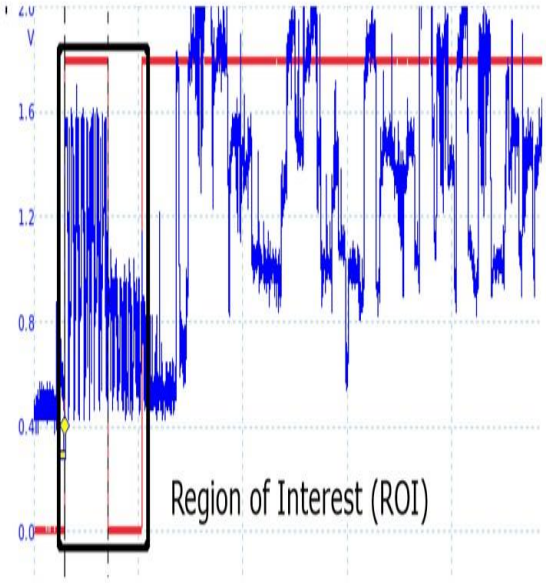


Fig8. ROI at a 60% rate with utilization control

Code

```
1  #include <cpufreq.h>
2  #include <pthread.h>  3     CW
   Wrapper () {  4    Policy Modify
   (core, “user space");
5  frequency Set (core, FREQ);
6  Create p thread ();
7  join p thread ();
8  }
```

With a wrapper code, we set in line 4 in the consumer mode & in 5 lines we have set up according to the Queecocci Linux 6-7 frequency. We support honor models that we have for the MODA Actor Code. Tool creates a CF for each processing factor. Therefore, this device primarily produces 8-fai names for architecture models used to represent the octasia core CPU in PREESM. They are basically basic core communication and sync.

Inside the files we plug in to set up a platform layout for ROI. Envelope code strengths that each central thread that has been played on the home has an extra thread, which includes a code for identification of identical identifiers, For ROI, and for this purpose, since the last date of schedule on

the mainstream topic, we make each other that has to control the implementation rate.

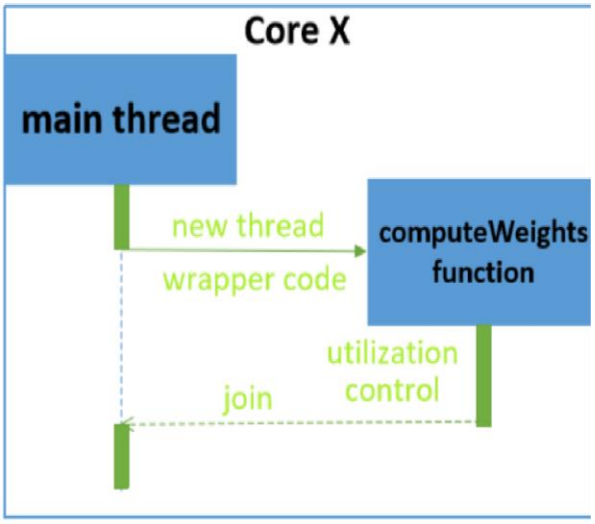


Fig9. Core Compute Model

Dynamic selection of core is best suitable.

To limit core set exchanging, the usage of every core ought to be as same as could be expected under the circumstances. To acknowledge such employment task, we present after calculation. At the point when core use is in a perfect world adjusted, the proportion of appointed undertaking task size is equal to the core execution p(cm). In this manner, perfect use of core cm is defined as u(cm)=p(cm) $\sum_j$ l(j)/tp(Cx). To begin with, the littlest activity is doled out to the most minimal execution core (c1). At that point, if its use does not surpass u_{ideal} (c1), the following littlest activity is relegated to c1. At the point when the use surpasses u_{ideal} (c1), there are two conceivable outcomes, regardless of whether the last doled out employment is doled out to the present core (c1) or next core (c2). For the best task, we look the two potential outcomes simultaneously. Subsequently, we can get $2^{M'-1}$ blends. Accept each task that has a place with same occupation is assigned out to same core, as it were, this is per-work premise task. To acknowledge progressively flexible task, per-errand task is conceivable. For this situation, the errands in a hyper interim ought to be considered in the meantime. In whatever is left of this paper, to streamline discourse, we use per-work task.

Task Scheduling:

To understand this execution, an assignment isn't executed following it arrives however put off until the point when a few errands arrive. At that point, these prepared tasks execute constantly on the set of cores Clow. On the off chance that the due date infringement is anticipated amid execution, the set of working core is converted to set of cores C_{high}. The lumped execution is unimaginable on Chigh because of absence of prepared assignments, at that point core set is changed to Clow. As referenced in segment 1, to understand this undertaking planning, we center around a slack-time, defined as aggregate of the distinction between the present time and the soonest due dates of each activity. At last it comes down to an issue that when the execution begins and what number of assignments are executed on every set of core Chigh and Clow.

Algorithm for Scheduling:

The accepting d(j) ≥ HI. For whatever length of time that satisfies the condition, Chigh can ensure that the deadline is met. Generally higher execution core set than Chigh might be required. Here, clarifying the subtleties of the proposed task planning calculation. Beginning state is S0, which demonstrates all the processors are off. Alternate states are Shigh and Slow. They compare to Chigh and Clow are dynamic individually. As mentioned the core which is working gets to changed based on the slack tie, as each work is determinant. Respectively and issue arises that is which task will get to be in execution at each state.

Obtaining Scheduling Energy Efficient:

In every core that tasks are executed some of them will execute on the basis of FCFS. For instance, accepting five occupations (indicated as A to E) 3 hetrogenous cores are executing (signified as c1, c2 and c3). The jobs interval which is inputed is half as compared to B job and tasks intervals of information as well C,D are as same as B job. There are 6 tasks a constant periodical exchange of cores is required inorder to run them. For limiting utilization of energy it is necessary to apply exchange interim.

To begin with, we begin from S0, which is control off state. Presently, we disclose how to get begin. At the point when first L tasks are executed on Slow, if any undertaking won't be prepared, the execution ought not to be begun yet to keep away from superfluous idle time. Next, discussing when core execution should increment from Slow to

Shigh. At the point when the following assignment is executed on Slow and after that next L errands are executed on Shigh, if any deadline infringement happens, the working state must be changed to Shigh before executing the following undertaking to meet due date limitation. Presently, we clarify whether the following undertaking ought to be executed on Clow or Chigh with Fig. 3. In this model, Clow comprises of{c1,c2}. c1 executes work A, B and C. c2 executes work D and
E. Chigh comprises of {c1,c2,c3}. c1 executes work A. c2 executes work B and C. c3 executes work D and E. How about accepting the present state is Slow and the following assignment is A3. Presently, can execute the following task without deadline infringement as appeared with a black box. After this execution, next L tasks will be executed on Shigh as appeared white boxes. At that point culmination times of these tasks are effortlessly anticipated. On the off chance that any assignment damages due date limitation, the following undertaking ought to be executed on Shigh. This technique is over and over connected until any due date infringement is normal. When it is normal, state current is changed to Shigh.

TABLE 1. EVALUATION SETTINGS FOR JOBS

Parameters	Job A, B, C	Job D, E, F
Number of jobs	3	1 to 3
Job ID j	A, B, C	D, E, F
Input interval $I(j)$	100ms	40ms
Deadline period $d(j)$	250ms	

TABLE 2. EVALUATION SETTINGS FOR THE HARDWARE PLATFORM

	MCU1	MCU2	MCU3	MCU4
core ID c	c_1	c_2	c_3	c_4
Relative Performance	1.0	2.0	3.0	4.0
Power Consumption in				
Active state [mW]	15.4	36.8	90.9	231
Sleep state [μW]	0.69	2.07	6.20	18.6
Energy Overhead [μJ]	51.0	124	253	430
Exec time of				
job A, B, C [ms]	2.4	1.2	0.8	0.6
job D, E, F [ms]	49.8	24.9	16.6	12.45

Untill any task is not ready this process will be applied repeatedly. When expected, current state is changed to S_{low}.

VI. Results:

[3] the two adaptations (single and various) of intraDAG processor blending performs essentially superior to the GSS calculation. Intra-DAG processor combining results in any event 23.3% (single union) and 25.05% (different unions) bring down energy utilization contrasted with the GSS for the symphonious periods. While thinking about the self-assertive periods, this decrease winds up 42.22% (single union) and 45.83% (different consolidations). On the off chance that the inclination based techniques is considered to locate a superior execution design after intra-DAG processor combining (single union), at that point it gets further enhancements energy utilization. By and large, for the symphonious task time frames, it prompts a decrease of the power utilization extending from 27.8 to 52.94% and for the discretionary assignment task time frames the range is 53.55 to 68.22%. [4] *Load Distribution Approximation* the first examination affirms that the proposed heuristics are without a doubt approximations to the fundamental load disseminations. Relative execution of the reasonable, relative-control, and minimal power heuristics on reproduced issue occurrences coordinates the execution of the heap conveyances. *Energy savings* the set of tasks are generated in a way that they can be best allocated.

Every concentrated heuristic, with the exception of KX3-DP, effectively misuse the slack on delicately used stages (U $\leq$ 50 %) for around. *Performance* the normal execution time to parcel a taskset for two issue sizes a factor of two separated. This execution time cost is regularly brought about at framework configuration time and might be caused at runtime in frameworks which resegment the taskset on the off chance that it changes at runtime. The expense of a heterogeneity mindful arrangement is something like a factor of three over that of the straightforward load adjusting heuristic. At the point when contrasted with MPWR,the arched streamlining sub-step expected to separate the greatest vitality investment funds with DL-CAP heuristic costs a further factor of three for the littler issue measure however just a factor of 1.58 for the bigger issue estimate, which proposes a high settled expense yet ideal scaling. [10] On a normal MFFDPS lessens the static energy by 199.59%, 33.42% and 1.99% over MFFNOPRO, MFFSTATICPRO and FFDPS calculations separately. As the processor executes at most extreme frequency and voltage amid both dynamic and inert period i.e., all through the calendar MFFNOPRO expends steady powerful energy regardless of the uses. The dynamic

utilization of energy pursues indistinguishable pattern from static energy utilization as the processor can work just with most extreme frequency and voltage i.e., no DVFS supports. MFFDPS outflanks different calculations along these lines. The quantity of lingering choices in MFFDPS is a lot lesser than MFFSTATICPRO and FFDPS. On a normal MFFDPS has 23.05% and 15.84% less lingering choices contrasted with MFFSTATICPRO and FFDPS calculations separately. This is on the grounds that the shutdown length at every choice point in MFFDPS is high which is because of the explicit game plan of assignments by MFFBP amid allotment. [2] solution results stated that As observed, OPT accomplishes 32% to half expected power funds regarding DVFS-DPMOPT, and 8% to 17% concerning Sub-OPT when the assignments are with Gaussian dissemination. In Fig. 13, OPT accomplishes 52% to 66% expected power funds concerning DVFS-DPMOPT, and 14% to 30% as for Sub-OPT when the assignments are with exponential conveyance. In Fig. 14, OPT accomplishes 53% to 60% expected power reserve funds concerning DVFS-DPMOPT, and 21% to 27% regarding Sub-OPT for sensible applications. OPT accomplishes progressively expected power funds regarding Sub-OPT when the quantity of canisters increments in all the engineered and sensible undertaking sets. This shows OPT not just takes the upside of the drawn out anticipated inert time, yet in addition make full utilization of the probabilistic circulation to produce a more vitality effective intra-undertaking planning and voltage task when the quantity of receptacles increments. [7] approach shows that using Linear programing for the determination of frequency of assignments significantly lessens the consumption of energy which is normalized upto 20%- 30% as comparison to another approach OMGREEDY which has greater amount of complexity of time to get a solution. OMGREEDY lessens upto 10- 20 percent compared to MGREEDY and states that inter-task scheduling can reduce the consumption of energy effectively. The proposed algorithms were effective in reducing (expected) energy consumption. [6] proposed technique PP spares extra energy contrasted with WP2, up to 15.5%, 19.0%, 9.1%, and 14.6% for the four undertaking task sets. This is on the grounds that PP performs likelihood based remaining task at hand adjusting that is firmly combined with the likelihood based DVS for individual processors. For the interactive media undertaking set, PP accomplishes greatest extra energy investment for 2 processors and the reserve funds decline as the quantity of processors increments. [5] *Effect of partitioning schemes*. RESERVATION(k) is an effective attempt for the maintenance of balance within the feasibility of good performance for First-Fit and Worst-Fit performance of energy. When comparing the performance of scheduling schemes where k is to 2,4,6 which defines RSRV2..RSRV6 respectively. RSRV2 results to be best feasible compared to other schemes for $\alpha = 1.0$ and is in fact comparable to FF and BF. Xing Liu, 2018) stated that upto 90 percent of the flow of control faults of multiple core system have a possibility to be recovered by there approach. Also the time of recovery for his approach nearly has 40 percent less than the other software's that also focus on recovery CFCSS.

VII. Conclusion:

Sub-OPT overlook the probabilistic circulation when booking. Interestingly, OPT all around coordinates DVFS, DPM and the probabilistic dispersion everything being equal. Moreover, between undertaking and intra-assignment booking are both used. For instance, OPT can quicken the bins with low likelihood, in this way draws out the rest time and spares static power. It can likewise relegate low voltage/frequency to the bins with extensive likelihood to spare unique power. Additionally, OPT can put the assignment that has a vast likelihood to complete right off the bat before an inactive interval to create a bigger idle interval in scheduling. The level of expected power investment funds that OPT accomplishes as for DVFS-DPM-OPT and Sub-OPT. As observed from Fig. 12, OPT accomplishes 32% to half expected power reserve funds concerning DVFSDPM-OPT, and 8% to 17% as for Sub-OPT when the assignments are with Gaussian conveyance. In Fig. 13, OPT accomplishes 52% to 66% expected power investment funds concerning DVFS-DPM-OPT, and 14% to 30% as for Sub-OPT when the undertakings are with exponential appropriation. In Fig. 14, OPT accomplishes 53% to 60% expected power funds concerning DVFS-DPM-OPT, and 21% to 27% as for Sub-OPT for reasonable applications. Adequacy of the refinement algorithm. The bin number is diminished from 20 to 5 in a critical position route and with the refinement calculation. Select system OPT is utilized to compute the numerical desire for power utilization. The exploratory outcomes, the refinement calculation accomplishes 4.5% to 5% expected power investment funds regarding the offset path with Gaussian circulation, 6.5% to 8.5% with exponential appropriation, and 4.9% with the sensible dispersion.

VIII. Future Work

In the future we will continue our research of minimizing energy consumption utilization by scheduling algorithms and also, we will be focusing on performance issues that arrive in homogenous multicore architectures.

References

[1] P. L. X. Y. C. Z. R. X. Xing Liu, "Energy Optimization and Fault Tolerance to Embedded System Based on Adaptive Heterogeneous Multi-Core Hardware Architecture," in *2018 IEEE International Conference on Software Quality, Reliability and Security* , Lisbon, Portugal, 2018.

[2] K. W. D. Z. Kai Huang, "Expected Energy Optimization for Real-Time Multiprocessor SoCs Running Periodic Tasks with Uncertain Execution Time," *IEEE Transactions on Sustainable Computing* , p. 14, 2018.

[3] Z. G. A. S. N. G. ASHIKAHMED BHUIYAN, "Energy-Efficient Real-Time Scheduling of DAG Tasks," *ACM Transactions on Embedded Computing Systems (TECS)*, p. 25, 2018.

[4] A. K. R. (. R. Alexei Colin, "Energy-Efficient Allocation of Real-Time Applications onto Heterogeneous Processors," in *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, Chongqing, China, 2014.

[5] T. A. A. a. H. Aydin, "Energy-Aware Task Allocation for Rate Monotonic Scheduling," in *11th IEEE Real Time and Embedded Technology and Applications Symposium*, San Francisco, 2005.

[6] Y. H. L. Changjiu Xian, "Energy-AwareSchedulingforReal-TimeMultiprocessor SystemswithUncertainTaskExecutionTime," in *2007 44th ACM/IEEE Design Automation Conference*, San Diego, CA, USA, 2007.

[7] J.-J. Chen, "Expected Energy Consumption Minimization in DVS Systems with Discrete Frequencies," in *SAC '08 Proceedings of the 2008 ACM symposium on Applied computing*, Fortaleza, Ceara, 2008.

[8] X. J. X. Z. KAI HUANG, "Energy-efficient Fault-tolerant Mapping and Scheduling on Heterogeneous Multiprocessor Real-Time Systems," *IEEE Access*, p. 17, 2018.

[9] A. C. K. R. (. Rajkumar, "Energy-Efficient Allocation of Real-Time Applications onto Single-ISA Heterogeneous MultiCore Processors," in *Journal of Signal Processing Systems*, 2015.

[10] B. K. R. Shubhangi K. Gawali, "DPS: A Dynamic Procrastination Scheduler for Multi-core/Multi-processor Hard Real Time Systems," in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, St. Julian's, Malta , 2016.

[11] H. Y. k. I. Takashi Nakada, "EnergyawareTaskSchedulingforNearReal-timePeriodicTasks onHeterogeneousMulticoreProcessors," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Abu Dhabi, United Arab Emirates, 2017.

[12] A. P. M. S. Santiago Pagani, "Energy Efficiency for Clustered Heterogeneous Multicores," *IEEE Transactions on Parallel and Distributed Systems*, vol. Volume: 28 , no. Issue: 5 , pp. 1315 - 1330, 2017.

[13] V. Yangchun lou, "Energy efficient speculative threads: dynamic thread allocation in Same-ISA heterogeneous multicore systems," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, Austria, 2010.

[14] S. Andrew Lukfahr, "Heterogeneous microarchitectures trump voltage scaling for low-power cores," in *Proceedings of the 23rd international conference on Parallel architectures and compilation*, Edmonton, 2014.

[15] A. K. Singh, M. Shafique and A. Kumar, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, 2013.

[16] R. Dennard, F. Gaensslen, V. Rideout and E. Bassous, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. Volume: 9 , no. Issue: 5 , pp. 256 - 268, 1974.

[17] M. B. Taylor, "A Landscape of the New Dark Silicon Design Regime," *IEEE Micro*, vol. Volume: 33 , no. Issue: 5, pp. 8 - 19, 2013.

[18] D. H. W. D. H. W. D. H. W. H. H. S. L. H. H. S. L. H. H. S. L. a. H. H. S. L. D. H. Woo, "Extending Amdahl's Law for EnergyEfficient Computing in the Many-Core Era. Computer,," vol. 41(12), pp. 24–31,, 2008.

[19] E. S. Chung, P. A. Milder, J. C. Hoe and K. Mai, "Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, Atlanta, 2010.

[20] J. S. V. Sparsh Mittal, "A Survey of CPU-GPU Heterogeneous Computing Techniques," in *ACM Computing Surveys (CSUR)* , NY, 2015.

[21] "Minimizing energy under performance constraints on embedded platforms: resource allocation heuristics for homogeneous and single-ISA heterogeneous multi-cores," in *ACM SIGBED Review - Special Issue on the 4th Embedded Operating Systems Workshop (EWiLi)*, 2014.

[22] M. P. D. M. a. A. M. E. Nogues, "Energy Efficient Scheduling of Real Time Signal Processing Applications through Combined DVFS and DPM.," in *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* , 2016.

[23] C. S. L. A. a. D. F. Juri Lelli, " Deadline scheduling in the Linux kernel.,," vol. 46, no. 6, June 2016..

[24] S. H. Fredric Hällis, W. Lund, R. Slotte and S. Lafond, "Thermal influence on the energy efficiency of workload consolidation in many-core architectures," in *2013 24th Tyrrhenian International Workshop on Digital Communications - Green ICT (TIWDC)*, Genoa, 2013.

[25] G. D. T. E. Del Sozzo, "Workload-aware power optimization strategy for asymmetric multiprocessors," in *DATE '16 Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, Dresdan, 2016.

[26] H. Rexha, S. Holmbacka and S. Lafond, "Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, St. Petersburg,, 2017.

[27] S. Holmbacka and R. Müller, "epEBench: True Energy Benchmark," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, St. Petersburg, 2017.

[28] L. E. Stijn Eyerman, "Fine-grained DVFS using on-chip regulators," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 1, 2011.