IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 17, 2021, accepted February 23, 2021, date of publication March 4, 2021, date of current version March 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063901

Energy Aware Parallel Scheduling Techniques for Network-on-Chip Based Systems

BICHI BASHIR YUSUF[1], TAHIR MAQSOOD[2], FAISAL REHMAN[2], AND SAJJAD A. MADANI[1]
[1]Department of Computer Science, COMSATS University Islamabad, Islamabad 45550, Pakistan
[2]Department of Computer Science, COMSATS University Islamabad at Abbottabad, Abbottabad 22060, Pakistan

Corresponding author: Tahir Maqsood (tmaqsood@cuiatd.edu.pk)

The work of Bichi Bashir Yusuf was supported by The World Academy of Science (TWAS) and COMSATS University Islamabad through the Award of 2016 CIIT-TWAS Full-Time Postgraduate Fellowship under Grant 3240293224.

ABSTRACT Minimizing execution time, energy consumption, and network load through scheduling algorithms is challenging for multi-processor-on-chip (MPSoC) based network-on-chip (NoC) systems. MPSoC based systems are prevalent in high performance computing systems. With the increase in computing capabilities of computing hardware, application requirements have increased many folds, particularly for real world scientific applications. Scheduling large scientific workflows consisting hundreds and thousands of tasks consume significant amount of time and resources. In this article, energy aware parallel scheduling techniques are presented primarily aimed at reducing the algorithm execution time while considering network load. Experimental results reveal that the proposed parallel scheduling algorithms achieve significant reduction in execution time.

INDEX TERMS Network-on-chip (NoC), multiprocessor system-on-chip (MPSoC), task scheduling, parallel scheduling.

I. INTRODUCTION

Multi-processor system-on-chip (MPSoC) technology allows multiple processing elements of varying capabilities and capacities to be embedded onto a single chip. MPSoCs consist of tens or hundreds of heterogeneous intellectual property (IP) cores capable of running multiple parallel applications [1], [2]. Often known as Processing Elements (PEs), the cores can be described as a general purpose processors, a field programmable modules, or a digital signal processors [3], [4]. MPSoC framework retained standardized tiled infrastructure composed of different heterogeneous compute processors, memory tiles for global memory access, dedicated accelerator tiles, and I/O tiles [5]–[7].

Improving energy efficiency of MPSoCs is important to realize the sustainable adoption of these systems in commercial and household devices [7]. According to [8], [9], energy consumption of high-performance multiprocessor systems including cloud datacenter, IT industry, and telecommunication equipment may rise up to 26 GW worldwide. Moreover, increase in energy consumption leads to increase CO_2 emissions that contributes to the global warming. Studies also revealed that majority of the servers within datacenters operate in the range of 15% to 30% of their total capacity [9], [10]. This underutilization comes with a cost as a study conducted by [11] shows that the operations of datacenters deployed by large IT service providers, such as Google and Microsoft accounts to a yearly cumulative spending of nearly $30 million for energy.

Improving MPSoCs efficiency is critical in achieving the sustainable deployment of these systems in industrial and residential devices that deals with large workflows. The MPSoCs continue to develop and expand in terms of processing power and core count. The core count of MPSoC is expected to hit hundreds, or even thousands by 2025 [12]. This growth comes with a cost, i.e., increased execution time and energy consumption. Workflow scheduling is already proven to be NP-hard problem [13]. Consequently, scheduling consumes significant amount of time, energy, and computational resources. In the literature, various schemes have been proposed to optimize the energy efficiency of MPSoCs [1], [2], [12], [14]–[16]. Most scheduling algorithms are developed using a linear or sequential approach, which consumes considerable time and, consequently, increases energy consumption of MPSoCs when applied to large workflows with hundreds of thousands of

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.

tasks. For instance, techniques designed to look for optimal or near-optimal solutions, such as integer linear programming (ILP), genetic algorithm (GA), ant colony optimization (ACO), and Pareto efficient algorithm have considerably higher execution time to achieve convergence, typically hours or even days [1], [2], [15]. Therefore, it is highly desirable to improve performance and energy efficiency of MPSoCs due to significant increase in computational demands of current and future application requirements particularly large scientific workflows.

The major contributions of this work are summarized as follows.

- Optimize the task scheduling algorithms namely b-level based (BL) scheme, HEFT ranking b-level scheme (HRBL), energy efficient scheme (E3FT), and base level task stealing (BLTS) algorithm by parallelizing different phases of the algorithms.
- Reduce execution time and energy consumption of scheduling algorithms.

The rest of this paper is structured in the following manner. In Section II related work is discussed. Section III deals with problem formulation and system model. The proposed parallel scheduling techniques are presented in Section IV. The experimental evaluation and results are discussed in Section V. Finally, Section VI concludes the major findings of this work with a glimpse of the future work.

II. RELATED WORK

Scheduling algorithms have a profound impact on MPSoC systems performance, energy efficiency and reliability [12, 16]. Numerous task mapping/scheduling schemes have been proposed for MPSoC based systems using different optimization techniques, such as integer linear programming [2], genetic algorithm [17], and ant colony optimization [15]. The technique proposed by [2], aims to minimize energy consumption of heterogeneous MPSoCs. Another technique proposed by [17] used genetic algorithm to improve energy efficiency and achieved higher bandwidth for the NoC-based MPSoC systems. The authors in [15] and [18] developed an ant colony optimization (ACO) based scheme with the aim of minimizing computational overhead of MPSoCs. The aforementioned strategies show promising results in terms of reducing energy consumption. However, a major drawback of these systems is that they are computationally intensive and have significantly higher execution time to achieve convergence [1, 2, 19] when dealing with large workflows of thousands or millions of jobs [1].

To this end, we further discuss on the schemes that are firmly related to this work. A dynamic task mapping technique for NoC based systems was proposed by the authors in [1]. The authors extended the nearest-neighbor based communication-aware packing algorithm (CPNN). Proposed extension aimed to reduce the communication overhead. In addition, a migration scheme was proposed that allows highly loaded PEs to move the task to lightly loaded PEs that can host the task without raising communication overhead. Similarly, the authors in [20] presented a method to optimize energy and communication by defining a novel method for bin packing problem namely Variable bEnefit Bin pAcking Problem (VEBAP).

In [21], the authors proposed different algorithms to tackle the issue of energy efficient task and data co-scheduling using different heuristics. The greedy algorithm traverses the task graph in breadth-first order to reduce makespan and energy. Critical path-based algorithm works by prioritizing the scheduling of tasks that are on the critical (longest) path within a given directed acyclic graph (DAG). The base level algorithm schedules tasks and the generated data considering base level priorities of tasks. The task assignment considering data allocation (TAC-DA), involving two-way operational stages, is another scheme presented in [22]. First, the critical path algorithm is utilized to find a suitable appropriate way of mapping tasks. Then the second phase involves gathering data items to maximize energy consumption, considering the time and task mapping constraints of the preceding phase. Task ratio greedy scheduling scheme (TRGS) deals with data allocation when scheduling tasks. The scheme allows each data element to be assigned to a specific processor's memory/local cache [22].

The authors in [23] proposed the idea of using two different types of PEs, i.e., special purpose and general-purpose PEs. The objective of the work is to achieve a transient fault aware mapping and scheduling (TFAMS) scheme that satisfies the deadline requirement of task as well as minimizes energy consumption during allocation. The proposed scheme attempts to dynamically adjust task mapping whenever a fault is experienced. However, the proposed scheme does not tackle permanent faults. Moreover, communication overhead will increase while remapping tasks. Furthermore, technique proposed in [23] assigns tasks dynamically and it is characterized by low utilization. Reference [24] focuses on integrating task mapping, task ordering and voltage scaling. To take advantage of the different computational resources available on a given system, researchers in [17], [25], [26] propose parallel genetic algorithms (PGA). In [17] PGA measures the fitness value of each person in a single population leveraging highly parallel processors, such as the master-slave-based GA platform. The master processor allocates the individuals among the slave processors. In this architecture, the master processor becomes the bottleneck for the whole system. Parallel ant colony optimization (PACO) [27] is an evolutionary optimization algorithm. The authors presented a technique for sharing information among processors by enabling each processor to select a partner to associate with their pheromone and adaptively update it. Secondly, a framework is developed to adapt the timeframe for exchanging information based on variety of solutions and enhance the consistency of the optimized solution as well as prevent early convergence. Table 1 summarizes the works discussed in related work.

In terms of modalities and implementation, majority of the strategies discussed in the related work are identical. For

TABLE 1. Comparison of techniques discussed in related work.

Technique	Classification	Criteria	Strengths	Limitations
Dynamic Task Mapping for Network-on-Chip [1] Energy and communication aware task mapping for MPSoCs [20]	Sequential	Reduce energy and communication overhead	Uses heuristics-based approach and dynamic programming to achieve the objectives	Some PEs can be overloaded
Task Mapping and scheduling for Network-on-Chip Multicore platforms with Transient Fault Tolerant [23]	Sequential	Tasks are assigned to earliest available PE	The proposed scheme attempts to dynamically adjust whenever a fault is experienced	Static approach as application characteristics must be known prior to scheduling
Leveraging on Deep Memory Hierarchies to Minimize Energy Consumption and Data Access Latency on Single-Chip Cloud Computers [28]	Sequential	Uses efficient heuristics to optimize system wide energy consumption	A holistic approach to reduce energy consumption and makespan	Slow performance when dealing with large workflows
Dynamic mapping of quality adjustable applications on NoC-based reconfigurable platforms [29]	Sequential	Employs a full search optimization technique to obtain solution for a simple problem with small number of applications	The approach effectively improves simultaneous mapping of application and resources	The scheme cannot be applied to different reconfiguration region
Energy Efficient Heuristic Algorithm for Task Mapping on Shared-Memory Heterogeneous MPSoCs [30]. Energy Efficient Task Mapping & Scheduling on Heterogeneous NoC-MPSoCs in IoT based Smart City [24]	Sequential	The mitosis heterogeneous genetic algorithm (MH-GA) works in three stages. Scheduling and voltage assignment, fitness function and crossover, and mutation stages.	There is an improvement in efficiency with regards to the MH-GA scheme by 21.5%	Using the fitness functions some PEs remain under-utilized while they are capable of processing more tasks
An Efficient Network-on-Chip (NoC) based Multicore Platform for Hierarchical Parallel Genetic Algorithms [17]	Parallel	Master-slave based parallelization of genetic algorithm, and hierarchical parallel algorithm	improves the execution speed by employing multiplexing techniques	Fixed bandwidth of master limits speedup as well as the low utilization of cores within the slave processor
A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem [27]	Parallel	Segments sequential ACO into several mutually independent subordinate colonies	The scheme has a high speed of convergence	Communication throughout PACO contributes to lower solution quality and more latency in PACO generated by the whole pheromone matrix.

example, the work in [1], [20], [23], [28], [29], [31], [24], [28] and [30] are based on the sequential mapping approach. The sequential schemes have high computational complexity and consume significant amount of time and resources, especially when dealing with large workflows, such as brain imaging workflows, health care data analysis, and Pegasus workflow that tackle thousands of tasks [32]–[34]. Alternatively, the parallel schemes presented in [17], [27] show significant improvement in terms of reduced algorithm execution time. However, their performance for large workflows is not known

III. PROBLEM STATEMENT AND SYSTEM MODEL

A. WORKFLOW MODEL

Workflow scheduling problem in heterogeneous multi-processor systems can be modeled as follows. A workflow can be depicted using a directed acyclic graph (DAG) $G(T, E)$. Here T is set of n tasks as $T = \{t_1, t_2, t_3, \ldots, t_n$ with E representing set of directed edges, $E = \{e_1, e_2, e_3, \ldots, e_m$. The relationship between task t_i and t_j creates an edge that maintains the precedence constraint between tasks i and j depicting a parent and child relationship between any two

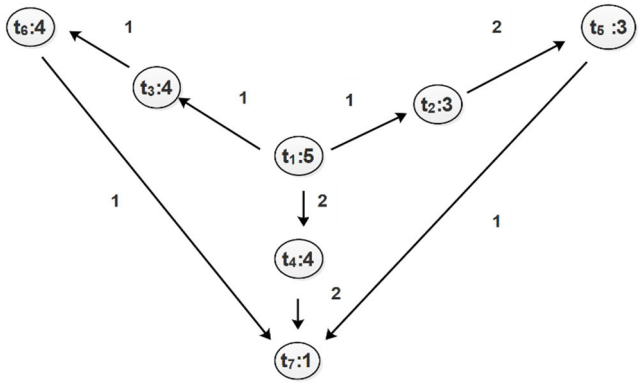

FIGURE 1. Application workflow example.

given nodes. Fig. 1 shows a sample DAG with relationship between tasks. Here t_1 is parent and t_2, t_3, and t_4 are child tasks of t_1. Task that does not have parent is referred to as an entry task while the task with no child is called exit task. A DAG may consist of multiple entry and exit tasks.

B. ENERGY MODEL

Energy model of PowerPC 405-based architecture implemented on the Xilinx Virtex II Pro FPGA (XupV2Pro) platform have been used. PowerPC 405 processors with a 16KB, 2-way set of associative instruction and data caches are included in the Virtex II Pro FPGA. The proposed energy model considers the computational energy consumed by processing cores and NoC components. Equation (1) gives the dynamic power Dp_i dissipated in the execution of a task t_i on a PE p_j as employed in [16].

$$Dp_i = L_c * V_{s_i}^2 * f_i \tag{1}$$

Here L_c denotes the effective load switching capacitance, V_{s_i} the supply voltage, and f_i operating frequency accordingly. The energy of NoC-based systems comprises of two major components: (a) energy consumption by processor referred as E_p, and (b) communication energy consumption E_c. Energy model used in this work is adopted from [1], [15], [35]. Communication energy represents the amount of energy being used by components of the NoC to transfer data among the PEs. The communication energy calculation is based on the bit energy model proposed in [1], [35]. The total energy E_T that a system consumes while executing a given workflow is given by equation (2).

$$E_T = E_c + E_p \tag{2}$$

E_p represents the energy consumed by the processing elements for executing the workflow. While E_c represents the communication energy consumed when transmitting data over the NoC. The communication energy is measured using bit energy as stated in [36], which determines the energy required for transporting one bit from tile j within the NoC to tile k as shown in equation (3).

$$En_{bit} = \left(n_{jk} * Er_{bit}\right) + \left(n_{jk} - 1 * El_{bit}\right) + (2 * Ec_{bit}) \tag{3}$$

En_{bit} is the energy that is required to transfer one bit from tile j to tile k in the NoC. Er_{bit} indicates the dynamic energy required to pass a bit through the router (wires, buffers, and logic gates). El_{bit} represents the dynamic energy consumed in moving the bit on the links across two adjacent grids as shown in equation (4).

$$El_{bit} = \frac{P_i}{f_l * b_w} \tag{4}$$

where P_i shows the total power consumption for one bit on the links that t_i traverses at f_l, while b_w represent the bandwidth of each link within the 2D NoC mesh.

Ec_{bit} indicates the dynamic energy consumed between router to PE link. Lastly, the number of routers or hops that the bit traverse from grid j to grid k is computed by using equation (5) where x_j and y_j are row and column indices of p_j on a 2D NoC mesh. Consequently, the total communication energy incurred within NoC can be measured using the equation (6) with N_{bit} representing total communication volume in number of bits and n_{avg} is the described as the number hops travelled by all bits on average.

$$n_{jk} = |x_j - x_k| + |y_j - y_k| \tag{5}$$

$$E_c = N_{bit} + \left(n_{avg} * Er_{bit}\right)\left(n_{avg} - 1 * El_{bit}\right) + (2 * Ec_{bit}) \tag{6}$$

Equation (7) represents the computational energy consumed by a given PE p_j with n_j^r as the number of cycles where p_j is running and n_j^i is when p_j is idle. E_r and E_i describes the energy consumed by the PE in running and idle states, accordingly. The overall consumption in terms of computational energy by all the PEs is calculated using equation (8).

$$E_j = n_j^r * E_r + n_j^i * E_i \tag{7}$$

$$E_p = \sum_{j=1}^{n} E_j \tag{8}$$

C. ARBITRATION

Multiprocessor systems' performance is dependent on the efficient communication between processing cores and balanced allocation of computation between them rather than solely on processor speed [37]. Round robin arbitration is a technique that maintains a high degree of fairness among agents by assessing each input port fairly and ensuring continuity in scheduling. An equal opportunity to reach the output port is given to each input port, this process helps in solving the problem of starvation [38].

In the mesh architecture, all routers have four adjacent routers and a PE attached to a local port with the exception of the routers at the borders. Major benefit of this system is scalability, which enables large number of PEs to be connected. The same output port can be appropriate for several packets arriving at various input ports. As such, the output port must be scheduled to ensure equal utilization of the output channel by the input channels. Scheduling must be carried out to ensure that no packet waits to be guided to its target

indefinitely. The router arbiters are responsible for managing the use of the output port [39]. This work uses distributed round robin arbitration, where each router port manages the routing and arbitration separately. There is a routing module connected with each input port, and each output port has an arbitrator. A specific output port is demanded by the routing module. The arbitrator chooses a specific input port if it has received several requests.

D. PROBLEM STATEMENT AND FORMULATION

Researchers have proposed several scheduling techniques for NoC based MPSoCs. However, a large proportion of the solutions focus on sequential strategy. Consequently, scheduling algorithms consume a significant amount of time and energy, especially when dealing with large workflows, such as brain imaging workflows, healthcare data analysis, service-oriented architectures, and so on having hundreds of thousands of tasks.

To mathematically formulate the problem at hand, task to processing cores mapping is represented by $|T| \times |P|$ matrix denoted by S. For the matrix S, $s_{k,x}$ demonstrates whether task t_k is placed on processor p_x or not. Specifically, when t_k is assigned to p_x, $s_{k,x}$ is one, otherwise it is zero. The utilization indicated by $Ur_x\,(S,t)$ of the processor p_x is given by equation (9) with CC_x representing the computing capacity of p_x and CR_k representing computing requirement of task t_k. It is worth noting that the processor's utilization rate cannot be greater than one as indicated in equation (10). The overall network load N_l incurred as a result of communication among interdependent tasks within the system in a given time interval (t_1 to t_2) is calculated using equation (11).

$$Ur_x\,(S,t) = \sum_{k=1}^{T} s_{k,x} * \frac{CC_x}{CR_k} \tag{9}$$

$$\sum_{k=1}^{T} s_{k,x} * \frac{CC_x}{CR_k} \leq 1, \quad \forall x \epsilon P \tag{10}$$

$$N_l = \sum_{k=1}^{T}\sum_{m=1}^{T}\sum_{x=1}^{P}\sum_{j=1}^{P} S_{k,x} * S_{m,j} * n_{kj} \tag{11}$$

The main objective of the task scheduling algorithm is to minimize energy consumption of the system which is reflected in objective function defined in equation (12).

$$minimize\ E_T = \sum_{j=1}^{n} E_j + E_c \tag{12}$$

This problem is subject to following constraints. (a) tasks allocated to any processing core p_j are processed in the first-come-first-serve sequence without breaching the execution order. (b) Task t_i, $1 \leq i \leq x$ is executed without preemption on only one core and is not migrated and split at any stage. (c) the processor p_j for $1 \leq j \leq y$ can house as many tasks at the same time as long as the utilization does not exceed one.

IV. PROPOSED SCHEMES

The objective of this work is to develop algorithmic solutions using parallel approach to reduce the execution time of scheduling algorithms for large workflows. Parallel scheduling algorithms can overcome the drawbacks of the sequential scheduling approach. However, achieving parallelism in scheduling algorithms is not trivial and introduces new challenges that need to be considered while developing parallel scheduling schemes. One of the major challenges is that certain data structures are shared by parallel threads and may need to be accessed and modified simultaneously, which leads to synchronization issues. Moreover, loading and pre-processing of large workflow graphs having hundreds of thousands of tasks consumes substantial amount of time that contributes towards increasing execution time of the scheduling algorithm as well as energy consumption. Therefore, in this work multiple parallel scheduling algorithms are proposed considering the aforementioned challenges.

NP-Hardness: Task scheduling for the general case is considered to be NP-hard [35]. Task scheduling problem is solved for finite or infinite number of processing cores. In case of finite number of processing cores, the objective is to find an algorithm and verify its schedulability that the given workflow is schedulable (or otherwise) on given fixed number of processing cores. This is analogous to the classical bin packing problem where certain items need to be packed in fixed number of bins having specified capacity [40]. Here, items (tasks) with certain weight (computing requirements in case of tasks) are required to be packed (scheduled) onto minimum possible number of bins (processing cores) with the constraint that weight of items packed in each bin must not exceed the capacity of bins. Since bin packing problem is NP-hard [40], [41], the task scheduling problem at hand is also NP-hard for the general case [40], [42], [43]. Moreover, scheduling for energy minimization is proved to be NP-hard problem. For detailed proofs on NP-hardness of the task scheduling problem, the readers are encouraged to see [43]–[45].

Since the energy efficient task scheduling on multicores is an NP-hard problem, therefore, in this work heuristics based parallel scheduling algorithms are proposed. Algorithms proposed as part of this work are: (a) modified base b-level (BL) scheme, (b) HEFT ranking base level (HRBL), (c) energy efficient technique (E3FT), and (d) modified base level task stealing (BLTS). Each of the proposed scheme is discussed in the subsequent sections and a generic flowchart of proposed task scheduling methodology is depicted in Fig. 2.

A. MODIFIED BASE LEVEL SCHEME (MBL)

The scheme uses b-level priorities when scheduling tasks. The b-level value of a task is determined by identifying the longest path from the task under consideration to the exit task. The b-level value for the task is measured as the sum of the computation and communication costs from the particular task to the exit task along the critical path. Scheduling is

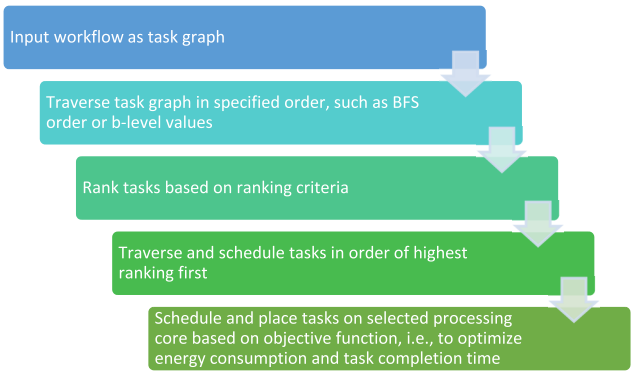

FIGURE 2. Generic flowchart of task scheduling methodology.

Algorithm 1 Modified B-Level Scheme

```
Input: Task set T, processing cores P
1. Parallel estimation of b-level priority
   for each task t_i ∈ T
2. Parallel sorting of tasks in non-increasing
   order of the b-level values
3. For each task t_i ∈ T
4.  p_x = compute PE that report minimum
    completion time of a task t_i
5.    begin assignment
6.      assign t_i to p_x
7.    end assignment
8. End For
```

conducted by assigning tasks according to their b-level value in decreasing order to a PE that reports minimum completion time. Following modifications have been applied to the BL scheduling technique. (a) Parallelize the calculation of b-level values of all tasks as indicated in line 1 of algorithm 1. (b) Modified the sorting phase by applying parallel sorting technique in algorithm 6 to reduce the execution time (line 2). The adopted parallel sorting technique is described in detail in the subsection E.

B. HEFT RANKING BASE LEVEL SCHEME (HRBL)

The HRBL scheme combines the feature of Base Level task priority algorithm and HEFT algorithm, consequently, named as HEFT ranking base level (HRBL). The advantage of using the algorithm (ranking algorithm) is that two dependent tasks can be allotted on the same PE that leads to reduced schedule length. Similar to the HEFT scheme, the HRBL consists of two phases: (i) the priority phase that chooses tasks with higher priority among the set of tasks, and (ii) the process phase schedules the task with higher priority to a PE with least finishing time for the given task.

Task Priority: The priority of each task is calculated using the upward rank value using equation (13) as proposed in [49, 50], which uses the average communication and computation

Algorithm 2 Ranking Algorithm

```
Input: Task set T, processing cores P
1 W = calculate on average the execution
  time for each task on all PE
2 For task t_i ∈ T in the DAG graph
3   IF t_i is the last task
4     Rank of t_i
      = average task execution time
5   Else
6    Compute raking of t_i using
     Equation (13)
       R_u(t_i) = W_i + max_{t_j ∈ s(t_i)} (C_{i,j} + R_u(t_j))
7   End IF
8 End For
```

cost. The list of tasks generated is then sorted in decreasing order of the task based on the rank computed for each task.

$$R_u(t_i) = W_i + \max_{t_j \in s(t_i)} (C_{i,j} + R_u(t_j)) \tag{13}$$

Here $s(t_i)$ represents set of successors task of t_i, W_i represents computation cost on average calculated using equation (14). In (14) w_{ij} gives the estimated execution time of t_i on PE p_j and q represent the total number of processing entities. In equation (13) $C_{i,j}$ represents the average communication cost of task t_i to t_j. $C_{i,j}$ is calculated using equation (15) where $C_{i,j}$ is the communications cost of transmitting data from task t_i scheduled on PE p_m to task t_j scheduled on p_n along edge (i, j). R_u in (13) represents the longest path of a given task t_i to its exit node along with its computational cost.

$$W_i = \sum_{j=1}^{q} {w_{ij}}/{q} \tag{14}$$

$$C_{i,j} = l_m + {data_{i,j}}/{B_{m,n}} \tag{15}$$

Here B is a matrix of size qxq representing data transfer rate between PEs and $B_{m,n}$ gives the transfer rate between PE m and n. In (15) l_m represents the communication startup cost of a given PE_m. $data_{i,j}$ gives the amount of data shared between tasks t_i and t_j.

PE selection: At this stage of scheduling, the scheme searches for a PEs that report earliest finish time for each task in the order of their priorities calculated using algorithm 2. We then further modify the ranking scheme to compute the task ranking, the sorting process to perform in a parallel manner as outlined in algorithm 3.

C. ENERGY EFFICIENT EARLIEST FINISHING TIME SCHEME (E3FT)

The central idea of E3FT scheme is to schedule tasks on PEs reporting minimum completion time while considering the energy consumption of PEs. In E3FT, PEs are sorted according to the energy consumption in increasing order. The E3FT technique uses the b-level priority of a given tasks

Algorithm 3 Modified HRBL Scheduling

```
Input: Task set T, processing cores P
1 Parallel Computation of task ranks
2 Parallel sorting of tasks in
  non-increasing order of the b-level
  values
3 For each task t_i ∈ T
4    p_x = compute PE that report
     minimum completion time of a
     task t_i from matrix W
5   Assign t_i to p_x
6 End For
```

similar to the BL scheme [20]. The b-level value of a task is determined as discussed in section 4.1. The E3FT scheme shown in algorithm 4, traverses the task list in the sorted order. For each task t_i, earliest finishing time of t_i is computed using equation (16). The task t_i is scheduled on PE that reports minimum completion time, if there exist a case of tie, i.e., when there are more than one PEs with same minimum completion time, then the PE with lower energy consumption is selected. In case of tie again, when there are more than one PEs with same energy consumption, then the PE with lower utilization is selected. To reduce the algorithm execution time, certain steps of the algorithm are executed in parallel that include line 1, 2, and 3 as indicated in algorithm 4. To compute the completion time (CT) we first consider the earliest start time (EST) of task t_i based on the CT of the predecessor tasks as shown in equation (16), where $CT\left(t_j, p_z\right)$ represents the completion time of task t_j on processor p_z. Therefore, the completion time of a task t_i for a given PE p_z is computed using the formulation presented in equation (17). Here ET_{iz} denotes the time required to execute task t_i on p_z. Note that equation (17) only holds when task t_i has more than one predecessor.

$$EST\left(t_i\right) = \min_{\forall p_z \in P} \max_{\forall t_j \in Pred(t_j)} CT\left(t_j, p_z\right) \tag{16}$$

$$CT\left(t_i\right) = \min_{\forall p_z \in P} \{EST\left(t_i\right) + ET_{iz}\} \tag{17}$$

D. MODIFIED BASE LEVEL TASK STEALING SCHEME (MBLTS)

This scheme is a modified BL algorithm, the task stealing method focuses on finding out an idle slot created on any given processing core after a task has been scheduled. Upon finding idle slot, then the algorithm tries to identify a suitable task using a precedence level (p-level) value. The p-level value for a given task $\boldsymbol{t_i}$ is computed by taking into account the number of edges along path starting at the entry task till task $\boldsymbol{t_i}$. Two separate sorted lists are maintained for b-level and p-level values of all tasks involved. To improve the sorting performance, parallel sorting algorithm is used. After the sorting phase, tasks are traversed in the sorted order of b-level values. If $\boldsymbol{t_i}$ is found to be a join task, the task is then scheduled on the same PE along with its predecessor that has the top or

Algorithm 4 Modified Energy Efficient Earliest Finishing Time Scheme (ME3FT)

```
Input: Task set T, processing cores P
1  Parallel sorting of PEs on energy
   consumption and group PEs as per
   their energy consumption
2  Parallel computation of b-level
   priority for all the tasks within
   the Task-DAG
3  Parallel sorting of tasks in order
   of decreasing b-level values
4  For each task t_i ∈ T
5    List p_x = compute PE(s) that report
     minimum completion time (MCT)
     using (16) and (17)
6     IF there are more than one PE
      in px
7       list p_xm = PE(s) having
        lowest energy consumption
        among px
8          IF there more than one PEs
           in p_xm then
9           list p_xmu = PE(s) with
            lowest utilization among
            p_xm
10             IF there are more than
               one PE(s) in p_xmu then
11                p_z = randomly select
                  a PE among p_xmu
12             End IF
13          Else
14             p_z = p_xm
15       Else
16          p_z = p_x
17    End IF
18    assign t_i to p_z
19 End For
```

highest values for $\min_{\forall p_z P \&\& p_z \neq f(j)} \left\{\boldsymbol{CT}\left(\boldsymbol{t_j}, \boldsymbol{p_z}\right)\right\}$ else the task will be scheduled on PE which provides minimum completion time (MCT) for $\boldsymbol{t_i}$. Algorithm 5 presents the modified BLTS algorithm.

E. TASK SORTING

Sorting is extensively used in all the presented schemes. To optimize the sorting process we developed a parallel sorting technique based on the well-known merge sort algorithm [48]. Merge-sort algorithms applies the divide-and-conquer principle. A key benefit of using the merge sort in the context of large workflow is that it enables quicker sorting because it does not sort the complete list repeatedly. Another befitting factor is that the average execution time of the algorithm is consistent as different stages of the sorting scheme are executed in a similar time factor. We add a

Algorithm 5 Modified Base Level Task Stealing Scheme (MBLTS)

Input: Task set T, processing cores P

1 *b_list* = Parallel computation of b-level priority for all the tasks in T
2 *p_list* = Parallel computation of p-level values for all the tasks T
3 Parallel sorting of tasks in non-increasing order in *b_list* and *p_list*
4 For task $t_i \in b\_list$
5 If t_i is a disjoined node
6 $t_a = \max_{\forall t_j \in Pred(t_j)} \min_{\forall p_z \in P \&\& p_z \neq f_j} \{CT(t_j, p_z)\}$
7 p_z = PE that holds t_a
8 Else
9 p_z = PE with MCT
10 assign t_i to PE
11 End If
12 If there exist idle slot(s) then
13 *p_list* = tasks which have same value of p-level that are not scheduled
14 For each task t_p in *p-list* in non- increasing order of the value of b-level
15 IF the size of $t_p \leq idle$
16 $t_i = t_p$
17 GOTO 9
18 End If
19 End For
20 End If
21 End For

parallel sorting strategy as presented in [48] to the scheduling schemes mentioned earlier. The parallel sorting algorithm (algorithm 6) is called by different techniques whenever a list needs to be sorted. The scheme as shown in algorithm 6, works by dividing the unsorted lists into the smallest possible sub-lists, and each sub-list is processed by a separate processing core, relative to the adjacent task and merged in sorted order. For N number of items and M number of processors, the parallel merge sort scheme proceeds in two stages. Firstly, the local sorting stage then the merging stage. Line 3 of the algorithm allows all the involved processing cores to perform parallel sorting of the unsorted list accordingly. The local sorting method provides items that are locally sorted in each processor while the processors merge the sorted list keys into some steps described as $logM$ steps in the second stage. The processor cores are grouped into sender and receiver, whereby in the first step every sender transmits its list of N/M keys to its recipient. Each recipient then merges the two lists to create a sorted list of $2N/M$ items.

Algorithm 6 parallel Sorting Technique

Input: unsorted list
Output: sorted list

M: the total number of processing cores
M_i: processing core having index i
n: number of all active processing cores
1 $n := M$
2 For all $0 \leq i \leq M-1$
3 M_i *locally* sorts list of N/M items
4 End For
5 For $j = 0 to (logM) - 1$
6 For all $0 \leq i \leq n-1$
7 If $(i < n/2)$ then
8 M_i accepts N/n keys from $M_{i-n/2}$
9 M_i merges two lists of N/n items in $2N/n$ sorted list
10 Else
11 M_i transmits its list to $M_{i-n/2}$
12 End If
13 *n: = n/2*
14 End For
15 End For

During the next stage, only the receivers in the previous step are paired as (sender and receiver) and every pair executes the same process and same merge operations to form a set of $2^2N/M$ items. The method continues until a full list of N sorted items is achieved. The performance analysis for the algorithm is revealed as $M/logM = M/2 + M/4 + M/8 \ldots + 1)/(logMsteps)$. Therefore, complexity of the parallel merge sort at its worst case can therefore be given as (work/span) which is estimated as $O(M/\log M)$.

V. RESULTS AND DISCUSSION

A. EXPERIMENTAL SETUP

Extensive simulations are performed to analyze performance of proposed algorithms on a number of mesh sizes spanning from 20×20 to 80×80 mesh NoCs. The results are averaged over four different sets of application graphs for each NoC size, where the number of tasks vary from 2000 to 100,000 on average based on the size of the NoC. The information about the number of tasks per simulation scenario are described in Table 2.

B. RESULT AND DISCUSSION

This section compares the performance of proposed sequential algorithms namely HRBL and E3FT with the existing BL and BLTS techniques proposed in [28] in the literature. Moreover, proposed parallel scheduling algorithms presented in Section IV including: MBL, MHRBL, ME3FT, and MBLTS are compared with the existing parallel genetic

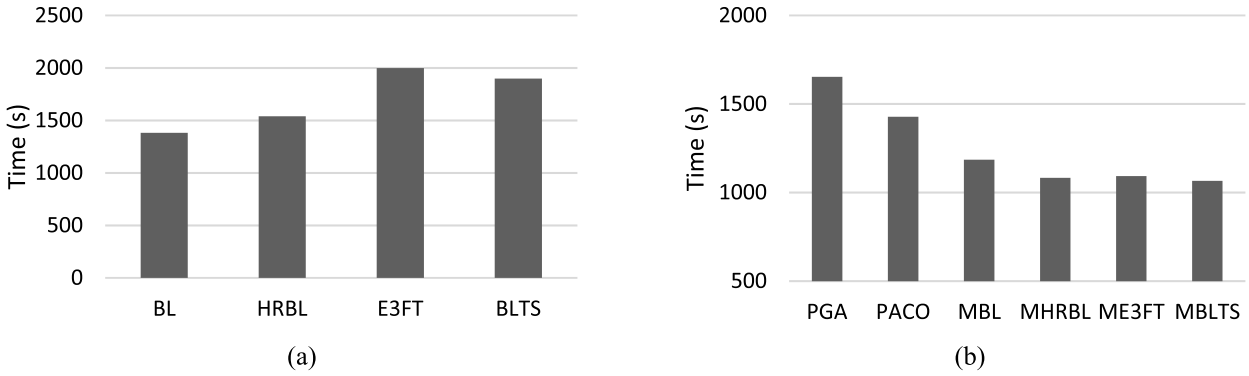

FIGURE 3. Consolidated run time for all scheduling scheme (a) Non-parallel (b) Parallel.

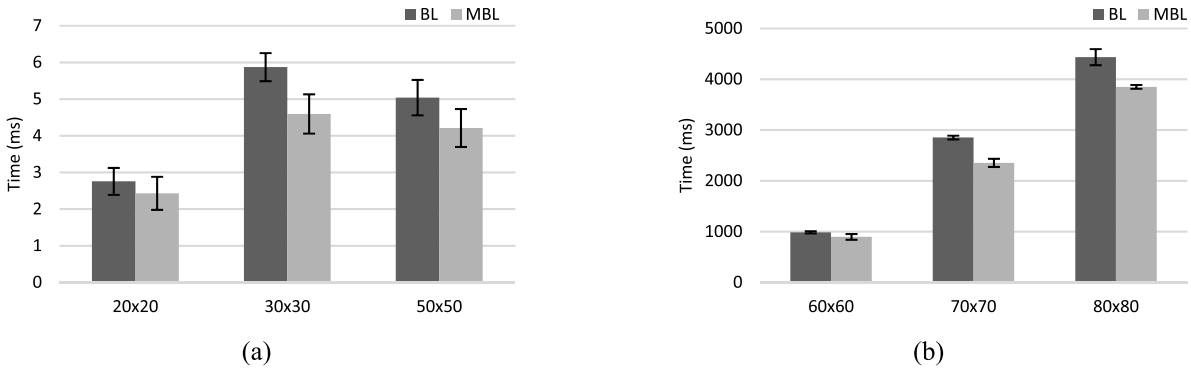

FIGURE 4. Run time for BL and MBL scheduling scheme (a) small mesh size (b) large mesh sizes.

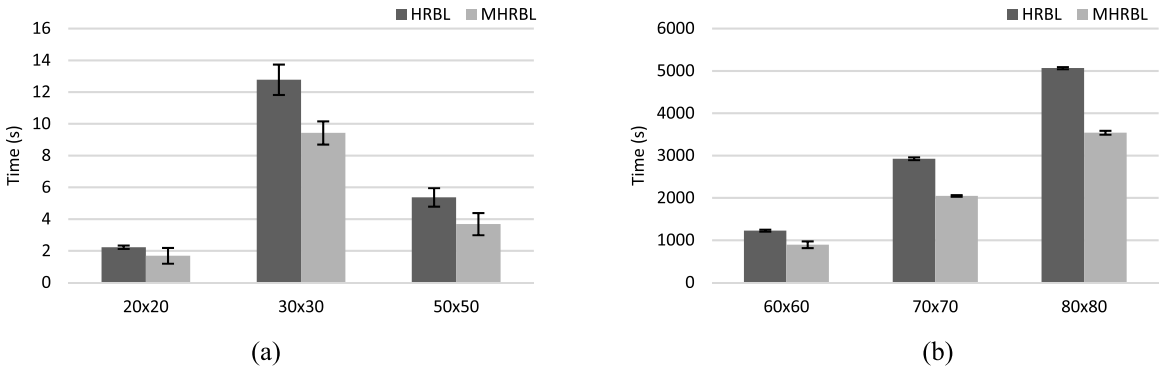

FIGURE 5. Run time for HRBL and MHRBL scheduling scheme (a) small mesh size (b) large mesh sizes.

algorithm (PGA) and parallel ant colony optimization (PACO) [17], [27]. The metrics used to measure the performance of the algorithms include: (a) execution time (b) energy consumption, and (c) network load. The results have been grouped into two categories based on NoC sizes and workflows. The smaller size workflow consisting of 2000 to 5000 tasks with an NoC mesh size between 20 × 20 to 50 × 50, respectively. Similarly, large workflows consisting of tasks between 30,000 to 100,000 with a NoC mesh size of 60 × 60 to 80 × 80, respectively. It must be noted that we use the acronyms MBL, MHRBL, ME3FT and MBLTS to represent the parallel schemes against the non-parallel techniques BL, HRBL, E3FT, and BLTS, respectively.

C. EXECUTION TIME

Fig. 3(a) shows the comparison of average execution time consolidated over all the mesh sizes, i.e., 20×20 to 80×80 of

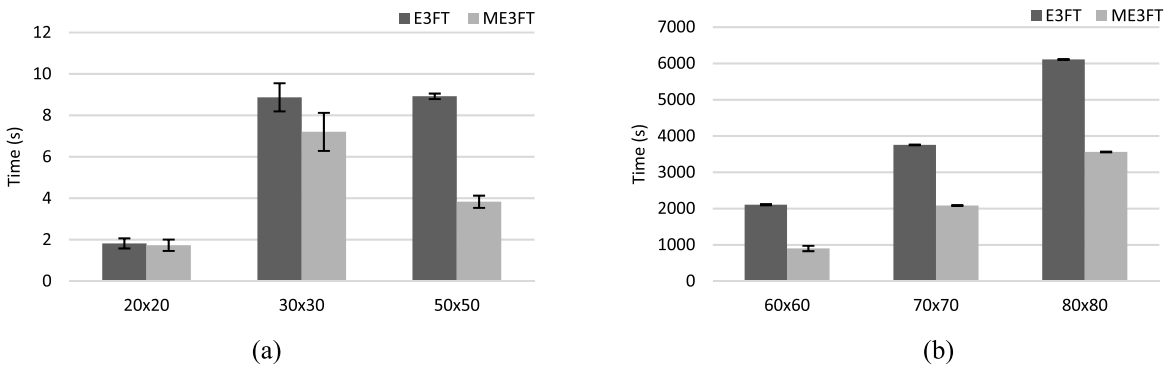

FIGURE 6. Run time for E3FT and ME3FT scheduling scheme (a) small mesh size (b) large mesh sizes.

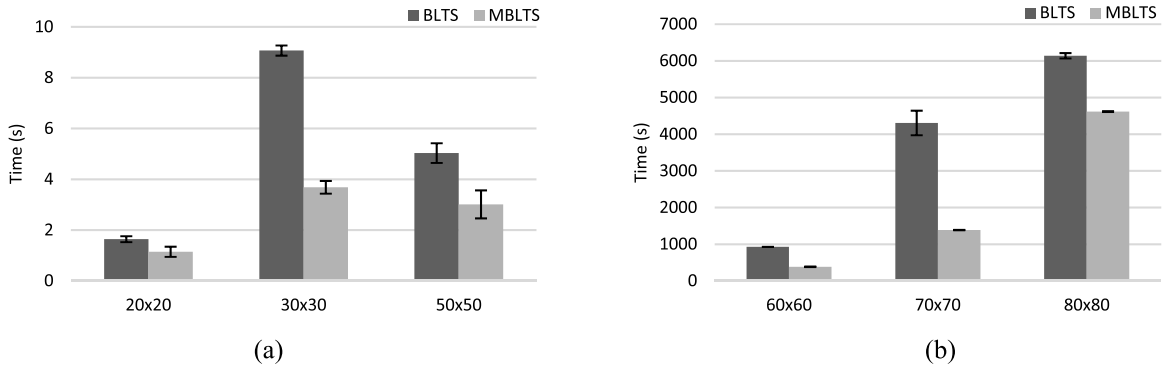

FIGURE 7. Run time for BLTS and MBLTS scheduling scheme (a) small mesh size (b) large mesh sizes.

TABLE 2. Details of experimental parameters.

Mesh NoC Size	Number of Task
20x20	2000
30x30	3000
50x50	5000
60x60	30,000
70x70	60,000
80x80	100,000

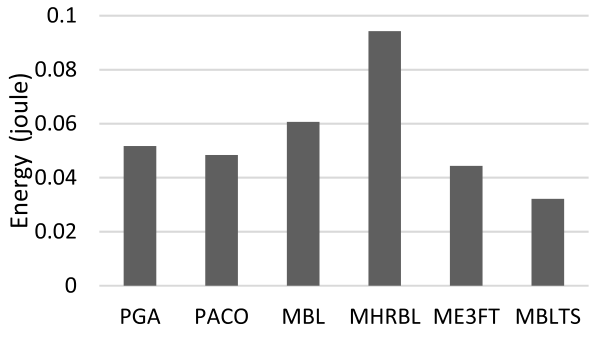

FIGURE 8. Consolidated energy consumption for all scheduling scheme.

all the scheduling schemes before introducing parallelization. The experimental results show that the E3FT scheme exhibit highest execution time among all the compared schemes. On the other extreme, BL scheme achieves lowest execution time amongst the compared schemes. BL has approximately 11%, 32%, and 37% lower execution time compared with HRBL, BLTS, and E3FT, respectively. Similarly, HRBL scheme reports second lowest average execution time having approximately 21% and 38% lower execution time compared to BLTS and E3FT, respectively. The reason behind the higher execution time of E3FT is due to the time spent in locating the appropriate PE in case of ties that may occur when multiple PEs have the same minimum completion time, same energy consumption, or same utilization. On the other hand, the BL scheme achieved the lowest execution time because the PEs are selected for task mapping from the already computed matrix W. Similarly, consolidated average execution time reported by the scheduling scheme after introducing parallelization is shown in Fig. 3(b). The experimental results reveal that the PGA scheme reports the highest execution time. On the other extreme, MBLTS reports the lowest execution time among all the compared schemes. Specifically, PACO, ME3FT, MHRBL, and MBLTS has approximately 15%, 41%, 42%, and 43% lower execution time compared with the MBL. Based on the percentage difference HRBL scheme achieved the second lowest execution time compared with ME3FT and MBL, respectively. It can be witnessed that all the parallel schemes reduce the execution time

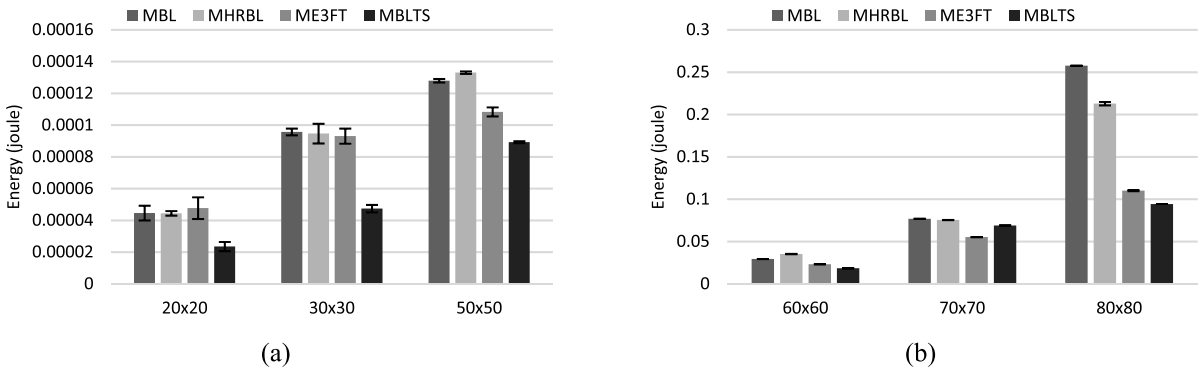

FIGURE 9. Energy consumption for scheduling schemes (a) small mesh size (b) large mesh sizes.

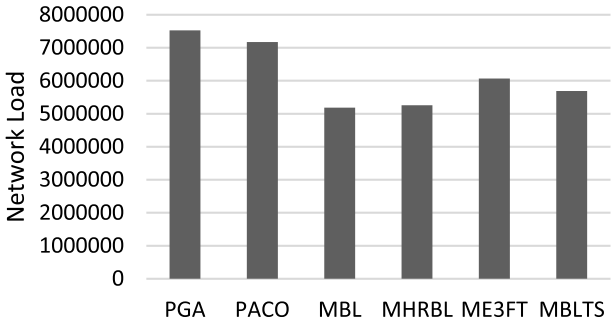

FIGURE 10. Consolidated network load for all scheduling scheme.

compared to their non-parallelized counterparts. However, MBL scheme reported the highest reduction in execution time compared to the other schemes.

Fig. 4(a) and (b) show the execution time of BL and MBL schemes, respectively. The MBL achieves 12%, 22%, and 16% reduction in execution time for 20 × 20, 30 × 30, and 50 × 50, respectively. It can be seen that 30 × 30 achieves maximum reduction while 20 × 20 achieves the minimum reduction. The execution time for 30 × 30 NoC mesh size takes more time than 50 × 50. This is due to higher task to PE ratio for 30 × 30 mesh size, which is 3.33 tasks per PE. Whereas, in case of 50 × 50 the ratio is 2.0 tasks per PE. Consequently, for 30 × 30 the algorithm must map more tasks per PE that leads to increase in execution time of the algorithm. The reason behind such behavior is that finding appropriate PE for each task consumes more time due to higher task to PE ratio. It is worth mentioning here that the same argument holds for all the other presented schemes. Fig. 4(b) show the same comparison for large mesh NoCs, i.e., 60 × 60 to 80 × 80. Similar trend is observed in Fig. 4(b) where MBL achieves 9%, 19%, and 14% reduction in execution time compared with BL for 60 × 60, 70 × 70, and 80 × 80, respectively.

Fig. 5(a) and (b) show the execution time of HRBL and MHRBL schemes, respectively. The MHRBL scheme achieves 24%, 26%, and 31% reduction in execution time compared with HRBL for 20 × 20, 30 × 30, and 50 × 50 respectively. Following the same pattern for large NoC mesh sizes shown in Fig. 5(b), the MHRBL scheme outperformed the HRBL scheme by achieving a decrease of 27%, 30% and 30% for 60 × 60, 70 × 70, and 80 × 80 mesh, respectively. Fig. 6(a) and (b) shows the running time of the E3FT scheme and the ME3FT scheme. The percentage reduction for ME3FT over E3FT scheme is 5%, 19%, and 57% for 20 × 20, 30 × 30 and 50 × 50, respectively. The maximum reduction is achieved by 50 × 50 while the minimum reduction was achieved by 20 × 20. Similar trend is observed for the large mesh NoCs presented in Fig. 6(b). Fig. 7(a) and (b) show comparison of the BLTS and MBLTS scheme. MBLTS exhibit 12%, 24%, and 16 % reduction in execution time compared with BLTS for 20 × 20, 30 × 30, and 50 × 50. Similar trend is observed where MBLTS outperforms BLTS by 9%, 17%, and 13% lower execution time for 60 × 60, 70 × 70, and 80 × 80 mesh sizes, respectively.

D. ENERGY CONSUMPTION

Consolidated energy consumption of scheduling schemes averaged over all mesh sizes is shown in Fig. 8. Experimental results reveal that MHRBL scheme reports the highest energy consumption. On the other extreme, MBLTS achieves the lowest energy consumption among all the compared schemes. Specifically, MBL, PGA, PACO, ME3FT, and MBLTS has approximately 43%, 58%, 64%, 72%, and 98% lower energy consumption compared with MHRBL. The reason behind higher energy consumption of MBL and MHRBL is that these techniques seek to jointly optimize overheads in communication as well as energy consumption. Consequently, to cut down the overhead some tasks are assigned to PEs having higher energy consumption that leads to increase in energy consumption. Alternatively, the factor behind lower energy consumption of MBLTS and ME3FT is that these schemes focus at reducing energy consumption without considering the communication overhead among interdependent tasks.

The experimental result presented in Fig. 9 shows the energy consumption of all the schemes over different mesh sizes. The experimental results show that all the algorithms follow almost similar trend for small and large mesh NoCs.

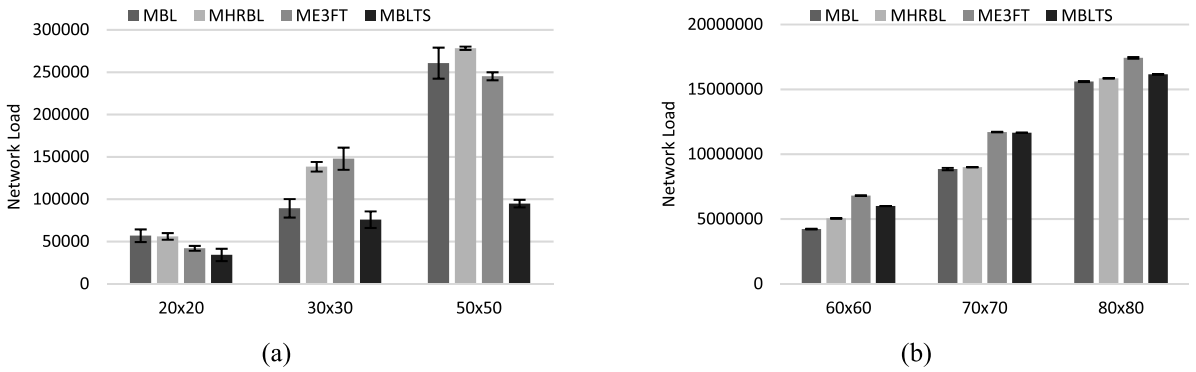

FIGURE 11. Network load for scheduling schemes (a) small mesh size (b) large mesh sizes.

The exceptional cases are of MHRBL for 50 × 50 and 60 × 60 where MHRBL has slightly higher energy consumption compared with MBL. The reason behind such behavior is that MHRBL explicitly seek to reduce overheads in communication by placing heavily communicating tasks to close by while sacrificing energy consumption. Another exceptional case is of MBLTS for 70 × 70 mesh NoC where MBLTS has higher energy consumption compared with ME3FT. The reason behind this is that MBLTS attempts to find idle slots for reducing energy consumption, which may not be available in all scenarios.

E. NETWORK LOAD

The consolidated network load of parallel scheduling schemes is presented in Fig. 10. The experimental result shows that on average PGA scheme reports the highest network load followed by the PACO. Specifically, PACO, ME3FT, MBLTS, MHRBL, and MBL have approximately 5%, 22%, 28%, 36%, and 37% higher network load compared to MBL scheme. The reason behind the higher network load exhibited by ME3FT and MBLTS can be attributed to the fact that these schemes focus more on lowering consumption in computational energy without explicitly considering the overheads in communication among interdependent tasks.

The experimental result presented in Fig. 11 shows network load of all the schemes over individual mesh size. It can be observed that MBLTS shows varying trend of small and large mesh NoCs, i.e., MBLTS achieves lower communication overhead for small NoCs whereas it has higher network load for large mesh NoCs. The rational associated with such behavior is that MBLTS seek to place successor tasks in the idle slots starting from the PE where the predecessor task is mapped. For small mesh NoCs the number of tasks to core ratio is low, consequently many successor tasks can be mapped onto the same PE as of the predecessor task leading to lower network load. However, for large mesh NoCs task to core ratio is high that leads to majority of tasks successor tasks being mapped onto different PEs that results in higher network load for large mesh NoCs.

VI. CONCLUSION

In this article, we presented parallel scheduling techniques aimed at reducing the execution time of the algorithm for large workflows. The experimental results show that parallel scheduling schemes achieve better time efficiency compared to their non-parallel counterpart. Among the compared schemes, MBLTS achieved lowest execution time followed by MHRBL. Similarly, MBLTs outperformed all compared schemes in terms of energy consumption. Whereas, MBL scheme exhibited lowest network compared to other presented schemes. The work presented in this article can be extended by implementing the graph partitioning schemes to partition the workflows and executing multiple instances of the application mapping algorithms on the partitioned workflow to further reduce the algorithm execution time.

REFERENCES

[1] T. Maqsood, S. Ali, S. U. R. Malik, and S. A. Madani, "Dynamic task mapping for network-on-chip based systems," *J. Syst. Archit.*, vol. 61, no. 7, pp. 293–306, Aug. 2015.

[2] S. Tosun, "Energy-and reliability-aware task scheduling onto heterogeneous MPSoC architectures," *J. Supercomput.*, vol. 62, no. 1, pp. 265–289, Oct. 2012.

[3] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, and J.-S. Pan, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 2, pp. 451–462, Feb. 2018.

[4] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2008, pp. 164–169.

[5] S. Roloff, F. Hannig, and J. Teich, "Fast architecture evaluation of heterogeneous MPSoCs by host-compiled simulation," in *Proc. 15th Int. Workshop Softw. Compil. Embedded Syst.*, 2012, pp. 52–61.

[6] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[7] T. Li, T. Zhang, G. Yu, J. Song, and J. Fan, "Minimizing temperature and energy of real-time applications with precedence constraints on heterogeneous MPSoC systems," *J. Syst. Archit.*, vol. 98, pp. 79–91, Sep. 2019.

[8] N. Prasad S and S. S. Kulkarni, "Performance and energy balanced algorithm for executing high performance computing application," in *Proc. Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Nov. 2019, pp. 252–257.

[9] K. L. A. Uchechukwu and Y. Shen, "Energy consumption in cloud computing datacenters," *Int. J. Cloud Comput. Services Sci.*, vol. 3, pp. 31–48, Jun. 2014.

[10] T. Maqsood, K. Bilal, and S. A. Madani, "Congestion-aware core mapping for network-on-chip based systems using betweenness centrality," *Future Gener. Comput. Syst.*, vol. 82, pp. 459–471, May 2018.
[11] L. Yu, T. Jiang, and Y. Cao, "Energy cost minimization for distributed Internet data centers in smart microgrids considering power outages," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 120–130, Jan. 2015.
[12] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, Apr. 2017.
[13] A. Majd, G. Sahebi, M. Daneshtalab, and E. Troubitsyna, "Optimizing scheduling for heterogeneous computing systems using combinatorial meta-heuristic solution," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput.*, Dec. 2017, pp. 1–8.
[14] S. Khan, S. Anjum, U. A. Gulzari, T. Umer, and B.-S. Kim, "Bandwidth-constrained multi-objective segmented brute-force algorithm for efficient mapping of embedded applications on NoC architecture," *IEEE Access*, vol. 6, pp. 11242–11254, 2018.
[15] Z. Li-yun and Z. Li-feng, "An ant colony optimization algorithm based on automatic dynamic updating," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng. (CSAE)*, May 2012, pp. 111–116.
[16] H. Ali, U. U. Tariq, Y. Zheng, X. Zhai, and L. Liu, "Contention & energy-aware real-time task mapping on NoC based heterogeneous MPSoCs," *IEEE Access*, vol. 6, pp. 75110–75123, 2018.
[17] Y. Xue, Z. Qian, G. Wei, P. Bogdan, C.-Y. Tsui, and R. Marculescu, "An efficient network-on-chip (NoC) based multicore platform for hierarchical parallel genetic algorithms," in *Proc. 8th IEEE/ACM Int. Symp. Networks-on-Chip (NoCS)*, Sep. 2014, pp. 17–24.
[18] F. Ferrandi, P. L. Lanzi, C. Pilato, D. Sciuto, and A. Tumeo, "Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 6, pp. 911–924, Jun. 2010.
[19] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 1–11, Sep. 2017.
[20] T. Maqsood, N. Tziritas, T. Loukopoulos, S. A. Madani, S. U. Khan, C.-Z. Xu, and A. Y. Zomaya, "Energy and communication aware task mapping for MPSoCs," *J. Parallel Distrib. Comput.*, vol. 121, pp. 71–89, Nov. 2018.
[21] D. Talia, "Workflow systems for science: Concepts and tools," *ISRN Softw. Eng.*, vol. 2013, p. 15, Dec. 2013.
[22] Y. Wang, K. Li, H. Chen, L. He, and K. Li, "Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 134–148, Jun. 2014.
[23] S. P. N. Chatterjee and S. Chattopadhyay, "Task mapping and scheduling for network-on-chip based multi-core platform with transient faults," *J. Syst. Archit.*, vol. 83, pp. 34–56, 2018.
[24] H. Ali, X. Zhai, U. U. Tariq, and L. Liu, "Energy efficient heuristic algorithm for task mapping on shared-memory heterogeneous MPSoCs," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun.*, Jun. 2018, pp. 1099–1104.
[25] D. I. Arkhipov, D. Wu, T. Wu, and A. C. Regan, "A parallel genetic algorithm framework for transportation planning and logistics management," *IEEE Access*, vol. 8, pp. 106506–106515, 2020.
[26] R. E. Ferreira, L. de Macedo Mourelle, and N. Nedjah, "A parallel genetic algorithm on a multi-processor system-on-chip," in *Proc. Int. Conf. Ind., Eng. Appl. Appl. Intell. Syst.*, 2010, pp. 164–172.
[27] L. Chen, H.-Y. Sun, and S. Wang, "A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem," *Inf. Sci.*, vol. 199, pp. 31–42, Sep. 2012.
[28] T. Maqsood, N. Tziritas, T. Loukopoulos, S. A. Madani, S. U. Khan, and C.-Z. Xu, "Leveraging on deep memory hierarchies to minimize energy consumption and data access latency on single-chip cloud computers," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 154–166, Apr. 2017.
[29] N. Van Cuong, N. T. Bang, L. Dinh Tuyen, and P. N. Nam, "Dynamic mapping of quality adjustable applications on NoC-based reconfigurable platforms," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Hanoi, Vietnam, Oct. 2016, pp. 322–327.
[30] H. Ali, U. Ullah Tariq, X. Zhai, and L. Liu, "Energy efficient task mapping & scheduling on heterogeneous NoC-MPSoCs in IoT based smart city," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun.*, Jun. 2018, pp. 1305–1313.
[31] R. Bamnote, P. M. RavaleNerkar, and S. S. Apte, "Task dependency aware IP core for dynamic scheduling in MPSoC environment," in *Proc. Int. Conf. Inf. Process. (ICIP)*, Dec. 2015, pp. 80–84.
[32] S. Puch, I. Sánchez, and M. Rowe, "Few-shot learning with deep triplet networks for brain imaging modality recognition," in *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*. Springer, 2019, pp. 181–189.
[33] A. Grando, A. Manataki, and S. K. Furniss, "Multi-method study of electronic health records workflows," in *Proc. AMIA Annu. Sympo Proc.*, 2018, p. 498.
[34] R. Prodan, "Specification and runtime workflow support in the ASKALON grid environment," *Sci. Program.*, vol. 15, no. 4, pp. 193–211, 2007.
[35] U. U. Tariq, H. Ali, L. Liu, J. Panneerselvam, and X. Zhai, "Energy-efficient static task scheduling on VFI-based NoC-HMPSoCs for intelligent edge devices in cyber-physical systems," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–22, Dec. 2019.
[36] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. Asia South Pacific Design Autom. Conf.*, 2003, pp. 233–239.
[37] M. N. Akhtar and O. Sidek, "An intelligent arbiter for fair bandwidth allocation," in *Proc. IEEE Student Conf. Res. Develop.*, Dec. 2011, pp. 304–309.
[38] K. Jain, S. K. Singh, A. Majumder, and A. J. Mondai, "Problems encountered in various arbitration techniques used in NOC router: A survey," in *Proc. Int. Conf. Electron. Design, Comput. Netw. Automated Verification (EDCAV)*, Jan. 2015, pp. 62–67.
[39] D. R. G. Silva, B. S. Oliveira, and F. G. Moraes, "Effects of the NoC architecture in the performance of NoC-based MPSoCs," in *Proc. 21st IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2014, pp. 431–434.
[40] A. Elewi, M. Shalan, M. Awadalla, and E. M. Saad, "Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2s, pp. 1–27, Jan. 2014.
[41] H. L. Ong, M. J. Magazine, and T. S. Wee, "Probabilistic analysis of bin packing heuristics," *Oper. Res.*, vol. 32, no. 5, pp. 983–998, Oct. 1984.
[42] W. Y. Lee, "Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 530–537, Mar. 2012.
[43] Z. Guo, A. Bhuiyan, A. Saifullah, N. Guan, and H. Xiong, "Energy-efficient multi-core scheduling for real-time DAG tasks," in *Proc. 29th Euromicro Conf. Real-Time Syst. (ECRTS)*, 2017, pp. 22:1–22:21.
[44] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo, "An approximation algorithm for energy-efficient scheduling on a chip multiprocessor," in *Proc. Design, Autom. Test Eur.*, 2005, pp. 468–473.
[45] S. Baruah, "Mixed criticality schedulability analysis is highly intractable," Washington Univ. St. Louis, St. Louis, MO, USA, Tech. Rep., 2009.
[46] Y. Samadi, M. Zbakh, and C. Tadonki, "E-HEFT: Enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2018, pp. 601–609.
[47] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[48] M. Jeon and D. Kim, "Parallel merge sort with load balancing," *Int. J. Parallel Program.*, vol. 31, pp. 21–33, Mar. 2003.

BICHI BASHIR YUSUF received the B.Sc. degree in computer science from KUST Wudil, Kano, Nigeria, in 2010, and the M.Sc. degree in computer engineering from Yasar University, Izmir, Turkey, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science, COMSATS University Islamabad, Pakistan. His research interests include application mapping, NoC-based MPSoC systems, and cloud and fog computing.

TAHIR MAQSOOD received the M.Sc. degree in computer networks from Northumbria University, U.K., in 2007, and the Ph.D. degree in computer science from COMSATS University Islamabad, Pakistan, in 2017. He is currently an Assistant Professor with COMSATS University Islamabad at Abbottabad, Pakistan. His research interests include resource allocation, multi/manycore systems, reliable systems, the Internet of Things, and mobile edge computing.

FAISAL REHMAN received the M.S. and Ph.D. degrees in computer science from COMSATS University Islamabad at Abbottabad, Pakistan, in 2010 and 2018, respectively. He is currently an Assistant Professor with COMSATS University Islamabad. His research interests include recommender systems, green computing, and wired and wireless networks.

SAJJAD A. MADANI received the M.S. degree in computer sciences from the Lahore University of Management Sciences and the Ph.D. degree from the Vienna University of Technology. He is currently a Professor with COMSATS University Islamabad, Pakistan. He has published more than 90 papers in peer-reviewed international conferences and journals. His research interests include low-power wireless networks, green computing, and optimization techniques.