BOTS: Balanced-Objective Task Selector for Application Graph Scheduling in Reconfigurable Computing Systems

Milad Gholamrezanejad Ferdos
Iran University of Science and Technology
School of Electrical Engineering
Tehran, Iran
m_gholamrezanejad94@elec.iust.ac.ir

Hadi Shahriar Shahhoseini
Iran University of Science and Technology
School of Electrical Engineering
Tehran, Iran
shahhoseini@iust.ac.ir

Abstract— **Hardware-based execution capability with resource reusing flexibility is one of the new features of Reconfigurable Computing (RC) systems. However, due to the reconfiguration constraints, it is difficult to fully exploit their advantages. For a better design with higher performance, several pivotal objectives should be investigated simultaneously such as task execution time, communication data between tasks, and required resources for implementation. In this paper, a new algorithm for task selection in an application graph has been introduced to balance objectives in RC system. This algorithm decides based on a selection merit to reduce the number of execution clusters and to save communication cost as much as possible. To remove forbidden solution, a set of free tasks has been constructed to satisfy the precedence constraints. The evaluation has been performed with different topological features such as parallelism degree and critical path. Results showed that the proposed method provides acceptable improvement in comparison with previous method for application execution time metric.**

Keywords- Reconfigurable Systems; Task Scheduling; Resource Allocation; Multi-objective Algorithm

I. INTRODUCTION

Hardware-based application implementation provides a significant speedup and in nowadays embedded systems, it is an undeniable part to accelerate the computations. Reconfigurable computing (RC) systems have been used in a wide range of applications such as cryptography [1], computer network security [2], multimedia [3], spatial data infrastructure (SDI) [4], and sensor networks [5], etc. According to the reconfiguration feature, RC systems have categorized in two groups, partially and fully RC systems [6, 7]. In partially RC systems, every single Configurable Logic Block (CLB) can be re-configured during the operation of other parts. But in fully RC systems, the whole device should be stopped to load new configuration bit-stream into it [8, 9]. The architecture of RC systems included two main parts. The first part is the reconfigurable hardware in which CLBs are gathered with one specific network to communicate with each other. One of the known platforms to act as a reconfigurable hardware is FPGA [10]. The second part is a General Purpose Processor (GPP) to handle the requests and to prepare the reconfigurable part for its process, as well. There is a local memory in this system to save and load the required data. In the memory unit there exists clustering structures to fully exploit the available memory resources [11]. As the hardware resources are limited and application requests are always more, a kind of coordinator is needed to execute tasks of an application portion by portion. Therefore, tasks will be executed in several cycles which are called Execution Cycles [12]. In each execution cycle, the required resources will be reserved for tasks and data will be accessed by local memory. After the execution, the processed data will be written back to the local memory. Accessing data from local memory suffers the system a large delay and to reduce it, those communicative tasks should be placed in the same cluster of execution cycle [13].

To demonstrate the communication condition of an application, it will be represented with a Task Graph (TG). In TG each node is a computational task with a predefined execution time and required resource with links showing the communication data between tasks. Considering TGs of applications, one scheduling algorithm should be developed to determine the start time of task clusters for implementation. Scheduling algorithms for partially RC systems are kind of online or dynamic algorithms [14] due to the non-stopped reconfiguration. Decisions flexibility are high in these type of algorithms but they should deal with more complexity [15]. On the other hand, in fully RC systems, all decisions will be made before executing the whole TG in offline mode. This type of scheduling is called offline scheduling. The main target in offline scheduling of RC systems is to reduce the number of local memory access by executing more tasks in one execution cycle [16].

Applications contain many different features and they will affect the performance of scheduling algorithm. One of them is communication feature. For example, Digital Signal Processing (DSP) applications [17] included several communicative functions while Bioinformatics applications [18] need more execution time speedup for each implemented task. The other one is the topological feature. Applications structure contain many different serial or parallel paths and based on them, they will be categorized as pipelined and non-pipelined structures. In pipelined structures, dependency constraint is more than other structures and as a result, scheduling options will be limited to some pre-determined states. Non-pipelined structures demonstrated more distributed patterns in which more independent parallel task can be found [19]. These features shows that for obtaining higher performance, it is recommended

to use strategies with different aspects based on different applications attributes.

The aim of this paper is to find the best clusters of a TG by considering several objectives such as communication cost, wasted resources and execution of TG. To achieve this purpose, we investigated inter and intra cluster interactions with a balanced selector function. A set of free nodes has been constructed to create validated solutions with no dependency constraint violation. By using a balanced selector function, it is possible to make cluster with an appropriate criteria.
The rest of this paper is organized as follows, in section 2 the scheduling problem in RC system has been defined, in section 3 the proposed scheduling algorithm with a balance-objective task selector has been introduced, in section 4 experimental results with comparisons and discussions have been presented, and finally, conclusion is the final section of this paper.

II. Problem Definition

As resources in the RC system are limited, the computational nodes of TG should be executed cluster by cluster. TGs are represented by $G(V,E)$ which is a Directed Acyclic Graph (DAG). V is the set of nodes including tasks $T_i (1 \leq i \leq N)$ and E is the set of communication links between tasks. N is the total number of nodes in TG. If two tasks T_i and T_j in the TG should communicate with each other, then there is one communicational link between them denoted by T_C. With this constraint, task T_i cannot be implemented before task T_j. Other attributes of each node of TG are execution time (T_E) and required resources (u_i) . Total resources of reconfigurable hardware is denoted by U_H and in each execution cycle, the reserved resources by a cluster must not exceed of this parameter. Considering these parameters, the target is to obtain a queue of clusters by satisfying conditions of equations (1) to (4).

$$uc_i \leq U_H, \quad i = 1,2,3,\dots,N_c \tag{1}$$

$$dependency(i,j) = 0, i,j \in clusters, i < j \tag{2}$$

$$Minimize \left(\sum_{i=i}^{N_c} (T_{ce} + T_{cc})_i \right) \tag{3}$$

$$Maximize \left(\frac{1}{N_c} \sum_{i=1}^{N_c} \frac{uc_i}{U_H} \right) \tag{4}$$

where T_{cE} is the total execution time of tasks in the cluster, T_{cC} is the inter-communication data between clusters, uc_i is the required resources in execution cycle i and N_c is the number of TG clusters. Figure (1) shows an example of making clusters of nodes in a TG. In this figure tasks T_1, T_3, T_4 are placed in cluster 1, tasks T_2, T_5 are placed in cluster 2, and task T_6 is placed in cluster 3. Each cluster has been implemented in reconfigurable hardware in one execution cycle. As can be seen, those communication links within the clusters will be saved from accessing the local memory. Also, it has been shown that in the third execution cycle, there are more resource wasting.

III. Balanced-Objective Task Selector

To make clusters of tasks in an application graph, reducing the total counts of clusters is one of the targets for scheduling. It causes to prevent frequent access from local memory and communication will be performed within the reconfigurable hardware part. This will also improve the resource utilization by placing more tasks in one cluster. The other target is communication data of the remained links between clusters. These links will access local memory and based on the amount of data, it will produce a considerable delay. For achieving a solution by balancing objectives, a task selector must be introduced. In this section, a Balanced-Objective Task Selector (BOTS) will be introduced. The aim of this task selector is to provide a selection metric to be used in each situation that more than one choice is available. This task selector has been built based on below considerations:

1- Degree of tasks in an application graph cause more dependency and as a result they have higher priority to be selected in a cluster comparing to low-degree tasks.

2- To save more communication cost it is recommended to place a task with more communication data within the cluster.

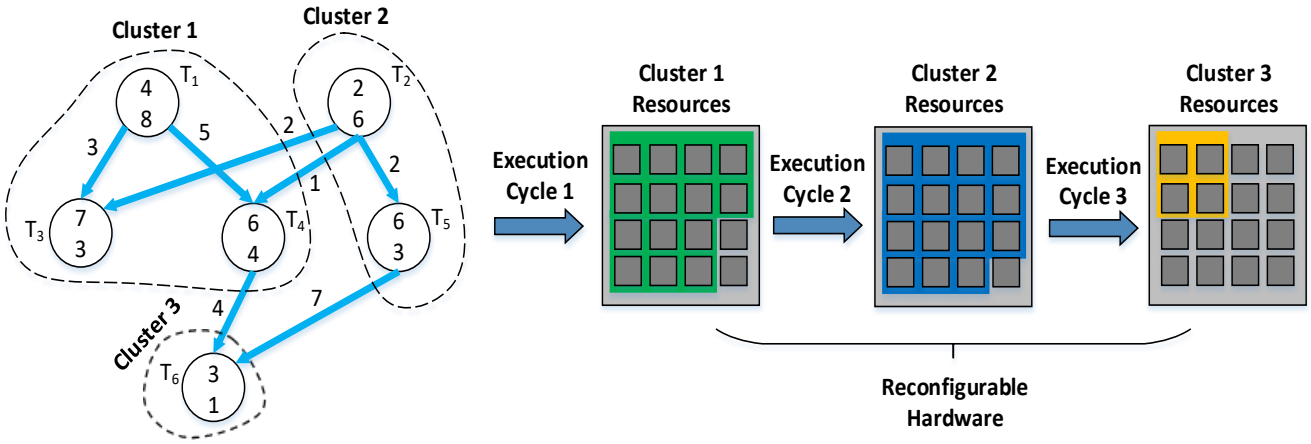

Figure 1. *Scheduling description of a DAG with 3 clusters in RC system*

3- As reconfigurable resources are limited, they should be assigned to more tasks of application graph. As a result, small tasks will be assigned as higher priority to be selected in a cluster.

4- Tasks within a cluster should contain similar execution time

5- Time to prevent resource waiting of finished task, so it is recommended to select tasks with small execution time within a cluster.

Using these considerations, one equation has been created to show the selection metric for each task in application graph. Equation (5) demonstrates the relationship between the selection metric and attribute of tasks in an application graph.

$$\omega_i = \frac{T_{C_i} + n_i}{T_{E_i} + u_i} \qquad (5)$$

where, T_{C_i} is the communication cost between the task and its children, T_{E_i} is the execution time of the task, n_i is the number of a task's children and u_i corresponds the required resources for each task. For producing validated solutions without any dependency constraint violation, the proposed algorithm will be performed in several updating phases. In each updating phase, a set of free nodes will be created and among them, the task with higher selection criteria will be selected. If the size of each cluster exceeds the size of reconfigurable resources (U_H), a new cluster will be created. Algorithm 1 shows the procedure of BOTS of making clusters in an application graph. Figure (2) shows an example of application graph on which BOTS has been performed. As mentioned before, ω criteria will be computed for each node of TG which is equal to $\omega = \{0.75, 0.54, 1.28, 0, 0\}$. Members of set F demonstrate nodes with no precedence dependency and for the first step of this TG, it has only included node T_1. After assigning this node to cluster 1 and removing it from F, the set of free nodes will be updated with nodes T_2 and T_3. As node T_3 has a higher ω, it will be selected in the first cluster. This procedure will continue until there is no resources for additional nodes in the cluster and therefore, the new cluster should be constructed for remained nodes. It can be seen from this example that, BOTS will provide better resource utilization while saving more communication and execution cost as they have been considered in equation (5).

Algorithm 1: Balanced-Objective Task Selector (BOTS) for making clusters

Input: TG attributes $\{T_E, T_C, u\}$, Number of nodes (N_T), Hardware resource (U_H)
Output: Number of clusters (N_c), Cluster members $(C(N_c))$
For $n = 1; n = N_T; n++$
  $\omega_n = \{equation(2)(T_E, T_C, u)\}$
End For
$N_c = 1$
For $i = 1; i = N_T; i++$
  $F = free\ nodes\ (TG)$
  $select_i = F(\max(\omega_i))$
  $u(N_c) \leftarrow u(N_c) + u_{select_i}$
  If $u(N_c) > U_H$
    $N_c = N_c + 1$
    $u(N_c) = u_{select_i}$
    $C(N_c) = select_i$
    End if
  If $u(N_c) < U_H$
    $C(N_c) \leftarrow C(N_c) + select_i$
    End if
End For
Return $N_c, C(N_c)$

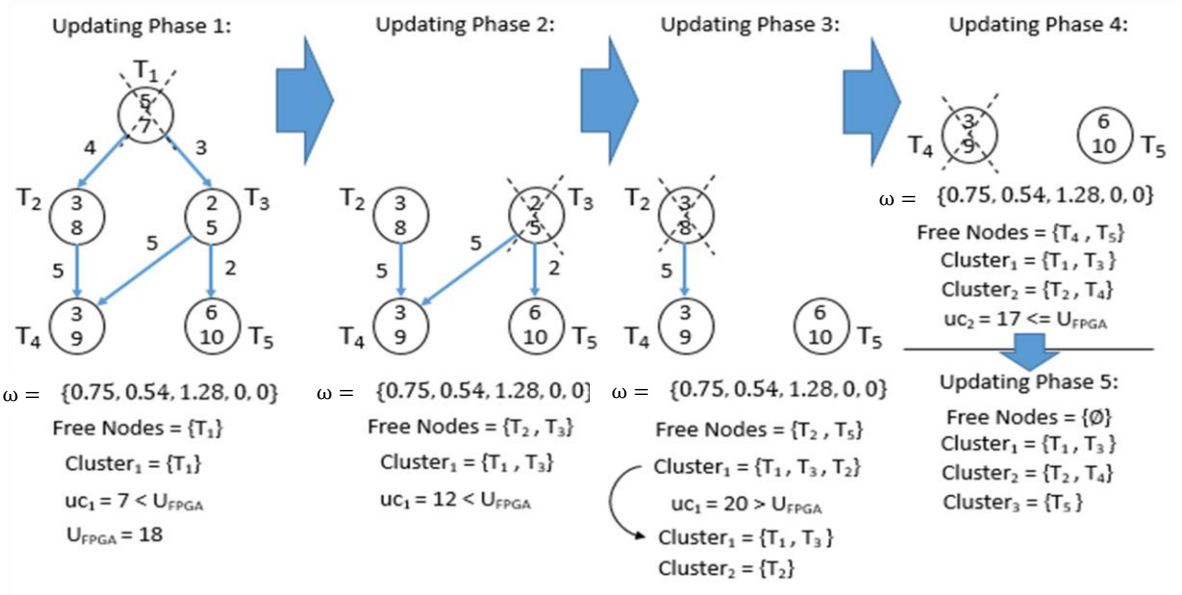

Figure 2. *Description of BOTS steps for selecting tasks in TG*

IV. Simulation Results and Discussion

In this section, the performance of the proposed selector function is evaluated and compared with three greedy algorithm based on the execution time, communication cost and resource utilization. The performance of BOTS has been compared with the heuristic method represented in [20], as well. We started our evaluations on a set of TGs generated with a random graph generator. A discrete time simulation framework in Matlab software has been constructed with an existed hardware model Xilinx Virtex II pro XC2VP30. The aim is to reduce the makespan of TGs, so we compared solutions derived from clustering methods based on the makespan metric. Beginning by TGs with sizes from 10 to 70 nodes, Figure (3) illustrates the effect of TG size on the proposed method in comparison with other methods.

As it is clear, since greedy algorithms focus only on one parameter at a time, they can't beneficially establish the required trade-off in the process of scheduling. Moreover, the [20] heuristic method also neglects one of the main parameters, communication cost. As the size grows, due to the consistency of the link distribution, there would be more communication cost. So, neglecting this factor would force great amount of communicatio cost which would affect the makespan of the TG. On the other hand, the proposed method can satisfy the problem requirements by considering several factors at the same time. The balanced solution, creates a trade-off between hardware utilization, execution time, communication time and number of children. As it was expected, the higher sizes of TGs will have more communication cost, logically leading to higher makespan but due to the savings in communication data in BOTS, the makespan would be less in comparison with other methods. With the increase in size, the greedy communication method would perform better due to the savings in the communication rather than the other greedy algorithms and heuristic method in [20].

We carry on our experiments to evaluate the effect of connectivity distribution on the performance of the proposed method. Figure (4), shows the simulation result for TGs with 30 nodes and Connectivity Distribution (CD) ranging from 0.1 to 1. With the rise of connectivity, the TGs would become tighter, limiting the algorithms to specific solutions and forcing more cost. BOTS perform better on less connected graphs, since more options are available and the decisions can be made with more freeness. In each task selection phase, BOTS tends to select a task with more children so there would be more free nodes in the next selection phase. In lower CDs, greedy communication and greedy utilization algorithms are not beneficial since application graph links are limited and the number of links in TG is less. On the other hand, the greedy execution and [20] method perform better since they prefer less communication. But with the increse in CD, it can be seen that greedy communication and greedy utilization outperform the other greedy algorithm and [20] since TG gets tightly connected and the communication cost gets higher, the parameter that the two approaches neglect completely. Since BOTS establishes a trade-off between the costs, it perform the best among other algorithms both in lower and higher CDs, but lower CDs are preferable due to the degree of freeness which the algorithm may have.

The last evaluation on random graphs was conducted on the effect of Execution to Communication ratio (ECR). ECR clarify the type of TG as being executive or communicative. Executive TGs needs more processing time while communicative TGs require more communication data. Figure (5) illustrates the comparison of BOTS performance versus other methods. As it can be seen, in lower ECR values, communication greedy

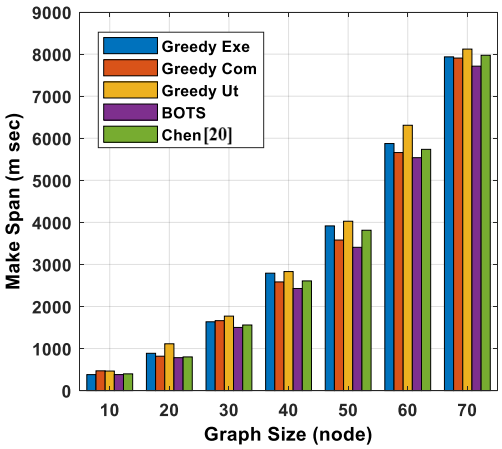

Figure 3. *The performance evaluation of the proposed method versus graph size*

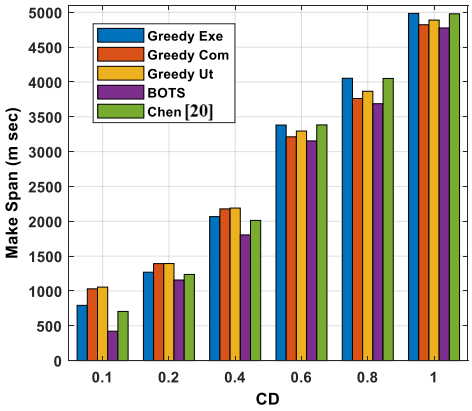

Figure 4. *The performance evaluation of the proposed method versus connectivity degree*

algorithm performs better than other greedy algorithms. In higher ECR values, the greedy execution algorithm and heruistic method in [20] achieved better solutions due to the lack of communicaiton cost. But BOTS, achieve the best solution among all the mentioned methods considering different factors simultaneously and balancing all the parameters. So, the established trade-off will provide less makespan and better performance rather than the greedy algorithms and the heuristic method in [20].

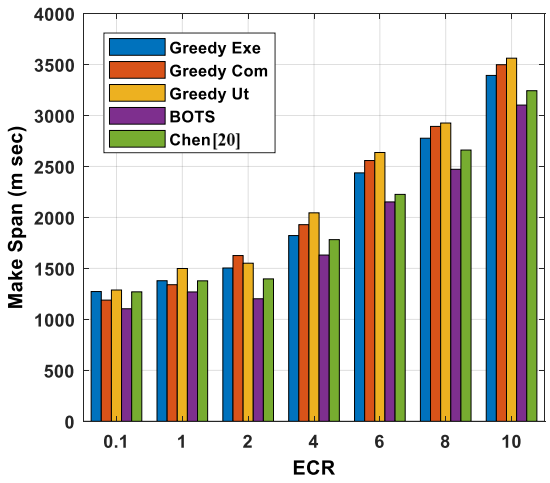

Figure 5. *The performance evaluation of the proposed method versus ECR*

In our experiments, evaluation of performance versus CD manifests another aspect of the problem. As it is clear, changing the CD will affect the structure of TGs, as well. TGs with higher CDs, have less degree of freeness. While in lower CDs, the proposed algorithm can decide on the clusters freely due to the less precedence constraints. So, as it was said, TG structures have effect on the performance of the scheduling algorithm. The proposed algorithm performs the best in executive TGs with lower number of nodes and less connectivity. We can infere that the topology of the graph plays an important role on the performance of the proposed task selector.

V. CONCLUSION

In this work, we represented a heuristic balanced-objective task selector in order to schedule task graphs. The proposed task selector decides based on execution time, communication cost, resource utilization and number of children which indicates the degree of freeness. Since the proposed method acts as a combination of greedy algorithms with various considerations, the experiments have been conducted on randomly genreated graphs and compared with three greedy algorithm and the heuristic approach presented in [20]. The results show that the proposed method provides an average 11% improvement in comparison with other methods. As future works, we aim to consider structural task graph features and extend the algorithm based on these topological features.

REFERENCES

[1] Gaillardon, Pierre-Emmanuel, Reconfigurable Logic: Architecture, Tools, and Applications. CRC Press, 2018.

[2] Saeed, M., Shahhoseini, H.S. APPMA - An Anti-phishing protocol with mutual authentication, Proceedings - 2010 IEEE Symposium on Computers and Communications pp. 308-313

[3] Choudhary, P.S. and Ali, M.S., "FPGA-Based Adaptive Task Scheduler for Real Time Embedded Systems" IEEE International Conference on Research in Intelligent and Computing in Engineering (RICE), pp. 1-4, 2018.

[4] A. Tabatabaei, M. R. Mosavi, A. Khavari and H. S. Shahhoseini, "Reliable Urban Canyon Navigation Solution in GPS and GLONASS Integrated Receiver using Improved Fuzzy Weighted Least-Square Method", Journal of Wireless Personal Communications, Vol.94, No.4, pp.3181-3196, 2017

[5] H. Jamali Rad, M. Azarafrooz, H. S. Shahhoseini and B. Abolhassani, "A new adaptive power optimization scheme for target tracking Wireless Sensor Networks," 2009 IEEE Symposium on Industrial Electronics & Applications, Kuala Lumpur, 2009, pp. 307-312.

[6] M. M. Bassiri and H. S. Shahhoseini, Configuration reusing in on-line task scheduling for reconfigurable computing systems, Journal of Computer Science and Technology 26(3), pp. 463-473, 2011

[7] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, N. Belanger, and A. Artiba, "Mathematical programming models for scheduling in a CPU/FPGA architecture with heterogeneous communication delays" Journal of Intelligent Manufacturing, Vol. 29, pp. 629–640, 2018.

[8] M. M. Bassiri and H. S. Shahhoseini, "On-line HW/SW partitioning and co-scheduling in reconfigurable computing systems," 2009 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, 2009, pp. 557-562.

[9] M. M. Bassiri and H. S. Shahhoseini, "Mitigating reconfiguration overhead in on-line task scheduling for reconfigurable computing systems," 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, 2010, pp. V4-397-V4-402.

[10] Pezzarossa, L., Kristensen, A.T., Schoeberl, M. and Sparsø, J., "Using dynamic partial reconfiguration of FPGAs in real-Time systems" Microprocessors and Microsystems, Vol. 61, pp. 198-206, 2018.

[11] H. S. Shahhoseini, M. Naderi and R. Buyya, "Shared memory multistage clustering structure, an efficient structure for massively parallel processing systems," Proceedings Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, Beijing, China, 2000, pp. 22-27 vol.1.

[12] M. Alam, A. Khan, and A. K. Varshney, "A review of dynamic scheduling algorithms for homogeneous and heterogeneous systems" Advances in Intelligent Systems and Computing, Vol. 32, pp. 73–83, 2018.

[13] M. Huang, V. K. Narayana, H. Simmler, O. Serres, and T. El-Ghazawi, "Reconfiguration and Communication-Aware Task Scheduling for High-Performance Reconfigurable Computing" ACM Transactions on Reconfigurable Technology and Systems (TRETS), Vol. 3, pp. 1–25, 2010.

[14] Mohtavipour, S.M. and Shahhoseini, H.S. "A Link-Elimination Partitioning Approach for Application Graph Mapping in Reconfigurable Computing Systems," The Journal of Supercomputing, pp. 1-25, 2019.

[15] Vipin, K. and Fahmy, S.A., "FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications" ACM Computing Surveys (CSUR), Vol. 51, pp. 72-111, 2018.

[16] M. M. Bassiri and H. S. Shahhoseini, "A New approach in on-line task scheduling for reconfigurable computing systems," ASAP 2010 - 21st IEEE International Conference on Application-specific Systems, Architectures and Processors, Rennes, 2010, pp. 321-324.

[17] Surendar, A., "FPGA based parallel computation techniques for bioinformatics applications" International Journal of Research in Pharmaceutical Sciences, Vol. 8, pp.124-128, 2017

[18] Rodríguez-Andina, J.J., Valdes-Pena, M.D. and Moure, M.J., "Advanced features and industrial applications of FPGAs—A review" IEEE Transactions on Industrial Informatics, Vol. 11, pp. 853-864, 2015.

[19] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G. and Vahi, K., "Characterizing and profiling scientific workflows" Future Generation Computer Systems, Vol. 29, pp. 682-692, 2013.

[20] Z. Chen, M. Qiu, Z. Ming, L. T. Yang, and Y. Zhu, "Clustering scheduling for hardware tasks in reconfigurable computing systems," J. Syst. Archit., vol. 59, no. 10, pp. 1424–1432, 2013.