A Resource Partitioning Approach for Task Allocation in Dynamically Reconfigurable Computing Systems

Seyed Mehdi Mohtavipour
Iran University of Science and Technology
School of Electrical Engineering
Tehran, Iran
mehdi_mohtavipour@elec.iust.ac.ir

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

Zahra Asgari
Iran University of Science and Technology
School of Electrical Engineering
Tehran, Iran
z_asgari@elec.iust.ac.ir

Abstract— **Dynamic reconfiguration feature in recent embedded systems provides flexibility in the execution of applications. Using this feature, several tasks of an application can be mapped simultaneously into a limited area of resources in several Execution Cycles (ECs). The complexity of resource management and efficient scheduling of tasks in such systems require more consideration of the task's attributes. In this paper, we proposed a new model for execution configuration which satisfies task constraints such as routing and resource wasting to derive multiple configurations for each task. In order to reduce the surface fragmentation and efficient resource utilization, one partition-based scheduling algorithm with a scheduled time metric has been proposed to properly select among possible configurations and partitions. Several experiments with different scenarios such as heavy and light workloads have been conducted and the results show that rejection ratio is reduced 42.10% and 24.14% in compared with two First and Best Fit algorithms, respectively. Some improvements in other evaluation metrics have been obtained, as well.**

Keywords-component; Reconfigurable Systems, Resource Allocation; Resource Management; Partitioning Algorithms

I. INTRODUCTION

Hardware-based application execution in embedded systems becomes a popular concept for high-performance computing design [1]. These systems include a large density of logic blocks that are able to execute applications simultaneously. However, hardware resources are limited and for implementing large applications, it is recommended to use reconfiguration features [2]. This makes it possible to use resources more efficiently and reduce the cost of the device. Reconfigurable Computing (RC) systems consist of reconfigurable hardware package which is called Reconfigurable Processing Unit (RPU) and a Central Processing Unit (CPU). The CPU in the RC system is used to manage applications, provide required processing data and prepare the reconfiguration data for reconfigurable hardware. Local and shared memories are considered in this system to store configuration and processing data [3]. As the RC system is capable of executing tasks of application simultaneously, those with more parallel features will result in significant performance and speedup such as cryptography [4] wireless sensor network [5] and multimedia [6] applications.

Field Programmable Gate Arrays (FPGAs) because of their block structure, are considered as instance prototyping and a resource for logic circuits, so they are used as a platform for the RC systems. Developments in the technology of FPGAs led to some effective progress in RC systems. Run-Time Reconfiguration (RTR) features in recently FPGA devices allow them to load new bit-streams when other parts of it are running. This is called the partial dynamic reconfiguration and can enable the customization of the limited logic resources to better satisfy application performance and power requirements [7]. Two considerations existed for the reconfigurable devices so that for a fixed application, small tasks can increase logic and area interconnection while large tasks can increase configuration memory. So with this issue, selecting proper tasks per application plays an important role in achieving a better performance. In order to achieve optimum performance, one scheduler and placer is needed for managing the sources and tasks efficiently. This will prevent resource wasting and low speed operation of the RC system. After converting a behavioral code which represents the relationships between input and output of the considered task to a set of logic circuits in synthesizing process, they are converted to one configurable logic blocks (CLB) of the target hardware. This process is called logic packing. In this process, the total number of CLBs needed is determined and each of them is placed on the target hardware. It is well known that the position of CLBs can affect directly on the performance. Routing process is the final part of the task compilation which connects CLBs to communicate their information. These 4 steps of task compilation procedure in the RC system are illustrated in figure 1. Multiple configurations of the incoming tasks in hardware devices are one of the options that have been considered in recent researches and can affect directly the scheduling and placement algorithms. Changing the structure of application tasks can be performed in several ways which using degree of parallelism or using different architectures and algorithms [8] are some of them. The most important parameter in changing the structure of task in the compilation is the resource wasting as it will remain unusable resources for other tasks. This will provide a trade-off between resource wasting and lower surface fragmentation.

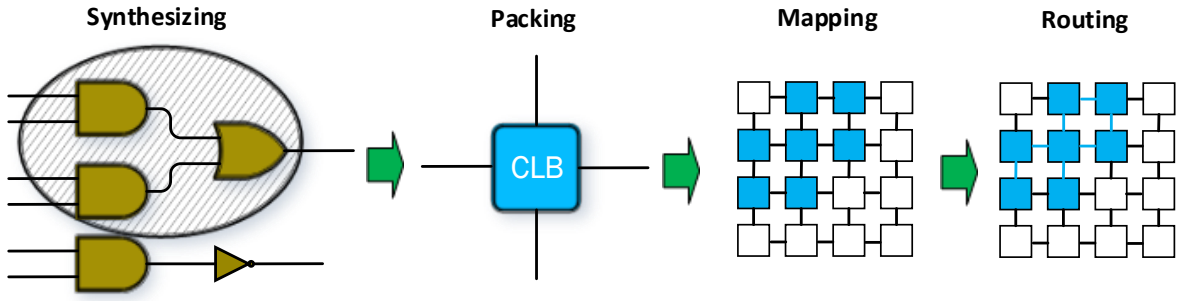

Figure 1. Task compilation process in the RC systems

There are some methods that use optimization algorithms. For example a hybrid placement strategy based on genetic and greedy algorithms is proposed in [9] to efficiently place a set of tasks onto the multi-context FPGA to achieve the best logic capacity utilization. Authors in [10] proposed one integer linear programming together with configuration pre-fetch to minimize the schedule length on the single context devices. [11-13] are two other examples of scheduling and placement researches that in [11-12] an algorithm is proposed to efficiently manage periodic tasks of application and in [13] by analyzing the inter-task communication as well as data dependencies among the existed tasks, number of configuration, communication overhead and task processing time is reduced.

In this paper by considering configuration group for each incoming task based on the constraints such as routability and resource wasting, the surface of reconfigurable hardware is divided into several partitions and with some appropriate metrics and criteria the best partition and configuration will be selected. The main purpose of the proposed algorithm is to reduce fragmentation and wasted logic blocks on the surface of the RPU.

The remainder of this paper is organized as follows, in section 2 problem definition of reconfigurable computing scheduling and placement is described, in section 3 task model with multiple configurations is introduced, in section 4 the new partition-based task allocation is proposed. In section 5, results and discussion is presented and conclusion is section 6 of this paper.

II. PROBLEM DEFINITION

Reconfigurable hardware width denoted by W represents the number of logic block columns. Each column consists of a CLB series that can be reconfigured partially and tasks will be implemented cycle by cycle in a rectangular size of some columns. Incoming executive tasks are considered sequentially and independent from an input buffer with an exponential distribution which the rate of this distribution determines the amount of workload of the system. Considered tasks included a set of logical circuits compatible with reconfigurable hardware. $T = \{T_1, T_2, \dots, T_n\}$ is the set of one application tasks which each $T_k \in T$ is a task with five main parameters $(t_a, w, e_{sw}, e_{hw}, s)$ denoted to arrival time, requirement width, execution time in software, execution time in hardware and task slack time respectively. Task slack means its deadline for hardware implementation and e_{hw} represents the total time for reconfiguration and execution. The amount of workload in each time interval can be achieved by equation (1).

$$WL_{\Delta T} = \frac{\sum_{\Delta T} w \times e_{hw}}{W \times \Delta T} \qquad (1)$$

Where ΔT is the interval between two specific times such first and last arrival time of the incoming tasks. To place incoming task constraints should be satisfied as follows:

Definition 1. Considering new task T_i and a set of placed tasks $T_j \rightarrow j \in \{1, \dots, n\}$ in reconfigurable device with W width, following placement conditions should be satisfied for new task T_i.

$$p_i + w_i \leq W$$
$$(p_i + w_i) \vee (p_j + w_j) = \emptyset \qquad (2)$$

p_i and p_j are the bottom-left positions of the new and placed tasks respectively.

Definition 2. Scheduled time of the new task T_i which denoted by sc_i among tasks $T_j \rightarrow j \in \{1, \dots, n\}$ should satisfy following conditions

$$if \quad (p_i + w_i) \vee (p_j + w_j) \neq \emptyset$$
$$\rightarrow \left((sc_i) \leq (sc_j + e_j)\right) \wedge \left((sc_i + e_i) \geq s_i\right) = \emptyset \qquad (3)$$

III. TASK MODEL WITH MULTIPLE CONFIGURATIONS

In most scheduling algorithms in RC systems such as [14] one rectangular bounding box is considered for each task and by moving the CLBs in this bounding box with some specific algorithms such as simulated annealing, better solutions will be derived. It is clear that one-step allocation of CLBs for each task is the best way to achieve higher performance but because of run-time re-configuration feature in newly RPUs, it is possible to execute one single task in several steps and in each step after the execution of a portion, the results will be saved in the memory for net portions. Works in [15] also emphasized these multiple execution steps as process cycles which each task is executed in several process cycles.

By assuming that each task has a rectangular shape and this concept that they can execute cycle by cycle in the determined area of the reconfigurable hardware, several shapes can be defined. Our main target in this research is to find that states which not wasted hardware resources. For example, if one task

needs 60 CLBs totally for implementation and conditions allows us to partition it in 3 partitions of 20 CLBs or 2 partitions of 30 CLBs then it can be executed by 3 cycles of 20 CLBs or 2 cycles of 30 CLBs. Task 1 consumes 20 CLBs in one cycle and two non-wasted sizes is available, 2×10 , 4×5. Another option is 3×7. It can be seen that if this option is selected then one CLB is wasted but it should be noted that when a temporary placement of several tasks is running, several options of task sizes will be a great opportunity to place more tasks in the limited space of hardware resources.

The incoming tasks include different configurations, but only some of them won't waste the resources of the reconfigurable hardware. In this paper, two main configuration groups have been considered for each incoming task. Each task has a normal configuration with a number of CLB requirements to implement. Configuration groups are categorized to T_{1k}, T_{2k} which the first group won't waste the resources of reconfigurable hardware while the second group wastes reasonable resources. The reasonable wasting percentage is considered 10% of all CLBs in the bounding box and configurations with more wasted resources will be removed form possible configurations. Algorithm 1 shows the procedure of configuration selection and the proposed algorithm uses only the configurations with reasonable wasting percentage. T_{1k} denotes the non-wasting group, T_{2ka} denotes the accepted wasting subgroup and T_{2kr} denotes the rejected wasting subgroup.

Algorithm 1: Configuration Groups of Task

Input: number of total needed CLB (NT), min number of bounding box CLBs (nNB) , max number of bounding box CLBs (xNB), number of CLBs in each row (N)
Output: T_1 , T_{2a} , T_{2r}, number of wasted CLB (WS)
$T_1 \leftarrow \emptyset, T_2 \leftarrow \emptyset$
For $i = nNB, i \leq xNB$, $i + N$ **do**
  If $mod(NT/i * N) = 0$ **then**
    $T_1 \leftarrow current\ size$
  else
    $WS(i) = i * N - \text{mod}(NT/i * N)$
    If $WS(i)/i * N < 10\%$
      $T_{2a} \leftarrow current\ size$
    else
      $T_{2r} \leftarrow current\ size$
    end if
  end if
  end for
  return T_1 , T_{2a}, T_{2r}, WS

IV. PROPOSED TASK ALLOCATION ON PARTITIONED RESOURCES

An efficient placement algorithm should prevent fragmentation of the RPU surface. This can lead to place more hardware tasks in the RPU. The more tasks placed in the RPU the sooner application will be executed. The main idea of the proposed partitioned-based scheduling algorithm is based on the using of resizing feature of the tasks and this will minimize the free and unutilized spaces in the RPU. Multiple configuration feature provides some flexibility to place more tasks in the limited area of the RPU. In the first step, the area of the RPU is partitioned to several parts. The size of each part is proportional to one single task and the incoming tasks should place in one of them. In this step, when the system starts to work and RPU is empty, first some tasks are placed in the RPU from left to right based on the order of their arrival time, after filling the RPU the scheduler looks forward for the first fitted place for other tasks which all these processes are done in the initialization step. This can provide some information about the width of the tasks to partition the RPU based on it. After gathering information about the width of the incoming tasks, algorithm 2 partitions the width of the RPU in the partition step. Figure 2 shows two initialization and partitioned sections for placing the incoming tasks. In initialization section, tasks are placed without any particular placement algorithm and the scheduler will place the task in the first fitted place but in the partitioned section, the width of the RPU is divided to several partitions and by the flexibility of task resizing, the incoming tasks will be placed only in one of the selected partition.

The partition selection in this approach will be done by searching for all possible solutions and finding one of them which has sooner free time. Possible partitions mean that they won't exceed the conditions of the task such as deadline and resource dependency. After determining the sooner free partition then one of the configurations should be considered for that. As decreasing the total fragmentation of the RPU width is the main target in the design of scheduling algorithm, those tasks which are fitted more to the partition will be selected. We considered difference between partition and task size as a selection criteria so that this difference has been considered α percent. It means that only tasks with difference value smaller than α percent should be selected for the partition. If no answer is founded, the task will be rejected to CPU to prevent the resource wasting. Parameter α can affect directly the placement algorithm so that smaller α can cause more task rejection while larger α can cause more resource wasting. An appropriate value for this parameter should be considered and in this research, α is equal to 20 percent. Algorithm 2 shows the placement procedure which is based on the scheduled time metric.

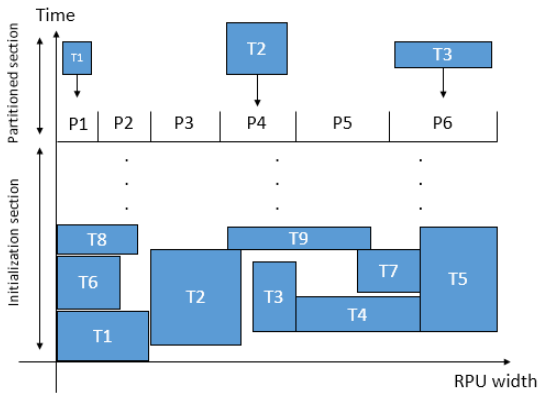

Figure 2. Description of the proposed scheduling algorithm

Algorithm 2: Allocation procedure with scheduled time metric

Input: first group of task size (T_1) , second group of task size (T_{2a}), number of partitions (np), partition free time (fp), task deadline (d), number of tasks size in first group ($nt1$), number of tasks size in second group ($nt2$)
Output: $partition(j), task\ size$
$After\ initialization\ placement$
for $each\ incoming\ task$ **do**
  for $k = 1, k \leq np\ , k++$ **do**
    if $fp(k) < d$
    $partitions \leftarrow partition(k)$
  end for
  for $j = 1, j \leq partitions\ , j++$ **do**
    $partition(j) = \min(partitions)$
    for $n1 = 1, n1 \leq nt1\ , n1++$ **do search in the first group**
      If $width\ of\ partition(j) - task\ size(n1) < 20\%$
      $partition(j)\ and\ task\ size\ (n1)\ is\ the\ answer$
      end if
      break
      end for
    for $n2 = 1, n2 \leq nt2, n2++$ **do search in the second group**
      If $width\ of\ partition(j) - size\ task(n2) < 20\%$
      $partition(j)\ and\ task\ size\ (n2)\ is\ the\ answer$
      end if
      break
      end for
    $partitions \leftarrow partitions - partition(j)$
    end for
  if no partition found
    task rejected to CPU
  end if
  end for
return $partition(j), task\ size$

V. SIMULATION RESULTS AND DISCUSSION

In this section, the performance of the proposed scheduling and allocation algorithm is evaluated. For this purpose, one simulation framework in a computer with 2GHz Core 2 Dou processor and Matlab software has been developed. The hardware model is derived from an existed device, Xilinx's XCV1000 which has 6144 reconfigurable computing units. To compare the results of the proposed algorithm, two First Fit and Best Fit [16] algorithms have been considered. Characteristics of all tasks and hardware are the same for all methods. Make span, Total Rejection Ratio (TRR) and Resource Utilization (RU) are the evaluation metrics in the simulation process. Make span is the finish time of all tasks in both RPU and CPU and rejected tasks are those which executed in the CPU. RU shows the amount of fragmentation in the width of RPU. This evaluation metric can be derived from time and resource values of the RPU with equation (4).

$$Resource\ Utilization = \frac{\sum_{i=1}^{n} w_i \times t_{w_i}}{W \times T_{HW}} \qquad (4)$$

where w_i is the width of task i on the hardware, t_{w_i} is the reconfigurable and execution time of task i on the hardware, W is width of the RPU and T_{HW} is the finish time of all n tasks. In each simulation run, 500 synthetic tasks have been generated with random parameters which are shown in table 1. The range of these parameters is derived from real applications such as DCT, FFT or DWT [17]. Arrival time of incoming tasks has an exponential distribution which determines the difference time between every two tasks. Reconfiguration time of each task is dependent to the number of CLBs and $0.25\mu s$ is the time for reconfiguring one simple CLB.

Figure 3 shows the ratio of total rejected tasks among 500 synthetic tasks. As can be seen in all algorithms, by increasing the rate of incoming tasks and reducing the time interval between them, the ratio of rejected tasks has been grown up. On the other side when the time interval between tasks are large and previous tasks are finished, more free spaces existed in the RPU to place the incoming tasks, so the number of rejected tasks in this state is not significant. The proposed algorithm with scheduled criteria shows the best performance in TRR curves in all workloads compared to Best Fit and First Fit algorithms. Although in low workload the rejection ratio of algorithms is nearly the same, the scheduled metric placed more tasks in the high workload. In the scheduled metric, the spaces with sooner free time will be filled first with the tasks and there is more time between scheduling and deadline time of tasks. Figure 4 shows the make span of the algorithms which the scheduled metric algorithm has less finish time between other algorithms. The trend in this figure is the same in the figure of TRR. Figure 5 shows the resource utilization of algorithms. It has to be said that resource utilization is dependent on the rate of incoming tasks and included peak intervals. This point is the interval between high and low rate of incoming tasks. With lower rate of incoming tasks more RPU resources will be wasted and the resource utilization is decreased and with higher rate of incoming tasks more resource utilization factor can be achieved. Similar previous figures, scheduled metric has higher resource utilization factor compared to other algorithms. Table 2 summarizes the comparison between proposed algorithm and two First and Best Fit algorithms in heavy and light workload conditions. In both heavy and light workloads the proposed algorithm achieved a significant improvement because resources and configurations have been selected properly without making any unused implementation region.

VI. CONCLUSION

In this paper, a partition-based scheduling algorithm for placing incoming tasks in the reconfigurable hardware was proposed. While fragmentation is the most important factor in performance degradation, the flexibility of using multiple configurations for each task made our proposed algorithm to be more efficient in optimum utilization of hardware surface. Configuration groups were derived by considering the routing and resource wasting constraints and scheduled time metric was defined to select the appropriate pair of configuration and partition. Our simulation results showed that a significant improvement in a wide range of workload has been achieved in comparison with two First and Best Fit algorithms. With the feature of multiple configurations, our proposed algorithm increased the resource utilization of reconfigurable hardware resources.

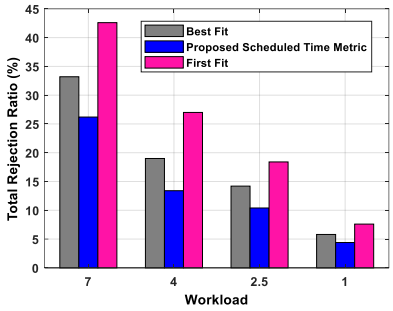

Figure 3. Rejection ratio results for heavy and light workload

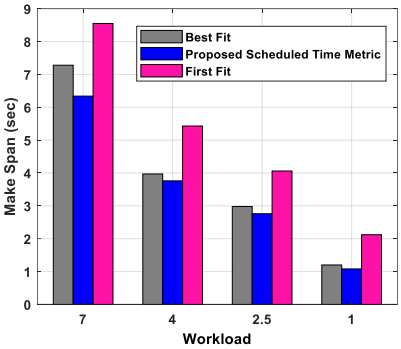

Figure 4. Make span results for heavy and light workload

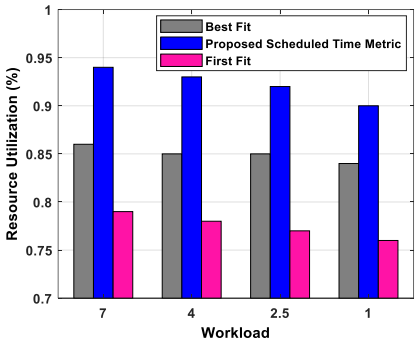

Figure 5. Resource utilization results for heavy and light workload

Table 1. Performance improvement of proposed partition-based scheduling approach

Improvement	Evaluation Metrics	Heavy Workload	Light Workload
Scheduled metric / First Fit	TRR	38.49%	42.10%
	Make Span	25.85%	53.77%
	RU	13.18%	15.55%
Scheduled metric / Best Fit	TRR	21.08%	24.14%
	Make Span	12.91%	18.33%
	RU	4.44%	6.66%

REFERENCES

[1] Awad, M., "FPGA supercomputing platforms: a survey" *IEEE International Conference on Field Programmable Logic and Applications*, 2009, pp. 564-568.

[2] Emmert, J.M., Stroud, C.E. and Abramovici, M., "Online fault tolerance for FPGA logic blocks." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 15, 2007 pp.216-226.

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

[4] Tabatabaei, A., Mosavi, M.R., Khavari, A. and Shahhoseini, H.S., "Reliable urban canyon navigation solution in GPS and GLONASS integrated receiver using improved fuzzy weighted least-square method." *Wireless Personal Communications*, Vol. 94, 2017, pp.3181-3196.

[5] Rad, H.J., Azarafrooz, M., Shahhoseini, H.S. and Abolhassani, B., "A new adaptive power optimization scheme for target tracking wireless sensor networks." *IEEE Symposium on Industrial Electronics & Applications*, 2009, pp. 307-312.

[6] Liu, W., Li, W., Un, P.S. and Cho, Y.B., "High-throughput HW-SW implementation for MV-HEVC decoder," *International SoC Design Conference (ISOCC)*, 2018, pp. 226-228.

[7] 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, 2018 pp.198-206.

[8] Mohtavipour, S.M. and Shahhoseini, H.S. "A Link-Elimination Partitioning Approach for Application Graph Mapping in Reconfigurable Computing Systems," *The Journal of Supercomputing*, 2019, pp. 1-29.

[9] Liang, H., Sinha, S., Warrier, R. and Zhang, W., "Static hardware task placement on multi-context FPGA using hybrid genetic algorithm." *IEEE 25th International Conference on Field Programmable Logic and Applications (FPL)*,2015, pp. 1-8.

[10] Banerjee, S., Bozorgzadeh, E. and Dutt, N.D., "Integrating physical constraints in HW-SW partitioning for architectures with partial dynamic reconfiguration." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, 2006, pp.1189-1202.

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

[12] Bassiri, M.M. and Shahhoseini, H.S., "A new approach in on-line task scheduling for reconfigurable computing systems." *IEEE International Conference on Application-specific Systems, Architectures and Processors*, 2010, pp. 321-324.

[13] Bassiri, M.M. and Shahhoseini, H.S., "Mitigating reconfiguration overhead in on-line task scheduling for reconfigurable computing systems." *2nd International Conference on Computer Engineering and Technology*, Vol. 4, 2010, pp. 391-397.

[14] Wassi, G., Benkhelifa, M.E.A., Lawday, G., Verdier, F. and Garcia, S., "Multi-shape tasks scheduling for online multitasking on FPGAs." *IEEE 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, 2014, pp. 1-7.

[15] Huang, M., Narayana, V.K., Simmler, H., Serres, O. and El-Ghazawi, T., "Reconfiguration and communication-aware task scheduling for high-performance reconfigurable computing." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, Vol. 3, 2010, p.20-34.

[16] Marconi, T., "Online scheduling and placement of hardware tasks with multiple variants on dynamically reconfigurable field-programmable gate arrays," *Computers & Electrical Engineering*, Vol. 40, 2014, pp.1215-1237.

[17] Bassiri, M.M. and Shahhoseini, H.S., "Configuration reusing in on-line task scheduling for reconfigurable computing systems," *Journal of Computer Science and Technology*, Vol. 26, 2011, pp.463-474.