A Time-based Leakage-aware Algorithm for Task Placement and Scheduling Problem on Dynamic Reconfigurable FPGA

Tingyu Zhou*‡, Tieyuan Pan*, Zhiguo Bao† and Takahiro Watanabe*
*Graduate School of Information, Production and Systems, Waseda University, Japan
†College of computer and information engineering, Henan University of Economics and Law, China
‡Email: shu-yu@asagi.waseda.jp

***Abstract*—Field-programmable gate array (FPGA) has enormous potential in the field of Integrated Circuit (IC) due to its programmability, short design cycle, and high flexibility in parallel computing. Nevertheless, increasing chip integration and shrinking transistor size lead to non-negligible power dissipation in FPGA. Specifically, leakage power dissipation issue as a crucial part of power consumption in FPGA requires being concerned urgently. In this paper, a time-based leakage-power aware algorithm (TBLA) is proposed to address the aforementioned issue on 2D dynamic partial reconfigurable FPGA. Experimental results show that the proposed TBLA algorithm reduces the leakage-power and scheduling overhead without increasing the overall execution time of an application compared to traditional algorithms.**

I. Introduction

In Reconfigurable Computing (RC) systems, a reconfigurable device such as Field-programmable Gate Array (FPGA) as a kind of Integrated Circuit (IC) is designed for doing cooperation with a host processor to speed up complex computation [1], such as large matrix computation, imagine processing and Convolutional Neural Network (CNN) algorithm etc,. In general, a reconfigurable FPGA is constructed by an array of Configurable Logic Blocks (CLBs) and other programmable resources so that the logic function of it can be converted to satisfy different requirements of designers [2]. Recent generations of FPGA (such as Xilinx Virtex-7) [3] have an ability of dynamic partial reconfiguration, which allows the RC system to dynamically change a part of FPGA circuits at runtime without affecting the operation of other parts. Therefore, the FPGA has higher flexibility compared to traditional Application Specific Integrated Circuit (ASIC) since the function of ASIC cannot be changed once manufactured [1]. Potential high flexibility, full-custom of functionality and short market time make dynamic partial reconfigurable FPGA become a research hotspot in recent years.

Same as traditional IC, shrinking transistor size (especially in 90nm and below) and increasing integration density make power dissipation become a significant problem on the reconfigurable FPGA. In addition, the principle of dynamic partial reconfiguration produces more serious leakage-power than that of traditional IC. Since resources on the reconfigurable FPGA cannot always be fully utilized due to partial reconfiguration, which produces idle regions on the FPGA with active transistors so as to lead to significant leakage current. In literature, 'power gating' technique is proposed to reduce the leakage-power through gating supply power of idle regions [4][5]. However, the 'power gating' method cannot be applied to reduce the leakage-power caused by 'configuration prefetching' technique for improving the performance of the FPGA.

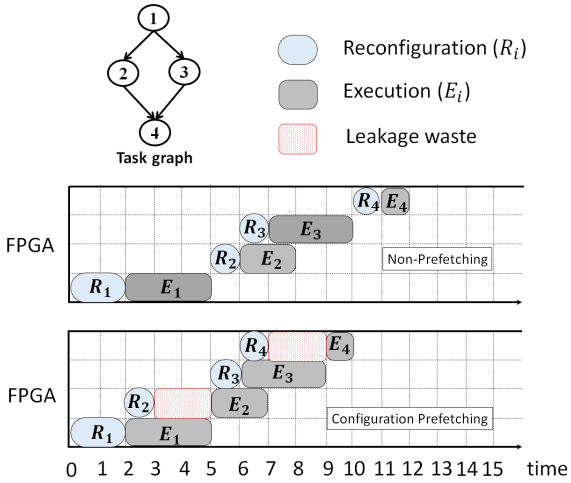


Fig. 1. An example of non-prefetching and prefetching technique

'Configuration prefetching' technique is applied to minimize the negative impact of additional 'reconfiguration time', which is the main drawback to limit the performance of the FPGA. Since tasks have to spend extra time to be reconfigured on the FPGA in advance to load necessary resources, the application's execution cycle much longer due to the additional reconfiguration time. The 'configuration prefetching' technique tries to configure tasks as soon as possible to hide the reconfiguration time [6][7], thereby improving the execution efficiency of the application.

Fig.1 shows different scheduling results with 'non-prefetching' and 'configuration prefetching' technique for targeted task graph. In this example, we assume that all tasks (t_1-t_4) arrive at the system at time 0 and the reconfigurable FPGA can accommodate up to two tasks simultaneously. Besides, the configuration of different tasks cannot be done at the same time due to the constraint of one configuration

978-1-7281-0120-0/18/$31.00 ©2018 IEEE

port. In the situation of 'non-prefetching', the task does not begin configuring on the FPGA until all its predecessors have finished executing. In contrast, the 'configuration prefetching' technique configures t_2 and t_3 (R_2 and R_3) as soon as possible once the resources on the FPGA are enough, which make the overall execution time of the task graph be reduced efficiently. Nevertheless, even if the configuration information of t_2 and t_3 has been loaded on FPGA for a long time, t_2 and t_3 as successors of t_1 have to wait until the data from t_1 is received to begin execution, that leads to the separation of reconfiguration and execution process. The time gap between them results in significant leakage current, which cannot be reduced by gating circuits since the reconfiguration data must be kept on the device and is used in the execution process.

In order to reduce the leakage-power generated by 'configuration prefetching' technique, some leakage-power aware task scheduling and placement algorithms are proposed in literature [8][9][10]. However, most of the existing algorithms are for one-dimensional FPGA and the entire task graph information is known in advance, so that the scheduling scheme can be found through long-term iteration and optimization. However they are not suitable for time-critical dynamic partial reconfigurable FPGA. On the other hand, almost all research for task scheduling and placement problem determine task execution order and placement location based on the current status of FPGA resources. Once a task cannot be placed on FPGA at the current time, it needs to be rescheduled and leads to a time waste.

In this paper, we propose a time-based leakage-power aware algorithm, which tries to schedule and place task according to the FPGA status from current to future to satisfy the needs to reduce the leakage-power caused by 'configuration prefetching' technique for two-dimensional dynamic partial reconfigurable FPGA without affecting performance. The rest of this paper is organized as follows. In section II, state-of-art about the reduction of leakage-power are introduced. Important models and problem definition are proposed in Section III, Section IV introduces the proposed leakage-power aware algorithm, Section V shows experimental results to verify the efficiency of the proposed algorithm. Section VI summarizes this paper.

II. Related works

Task scheduling algorithm for reconfigurable FPGA has been studied for many years. Steiger et al. [11] firstly focused on scheduling problem and proposed Horizon and Stuffing heuristics for 1D and 2D FPGA models. However, these algorithms simplify the task model so as to ignore task dependencies and reconfiguration overhead. Obviously, the leakage-power issue is also completely unconsidered.

Yuh et al. [8] firstly proposed a two-stage scheduling approach that addressed the leakage reduction of 1D static partial reconfigurable FPGA. Firstly, a performance-driven scheduler is proposed to minimize the overall execution time of an application to obtain a available placement. Then, an optimal algorithm based on Integrate Linear Programming (ILP) is proposed to refine this placement to achieve near-optimal. Hsieh et al. [9] proposed an enhanced leakage-aware scheduling algorithm which consists of 3 phases: binding, priority dispatching, and split-aware placing, to minimize the leakage-power without sacrificing the task schedulability based on the same model in [8].

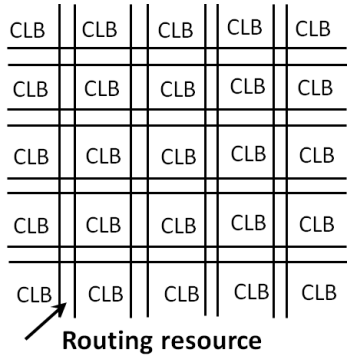


Fig. 2. Two-dimensional FPGA (5×5) model

However, these works focused on 1D FPGA and the information of whole task graph is known in advance so that the system can get optimal scheduling result by disposing all tasks for a long time, while they are not suitable for run-time system because the system can only obtain task information when a task arrives at system and need to do scheduling and placement decision quickly.

Wang et al.[10] proposed a heuristic approach to adress the leakage-power dissipation problem on 2D reconfigurable FPGA. Their proposed LBHS2D algorithm focuses on the trade-off between the application completion time and leakage-power. Nevertheless, same as most traditional scheduling and placement algorithms, the limitation concentration of FPGA status at the current time results in rescheduled time waste especially in the system where tasks come frequently and FPGA resources are less.

III. System model and problem definition

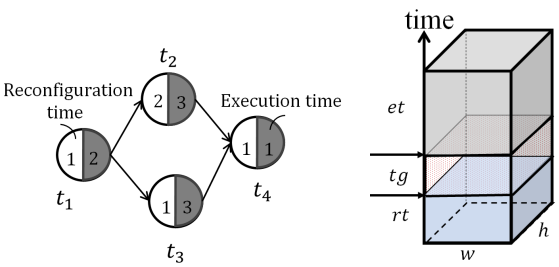
In this section, we present several models firstly. Then, problem definition and system constraints are proposed.

A. FPGA Model

Targeted FPGA is two-dimensional (2D) architecture, where Configuration Logic Blocks (CLBs) as reconfigurable units, are evenly distributed in rows and columns. There are unlimited routing resources and tasks can be loaded and executed at arbitrary positions on the device. Moreover, dynamic partial reconfiguration is supported: a part of the FPGA can be reconfigured while other parts continue to execute without suspending. As Fig. 2 shows, we define the FPGA as $F\ (W \times H)$, where W and H are the number of CLBs in the horizontal and vertical direction respectively.

B. Time-based Task Model

In our model, the application is divided into a number of hardware tasks, which can be synthesized and implemented on



(a) DAG (b) Time-based task

Fig. 3. Directed Acyclic Graph (DAG) and time-based task model

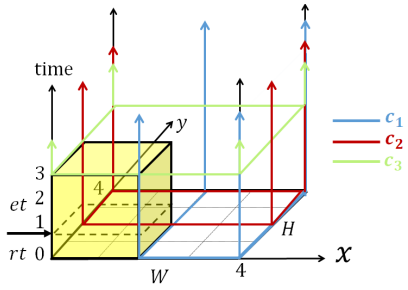


Fig. 4. An FPGA and a task, shown in three-dimensional space-time

targeted 2D FPGA platform. Directed Acyclic Graph (DAG) is used to represent the dependency relationship between tasks. Fig. 3(a) shows an example of the task graph, where each node and directed edge represent a task and the dependency between tasks, respectively.

A hardware task t_i is denoted as $t_i(w, h, at, rt, et, < t_{pre} >)$, where w, h and at are task's width, height and arrival time, respectively. In dynamic partial reconfiguration, the whole task graph information is unknown in advance. Thus, each task only contains basic information of itself and a task set of its predecessors $< t_{pre} >$. Different from software task, the hardware task has two major processes: reconfiguration and execution. In the reconfiguration process, the necessary resources for task execution are loaded on the FPGA. A task cannot begin to be executed until finishing reconfiguration and receiving data from its predecessors. rt and et are defined as the reconfiguration and execution time of the task.

In past literature, the task is modeled as a two-dimensional rectangle based on the spatial dimension. In fact, taking into account both in time and space to determine the position of the task rather than merely focusing on the current status of the FPGA, can make better utilization of device resources in entire time dimensional. As Fig. 3(b) shows, the basic hardware task is extended into a time-based task $t\_t_i(w, h, at, rt, et, rst, tg, est, < t_{pre} >)$ in this work, which is three-dimensional architecture in space-time. Compared with the 2D model, additional parameters rst, est and tg in 3D time-based task model are defined as the reconfiguration start time, execution start time and the time gap between task reconfiguration and execution process, respectively. The time gap tg and the task size determine the leakage-power together. Thus, we define the leakage-power(LK) of an application as follows:

$$LK = \sum_{i=1}^{N} w_i \times h_i \times tg_i \tag{1}$$

where N is the number of tasks in this application and i is the index for each task.

C. Time-based Maximal Empty Cuboid

Reconfigurable resources (CLBs, etc,.) on an FPGA are limited, thus plenty of data structures, such as quad-corner [12], maximal empty rectangle [13] and vertex list [14], are proposed to manage 'free space' on it. The definition of 'free space' is given as follows:

Definition 1. *The regions on an FPGA that have not been used by other tasks are defined as 'free space' and the new task can be configured and executed on them.*

Above data structures only represent the FPGA resources utilization status at the current time, which make the task that cannot be placed on the FPGA immediately, have to wait to be rescheduled, thus increasing the scheduling overhead and decrease the efficiency of the run-time system. In this paper, a time-based maximal empty cuboid (TMEC) is proposed to represent the free space on the 2D FPGA in time dimensional. The definition of TMEC [15] is given as follows:

Definition III.1. *A TMEC $c_i(cx_i, cy_i, cw_i, ch_i, cst_i, cft_i)$ is an empty cuboid that cannot be fully covered by other ones and in which a 2D task may be configured and executed.*

where (cx_i, cy_i) is the left-most front corner coordinate of the TMEC c_i. cw_i, ch_i, cst_i and cft_i are the width, height, start time and finish time of the free region. It is noticed that, the value of cft_i may be infinity because the time is unlimited.

Fig. 4 shows an example of TMEC, where a task (2, 2, 0, 1, 2, $\emptyset$) is placed in position (0, 0) at time 0. Three TMECs, $c_1(2, 0, 2, 4, 0, +\infty)$, $c_2(0, 2, 4, 2, 0, +\infty)$ and $c_3(0, 0, 4, 4, 3, +\infty)$, are generated, which are marked with blue, red and green frames in Fig. 4 respectively.

D. Problem Definition

The problem targeted in this work considers the following set of inputs, constraints, and objective.

1) Input: An application (task graphs) and FPGA structure (number of CLBs in rows and columns, one reconfiguration controller).

2) Constraints: Tasks in DAG have to obey the dependency to wait its predecessors end execution and reconfiguration completed. Multiple tasks cannot be configured simultaneously due to one reconfiguration controller limitation.

3) Objective: Minimize the leakage-power (LK) caused by the time gap between the reconfiguration and execution processes, without increasing the overall execution time of the application.

Algorithm 1 Proposed Time-based Leakage Aware Algorithm

Require: Task graph: G;
Waiting task list: WL;
Running task list: RL;
TMEC list: C;
Current time: $time$;
Earliest start execution time of a task: ESET
Latest end time a task's predecessors: PET
Reconfiguration controller available time list: $A < a_0(s_0, e_0), a_1(s_1, e_1), ...a_i(s_i, e_i) >$

Ensure: Schedule all tasks in task graph G with minimal leakage-power (LK).

1: **while** $G \neq \emptyset$ **do**
2: Add all tasks that $at_i \geq time$ into WL;
3: t_i = **SelectNextScheduledTask**(WL); // Scheduling
4: Get the latest end time of t_i's predecessors PET_i;
5: **FindBestTMEC**($t_i, C, time, A$); //Pre-placement
6: **if** $PET_i < ESET_i$ **then**
7: **OverallExecutionTimeOpimize**($t_i < pre >$, t_i, A, C); //Optimization
8: **end if**
9: Update TMEC list C;
10: Update reconfiguration list A;
11: Remove task t_i from WL;
12: Add task t_i into RL;
13: **end while**

IV. Proposed approach

An overview of the proposed time-based leakage-power aware algorithm is provided in Algorithm 1. The approach consists of three main stages: scheduling (line 3), pre-placement (line 5) and optimization (line 7). At first, tasks arrive at the FPGA according to their arrival time. If multiple tasks are waiting to be scheduled at the same time, the scheduler will calculate the priority for each task and select one task with the highest priority to be loaded on the FPGA (line 3). Then the placer searches the TMEC list C on FPGA and chooses the best one(line 5). If the feasible location cannot hide the reconfiguration time of the task (line 6), the system will do optimization of overall execution time (line 7). The details of the proposed approach are introduced in the following parts.

A. Scheduling

When multiple tasks with different size, reconfiguration time and execution time, are waiting for scheduling, a proper scheduling rule has a significant influence on the execution efficiency. In traditional algorithms, Larger Size First (LSF) and Earlier Deadline First (EDF) are two main rules to schedule tasks. However, it is not suitable for the scheduling of DAG due to the existence of task dependencies. In this paper, a priority function, which aims at hiding the reconfiguration time of a task as much as possible under consideration of task predecessors, is proposed as follows:

$$F = \alpha \times RT - \beta \times PET \tag{2}$$

where,

- RT: reconfiguration time of a task.
- PET: the latest end time of the task's predecessors.

α and β are coefficients corresponding to each parameter and used to decide the degree of their impact on the priority function. The longer RT means the reconfiguration time of the task is more difficult to be hidden in the execution process of its predecessors so that it should be scheduled as soon as possible. Earlier PET means the task can obtain the data from its predecessors earlier and begin to execute. All the parameters are obtained dynamically according to the status of FPGA and the task with the highest value of F is scheduled earlier.

B. Pre-placement

In this stage, a placer is used to find the best placement position for the scheduled task, whose main purpose is to minimize the time gap between reconfiguration and execution process. Therefore, we consider the two process as an inseparable whole to ensure the execution process can be done immediately after being configured. The placer tries to find a TMEC that can accommodate the task and does not generate a time gap between reconfiguration and execution as much as possible.

1) Find all feasible TMEC: In order to ensure the TEMC $c(cx, cy, cw, ch, cst, cft)$ is available for the scheduled task $t\_t(w, h, at, rt, et, rst, tg, est, < \ t_{pre} \ >)$, there are some constraints must be obeyed.

- Area constraint: $cw \geq w$, $ch \geq h$;
- Time constraint: $cft - cbt \geq rt + et$;
- Dependency constraint: $cft - et \geq PET$;

The area and time constraints guarantee that the TMEC can accommodate the task in space and time dimensional. In addition, task dependency must be considered, which means even if the scheduled task has to wait for its predecessors to end execution, there is still enough time in TMEC for the task to finish executing.

Since there is only one reconfiguration controller in the proposed FPGA model, it cannot be used to reconfigure different tasks at the same time. Therefore, the configuration constraint is shown as follows:

- Reconfiguration controller constraint:
  $\exists \ a_i(s_i, e_i) \subseteq A, \ s_i \leq cft - (rt + et)$.

2) Best TMEC selection: If lots of TMEC satisfy all constraints, the placer has to select the best one to complete the pre-placement. For each TMEC, the Earliest Start Execution Time (ESET) of the task can be calculated by Eq.3.

$$ESET = Max(PET, cst + rt) \tag{3}$$

In order to minimize the overall execution time, the TMEC with minimal ESET is selected as the best one. If lots of TMEC has the same ESET, the TMEC with minimal volume is selected.

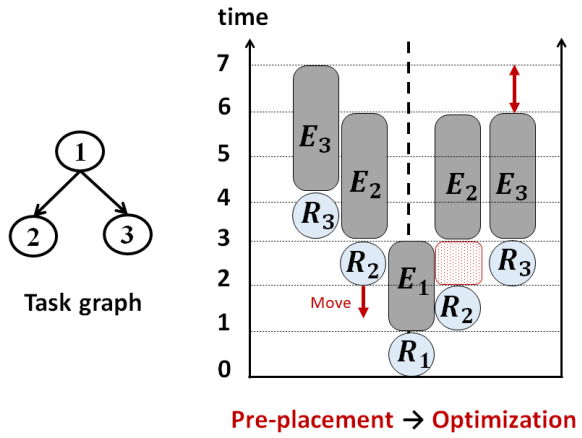


Fig. 5. An example of non-optimization and optimization

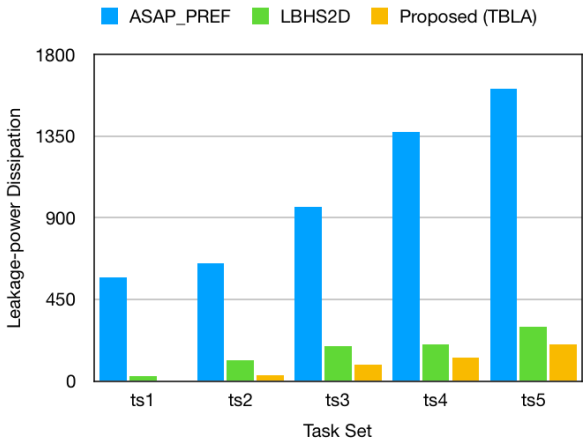


Fig. 6. Experimental results of leakage-power

C. Optimization

In the pre-placement stage, the task reconfiguration and execution are considered as a whole that cannot be separated to keep the time gap 0. However, the method will cause the overall execution time increase for the entire application. As Fig. 5 shows, t_1 and t_2 have been scheduled at the FPGA. The reconfiguration time of new arrival task t_3 cannot be hidden in the execution of t_1 since the reconfiguration controller constraint make t_3 have to wait the end reconfiguration of t_2. In the optimization stage, once the $ESET > PET$, the placer will check the previous user of reconfiguration controller and determine whether moving the reconfiguration of the previous user.

V. Evaluation

A. Simulation Setup

In this section, we construct an experimental framework in C language to implement our proposed time-based leakage-power aware algorithm (TBLA). For the purpose of evaluating the efficiency of the proposed algorithm, we compare it with existing algorithms which focus on leakage power reduction for 2D dynamic partial reconfigurable FPGA.

Based on the literature survey, we implement two algorithms for comparison. The first one is LBHS2D algorithm proposed in [10], which focuses on the trade-off between the leakage-power and overall execution time. Another one is ASAP scheduling algorithm combined with 'configuration prefetching' technique (ASAP_PREF), whose main purpose is to optimize overall execution time of the application as much as possible.

In our simulation, the targeted reconfigurable FPGA consists of 50×50 CLBs. Task Graph For Free (TGFF) tool [16] is used to generate 5 task sets ($ts1$ - $ts5$) and each one has 10 task graphs. Each task has its size (width, height), reconfiguration time, execution time, arrival time and predecessors. For each task graph in the task sets t_1 to t_5, the average number of tasks are 10, 20, 30, 40 and 50, respectively. Tasks have reconfiguration time in a range of 10±5 time units, execution time in a range of 20±10 time units, w and h in range of 10±5. All parameters are randomly generated.

The experimental environment is OS X 10.12.4, Xcode 9.4.1 on Intel(R) Core i5 CPU @2.5GHz with 8GB memory.

B. Experimental Results

1) Comparison of leakage-power dissipation:

Fig. 6 shows the experimental results of leakage-power of ASAP_PREF, LBHS2D [10] and proposed TBLA algorithms. It can be observed that, for each task set, the ASAP_PREF algorithm dissipates more energy than that of LBHS2D and proposed TBLA algorithm. It is because that the 'configuration prefetching' technique tries to load the configuration of each task as soon as possible to ensure the execution efficiency of the application, while the time gap between the configuration and execution process results in huge leakage-power. The proposed TBLA algorithm has the least leakage-power compared to others because the pre-placement stage in proposed approach considers the configuration and execution as an inseparable process so as to reduce the leakage-power as much as possible. Since the FPGA resources are limited, the leakage-power gradually increases with the increasing of task number for each task graph.

2) Comparison of overall execution time:

In order to verify whether the reduction of leakage-power leads to an increase in the overall execution time of the task sets, Application Completion Time (ACT) is detected to evaluate three algorithms, which is defined as follows:

Definition V.1. *Application Completion Time (ACT) is the time from the arrival of the first task to the end execution of the last task in a task set.*

As Fig. 7 shows, the y-axis is the number of execution time unit for each task set. It is observed that the ACT of proposed TBLA algorithm is a little longer than that of ASAP_PREF and LBHS2D, although the proposed TBLA algorithm has a significant reduction in leakage-power. It is because that the leakage-power and ACT are a trade-off, the system has to do

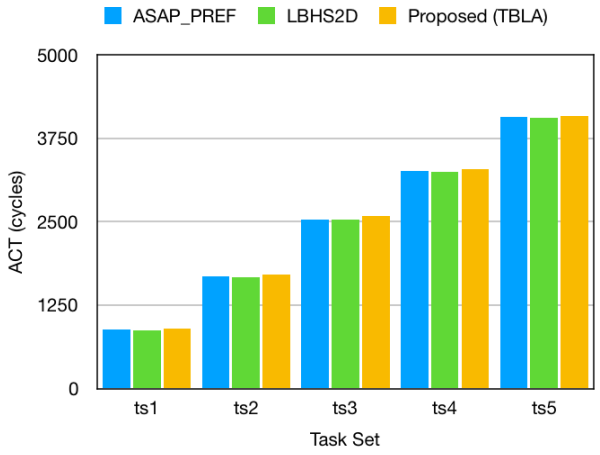


Fig. 7. Experimental results of overall execution time

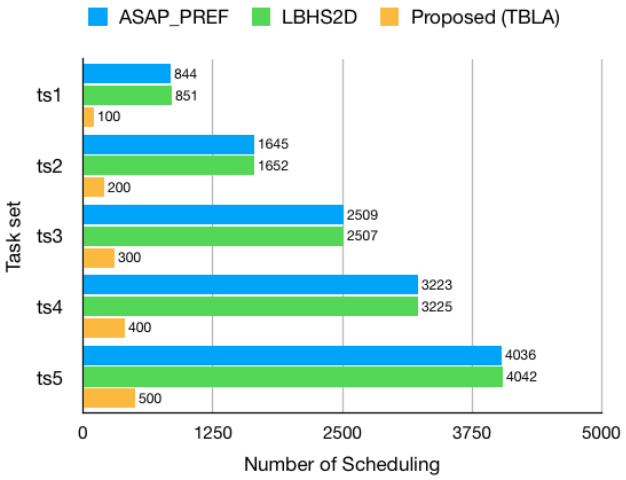


Fig. 8. Experimental results of scheduling overhead

a sacrifice in time to obtain the overall improvement of FPGA performance.

3) Comparison of scheduling overhead:

In this paper, the scheduling overhead is proposed to evaluate the performance of the new concept of TMEC, which means the number of times the scheduler is called throughout the application execution. As Fig. 8 shows, the proposed TBLA has the least scheduling overhead, because the TMEC list records the FPGA available resources from now on so that the start reconfiguration time and position for each task can be determined once it arrives at the system. The tasks that cannot be scheduled at the current time in traditional ASAP_PREF and LBHS2D algorithms, have to be rescheduled in the future and greatly increase the scheduling load of the system. As Fig. 8 shows, the average number of scheduling for each task in ASAP_PREF and LBHS2D is increased by at least 7 times compared to the proposed TBLA algorithm.

VI. Conclusion

In this paper, we proposed a time-based leakage-power aware algorithm based on a data structure of TMEC list to reduce the significant leakage-power produced by 'configuration prefetching' technique. The proposed approach consists of three stages: scheduling, pre-placement and optimization. The experimental results indicate that the proposed TBLA algorithm reduces the leakage-power and scheduling load of the device with little increase in overall execution time compared to existing algorithms.

Acknowledgment

This work was supported by Waseda University TokuteiKadai 2018K-301.

References

[1] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys (csuR)*, vol. 34, no. 2, pp. 171–210, 2002.

[2] S. Donthi and R. L. Haggard, "A survey of dynamically reconfigurable fpga devices," in *System Theory, 2003. Proceedings of the 35th Southeastern Symposium on*. IEEE, 2003, pp. 422–426.

[3] Xilinx, "7 series fpgas configuration user guide," 2018.

[4] R. P. Bharadwaj, R. Konar, P. T. Balsara, and D. Bhatia, "Exploiting temporal idleness to reduce leakage power in programmable architectures," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. ACM, 2005, pp. 651–656.

[5] J. H. Anderson and F. N. Najm, "Active leakage power optimization for fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 3, pp. 423–437, 2006.

[6] S. Hauck, "Configuration prefetch for single context reconfigurable coprocessors," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*. ACM, 1998, pp. 65–74.

[7] A. Morales-Villanueva, R. Kumar, and A. Gordon-Ross, "Configuration prefetching and reuse for preemptive hardware multitasking on partially reconfigurable fpgas," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 1505–1508.

[8] P.-H. Yuh, C.-L. Yang, C.-F. Li, and C.-H. Lin, "Leakage-aware task scheduling for partially dynamically reconfigurable fpgas," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 4, p. 52, 2009.

[9] J.-W. Hsieh, Y.-H. Chang, and W.-L. Lee, "An enhanced leakage-aware scheduler for dynamically reconfigurable fpgas," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*. IEEE Press, 2011, pp. 661–667.

[10] S. Wang, P. N. Khanh, A. K. Singh, and A. Kumar, "Leakage and performance aware resource management for 2d dynamically reconfigurable fpga architectures." in *FPL*, 2014, pp. 1–4.

[11] C. Steiger, H. Walder, and M. Platzner, "Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks," *IEEE Transactions on computers*, vol. 53, no. 11, pp. 1393–1407, 2004.

[12] T. Marconi, Y. Lu, K. Bertels, and G. Gaydadjiev, "A novel fast online placement algorithm on 2d partially reconfigurable devices," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*. IEEE, 2009, pp. 296–299.

[13] T. Pan, L. Zeng, Y. Takashima, and T. Watanabe, "A fast mer enumeration algorithm for online task placement on reconfigurable fpgas," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 99, no. 12, pp. 2412–2424, 2016.

[14] J. Tabero, J. Septién, H. Mecha, D. Mozos, and S. Roman, "A vertex-list approach to 2d hw multitasking management in rtr fpgas," *Design of Circuits and Integrated Systems (DCIS)*, pp. 545–550, 2003.

[15] T. Zhou, T. Pan, and T. Watanabe, "A fast online task placement algorithm on 3d partially reconfigurable devices," in *Region 10 Conference, TENCON 2017-2017 IEEE*. IEEE, 2017, pp. 427–432.

[16] R. P. Dick, D. L. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Proceedings of the 6th international workshop on Hardware/software codesign*. IEEE Computer Society, 1998, pp. 97–101.