TELKOMNIKA, Vol.15, No.1, March 2017, pp. 438~447
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v15i1.3878

A Dynamic Programming Approach to Energy-Efficient Scheduling on Multi-FPGA based Partial Runtime Reconfigurable Systems

Chao Jing*[1], Qi Song[2]
[1]Guangxi Universities Key Laboratory for Embedded Technology and Intelligent Information
[1,2]School of Information Science and Engineering, Guilin University of Technology, Guilin, 541004, China
[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
*Corresponding author, e-mail: gljingchao@hotmail.com

Abstract

This paper has been studied an important issue of energy-efficient scheduling on multi-FPGA systems. The main challenges are integral allocation, reconfiguration overhead and exclusiveness and energy minimization with deadline constraint. To tackle these challenges, based on the theory of dynamic programming, we have designed and implemented an energy-efficient scheduling on multi-FPGA systems. Differently, we have presented a MLPF algorithm for task placement on FPGAs. Finally, the experimental results have demonstrated that the proposed algorithm can successfully accommodate all tasks without violation of the deadline constraint. Additionally, it gains higher energy reduction 13.3% and 26.3% than that of Particle Swarm Optimization and fully balanced algorithm, respectively.

Keywords: *reconfigurable computing, multi-FPGA systems, dynamic programming*

1. Introduction

With the rapid development of reconfigurable computing technology, Field-Programmable Gate Array (FPGA) as one of the most representatives is widely used to various fields. On the contrary to traditional Application Specific Integrated Circuit (ASIC), such system performs outstanding with its high performance, low-cost, flexibility and reconfigurability. And many commercial FPGA systems support Partial Runtime Reconfiguration (PRTR). This allows part of the logic blocks to be reconfigured, without affecting the rest of logic blocks. Thus, many high performance systems have adopted dynamically reconfigurable system for system construction. It can be used in image encryption [1], real-time systems [2] and high speed computation [3]. Since the diversity of requirement in applications, dynamically reconfigurable systems equipped with one FPGA board could not attain various requirements. Recently, as the new emerging multi-FPGA system, they have witnessed the significant improvement of computing throughput by comparing with general-purpose processors. Therefore, multi-FPGA system is becoming popular in numerous fields [4-6].

The Multi-FPGA system has the capability in high performance, but energy dissipation is also growing. Hence, energy efficiency plays an important role in design issue. First, the less energy dissipates in systems, the less electricity bills will be paid. Second, less heat would be generated and the effort for releasing heat could be greatly saved. Many contemporary reconfigurable platforms provide mechanisms for energy efficient operations. For instance, Xilinx Vertiex-5 is designed based on CMOS circuitry [7], and allows adjustable power consumption rates. Last, less heat provides a higher reliability to reconfigurable system [8]. With the increasing temperature, FPGA fabrics are going to be more vulnerable.

It has become evident, however, that there is an intrinsic tradeoff between processing performance and energy consumption on multi-FPGA systems. To achieve lower power consumption rate, the system can decrease the operating frequency and this leads to longer processing time. In practice, it is highly desirable to complete tasks within given deadlines while minimizing the overall power consumption of the whole system.

This paper studies the crucial problem of energy efficiency scheduling on multi-FPGA

Received October 25, 2016; Revised January 2, 2017; Accepted January 28, 2017

systems without missing deadline constraint. There are several issues have to be addressed for the energy reduction on multi-FPGA systems. First, a task may use a number of logic blocks and it is required that these blocks be located together. We define this as integral allocation of logic blocks on a reconfigurable system to a task. Second, the reconfigurable systems take into account the reconfiguration overheads. It takes a non-eligible amount of time to reconfigure logic blocks in order to execute a new task. Moreover, most partial reconfigurable systems only allow one task to be executed for reconfiguration on the systems. So, the tasks on multi-FPGA systems have to distribute onto different FPGA at one time to avoid the reconfiguration overheads. Last, it is an NP-complete problem for minimizing the overall energy consumption while meeting the deadline constraints on multi-FPGA systems.

There have been a few researches on energy conversation in reconfigurable systems. Many of them are dedicated to proposing efficient hardware design in circuit or architecture level [9, 10] and system level [11, 12]. Authors in [10] have developed a dual V_{DD}/V_t voltage on LUT-Based FPGA in order to tune the overall energy consumption. In [6], DVS (Dynamic Voltage Scaling) is applied to adjust voltage level in commercial FPGAs embedded with a logic delay measurement circuit. The other researches are focusing on the scheduling technique for reduction of energy consumption in existed dynamically reconfigurable systems. In [11] a post-placement leakage ware scheduler is invoked to decrease leakage waste, which is a delay between task reconfiguration and execution. In [12], optimal algorithm is proposed for the case when a task partition over contexts is given, and approximation algorithms are developed for the case when no task partition is given. In [13], a transformable task model has been proposed based on the feature of high parallel computing on FPGA-based system. Then, they proposed an ant-colony optimization based algorithm to reduce the energy dissipation on reconfigurable system. Moreover, the work in [14, 15] the effective methods are proposed with the consideration of reconfiguration overhead as well as exclusiveness. However, all these work either neglects the reconfiguration overhead and exclusiveness or concerns single FPGA-based reconfigurable systems. It cannot be applied to solve the problem of energy-efficient on multi-FPGA systems.

Currently, most work on multi-FPGA systems [4, 6], [16-17] are concentrated on high performance. Researchers on work [18] focus on energy reduction on multi-FPGA based systems. They have proposed a heuristic algorithm for reducing the energy consumption. However, it is for online scenario. The algorithm is mainly based on ant-colony optimization. The results are suboptimal. Although there are considerable attentions on energy-efficient scheduling on multi-processor or core [19-20].

Toward challenges for energy-efficient scheduling on multi-FPGA systems, a dynamic programming based algorithm is proposed. And, a comprehensive trace-driven simulation experiments to evaluate our algorithm and results demonstrate that our algorithm successfully process all tasks without violating deadline constraint.

The main contributions of the paper are listed as follows:

1. The problem of energy-efficient scheduling on multi-FPGA system has been strictly formulated with the objective of energy reduction and deadline satisfaction.
2. The complexity of optimization energy on multi-FPGA system has been analyzed. Because it is an NP-Complete problem, we have designed and implemented a dynamic programming based energy-efficient scheduling algorithm to save the overall energy when meeting the deadline constraint.
3. We conduct the trace-driven simulations. The results have demonstrated that the overall energy conservation can be improved 26.3% and 13.3% than that of fully balanced algorithm and Particle Swarm Optimization, respectively.

The remainder of paper is organized as follows. We introduce related work in Section II. In Section III, we formally formulate the problem. In Section IV, we detail the dynamic programming based energy-efficient algorithm on multi-FPGA based reconfigurable systems. The performance evaluation methodology and discuss evaluation results in Section V. We conclude the paper in Section VI.

2. Related Work

There are lots of studies interested in performance versus power on dynamically reconfigurable systems. These works can be divided into two parts, the former one requires modifying the architecture or circuitry of FPGA, and latter one is from the view of system-level. Fei Li, et al, [10] devote their works from the aspect of developing architecture or circuit level. With the programmability of SRAM on LUT-based FPGA, they have developed a pre-define dual V_{DD}/V_t voltages on circuit. And they present a power-aware CAD algorithm to save power. C.T. Chow, et al., [9] have presented a methodology for commercial FPGA supporting DVS to adjust FPGA power consumption. To avoid the logic latency, they also use measurement circuit to determine a speed.

Moreover, from the view of system-level, many researchers adopt the scheduling techniques to reach the optimized energy designs on systems. Because the scheduling techniques can substantially help in further saving the system level power. In the work [11] a post placement method is leveraged to decrease the leakage waste. In their way, they firstly obtain the minimum-length schedule. Latter, to eliminate leakage waste, a leakage-aware scheduler is triggered to guarantee the schedule length not increasing. However, their approach only can be applied while the data has dependency among the tasks. Authors in [12] have implemented an optimal algorithms for the case while giving the task partition over contexts, and approximation algorithms are implemented while not giving the task partitions. However, these algorithms ignore the reconfiguration overhead and the exclusiveness requirement.

Some of research work focus on the reconfiguration overheads and exclusiveness for scheduling on reconfigurable system in [14, 15]. In [14], reconfigurable device model is extended which similar to parallel machine with single server model. Furthermore, a heuristic algorithm for task placement with the consideration of reconfiguration overhead is proposed. And the experimental results are demonstrated the improvement of proposed algorithm. But, they assume that reconfiguration overhead is to be equal for all tasks. Y. Lu, et al., [15] have developed an efficient task scheduling algorithm on partially dynamical reconfigurable systems with consideration of reconfiguration overhead and exclusiveness. In [13], they have proposed an ant-colony optimization based algorithm to reduce energy consumption. However, this only focus on scheduling on single-FPGA based reconfigurable system. They cannot be extended to multi-FPGA based reconfigurable systems.

A number of work for multi-FPGA work have emphasized on high performance, but they have degraded the impact on power consumption [4, 6, 16]. F. Redaelli, et al., [16] have proposed a baseline scheduling algorithm. The result can give a suggestion number of FPGAs for later use. Maxwell is a supercomputing machine have developed by FHPCA [4] (FPGA High Performance Computing Alliance) aims at providing high performance by FPGA technology for applications in oil and gas prospecting, medical imagery and financial services. In 2015, Dr. Tang [21] has given the methodology of multi-FPGA platform generation. Only BEE2 [5] mentioned power consumption by simply tuning clock rate. Work in [18] is for energy reduction on multi-FPGA based systems. But, it is for online scheduling, the results are suboptimal far away from the optimal energy reduction.

Even though there are considerable work in energy-efficient scheduling on multi-core or processor [19-20]. However, scheduling on multi-FPGA has several important aspects, such as integral allocation, reconfiguration overhead and the exclusiveness, energy minimization on multi-FPGA based reconfigurable systems. Therefore, it is important to develop an energy-efficient algorithm for multi-FPGA based systems.

3. Problem Formulation

3.1. System Model

The multi-FPGA based system is consisted of a set of M homogeneous partial runtime reconfigurable FPGAs, denoted by $F = \{F_1, F_2, \dots, F_M\}$. Each FPGA has reconfigurable logic resources arranged in 1D array of logic blocks [22]. Some latest commercial FPGAs can support the one dimensional reconfigurability. The number of logic blocks available in each FPGA are same L, and the number of logic blocks in FPGA j is denoted by $L_j = \{b_1^j, b_2^j, \dots, b_L^j\}$, where $L_1 = L_2 = \dots = L_M = L$. At any given time, the subset of L_j can be reconfigured one task. The number of reconfiguration ports in each FPGA units are limited which leads to reconfiguration overheads caused by exclusiveness. The basic frameworks of multi-FPGA based reconfigurable systems are illustrated in Figure 1. The FPGA units are connected with high speed bandwidth. The central control unit can be implemented by DSP, CPU, or FPGA. It performs as a global process function, such as resource management and speed adjustment.

Given a set of N independent and non preemptive tasks, $T = \{T_1, T_2, \dots, T_N\}$. At begin, all tasks are released, and must be completed within a given deadline $\mathcal{D}$. We assume the workload of a task is its cycle counts $C = \{C_1, C_2, \dots, C_N\}$. In our model, due to the feature of reconfigurable system, a task completion is needed to experience two steps: reconfiguration and execution. So, the cycle counts can be categorized to reconfiguration and execution cycle counts, task $j's$ cycle counts is denoted as $C_j =< C_j^r, C_j^e >, \forall j \in [1, N]$. Besides, for completion of a task, the logic blocks is needed to integrally allocate to task, which is defined as a number of contiguous logic blocks denote $R = \{R_1, R_2, \dots, R_N\}$.

According to [23], in some situations, we only take execution phase into account, if the logic blocks have already been reconfigured by the same type tasks and not overwritten by the other tasks. Note that same type tasks indicate the tasks share the same logic blocks, reconfiguration cycle and speed level in certain FPGA. For example, suppose that two tasks T_q and T_p run on F_j at speed S_j with same logic blocks and $C_q^r = C_p^r$. T_q Reconfiguration phase is earlier than that of T_p. Thus, for completion of T_p, it only needs to run execution phase, and reconfiguration phase can be ignored.

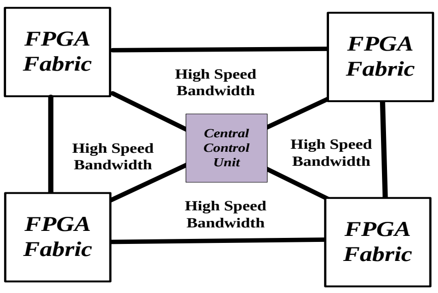


Figure 1. Illustration of Multi-FPGA based Reconfigurable Systems [19]

3.2. Energy Model

The FPGAs circuitry in our model is implemented by CMOS technology. For a CMOS circuit, the dominant power consumption is from the dynamical power dissipation, which is defined as:

$$P(s) = \mathrm{K}_{ef} ' V_{dd}^2 ' S \tag{1}$$

Where $\mathcal{K}_{ef}$, V_{dd}, S and $P(S)$ denote the effective load capacitance, supply voltage, running speed and power consumption under the certain speed, respectively.

The adjustment of voltage level aims at lowering energy consumption. However, with the lessening of V_{dd}, the circuit delay may exist. Now, since delay τ is subjected to $\tau = C/f$, where C, f is cycle count and frequency respectively. We substitute frequency to running speed S, and can be given by:

$$S = k_h ' (V_{dd} - V_t)^a / V_{dd} \tag{2}$$

Where parameters V_t, k_h and α stand for the threshold voltage, proportionality constant and velocity saturation index.

From (1) and (2) , where usually most literatures [11] use the value $\alpha = 2$ and $V_t = 0$, we obtain the power consumption function as (3).

$$P(S) = e \times S^3 \tag{3}$$

Where ε is derived from $\mathcal{K}_{ef}/{k_h}^2$ .Therefore, S is the major effective factor in power consumption.

Specifically, each FPGA unit shares a common speed with N_S levels. So, T_i running time on F_j at speed S can be calculated by $\tau(i,j) = C(i,j)/S$, while T_i with workload C_i.

Definition 1 (Runtime Set): *While the speed on FPGA unit is fixed, tasks running time can be calculated, so* Γ_j *is defined as a set of* n_j *tasks running time on* F_j*,* $\Gamma_j = \{\tau(1,j), \tau(2,j), \dots, \tau(n_j, j)\}$.

Definition 2 (Feasible Timeline): γ_j *is defined as a feasible timeline, which is formed by the* n_j *tasks placed onto* F_j *from runtime set* Γ_j*.*

Now, let suppose that n_j tasks are assigned and energy consumption at F_j can be formulated:

$$\begin{aligned} E_j &= P_j \times \Gamma_j \\ &= \varepsilon_j \times S^3 \times \sum_{i=1}^{n_j} C(i,j)/S \\ &= \varepsilon_j \times S^2 \times \sum_{i=1}^{n_j} C(i,j). \end{aligned} \tag{4}$$

3.3. Problem Statement

Given a set of N tasks with no precedence constraint, and M homogenous FPGAs in the system, each FPGA contains L logic blocks, and shares a common speed S with N_S level. With the features of FPGAs system, the reconfiguration overheads, exclusive reconfiguration, integral allocation and constrained number of availability logic blocks are required to be considered. The FPGAs energy consumption can be adjusted by tuning the speed. Under the these assumptions, for the completion of all tasks, the goal is to minimize overall energy dissipation without violating the deadline $\mathcal{D}$. It is an NP- Complete problem [24]. Now, we formally define the problem of minimizing overall energy consumption on multi-FPGA based systems (MOM) with deadline constraint:

Definition 3 (Minimum Overall Energy Consumption on Multi-FPGA based Reconfigurable Systems: MOM): The multi-FPGA systems energy minimum is defined as the goal by minimizing total energy consumption E without violating deadline $\mathcal{D}$, which define as below.

$$\begin{aligned} & \min\Big(\sum_{j=1}^{M} E_j\Big) \\ &= \min\Big(\sum_{j=1}^{M} P_j \times \Gamma_j\Big) \\ &= \min\Big\{\sum_{j=1}^{M} \Big(\varepsilon_j \times S^3 \times \sum_{i=1}^{n_j} C(i,j)/S\Big)\Big\} \\ &= \min\Big\{S^2 \times \sum_{j=1}^{M} \Big(\varepsilon_j \times \sum_{i=1}^{n_j} C(i,j)\Big)\Big\} \\ & s.t., \max\{\gamma_1, \gamma_2, \dots, \gamma_M\} \le \mathcal{D} \end{aligned} \tag{5}$$

The maximum $\gamma_{1,2,\dots,M}$ should be less than the deadline constraint. And S is proportion to $1/\gamma$.

4. Algorithm Design

4.1. Overview

Through analysis of the complexity of MOM, it is difficult to solve the MOM. From (5), we can find that the overall energy consumption is greatly impact by the maximum feasible timelines $\gamma_{1,2,\dots,M}$, which we call global time $\mathcal{G}$. Formally:

Definition 4 (Minimum Global Timeline): Given a deadline $\boldsymbol{\mathcal{D}}$, when feasible timelines in each FPGA are obtained by tasks placement, we want to minimize global timeline $\mathcal{G}$, the maximum one among feasible timelines, which can be defined as:

$$\begin{aligned} &\min(\mathcal{G}) \\ =\ &\min\{\max(g_1, g_2, \ldots, g_M)\} \\ &s.t., \mathcal{G} \le \mathcal{D} \end{aligned} \tag{6}$$

By doing so, energy-efficient scheduling algorithm with the deadline constraint on multi-FPGA system is proposed. The general steps can be described as follows: with the purpose of minimizing global timeline, FPGAs are initially set at maximum speed. Because if it is set at minimum speed level, the overall energy consumption can be conserved but it may violate the deadline constraints. Later, a dynamic programming based energy-efficient algorithm is exploited by integrating a tasks placement algorithm MPLF which is to obtain global timeline to fulfill the deadline constraint. Finally, once it gains the optimal scheduling solution. FPGAs speed are readjusted to minimize the overall energy consumption with no violating deadline constraint.

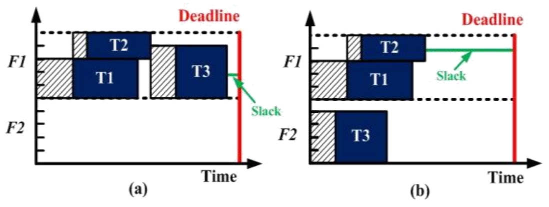


Figure 2. Illustration of Reconfiguration Overheads caused by Exclusiveness

```
Algorithm 1. MLPF
Input:
  $T_{\mathcal{N}_{Seq}}$ : A task sequence with $\mathcal{N}_{Seq}$ tasks,
  $\mathcal{M}$, $\mathcal{L}$ : Number of FPGA, and available logic blocks
  $Y_j$ : feasible timeline on $F_j$
  $\Gamma_j$ : runtime set on $F_j$
  Algorithm 1
Output:
   Global timeline $\mathcal{G}$

1.  Initially $i := 1$
2.  While $(i! = \mathcal{N}_{Seq})$ Do
3.  // Try $T_i$ to the lowest left-bottom point on FPGA $j$
4.     For $j := 1$ to $\mathcal{M}$
5.        $\Gamma[i,j] \leftarrow T_i$
6.     End for
7.    // Calculation of $\Gamma$ to obtain feasible timeline $Y$
8.     $Y[i,\mathcal{M}] \leftarrow Cal[\Gamma[i,\mathcal{M}]]$
9.  //Reserve the minimum $Y$ at $j$, and place $T_i$ onto FPGA $j$
10.    $\Gamma[i,j] \leftarrow min\ \{Y[i,\mathcal{M}]\}$
11. //Update the each FPGA points and available logic blocks $\mathcal{L}$
12.    $Update[F_1(\mathcal{L}), F_2(\mathcal{L}), \ldots, F_{\mathcal{M}}(\mathcal{L})]$
13.    $i++$
14. End while
15. Output: $\mathcal{G} \leftarrow max[Y_1, Y_2, \ldots, Y_{\mathcal{M}}]$
```

Figure 3. Pseudo Code of Algorithm 1

```
Algorithm 2.  DP-based Energy-Efficient Algorithm
Input:
  $\mathcal{N}$, $\mathcal{N}_{\mathcal{U}}$: Number of tasks, number of tasks in $\mathcal{U}$
  $\mathcal{D}$, $\mathcal{G}$: User defined deadline, feasible timeline, minimum
    global timeline
  $\mathcal{S}$ : Speed FPGAs with $\mathcal{N}_{\mathcal{S}}$ levels
  $E(\mathcal{S})$ : Overall energy consumption at speed $\mathcal{S}$
  $MFIT$
  Algorithm 2
Output:
   Minimum $E(\mathcal{S})$

1.  For $\mathcal{N}_{\mathcal{U}} := 0$ to $\mathcal{N} - 1$
2.     For $i := 1$ to $\mathcal{N}$
3.       While $(j \in \mathcal{U})$ Do
4.          $\mathcal{H}[i,\mathcal{U}] = \min\{MLPF[h(i,j) + \mathcal{H}[j,\mathcal{U} - \{j\}]]\}$
5.       End while
6.     End for
7.   End for
8.   $\mathcal{G} = min\{\mathcal{H}[i,\mathcal{U}] \mid i \in \{T_1, T_2, \ldots, T_{\mathcal{N}}\}, \mathcal{N}_{\mathcal{U}} = \mathcal{N} - 1\}$
9.   // FPGAs speed is decreased, unless violate the $\mathcal{D}$
10.        While $(\mathcal{G} \le \mathcal{D})$ Do
11.           $\mathcal{S} \leftarrow Dec(\mathcal{S})$
12.           $E(\mathcal{S}) \leftarrow \mathcal{S}$
13.           $\mathcal{G} \leftarrow Inc(\mathcal{G})$
14.        End while
15. Output: $E(\mathcal{S})$
```

Figure 4. Pseudo Code of Algorithm 2

4.2. DP-based Energy-Efficient Algorithm on Multi-FPGA Systems

In this subsection, we are going to detail the DP-based energy-efficient algorithm to obtain optimal scheduling solution. First, we will give the details of MLPF to obtain the minimum global timeline. Based on the MLPF, we next present the DP-based algorithm to gain minimum energy consumption solution.

4.2.1. MLPF: Multi-FPGA Left-bottom Point First Placement

On the contrary to the task scheduling on multi-processor and single FPGA, tasks placement to avoid the reconfiguration overheads brought by exclusiveness have to be considered in multi-FPGA based reconfigurable systems. Otherwise, the improved capability of high processing in such systems could not be fully exploited. As

Figure 2 is shown, there are two FPGA F_1 and F_2 resources arrayed in 1D model with five resources, three tasks T_1, T_2 and T_3.

Figure 2(b) outperforms than that of

Figure 2(a) both in processing time and energy saving, since much more slack to deadline are available in

Figure 2(b) to tune the speed level. Except of that, resources utilization in each FPGA is also a noticeable issue which may lead to the problem of fragmentation, hence result in prolonging global timeline. In the course of tasks placement, as long as there are not enough resources, tasks have to wait until the resources availability in terms of prolonging running time. As long as the available resources are highly fragmented, even though resources are sufficient, task cannot be fitted onto a proper FPGA. So, it is important to develop a placement algorithm on multi-FPGA systems to eliminate the reconfiguration overheads. The general idea of MLPF is as below:

Given tasks cycle counts and speed level, every FPGA can provide available placement information (left-bottom point). With the forthcoming tasks, tasks are trying to place onto the first left bottom point. Through such way, each FPGA will obtain a feasible timeline, and the minimum feasible timeline is reserved. If the minimum timeline in different FPGAs are same, the least number of placed tasks FPGAs will be selected for tasks placement. When all tasks are fitted onto FPGA units, we therefore select the maximum feasible timeline among the FPGAs as the global timeline. MLPF is detailed in Figure 3. The complexity of MLPF is bounded by $\mathcal{O}(\mathcal{N} \times \mathcal{M})$.

4.2.2. DP-based Energy-Efficient Scheduling Algorithm

Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems. And it solves the subproblem only once, thus reducing the number of computations. It is used to apply to solve NP-Complete problem, such as (TSP) Travelling Salesman Problem, Knapsack and so on.

On multi-FPGA systems, DP-based energy-efficient scheduling algorithm is adopted. Firstly, we define the recursive formula to find the minimum global timeline as well as tasks placement. Later, by the obtained results, the common speed is adjusted to lower the overall energy consumption.

Initially, MOM problem is broken into N subproblems as the number of tasks. Suppose that M FPGAs are tuned at maximum speed level to avoid violating the deadline constraint. i denotes a starting task from $T = \{T_1, T_2, \dots, T_N\}$; $\mathcal{U}$ is a set of tasks yet to be placed, and $i \cap \mathcal{U} = \emptyset$; j is one of the elements from $\mathcal{U}$. We further define $\mathcal{H}[i, \mathcal{U}]$ to document minimum global timeline that begins at T_i, and follow with task set $\mathcal{U}$. In summary, we define recursive formula as:

$$H[i, U] = \min_{j \in U}\{MFIT[h(i,j) + H[j, U - \{j\}]]\} \tag{7}$$

By iteratively doing (8), as the incremental number of placed tasks, we eventually obtain the minimum global timeline. For each iterative process the global timeline is calculated by $MFIT$.

Up to now, we have gain a completed tasks placement in multi-FPGA systems. And the global timeline is minimized not violated the deadline constraint. Now, we next need to tune the FPGA speed to minimize the overall energy consumption. The Pseudo code of DP-based Energy-Efficient Scheduling Algorithm is illustrated in Figure 4.

5. Performance Evaluation

5.1. Methodology and Setting

We adopt a competitive study, comparing our algorithm with other alternative algorithms that will be introduced in the following subsection. Overall energy consumption is the performance metric for evaluation and comparison. So, it is defined as normalized energy consumption:

$$x = E(S') / E(S_{\max}) \tag{8}$$

Where $E(S')$ and $E(S_{max})$ are overall energy consumptions after adjustment of speed level and maximum speed level, respectively.

A series of simulations are conducted to demonstrate the effectiveness of our algorithm. We evaluate our algorithms on a simulated multi-FPGA system consisting of four FPGA units, each of which include same number of continuous logic blocks denote as $\mathcal{L}$. For each simulation, we vary L in $\{100, 150, 200\}$ and by changing the number of tasks $\{12, 15, 18\}$ and workload of tasks.

Each simulation is run 10 times, and average value is taken for comparison. The experiment is conducted on Intel(R) Xeon(R) dual CPUs@2.13 GHz and 8GB of RAM. The algorithm is programmed by C#.

5.2. Compared Algorithms

We compare our algorithm with other two algorithms. Both algorithms are based on idea: (1) let tasks fit onto the FPGAs with the maximum speed level; (2) while the tasks placement is determined, the speed level is allowed to adjust.

5.2.1. Particle Swarm Optimization (PSO)

PSO algorithm is a well-known heuristic algorithm for solving combinatorial optimization problem. It has a population (swarm) of candidate solutions (particles). Given a few sample formulation, the particles move around in the search-space. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so, but not guaranteed, that a satisfactory solution will eventually be discovered. We adopt an iterative procedure that starts with an arbitrary candidate solution to problem. Then, with incrementally changing a multi-element, a better solution is attempting to find. By repeatedly doing so, the results output until it reaches the terminated condition.

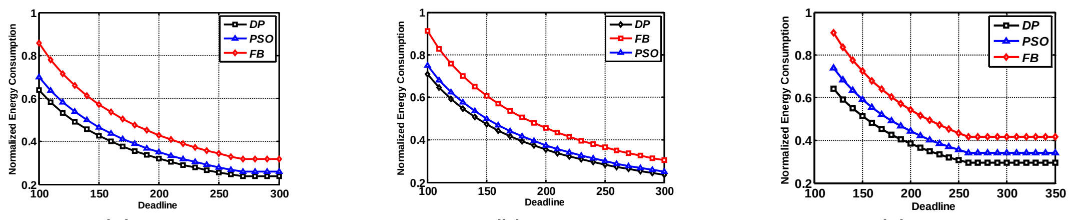
5.2.2. Fully-balanced algorithm (FB)

For the sakes of elimination reconfiguration overhead, FB can evenly assign tasks to FPGA units. Number of tasks is initially determined per FPGA unit. Later, tasks are randomly selected and placed onto FPGA units. Because speed and cycle counts are given, tasks are trying to fit onto FPGA unit and the least feasible timeline is reserved.

5.3. Comparisons

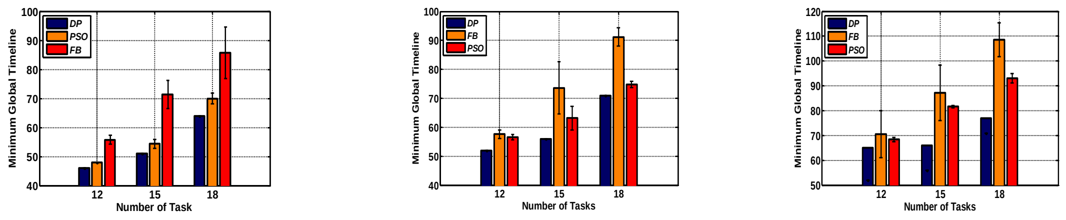
In Figure 5, the performance of the three algorithms in terms of normalized energy consumption is shown. Three logic block groups with same task number are used, i.e., 100, 150 and 200 with tasks number 18. We can see that our algorithm for energy conservation is at most 13.3% and 26.3% more than that of PSO and FB, respectively. FB performs the worst. Because FB only concentrates on the loading balance of task number in order to elimination of reconfiguration overheads caused by exclusiveness. Although the tasks are evenly fitted onto FPGA units, the FB cannot perform better than that of two other algorithms in energy reduction. PSO is better than FB, but worse than our algorithm. Because problem of MOM is a discrete problem, PSO is not suitable for solving the discrete problem. Additionally, it is easy to trap into a local optimal result and obtain the suboptimal results.

We further show comparisons of minimum global timeline in Figure 6. Three tasks set with different number of tasks and L are used. We can see that the DP-based algorithm in obtaining minimum global timeline performs best. With comparison of Figure 5 and 6, we also find that less global timeline is obtained; more overall energy dissipation can be reduced. This results well demonstrate that the importance of minimizing global timeline and effectiveness of our algorithm in tasks placement.



(a) $\mathbf{L} = \mathbf{100}$ (b) $\boldsymbol{L} = \mathbf{150}$ (c) $\mathbf{L} = \mathbf{200}$

Figure 5. Performance Comparisons of Normalized Energy Consumption



(a) $\mathrm{L} = 100$ (b) $\mathrm{L} = 150$ (c) $\mathrm{L} = 200$

Figure 6. Comparisons of Minimum Global Timeline

6. Conclusion and Future Work

This paper has proposed an energy-efficient scheduling algorithm for multi-FPGA based reconfigurable systems based on the dynamic programming. The objective is to minimize the overall energy consumption without deadline constraints. There are several challenges for the energy reduction, such as integral allocation, reconfiguration overheads caused by exclusiveness. To address these issues, we give the details of problem formulation. Then, due to the difficulty of NP-complete, we adopt the dynamic programming to solve the energy minimization problem on multi-FPGA based systems. Furthermore, for the features of multi-FPGA systems, we also devise an algorithm for task proper placement. At last, we conduct a series of simulations to demonstrate the effectiveness of our algorithms. The results show that 13.3% and 26.3% improved in energy saving than that of PSO and FB, respectively.

Acknowledgements

This research work is partially supported by National Natural Science Foundation of China (Grant No. 61563012, 61540054), Natural Science Foundation of Guangxi (Grant No. 2015GXNSFBA139260), Guangxi Universities key Laboratory Fund of Embedded Technology and Intelligent Information Processing (Grant No.2016-01-05), Research Foundation for Guilin University of Technology (Grant No. 002401003456).

References

[1] Barlian Henryranu Prasetio, Eko Setiawan, Adharul Muttaqin. Image Encryption using Simple Algorithm on FPGA. *TELKOMNIKA*. 2015; 13(4): 1153-1161.
[2] Xu Yinhui, Zeng Dazhi, Yan Tao, Xu Xiaoheng. A Real-time SAR Echo Simulator Based on FPGA and Parallel Computing. *TELKOMNIKA*. 2015; 13(3): 806-812.
[3] Tole Sutikno, Nik Rumzi Nik Idris, Auzani Jidin. High-Speed Computation using FPGA for Excellent Performance of Direct Torque Control of Induction Machines. *TELKOMNIKA*. 2016; 14(1): 1-3.
[4] HPCA (High Performance Computing Alliance). http://www.fhpca.org/maxwell.html.
[5] C Chang, J Wawrzynek, RW Brodersen. BEE2: A High-End Reconfigurable Computing System. *IEEE Design & Test of Computers.* 2005; 22(2): 114-125.
[6] J Fender, J Rose, D Galloway. *The Transmogrifier-4: An FPGA-Based Hardware Development System with Multi-Gigabyte Memory Capacity and High Host and Memory Bandwidth.* Technical Report, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto. 2006.

[7] *Virtex-5 Family Overview.* Technical Report, Xilinx. 2009.
[8] *40-nm FPGA Power Management and Advantages - Altera.* Technical Report. 2008.
[9] Chow, L Tsui, PHW Leong, W Luk, SJE Wilton. *Dynamic Voltage Scaling for Commercial FPGAs.* Proc. IEEE International Conference on Field-Programmable Technology (FPT). 2005: 173-180.
[10] F Li, Y Lin, L He, J Cong. *Low-Power FPGA Using Pre-Defined Dual-Vdd/dual-Vt Fabrics.* Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGAs). 2004: 42-50.
[11] CF Li, PH Yuh, CL Yang, YW Chang. *Post-Placement Leakage Optimization for Partially Dynamically Reconfigurable FPGAs.* Proc. ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED). 2007.
[12] NC Perng, JJ Chen, CY Yang, TW Kuo. *Energy-Efficient Scheduling on Multi-Context FPGA's*. Proc. IEEE International Symposium on Circuits and Systems (ISCAS). 2006.
[13] C Jing. *Ant-Colony Optimization Based Algorithm for Energy-Efficient Scheduling on Dynamically Reconfigurable Systems.* Proc. FCST. 2015: 127-134.
[14] J Angermeier, J Teich. *Heuristics for Scheduling Reconfigurable Devices with Consideration of Reconfiguration Overheads.* Proc. IEEE Reconfigurable Architecture Worshop (RAW) on IPDPS. 2008.
[15] Y Lu, T Marconi, K Bertels, G Gaydadjiev. *Online Task Scheduling for the FPGA-Based Partially Reconfigurable Systems.* Proc. International Workshop on Reconfigurable Computing: Architectures, Tools and Applications. 2009.
[16] F Redaelli, MD Santambrogio, D Sciuto, SO Memik. *Reconfiguration Aware Task Scheduling for Multi-FPGA Systems.* Proc. Reconfigurable Computing Workshop on HiPEAC. 2010.
[17] BeeCube Products. http://www.beecube.com/bee7.html. 2016.
[18] C Jing, Y Zhu, M Li. Energy-efficient scheduling on multi-FPGA reconfigurable systems. *Microprocessors and Microsystems - Embedded Hardware Design*. 2013; 37(6-7): 590-600.
[19] LK Goh, B Veeravalli, S Viswanathan. Design of Fast and Efficient Energy-Aware Gradient-Based Scheduling Algorithms for Heterogeneous Embedded Multiprocessor Systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS).* 2009; 20(1).
[20] TY Huang, YC. Tsai, ETH Chu. *A Near-optimal Solution for the Heterogeneous Multi-processor Single-level Voltage Setup Problem.* Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS). 2007.
[21] Qingshan Tang. Methodology of Multi-FPGA Prototyping Platform Generation. Hardware Architecture. Universit´e Pierre et Marie Curie - Paris VI. 2015.
[22] K Compton, S Hauck. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Survey.* 2002; 34(2): 171-210.
[23] S Ghiasi, A Nahapetian, M Sarrafzadeh. An Optimal Algorithm for Minimizing Run-time Reconfiguration Delay. *ACM Transactions on Embedded Computing Systems (TECS).* 2004; 3(2): 237-256.
[24] MR Garey, DS Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman. 1979.