TOWARDS A DYNAMIC EVOLUTIONARY APPROACH TO FPGA TEMPERATURE MANAGEMENT

Peter R. Lewis, Walter C. Chibamu and Xin Yao

CERCIA, School of Computer Science
University of Birmingham, UK
{p.r.lewis – wcc081 – x.yao}@cs.bham.ac.uk

ABSTRACT

Reconfigurable computing systems, such as FPGAs, provide great promise for on-the-fly adaptive computation. Such systems can be modified while in use, to deal with newly arriving computational tasks. However, how to adapt, and with respect to what objectives, is not yet well established. Typically, one might be interested to achieve an allocation of tasks, as they arrive, to differently sized regions of the FPGA, in order to provide the most efficient computation. At the same time, it is important to also maintain healthy on-chip temperature. In this paper, we propose a novel description of this dynamic FPGA temperature management problem. We formulate the problem as a dynamic multi-objective optimisation problem, with three objectives and a time-linkage characteristic. We discuss the implications of the availability or otherwise of a model of the FPGA, and propose the use of evolutionary algorithms as a method to tackle the problem.

1. INTRODUCTION

A current trend in field-programmable gate array (FPGA) technology is an increase in the number and density of programmable logic components. However, increased density results in both higher and uneven chip temperature distributions (hot spots), the management of which is becoming increasingly important [1]. One factor that influences temperature is the distribution of tasks across the chip. This calls for efficient, temperature-aware methods, able to map incoming tasks and migrate existing tasks to cooler regions, such that the overall temperature is kept as low as possible and hot spots are avoided. However, migrating tasks which are already executing incurs an overhead in terms of execution time that should also be avoided. In this paper, we formalise the thermal management of FPGAs as a dynamic multi-objective optimisation problem. The task is to find a mapping of tasks to computational cores at a series of decision points, with the aim of achieving an efficient trade-off between conflicting objectives. Though in this paper we focus on FPGAs, the problem definition and proposed solution generalise beyond current technology, and we expect will apply equally to future massively parallel reconfigurable computing platforms, where dynamic temperature management will have an even bigger role to play.

This research was conducted in the EPiCS project and received funding from the European Union Seventh Framework Programme under grant agreement n^o 257906. `http://www.epics-project.eu/` The authors would like to thank Markus Happe, Andreas Agne, Christian Plessl and Marco Platzner for their contribution to fruitful discussions.

2. BACKGROUND AND RELATED WORK

Modern FPGAs allow partial reconfiguration, where only a part of the FPGA is reconfigured (for example, a new piece of functionality is placed onto a currently idle region of the chip) while the rest of the FPGA remains active and continues to operate [2]. To enable granular chip temperature sensing, the FPGA may be partitioned into a regular grid of tiles [3]. A sensor is placed on each tile, which records the tile's temperature periodically. Processing elements (called *cores* in this paper) are laid over at least one tile. Figure 1 illustrates this idea whereby a 10×15 tile FPGA is configured into eight cores, occupying varying sized regions.

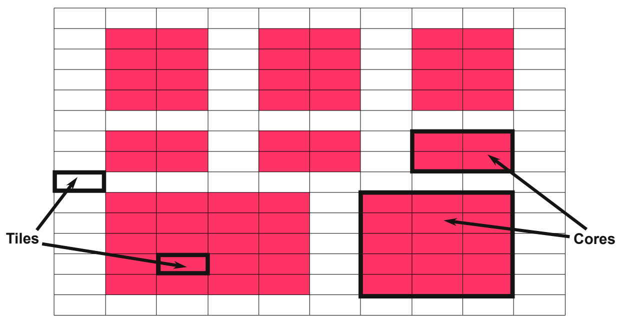

Fig. 1: Illustration of a multi-core FPGA with different sized cores and temperature sensing tiles. Example tiles and cores are indicated.

A number of *dynamic thermal management* techniques have been developed, which aim to control a chip's temperature [4, 5] at runtime. Our main contribution is to formulate the problem formally, as a dynamic multi-objective optimisation problem. A clear formalisation of the FPGA temperature management problem helps us to gain a deeper understanding of the problem and various hidden factors and assumptions that are associated with it. It helps us to focus on the most important challenges and thus to propose most appropriate solutions.

Evolutionary computation (EC) has a long and successful history of application to complex optimisation problems [6]. The main idea of evolutionary algorithms, of which there are now many variants, is to maintain a population of candidate solutions to a problem (called *individuals*), which are evaluated against an objective function (called a *fitness* function in the parlance of EC), which defines the problem to be optimised. Based on the fitness of candidate solutions, they are selected (analogously to natural selection in biological evolution), before undergoing a perturbation (called *mutation* in EC), and optionally some recombination between individuals in the population. This results in a further *generation* of candidate solutions, which are then evaluated against the fitness function. The process iterates, typically until a time budget is exhausted, or a suitably optimal solution is found.

Relatively recently, EC has also begun to be applied to *dynamic optimisation problems* [7]. These are problems where there are multiple decision points, each at which a solution is implemented. Crucially, between these multiple decision points, the problem may have changed. In the case where the changes are incremental, i.e. the objective function at time t, f_t has some similarity to that at a previous time, e.g. f_{t-1}, then the incremental exploratory nature of evolutionary algorithms lends them well to the adaptation of solutions between decision points, since it acts to transfer knowledge from past problem instances to the current one. A more formal definition of a dynamic optimisation problem is given in [8]. A more comprehensive treatment of the topic is provided by a recently edited volume [9].

Several performance measures have been developed for evolutionary dynamic optimisation [7, 8]. Depending on the assumptions present in the real world problem, different performance measures may be more appropriate. Firstly, the *online performance metric* [7] is defined as the sum or average of all fitness evaluations over the entire problem:

$$F_{online} = \sum_{t \in T'} f_t(x) \tag{2.1}$$

where T' is the set of all time points at which fitness may be evaluated, f_t is the fitness function at time $t \in T'$ and x is a candidate solution to be evaluated.

The online measure is appropriate in the case when there exists no simulation or mathematical model of the problem, and therefore each and every evaluation must be carried out in the real world. In this case, every fitness evaluation counts. This has significant implications for the exploratory behaviour of the algorithm, since any bad individuals generated through exploration of new regions of the search space, impact directly upon overall performance.

In the case when either a model of the problem may be assumed, or else when the algorithm may obtain the use of the real-world problem instance during a *training phase* that does not impact upon problem performance, we may employ an offline performance measure. An original *offline performance metric* was defined by Branke [7], which takes the average of the best individual found so far, at each time step. As Branke points out however, this does not account for the case when an individual from a previous time has a reduced fitness at later time points. Several variants of offline performance have since been defined in the literature, though they are typically tied either to generations (as in the *best of generation* measure) or else to detected changes (as in the *modified offline* measure) [8].

In the FPGA temperature management problem as described in this paper, it appears more intuitive to tie a possible offline performance evaluation to discrete decision points, which correspond to opportunities to reconfigure the FPGA. In this case, we define offline decision-based performance to be:

$$F_{offline} = \sum_{t \in T} f_t(x) \tag{2.2}$$

where $T \subset T'$ is the set of time points at which a decision is made and a solution implemented in the FPGA, a subset of the full set of time points at which the evolutionary algorithm performs fitness evaluations (on the model). Note that a decision may involve making no change to the FPGA relative to the previous decision, which may be desirable in the case when no better solution has been found in the interim.

In addition to being a dynamic optimisation problem, the FPGA temperature management problem is inherently multi-objective, since we are concerned both with computational performance of the chip as well as managing the chip's temperature. Multi-objective optimisation problems are a class of optimisation problems in which our aim is to simultaneously optimise two or more objectives [10]. Since the objectives are often in conflict, we cannot typically find solutions which obtain the global optimum for all objectives simultaneously. Instead, we aim to find Pareto-optimal solutions, i.e. those which cannot be further improved on one objective, without being strictly worse on at least one other objective. A preference model is then used to rank the found Pareto-optimal solutions, and to select one for implementation. Evolutionary computation has also long been successfully applied to multi-objective optimisation problems [11].

Evolutionary algorithms have been used to solve several optimisation problems arising in FPGA configuration (e.g. [12, 13]). However, prior work has not considered run time reconfiguration as a dynamic optimisation problem, where multiple decisions are made during run time, as the problem changes with unforseen incoming tasks and the effect of past decisions. Instead, these problems consider the entire configuration over time as a static optimisation problem to be solved ahead of time.

The problem of determining task placement online in partially reconfigurable FPGAs has also been tackled using evolutionary algorithms [14]. However, this early work did not consider the problem over time, as a dynamic optimisation problem. Instead, evolutionary algorithms were used to solve multiple independent task rearrangement problems during run time. This assumption of independence ignores the relationship between decisions, and cannot account for time linkage. Additionally, the work did not consider temperature, and though multiple performance-relevant metrics were used, a multi-objective approach was not taken. The metrics were instead combined into a single scalar objective to be optimised. Crucially however, Middendorf et al. [14], showed the effectiveness of using evolutionary computation for online management of partially reconfigurable FPGAs. In this paper, we present the first formulation of the problem as a dynamic multi-objective optimisation problem, which allows for run time optimisation, considering trade offs between conflicting objectives as well as time linkage. Our formulation is also the first to consider temperature as an additional explicit objective in such an optimisation problem, using real time readings. This is considered alongside task performance objectives, in a multi-objective setting.

3. PROBLEM FORMULATION

In the dynamic FPGA temperature management problem, we seek to schedule incoming tasks and migrate tasks currently in execution, during run time, in order to:

1. Minimise average chip temperature,
2. Minimise spatial temperature variations (hot spots),
3. Minimise the total time taken to execute all tasks.

We consider temperature measurement at the granularity of a *tile*, a small surface area over which a physical sensor records the temperature. A *core* is a processing element on which computation can be carried out. These are depicted in Figure 1. The configuration in which tasks are assigned to cores has an impact on the resultant chip temperature landscape over time. In addition, cores may be heterogeneous, such that, depending on which core it is executed, a task may take a different amount of time to complete, and may generate a thermal contribution with a different distribution.

3.1. Notation

Our problem definition makes use of the following notation:

- A finite set of discrete time intervals over which the FPGA is to be managed, $T = \{0, 1, 2, 3, \dots t_{max}\}$, s.t. $T \subset \mathbb{N}_0$ and $t_{max} = |T| + 1$.
- A single task in the set of tasks which, at time t, have arrived but have not yet been completed, $\tau \in \mathcal{T}_t$.
- The arrival time of a task, $\alpha : \bigcup_{t \in T} \mathcal{T}_t \to T$.
- The completion time of a task, $\phi : \bigcup_{t \in T} \mathcal{T}_t \to T$.
- A fixed set of *processing* cores, $C = \{c_0, \dots, c_{|C|-1}\}$, able to process incoming tasks.
- A notional *waiting* core, w, to which a task may be assigned to indicate that it is in a waiting state (i.e. the task is not currently being executed).
- A notional *unusable* core, u, to which a grid tile may be mapped to indicate that it measures an unused area of the FPGA.
- A set of grid tiles, $G = \{g_{0,0}, g_{0,1} \dots, g_{x,y}\}$, used for temperature sensing; x and y denote the number of horizontal and vertical tiles, respectively.
- A mapping of grid tiles to cores, $\sigma : G \to C \cup \{n\}$, indicating the layout of the FPGA. σ is required to be surjective.
- A mapping of tasks to cores at time t, $\mu_t : \mathcal{T}_t \to C \cup \{w\}$. μ_t is required to be injective, except that many tasks may be mapped to the *waiting* core, w.
- A reading of each tile's temperature at time t, $\theta_t : G \to \mathbb{R}$, s.t. $t \in T$.
- The ambient chip temperature, $\Theta_{ambient}$.
- The total workload of a task, in (arbitrary) workload units, $\omega : \bigcup_{t \in T} \mathcal{T}_t \to \mathbb{N}$
- The current progress of a task, more precisely a counter indicating the next workload unit of a task to be executed, $\lambda : \bigcup_{t \in T} \mathcal{T}_t \to \mathbb{N}$.
- The task-core workload rate, $\rho : \cup_{t \in T} \mathcal{T}_t \times C \cup \{w\} \to \mathbb{N}$ (i.e. how much of a task's workload is completed per time interval by a given core). $\rho(\tau, w)$ is defined to be 0 for any τ.
- The task-core reconfiguration penalty, $\pi : \cup_{t \in T} \mathcal{T}_t \times C \cup \{w\} \to \mathbb{N}$ (i.e. the time taken, in workload units, to reconfigure a given core to execute a given task). E.g. $\pi(\tau, c_1)$ indicates the workload not carried out during a time instance, when migrating task τ to core c_1. It is assumed that $\pi(\tau, c) \leq \rho(\tau, c)$ for all τ and c,

i.e. that all migrations may be completed within one discrete time step.

- The task-tile thermal contribution, $\delta : \cup_{t \in T} \mathcal{T}_t \times G \times \mathbb{N} \times C \times \Sigma \rightarrow \mathbb{R}$ (i.e. how much temperature contribution a task makes to a tile, for a given workload unit of that task, when running on a given core). Σ represents the space of possible grid-tile mappings, i.e. possible instances of σ. $\delta(\tau, g, x, c, \sigma)$ is undefined if $x > \omega(\tau)$, for all τ and any g, c and σ.

  Typically, the thermal contribution of a task τ to tile g occupied by core c running task τ (i.e. $\sigma(g) = c$ and if $\mu(\tau) = c$) during one time step will be $\sum_{x=\lambda(\tau)}^{\lambda(\tau)+\rho(\tau,c)} \delta(\tau, g, x, c, \sigma)$, except when τ is migrated to c in that time step, in which case the thermal contribution will be $\sum_{x=\lambda(\tau)+\pi(\tau,c)}^{\lambda(\tau)+\rho(\tau,c)} \delta(\tau, g, x, c, \sigma)$.

  If the core does not occupy the given grid tile, i.e. $\sigma(g) \neq c$ then $\delta(\tau, g, x, c, \sigma) = 0$ for any τ and x. In this case, any heat on tile g due to task τ will be through dissipation through the chip, and not due to a direct contribution.

3.2. Assumptions

We make the following simplifying assumptions:

- Workload rates ρ do not vary with temperature.
- For a given task and core, the task-core workload is constant throughout the lifetime of the task.
- Cores which complete a task during one time interval generate a thermal contribution as if they had executed that task for the entire time interval.
- Reconfiguration itself does not generate any heat.
- There are no data dependencies between tasks.

Additionally, we assume that since tasks arrive over time, no information about them is available until they have arrived. Therefore, one cannot reason about $\mathcal{T}_{t+n}$, where $n \geq 1$, at time t, since those tasks have not yet been seen.

3.3. Problem Framework

A problem instance begins with initial temperatures, $\theta_t(g) = \Theta_{ambient}$ for all $g \in G$. The problem instance iterates:

1. Initialisation:
   (a) $t = 0$.
   (b) **Arrival of first tasks:**
   $\mathcal{T}_t = \{\tau : \tau \in \bigcup_{s \in T} \mathcal{T}_s, \alpha(\tau) = 0\}$,
   the set of tasks arriving at time 0.
   (c) Since no task is running when it arrives,
   $\mu_t(\tau) = w$, for all $\tau \in \mathcal{T}_t$.
   (d) Set counters to the start of each arriving task: for all $\tau \in \mathcal{T}_t$, $\lambda(\tau) = 1$.
2. $t = t + 1$.
3. **Measure temperature on each tile:**
   Record values for $\theta_t(g)$, for all $g \in G$.
4. **Decide on mapping** $\mu_t(\tau)$ for each $\tau \in \mathcal{T}_t$.
5. **Reconfigure FPGA** according to μ_t, if necessary.
6. **Carry out computational work:**
   For each $\tau \in \mathcal{T}_t$,
   $$\lambda(\tau) = \begin{cases} \lambda(\tau) - \rho(\tau, \mu(\tau)), \text{ if } \mu_t(\tau) = \mu_{t-1}(\tau) \\ \lambda(\tau) - \rho(\tau, \mu(\tau)) + \pi(\tau, \mu(\tau)), \text{ otherwise.} \end{cases}$$
   During this step, the thermal contributions are made and heat dissipates.
7. Build $\mathcal{T}_{t+1}$:
   (a) **Remove completed tasks:**
   $\mathcal{T}_{t+1} = \mathcal{T}_t \setminus \{\tau : \tau \in \mathcal{T}_t, \lambda(\tau) > \omega(\tau)\}$
   (b) **Record completion times of completed tasks:**
   $\phi(\tau_c) = t$,
   for all $\tau_c \in \{\tau : \tau \in \mathcal{T}_t, \lambda(\tau) > \omega(\tau)\}$
   (c) **Arrival of new tasks:**
   $\mathcal{T}_{t+1} = \mathcal{T}_{t+1} \cup \{\tau : \tau \in \cup_{s \in T} \mathcal{T}_s, \alpha(\tau) = t+1\}$.
   (d) Set initial task state for new tasks:
   For all $\tau \in \mathcal{T}_{t+1}$, if $\alpha(\tau) = t + 1$:
      i. $\mu_t(\tau) = w$, and
      ii. $\lambda(\tau) = 1$.
8. If $t < t_{max}$, go to step 2, otherwise end.

3.4. Instantaneous Optimisation Task

In the above problem framework, the task is to make the decision at each point 4, to decide the mapping from tasks to cores. This is encapsulated at each time point t, in the mapping μ_t, which an optimisation algorithm is free to set. Informally, we described the objectives which we would like to make this decision with respect to, at the start of section 3. We now formalise these objectives for a given time point, t:

1. Minimise average chip temperature, specifically:
   $$\textit{minimise } f_{1t}(\mu_t) = \frac{1}{|G|} \sum_{g \in G} \theta_t(g) \qquad (3.1)$$
2. Minimise spatial temperature variations (hot spots). There are various ways in which this could be tackled. Here we take a minimax approach:
   $$\textit{minimise } f_{2t}(\mu_t) = \max\left(|\theta_t(g) - \theta_t(h)|\right) \qquad (3.2)$$

$$\forall g \in G, \forall h \in \nu(g)$$

where $\nu(g)$ is the set of all tiles in the Moore neighbourhood of g.

3. Minimise the total time taken to execute all tasks. At a particular time instance, this may be seen as maximising the workload carried out at this time, for all currently existing tasks:

$$\textit{maximise } f_{3t}(\mu_t) =$$

$$\sum_{\tau \in \mathcal{T}_t} \rho(\tau, \mu(\tau)) - \begin{cases} \pi(\tau, \mu(\tau)) & \text{if } \mu_t(\tau) \neq \mu_{t-1}(\tau) \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The performance of a given mapping μ_t, with respect to the instantaneous optimisation objectives, f_{1t}, f_{2t} and f_{3t} is calculated based on temperature readings θ_t, the workload and penalty rates ρ and π, and an observation of required reconfigurations (mappings and migrations), due to a difference between μ_{t-1} and μ_t. In a naive way, we may now tackle the problem of selecting a μ_t for step 4, evaluating candidates for μ_t against the three objectives.

3.5. Objectives Over Time

Although decisions are made at each of the time points in T, from the problem perspective, we are interested in the objectives over the entire lifetime of the management of the FPGA, i.e. over all of T. Therefore, when evaluating the performance of a particular algorithm for determining task to core mappings, we must define metrics which quantify these over time. In section 3.4 we defined objectives for a naive decision at time t. There are many ways in which we could define complementary performance metrics for an entire problem instance, e.g. by borrowing some ideas from robust optimisation over time [15, 16, 17].

Here, we propose three performance metrics over time:

1. Minimise average chip temperature over time:

$$\textit{minimise } f_1 = \frac{1}{T} \sum_{t \in T} f_{1t}(\mu_t) \tag{3.4}$$

2. Minimise spatial temperature variations over time:

$$\textit{minimise } f_2 = \frac{1}{T} \sum_{t \in T} f_{2t}(\mu_t) \tag{3.5}$$

3. Minimise the total time taken to execute all tasks. Since, assuming all tasks completed, a simple summation of f_{3t} would not capture penalties (i.e. time wasted performing migrations), and a simple count of migrations would ignore differences in $\rho(\tau, c)$ as c varies, we instead measure finish times of tasks:

$$\textit{minimise } f_3 = \sum_{\tau \in \bigcup_{t \in T} \mathcal{T}_t} \phi(\tau) \tag{3.6}$$

It is important to note, that while decisions are made locally within each time step, the performance of any algorithm generating such decisions should be evaluated in terms of these three objectives calculated over all of T. Importantly, whether these objectives are measured online or offline will depend on the availability of a simulation or mathematical model of the problem at each decision point, with sufficient fidelity to provide meaningful fitness evaluations between decision points. Since f_1 and f_2 depend on θ_t values, which must either be measurements from the real system at time t or predictions from a model, then the lack of a model means that each and every fitness evaluation must be carried out on the real system. Furthermore, in a possible extended version of this problem, ρ may not be known, or may vary, and may indeed also need to be a runtime observation.

3.6. Time linkage

Though f_{3t} considers μ_{t-1} explicitly, f_{1t}, f_{2t} and f_{3t} are driven by various μ_s where $s < t$. This is since the current state of the FPGA at time t will be dependent on work carried out prior to time t. Specifically, both the current temperature of the grid tiles and the currently running tasks on each core, can alter the values for each of f_{1t}, f_{2t} and f_{3t} for a given input. In the language of evolutionary dynamic optimisation, this is referred to as *time linkage* [18].

The implication of this is important, since the quality of decisions made at time t is affected directly by decisions made prior to t. In some cases, this means that a seemingly good decision at one point in time, acts to limit the quality of a decision at a later point in time. For example, greedily placing an incoming task onto the core which will complete that task the quickest, may mean that a later task, which would benefit more from being placed on that core, cannot be placed there without incurring a penalty incurred by migrating the first task away first. Thus, greedily optimising the instantaneous objective functions, f_{1t}, f_{2t} and f_{3t}, is very likely not to be optimal in terms of the objectives specified in f_1, f_2 and f_3. An high performing optimisation algorithm should therefore be designed to account for this time-linkage, in the presence of uncertainty over future incoming tasks.

4. CONCLUSIONS AND OUTLOOK

In this paper we have defined a novel formulation for the problem of dynamically managing partially reconfigurable

FPGAs, with respect to completion time of incoming tasks and the management of on-chip temperature. The problem is formulated as a dynamic multi-objective optimisation problem, which enables us to explore the trade-off between computational efficiency, heat generation and temperature imbalance. Although three objectives were proposed in this paper, our formulation can easily accommodate additional objectives, e.g., energy consumption of the chip, security, reliability, etc. All these are important issues, which can now be considered under the same problem formulation.

In tackling this problem, we propose the use of evolutionary algorithms, since these have been used successfully for both dynamic and multi-objective problems for many years. Their incremental exploratory nature enables them to re-use information from previous time instances, for solutions in the present, when the problem consists of a sequence of related problem instances over time. The performance of such algorithms should be compared with that of existing heuristics, such as those described by Happe [5, pp.73–81].

We highlighted a number of challenges which need to be addressed in order to tackle this problem. Firstly, the issue of time linkage means that optimisation algorithms which are greedy with respect to time, will likely not be optimal over the entire problem instance. Instead, algorithms will need to make predictions about future tasks and chip behaviour. Secondly, techniques from both dynamic and multi-objective optimisation will need to be combined in this problem which contains both characteristics. Thirdly, a question arises on the availability of a suitable simulation or mathematical model, which would enable an evolutionary algorithm to evaluate candidate solutions between decision points. Ultimately, the availability (or otherwise) of such a model will drive the selection of appropriate evolutionary techniques.

Finally, it will be important to evaluate the overhead of the computational work required to apply evolutionary techniques to tackle this problem at runtime, in terms of the objectives considered here. Depending on the exact form the evolutionary algorithm takes (importantly, e.g. either online or offline), this overhead may actually make things worse. Indeed, regardless of the approach taken, the costs as well as benefits of a given technique should be considered. One approach could be for the decision process to be seen as a task itself, whose impact is evaluated as part of the whole system.

5. REFERENCES

[1] S. Borkar, "Thousand core chips: a technology perspective," in *Proc. 44th Annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 746–749.

[2] A. Gupte and P. Jones, "Hotspot mitigation using dynamic partial reconfiguration for improved performance," in *Reconfigurable Computing and FPGAs, 2009. ReConFig '09. International Conference on*, 2009, pp. 89 –94.

[3] M. Happe, H. Hangmann, A. Agne, and C. Plessl, "Eight ways to put your FPGA on fire – A systematic study of heat generators," in *Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE Computer Society, 2012.

[4] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *7th Int. Symp. on High-Performance Computer Architecture (HPCA)*, 2001, pp. 171–182.

[5] M. Happe, "Performance and thermal management on self-adaptive hybrid multi-cores," Ph.D. dissertation, Paderborn University, 2013.

[6] R. Chiong, T. Weise, and Z. Michalewicz, Eds., *Variants of Evolutionary Algorithms for Real-World Applications*. Springer, 2012.

[7] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

[8] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[9] S. Yang and X. Yao, Eds., *Evolutionary Computation for Dynamic Optimization Problems*. Springer, 2013.

[10] J. Branke, K. Deb, K. Miettinen, and R. Slowiski, Eds., *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer, 2008.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evol. Comp.*, vol. 6, no. 2, pp. 182–197, 2002.

[12] J. Harkin, T. McGinnity, and L. Maguire, "Genetic algorithm driven hardware-software partitioning for dynamically reconfigurable embedded systems," *Microprocessors and Microsystems*, vol. 25, no. 5, pp. 263–274, 2001.

[13] H. Fröhlich, A. Kosir, and B. Zajc, "Optimization of FPGA configurations using parallel genetic algorithm," *Information Sciences*, vol. 133, no. 3–4, pp. 195–219.

[14] M. Middendorf, B. Scheuermann, H. Schmeck, and H. ElGindy, "An evolutionary approach to dynamic task scheduling on FPGAs with restricted buffer," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1407 – 1420, 2002.

[15] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Finding robust solutions to dynamic optimization problems," in *Proc. EvoApplications 2013, LNCS 7835*, A. I. Esparcia-Alcázar *et al.*, Eds. Springer-Verlag, 2013, pp. 616–625.

[16] ——, "Characterizing environmental changes in robust optimization over time," in *Proc. 2012 IEEE Congress on Evolutionary Computation (CEC'12)*. IEEE Press, 2012, pp. 551–558.

[17] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time – a new perspective on dynamic optimization problems," in *Proc. 2010 IEEE Congress on Evolutionary Computation (CEC2010)*. IEEE Press, 2010, pp. 3998–4003.

[18] T. T. Nguyen and X. Yao, "Dynamic time-linkage evolutionary optimization: Definitions and potential solutions," in *Metaheuristics for Dynamic Optimization*, ser. Studies in Computational Intelligence, E. Alba, A. Nakib, and P. Siarry, Eds. Springer, 2013, vol. 433, pp. 371–395.