www.ietdl.org

Published in IET Computers & Digital Techniques
Received on 30th April 2014
Revised on 25th July 2014
Accepted on 31st August 2014
doi: 10.1049/iet-cdt.2014.0089

Special Issue: Energy Efficient Computing with Adaptive and Heterogeneous Architectures

ISSN 1751-8601

Coarse-grained reconfiguration: dataflow-based power management

Francesca Palumbo[1], Carlo Sau[2], Luigi Raffo[2]

[1]*PolComIng – Information Engineering Unit, University of Sassari, Sassari, Italy*
[2]*DIEE – Department of Electronics Engineering, University of Cagliari, Cagliari, Italy*
E-mail: fpalumbo@uniss.it

Abstract: Power reduction in modern embedded systems design is a challenging issue exacerbated by the complexity and heterogeneity of their architecture. In the field of Reconfigurable Video Coding (RVC), to challenge these issues and cut-down time to market, dataflow-based techniques have been adopted. In particular, to master management and composability of dynamically reconfigurable systems, the authors have developed the multi-dataflow composer. Nevertheless, despite the RVC offers several different tools, in its reference design framework power management is still an open issue. To make some steps forward towards filling this gap, in this study, they address power management for coarse-grained reconfigurable systems combining structural and dynamic strategies, both to be applied at the dataflow level.

1 Introduction

Nowadays preserving natural resources is of paramount importance; therefore power consumption of electronic devices is under scrutiny. Some hardware designers are starting to say that power is the new timing constraint. High demand of handheld embedded devices, requiring efficient execution of multiple applications while complying with battery-life limits, is pushing consumer electronics. The same apply for biomedical image processing systems (e.g. portable ultrasound devices) where mixing a low-power design approach to computational efficiency is extremely challenging.

In signal processing very often algorithms can be effectively described exploiting dataflow-based formalism. The moving picture experts group (MPEG) has defined a complete design framework enabling decoders specification, by selecting components from a given library, leveraging on the dataflow formalism [1]. Dynamic and incremental codecs configuration and reconfiguration are then favoured, although required the definition of dedicated methodologies and tools constituting, at the moment, the MPEG Reconfigurable Video Coding (MPEG-RVC) framework. Examples are the Open RVC-CAL Compiler (Orcc) [2], Xronos [3] and Turnus [4] capable of high level synthesis, automatic mapping and design space exploration. This framework, conceived to handle complexity and heterogeneity, is still growing to support new features. Brand new tools and/or updated releases are continuously integrated. However, power consumption is still an open issue.

In this context, we have demonstrated that combining the dataflow high-level formalism to a coarse-grained reconfigurable hardware design approach may lead to optimal multi-context systems, where flexibility and area minimisation are realistically challenged [5]. The outcome of those studies led to the multi-dataflow composer (MDC) tool. Its application in the Reconfigurable Video Coding (RVC) field was straightforward [6]. Nevertheless, the possibility of automatically managing the composition of reconfigurable platforms, power and area aware, made it suitable for other application domains too, such as image/video processing [7] and neural signal processing [8, 9].

While reducing the power consumption of a register transfer level (RTL) design by hand is tedious but doable, it is no longer practical for larger systems such as System on Chips. Therefore the definition of automated methodologies for power management is of primary importance to aid designers, while reducing debugging and deployment costs. In this paper, we are challenging these aspects proposing the combination of structural and dynamic strategies to be applied at the modelling level. We are going to demonstrate how topology affects power in coarse-grained reconfigurable architectures and to present the MDC profiling-aware methodology to limit it. Then, we are going to show a graph-based strategy to manage dynamic power consumption.

The proposed techniques have been assessed on an image zoom application, targeting an ASIC design flow. Our topology optimisation strategy led to find the lowest power consuming configurations. The graph-based dynamic power reduction strategy, then, provided an additional 70% of saving in power. An FPGA prototype has also been developed to explore the applicability of the proposed power-aware design flow for these devices.

The rest of this paper is organised as follow: Section 2 describes the scenario of the proposed design flow, explained in Section 3. Section 4 presents the experimental results prior to conclude in Section 5.

2 Background

In this section, we are going to present the context of application we are referring to (Section 2.1), the basic instrument (Section 2.2) we have extended to provide power management and the foundations of the proposed techniques (Section 2.3).

2.1 Dataflow model of computation and RVC

Dataflow models are very well known and have a rich history, dating back to the work by Dennis [10] and Kahn [11]. A *dataflow program* is a directed graph where nodes represent computational units (*actors*), while edges represent loss-less, order-preserving point-to-point connections (*channels*) between actors, which are used to communicate sequences of data packets (*tokens*). In terms of notation, let us define DFG$\langle V, E\rangle$ as a directed graph, where V is the set of vertices of the graph (the actors) and E is the set of edges (the channels).

Several variations of this kind of dataflow model have been introduced in the literature [11–13], often referred to as different dataflow *Models of Computation*. In the MPEG-RVC context, the *Dataflow Process Network* model [13] with firing rules is used, because of its expressiveness and for the existence of a formal programming language (the CAL actor language) supporting it [14]. In a CAL dataflow network, as depicted in Fig. 1, the actors are abstract representations of computing elements that can be transformed either into software agents or into physical functional units. All the actors concur asynchronously to the computation, generating output token sequences from their inputs. Actions are enabled according to specific input token sequences and *guard* expressions, possibly leading to state variations and/or to tokens emission.

2.2 Coarse-grained reconfigurable systems: automatic multi-dataflow network composition

The MDC tool was conceived for the automatic creation and management of 'multi-dataflow' systems. It was meant to address the difficulty of mapping different applications onto a coarse-grained reconfigurable architecture [15, 16]. Its final goal is automating such a mapping process while minimising hardware resources, with consequent area/energy saving [5, 7, 17]. MDC works at a high level of abstraction. This means that, as soon as the different specifications are acquired and transformed into directed graphs, it does not matter the starting dataflow model. At the moment, as it can be seen in Fig. 2, the MDC tool is directly coupled to Orcc [2], so that Dataflow Process Networks (expressed as XDF files) are processed. The XDF, XML Dataflow Format, is an XML dialect, used to describe Dataflow Process Networks and standardised by MPEG. In contexts different by the RVC one, it would still be possible using MDC by removing Orcc and defining just an *ad-hoc* parser.

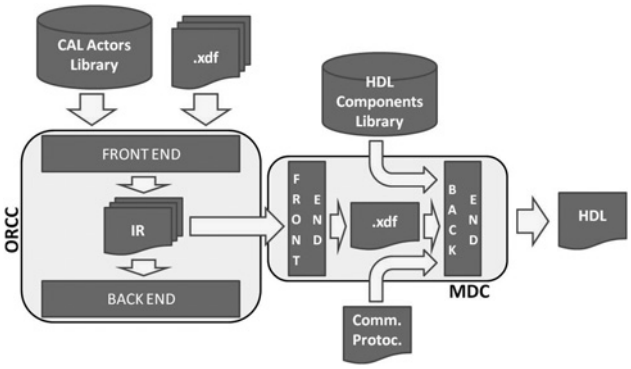

Fig. 2 *MDC tool: an overview*

In the current flow, the MDC front-end leverages the Intermediate Representation (in java code) generated by the Orcc front-end, to assemble a single multi-dataflow specification. The MDC back-end, then, creates the respective HDL coarse-grained reconfigurable hardware, mapping each actor on a different functional unit. These latter are passed as input to the MDC within the HDL components library and can be manually or automatically created [3, 18]. MDC is capable of assembly any type of reconfigurable platform, without any knowledge of the units granularity/functionality.

MDC maximises resource sharing among the given input specifications and implements quick, single-cycle, reconfiguration based on low overhead switching blocks (*Sboxes*) placed at the crossroads between the different paths of data. Sboxes are simply responsible of data routing, without any computational overhead.

Fig. 3 shows an application example of the standard MDC flow. The Orcc front-end is used to import all the input networks ($\alpha.xdf$, $\beta.xdf$, $\gamma.xdf$, $\delta.xdf$ and $\epsilon.xdf$) that, once flattened and transformed into their respective directed

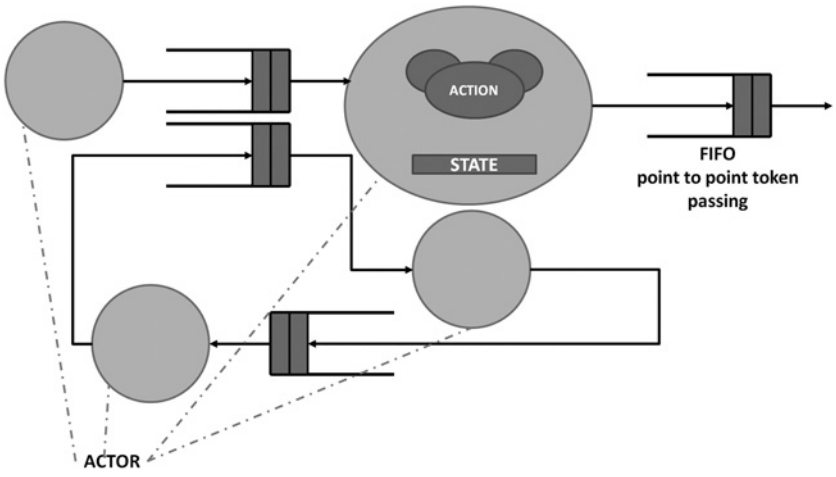

Fig. 1 *CAL dataflow network structure*

doi: 10.1049/iet-cdt.2014.0089

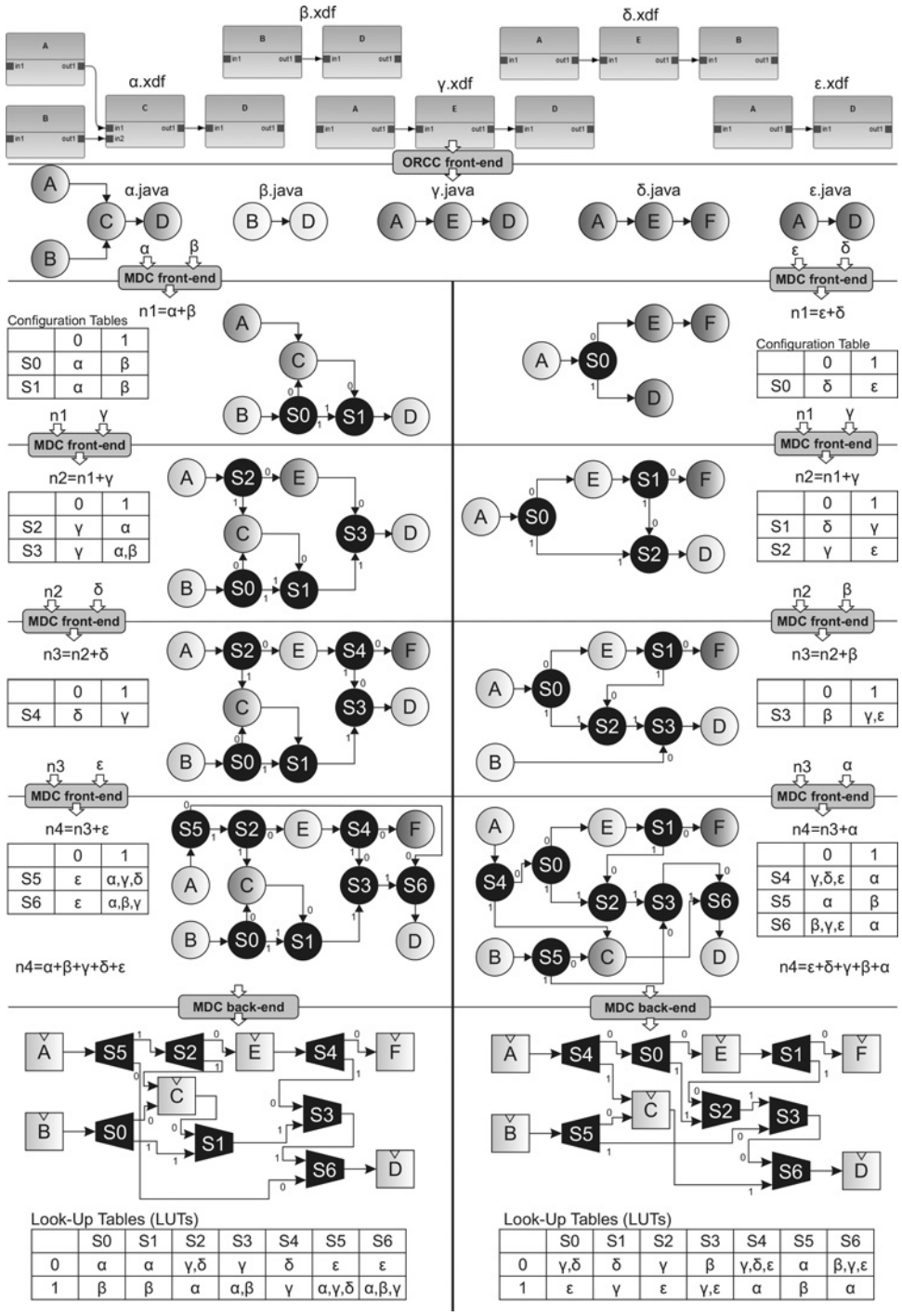

Fig. 3 *MDC tool: step-by-step example of the baseline flow*

graphs (*α.java*, *β.java*, *γ.java*, *δ.java* and *ε.java*), are processed by MDC. The MDC front-end is responsible of merging all the graphs within a unique multi-functional one and of keeping trace of the system programmability through the *Configuration Table*. As you can see in the provided example, having five different input networks implies four iterations of MDC front-end, since it merges incrementally two networks at a time. The merging process acts at the actor level; therefore, the feeding order of the given input networks may change the output specification, in terms of cascade of Sboxes to access the shared actors. The proposed example demonstrates that different orders, *α.java-β.java-γ.java-δ.java-ε.java* on the left-hand side and *ε.java-δ.java-γ.java-β.java-α.java* on the right-hand side, produce different final multi-dataflow graphs and, accordingly, different final *Configuration Tables*. At last, the MDC back-end parses the multi-functional graph and maps it onto a coarse-grained HDL top-module. To ensure functional correctness, the Sboxes are driven by dedicated look-up tables, which content is defined according to the *Configuration Table*.

2.3 Power-awareness in reconfigurable architectures

To keep hot chips cool, the more the integration on a single die the more holistic power minimisation approaches are required across the whole design stack [19]. The power-aware approach we are presenting combines

doi: 10.1049/iet-cdt.2014.0089

structural and dynamic aspects. The former are accounted by selecting power-efficient circuits at the topology level, the latter by automatically deriving different homogeneous clock areas. Leveraging on a dataflow-based design approach, the techniques we are presenting act both at a high-level and are automatically applied. Users just need to define the target device.

2.3.1 Structural power management: Within MPEG-RVC, a first tentative approach to structural power management has been targeted in [20] through the analysis of the causation trace of a Dataflow Process Network by applying then multi-clocking strategies. This technique, as far as we know, is still under refinement and has not been applied yet to reconfigurable systems. In [21], an energy estimation methodology, based on Performance Monitor Counters, has been proposed to estimate the energy consumption of RVC-CAL video codec specifications. This approach addresses software solutions, but monitor-based energy estimation methodologies are extensible to hardware-based RVC tools, such as the MDC one. Obviously monitor-based hardware estimation methodologies [22] would then require also powering down strategies, such as those in Section 3.

A large number of software frameworks and simulators have been proposed at the state of the art to provide structural power management and design space exploration [23–25]. Within MPEG-RVC, exploration techniques have been used for actors re-factoring to improve throughput. The CAL Design Suite [23] works at software executable level while YACE, exploited in [24], performs analysis at the dataflow level. Both these tools did not target coarse-grained systems or power-minimisation. Nevertheless, re-factoring techniques may always be implemented on any given input network, so that it would be possible to combine them with the proposed design flow.

In this paper, we do not address multi-parametric hardware characterisation. In different contexts, accurate analytical models would result extremely useful to improve the proposed structural level power management (Section 3.1). Such models, for example, have been derived in [26] to introduce technology-awareness in the early stage characterisation of a Network-on-Chip.

2.3.2 Dynamic power management: Power consumption in digital devices is composed of two different contributions: dynamic and static. The former is the larger amount and it is because of capacitance charging/discharging, when logic transitions occur (i.e. switching activity). The latter, because of leakage currents, is consumed while no circuit activity is present.

The most straightforward technique to limit the dynamic contribution is switching off unused units. This, as already well known at the state of the art [27, 28], can be done implementing clock gating techniques. Clock gating acts at the clock net level (responsible for more than the 40% of the dissipation [29]). Different techniques have been proposed according to the different chosen targets. For ASIC designs, AND gates can be used directly on the clock to disable it: changes in the clock enable signal are immediately captured, even though glitches may occur. In FPGA designs, the clock network cannot be modified by the insertion of any custom logic. Xilinx boards are equipped with dedicated blocks (BUFG), whose outputs can drive distinct regions of logic powering them down.

3 Coarse-grained reconfigurable systems: automated power management

In this section, we are going to present the early-stage power management strategy we have envisioned. This strategy is embedded in the MDC tool that has been extended with two different features:

– the *Topology Definer* (Section 3.1) acts at the structural topology level, defining the optimal system configuration(s) capable of minimising the implementation costs;
– the *Clock Set Definer* (Section 3.2) acts at the clock net level, identifying disjointed logic clock regions to be physically switched off when unused.

The *Topology Definer* may provide two different topology specifications as output: *TOP.f* and *TOP.p*, since the optimal multi-functional specification in the area/power domain (*TOP.p*) may not be optimal in the frequency one (*TOP.f*). The *Clock Set Definer* will deal in parallel with both, when present. The final system to be deployed will be selected then according to the given requirements.

3.1 Structural level: profiling-aware topology definition

At the base of the structural management there is the MDC combination algorithm. As explained in [7] and shown also in Fig. 3, given N input networks, MDC merges incrementally two networks at a time and the feeding order may change the output specification. In turn, different multi-functional graphs may correspond to different implementation costs, which amount can be estimated at the dataflow level [30]. Here follows the different phases of the implemented profiling-aware topology strategy:

1. Sequences extraction: The *Sequences Generator* defines all the possible network sequences the MDC tool will be fed with (Section 3.1.1).
2. Multi-dataflow specification definition: For each sequence the MDC tool extracts the multi-functional directed graphs (Section 2.2).
3. Profiling: Each graph, representing a design point, is back-annotated with synthesis information (currently extracted off line). Area, power dissipation and operating frequency are the estimated implementation costs (Section 3.1.2).
4. Pareto analysis: The Pareto-based analysis is carried out on the entire design space to determine the optimal system configuration(s), according to the selected design effort (Section 3.1.3).

Users can choose, via the MDC Graphical User Interface (GUI), enabling or not the *Topology Definer*. Without it, MDC will merge the networks as provided. Design effort can be set to find the optimal topology in the area/power domain or in the frequency one. The rest of this section details the above listed phases and concludes with a step-by-step example of the defined methodology.

3.1.1 Sequences extraction: Equation (1) determines the number and the type of multi-functional specifications that the *Sequences Generator* is able to compute. Please notice that it processes the directed graphs (DFGs) of the inputs networks, already acquired by Orcc. Defining as D the number of

doi: 10.1049/iet-cdt.2014.0089

different feeding sequences and N that of the input networks, the three terms in (1) are:

– *D_notMer* – The 'static', not merged, composition of the N networks in parallel;
– *D_Mer* – MDC shares as much resources as possible among the N given networks, by merging all the common actors. 'All-merged' solutions are provided. The number of possible sequences is equal to the number of all the possible permutations of the input networks;
– *D_partMer* – MDC will not maximise resource sharing, generating 'partially-merged' solutions. All the possible combinations that can be extracted from the set of input networks are placed in parallel with all the merged permutations of the other N–k networks, where k is the number of the objects of the current combination. Please note that a combination is intended as a selection of all or part of a set of objects, regardless to the order in which they are selected. Given A, B and C, the complete list of possible selections would be: AB, AC and BC (BA is not a combination since it contains the same object as AB).

$$D = D\_notMer + D\_Mer + D\_partMer =$$

$$D = 1 + N! + \sum_{k=1}^{N-2} C_{N,k} * (N-k)!$$

$$D = 1 + N! + \sum_{k=1}^{N-2} \frac{N!}{(N-k)! * k!} * (N-k)! = 1 + N! + \sum_{k=1}^{N-2} \frac{N!}{k!} \quad (1)$$

3.1.2 MDC profiler: The *MDC Profiler* estimates the implementation costs. Having already back-annotated the HDL components library with one value of estimated area and power consumption per functional unit, it retrieves $\forall v_i \in V$ the correspondent back-annotated values, a_i and p_i. All these values have been retrieved through *a priori* synthesis trials with the RTL Compiler of Cadence SoC Encounter using a 90 nm CMOS technology. Given as M the size of the V set, we adopt the following equations

$$\text{Area(DFG)} = \sum_{i=1}^{M} a_i \quad (2)$$

$$\text{Power(DFG)} = \sum_{i=1}^{M} p_i \quad (3)$$

to determine area and power consumption of the considered design point. The fewer are the units involved the smaller will be not only the static power consumption, but also the dynamic one (having less switching activity overall).

The operating frequency estimation is less straightforward. Different input networks feeding orders may result in different cascades of Sboxes in the final implementation. Sboxes are logically simple, not providing any type of pipelining. A cascade of Sboxes may negatively impact on the critical path (CP). The *MDC Profiler* $\forall n_i \in InN$ (being InN the set of input networks) retrieves the correspondent back-annotated CP, CP_i, and defines $CPstatic = max(CP_i)$ as the CP of the 'static' system configuration. SoC Encounter has been used to extract the CP associated to the N different input networks synthesised stand-alone using a 90 nm CMOS technology. Then it estimates the longest cascade of Sboxes (*seqSB*) of the considered design point. Given N_S as the number of Sboxes composing *seqSB* and cpS_i as the back-annotated CP associated to the ith Sbox within the HDL components library, we can assume

$$CP\_seqSB = \sum_{i=1}^{N_S} cpS_i \quad (4)$$

as the possible CP because of the cascade of Sboxes. The *MDC Profiler* finally compares *CPstatic* and *CP_seqSB* and determines the estimation of the CP of the considered design point

$$\text{CP} = max(CPstatic, CP\_seqSB) \quad (5)$$

3.1.3 Topology selection: The final stage of the *Topology Definer* involves a Pareto analysis of the design space, driven by the design effort selected by the user. For power management purposes it will be extremely important to determine the least consuming configuration, but minimising power consumption not necessarily implies having also the best operating frequency. Therefore, this final stage provides as outputs two sub-optimal multi-functional specifications, the area/power (*TOP.p*) and the frequency (*TOP.f*) one.

3.1.4 Step-by-step example: Fig. 4 provides a step-by-step example of the discussed procedure. Let us consider the same input networks adopted to describe the baseline MDC flow in Fig. 3.

The proposed structural level power management strategy receives as input the directed graphs (*α.java*, *β.java*, *γ.java*, *δ.java* and *ε.java*) of the already acquired input specifications. The *Sequences Generator*, according to (1) and given 5 input networks, extracts 321 different feeding sequences. Therefore 321 points will constitute the design space of the possible system specifications. Fig. 3 shows also how the 'static', the 'all-merged' and the 'partially-merged' multi-functional output graphs will look like.

The *MDC Profiler*, leveraging on (2), (3) and (5), determines the implementation costs of each design point. The *Topology Definer*, according to the estimated costs, is able to choose *TOP.p*, the power-optimal system specification and *TOP.f* the frequency-optimal one. *TOP.p*, in this example, corresponds to an 'all-merged' solution, where sharing is maximised and the feeding sequence is *β–γ–ε–α–δ*. *TOP.f*, instead, corresponds to a 'partially-merged' solution, where the not-merged *combination α–β* is placed in parallel with the merged *ε–γ–δ permutation*.

3.2 Dynamic level: early-stage clock set definition

The second step of the proposed power management approach enables early stage identification of disjointed logic clock regions. The *Clock Set Definer* derives, from the directed graphs of the input networks to be merged, the actors active/inactive together and groups them within homogeneous logic regions. At the hardware level, then, clock gating procedures are applied. Users are just required to specify, on the MDC GUI, the final target device, since

doi: 10.1049/iet-cdt.2014.0089

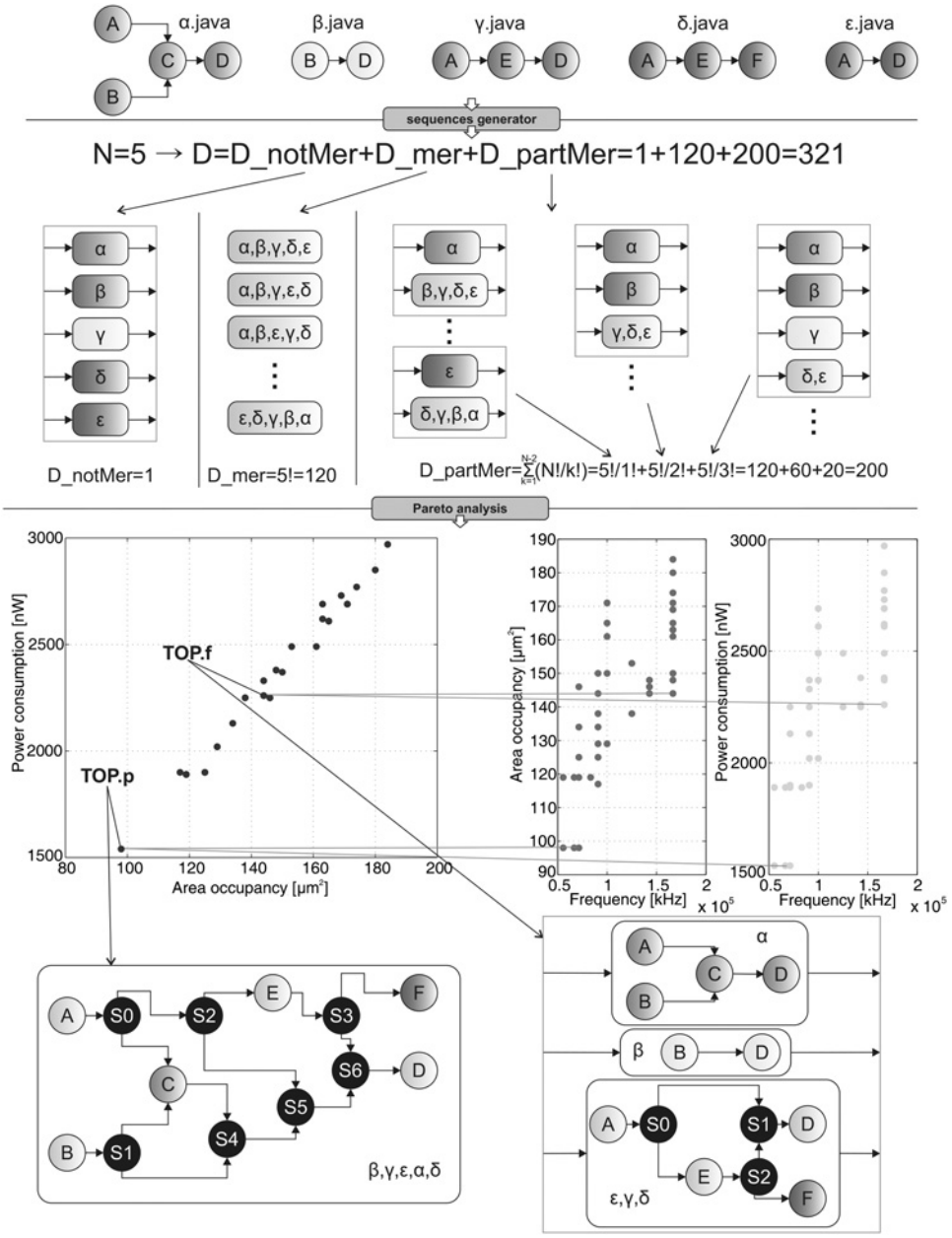

Fig. 4 *Structural level power management: a step-by-step example*

different targets require different physical solutions. At the moment, both AND gates (glitches are a false problem in the RVC domain: clock enables do not have a high switching activity) cells for ASIC designs and BUFG cells for FPGA implementations on Xilinx boards are featured.

The *Clock Set Definer* acts in two steps: (i) identification of the minimal set of logic regions and (ii) merging process of the logic regions (if necessary). The rest of this section details this process and concludes with a step-by-step example of the defined methodology.

3.2.1 Logic regions identification process: Logic regions identification is performed regardless the hardware target chosen, to minimise on-chip redundancy. Considering S as the complete set of logic regions, this step implies to minimise the number of elements, N, within S

$$\text{minimise } N, \quad S = \{S_1, S_2, \ldots, S_N\} \tag{6}$$

where S_i is the ith logic region.

The minimum N solving this problem is the number of input networks M, when they do not share any actor. When shared actors are present, the partition S, composed of M sets, constitutes the starting point to determine N. Normally, S will be composed of two different types of sets:

– S_{comp} contains those sets corresponding to a specific input network, involving non-shared actors;
– S_{shared} contains those sets corresponding to more than one network, involving shared actors.

doi: 10.1049/iet-cdt.2014.0089

Two dedicated map structures, namely the 'complementary map (*Cmap*)' and the 'intersection map (*Imap*)', are necessary to keep trace of the identified logic regions.

Given $A = \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$ as the set of the input networks, let us define the 'complementary map' as

$$Cmap = \begin{array}{c} \\ c_1 \\ c_2 \\ \vdots \\ c_j \\ \vdots \\ c_M \end{array} \begin{array}{c} \begin{array}{cccccc} \alpha_1 & \alpha_2 & \cdots & \alpha_j & \cdots & \alpha_M \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1j} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2j} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{jj} & \cdots & w_{jM} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{Mj} & \cdots & w_{MM} \end{pmatrix} \end{array}$$

where

$$c_j = \alpha_j \backslash \bigcup_{\substack{i \neq j \\ i=1}}^{M} \alpha_i$$

and

$$w_{ij} = \begin{cases} 0, & \text{if } i \neq j \\ 0, & \text{if } i = j \text{ and } c_j = \emptyset \\ 1, & \text{if } i = j \text{ and } c_j \neq \emptyset \end{cases}$$

Having $w_{jj} \neq 0$ means that: (i) the input network α_j has, at least, one actor that is not shared with the other networks; (ii) c_j is a logic region containing the non-shared actor(s) of α_j. Therefore, once *Cmap* is completely determined, S_{comp} can be derived as all the $c_j \neq \emptyset$. S_{comp} sets can be M as maximum (if all the M input networks are composed, at least, of one non-shared actor).

To define the intersection map you need to compute the set of all the non-shared actors (Γ) as

$$\Gamma = \bigcup_{i=1}^{M} c_i$$

and the complementary set of all the shared actors (Δ) as

$$\Delta = \bigcup_{i=1}^{M} \alpha_i \backslash \Gamma$$

the intersection map then is given by

$$Imap = \begin{array}{c} \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_j \\ \vdots \\ \delta_K \end{array} \begin{array}{c} \begin{array}{cccccc} \alpha_1 & \alpha_2 & \cdots & \alpha_j & \cdots & \alpha_M \end{array} \\ \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1j} & \cdots & p_{1M} \\ p_{21} & p_{22} & \cdots & p_{2j} & \cdots & p_{2M} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{j1} & p_{j2} & \cdots & p_{jj} & \cdots & p_{jM} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{K1} & p_{K2} & \cdots & p_{Kj} & \cdots & p_{KM} \end{pmatrix} \end{array}$$

where K is the cardinality of Δ, $\delta_j \in \Delta$ and

$$p_{ij} = \begin{cases} 0, & \text{if } \delta_j \cap \alpha_i = \emptyset \\ 1, & \text{if } \delta_j \cap \alpha_i \neq \emptyset \end{cases}$$

Having $p_{ij} \neq 0$ means that the shared actor δ_j belongs to the input network α_i. *Imap* may contain replicated rows: δ_x and δ_y associated both to α_m and α_n need to be grouped within a unique logic region, since they will be switched on and switched off together. To eliminate possibly replicated rows, consider *Imap* as a block matrix

$$Imap = \begin{pmatrix} \text{Imap}_1 \\ \text{Imap}_2 \\ \cdots \\ \text{Imap}_j \\ \cdots \\ \text{Imap}_K \end{pmatrix}$$

and process it as follows

$$\forall i > j \text{ if } \text{Imap}_i = \text{Imap}_j \Rightarrow \text{Imap}_i = \emptyset \text{ and } \delta_j = \delta_j \cup \delta_i \text{ and } \delta_i = \emptyset$$

At this point S_{shared} can be derived as all the $\delta_j \neq \emptyset$. S_{shared} sets can be K as maximum (if no replicated or null rows are present on *Imap*).

Summarising, S_{comp} and S_{shared} represent all the logic regions in the system, which are $M+K$ at maximum. These regions are driven accordingly to *Cmap* and *Imap* that, together, represent the *Association map* of the system, relating each logic region to the original input networks triggering it. Given the input network α_x to determine its associated logic regions you need to:

1. check on the *Cmap* which row has a 1 in the α_x column:

   – none: α_x is composed of shared actors only;
   – the qth row: α_x triggers the c_q logic region, composed of non-shared actors.

2. check on the *Imap* which row(s) has a 1 in the α_x column:

   – none: α_x is composed of non-shared actors only;
   – the qth row(s): α_x triggers the δ_q logic region(s), composed of shared actors.

3.2.2 Logic region merging process: The second phase of the *Clock Set Definer* is the logic regions merging, performed to reduce their amount. FPGAs have a limited number of BUFG cells available. Sub-optimal regions are identified: there will be switching activity also in unused functional units. Reduction can be done, according to two cost functions based on:

– Minimising the contribution of useless activity because of the number of functional units per logic region – given c_i as the cardinality of the ith logic region and P as the number

doi: 10.1049/iet-cdt.2014.0089

of networks activating it, we can define wN_i

$$wN_i = c_i * P \tag{7}$$

as the *weight* of considered region.

– Minimising the power amount because of the functional units per logic region – given p_i as the estimated power consumption [see (3)] of the ith logic region and P as the number of networks activating it, we can define wP_i

$$wP_i = p_i * P \tag{8}$$

as the *weight* of considered region.

The *Clock Set Definer*, in both cases, while merging the different logic regions will aim at keeping all the weights as minimal as possible.

3.2.3 Step-by-step example: To clarify this second part of the proposed power management methodology let us continue the example provided in Section 3.1.4, carrying out all the steps of the dynamic level process on *TOP.p*. This latter is an 'all-merged' solution, so that the *Clock Set Definer* needs to consider all the five input networks: $\alpha = \{A, B, C, D\}$, $\beta = \{B, D\}$, $\gamma = \{A, E, D\}$, $\delta = \{A, E, F\}$ and $\varepsilon = \{A, D\}$.

The *Complementary map* is the following

$$Cmap = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \begin{array}{c} \begin{array}{ccccc} \alpha & \beta & \gamma & \delta & \varepsilon \end{array} \\ \begin{pmatrix} w_{1\alpha} & 0 & 0 & 0 & 0 \\ 0 & w_{2\beta} & 0 & 0 & 0 \\ 0 & 0 & w_{3\gamma} & 0 & 0 \\ 0 & 0 & 0 & w_{4\delta} & 0 \\ 0 & 0 & 0 & 0 & w_{5\varepsilon} \end{pmatrix} \end{array}$$

where

$$\begin{aligned} c_1 &= \alpha\backslash(\beta \cup \gamma \cup \delta \cup \varepsilon) = \{A, B, C, D\}\backslash(\{B, D\} \\ &\quad \cup \{A, E, D\} \cup \{A, E, F\} \\ &\quad \cup \{A, D\}) = \{A, B, C, D\}\backslash\{B, D, A, E, F\} \\ &= \{C\} \Rightarrow w_{1\alpha} = 1 \\ c_2 &= \beta\backslash(\alpha \cup \gamma \cup \delta \cup \varepsilon) = \{B, D\}\backslash(\{A, B, C, D\} \\ &\quad \cup \{A, E, D\} \cup \{A, E, F\} \\ &\quad \cup \{A, D\}) = \{B, D\}\backslash\{A, B, C, D, E, F\} \\ &= \{\emptyset\} \Rightarrow w_{2\beta} = 0 \\ c_3 &= \gamma\backslash(\alpha \cup \beta \cup \delta \cup \varepsilon) = \{\emptyset\} \Rightarrow w_{3\gamma} = 0 \\ c_4 &= \delta\backslash(\alpha \cup \beta \cup \gamma \cup \varepsilon) = \{F\} \Rightarrow w_{4\delta} = 1 \\ c_5 &= \varepsilon\backslash(\alpha \cup \beta \cup \gamma \cup \delta) = \{\emptyset\} \Rightarrow w_{5\varepsilon} = 0 \end{aligned}$$

According to these values, the *Complementary map* is

$$Cmap = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \begin{array}{c} \begin{array}{ccccc} \alpha & \beta & \gamma & \delta & \varepsilon \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

and $S_{\text{comp}} = \{c_1, c_4\}$, with $c_1 = \{C\}$ and $c_4 = \{F\}$.

The set of all the non-shared actors, Γ, is

$$\Gamma = \bigcup_{i=1}^{M} c_i = \{C\} \cup \{F\} = \{C, F\}$$

and its complementary Δ is

$$\Delta = (\alpha \cup \beta \cup \gamma \cup \delta \cup \varepsilon)\backslash\Gamma = \{A, B, D, E\}$$

The *Intersection map* is given by

$$Imap = \begin{array}{c} \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{array} \begin{array}{c} \begin{array}{ccccc} \alpha & \beta & \gamma & \delta & \varepsilon \end{array} \\ \begin{pmatrix} p_{1\alpha} & p_{1\beta} & p_{1\gamma} & p_{1\delta} & p_{1\varepsilon} \\ p_{2\alpha} & p_{2\beta} & p_{2\gamma} & p_{2\delta} & p_{2\varepsilon} \\ p_{3\alpha} & p_{3\beta} & p_{3\gamma} & p_{3\delta} & p_{3\varepsilon} \\ p_{4\alpha} & p_{4\beta} & p_{4\gamma} & p_{4\delta} & p_{4\varepsilon} \end{pmatrix} \end{array}$$

where

$$\begin{aligned} \delta_1 &= \{A\} \\ &= \begin{cases} \delta_1 \cap \alpha = \{A\} \cap \{A, B, C, D\} \neq \emptyset \Rightarrow p_{1\alpha} = 1 \\ \delta_1 \cap \beta = \{A\} \cap \{B, D\} = \emptyset \Rightarrow p_{1\beta} = 0 \\ \delta_1 \cap \gamma = \{A\} \cap \{A, E, D\} \neq \emptyset \Rightarrow p_{1\gamma} = 1 \\ \delta_1 \cap \delta = \{A\} \cap \{A, E, F\} \neq \emptyset \Rightarrow p_{1\delta} = 1 \\ \delta_1 \cap \varepsilon = \{A\} \cap \{A, D\} \neq \emptyset \Rightarrow p_{1\varepsilon} = 1 \end{cases} \end{aligned}$$

$$\delta_2 = \{B\} = \begin{cases} \delta_2 \cap \alpha \neq \emptyset \Rightarrow p_{2\alpha} = 1 \\ \delta_2 \cap \beta \neq \emptyset \Rightarrow p_{2\beta} = 1 \\ \delta_2 \cap \gamma = \emptyset \Rightarrow p_{2\gamma} = 0 \\ \delta_2 \cap \delta = \emptyset \Rightarrow p_{2\delta} = 0 \\ \delta_2 \cap \varepsilon = \emptyset \Rightarrow p_{2\varepsilon} = 0 \end{cases}$$

$$\delta_3 = \{D\} = \begin{cases} \delta_3 \cap \alpha \neq \emptyset \Rightarrow p_{3\alpha} = 1 \\ \delta_3 \cap \beta \neq \emptyset \Rightarrow p_{3\beta} = 1 \\ \delta_3 \cap \gamma \neq \emptyset \Rightarrow p_{3\gamma} = 1 \\ \delta_3 \cap \delta = \emptyset \Rightarrow p_{3\delta} = 0 \\ \delta_3 \cap \varepsilon \neq \emptyset \Rightarrow p_{3\varepsilon} = 1 \end{cases}$$

$$\delta_4 = \{E\} = \begin{cases} \delta_4 \cap \alpha = \emptyset \Rightarrow p_{4\alpha} = 0 \\ \delta_4 \cap \beta = \emptyset \Rightarrow p_{4\beta} = 0 \\ \delta_4 \cap \gamma \neq \emptyset \Rightarrow p_{4\gamma} = 1 \\ \delta_4 \cap \delta \neq \emptyset \Rightarrow p_{4\delta} = 1 \\ \delta_4 \cap \varepsilon = \emptyset \Rightarrow p_{4\varepsilon} = 0 \end{cases}$$

According to these values, the *Intersection map* is

$$Imap = \begin{array}{c} \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{array} \begin{array}{c} \begin{array}{ccccc} \alpha & \beta & \gamma & \delta & \varepsilon \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

There are no replicated rows, so that $S_{\text{shared}} = \{\delta_1, \delta_2, \delta_3, \delta_4\}$.

Fig. 5 depicts the identified six logic regions and their respective *Association map*, as formally derived above. Generally speaking, having so few logic regions their merging would not be required. Nevertheless, for the sake of completeness, let us assume the need of reducing their amount from 6 to 4. In this example, to maximise power reduction, we adopted (8): *set1* and *set3* present the smallest *wP* values so that they are the perfect candidates for merging. In the *Association map set1* and *set3* are

doi: 10.1049/iet-cdt.2014.0089

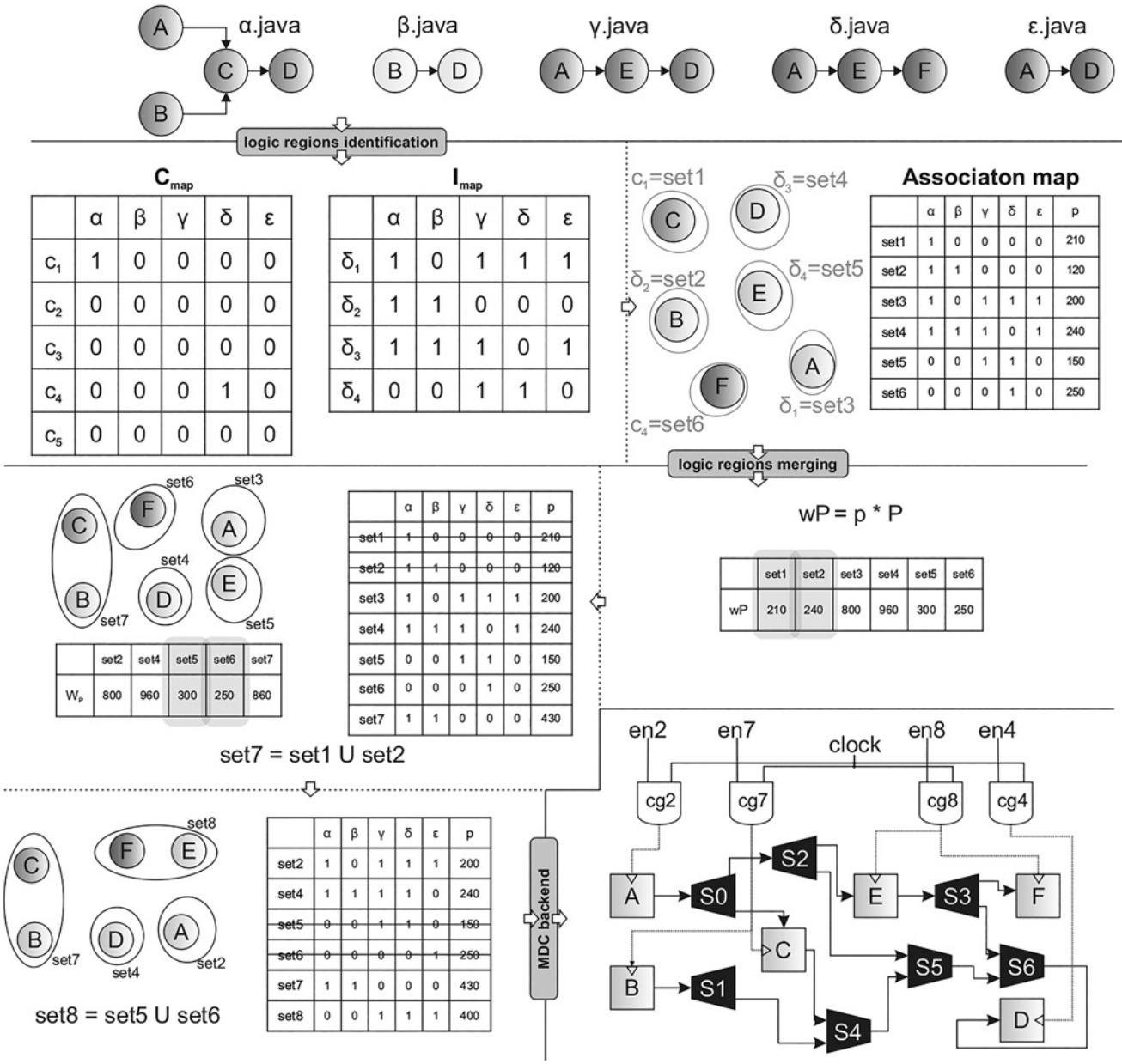

Fig. 5 *Dynamic level power management: a step-by-step example*

removed and *set7 = set1 U set3* is inserted. Exploiting again (8), it is possible to identify the next two candidates: *set5* and *set6* are removed and *set8 = set5 U set6* is defined.

4 Experimental results

To validate the proposed power-aware design flow we have adopted MDC to assemble the computing core of an accelerator for an image zoom application. The application has been profiled to identify the most computationally intensive code segments (*computational kernels*). These kernels have been modelled as RVC-CAL dataflow specifications, composing the set of networks to be passed to MDC. The *Topology Definer* retrieves the optimal structural configurations, *TOP.f* and *TOP.p*, of the accelerator computing core. These two are then exploited by the *Clock Set Definer* to assemble their respective clock-gated and non-clock-gated versions of the multi-dataflow platforms. Results are going to be discussed first of all on the ASIC implementation and then, for verification purposes, also the FPGA prototype has been assembled.

4.1 Use case generalities

The image zoom application aims to scale an image by a given zooming factor. The zoomed image pixels are interlaced with other pixels obtained adaptively from the interpolations of the adjacent original pixels.

Starting from the C code of the zoom, we have performed a software profiling to identify the kernels to be accelerated with dedicated hardware blocks. Seven kernels have been isolated and modelled as dataflow networks, as summarised in Table 1. The identified kernels differ for their functionality and for the size of the computed data, which determine their composition in terms of number of involved actors. Table 1 shows also the number of occurrences of each kernel while processing the zoom of a 128×128 pixels image.

doi: 10.1049/iet-cdt.2014.0089

Table 1 Computational kernels of the zoom application

Kernel	No. of actors	No. of occ	Data size	Functionality
abs	1	3150	1	absolute value calculation
min_max	1	1050	2	maximum/minimum finding
chgb	7	3072	16 × 16	bilevel/grey-scale block checking
median	9	1069	4	median calculation
cubic_conv	6	408	16 × 16	cubic filter convolution
cubic	10	1070	4	linear combination calculation
sbwlabel	17	2722	16 × 16	edge block checking

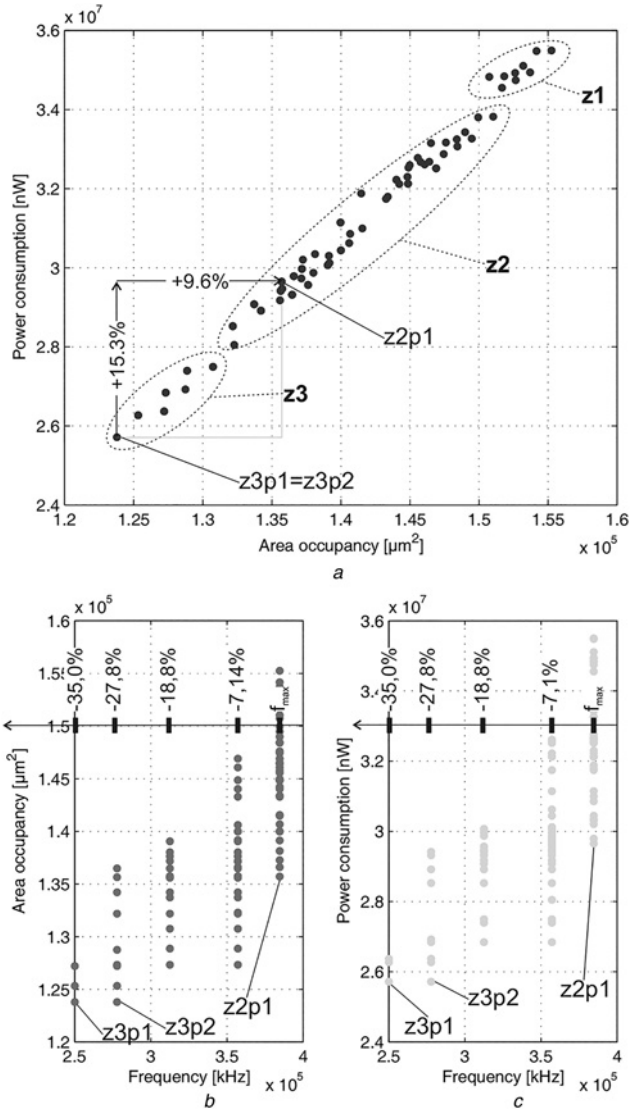

Fig. 6 *Structural level power management methodology: Pareto analysis for*

a Area against power
b Area against frequency
c Power against frequency

Table 2 Composition for the sub-optimal platforms

	TOP.f	*TOP.p*
total actors	78	86
sbox actors (%)[a]	37 (47)	53 (62)
other actors (%)[a]	41 (52)	33 (38)
shared actors (%)[b]	10 (20)	18 (35)

[a] Calculated with respect to the total number of actors for the optimal solution.
[b] Calculated with respect to the total number of actors of the input nets.

4.2 Structural power management discussion

The seven identified networks, to be implemented by the MDC on a coarse-grained datapath, required the evaluation of a design space of 13 693 points [see (1)]. Two possible colliding design efforts have to be considered (frequency against area occupation/power consumption) and the *Topology Definer* explores all the possible design points to determine the optimal structural solutions.

Fig. 6*a* depicts the distribution of the design points in terms of area against power. The trend is linear, since these two metrics are estimated in the same manner (Section 3.1.2). It is possible to identify three different clusters in the graph: *z1* involving points with 0–4 merged networks (the rest are placed in parallel), *z2* with 2–6 merged networks and *z3* with 5–7 merged networks. According with the area/power graph, the optimal design points are *z3p1* and *z3p2*, where all the seven input networks are merged. Looking at the frequency domain (shown in Figs. 6*b* and *c*), with respect to the maximum achievable frequency, *z3p1* looses nearly the 35.0% and *z3p2* the 27.8%. Therefore, the *Topology Definer* selects *z3p2* as *TOP.p*, presenting the best area/power value at the smaller frequency price.

In the frequency domain, five clusters are visible in Figs. 6*b* and 6*c*. The optimal configuration in this domain is *z2p1* where all the networks are merged but the *chgb* and the *cubic_conv* ones. With respect to *TOP.p*, in the area/power domain, *z2p1* presents an overhead of 9.6% in area and 15.3% in power. For this use case, it is not possible to identify the perfect optimal solution for all the considered implementation costs. The *Topology Definer* selects *z2p1* as *TOP.f*: the configuration with the highest frequency at the smallest area/power penalty.

Table 2 highlights the composition of *TOP.f* and *TOP.p*. Their seven input networks are composed of 51 actors overall. Merging them implies adding Sboxes. Therefore, *TOP.f* integrates 78 actors that are 8 less than *TOP.p*, but the shared actors (not considering the Sboxes) are 10 against 18. Less overlapping means less Sboxes adoption. Keeping the *chgb* and the *cubic_conv* kernel in parallel saves 16 Sboxes, but requires replicating 8 actors determining the 15.3% of additional estimated power consumption.

4.2.1 Remarks: Dealing with design space exploration problems you need to consider that: full search approaches, where all the design points within the design space are explored and characterised in terms of objective functions, are generally unfeasible. Design space cardinality, and consequently the exploration time, may explode. The explorations proposed in this paper required to evaluate 13 693 points, which is still feasible since it took a couple of minutes. Nevertheless, as soon as the number of input

doi: 10.1049/iet-cdt.2014.0089

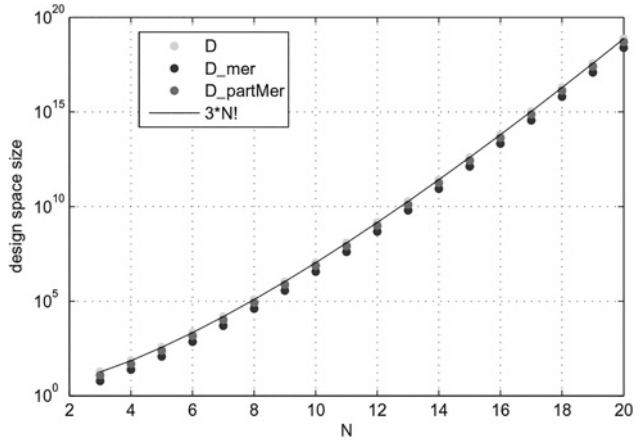

Fig. 7 *Design space size trend with respect to the number of input networks*

networks grows, that may not be doable anymore. Fig. 7 shows the 3**N!* relationship that holds between the design space size and the number of input networks *N*.

Considering our problem, the exploration of the design space to retrieve *TOP.f* and *TOP.p*, these considerations are valid:

– *TOP.p* normally is an 'all-merged' solution. By construction, the smallest is the number of actors, the smallest would be the values retrieved by (2) and (3).
– *TOP.f* may be a 'partially-merged' solution. The fewer networks you merge, the shorter should be the worst critical path.

Therefore, it would possible to exploit these assumptions to analyse just those parts of the design space relevant for the given problem, basing on the effort set by the user. Still, on the considered portions, you will end up doing an exhaustive exploration.

To avoid exhaustive explorations, heuristic algorithms can be successfully used to reduce the overall exploration time by computing an approximated Pareto set of configurations, as in [31, 32]. In our case, we may use similar approaches to reduce the cardinality of our design space. Moreover, automated strategies based on high-level information can be proposed. In particular, with respect to the exploration of the 'partially-merged' part of the design space, two meaningful metrics could be considered:

– Probability of each actor to impact on the system operating frequency, which is proportional to the number of networks accessing it. If the network(s) containing the actor(s) with the highest probability are placed in parallel, rather than merged, higher operating frequency values are achievable, since smaller Sbox chains will be present. A smart selection of the *combinations* in the third term of (1) can be implemented.
– Power overhead introduced when a network is placed in parallel, rather than merged.

Table 3 Area occupancy on ASIC for the implemented designs

	freq_nocg	*freq_cg*	*pwr_nocg*	*pwr_cg*
area, μm^2	135 819	136 076	124 026	124 579

Table 4 Power consumption on ASIC for the implemented designs

	freq_nocg	*freq_cg*	*pwr_nocg*	*pwr_cg*
static power, mW	0.072	0.072	0,066	0,067
dynamic power, mW	32 569	1002	27 887	1885
total power, mW	32 641	1073	27 953	1952

With respect to the exploration of the 'all-merged' part of the design space, instead, statistical studies may lead to the definition of a proper heuristic method for the exploration of the design points that have different frequencies, at the minimum area and power values.

As a conclusion, clearly the design space size explosion, in some cases, may be an issue. Different properties/characteristics/metrics of the design space can be exploited to define a robust and scalable heuristic algorithm performing approximated Pareto analysis. The actual implementation of those heuristics will constitute one of the future improvements of the proposed work.

4.3 Dynamic power management discussion

To evaluate the effectiveness of the proposed dynamic power management strategy, we are going to show the results related to four different assembled platforms:

– *freq_nocg*: *TOP.f* without logic regions identification;
– *freq_cg*: *TOP.f* with logic regions identification;
– *pwr_nocg*: *TOP.p* without logic regions identification;
– *pwr_cg*: *TOP.p* with logic regions identification.

The *Clock Set Definer* extracts different logic regions for *TOP.f* and *TOP.p*, since they have different compositions. In particular, 9 regions are identified for *TOP.f* and 13 for *TOP.p*. On ASIC their merging process is not necessary. We have synthesised the above-listed designs using Cadence SoC Encounter on the same technology and at the same frequency used to carry out the structural power management process.

Table 3 shows the synthesis results in terms of area occupancy. *freq_nocg*, as expected, occupies more area than *pwr_nocg*. The percentage of saving adopting the latter is about 8.7%. Table 3 shows also clock-gating overhead: 0.15% (*freq_nocg* against *freq_cg*) and 0.45% (*pwr_nocg* against *pwr_cg*). The estimation error among the real synthesis values and those achieved with (2) is, on average, 0.14%.

Table 4 shows the synthesis results in terms of power consumption. The power consumption of *pwr_nocg*, as expected, is 14.3% lower than the *freq_nocg* one. The estimation error among the real synthesis values of *TOP.f* and *TOP.p* and those achieved with (3) is, on average, 9.8%.

When clock gating techniques are applied a considerable power saving seems achievable. Nevertheless, the estimations provided by the synthesiser are not realistic, not taking into account real execution conditions (default

Table 5 Power consumption on ASIC for the implemented designs during the execution of the zoom application

	freq_nocg	*freq_cg*	*pwr_nocg*	*pwr_cg*
total power, mW	6908	1737	6036	1732

doi: 10.1049/iet-cdt.2014.0089

switching rates are adopted, the same holds for the back-annotated values used by the MDC profiler). Consequently, you can notice as in Table 4 *pwr_cg* seems dissipating more than *freq_cg*. Therefore, to achieve more accurate estimations, we have retrieved the switching activity executing each computational kernel stand-alone on the four assembled platforms and we have used it to compute the values provided in Table 5. These more realistic values for power consumption have been estimated as the sum of the single kernel dissipation values weighted for their occurrences within the zoom execution (see Table 1). They are considerably different with respect to the estimations in Table 4, especially for the designs without clock gating (switching off several units at the same time clock gated platforms are less affected, on average, by default switching rates). The *pwr_nocg* design still allows saving the 12.6% of power with respect to the *freq_nocg* one. When the real switching activity is considered, you can notice that the *pwr_cg* and the *freq_cg* values are quite similar. This is due to the fact that the identified logic regions, in both designs, are optimal either in the area/power domain either in the frequency one. Therefore, in both cases, when clock gating is applied just the minimum set of functional units to guarantee computing correctness is turned on.

To conclude, dynamic power management leads to save, respectively, the 74.9 and the 71.3% of power in the *TOP.f* and the *TOP.p* designs. This large saving amount is determined by the fact that we are dealing with multi-functional designs, where several functional units may be unused while executing a specific computation: for example, when *abs* or *min_max* kernels are executed (see Table 1) only one actor is active on 41 (*TOP.f*) or 33 (*TOP.p*).

4.4 FPGA prototype

This section is meant to discuss a prototype implementation of a processor-coprocessor system on FPGA to accelerate the zoom application, adopting the proposed power-aware design flow. MDC assembles the coarse-grained reconfigurable computing core of the accelerator that embeds also:

– a local memory, to store given inputs and processed results,
– a finite state machine, to manage data transfers to/from memory,
– a bus interface, to communicate with the host processor.

In this paper, the described acceleration unit has been coupled to a MicroBlaze soft-core via a PLB system bus on a Xilinx Virtex 5 FPGA board. Execution trials demonstrated a speedup of 1.84 when the coprocessor is adopted during the zoom of a 128 × 128 pixels image.

Table 6 Resource occupancy on FPGA for the implemented designs

Resource (available)	*freq_nocg*	*freq_cg*	*pwr_nocg*	*pwr_cg*
Slices (7200)	1565 (22)	1629 (23)	1387 (19)	1519 (21)
Slice Regs (28 800)	353 (12)	3556 (12)	3092 (11)	3119 (11)
Slice LUTs (28 800)	5031 (17)	5092 (18)	4574 (16)	4661 (16)
BRAMs (60)	3 (5)	3 (5)	1 (2)	1 (2)
DSPs (48)	7 (15)	7 (15)	7 (15)	7 (15)
BUFGs (32)	2 (6)	9 (28)	2 (6)	13 (41)

Table 7 Power consumption estimation, per resource, on FPGA for the implemented designs

Resource	*freq_nocg*	*freq_cg*	*pwr_nocg*	*pwr_cg*
Logic, mW	5.1	4.8	4.9	3.9
Signal, mW	8.1	9.1	8.8	7.9
BRAMs, mW	15.2	2.9	4.9	2.4
DSPs, mW	0.5	0.5	0.5	0.5
TOT, mW	28.9	17.3	19.0	14.7

We have prototyped all the four previously discussed designs. Table 6 depicts the resource utilisation on the targeted FPGA, extracted by the Xilinx Synthesis Technology (XST) tool. It is possible to appreciate more or less the same trend of the ASIC area occupancy: the *pwr_nocg* occupies a bit less slices (about the 11%) than the *freq_nocg*. In the latter case there are also two additional BRAMs occupied: actors duplication involves memory blocks. When clock gating is applied there is a slices overhead correspondent to 4% (*freq_nocg* against *freq_cg*) and 10% (*pwr_nocg* against *pwr_cg*). This overhead is shown also by the used BUFGs: 9 for *TOP.f* and 13 for *TOP.p*. Not reaching the 32 available BUFGs, logic regions merging is not applied.

To analyse the power consumption we have adopted the Xilinx Power Analyser (XPA) tool. Estimations are provided in Table 7 and have been computed as in the ASIC case: each computational kernel has been executed stand-alone to record its switching activity and then the dissipated power has been estimated as the sum of the kernels dissipation values weighted for their occurrences within the zoom execution. This power estimation has been obtained with an operating frequency of 100 MHz (the maximum one of the considered designs).

On FPGA, *pwr_nocg* saves the 34% of power with respect to *freq_nocg*, still validating the effectiveness of the topology optimisation. The system composition is fundamental: in the *TOP.f* design, three BRAMs are required because of actors duplication, but they are used by different kernels. So that in the *freq_cg* design they are mutually switched off, saving nearly the 81% of the BRAMs related power, a significant portion of the overall consumption.

Clock gating is effective for both the designs, with a resulting saving of 40% for the *TOP.f* design and of 23% for the *TOP.p* one, but mapping has a key role. Clock gating not only implies additional resources overhead, but forces the utilisation of the BUFGs that have a fixed position. This may cause a larger spreading of the design on the board, negatively affecting power dissipation because of data transmissions on longer links.

To conclude the proposed design flow can be adopted also for FPGA targets, but additional constraints and more accurate models (e.g. [22] or [26]) should be adopted to improve the achievable results.

5 Conclusions

In this paper, we addressed the problem of providing high-level power management while implementing coarse-grained platforms running different applications on shared resources. Heterogeneous runtime reconfigurable systems are deployed.

doi: 10.1049/iet-cdt.2014.0089

Two are the basic assumptions of this work: (i) power reduction is of paramount importance in modern battery dependent designs; and (ii) systems heterogeneity and complexity is so challenging that manual strategies require too much effort to be considered still viable.

The proposed methodologies have been used to extend a dataflow-based design framework for coarse-grained reconfigurable designs, the MDC tool. MDC has been conceived within MPEG-RVC research studies [5, 6], but it has already been adopted to characterise systems in different signal processing contexts [7–9]. The importance of the proposed power-aware extension of the MDC tool is mainly related to the fact that, within the MPEG-RVC framework, power consumption is still an open issue and, as far as we know, early-stage power estimation/management strategies are currently under scrutiny. Our solution combines structural and dynamic strategies applied automatically at the modelling level. We have demonstrated how topology may affect power in coarse-grained architectures and that it is possible to adopt a graph-based strategy to lower also dynamic power consumption.

Results on ASIC confirmed the expectations both at the structural and at the dynamic level. With respect to the presented zoom application, the topology exploration allowed to define two sub-optimal solutions (in the frequency and in the area/power domains) within a design space of more than 13 thousand points. The dynamic power optimisation applied to those designs led to save approximately the 70% of the consumed power. On the FPGA prototypes, it is still possible to appreciate the benefits of the proposed methodology, although dedicated models would be required.

6 Acknowledgments

Prof. Luigi Raffo and Dr. Carlo Sau are grateful to Sardinia Regional Government for funding the RPCT Project (L.R. 7/2007, CRP-18324) that led to these results. Dr. Sau is also grateful to Sardinia Regional Government for supporting his PhD scholarship (P.O.R. F.S.E., European Social Fund 2007-2013 – Axis IV Human Resources).

7 References

1 Bhattacharyya, S.S., Eker, J., Janneck, J.W., Lucarz, C., Mattarelli, M., Raulet, M.: 'Overview of the mpeg reconfigurable video coding framework', *Signal Process. Syst.*, 2011, **63**, (2), pp. 251–263
2 Open RVC-CAL Compiler. (web) http://www.orcc.sourceforge.net/
3 Endri Bezati, J.J., Mattavelli, M.: 'High-level synthesis of dataflow programs for signal processing systems'. Symp. on Image and Signal Processing and Analysis, 2013
4 Brunet, S.C., Mattavelli, M., Janneck, J.W.: 'Turnus: A design exploration framework for dataflow system design'. Symp. on Circuits and Systems, 2013
5 Palumbo, F., Carta, N., Raffo, L.: 'The multi-dataflow composer tool: A runtime reconfigurable hdl platform composer'. Conf. on Design and Architectures for Signal and Image Processing, 2011
6 Sau, C., Raffo, L., Palumbo, F., *et al.*: 'Design flow of a rvc-cal multi-standard decoder implementation'. Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation, 2014, (to appear)
7 Palumbo, F., Carta, N., Pani, D., Meloni, P., Raffo, L.: 'The multi-dataflow composer tool: generation of on-the-fly reconfigurable platforms', *J. Real-Time Image Process.*, 2014, **9**, (1), pp. 233–249
8 Carta, N., Sau, C., Pani, D., Palumbo, F., Raffo, L.: 'A coarse-grained reconfigurable approach for low-power spike sorting architectures'. IEEE/EMBS Conf. on Neural Engineering, 2013
9 Carta, N., Sau, C., Palumbo, F., Pani, D., Raffo, L.: 'A coarse-grained reconfigurable wavelet denoiser exploiting the multi-dataflow composer tool'. Conf. on Design and Architectures for Signal and Image Processing, 2013
10 Dennis, J.B.: 'First version of a data flow procedure language'. Programming Symp., 1974
11 Kahn, G.: 'The semantics of a simple language for parallel programming', in Rosenfeld, J.L. (Ed.): 'Information processing' (North Holland, Amsterdam, 1974), pp. 471–475
12 Lee, E.A., Messerschmitt, D.G.: 'Static scheduling of synchronous data flow programs for digital signal processing', *IEEE Trans. Comput.*, 1987, **36**, (1), pp. 24–35
13 Lee, E., Parks, T.: 'Dataflow process networks', *Proc. IEEE*, 1995, **83**, (5), pp. 773–801
14 Eker, J., Janneck, J.W.: 'Cal language report specification of the cal actor language'. Technology Report, EECS Department, University of California, Berkeley, 2003
15 Carta, S.M., Pani, D., Raffo, L.: 'Reconfigurable coprocessor for multimedia application domain', *J. VLSI Signal Process. Syst.*, 2006, **44**, pp. 135–152
16 Kumar, V.V., Lach, J.: 'Highly flexible multimode digital signal processing systems using adaptable components and controllers', *EURASIP J. Appl. Signal Process.*, 2006
17 Palumbo, F., Pani, D., Manca, E., *et al.*: 'Rvc: A multi-decoder cal composer tool'. Conf. on Design and Architectures for Signal and Image Processing, 2010
18 Wipliez, M., Siret, N., Carta, N., Palumbo, F., Raffo, L.: 'Design ip faster: Introducing the c high-level language'. IP-SOC Conf. & Exhibition, 2012
19 Puri, R., Stok, L., Bhattacharya, S.: 'Keeping hot chips cool'. Design Automation Conf., 2005
20 Casale Brunet, S., Bezati, E., Alberti, C., *et al.*: 'Partitioning and optimization of high level stream applications for multi clock domain architectures'. Workshop on Signal Processing Systems, 2013
21 Ren, R., Wei, J., Martnez, E.J., *et al.*: 'A pmc-driven methodology for energy estimation in rvc-cal video codec specifications', *J. Image Commun.*, 2013, **28**, (10), pp. 1303–1314
22 Schmidt, A.G., Steiner, N., French, M., Sass, R.: 'Hwpmi: an extensible performance monitoring infrastructure for improving hardware design and productivity on fpgas', *J. Reconfigurable Comput.*, 2012
23 Lucarz, C., Roquier, G., Mattavelli, M.: 'High level design space exploration of rvc codec specifications for multi-core heterogeneous platforms'. Conf. on Design and Architectures for Signal and Image Processing, 2010
24 Rahman, A.A.-H.A., Thavot, R., Brunet, S.C., Bezati, E., Mattavelli, M.: 'Design space exploration strategies for fpga implementation of signal processing systems using cal dataflow program'. Conf. on Design and Architectures for Signal and Image Processing, 2012
25 Meloni, P., Pomata, S., Tuveri, G., *et al.*: 'Enabling fast asip design space exploration: An fpga-based runtime reconfigurable prototyper'. VLSI Design, 2012
26 Meloni, P., Loi, I., Angiolini, F., *et al.*: 'Area and power modeling for networks-on-chip with layout awareness'. VLSI DESIGN, 2007
27 Benini, L., Micheli, G.d.: 'Dynamic power management: design techniques and CAD Tools' (Kluwer Academic Publishers, 1998)
28 Pedram, M.: 'Power aware design methodologies' (Kluwer Academic Publishers, 2002)
29 Zhang, Y., Roivainen, J., Mammela, A.: 'Clock-Gating in FPGAs: a novel and comparative evaluation'. Conf. on Digital System Design: Architectures, Methods and Tools, 2006
30 Palumbo, F., Sau, C., Raffo, L.: 'Dse and profiling of multi-context coarse-grained reconfigurable systems'. Symp. on Image and Signal Processing and Analysis, 2013
31 Palermo, G., Silvano, C., Zaccaria, V.: 'Multi-objective design space exploration of embedded systems', *J. Embed. Comput., - Low-Power Embed. Syst.*, 2005, **1**, (3), pp. 305–316
32 Palesi, M., Givargis, T.: 'Multi-objective design space exploration using genetic algorithms'. Symp. on Hardware/Software Co-Design, 2002

doi: 10.1049/iet-cdt.2014.0089