Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2016, Article ID 4237350, 27 pages
http://dx.doi.org/10.1155/2016/4237350

Research Article

Modelling and Automated Implementation of Optimal Power Saving Strategies in Coarse-Grained Reconfigurable Architectures

Francesca Palumbo,[1] Tiziana Fanni,[2] Carlo Sau,[2] Paolo Meloni,[2] and Luigi Raffo[2]

[1]*POLCOMING, Information Engineering Unit, University of Sassari, Sassari, Italy*
[2]*Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, Cagliari, Italy*

Correspondence should be addressed to Tiziana Fanni; tiziana.fanni@diee.unica.it

Received 18 April 2016; Accepted 14 September 2016

Academic Editor: Wen B. Jone

This paper focuses on how to efficiently reduce power consumption in coarse-grained reconfigurable designs, to allow their effective adoption in heterogeneous architectures supporting and accelerating complex and highly variable multifunctional applications. We propose a design flow for this kind of architectures that, besides their automatic customization, is also capable of determining their optimal power management support. Power and clock gating implementation costs are estimated in advance, before their physical implementation, on the basis of the functional, technological, and architectural parameters of the baseline design. Experimental results, on 90 and 45 nm CMOS technologies, demonstrate that the proposed approach guides the designer towards optimal implementation.

1. Introduction

Electronic devices on the market rely on the execution of computation-intensive applications on complex heterogeneous systems. Coarse-grained reconfigurable (CGR) platforms combine the high performance levels provided by Application Specific Integrated Circuit (ASIC) designs with an increased flexibility, allowing the execution of a larger set of applications over the same substrate [1, 2]. However, in the dark silicon era, due to the limited available power budget, a gap exists between the number of transistors that can be placed within a die and the number that can be actually active during execution [3, 4]. Therefore, systems are also required to be energy efficient and CGR designs must integrate specific power management techniques of the functional logic regions constituting them.

Several effective techniques for power monitoring [5] and reducing [6] have been presented at the state of the art. Among them, voltage/frequency scaling [7, 8] and power shut-off schemes [9, 10] can be extremely beneficial. However, their integration requires manual intervention of the designer, resulting in a complex, error prone, and time consuming process. While commercial synthesizers [11] allow the automatic implementation of low overhead saving strategies at the gate level, such as fine-grained clock gating, they only provide implementation-level instruments to apply more complex strategies, like power gating. In the CGR systems field, the Multi-Dataflow Composer tool (MDC), combining the dataflow-based system specification approach with the coarse-grained reconfigurable design paradigm, is capable of automatically generating run-time reconfigurable multifunctional systems, featuring flexibility and area minimization [12]. MDC was originally meant to address reconfigurable codec implementations and was conceived to be exploited within MPEG Reconfigurable Video Coding (MPEG-RVC) studies. However, it was successfully adopted also in different resource and power-constrained scenarios [13, 14], where only microprogrammed solutions have been used so far, either exploiting single-core digital signal processors [15] or custom multicore embedded processors [16]. MDC design suite is composed of different extensions. The work presented

in this paper is related to MDC power management extension, which has been previously addressed in [17, 18]. MDC tool identifies in the generated CGR system the minimum set of disjointed functionally homogeneous logic areas of the system, called logic regions. These latter are exploited to automatically implement dynamic power management strategies, applying indistinctly to all of them either clock [17] or power gating methodologies [18].

The work we are presenting in this paper intends to propose a power modelling methodology and to improve the MDC power management extension by integrating such methodology within its automated flow. We introduce in this paper an algorithm that analyses the identified logic regions and, on the basis of one single synthesis and a minimal set of simulations (one for each scenario of the multifunctional problem), is capable of optimally characterizing the power management support. This flow, in a separate manner for each logic region of the CGR design, is capable of assessing both clock and power gating management costs and of determining which is the optimal power saving strategy (if any) prior to any physical system implementation. The algorithm is based on detailed static and dynamic power consumption models that take into account functional, architectural, and technological parameters to define the potential overhead and benefits of the considered solutions. As a future perspective, besides its application within the MPEG-RVC scenario, the proposed modelling strategy may also be extended to support other complex autonomous computing systems [19], where the number of involved resources may change at runtime.

The rest of this paper is organized as follows. Section 2 reports the background of this work. Section 3 describes the operation of MDC: in particular, Section 3.1 focuses on the base operation, while Section 3.2 details the proposed power estimation models and their integration within the tool. Section 4 presents the designs under test used to validate the proposed approach: Section 4.1 involves a FFT use case targeting a 90 nm CMOS technology, while Section 4.2 presents the experimental results conducted to assess the enhanced MDC flow on a zoom coprocessor (targeting both 90 nm and 45 nm CMOS technologies). Finally, Section 4.4 details the benefits of the proposed models, before concluding with some final remarks in Section 5.

2. Background

This work presents a model of power consumption for CGR systems. The model is capable of estimating the system power dissipation since the early stages of the design flow and it has been integrated within an automated flow that decides which power saving technique, between clock gating and power gating, has to be applied to each portion of the design.

This section provides an overview of the state of the art in the field of power-aware optimization and, in more detail, on the main aspects involved in the proposed approach. Section 2.1 introduces the distinctive features of CGR systems and the main architectural trends for such a kind of devices. Section 2.2 deals with power issues in digital system, with a particular emphasis on modelling strategies and design automation.

2.1. Coarse-Grained Reconfigurable Systems. Reconfigurable architectures are usually conceived as collections of functional units (FUs) whose functionality and connections can be configured at run time, to adapt them to different applications or operating modes. Such systems can be classified according to the granularity of the FUs. Fine-grained approaches, typically exploiting FPGA devices as underlying technology, involve bit-level FUs, resulting in a higher flexibility but requiring long configuration time (due to the configuration bitstream size). Coarse-grained reconfigurability, on the other hand, provides word-level FUs, thus providing less flexibility while guaranteeing faster configuration phases. CGR architectures are usually exploited to design flexible ASICs, making them capable of switching among a finite set of functionality. In such design cases, high efficiency in terms of area obstruction of the designed system can be easily obtained, but not all the resources are evenly involved in the computation. Dedicated power management techniques are needed to reduce the overhead, in terms of power consumption, related to resources that are not involved in each operating mode [17]. CGR architectures already demonstrated being suitable to address application scenarios that require flexibility along with strong area, power, and execution efficiency [13, 20].

One of the main issues of CGR architectures is their complex mapping and programming [21, 22]. Several works tried to automate the mapping of applications and computational kernels onto CGR and multicore systems [23–25]. The mapping problem requires specific knowledge of the considered kernels that usually have to be identified and specified by means of hardware description languages. The mapping effort is directly proportional to the number of involved kernels [26]. Recently, dataflow models demonstrated to be very useful in this scenario [27, 28]. Dataflows describe programs through a graph whose nodes are processing elements (*actor*) linked by point-to-point unidirectional channels managed according to a FIFO protocol. Actors encapsulate their own state and communicate only through atomic packets of data (*tokens*). Due to their intrinsic modularity, dataflows favour hardware and software components definition and reuse. Furthermore, they are natively capable of highlighting the intrinsic parallelism of the specified applications. The Multi-Dataflow Composer (MDC) tool, adopted within the presented work, relies on dataflows (RVC-CAL formalism by MPEG is currently supported) to solve the CGR mapping problem. It exploits the characteristics of such kind of models to provide several advanced features (e.g., power management [17, 18] or coprocessing units automatic generation [29]).

2.2. Power Management. Power consumption in digital devices is composed mainly of two different contributions: dynamic and static. The former is due to capacitance charging/discharging when logic transitions occur (i.e., switching activity). The latter is due to leakage currents and it is consumed also when no circuit activity is present. Modern

designers need to consider both terms when conceiving smart management strategies. Several techniques (clock gating, multifrequency, operand isolation, multithreshold, multisupply libraries, power gating, etc.) exist and, in some cases, they are automatically implemented by commercial synthesis/place-and-route tools. In custom computing systems, some advanced design tools support the designers in the application-driven customization of the hardware architectures [30, 31]. However, generally speaking, invasive techniques (requiring insertion of additional logic and target technology support (such as the availability of dedicated cells and processes on the implementation stack)) still need tools to be guided with significant manual effort by the designer.

Clock gating is an example of quite noninvasive technique. It may reduce the dynamic power consumption due to the clock tree and to sequential logic up to the 40% [32]. It consists in shutting off the clock of the unused synchronous logic, by means of simple *AND* gates. Clock gating has been deeply automated and it is available on most of the commercial synthesizers. In the MPEG-RVC community, recent studies [33] presented an extension of a High-Level Synthesis tool, Xronos, to selectively switch off clock signal for parts of the circuit that are idle due to stalls in the pipeline, to reduce power consumption. Moreover, as mentioned, the MDC tool has the capability of identifying, by means of a graph-based analysis of the input dataflow specifications, independent circuitry regions. These logic regions can be clock gated to dynamically adapt power consumption when switching between different functionalities [17, 18]. From the technical point of view, in ASIC designs *AND* gates can be used directly on the clock to disable it, while in FPGA designs the clock network cannot be modified by the insertion of any custom logic and dedicated cells are required (Xilinx boards, e.g., are equipped with dedicated blocks (*BUFGs*), whose outputs can drive distinct regions of logic powering down different design portions (when enabled)). Clock gating can be applied at different granularities: fine-grained approaches act on single registers, whereas coarse-grained ones are referred, as in [17, 18, 33], to a set of resources. Commercial synthesizers normally can automate only fine-grained strategies.

Power gating is quite invasive. The main idea behind it is as follows: if a specific portion of the design is not used in a given computation mode, then it can be completely switched-off by means of a sleep transistor. This technique, as the clock gating one, is applicable at different granularities: fine-grained approaches require driving a different sleep transistor for every cell in the system, while coarse-grained ones, again, operate on a set of resources instantiating one sleep transistor to drive different cells connected to a shared power network. MDC, as discussed in [18], supports also automatic power gating for CGR architectures. Each identified logic region in the CGR system is implemented (no matter of its nature or characteristics) as a different power domain (PD) that, in order to be managed, requires to insert and drive the following resources:

(i) The *sleep transistor* between the gated region and the main power supply to switch on/off the derived power supply

(ii) The *isolation logic* between the gated region and normally-on cells to avoid the transmission of spurious signals in input to the normally-on cells

(iii) The *state retention logic* to maintain, where needed, the internal state of the gated region

MDC, besides defining the power gated design netlist, provides also the automatic definition of the power format file. This file specifies the shut-off, isolation, and state retention rules (if you have 10 PD you are required to define $3 * 10 = 30$ interfaces), along with their respective enable signals ($[1 *$ shut-off $+ 1 *$ isol. $+ 2 *$ reten.$] * 10 = 40$ signals) and a dedicated Power Controller to properly drive them. The power format file automatically created by MDC is compliant with the Silicon Integration Initiative's Common Power Format (CPF), whose definition is driven mainly by developers using Cadence [34] tools.

2.2.1. Modelling. To the best of our knowledge, literature does not treat the problem of modelling power gating and clock gating costs in CGR designs. Some approaches only partially address the issue. For example, [35] focusses on low-power techniques and power modelling for FPGAs. In [36], only clock gating is taken into account: different power states (on the basis of the clock enable signals) are defined and their consumption is characterized by low-level Power Analysis results. [37] focusses on estimating the leakage reduction for power gating and reverse body bias.

The CASPER simulator for shared memory many-core processors [38] includes precharacterized libraries containing power dissipation models of different hardware components, enabling accurate power estimation at a high-level exploration stage. In particular the authors implement Chip-wide Dynamic Voltage, Frequency Scaling, and Performance Aware Core-Specific Frequency Scaling. The FALPEM framework [39] provides power estimations at preregister transfer level (RTL) stage, specifically targeting the power consumed by clock network and interconnect, but power and clock gating costs are not defined. Other approaches perform an estimation that considers different components. Li et al. [40] propose an architecture-level integrated power, area, and timing modelling framework for multicore systems, which evaluates system building blocks (CPU, buses, etc.) for different technology nodes, providing also power gating support. Finally, the work presented in [41] focuses on on-chip networks.

3. Design Suite for Coarse-Grained Power-Aware Systems

This section discusses the proposed technique for modelling the power consumption of a CGR system when clock gating or power gating are applied. These models, combined in a selection algorithm, can be exploited for developing an automated design flow for power-efficient CGR systems, where the optimal saving strategy is selected for each identified working set of resources.

In this work, we have embedded these models and the algorithm in the Multi-Dataflow Composer (MDC) tool,

a framework capable of CGR systems characterization. MDC provides a comprehensive design suite automating several development tasks of the synthesis and development of CGR systems, within design flows targeting both FPGA [29] and ASIC. The tool provides extensive support to dynamic power management [17, 18], addressing power-constrained design cases scenarios, completely automating implementation and control of clock gating and power gating strategies in the final CGR platform. Nevertheless, such techniques are not currently addressed in a hybrid manner, the users must choose the approach to be used in the design without an a priori automated analysis process. Such an unsupported selection may easily lead to suboptimal implementations on the final platform.

In the following, Section 3.1 provides an overview of the MDC baseline functionality and of the current power management support. Section 3.2 discusses the proposed power models and automated selection algorithm identifying optimal power management strategy in CGR systems. In both sections, step-by-step examples are presented to clarify the methodology.

3.1. The Multi-Dataflow Composer Tool. MDC automates generation and management of CGR systems, facing the complex mapping of multiple applications onto a single reconfigurable architecture. It automates the mapping process and guarantees the minimization of hardware resources, allowing for significant area/energy savings [12, 42]. In literature, this problem is known as *datapath merging* and it deals with the combination of a set of input datapaths, described by means of graphs, onto a single reconfigurable datapath. It aims at maximally sharing (among the different input graphs) both processing nodes and connections.

MDC is naturally compliant with the RVC-CAL formalism and natively supports Dataflow Process Network (DPN) models as input. Currently, it is interfaced with the Open RVC-CAL Compiler, Orcc [43]. The Orcc front-end is responsible for parsing, one-by-one, the high-level DPN specifications of the different datapaths that MDC will merge within the CGR system. Please note that MDC can be interfaced with other graph parsers, so that it will be able to be easily adapted to any other dataflow-based modelling environment.

As depicted by Figure 1, the MDC baseline flow involves three main phases:

(1) The input *DPNs parsing*, performed by the Orcc front-end, translates the RVC-CAL specifications into Java Intermediate Representations (IRs), which are basically directed graphs.

(2) The *datapath merging*, performed by the MDC front-end, combines the IRs into a reconfigurable IR, inserting (where necessary) special switching actors (responsible for properly distributing the token flow among the different merged DPNs) keeping trace of the system programmability through a dedicated *Configuration Table* (TAB in Figure 1).

(3) The *hardware platform generation*, performed by the MDC back-end, leads to the creation of the RTL that describes the CGR system itself, where each actor of the reconfigurable IR is mapped onto a different hardware FU. In this phase, the hardware communication protocol and the *HDL (Hardware Description Language) components library* (providing the RTL descriptions of the required FUs, manually or automatically generated) are provided as input to the tool.

At the hardware level, reconfiguration takes place in a single clock cycle. It is achieved through low overhead switching elements (SBoxes) that allow the sharing of common resources among different input DPNs. SBoxes are simple combinatorial multiplexers and demultiplexers, whose configuration is stored into dedicated Look-Up tables that, according to the *Configuration Table*, compute the selectors necessary for the correct data forwarding in order to implement the requested functionality.

3.1.1. Automated Power Management. Dealing with reconfigurable architectures, and in particular with CGR systems, the power consumption has to be carefully taken into consideration. Such a kind of systems is affected by resource redundancy, mainly due to the FUs that are not shared among different functionalities. Thus, when a certain functionality is executed, part of the design (not involved in the computation) is in an idle state and can uselessly consume precious power. Fortunately the unused resources, for each implemented functionality, depend on the input specifications and, therefore, are fixed at design-time.

Given these considerations, then, a CGR system can be characterized by a set of disjointed *logic regions* (LRs), grouping the resources that are always active/inactive at the same time. The MDC power management extension is capable of automatically identifying LRs. It performs the *LRs identification* at a high-level of abstraction, on the reconfigurable IR, by exploiting the intrinsic modularity of the dataflow graphs. Once the LRs have been identified, the MDC dynamic power manager automatically applies, according to the user selection, either clock gating [17] or power gating [18] on the resulting CGR hardware platform. The identification of the minimal number of LRs is guaranteed, to minimize the power overhead of the extra logic needed to implement the selected power saving strategies. An overview of the power management extension is provided by Figure 1.

For each input DPN G_i, the currently available algorithm determines the set V_i', which contains all the resources of the reconfigurable IR activated by G_i. These are the original sets of LRs that represent the starting point for the algorithm to find the final LRs by iteratively comparing two V_i' sets at a time, determining their possible overlapping. If overlapping is found, its resources are removed by the two considered sets and a new V_j' (corresponding to a new LR) involving these shared resources is issued. V_j' groups resources that are shared among different input DPNs, while the remaining resources in the two V_i' will uniquely belong to the originally considered DPNs. This compare and split identification process guarantees that the number of LRs found by the MDC dynamic power manager is the minimum achievable one.

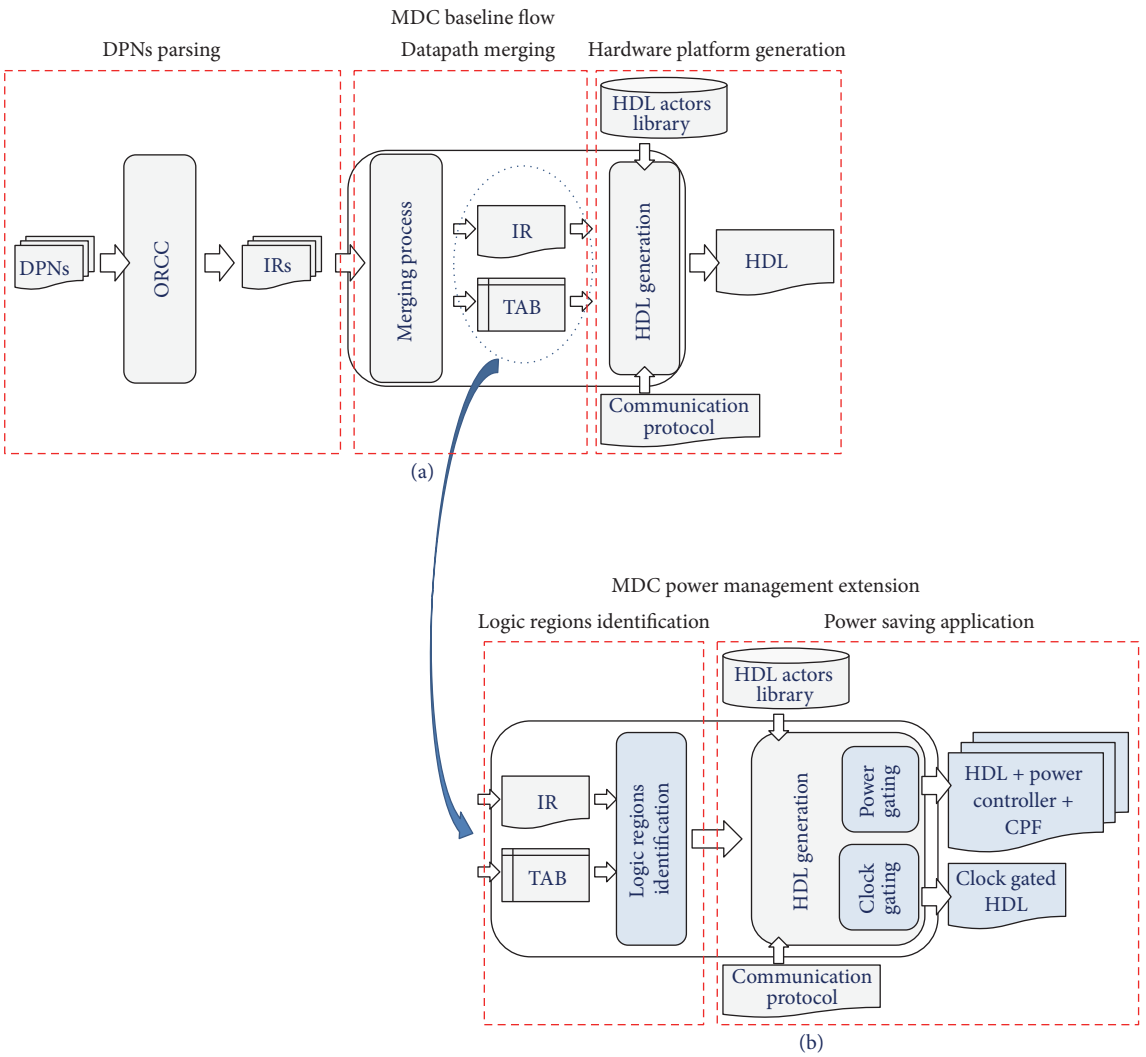

Figure 1: MDC design suite: baseline flow (a) and corresponding power extension (b).

If this number is still too high for the considered target platform, as it can happen if FPGAs are the target devices (in FPGA devices the number of hardware blocks that can drive the different LRs is limited; e.g., 32 BUFG units are available in Xilinx boards for clock management purposes), a *LRs merging* process has to be applied. MDC users are required to specify the target technology and the maximum number of implementable LRs. This latter is compared with the number of LRs determined by the compare and split identification process and, if necessary, the *LRs merging* process is applied. Two LRs at a time are unified (details on how to merge different LR sets can be found in [17], where two merging strategies (a power-aware one and a number-aware one) are presented) until the constraint fixed by the user is met. This process leads to a suboptimal system implementation: each DPN, while activating its corresponding LRs, may also activate some resources that do not contribute to its computation, leading to extra unnecessary power consumption.

MDC power management extension, during the HDL generation phase, provides also the implementation of the chosen power management strategy upon the identified LRs. It blindly applies the selected strategy to all the identified LRs, without any warranty on the approach effectiveness. Clock gating acts only on the dynamic contribute of the power consumption and requires a minimum logic overhead on the final platform. Indeed, the simplest implementation is achieved by means of one *AND* gate for each LR plus one unique *Enable Generator* to properly set the enable signals of the *AND* gates according to the desired functionality. On the contrary, power gating is able to reduce both power contributions, static and dynamic, by shutting off the power supply of the region. However, it is quite more invasive, since it requires one *power switch* for each LR, one *state retention* cell for each Flip-Flop whose state has to be kept also when the corresponding LR is off, and one *isolation* cell for each bit-wise wire that goes from a disabled LR to an enabled one. A different clock gating cell (again an *AND* gate) is required for each LR, according to the switching-off protocol for the proper operation of the retention cells (details on the power gating switch on/off protocol can be found in [11]). Furthermore,

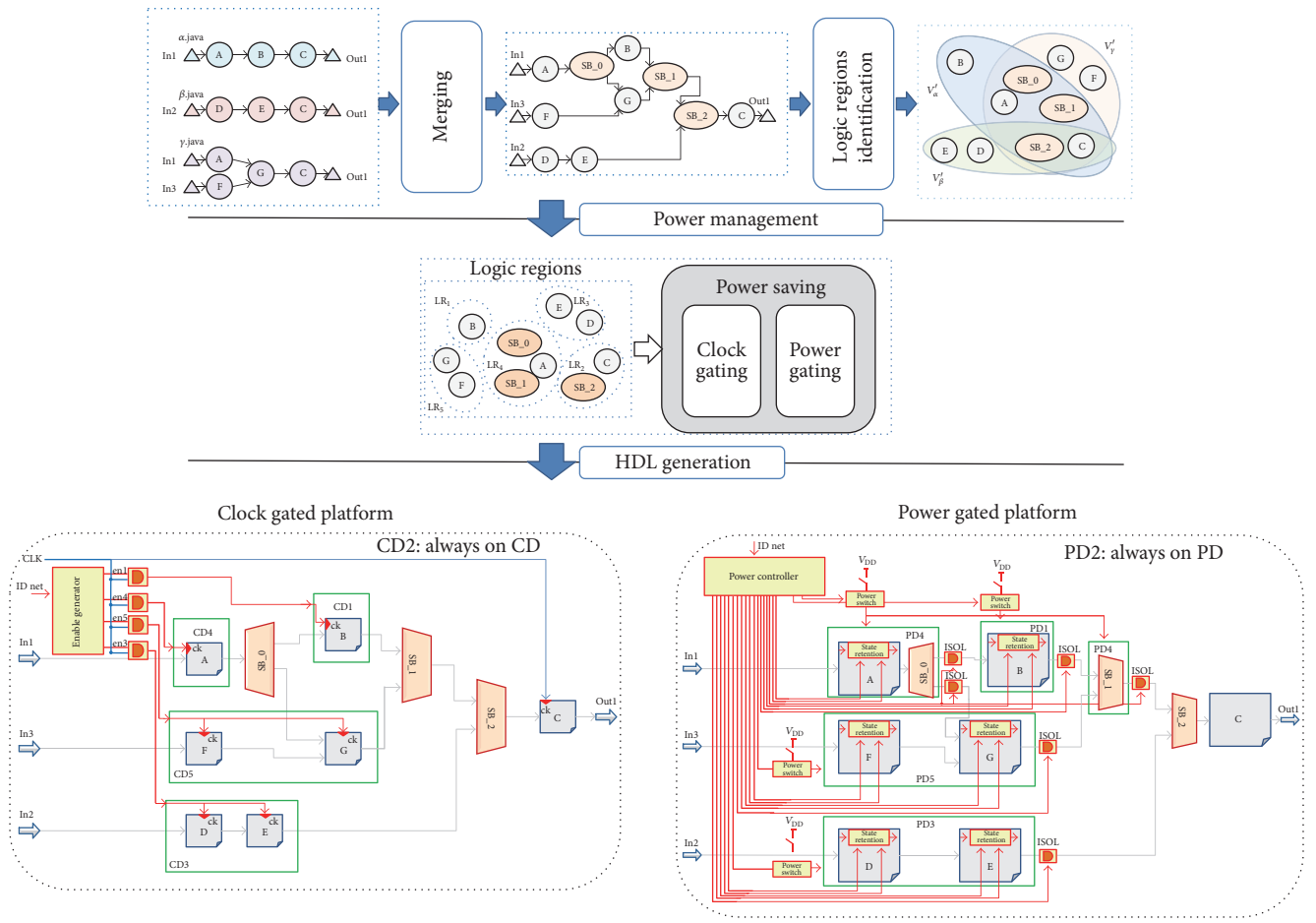

FIGURE 2: Step-by-step example of the MDC baseline and dynamic power management features. Saving strategies are blindly applied by the dynamic power manager on each identified LR.

one *Power Controller* block (involving a different finite state machine for each LR) is needed to properly drive the inserted *power switch*, *state retention* and *isolation* cells.

3.1.2. Step-by-Step Example. In order to clarify the features provided by MDC baseline functionality and its related dynamic power manager extension, this section describes a step-by-step example of the whole flow. Three different input functionalities, labelled α, β, and γ, are considered and modelled as DPNs. As first step of the baseline MDC functionality, the DPNs are parsed by the Orcc font-end and translated into Java IR graphs. Figure 2 depicts an overview of the whole flow starting from these input IRs (*α.java*, *β.java*, and *γ.java*). At this level, MDC combines the dataflows into a reconfigurable IR inserting the SBox actors (SB in Figure 2). Three SBoxes are required to share actor A between α and γ and actor C among all the three functionalities.

Once the reconfigurable IR has been derived, the dynamic power manager can identify the corresponding LRs. The starting V_i' sets are

(i) $V_\alpha' = \{A, B, C, \text{SB\_0}, \text{SB\_1}, \text{SB\_2}\}$;

(ii) $V_\beta' = \{D, E, C, \text{SB\_2}\}$;

(iii) $V_\gamma' = \{A, F, G, C, \text{SB\_0}, \text{SB\_1}, \text{SB\_2}\}$.

The compare and split identification process produces five different LRs:

(i) $\text{LR}_1 = \{B\}$.

(ii) $\text{LR}_2 = \{C, \text{SB\_2}\}$.

(iii) $\text{LR}_3 = \{D, E\}$.

(iv) $\text{LR}_4 = \{A, \text{SB\_0}, \text{SB\_1}\}$.

(v) $\text{LR}_5 = \{F, G\}$.

LR_4 and LR_2 involve shared resources, being activated, respectively, by α and γ and by α, β, and γ, while the remaining LRs involve nonshared resources. LR_1 is activated only by α, LR_3 only by β, and LR_5 only by γ.

At this point, the selected power saving strategy is applied during the CGR system HDL generation. Figure 2 shows both the final designs resulting from the application of clock gating and power gating.

The clock gated platform is shown in the bottom left corner of Figure 2. In this case, the identified LRs become the Clock Domains (CDs) of the resulting architecture, meaning that the involved actors are all driven by the same gated clock. SBoxes are not included in any CD since they are fully

combinatorial modules. It can be noticed by Figure 2 that the clock gating overhead is limited to four *AND* gates (LR_2 is activated by all the implemented functionalities and does not need to be turned off) and one *Enable Generator* that properly assigns the clock enable values.

The power gated platform is depicted in the bottom right corner of Figure 2. LRs define the architecture power domains (PDs) where, in this case, also SBoxes are taken into consideration. Power gating turns off the whole PD power supply and it has effect also on combinatorial blocks. Figure 2 clearly shows that the logic overhead of power gating is larger than the clock gating one. In the power gated platform *power switches*, *state retention* cells, *isolation* cells, and one *Power Controller* are inserted. Please note that clock gating cells are not reported for simplicity. Again the logic necessary to switch off LR_2 is avoided since this region is an always on one (being activated by all the input DPNs).

3.2. Automated Power Management with Hybrid Clock and Power Gating. To overcome the limits of a blindly applied unique power management strategy, we propose, in this paper, a power estimation flow capable of (1) characterizing, at a high-level of abstraction, the LRs identified by the MDC power extension, and of (2) autonomously applying the optimal power reduction technique for each LR. Power and clock gating overhead are estimated, based on LRs characteristics, before any physical implementation. This strategy is meant for ASIC technologies, which allow hybrid power and clock gating support over the same CGR design.

The estimation is based on two sets of models that determine the static and dynamic consumptions of each LR when clock gating or power gating are applied. The proposed models are derived after a single logic synthesis of the baseline CGR system generated by MDC, carried out with commercial synthesis tools from the analysis of the power reports obtained after netlist simulation. Such a synthesis constitutes the only implementation effort required for the designer, besides the characterization of technique-specific blocks such as the *Enable Generator* or the *Power Controller*. Models are technology-dependent since they include parameters that are characteristic of the chosen target technology library, as it will be discussed in Section 3.2.4.

3.2.1. Power Gating: Static Power Consumption Model. Static power can be estimated on the basis of the leakage contributes provided, for each cell, by the targeted ASIC library. Given any hardware FU (uniquely corresponding to an actor of the reconfigurable IR) its static power can be obtained by summing up the single contributions of the adopted cells. The static power consumption term is tightly related to the LR area: the more cells are included in the considered region, the more is its corresponding static dissipation.

The proposed model for the static power consumption is defined as follows:

$$\begin{aligned} P_{\text{lkg}}(\text{LR}_i) &= P_{\text{lkgON}}(\text{LR}_i) + \text{Ext\_Over}_{\text{lkg}}(\text{LR}_i) \\ &= \sum_{\text{actors}\in\text{LR}_i} \Bigg[P_{\text{lkg}}(\text{cmb}) + P_{\text{lkg}}(\text{RC}) * \#\text{rtn} \\ &\quad + P_{\text{lkg}}(\text{reg}) * \frac{\#\text{reg} - \#\text{rtn}}{\#\text{reg}} \Bigg] * \text{Ti}_{\text{ON}} + \left[P_{\text{lkg}}(\text{ISO}_{\text{ON}}) \right. \\ &\quad \left. * \text{Ti}_{\text{ON}} + P_{\text{lkg}}(\text{ISO}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right] * \#\text{iso} \\ &\quad + \left[P_{\text{lkg}}(\text{Contr}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{lkg}}(\text{Contr}_{\text{OFF}}) \right. \\ &\quad \left. * \text{Ti}_{\text{OFF}} \right] + \left[P_{\text{lkg}}(\text{CG}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{lkg}}(\text{CG}_{\text{OFF}}) \right. \\ &\quad \left. * \text{Ti}_{\text{OFF}} \right]. \end{aligned} \tag{1}$$

Dealing with a prospective power gating implementation, the static estimation (1) for each LR involves two terms: $P_{\text{lkgON}}(\text{LR}_i)$ corresponds to the static consumption when the LR is active and $\text{Ext\_Over}_{\text{lkg}}(\text{LR}_i)$ refers to the power overhead due to the additional power gating logic. This second term does not consider the *power switch* overhead, since it is not included in the prelayout netlist. Power gating prevents, by definition, any static dissipation on the LR when disabled; therefore, (1) does not present any $P_{\text{lkgOFF}}(\text{LR}_i)$.

$P_{\text{lkgON}}(\text{LR}_i)$ is obtained as the multiplication of the LR activation time Ti_{ON} and the sum of leakage power of the involved actors, considering separately combinatorial and sequential logic. The former, $P_{\text{lkg}}(\text{cmb})$, is equal to the leakage of the combinatorial cells within the considered LR. The latter is related to the number of registers (#reg) within the LR and their need (according to the implemented functionality) of preserving or not their status, by means of *state retention* cells, when the region is inactive. Then it involves, in turn, two terms. The first one refers to the registers whose state can be lost and it is estimated on the basis of the static consumption of the sequential cells ($P_{\text{lkg}}(\text{reg})$), as an average on the number of registers that are not retained. The second one refers to the retention cells and it is estimated starting from the number of registers whose state has to be maintained (#rtn) multiplied by the leakage of a single *state retention* cell ($P_{\text{lkg}}(\text{RC})$), whose value is retrieved from the target ASIC library.

$\text{Ext\_Over}_{\text{lkg}}(\text{LR}_i)$ is composed of three terms: the first one is related to the *isolation* cells (#iso), the second one to the *Power Controller,* and the third one to the clock gating cell (the power gating switch off protocol requires applying clock gating at the region level, before retaining the registers value). Note that, unlike P_{lkgON}, for the three abovementioned terms, $\text{Ext\_Over}_{\text{lkg}}$ characterizes the LR static consumption in both its on and off states. In the on state, the model accounts for the static consumption in the on state (e.g., $P_{\text{lkg}}(\text{ISO}_{\text{ON}})$) multiplied by the activation time Ti_{ON} and by the overall number of cells within the LR (e.g., #iso). In the off state, the model accounts for the static consumption in the off state (e.g., $P_{\text{lkg}}(\text{ISO}_{\text{OFF}})$) multiplied by the inactive time Ti_{OFF} and by the overall number of cells within the LR (e.g., #iso). Please note that there is just one *Power Controller* for all the LRs and one clock gating cell per LR, but an a priori characterization phase would be required to the designer, since their consumption values cannot be retrieved directly from any ASIC library.

3.2.2. Power Gating: Dynamic Power Consumption Model. Estimating the dynamic power is more complex than estimating the static one, since this term strongly depends on the nodes switching activity. Frequently, commercial tools (e.g., Cadence Encounter Digital Implementation System) consider dynamic power as composed of two main terms, as depicted by (2): a *net* contribution due to the power dissipated throughout the wires linking the cells, and an *internal* contribution due to the dissipation occurring inside the cells [44]:

$$\begin{aligned} P_{\text{dyn}} &= P_{\text{net}} + P_{\text{int}} \\ &= \frac{1}{2} f V_{DD}^2 \sum_{\text{net}_j} C_{\text{load}_j} \text{SW}_j + f \sum_{\text{cell}_i} P_i \text{SW}_i. \end{aligned} \tag{2}$$

The operating frequency, f, influences both terms. P_{net} accounts for the load capacitance of each net_j (bearing a specific capacitance C_{load_j}) and the related switching activity (SW_j), whereas P_{int} depends on the power per MHz dissipated by each cell (P_i) and the related switching activity (SW_i).

Currently, the developed model is able to estimate only the P_{int} contribution that can be expressed for each single LR as follows:

$$\begin{aligned} P_{\text{int}}(\text{LR}_i) &= P_{\text{intON}}(\text{LR}_i) + \text{Ext\_Over}_{\text{int}}(\text{LR}_i) \\ &= \sum_{\text{actors} \in \text{LR}_i} \left[P_{\text{int}}(\text{cmb}) + P_{\text{int}}(\text{RC}) * \#\text{rtn} \right. \\ &\quad \left. + P_{\text{int}}(\text{reg}) * \frac{\#\text{reg} - \#\text{rtn}}{\#\text{reg}} \right] * \text{Ti}_{\text{ON}} + \left[P_{\text{int}}(\text{ISO}_{\text{ON}}) \right. \\ &\quad \left. * \text{Ti}_{\text{ON}} + P_{\text{int}}(\text{ISO}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right] * \#\text{iso} \\ &\quad + \left[P_{\text{int}}(\text{Contr}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{int}}(\text{Contr}_{\text{OFF}}) \right. \\ &\quad \left. * \text{Ti}_{\text{OFF}} \right] + \left[P_{\text{int}}(\text{CG}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{int}}(\text{CG}_{\text{OFF}}) \right. \\ &\quad \left. * \text{Ti}_{\text{OFF}} \right]. \end{aligned} \tag{3}$$

Considering a prospective power gating implementation, the different parts of (3) basically reflect the ones of (1). The main difference among the static power model and the dynamic one is that this latter requires accurate data in terms of nodes switching activity. For this reason, the netlist of the baseline CGR system is not sufficient to retrieve accurate values from the power reports and one different simulation of the netlist for every implemented functionality is required. Thus, dynamic power model takes into consideration the real system switching activity provided by the hardware simulations.

The P_{net} term of (2) is not currently addressed in our model. Nevertheless, as demonstrated further on in this work (please see Tables 8, 9, 12, and 14) neglecting this term seems not to affect the optimal identification of the region to be gated. We have already planned to extend the proposed methodology to include the nets contribution in a near future.

3.2.3. Clock Gating: Static and Dynamic Power Consumption Models. Clock gating static and dynamic models are less complicated than the power gating ones, since clock gating requires a very low logic overhead and it positively acts only on the dynamic dissipation. Equations (4) and (5) report the models adopted, respectively, for the static power estimation and for the dynamic power one, referring to a clock gated design:

$$\begin{aligned} P_{\text{lkg}}(\text{LR}_i) &= P_{\text{lkg}}(\text{LR}_i) + \text{Ext\_Over}_{\text{lkg}}(\text{LR}_i) \\ &= \sum_{\text{actors} \in \text{LR}_i} \left[P_{\text{lkg}}(\text{cmb}) + P_{\text{lkg}}(\text{reg}) \right] \\ &\quad + \left[P_{\text{lkg}}(\text{Enab}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{lkg}}(\text{Enab}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right] \\ &\quad + \left[P_{\text{lkg}}(\text{CG}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{lkg}}(\text{CG}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right], \end{aligned} \tag{4}$$

$$\begin{aligned} P_{\text{int}}(\text{LR}_i) &= P_{\text{int}}(\text{LR}_i) + \text{Ext\_Over}_{\text{int}}(\text{LR}_i) \\ &= P_{\text{int}}(\text{combLR}_i) + P_{\text{intON}}(\text{seqLR}_i) \\ &\quad + \text{Ext\_Over}_{\text{int}}(\text{LR}_i) \\ &= \sum_{\text{actors} \in \text{LR}_i} \left[P_{\text{int}}(\text{cmb}) + P_{\text{int}}(\text{reg}) * \text{Ti}_{\text{ON}} \right] \\ &\quad + \left[P_{\text{int}}(\text{Enab}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{int}}(\text{Enab}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right] \\ &\quad + \left[P_{\text{int}}(\text{CG}_{\text{ON}}) * \text{Ti}_{\text{ON}} + P_{\text{int}}(\text{CG}_{\text{OFF}}) * \text{Ti}_{\text{OFF}} \right]. \end{aligned} \tag{5}$$

At the logic region level, (4) considers always the combinatorial and sequential contributions for both the on or off states, since clock gating does not affect the system leakage, whereas (5) considers always the combinatorial part for both the on or off states, since combinatorial logic cannot benefit from clock gating and the sequential contribution only during the LR active time. The overhead, $\text{Ext\_Over}_{\text{int}}(\text{LR}_i)$, is given by the clock gating cell and the *Enable Generator*. Please remember that implementing clock gating management at a coarse-grained level, just one clock gating cell per LR has to be inserted within the system. Equation (5) is pretty much the same as (4), a part from the fact that, dealing with the dynamic model, clock gating effects are estimated by omitting the contribute of sequential logic when the LR is off.

3.2.4. Parameters Discussion. The proposed models are determined by the intrinsic features of the LRs. In particular, they consider the following:

(i) *Architectural Parameters*. LRs composition determines the amount of involved combinatorial and sequential cells.

(ii) *Functional Parameters*. LRs behaviour defines the region activation time and if its status has to be preserved or not.

(iii) *Technological Parameters*. Target technology has an impact on the ratio between dynamic and static power (as it will be demonstrated in Section 4.2) and on the different cells characterization.

Table 1 reports, for each parameter considered in (1), (3), (4), and (5), their classification. A deeper explanation about $P_{\text{lkg/int}}(\text{cmb})$ and $P_{\text{lkg/int}}(\text{reg})$ is necessary. They are not associated with any specific parameters class, indeed they depend on type and number of involved cells composing the considered LR and also on the system switching activity (especially for the internal contribute). These values are gathered by the reports of the baseline CGR system netlist, assuming that the amount and type of cells composing the FUs do not change as power saving strategies are applied (except for the retained registers).

3.2.5. Step-by-Step Example. The example proposed in Figure 2 shows the merging process of three DPNs in a CGR system, where five LRs have been identified. Equations (1), (3), (4), and (5) can be applied to all of them. However, as already discussed, LR_2 can be discarded: being common to all the input DPNs, it is always active and does not require to be switched-off. As defined in the previous section, the parameters reported in Table 2 are extracted by the reference technology library or characterized by synthesis trials (see the definition provided in Table 1). The power consumption values in Table 3 have been extracted by the synthesis reports of the baseline (with no power saving applied) CGR platform and required three hardware simulations (one for each input kernel). These simulations are necessary to correctly estimate the internal power consumption of the different LRs, taking into account the real switching activity of the design. In practice, power values are determined as an average of those obtained according to the different switching activity profiles.

Starting from the data in Tables 3 and 2, here follows the detailed equations characterization for LR_5, which include actors F and G.

When power gating is applied, the static power consumption of LR_5 is derived according (1), as follows:

$$\begin{aligned}
P_{\text{lkg}}(\text{LR}_5) &= \left[\left(P_{\text{lkg}}(\text{comb}_F) + P_{\text{lkg}}(\text{RC}) * \#\text{rtn}F \right.\right. \\
&\quad \left. + P_{\text{lkg}}(\text{reg}_F) * \frac{\#\text{reg}F - \#\text{rtn}F}{\#\text{reg}F}\right) + \left(P_{\text{lkg}}(\text{comb}_G)\right. \\
&\quad + P_{\text{lkg}}(\text{RC}) * \#\text{rtn}G + P_{\text{lkg}}(\text{reg}_G) \\
&\quad \left.\left. * \frac{\#\text{reg}G - \#\text{rtn}G}{\#\text{reg}G}\right)\right] * T5_{\text{ON}} + \left[P_{\text{lkg}}(\text{ISO}_{\text{ON}})\right. \\
&\quad \left. * T5_{\text{ON}} + P_{\text{lkg}}(\text{ISO}_{\text{OFF}}) * T5_{\text{OFF}}\right] * \#\text{iso}_5 \\
&\quad + \left[P_{\text{lkg}}(\text{Contr}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{lkg}}(\text{Contr}_{\text{OFF}})\right. \\
&\quad \left. * T5_{\text{OFF}}\right] + \left[P_{\text{lkg}}(\text{CG}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{lkg}}(\text{CG}_{\text{OFF}})\right. \\
&\quad \left. * T5_{\text{OFF}}\right] = [(213 + 17.15 * 128) + (273 + 17.15 \\
&\quad * 64 + 1385 * 0.5)] * 0.3 + [4.27 * 0.3 + 1.39 \\
&\quad * 0.7] * 32 + [95.44 * 0.3 + 88.63 * 0.7] + [5.77 \\
&\quad * 0.3 + 4.71 * 0.7] = 1509.219.
\end{aligned} \tag{6}$$

The internal power consumption is given by (3):

$$\begin{aligned}
P_{\text{int}}(\text{LR}_5) &= \left[\left(P_{\text{int}}(\text{comb}_F) + P_{\text{int}}(\text{RC}) * \#\text{rtn}F \right.\right. \\
&\quad \left. + P_{\text{int}}(\text{reg}_F) * \frac{\#\text{reg}F - \#\text{rtn}F}{\#\text{reg}F}\right) + \left(P_{\text{int}}(\text{comb}_G)\right. \\
&\quad + P_{\text{int}}(\text{RC}) * \#\text{rtn}G + P_{\text{int}}(\text{reg}_G) \\
&\quad \left.\left. * \frac{\#\text{reg}G - \#\text{rtn}G}{\#\text{reg}G}\right)\right] * T5_{\text{ON}} + \left[P_{\text{int}}(\text{ISO}_{\text{ON}})\right. \\
&\quad \left. * T5_{\text{ON}} + P_{\text{int}}(\text{ISO}_{\text{OFF}}) * T5_{\text{OFF}}\right] * \#\text{iso}_5 \\
&\quad + \left[P_{\text{int}}(\text{Contr}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{int}}(\text{Contr}_{\text{OFF}})\right. \\
&\quad \left. * T5_{\text{OFF}}\right] + \left[P_{\text{int}}(\text{CG}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{int}}(\text{CG}_{\text{OFF}})\right. \\
&\quad \left. * T5_{\text{OFF}}\right] = [(537 + 383.25 * 128) + (363 + 383.25 \\
&\quad * 64 + 44068 * 0.5)] * 0.3 + [2.7 * 0.3 + 0 * 0.7] \\
&\quad * 32 + [1449 * 0.3 + 1488 * 0.7] + [169 * 0.3 \\
&\quad + 292 * 0.7] = 30217.72.
\end{aligned} \tag{7}$$

When clock gating is considered, (4) and (5) are computed as follows:

$$\begin{aligned}
P_{\text{lkg}}(\text{LR}_5) &= \left[\left(P_{\text{lkg}}(\text{comb}_F) + P_{\text{lkg}}(\text{reg}_F)\right)\right. \\
&\quad \left. + \left(P_{\text{lkg}}(\text{comb}_G) + P_{\text{lkg}}(\text{reg}_G)\right)\right] + \left[P_{\text{lkg}}(\text{Enab}_{\text{ON}})\right. \\
&\quad \left. * T5_{\text{ON}} + P_{\text{lkg}}(\text{Enab}_{\text{OFF}}) * T5_{\text{OFF}}\right] + \left[P_{\text{lkg}}(\text{CG}_{\text{ON}})\right. \\
&\quad \left. * T5_{\text{ON}} + P_{\text{lkg}}(\text{CG}_{\text{OFF}}) * T5_{\text{OFF}}\right] = [(213 + 1232) \\
&\quad + (273 + 1385)] + [84.51 * 0.3 + 76.54 * 0.7] \\
&\quad + [5.77 * 0.3 + 4.71 * 0.7] = 3186.96, \\
P_{\text{int}}(\text{LR}_5) &= [(P_{\text{int}}(\text{comb}_F) + P_{\text{int}}(\text{reg}_F) * T5_{\text{ON}}) \\
&\quad + (P_{\text{int}}(\text{comb}_G) + P_{\text{int}}(\text{reg}_G) * T5_{\text{ON}})] \\
&\quad + [P_{\text{int}}(\text{Enab}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{int}}(\text{Enab}_{\text{OFF}}) \\
&\quad * T5_{\text{OFF}}] + [P_{\text{int}}(\text{CG}_{\text{ON}}) * T5_{\text{ON}} + P_{\text{int}}(\text{CG}_{\text{OFF}}) \\
&\quad * T5_{\text{OFF}}] = [(537 + 22489 * 0.3) \\
&\quad + (363 + 44068 * 0.3)] + [1351 * 0.3 + 1320 * 0.7] \\
&\quad + [169 * 0.3 + 292 * 0.7] = 22451.8.
\end{aligned} \tag{8}$$

Table 4 summarizes all the values achieved applying the proposed static and dynamic models to all the different logic regions.

3.2.6. Hybrid Clock and Power Gating Support and Integration in MDC. The discussed models (described in (1), (3), (4), and (5)) have been integrated in the MDC design flow, in order to implement a fully automated power management

Table 1: Parameter classification. Table reports for each considered parameter, typology (architectural, functional, and technological), description, and extraction method.

Parameter	Arch.	Funct.	Tech.	Description	Extraction
#reg	x			Number of sequential cells in the considered LR	Provided by the synthesis reports
#iso	x			Number of estimated isolation cells in the design	Obtained by counting the number of wires that connect the different LR among each other in the dataflow model
Ti_{ON}		x		Activation time	Found by means of a high-level (directly on the dataflow) profiling of the targeted scenario
Ti_{OFF}		x		Off time	
#rtn		x		Number of retention cells in the design	Strictly related to the LR functionality and is chosen by the designer
$P_{lkg/int}(RC)$			x	Power estimation of retention and isolation cells	Determined by the selected synthesis process and can be obtained without any implementation run
$P_{lkg/int}(ISO_{ON})$			x		
$P_{lkg/int}(ISO_{OFF})$			x		
$P_{lkg/int}(Contr_{ON})$			x	Power estimation of PG and CG controller when the LR is on or off	It requires being characterized, according to the target technology, with dedicated synthesis trials
$P_{lkg/int}(Contr_{OFF})$			x		
$P_{lkg/int}(Enab_{ON})$			x		
$P_{lkg/int}(Enab_{OFF})$			x		
$P_{lkg/int}(cmb)$	x	x	x	Power consumed by combinatorial and sequential cells within the considered LR	Gathered by the reports of the baseline CGR system netlist
$P_{lkg/int}(reg)$	x	x	x		

Table 2: Contributions of static and internal power consumption extracted by the reference technology library or characterized by synthesis trials.

Parameters	lkg power [nW]	int power [nW]
$Enab_{ON}$	84.51	1351
$Enab_{OFF}$	76.54	1320
$Contr_{ON}$	95.44	1449
$Contr_{OFF}$	88.63	1488
CG_{ON}	5.77	169
CG_{OFF}	4.71	292
ISO_{On}	4.27	2.7
ISO_{OFF}	1.39	0
P(RC)	17.15	383.25

```
PG_set is empty;
CG_set is empty;
foreach LR_i in set LRs do
    evaluate_area(LR_i, area_th)
end
function: evaluate_area(LR_i, area_th):
calculate_LR_i_area;
if area_LR > area_th then
    evaluate_PG(LR_i);
else
    evaluate_CG(LR_i);
end
function: evaluate_PG(LR_i):
estimate_PG_total_overhead;
    if PG_total_overhead < 0 then
        estimate_CG_total_overhead;
        if PG_total_overhead < CG_total_overhead then
            add LR_i to PG_set;
        else
            add LR_i to CG_set;
        end
    else
        evaluate_CG(LR_i);
end
evaluate_CG(LR_i);
function: evaluate_CG(LR_i):
estimate_CG_total_overhead;
if CG_total_overhead < 0 then
    add LR_i to CG_set;
end
```

Algorithm 1: Automatic power saving strategy selection for CGR systems.

strategy. Designers are guided towards the optimal solution for each LR, rather than choosing a one-fit-to-all switching-off technique for all of them.

This automated selection flow is implemented as reported in Algorithm 1. For each LR, identified by the MDC power management extension, Algorithm 1 executes the following steps, embodied by different functions.

(1) Area Thresholding (See evaluate_area Function). As previously discussed, power gating is a quite invasive technique, requiring a lot of extra logic to be inserted in the nonswitchable always on domain. Thus, for small LRs, we can assume that it will not bring any benefit, so that power gating is not to be considered for implementation. Indeed, clock gating may still be beneficial, due to its very small additional logic amount.

(2) Power Gating Overhead Estimation (See evaluate_PG Function in Algorithm 1). Power gating cost is estimated in order to find out if it can lead to power saving or not. The prospective power and clock gating implementations are compared on the basis of their overall consumption. Equation (1) is applied and summed to (3), if there is not total power saving the algorithm goes to the *clock gating overhead estimation*. On the contrary, if there is saving it has to be compared with the sum of (4) and (5) to determine whether the current LR may benefit from power gating (despite its larger overhead) or from clock gating.

(3) Clock Gating Overhead Estimation (See evaluate_CG Function in Algorithm 1). Clock gating cost is estimated to investigate the possibility of achieving power saving with this technique. If the LR clock gating achievable saving does not counterbalance its implementation costs, the LR is discarded. This means that when the MDC back-end generates the RTL description of the CGR system, the LR logic is included in the always on domain. On the contrary, if the clock gating leads to an overall saving in terms of total power, the LR will be clock gated during the implementation.

The output of Algorithm 1 is the classification of the LRs, stating which one should be power gated (see PG_set in Algorithm 1), which one should be clock gated (see CG_set in Algorithm 1), and which ones should be included in the always on domain.

Figure 3 provides an overview of the modified design flow. As it can be noticed, the MDC tool and its power management extension are directly interfaced with the logic synthesizer. Algorithm 1 is implemented within the *Power Analysis* block. MDC baseline tool provides the HDL description of the plain CGR system and all the scripts to perform the synthesis of the CGR design and all the different hardware simulations (one for each input DPNs), as required by the proposed power estimation models. The power reports are then fed back to the MDC power management extension and parsed within the *Power Analysis* to execute Algorithm 1. The LRs classification (see LR class in Figure 3), generated by the *Power Analysis* block, is used by the *CG/PG HDL Generation* block to automatically define the hybrid, clock and power gating, power management support for the given CGR design.

Summarizing, this flow, with respect to what is discussed in Section 3.1, does not require designers to opt for a specific power management technique. On the basis of the proposed power estimation models and by linking MDC with a logic synthesis tool, the presented flow is capable of overcoming the limit of providing a one-fit-to-all solution. Each LR, in a CGR design, is supported (where necessary) with the optimal power management technique.

Table 3: Parameter and power consumption of each LR, extracted by the synthesis reports of the baseline CGR platform.

Logic region	Kernel	T_{ON}	Actors	#iso	#reg	#rtn
LR_1 LR_1	α	0.1	B	32	514	24
LR_3	β	0.6	D, E	32	8	8
LR_4	α, γ	0.4	A, SB_0, SB_1	96	256	64
LR_5	γ	0.3	F, G	32	265	192
Actor	Power [nW]					
	lkg seq.	int seq	lkg comb.	int comb.	#reg	#rtn
B	801	104987	121411	3916599	512	24
D	48	1104	51	319	4	4
E	56	1437	53	198	4	4
A	3264	89238	0	0	256	64
SB_0	0	0	307	409	—	—
SB_1	0	0	225	350	—	—
F	1232	22489	213	537	128	128
G	1385	44068	273	363	128	64

Table 4: Resulting power consumption of the different LRs when the proposed models are applied.

Logic region	lkg PG [nW]	int PG [nW]	lkg CG [nW]	int CG [nW]
LR_1	1240.69	404358.71	122294.15	3928700.60
LR_3	342.67	3884.44	294.67	3599
LR_4	2062.11	38705.08	3880.86	5598.6
LR_5	1509.58	30712.72	3186.96	22451.8

3.2.7. Step-by-Step Example. In this section, a step-by-step example of the application of Algorithm 1 is presented, considering the same example proposed in Figure 2. In that case, MDC *LRs identification* led to determining five LRs and the user-specified power management technique is blindly applied to all of them except LR_2. This region is used by all the input DPNs; thus, it is never disabled and does not require any power management support. In the following step-by-step example, shown in Figure 4, the threshold on the area ($area_{th}$) is set to 5%.

(i) LR_1 is processed by invoking evaluate_area(LR_1, 5).

(a) Its area is calculated: $area_{LR_1}$ = 52% of total area.

(b) $area_{LR_1} > area_{th}$, so that a prospective power gating implementation on LR_1 is taken into consideration by invoking evaluate_PG(LR_1).

(1) The static and the dynamic overheads are estimated, respectively, applying (1) and (3). The power gating overhead on the overall consumption is calculated by subtracting the power consumption of the LR when PG is applied, to the power consumption of the LR in the baseline design, the result is then divided by the total power consumption of the baseline design in order to estimate the total percentage power variation. The power variation when PG is applied to region LR_1 is −86.45%. Since this value is negative, power gating may be convenient if its total saving is larger than the clock gating one.

(2) Equation (4) is calculated and summed up to (5) to determine clock gating overhead on the overall consumption, which is −2.15%.

(3) Power gating is more beneficial than clock gating determining, overall, a larger power saving. Thus, LR_1 is added to PG_set.

(ii) LR_3 is processed by invoking evaluate_area(LR_3, 5).

(a) Its area is calculated: $area_{LR_3}$ = 0.4% of total area.

(b) $area_{LR_3} < area_{th}$, so that a prospective clock gating implementation on LR_3 is considered straight away by invoking evaluate_CG(LR_3).

(1) Equations (4) and (5) are evaluated to determine clock gating overhead on the overall consumption: CG_{over} = +0.01%.

(2) Clock gating is not beneficial since its overhead is positive. Thus LR_3 is discarded and no power management policy will be applied to it.

(iii) LR_4 is processed by invoking evaluate_area(LR_4, 5).

(a) Its area is calculated: $area_{LR_4}$ = 7% of total area.

(b) $area_{LR_4} > area_{th}$, so that a prospective power gating implementation on LR_4 is taken into consideration by invoking evaluate_PG(LR_4).

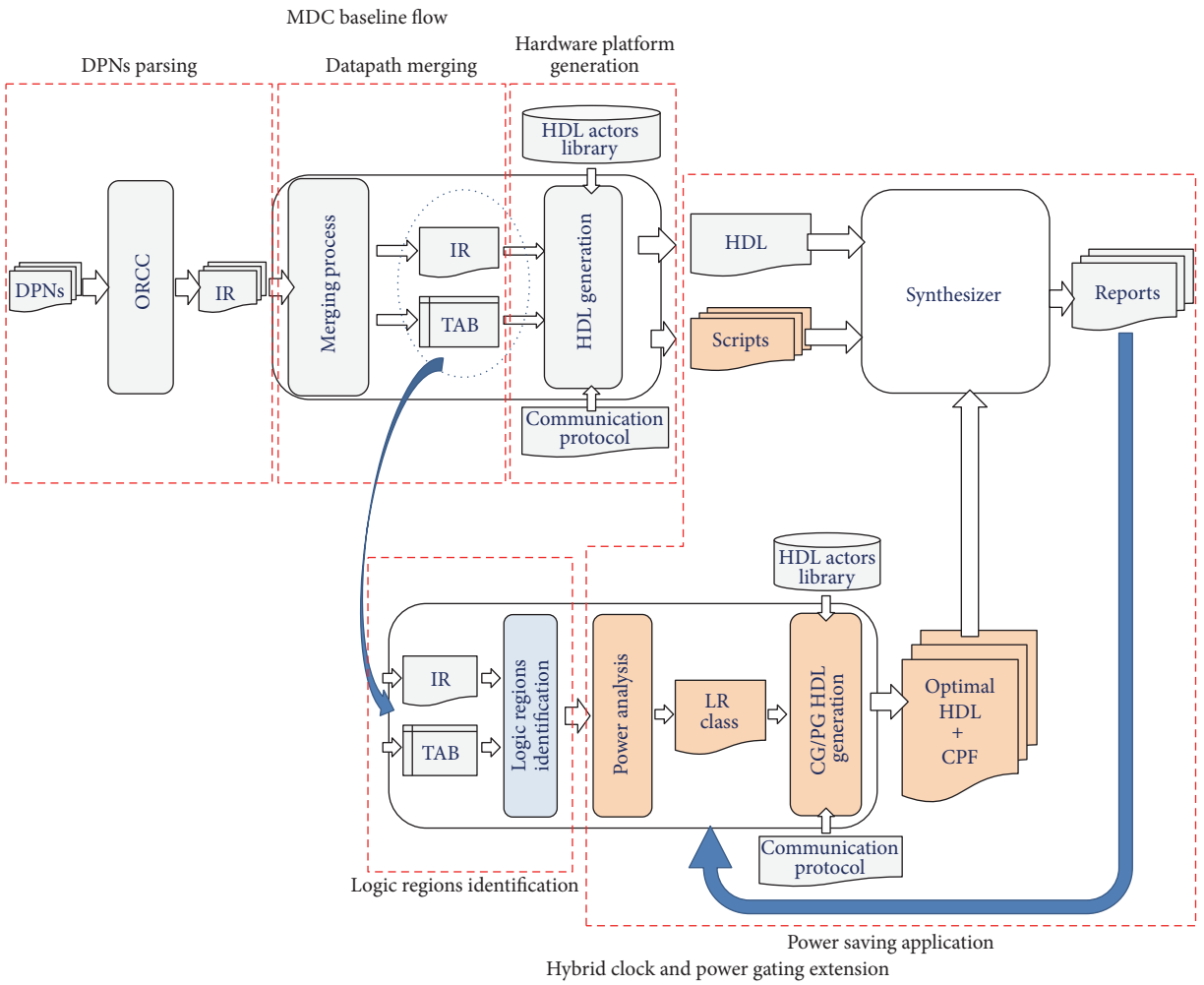

Figure 3: Enhanced MDC design suite: integration of the automated hybrid, clock, and power gated support.

(1) The static and the dynamic overheads are estimated, respectively, applying (1) and (3). The power gating overhead on the overall consumption is −1.2%. Since this value is negative, power gating may be convenient if its total saving is larger than the clock gating one.

(2) Equation (4) is calculated and summed up to (5) to determine clock gating overhead on the overall consumption, which is −2.0%.

(3) Clock gating is more beneficial than power gating determining, overall, a larger power saving. Thus, LR_4 is added to CG_set.

(iv) Finally, LR_5 is processed by invoking evaluate_area(LR_5, 5).

(a) Its area is calculated: $area_{LR_5}$ =15% of total area.

(b) $area_{LR_5} > area_{th}$, so that a prospective power gating implementation on LR_5 is taken into consideration by invoking evaluate_PG(LR_5).

(1) The static and the dynamic overheads are estimated, respectively, applying (1) and (3). The power gating overhead on the overall consumption is −0.89%. Since this value is negative, power gating may be convenient if its total saving is larger than the clock gating one.

(2) Equations (4) and (5) are evaluated to determine clock gating overhead on the overall consumption, which is −1.04%.

(3) Clock gating is more beneficial than power gating determining, overall, a larger power saving. Thus, LR_5 is added to CG_set.

The resulting hardware design with the hybrid application of clock gating and power gating is shown in Figure 5. Comparing this design with the two reported in Figure 2, we can notice as now the power gating is applied only to region LR_1 (called PD1 in the figure), while the clock gating is applied to regions LR_4 and LR_5 (called CD4 and CD5 in the figure); the SBoxes SB_0 and SB_1 included in region LR_4 are purely combinatorial, so they are not affected by the application

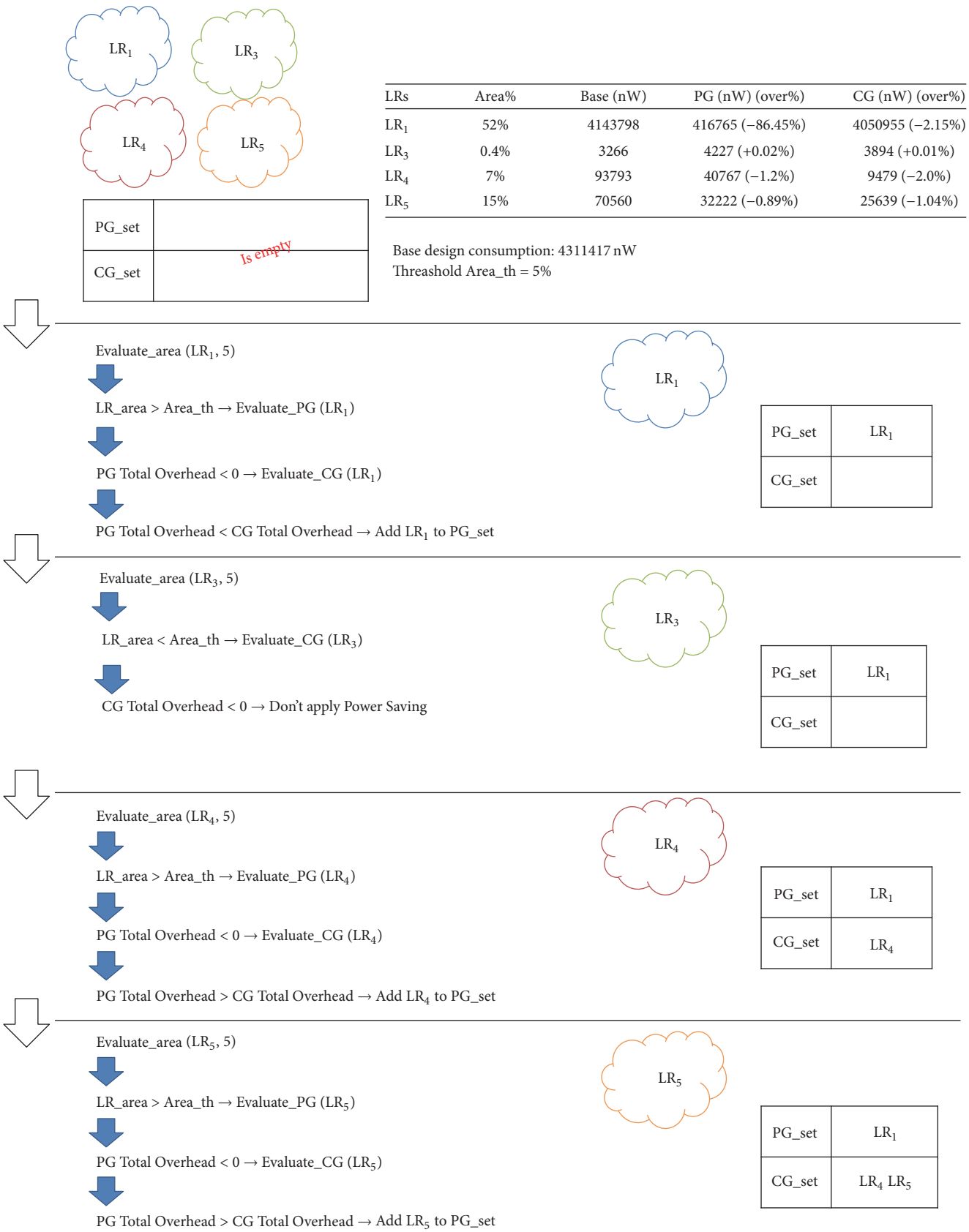

Figure 4: Step-by-step example of the enhanced MDC power extension implementing Algorithm 1.

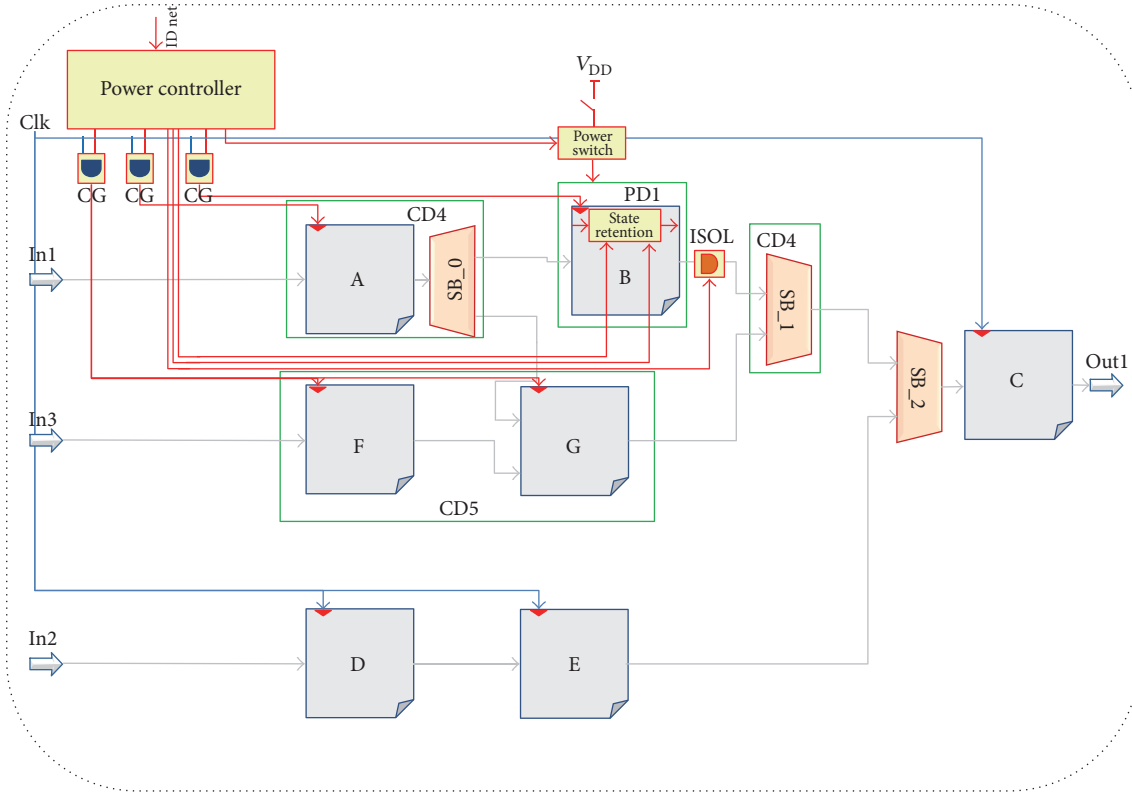

Figure 5: Enhanced MDC design suite: hardware platform with hybrid application of clock gating and power gating methodologies.

of the clock gating. All the remaining logic, which includes region LR_3, is always on.

4. Assessments

In order to assess the proposed power estimation flow and the effectiveness of the hybrid clock and power gating management, in this section we discuss two use cases, which are completely different in terms of both behaviour and resulting power consumption contributions. The first one deals with a simple FFT algorithm implemented on a 90 nm CMOS technology and it has been mainly adopted to evaluate in detail the proposed flow. The second one presents a more complex scenario. An image coprocessing unit, accelerating a zoom application, has been implemented both on a 90 nm and on a 45 nm CMOS technology in order to access the robustness of the proposed flow with different technology parameters.

In the following, Section 4.1 discusses the evaluation phase involving the FFT use case, while Section 4.2 describes the experimental results conducted to validate the approach on the zoom application. Finally, Section 4.4 details the benefits of adopting the proposed models and their correlated flow.

4.1. Evaluation Phase. This section deeply discusses the results obtained considering the FFT use case targeting a 90 nm CMOS technology.

4.1.1. Fast Fourier Transform Algorithm. Fast Fourier Transform (FFT) is an optimised algorithm for the Discrete Fourier Transform (DFT) calculation. It is widely adopted in several applications, ranging from the solving of differential equations to the digital signal processing. We refer to the original DFT equation:

$$X_k = \sum_{n=0}^{N} x_n e^{-i2\pi k(n/N)}, \quad k = 0, 1, \ldots, N-1. \tag{9}$$

The FFT algorithm that has been adopted for this use case has been proposed by Cooley and Tukey [45]. It aims at speeding up the calculation of a given size N DFT by considering smaller DFTs of size r, called radix. To obtain the whole original DFT, M stages of size r DFTs are required, where $N = r^M$. Small DFTs have to be multiplied by the so-called twiddle factors, according to the decimation in time variant of the algorithm. When the radix $r = 2$, the DTFs take the name of *butterflies* by their block scheme. The equations describing a butterfly are

$$\begin{aligned} X_0 &= x_0 + x_1 w_n^k, \\ X_1 &= x_0 - x_1 w_n^k, \end{aligned} \tag{10}$$

where X_0 and X_1 are the outputs, while x_0 and x_1 are the corresponding inputs. w_n^k are the twiddle factors, defined as

$$w_n^k = e^{-i2\pi k(k/N)}, \tag{11}$$

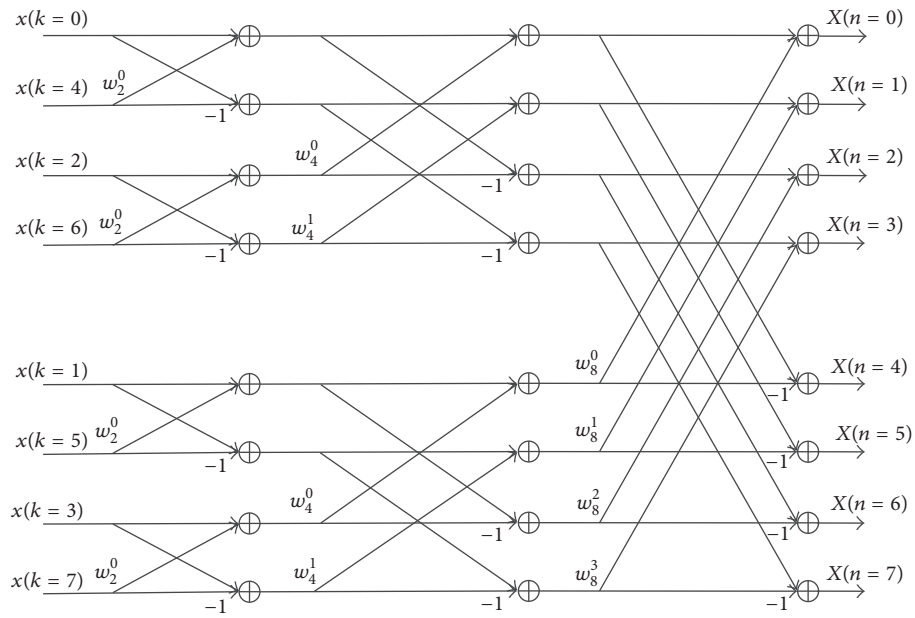

Figure 6: FFT use case: original design with 12 radix-2 butterflies for an FFT of size 8. Twiddle factors w_n^k are calculated according to (11).

where k and n are integers depending on the butterfly position in the FFT.

The adopted use case involves a radix-2 FFT of size 8, as depicted in Figure 6, obtained by means of three stages involving four butterflies each, meaning 12 butterflies overall ($r = 2$, $M = 3$, $N = r^M = 2^3 = 8$). Stages have been pipelined to keep the system critical path short. The baseline 12 butterflies design then requires three clock periods for the outputs elaboration.

From the baseline 12 butterfly design, several variants have been derived through the decrease of the involved butterflies number. In such a way, the available resources of the design must be multiplexed in time and reused. Therefore, the overall computation latency increases and the throughput becomes lower. In particular, four size 8 FFT configurations are considered:

(i) 12b is the baseline 12 butterflies FFT design, taking 3 clock periods to finalize the transform.

(ii) 4b involves 4 butterflies for an overall execution latency of 6 cycles.

(iii) 2b involves 2 butterflies for an overall execution latency of 12 cycles.

(iv) 1b involves 1 single butterfly for an overall execution latency of 24 cycles.

4.1.2. FFT CGR System Implementation. The abovementioned configurations have been modelled as dataflow networks and the corresponding CGR system has been assembled with MDC. The activation percentage, resource utilization, and power consumption of each FFT variant are shown in Table 5. In general, the higher the number of butterflies is, the more the corresponding area and dissipation are.

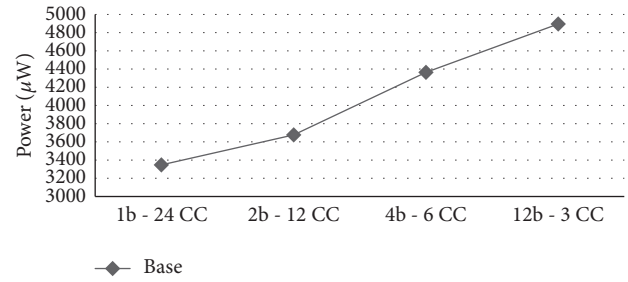

Figure 7: FFT use case: latency versus power consumption trade-off for the 4 different 8-size FFT configurations.

The main purpose of the resulting CGR system is to enable several trade-off levels between power dissipation and throughput, as illustrated in Figure 7. Such a system is capable of dynamically switching among the different configurations, fitting to external environment requests. For instance, in a battery operated environment, when the battery level becomes lower than a given threshold, some throughput can be waived to consume less power.

MDC identifies 8 different LRs in the CGR system. The LRs activated by each FFT variant are listed in Table 5, while their characteristics are reported in Table 6. In this table, given any LR, its activation time (T_{ON}) has been obtained summing up the activation times of the FFT configurations activating the same region (provided in Table 5). For example, LR_2 is activated by 1b, 2b, and 4b. Its T_{ON} is 0.67, which is the sum of $T_{ON}(1b) = 0.42$, $T_{ON}(2b) = 0.21$, and $T_{ON}(4b) = 0.04$.

4.1.3. Power Modelling and Hybrid Power Management Assessment. As explained in Section 3.2, the proposed flow requires

Table 5: FFT use case: features of the different configurations integrated on the CGR design. Data refer to a 90 nm CMOS target technology.

FFT configuration	T_{ON}	LRs	Area%	Static [nW]	Internal [nW]
1b	0.42	(2, 6, 8)	12.86	1750407	1307259
2b	0.21	(2, 5, 7, 8)	20.81	1752106	1539855
4b	0.04	(2, 3, 4, 7)	35.28	1757485	2020953
12b	0.33	(1, 3, 7, 8)	98.03	1776056	2411257

Table 6: FFT use case: logic regions architectural and functional characteristics.

	LR_1	LR_2	LR_3	LR_4	LR_5	LR_6	LR_7	LR_8
T_{ON}	0.33	0.67	0.37	0.04	0.21	0.42	0.58	0.96
#iso(e)	1024	1112	512	2324	1948	1756	256	5259
#rtn	1024	518	256	0	0	0	128	512
#reg	1024	518	256	0	0	0	128	512
#iso(r)	1024	1032	512	2306	1926	1740	256	4934

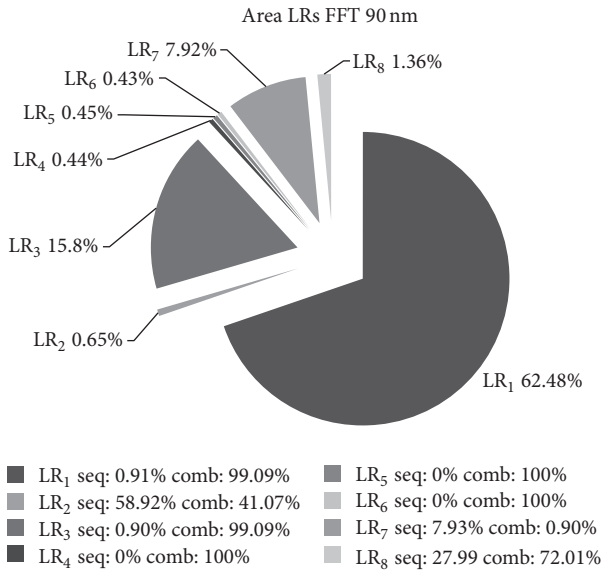

Figure 8: FFT use case: area percentage per LR.

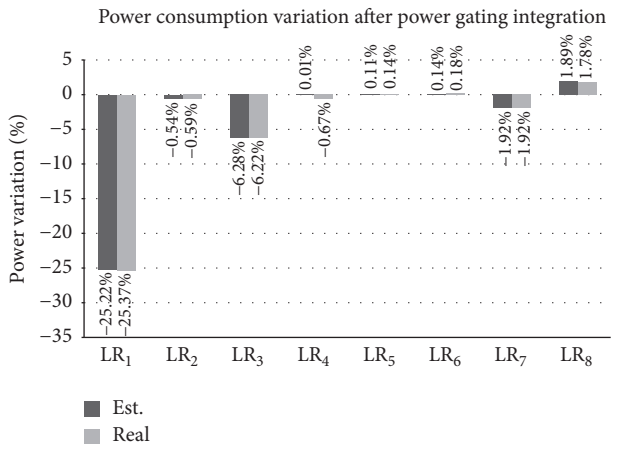

Figure 9: FFT use case: comparison between the estimated and real power variation due to the power gating integration.

a preliminary synthesis of the baseline (without any implemented power saving strategy) CGR system. From the synthesized design, it is possible to retrieve area occupancy and logic composition (combinatorial and sequential contributes) of the 8 LRs, as depicted in Figure 8. The area is given in terms of percentage with respect to the overall system area. The biggest region is LR_1; it occupies more than 60% of the whole system area. It is the region that mainly impacts power consumption. Furthermore, it is quite entirely combinatorial (99.18%), so that power gating should be a very suitable strategy for this LR. By Figure 8 it is possible to notice that LR_2, LR_4, LR_5, LR_6, and LR_8 are extremely small. For all these regions, the proposed power modelling strategy can be extremely beneficial to investigate if power saving techniques may lead or not to an effective power saving. Figure 8 also suggests that LR_4, LR_5, and LR_6 cannot benefit from clock gating, since they are fully combinatorial.

In order to evaluate the proposed power model, Figures 9 and 10 compare the estimated and measured (retrieved from the postsynthesis reports) overhead, respectively, due to power gating and clock gating. In both cases, the reported power refers to both the static and internal contributions, as taken into consideration by the power model. The remaining term, the net one, is neglected. The error of neglecting the net contribution is discussed in Section 4.1.4. The proposed power models are generally able to accurately approximate the power saving strategy overhead. As expected, LR_1 is the region with the highest power saving, regardless of the considered strategy (please note that a negative overhead implies a saving in power). It is interesting to notice that LR_8, despite being one of the smallest regions, does not provide any saving if power gated, but it can achieve a little power reduction when clock gated.

The static and internal power estimations obtained by applying (1) and (3), for a prospective power gating implementation, and (4) and (5), considering a possible clock gating implementation, are shown in Table 7. Please notice

Table 7: FFT use case: detailed static and dynamic saving due to power (PG) and clock gating (CG).

LR	PG overhead%		CG overhead%	
	Static	Internal	Static	Internal
LR_1	−40.87	−9.47	0.00	−11.33
LR_2	+0.23	−1.32	0.00	−2.85
LR_3	−10.44	−2.11	0.00	−2.59
LR_4	−0.08	0.10	—	—
LR_5	0.08	0.13	—	—
LR_6	0.11	0.17		
LR_7	−3.44	−0.39	0.00	−0.79
LR_8	1.42	2.35	0.00	−0.25

Table 8: FFT use case: *power gating overhead estimation* step accuracy.

LR	PG overhead%			Overhead error%			Net
	Est.	Real	Err.%	Iso	Rtn	Contr.	Err.%
LR_1	−25.22	−25.37	0.59	5.18	4.64	0.75	10.3
LR_3	−6.28	−6.22	1.07	16.34	2.97	0.76	12.6
LR_7	−1.92	−1.92	0.24	9.24	1.08	0.83	13.6

Table 9: FFT use case: *clock gating overhead estimation* step accuracy.

LR	CG overhead%			Overhead	Net
	Est.	Real	Err.%	CG_{cell} Err.%	Err.%
LR_1	−5.73	−5.77	0.06	0.83	0.18
LR_2	−1.42	−1.42	0.12	1.03	0.40
LR_3	−1.34	−1.38	0.05	0.84	0.21
LR_7	−0.39	−0.39	0.05	0.96	0.22
LR_8	−0.12	−0.12	0.29	1.08	0.47

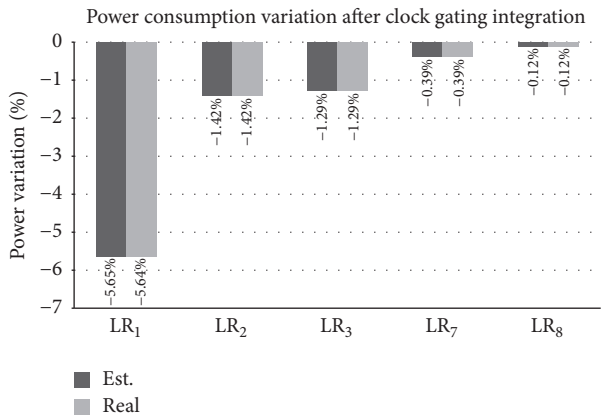

Figure 10: FFT use case: comparison between the estimated and real power variation due to the clock gating integration.

that clock gating static overhead is not appreciable, since one single clock gating cell is required per LR.

Algorithm 1 (see Section 3.2.6) implies a preliminary *area thresholding* step. Two different thresholds have been considered for the algorithm evaluation:

(i) DAT_5%: Threshold set to 5%. Regions with area above the 5% are LR_1, LR_3, and LR_7, so that the *power gating overhead estimation* step is performed for each of them. All the considered regions lead to an overall saving (negative overhead) larger than those achievable with a prospective clock gating implementation; therefore, they are selected as eligible regions for power gating. *Clock gating overhead estimation* is performed on all the remaining subthreshold regions. The regions capable of providing saving, when clock gated, are LR_2 and LR_8, since LR_4, LR_5, and LR_6 are fully combinatorial. Thus, clock gating will be implemented only on LR_2 and LR_8.

(ii) DAT_10%: Threshold set to 10%. Only LR_1 and LR_3 are above the area threshold and, as occurred also for DAT_5%, they both achieve power saving if implemented with power gating strategies. In this second case, the *clock gating overhead estimation* step is performed also on LR_7, which results in a negative overhead. Then, the regions to be clock gated are LR_2, LR_7, and LR_8, while LR_4, LR_5, and LR_6 are again discarded.

To access the proposed flow, five designs have been assembled:

(i) Base: the baseline CGR design without any power saving

(ii) PG_full: the CGR design, where power gating is applied blindly to all the regions

(iii) CG_full: the CGR design, where clock gating is applied blindly to all the regions

(iv) DAT_5%: derived with the proposed automated flow capable of hybrid power and clock gating support, setting the Area Threshold to 5% in Algorithm 1

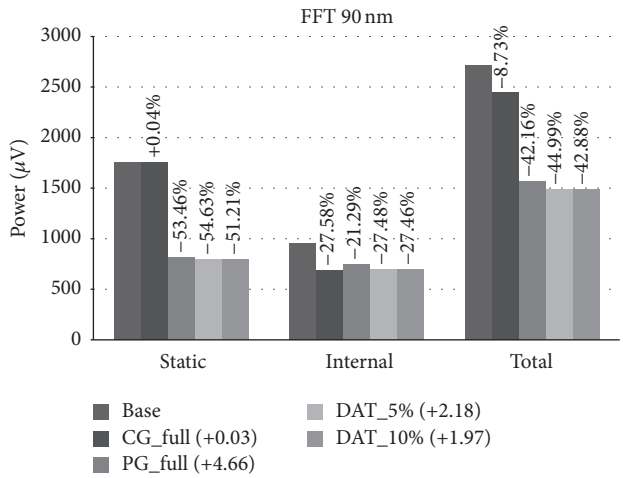

Figure 11: FFT use case: comparison between the base design and the four gated designs. Legend shows, in brackets, the power management area overhead for each design with respect to the Base one.

(v) DAT_10%: derived with the proposed automated flow capable of hybrid power and clock gating support, setting the Area Threshold to 10% in Algorithm 1

These designs have been synthesized with Cadence RTL Compiler, targeting the same 90 nm CMOS technology adopted to synthesise and simulate the baseline CGR design, whose results have fed the *Power Analysis* block of the proposed enhanced power management flow to assemble DAT_5% and DAT_10%.

The power consumption (internal, static, and total) of these designs is depicted in Figure 11. The reported data correspond to the real power consumed by the synthesised designs. Since LR_1 occupies more than the 60% of the design area and it is mainly combinatorial, little differences among the entirely power gated design (PG_full) and the hybrid clock and power gating ones (DAT_5% and DAT_10%) are visible. Nevertheless, DAT_5% achieves the largest power saving (−45.12%) among all the designs, validating the proposed hybrid and selective management with respect to a one-fit-to-all solution. CG_full, capable of diminishing only the dynamic power consumption, is the worst design among those applying power management.

The area overhead of the implemented power management strategies, reported in the legend of Figure 11, proves that the proposed hybrid management leads to less area hungry designs than the entirely power gated one. In fact, DAT_5% and DAT_10% present half of the area overhead of PG_full. CG_full data confirms that clock gating has a very little impact on the baseline design, presenting a negligible area overhead (two orders of magnitude smaller than DAT_5% and DAT_10%).

For the sake of completeness, in Figure 12 the trade-off levels between power and latency (and, in turn, throughput) are illustrated for all the considered designs. The trade-off curves demonstrate that power management strategies are generally extremely beneficial within a CGR scenario.

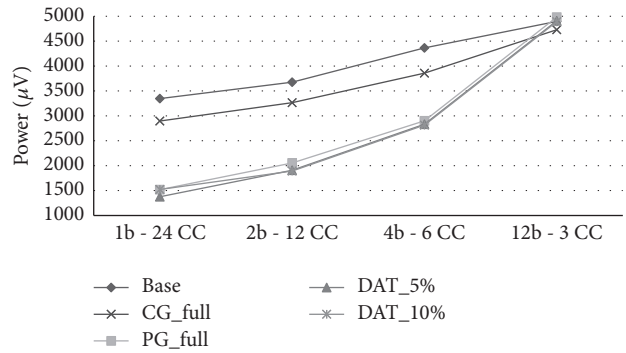

Figure 12: FFT use case: atency versus power consumption trade-off for the 4 different 8-size FFT configurations, when gated designs are adopted.

4.1.4. Accuracy and Errors. The accuracy of the proposed power modelling approach is assessed by Tables 8 and 9, respectively, considering the *power gating overhead estimation* step and the *clock gating overhead estimation* step of Algorithm 1. These tables, in each row, report the estimation errors with respect to the real consumption of the baseline CGR system, where the given power saving strategy (i.e., power gating in Table 8 and clock gating in Table 9) is applied only on the LR specified in the first column.

Looking at Table 8, power overhead estimations per LR demonstrate to be very accurate, leading to errors that are always below 1.1%. Also errors related to the estimation of state retention and Power Controller overhead are quite low (resp., below 5% and 1%). The isolation cells overhead estimation is less precise, resulting in an error of 16.36% for PD3, due to the fact that the static and internal values of $P(\mathrm{ISO}_{\mathrm{ON}})$ and $P(\mathrm{ISO}_{\mathrm{OFF}})$ are characterized as average values, the same for each LR. Nevertheless, this error has no visible impact on the total estimation one that is 1.07%.

Table 9 depicts an overview of the estimations accuracy for the clock gating overhead. In this case, estimations are even more accurate. The error on the clock gating overhead is always below the 0.3%. The error due to the clock gating cells overhead is very limited too, being always under the 1.1%.

Tables 8 and 9 report also the errors caused by omitting the power net term (column net % (Err.)) in (3) and (5). This error is obtained by comparing the estimated overhead (not comprehensive of the net contribute) with the real measured overhead comprehensive of the net term, as extracted by the synthesis reports. The net error is higher in the power gating overhead estimation (13.6% for LR_7) with respect to the clock gating one (at maximum 0.47% for LR_8), since power gating requires more additional logic to be implemented.

4.2. Validation Phase. In order to validate the proposed approach, a second use case has been assessed targeting the same 90 nm technology used for the FFT use case and a smaller 45 nm library. The reconfigurable computing core of an image coprocessing unit, accelerating a zoom application, has been assembled. Its characteristics are discussed in Section 4.2.1, while Sections 4.2.2 and 4.2.3 analyse the achieved results.

Table 10: Zoom coprocessor use case: computational kernels distinctive features.

Kernel	# actors	T_{ON}	Functionality	LRs
abs	1	0.03	Absolute value calculation	(4)
chgb	7	0.33	Bilevel/grayscale block checking	(5, 6, 7, 12, 13)
cubic	10	0.06	Linear combination calculation	(1, 5, 9, 10, 13)
cubic_conv	6	0.09	Cubic filter convolution	(5, 7, 8, 9)
median	9	0.06	Median calculation	(1, 3, 5, 11, 13)
min_max	1	0.01	Maximum/minimum finding	(11)
sbwlabel	17	0.42	Edge block checking	(2, 4, 5, 6, 13)

Table 11: Zoom coprocessor at 90 nm CMOS technology: characterization of the hybrid, clock, and power gated designs, achieved with the proposed automated flow. DAT_5%: area threshold 5%. DAT_10%: area threshold 10%. NA stands for not assigned and includes those LRs that placed in the always on domain.

Design	>Th	PG_set	CG_set	NA
DAT_5%	$LR_1, LR_2, LR_3, LR_6, LR_8, LR_{10}, LR_{12}, LR_{13}$	$LR_1, LR_2, LR_3, LR_8, LR_{10}, LR_{12}$	$LR_4, LR_6, LR_7, LR_9, LR_{11}, LR_{13}$	LR_5
DAT_10%	LR_1, LR_8	LR_1, LR_8	$LR_2, LR_3, LR_4, LR_6, LR_7, LR_9, LR_{10}, LR_{11}, LR_{12}, LR_{13}$	LR_5

4.2.1. CGR System of the Zoom Coprocessor. The zoom application is meant to scale an image according to the given zooming factor. Missing pixels of the zoomed image are calculated by adaptively interpolating the neighbouring values. The original application is a C program. It has been executed and profiled on a general purpose machine in order to identify the most computational intensive portions of the code (computational kernels) with the intention of accelerating them on a CGR hardware accelerator. Seven computational kernels have been identified and modelled as dataflow networks. Table 10 summarizes the kernels composition (in terms of number of dataflow actors), activation time, and main functionality. Then, these dataflow kernels have been combined by MDC to obtain a multidataflow specification, constituting the computing core of the CGR accelerator in charge of accelerating the zoom application.

Thirteen LRs are identified on the CGR zoom coprocessor. The main difference between this scenario and the FFT one is that in the zoom coprocessor it is not necessary to retain the status of any kernel when switching among them. This means that, applying power gating, no retention cells are needed in the identified regions.

4.2.2. Zoom Coprocessor Validation Results at 90 nm CMOS Technology. This section is meant to provide the discussion of the achieved results in the zoom coprocessor scenario using the same 90 nm CMOS technology adopted for the FFT CGR designs assessment. From the implementation point of view, we will discuss the same designs we considered for the FFT use case, defined as follows:

(i) Base: the baseline CGR design without any power saving

(ii) PG_full: the CGR design, where all the 13 LRs are power gated

(iii) CG_full: the CGR design, where all the 13 LRs are clock gated

(iv) DAT_5%: the CGR design, where hybrid power and clock gating support are implemented by means of the proposed flow, setting the Area Threshold to 5% in Algorithm 1

(v) DAT_10%: the CGR design, where hybrid power and clock gating support are implemented by means of the proposed flow, setting the Area Threshold to 10% in Algorithm 1

Please refer to Table 11 for the composition of DAT_5% and DAT_10%.

Figure 13 depicts static, internal, and total power consumption for each considered design. In this case, the differences among CG_full, PG_full, DAT_5%, and DAT_10% are not so evident. The reason is that, in this scenario, the dynamic power consumption (due to the internal power) is considerably higher than the static one. As you can see in the reported histograms, on average, there are approximately more than two orders of magnitude of difference. Power gating and clock gating demonstrate to be equally capable of cutting down the internal power consumption.LR_5 is the only region that Algorithm 1 completely discards by any form of power management, both in the DAT_5% design and in the DAT_10% one. It is fully combinatorial; therefore, clock gating does not provide any positive effect on it. Nevertheless, it is so small (0.65% of the whole system area) that, if power gated, it cannot provide any substantial benefit. A closer observation of the histograms confirms what we already got for the FFT: despite the similar trend for all the designs, which lead to more than the 62% of power saving, DAT_5% consumes less than any other (62.61% of saving), while the CG_full design is the less beneficial (62.29% of saving).

Focusing on the static histograms, as expected, the CG_full design introduces a small overhead with respect to

Table 12: Zoom coprocessor at 90 nm CMOS technology: *power gating overhead estimation* step and clock gating overhead estimation step accuracy.

LR	PG saving%			Net	CG saving%			Net
	Est.	Real	Err.%	Err.%	Est.	Real	Err.%	Err.%
LR_1	−6.322	−6.331	0.15	0.18	−6.307	−6.313	0.09	0.17
LR_2	−23.464	−23.463	0.00	1.26	−23.369	−23.373	0.02	1.67
LR_3	−6.537	−6.544	0.10	1.65	−6.507	−6.514	0.10	1.59
LR_4	—	—	—	—	−0.958	−0.964	0.68	1.34
LR_5	—	—	—	—	—	—	—	—
LR_6	—	—	—	—	−0.870	−0.878	0.81	1.22
LR_7	—	—	—	—	−1.033	−1.039	0.63	0.15
LR_8	−7.062	−7.058	0.05	2.02	−6.987	−6.986	0.01	1.83
LR_9	—	—	—	—	−2.436	−2.443	0.27	1.55
LR_{10}	−5.498	−5.503	0.10	1.65	−5.474	−5.480	0.11	1.57
LR_{11}	—	—	—	—	−3.329	−3.285	1.32	3.67
LR_{12}	−4.278	−4.288	0.23	1.83	−4.267	−4.273	0.15	1.86
LR_{13}	—	—	—	—	−0.649	−0.656	1.01	1.13

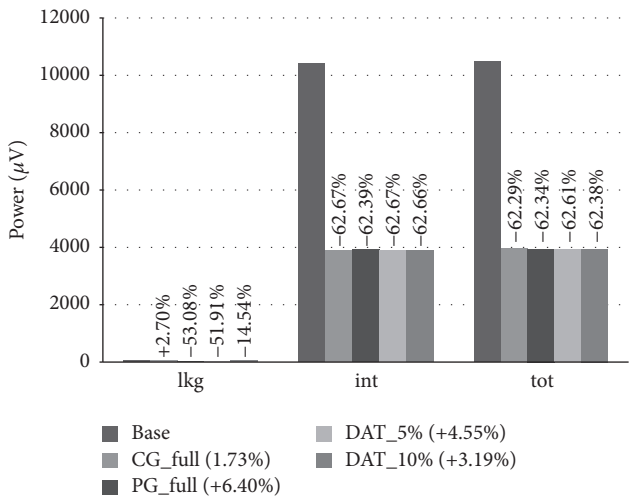

Figure 13: Zoom coprocessor at 90 nm CMOS technology: comparison between the base design and the four gated designs. Legend shows, in brackets, the power management area overhead for each design with respect to the Base one.

base. That is due to the 12 (one for each region but LR_5) clock gating cells introduced in the always on domain of this design, which never contribute to save any static power consumption. When power gating is applied, there is always a benefit in terms of static power consumption: DAT_5% saving is slightly higher than the PG_full one, both being over 51%; DAT_10% is still beneficial, but its saving is limited to the 15%. Please note that the difference between DAT_5% and DAT_10% (in terms of static consumption) demonstrates that, in the *area thresholding* step of the proposed Algorithm 1, it is better to opt for small area threshold values to achieve higher saving results.

In terms of area occupancy, reported in the legend of Figure 13, the PG_full design is the one with the largest overhead, +6.4%. DAT_5%, which is the most beneficial in terms of power, shows a slightly smaller overall overhead, +4.55% of the whole system area. The CG_full is again the less invasive one, leading just to +1.73% of area overhead. Summarizing, DAT_5% constitutes the optimal solution for the zoom coprocessor scenario, considering a 90 nm technology. DAT_10%, which is less beneficial than DAT_5% in saving static power consumption, is a better solution than a fully power gated design, presenting basically the same power saving (−62.38% for DAT_10% versus −62.29% for PG_full) but a smaller area overhead (+3.19% for DAT_10% versus +6.4% for PG_full).

Table 12 reports the estimation error of the proposed automated hybrid power management design flow, when the power saving percentages for the considered domains, respectively, considering power gating (*power gating overhead estimation*) and clock gating (*clock gating overhead estimation*), are evaluated. *PG saving*% errors are always below 0.3% and *CG saving*% ones do not exceed 1.5%. These data confirm the accuracy of the proposed models, as in the FFT use case. Table 12, for both power and clock gating implementations, depicts also the error of neglecting the net term in the dynamic power consumption. Again, as in the previously discussed scenario, models are not affected by this simplification.

4.2.3. Zoom Coprocessor Validation Results at 45 nm CMOS Technology. In order to provide a robust validation of the proposed approach, we decided to assess the same zoom coprocessor designs on a different technology targeting a 45 nm CMOS technology. The idea is assessing what changes when the ratio between the static and the dynamic consumption is varied. Here follows the list of the implemented designs:

(i) Base: the same as in the 90 nm synthesis trial

(ii) PG_full: the same as in the 90 nm synthesis trial

(iii) CG_full: the same as in the 90 nm synthesis trial

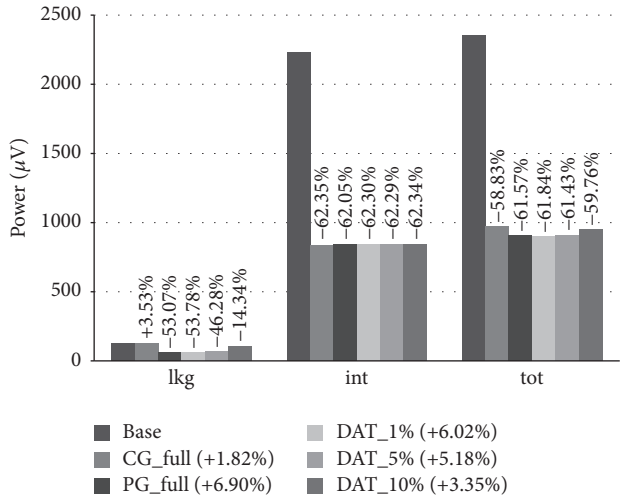

Figure 14: Zoom Co-Processor at 45 nm CMOS technology: Comparison between the base design and the five gated designs. Legend shows, in brackets, the power management area overhead for each design with respect to the Base one.

(iv) DAT_1%: the CGR design, where hybrid power and clock gating support are implemented by means of the proposed flow, setting the Area Threshold to 1% in Algorithm 1

(v) DAT_5%: the same as in the 90 nm synthesis trial

(vi) DAT_10%: the same as in the 90 nm synthesis trial

Targeting a smaller technology and having already established that the 10% area threshold leads to power results comparable to those of the fully power gated design, we have decided to add in this second trial an additional design: DAT_1%. Setting the area threshold to 1%, quite all the LRs are considered for a prospective power gating implementation. Please refer to Table 13 for the composition of DAT_1%, DAT_5%, and DAT_10%.

The power consumption in terms of static, internal, and total contributions is illustrated in Figure 14. The dynamic power consumption is still higher than the static one, determining the overall trend of the total power. However, with the scaling of the channel length, the ratio among internal and static power on average has decreased from a factor of 100 to approximately 10. In this second trial, the influence of the static power consumption is partially reflected on the total one. Technology scaling and the different static versus dynamic power ratio are such that PG_full is capable of providing better overall saving results than DAT_5% and DAT_10%. At 45 nm technology, designers are required to select a very low area threshold in Algorithm 1 to achieve really optimal results. DAT_1%, which basically excludes from power gating only the 3 LRs, saves up to 61.84% of total base power, and represents the optimal design solution for the zoom coprocessor in this second synthesis trial. Please note also that, lowering down the area threshold, the area of the optimal design and that of the fully power gated one are pretty similar.

We can conclude that as technology gets smaller the *area thresholding* step of the proposed algorithm is less beneficial still, in the automated flow, its presence makes the overall process more robust, avoiding useless iterations on not convenient by construction designs when the technology are not so constrained or the ratio among static and dynamic consumption is larger.

The accuracy of the proposed models, targeting the 45 nm CMOS technology, is reported in Table 14, which contain both power gating and clock gating estimation errors. The models, even neglecting the net contribution in the discussed equations, are extremely accurate (the error never exceeds 3.70%).

4.3. Power Switch Overhead. The sleep transistors are inserted in the design during the place-and-route process and their overhead is strictly use case dependent since it is related to the aspect ratio of the macro and to the style of power routing that is selected in the target design. Since the proposed power estimation model is based on the synthesis of the design, the contribution of these cells is not considered yet.

The insertion of header/footer switches (we used header ones in this work as reference) determines two kinds of power overhead: (1) a leakage-related overhead; (2) the dynamic power dissipated during sleep and wake-up transition. Another overhead that has to be taken into consideration is the time necessary to wake up the power domain. For the proper operation of the power gating methodology, the gating logic has to be enabled/disabled according to a switch on/off protocol [11] that requires 4 clock cycles for each transition. Thus, when a kernel is off, at least 4 clock cycles are necessary before it is switched on and then it can receive new input data.

The leakage-related contribution is fixed by the technology library and it is always present regardless of the ON/OFF state of the power domain. It could be inserted in (1) as $P_{\text{lkg}}(\text{SW}) * \#\text{switches}$ where $P_{\text{lkg}}(\text{SW})$ is the static power consumption of the considered power switch, as reported in the technology library, and #switches is the number of power switches inserted in the power domain. In the library that we took as reference, a header switch has the same leakage power of a 64-bit wide register, and one column of N switches can be used to switch down N horizontal virtual power stripes of the power grid. If we assume to use only one vertical real power stripe to supply power to N horizontal virtual power stripes of a power domain, only one column of N power switches is inserted. In this case, gating the virtual power supply easily saves enough leakage power to counterbalance the leakage dissipation of the inserted switch.

The dynamic power contribution is only relevant when intervals between successive kernel switches are in the order of tens of cycles (Hu et al. [46]). When the computation of the kernels last tens of cycles also the wake-up time is not relevant. Thus, in designs with low switching rates, these two overhead contributions could be neglected.

The FFT use case is a really simple design, used only for the development of the power estimation model and, as reported in Section 4.1, its kernels are far from lasting tens of cycles. The zoom application adopted for the validation

Table 13: Zoom coprocessor at 45 nm CMOS technology: characterization of the hybrid, clock, and power gated designs, achieved with the proposed automated flow. DAT 1%: area threshold 1%. DAT 5%: area threshold 5%. DAT_10%: area threshold 10%. NA stands for not assigned and includes those LRs that are placed in the always on domain.

Design	>Th	PG_set	CG_set	NA
DAT_1%	$LR_1, LR_2, LR_3, LR_4, LR_6, LR_7, LR_8, LR_9, LR_{10}, LR_{11}, LR_{12}, LR_{13}$	$LR_1, LR_2, LR_3, LR_4, LR_7, LR_8, LR_9, LR_{10}, LR_{11}, LR_{12}$	LR_6, LR_{13}	LR_5
DAT_5%	$LR_1, LR_2, LR_3, LR_6, LR_8, LR_{10}, LR_{12}, LR_{13}$	$LR_1, LR_2, LR_3, LR_8, LR_{10}, LR_{12}$	$LR_4, LR_6, LR_7, LR_9, LR_{11}, LR_{13}$	LR_5
DAT_10%	LR_1, LR_8	LR_1, LR_8	$LR_2, LR_3, LR_4, LR_6, LR_7, LR_9, LR_{10}, LR_{11}, LR_{12}, LR_{13}$	LR_5

Table 14: Zoom coprocessor at 45 nm CMOS technology: *power gating overhead estimation* step and *clock gating overhead estimation* step accuracy.

LR	PG saving%			Net	CG saving%			Net
	Est.	Real	Err.%	Err.%	Est.	Real	Err.%	Err.%
LR_1	−5.686	−5.699	0.23	0.47	−5.507	−5.501	0.13	0.53
LR_2	−22.990	−22.982	0.03	1.02	−22.063	−22.029	0.15	0.81
LR_3	−6.569	−6.566	0.04	1.13	−6.273	−6.262	0.18	0.91
LR_4	−0.946	−0.944	0.15	1.82	−0.919	−0.917	0.22	1.52
LR_5	—	—	—	—	—	—	—	—
LR_6	−0.747	−0.748	0.25	0.25	−0.853	−0.833	0.26	1.59
LR_7	−0.955	−0.957	0.17	1.36	−0.935	−0.933	0.21	1.49
LR_8	−7.464	−7.460	0.06	1.36	−6.724	−6.714	0.16	0.66
LR_9	−2.475	−2.482	0.25	1.10	−2.350	−2.345	0.18	1.09
LR_{10}	−5.473	−5.470	0.04	1.09	−5.270	−5.261	0.17	0.91
LR_{11}	−3.422	−3.361	1.79	2.58	−3.257	−3.205	1.63	2.04
LR_{12}	−4.327	−4.330	0.08	1.19	−4.176	−4.169	0.17	0.98
LR_{13}	−0.587	−0.580	1.25	3.70	−0.623	−0.621	0.28	1.85

phase of the proposed model is a real use case but it is a small size design, where the execution of the fastest kernels lasts 24 clock cycles. Then, the wake-up time overhead is, in the worst case, almost 17% of the total execution time. If we consider a bigger and more complex real use case, such as interpolation filtering for motion compensation in High Efficiency Video Coding [47], we would achieve the condition for neglecting the dynamic power consumption of the sleep transistors and the wake-up time overhead. This application involves 2-dimensional filters working on subblocks of pictures belonging to the same video sequence. The smallest block, corresponding to the fastest execution time, has 8×8 pixels. In this case, 160 overall cycles are required to filter the whole block and the wake-up time overhead is just 2.5% of the total execution time.

Table 15: FFT use case at 90 nm CMOS technology: *power gating overhead estimation* step accuracy, using reports generated without the real switching activity.

LR	PG overhead%		
	Est.	Real	Err.%
LR_1	−30.54	−26.10	17.07
LR_2	−0.19	−1,99	90.22
LR_3	−21.83	−6.95	214.30
LR_4	−0.12	−0.67	80.97
LR_5	−0.06	−0.59	90.11
LR_6	−0.07	0.56	86.60
LR_7	−15.39	−2.64	482.93
LR_8	−0.38	−0.02	1941.81

4.4. Advantages of the Proposed Approach. Considering a CGR system implementing N different functionalities and partitioned into k different LRs, the proposed selection algorithm, based on the power models embodied in (1), (3), (4), and (5), requires the synthesis of the baseline CGR design (without any power saving strategy applied) and N hardware simulations, each one running a different functionality (i.e., executing the different DPNs provided as input to the MDC tool). The hardware simulations are needed since the real switching activity is essential for correct dynamic power estimation. Table 15, targeting the FFT scenario and the power gating implementation, depicts the estimated and real power overhead when estimations are performed adopting the default synthesis reports (without taking into account the real switching activity). The estimation errors are extremely high when the switching activity is neglected; therefore, the proposed models are not capable of properly determining which LRs would actually benefit of power gating.

In order to understand the advantages of the proposed approach, let us compute the effort needed to determine the optimal saving strategy for each region if our flow is not adopted. It is required to

(1) synthesize the baseline design without any power management support;

(2) synthesize one power gated design and one clock gated design for each LR;

(3) perform N different hardware simulations for the baseline design, to retrieve the real switching activity of the system;

(4) perform N different hardware simulations for each power gated and clock gated design, to retrieve their real switching activity;

(5) compare each power gated design and clock gated design, in the different operating conditions, with respect to the synthesized baseline CGR design.

Our flow requires only point 1 and point 3. In numbers it corresponds to one single synthesis and N hardware simulations. On the contrary, not using our approach, $2 * k + 1$ synthesis

(k for power gating evaluation, k for clock gating evaluation plus the baseline one) and $N*(2*k+1)$ hardware simulations are necessary. The only simplification that may be done, even without adopting the proposed approach, is when a given region is fully combinatorial. This would save the effort related to its perspective clock gating evaluation.

Dealing with the presented use cases, for the FFT (Section 4.1) there are $N = 4$ different functionalities and $k = 8$ LRs. Among these latter 3 are fully combinatorial. The proposed approach required one synthesis and 4 hardware simulations, rather than 14 synthesis (8 power gated LRs, 5 clock gated LRs and the baseline design) and 56 hardware simulations (4 for each synthesized design). Considering the zoom coprocessor (Section 4.2), $N = 7$ and $k = 13$, with only 1 fully combinatorial LR. The proposed approach required one synthesis and 7 hardware simulations, rather than 26 synthesis (13 power gated, 12 clock gated and the baseline designs) and 182 hardware simulations.

5. Conclusions

In this paper, we addressed the problem of power management in coarse-grained reconfigurable (CGR) systems. Such systems are as suitable to accelerate multifunctional applications as, potentially, energy inefficient. In fact, on a CGR substrate, while a particular task is executed, the resources not involved in the computation may potentially waste precious power if not properly managed. On top of that, these architectures are also characterized by an intrinsic design difficulty: mapping several applications on the same substrate, customizing the datapath, is not so straightforward and requires a deep knowledge of the target applications. Dataflow models of computation turned out to be very efficient for the development of automated mapping problems. In our studies, we have exploited a dataflow-based approach to define a complete design suite for multifunctional CGR systems: the Multi-Dataflow Composer (MDC) tool. Besides automatically managing dataflow to hardware systems composition, MDC also supports the automated characterization of power and clock gated platforms.

The proposed work makes some steps further both in the MPEG-RVC field, which the MDC tool belongs to, and in the definition of optimal power management strategies for CGR designs. In this paper, we have presented an automated methodology capable of estimating, prior to any physical implementation, the effectiveness and costs that power gating or clock gating would have when implemented on top of the functional logic regions constituting a CGR system. This methodology is based on static and dynamic power estimation models that, in a separate manner for each logic region in the CGR design, are capable of determining the overhead of clock gating and power gating on the basis of the functional, technological, and architectural parameters of the baseline CGR system. These models and the corresponding estimation algorithm are applicable in any CGR scenario and are currently integrated in the MDC tool, improving its basic functionality. In fact, MDC was normally applying a one-fit-to-all user-specified power reduction technique, either clock or power gating, without any warranty of its effectiveness on the different identified regions.

By considering two different scenarios and adopting different ASIC technologies, our assessments proved that the enhanced MDC flow is capable of guiding the designers towards the definition of the optimal power management support. It is more efficient than the previous, blindly applied, methodology and the proposed models turned out to be extremely accurate. Finally, as demonstrated in Section 4.4, the new flow drastically reduces the number of designs to be synthesized and simulated, leading to saving both designer effort and computational time.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

Tiziana Fanni is grateful to Sardinia Regional Government for supporting her Ph.D. scholarship (P.O.R. F.S.E., European Social Fund 2007–2013, Axis IV Human Resources).

References

[1] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '01)*, pp. 642–649, March 2001.

[2] P. Meloni, G. Tuveri, L. Raffo et al., "System adaptivity and fault-tolerance in NoC-based MPSoCs: the MADNESS project approach," in *Proceedings of the 15th Euromicro Conference on Digital System Design (DSD '12)*, pp. 517–524, 2012.

[3] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the International Symposium on Computer Architecture*, pp. 365–376, San Jose, Calif, USA, 2011.

[4] M. B. Taylor, "Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proceedings of the 49th Annual Design Automation Conference (DAC '12)*, pp. 1131–1136, San Francisco, Calif, USA, June 2012.

[5] F. Oboril, J. Ewert, and M. B. Tahoori, "High-resolution online power monitoring for modern microprocessors," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 265–268, March 2015.

[6] Synopsys, "Advanced low power techniques," http://goo.gl/2FqEjf.

[7] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '07)*, pp. 38–43, August 2007.

[8] S. Eyerman and L. Eeckhout, "Fine-grained DVFS using on-chip regulators," *Transactions on Architecture and Code Optimization*, vol. 8, no. 1, article 1, 2011.

[9] M. Arora, S. Manne, Y. Eckert, I. Paul, N. Jayasena, and D. M. Tullsen, "A comparison of core power gating strategies implemented in modern hardware," in *Proceedings of the Conference on Measurement and Modeling of Computer Systems*, pp. 559–560, 2014.

[10] B. Jeff, "Advances in big.LITTLE technology for power and energy savings," ARM White Paper, 2012.

[11] Power Forward Initiative, *A Practical Guide to Low Power Design*, 2009.

[12] F. Palumbo, N. Carta, D. Pani, P. Meloni, and L. Raffo, "The multi-dataflow composer tool: generation of on-the-fly reconfigurable platforms," *Journal of Real-Time Image Processing*, vol. 9, no. 1, pp. 233–249, 2014.

[13] N. Carta, C. Sau, F. Palumbo, D. Pani, and L. Raffo, "A coarse-grained reconfigurable wavelet denoiser exploiting the multi-dataflow composer tool," in *Proceedings of the 7th Conference on Design and Architectures for Signal and Image Processing (DASIP '13)*, pp. 141–148, October 2013.

[14] N. Carta, C. Sau, D. Pani, F. Palumbo, and L. Raffo, "A coarse-grained reconfigurable approach for low-power spike sorting architectures," in *Proceedings of the 6th International IEEE EMBS Conference on Neural Engineering (NER '13)*, pp. 439–442, San Diego, Calif, USA, November 2013.

[15] D. Pani, F. Usai, L. Citi, and L. Raffo, "Real-time processing of tfLIFE neural signals on embedded DSP platforms: a case study," in *Proceedings of the 5th International IEEE/EMBS Conference on Neural Engineering (NER '11)*, pp. 44–47, Cancun, Mexico, April 2011.

[16] N. Carta, P. Meloni, G. Tuveri, D. Pani, and L. Raffo, "A custom MPSoC architecture with integrated power management for real-time neural signal decoding," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 2, pp. 230–241, 2014.

[17] F. Palumbo, C. Sau, and L. Raffo, "Coarse-grained reconfiguration: dataflow-based power management," *IET Computers and Digital Techniques*, vol. 9, no. 1, pp. 36–48, 2015.

[18] F. Palumbo, T. Fanni, C. Sau, and P. Meloni, "Power-awarness in coarse-grained reconfigurable multi-functional architectures: a dataflow based strategy," *Journal of Signal Processing Systems*, 2016.

[19] D. Pani and L. Raffo, "Self-coordinated on-chip parallel computing: a swarm intelligence approach," in *Parallel and Distributed Computational Intelligence*, pp. 91–112, Springer, Berlin, Germany, 2010.

[20] M. Yan, Z. Yang, L. Liu, and S. Li, "ProDFA: accelerating domain applications with a coarse-grained runtime reconfigurable architecture," in *Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS '12)*, pp. 834–839, December 2012.

[21] S. M. Carta, D. Pani, and L. Raffo, "Reconfigurable coprocessor for multimedia application domain," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 44, no. 1-2, pp. 135–152, 2006.

[22] V. V. Kumar and J. Lach, "Highly flexible multimode digital signal processing systems using adaptable components and controllers," *EURASIP Journal on Applied Signal Processing*, vol. 2006, no. 1, Article ID 079595, pp. 1–9, 2006.

[23] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 296–301, March 2003.

[24] F. Palumbo, D. Pani, L. Raffo, and S. Secchi, "A surface tension and coalescence model for dynamic distributed resources allocation in massively parallel processors on-chip," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, vol. 129 of *Studies in Computational Intelligence*, pp. 335–345, Springer, Berlin, Germany, 2007.

[25] G. Ansaloni, K. Tanimura, L. Pozzi, and N. Dutt, "Integrated kernel partitioning and scheduling for coarse-grained reconfigurable arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 12, pp. 1803–1816, 2012.

[26] C. C. de Souza, A. M. Lima, G. Araujo, and N. B. Moreano, "The datapath merging problem in reconfigurable systems: complexity, dual bounds and heuristic evaluation," *ACM Journal of Experimental Algorithmics*, vol. 10, no. 2.2, Article ID 1180613, 2005.

[27] R. Giorgi and A. Scionti, "A scalable thread scheduling co-processor based on data-flow principles," *Future Generation Computer Systems*, vol. 53, pp. 100–108, 2015.

[28] L. Verdoscia, R. Vaccaro, and R. Giorgi, "A clockless computing system based on the static dataflow paradigm," in *Proceedings of the 4th Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM '14)*, pp. 30–37, Edmonton, Canada, August 2014.

[29] C. Sau, L. Fanni, P. Meloni, L. Raffo, and F. Palumbo, "Reconfigurable coprocessors synthesis in the MPEG-RVC domain," in *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig '15)*, pp. 1–8, IEEE, Riviera Maya, Mexico, December 2015.

[30] L. Jozwiak, M. Lindwer, R. Corvino et al., "ASAM: automatic architecture synthesis and application mapping," in *Proceedings of the 15th Euromicro Conference on Digital System Design (DSD '12)*, pp. 216–225, September 2012.

[31] L. Jozwiak, M. Lindwer, R. Corvino et al., "ASAM: automatic architecture synthesis and application mapping," *Microprocessors and Microsystems*, vol. 37, no. 8, pp. 1002–1019, 2013.

[32] Y. Zhang, J. Roivainen, and A. Mammela, "Clock-gating in FPGAs: a novel and comparative evaluation," in *Proceedings of the 9th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, p. 590, 584, Dubrovnik, Croatia, August-September 2006.

[33] E. Bezati, S. Casale-Brunet, M. Mattavelli, and J. W. Janneck, "Coarse grain clock gating of streaming applications in programmable logic implementations," in *Proceedings of the Electronic System Level Synthesis Conference*, pp. 1–6, 2014.

[34] Silicon Integration Initiative, Si2 Common Power Format SpecificationTM-Version 2.1, 2014.

[35] M. Shafique, L. Bauer, and J. Henkel, "Adaptive energy management for dynamically reconfigurable processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 50–63, 2014.

[36] J. Yi and J. Kim, "Power modeling for digital circuits with clock gating," *IEICE Electronics Express*, vol. 12, no. 24, Article ID 20150817, 2015.

[37] H. Xu, R. Vemuri, and W.-B. Jone, "Run-time active leakage reduction by power gating and reverse body biasing: an energy view," in *Proceedings of the 26th IEEE International Conference on Computer Design (ICCD '08)*, pp. 618–625, IEEE, October 2008.

[38] K. Datta, A. Mukherjee, G. Cao et al., "CASPER: embedding power estimation and hardware-controlled power management in a cycle-accurate micro-architecture simulation platform for many-core multi-threading heterogeneous processors," *Journal of Low Power Electronics and Applications*, vol. 2, no. 1, pp. 30–68, 2012.

[39] A. Chhabra, H. Rawat, M. Jain et al., "FALPEM: framework for architectural-level power estimation and optimization for large memory sub-systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1138–1142, 2015.

[40] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT framework for multicore and many-core architectures: simultaneously modeling power, area, and timing," *ACM Transactions on Architecture and Code Optimization*, vol. 10, no. 1, article 5, pp. 1–29, 2013.

[41] D. Zoni and W. Fornaciari, "Modeling DVFS and power-gating actuators for cycle-accurate NoC-based simulators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 12, no. 3, article 27, pp. 1–24, 2015.

[42] C. Sau, L. Raffo, F. Palumbo, E. Bezati, S. Casale-Brunet, and M. Mattavelli, "Automated design flow for coarse-grained reconfigurable platforms: an RVC-CAL multi-standard decoder use-case," in *Proceedings of the 14th International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS '14)*, pp. 59–66, Agios Konstantinos, Greece, July 2014.

[43] Open RVC-CAL Compiler, http://orcc.sourceforge.net/.

[44] Cadence, Low Power in Encounter RRTL Compiler Product Version 14.1, July 2014.

[45] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[46] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '04)*, pp. 32–37, IEEE, August 2004.

[47] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel, "A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 238–251, 2015.