J Sign Process Syst (2017) 87:81–106
DOI 10.1007/s11265-016-1106-9

Power-Awarness in Coarse-Grained Reconfigurable Multi-Functional Architectures: a Dataflow Based Strategy

Francesca Palumbo[1] · Tiziana Fanni[2] · Carlo Sau[2] · Paolo Meloni[2]

Received: 1 March 2015 / Revised: 5 January 2016 / Accepted: 14 January 2016 / Published online: 9 February 2016

Abstract Modern embedded systems, to accommodate different applications or functionalities over the same substrate and provide flexibility at the hardware level, are often resource redundant and, consequently, power hungry. Therefore, dedicated design frameworks are required to implement efficient runtime reconfigurable platforms. Such frameworks, to challenge this scenario, need also to offer application specific support for power management. In this work, we adopt dataflow specifications as a starting point to feature power minimization in coarse-grained reconfigurable embedded systems. The proposed flow is composed of two subsequent steps: 1) the characterization of the optimal topological system specification(s) and 2) the identification of disjointed logic regions. These latter are then used to implement clock and power gating methodologies. The validity of this model-based approach has been proved over the reconfigurable computing core of a multi-functional coprocessor for image processing applications. Results have been assessed targeting both an ASIC 90 nm technology and a 45 nm one.

Keywords Power management · Coarse-grained reconfiguration · Dataflow · Power gating · Clock gating · MPEG-RVC · 90 nm CMOS · 45 nm CMOS · Common Power Format

✉ Francesca Palumbo
fpalumbo@uniss.it

Tiziana Fanni
Tiziana.Fanni@diee.unica.it

Carlo Sau
Carlo.Sau@diee.unica.it

Paolo Meloni
Paolo.Meloni@diee.unica.it

[1] POLCOMING - Information Engineering Unit, University of Sassari, Sassari, Italy

[2] DIEE - Department of Electronics Engineering, University of Cagliari, Cagliari, Italy

1 Introduction

From portable consumer electronics to wearable medical devices, power reduction is one of the biggest challenges in the development of modern systems. We are in the dark silicon era: a gap exists between the number of transistors that can be placed within a die and the number that can be actually driven during execution [1, 2], due to the limited available power budget. This metric, unfortunately, collides with the possibility of embedding fancy resource-intensive applications on complex heterogeneous systems, increasing the effort needed by the designer to obtain effective solutions. This issue poses the need for cross-layer design approaches, considering power-related aspects from the application to the physical levels. Several techniques for power monitoring and reduction have been presented at the state of the art. Prior works investigated voltage/frequency scaling [3, 4] and power shut-off schemes [5, 6]. Their implementation requires the manual intervention of the designers in different steps of the design flow that becomes, in turn, quite complex, error prone and time consuming. While commercial synthesizers [7] allow, for example, the automatic implementation of fine-grained clock-gating strategies, they only provide implementation-level instruments to apply power gating techniques. Without a complete automatic flow, designers need to manually identify the regions

to shut down and to manually manage the power logic insertion and control. Nevertheless, many commercial devices already embed several different power domains. According to a recent survey by SONICS, more than 300 customers expect half of their products on the market [8] to be partitioned in approximately five power domains. Thus, reducing the manual effort and implementation risks related to the design process, it would result in significant benefits for industrial design cycles. More specifically, when considering coarse-grained power gating methodologies, the most critical aspect is deciding the resource set to be inserted within a specific domain. This issue limits the extensive use of such technique in application specific scenarios. In communication networks design, SONICS exploited many of traditional techniques for power management (from clock gating to voltage scaling etc.) to deploy an efficient interconnect [8]. To boost application performance, while preserving power efficiency, the *ARM big.LITTLE* processor features two sets of processors optimized for different purposes that may be alternatively shut-off [6] when not used. However, to foster the adoption of energy management techniques, efficient automated strategies are still required, in order to map the resources over different operating regions on the die, locally reducing the power consumption and allowing a better usage of the available power budget.

Our work focuses on automated strategies for local power reduction. We have developed a design framework for coarse-grained reconfigurable platforms [9], the Multi-Dataflow Composer (MDC) tool [10], aiding the designers to assemble efficient application-specific multi-functional accelerators [11, 12]. The foundations of this work are related to the MPEG Reconfigurable Video Coding (MPEG-RVC) standard [13, 14] that adopts dataflow-based techniques to challenge codec design complexity. MPEG-RVC exploits the intrinsic modularity of the dataflow models to enable dynamic and incremental configuration and reconfiguration of codec descriptions. Such an approach is extremely useful to tackle design complexity, allowing flexible optimization and debugging of the different parts of the system. The methodology we have developed to cope with power management is coupled with the MDC tool. MDC combines the dataflow formalism with the coarse-grained reconfigurable approach to system design for deploying multi-functional systems, featuring flexibility and area minimization. Within the context of the MPEG-RVC framework power management has been just partially addressed, as we will discuss in Section 2.

This paper presents the combination of structural and dynamic strategies to master power reduction in coarse-grained reconfigurable architectures, extending the works presented in [15, 16]. With respect to the previous works, the overall approach has been maintained. At the structural level, the optimal system specification(s) (capable of maximizing performance while minimizing the implementation costs) are selected. Whereas, at the dynamic one, disjointed functionally homogeneous logic areas of the system, called logic regions, are identified. These latter are exploited to implement dynamic power management strategies. The key novelty of this work is the possibility of providing automatic power gating in ASIC designs, automating the generation of both the HDL power gated platform and the power format file to manage the additional power dedicated logic. Moreover, with respect to the previous works, we investigated the impact of the targeted technology on the ratio between static and dynamic contributions of the power consumption by implementing the assessed designs on both an ASIC 90 nm CMOS technology and a 45 nm one.

The rest of this paper is organized as follows. Section 2 defines the scientific context of this work. Section 3 describes the proposed power management approach. Section 4 discusses the achieved results, prior to conclude in Section 5.

2 Background

In this section we present the target application domain (Section 2.1) considered in this work, the basic instrument (Section 2.2) that we have extended to enable power management, and the foundations of the proposed techniques (Section 2.3).

2.1 Dataflow and Reconfigurable Video Coding

A *dataflow program* can be described with a directed graph, where nodes represent computational units (*actors*), while edges represent loss-less, order-preserving point-to-point connections between actors, used to communicate sequences of data packets (*tokens*). In terms of notation, let's define $DFG\langle V, E\rangle$ as a directed graph, where V is the set of vertices of the graph (the actors) and E is the set of edges (the connections). Actors asynchronously concur to the computation and can be transformed either into software agents or into physical Functional Units (FUs). Communication among the actors is token mediated. Several variations of this kind of dataflow model have been introduced in literature [17, 18]. The MPEG-RVC standards ISO/IEC 23001-4 and 23002-4 adopt a Dataflow Process Network (DPN) model of computation with firing rules [18]. DPN is extensively used due to its expressiveness and for the existence of a formal programming language, called CALTROP Actor Language (CAL), supporting it [19]. CAL directly captures the description of DPN actors. As depicted in Fig. 1, connections are implemented as FIFO channels, which are used to transmit tokens. Actors contain, besides a given state, a set of actions. Specific token configurations

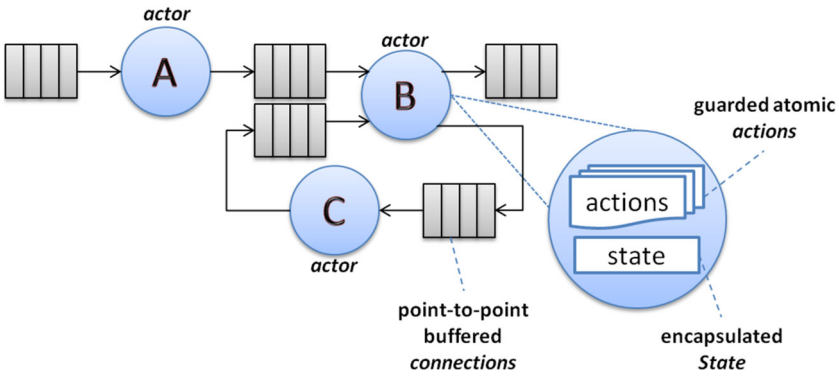


Figure 1 CAL dataflow network design example.

or sequences can fire actions inside the actors, according to predetermined guard expressions. Guards are boolean expressions on the current state and/or on input sequences that need to be satisfied for enabling the execution of an action. The actions execution may lead to state variations or to specific output tokens emission.

Since the definition of the MPEG-RVC standard, in 2010, several support tools have been designed, examples are Orcc [20], Xronos [21], Turnus [22], MDC [10] etc. However within MPEG-RVC power-management has been addressed only to a limited extent.

2.2 MDC: Multi-Dataflow Network Composition

The Multi-Dataflow Composer (MDC) was designed for the automatic creation and management of multi-functional systems. It was meant to address the difficulty of mapping different applications onto a coarse-grained reconfigurable architecture [23, 24]. Its final goal [10, 25, 26] is to automate such a mapping process while minimizing hardware resources, with consequent area/energy saving. This issue is known in literature as the datapath merging problem. MDC solves it by exploiting a heuristic algorithm. The tool is designed to be connected to higher-level utilities by means of an adequate front-end, in charge of parsing the high-level descriptions of the datapaths to be combined. In this way, relying on the chosen front-end, MDC in principle is able to process any type of dataflow model. At the moment, as it can be seen in Fig. 2, MDC is coupled to Orcc [20].

As it will be further detailed in Section 2.2.1, three major steps are required to generate the HDL description of a multi-functional reconfigurable architecture, starting from the DPN models of the functionalities to be implemented:

1. Orcc parses the Input DPNs (along with their actors) and translates each of them into an Intermediate Representation (IR), which is basically a directed graph.
2. The MDC front-end leverages on such flattened IRs to assemble a single multi-functional specification (*multi-flow IR* in Fig. 2). MDC front-end also keeps trace of

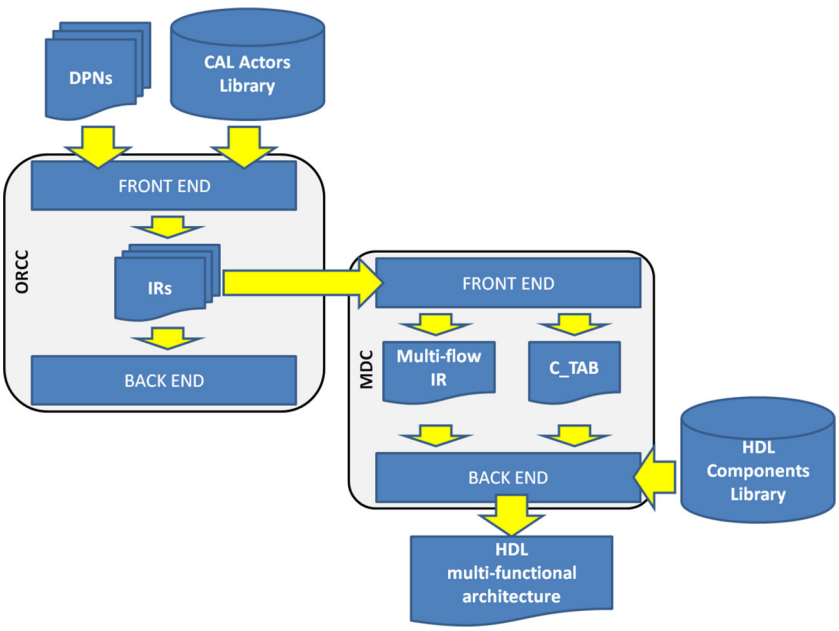


Figure 2 Multi-Dataflow Composer tool: an overview.

the system programmability through the *Configuration Table* (*C_TAB* in Fig. 2).

3. The MDC back-end then creates the respective HDL coarse-grained reconfigurable hardware (*HDL multi-functional architecture* in Fig. 2), mapping each actor on a different FU. FUs are passed as input to MDC within the *HDL components library* that can be manually or automatically created [21, 27]. FU granularity/functionality does not affect the described flow.

Reconfiguration can take place in a single-cycle, and is implemented by means of low overhead switching modules (SBoxes) placed at the crossroads between the different paths of data and driven by dedicated Look-Up Tables (LUTs), whose content is defined according to the *Configuration Table*. SBoxes are responsible of data routing, without computational overhead. In the current implementation, they are simple combinatorial multiplexers; therefore, no dedicated FIFO buffers are inserted for the SBox units. Nevertheless, the FIFOs of the upstream/downstream actors have to be managed. Sbox_1x2 units, inserted to split a path of data, require one FIFO for each outgoing connection. In the case of Sbox_2x1 units, inserted to access a common shared actor, the FIFO buffers are placed before the SBox along the incoming connections.

Since the SBoxes are fully combinatorial and the FIFO buffers always belong to the other actors, the well known dataflow problem of the FIFO buffers optimal sizing does not affect the MDC merging process. Input DPNs have only to be properly sized before the MDC execution.

MDC has been lately extended to provide automatic clock gating support [15, 16]. It is able to analyse the input DPNs and to extract from them the optimal actors partitioning in order to implement clock gating power management strategies. As shown in the following, a similar approach can be used to enable power gating.

2.2.1 Step-by-Step Example

To clarify how the reconfigurable multi-functional architecture is derived Fig. 3 depicts a step-by-step example. The starting points are three different DPN specifications (α, β and γ). The generated output is the HDL description of the multi-functional reconfigurable system.

At first α, β and γ are acquired by Orcc. It parses them and builds the corresponding IRs. Three different output IRs are produced. All of them are flattened representations of the corresponding DPNs, where the hierarchy of complex actors (i.e. non-atomic actors composed of a sub-network of actors) is exploded. In the considered example, β is composed of atomic actors only. On the contrary, the actor H of α and the actor J of γ are complex. These latter in the output flattened IRs are substituted by their corresponding sub-networks.

Then the MDC front-end starts an iterative merging process: it analyses the IRs in pairs to determine their similarities. Identical actors are shared in the output IR by introducing dedicated switching elements, used to fork ($Sbox\_1x2$) or re-join ($Sbox\_2x1$) the path of data. It is important to notice that for N input IRs, N-1 iterations are required to complete the merging process. In the considered case, 2 iterations are required. In the first run, the actors A and C are found to be identical among α and γ and two SBoxes are inserted. In the second run, the actor C is found to be identical among the previously generated *multi-flow IR* and β; therefore, just one additional SBox is inserted. During each iteration the C_TAB is updated to keep trace of the correct path of the token flow, in order to guarantee the computing correctness of each input DPN.

At last the MDC back-end generates the *HDL multi-functional architecture*, mapping the different actors of the *multi-flow IR* over the FUs provided within the *HDL components library*. The control signals of the physical SBoxes are generated by the LUTs, whose content depends on the final C_TAB produced by the MDC front-end.

2.3 Power-Awareness in Reconfigurable Architectures

In the *dark silicon* era [1, 2], when not all the available resources on a die are usable due to the limited power budget, power management strategies are of paramount importance. The ever-increasing integration on a single die determines the need of holistic power minimization approaches, acting across the whole design stack [28]. The proposed power management strategy combines structural (i.e. related with the composition and design-time configuration of the hardware platform architecture) and dynamic aspects (i.e. related with the management of the architecture at runtime). Moreover, leveraging the dataflow-based design approach, the automated application of this strategy is directly driven by system-level information.

2.3.1 Structural Power Management

Within MPEG-RVC, a first tentative approach to structural power management has been targeted in [29] that presents an analysis of the causation trace of a DPN resulting in the application of multi-clocking strategies. Such technique has never been applied to coarse-grained reconfigurable systems. In [30] an energy estimation methodology, based on Performance Monitor Counters (PMC), has been proposed to estimate the energy consumption of RVC-CAL video codec specifications. This approach addresses software-based solutions. Hardware oriented PMC-based energy estimation methodologies have never been applied

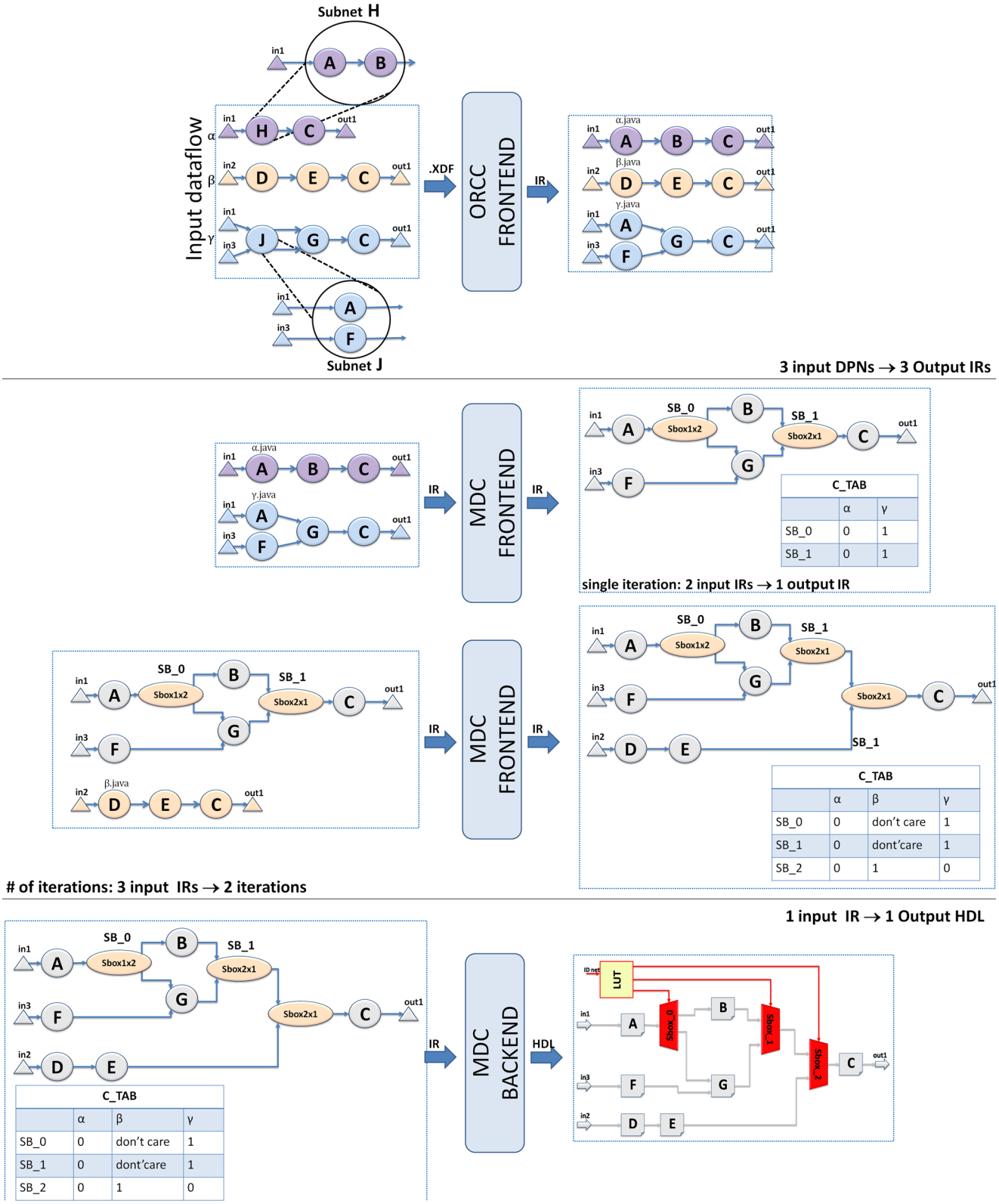


Figure 3 Multi-Dataflow Composer tool: a step-by-step example.

in RVC studies [31]. Moreover, please note that any kind of estimation methodology would require the implementation of a physical power saving strategy, such as those presented in Section 3.

Software frameworks and simulators have been proposed at the state of the art to account for structural power management and design space exploration (DSE) [32–34]. Within MPEG-RVC, exploration techniques have been used for actors re-factoring to improve throughput. The CAL Design Suite [32] works at software executable level while YACE, exploited in [33], performs analysis at the dataflow level. Both these tools do not target coarse-grained systems nor power minimization. Nevertheless, re-factoring techniques may always be implemented on any given input network, so that it would be possible to combine them with the proposed approach.

In this paper, exploiting the dataflow-based MDC profiling capabilities [35], we propose an exploration process that compares all the possible topologies that could be selected for the coarse-grained reconfigurable platform and selects the optimal configurations.

2.3.2 Dynamic 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, multi-frequency, operand isolation, multi-threshold, multi-supply libraries, power gating, etc.) exist and, in some cases, they are automatically implemented by commercial synthesis/place-and-route tools. However, generally speaking, invasive techniques (requiring insertion of additional logic and target technology support)[1] still need tools to be guided with significant manual effort by the designer.

Clock gating (CG) is an example of quite non-invasive technique. It may reduce the dynamic power consumption due to the clock tree and to sequential logic up to the 40 % [36]. CG consists in shutting off the clock of the unused synchronous logic, by means of simple *AND* gates. CG has been deeply automated and it is available on most of the commercial synthesizers. In the MPEG-RVC community, recent studies [37] presented an extension of an 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 regions can be clock gated to dynamically adapt power consumption when switching between different functionalities [15, 16]. 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.[2] CG can be applied at different granularity: fine-grained approaches act on single registers, whereas coarse-grained ones are referred, as in [15, 16, 37], to a set of resources. Commercial synthesizers normally can automate only fine-grained CG.

Power gating (PG) is quite invasive. The main idea behind PG is: 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. PG, as well as CG, is applicable at different granularity: fine-grained approaches require to drive 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. In this paper, we extend the logic regions identification process proposed in [15, 16] to support also automatic power gating with MDC. Each identified region will correspond to a different Power Domain (PD) that, in order to be managed will require inserting and driving the following resources:

- the *sleep transistor* between the gated region and the main power supply to switch on/off the derived power supply;
- 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;
- the *state retention logic* to maintain, where needed, the internal state of the gated region.

As far as we know, this additional logic can be manually inserted by the designers in the RTL architecture or through a power format file. Manual definition is highly error prone: it requires modelling the impact of low-power during simulation and providing multiple definitions for synthesis, placement, verification and equivalence checking [7]. On the contrary, the power format file allows designers specifying power saving techniques early in the design and do not imply any direct modification of the RTL code. Nevertheless, the definition of a power format file is still error prone. To deal with 10 different PDs, the designer needs to define the shut-off, isolation and state retention rules ($3 * 10 = 30$ interfaces). Moreover, for each rule, the enable signals have to be specified ([1*shut-off + 1*isol. + 2*reten.] * $10 = 40$

[1] Such as the availability of dedicated cells and processes on the implementation stack.

[2] Xilinx boards, for example, are equipped with dedicated blocks (*BUFGs*), whose outputs can drive distinct regions of logic powering down different design portions (when enabled).

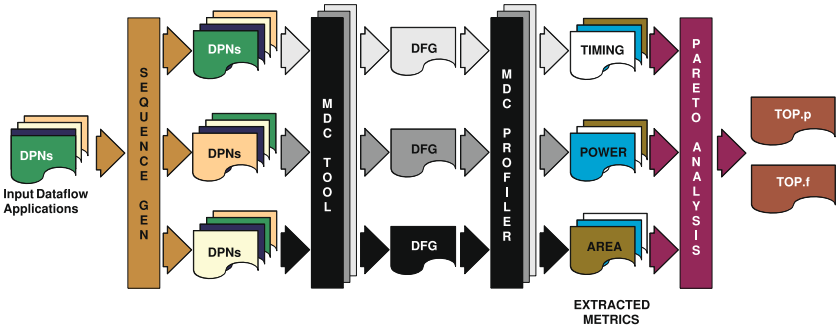


Figure 4 Topology Definer: an overview.

signals) and a dedicated power controller to properly drive them is also required. Therefore, the manual definition of the power format file may easily require significant effort. We have adopted the Silicon Integration Initiative's Common Power Format (CPF), whose definition is driven mainly by developers using Cadence [38] tools. Our strategy allows the automatic definition of the power controller and the CPF for coarse-grained reconfigurable systems.

3 Dataflow-Based Power Awareness

In this section we are going to present the early-stage power management strategy embedded in MDC. We have been working on two different extensions. The *Topology Definer* (Section 3.1) acts at the structural topology level by defining the optimal system configuration(s) minimizing the implementation costs. The *Logic Set Definer* (Section 3.2) acts at the dynamic level, identifying disjointed logic regions to be physically switched off when unused.

With respect to our previous works [15, 16, 35], the *Topology Definer* implements a more accurate frequency model and the *Logic Set Definer* has been enhanced allowing the possibility of providing power gating.

3.1 Structural Level: Profiling-Aware Topology Definition

Structural power management for coarse-grained reconfigurable architectures is needed for the following reasons:

- in a multi-functional environment, merging all the DPNs together is not always the optimal solution. An excessive number of switching elements may overcome the benefits of sharing an actor, causing both area and static power loss.
- in the adopted iterative merging algorithm, two networks at a time are processed by MDC. This merging process generates chains of SBoxes[3]. This may negatively affect the operating frequency. Therefore, it would be more efficient to merge just a subset of the input DPNs.

It is fundamental to determine the (sub-)optimal design specification(s). The *Topology Definer* coupled to MDC, as demonstrated in [35], is capable of determining the design costs of different implementations, before prototyping, through the back-annotation of low-level information on the DFGs. Figure 4 depicts the implemented profiling-aware topology strategy. In the following we define the steps in details.

Ordering and Sequence Extraction The *Sequence Generator* defines all the D possible DPNs sequences that are given in input to MDC according to the following equation:

$$
\begin{aligned}
D &= D\_notMer + D\_Mer + D\_partMer \\
D &= 1 + N! + \sum_{k=1}^{N-2} C_{N,k} * (N-k)! \\
D &= 1 + N! + \sum_{k=1}^{N-2} \frac{N!}{(N-k)! * k!} * (N-k)! \\
D &= 1 + N! + \sum_{k=1}^{N-2} \frac{N!}{k!}
\end{aligned} \tag{1}
$$

where *D_notMer* is the *static*, not merged, composition of the N DPNs in parallel; *D_Mer* is the *all merged* term that, maximizing resource sharing, is given by all the possible permutations of the DPNs merged into a unique one; *D_partMer* is the *partially-merged* term that, not following the resource maximization principle, provides all the sequences composed of the combinations[4] that can be extracted from a subset of k input DPNs placed in parallel with all the permutations of the other N-k networks merged together.

Multi-Dataflow Specification Definition For each sequence, the MDC tool extracts the DFG (Section 2.2).

[3] In the worst case scenario, where all the input DPNs share the same actor, the chain will have as many element as the iterations are.

[4] A combination is a selection of all or part of a set of objects, regardless to the order in which they are selected. Given A, B and C, the complete list of possible selections of two items would be: AB, AC, and BC.

Profiling For each DFG, implementation costs are computed. Having already back-annotated the HDL components library with one value of estimated area and power consumption[5] per FU, the *MDC Profiler* retrieves $\forall v_i \in V$ the corresponding back-annotated values, a_i and p_i, and sum them up to estimate respectively area and power[6] consumption of each DFG. Therefore, given as M the size of the V set, area and power consumption are determined as:

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

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

Operating frequency estimation is less straightforward. Different feeding orders may result in different cascades of SBoxes that may negatively impact on the critical path (CP). The *MDC Profiler* $\forall n_i \in InN$ (being InN the set of input DPNs) retrieves the corresponding back-annotated CP,[7] CP_i, and defines $CPstatic = max(CP_i)$ as the CP of the non reconfigurable system configuration (with all the given DPNs in parallel). Then it estimates the longest cascade of SBoxes ($seqSB$) within the considered DFG. Given N_S as the number of SBoxes composing $seqSB$ and b as the number of bits of SBoxes data, the CP due to the cascade of SBoxes is given by the following empirical equation:[8]

$$CP\_seqSB = f(b) * ln(N_S) + g(b) \tag{4}$$

Coefficients f(b) and g(b) change depending on the type of SBox (Sbox_1x2 or Sbox_2x1).

- Sbox1x2: $f(b) = 0.268 * b + 59.85$, $g(b) = -0.294 * b + 406.3$
- Sbox2x1: $f(b) = 0.114 * b + 87.70$, $g(b) = 0.185 * b + 393.1$

The *MDC Profiler* finally compares $CPstatic$ and CP_seqSB: the maximum of these two values determines the operating frequency of the given design point.

[5]The back-annotation requires performing of a training set of synthesis trials. In this paper we have used the RTL Compiler of Cadence SoC Encounter and a 90 nm CMOS technology.

[6]Only static contribute of the power consumption is considered.

[7]SoC Encounter has been used to extract the CP associated to the N different input specifications synthesized stand-alone with a 90 nm CMOS technology.

[8]Coefficients in Eq. 4 are technology dependent and have to be modeled for each target technology node by interpolating a training set of experimental results. This form derives by experiments carried out by means of the RTL Compiler (Cadence SoC Encounter), using ASIC CMOS 90 nm technology as reference, varying the number of SBoxes (from 1 to 100) and the number of bits of SBoxes data (1, 8, 16, 32, 64).

Pareto Analysis The Pareto-based analysis is carried out exhaustively on the entire design space to determine the optimal system configuration(s) according to the selected design effort. For power management purposes it will be extremely important to determine the least consuming configuration, but minimizing power consumption not necessarily implies having also the best operating frequency. Therefore, two sub-optimal $DFGs$ are provided as output: the area/power (*TOP.p*) one and the frequency (*TOP.f*) one.

3.1.1 Design Space Exploration Time

Exhaustive exploration approaches, where all the design points are characterized in terms of objective functions, may lead quickly to search time explosion. According to Eq. 1, the design space size grows with the number of input DPNs. Therefore, it is clearly application specific. In [15] it has been demonstrated that the design space dimension grows with $3 * N!$, where N is the number of input DPNs. Nevertheless, as demonstrated in [35], different properties of the design space can be exploited to define a robust and scalable heuristic algorithm performing approximated Pareto analysis:

- *TOP.p* normally is an "all-merged" solution. By construction, the smallest number of actors would provide the smallest area and power consumption according to Eqs. 2 and 3.
- *TOP.f* may be a "partially-merged" solution, in fact the fewer networks you merge, the smaller is the CP associated to the SBox units chain $seqSB$. Therefore, if the operating frequency is determined by CP_seqSB you can improve it by limiting $seqSB$, placing in parallel to the rest of the design one of the networks contributing to $seqSB$.

The actual implementation of these heuristics will constitute one of the future improvements of the proposed work.

3.1.2 Step-by-Step Example

To clarify the different logic phases of the proposed methodology Fig. 5 depicts a step-by-step example. The starting points of the dataflow-based power management strategy are three different DPN specifications (α, β and γ). The generated outputs of the structural level step are the two multi-dataflow DFGs, *TOP.p* and *TOP.f*. By applying Eq. 1, 13 points constitute the design space as possible system specifications. Among these, the *Topology Definer* is capable of identifying the power optimal and the frequency optimal ones. The former, *TOP.p*, is an *all-merged* solution (the optimal feeding order is α, γ and β), while the latter, *TOP.f*, is a *partially-merged* one (β is kept in parallel while α and γ are merged).

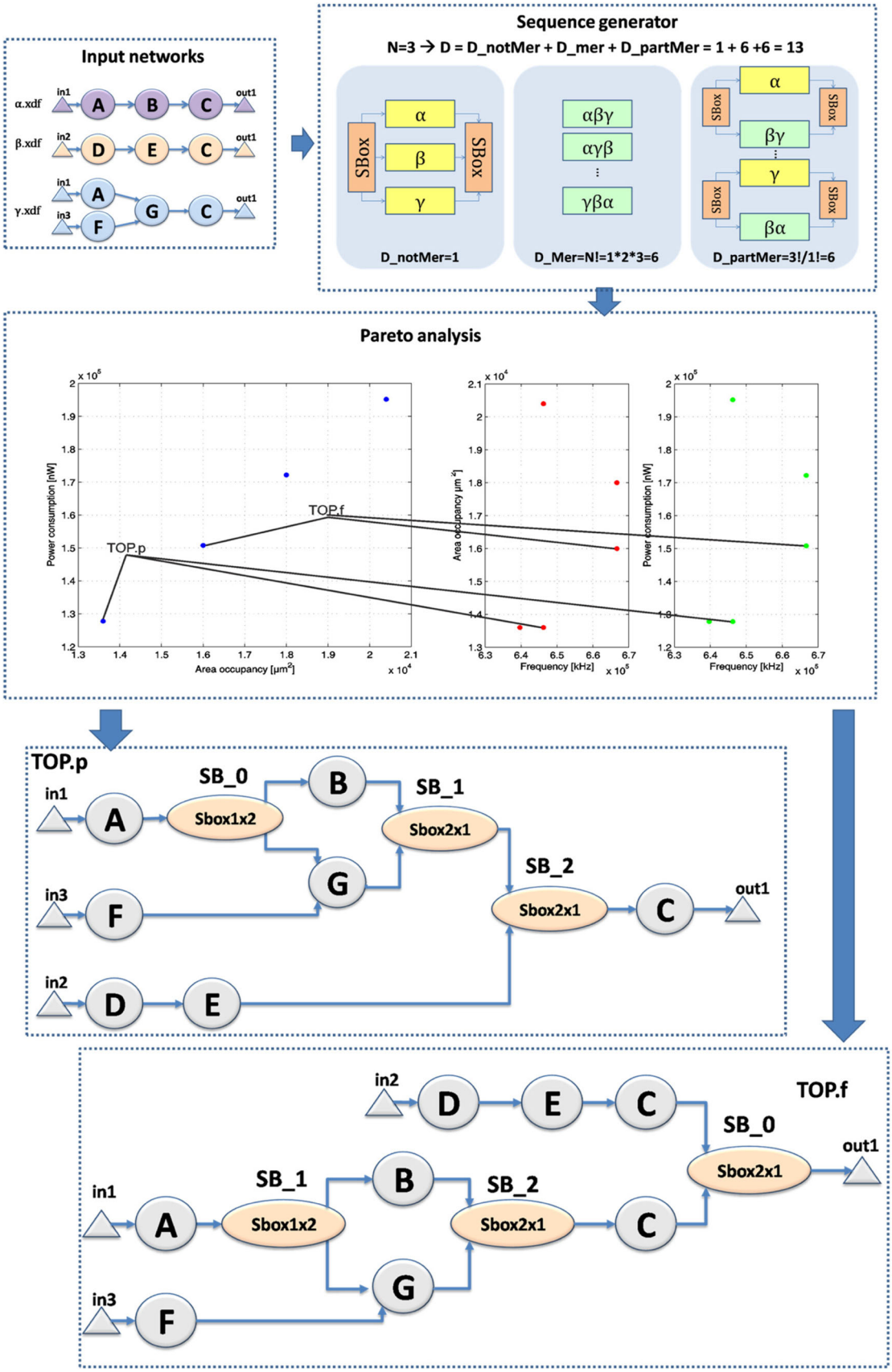


Figure 5 Topology Definer: a step-by-step example.

3.2 Dynamic Level: Homogeneous Logic Regions Definition

Coarse-grained reconfigurable architectures may benefit from the application of dynamic power management strategies, since:

- within a multi-functional architecture, when an application is executed, the rest of the design (not involved on it) is in an idle state;
- the part of unused resources is fixed as soon as the multi-functional architecture is deployed; therefore, it can be determined at design-time.

According to these considerations, a reconfigurable multi-functional architecture is characterized by disjointed Logic Regions (*LRs*) composed of resources that are active/inactive together. In this work, we propose an algorithm for the automatic identification of these *LRs* at the specification level. It exploits the intrinsic modularity of the DPN models. The given DPNs are analysed to determine and group together the actors active/inactive at the same time within homogeneous logic sets. On the MDC GUI users can choose enabling or not dynamic powering down strategies. In the former case they need to specify how they intend to exploit the identified *LRs* to deploy an efficient reconfigurable platform, adopting either power gating or clock gating strategies, by ticking the strategy to be supported.

Algorithm 1: Logic Set Definer: baseline Logic Regions identification. ($N = |LR\_MAP|$).

```
foreach DPN_i in input DPNs do
    V'_i =mapping of DPN_i in DPN;
    if isEmpty(LR_MAP) then
        //First iteration
        S_0 = V'_i;
        put key S_0 with value DPN_i in LR_MAP;
    else
        foreach S_j in LR_MAP keys do
            if V'_i = S_j then
                //V'_i is coincident with an already identified LR
                add DPN_i to value of key S_j in LR_MAP;
                V'_i = 0
                break;
            end
            if V'_i ∩ S_j ≠ ∅ then
                //V'_i partially overlaps with an already identified LR
                S_N = V'_i ∩ S_j;
                put key S_N with value DPN_i in LR_MAP;
                S_j = S_j − S_N;
                V'_i = V'_i − S_N;
            end
        end
        if !isEmpty(V'_i) then
            //The elements left in V'_i need to constitute a new LR
            S_N = V'_i;
            put key S_N with value DPN_i in LR_MAP
        end
    end
end
```

The MDC tool, as described in [16] and deeply mathematically demonstrated in [15], acts on the (sub-)optimal specification identified by the *Topology Definer* (either *TOP.p* or *TOP.f*) to identify all the *LRs*. It maps each set of actors V_i, belonging the i-th input network, to a set of actors $V_i' \subseteq V$, where $DFG\langle V, E\rangle$ corresponds to the multi-functional specification. Originally, in the *LRs* identification process, which algorithm is referred as Algorithm 1, the SBoxes were not included. We will see in Section 3.2.3 that, for power gating purposes, the identification will have to be extended to include also the SBoxes, which contribute both to the static and the dynamic power consumption. This implies that the final *LRs* partitioning cannot be optimal for both *TOP.f* and *TOP.p*: they normally present different SBox chains/number.

Considering the baseline identification algorithm, Algorithm 1, for each input DPN MDC extracts its corresponding set V_i' composed of computational actors only. Considering S as the complete set of *LRs*, the aim is minimizing the number of elements, N, within S, in order to minimize the additional logic needed to drive the *LRs*. This is true both for clock gating and power gating. Moreover, S must contain a partition of the set of actors V_i' of each input network. In particular, this would mean that, when an input network is executed, only its actors will be triggered. Beside S, Algorithm 1 defines also the association map (LR_MAP) of correspondences between the input DPNs and the elements of S. At the beginning this map is empty, so that the first V_i' constitutes the first *LR*. For all the other iterations, as shown in the pseudo-code, LR_MAP is not empty anymore and the algorithm extracts, one at a time, all the already identified *LRs* $S = \{S_1, S_2, ...S_P\}$ to be compared with V_i'. Three different situations may occur:

- the current set is equal to an already identified one ($V_i' = S_j$): the association map is updated so that the matching *LR* points also to the current DPN;
- the current set intersects one of the identified ones ($V_i' \cap S_j \neq \emptyset$): a new *LR*, containing the intersected instances, is issued and the pre-determined set S_j and V_i' are modified by removing the intersection. A new entry in the LR_MAP is inserted to match the new set with the current input DPN and all the DPNs already associated to S_j.
- the current set is completely disjointed with respect to the previous ones or there are some elements within it that are not overlapped with any *LR* (at the end of the comparison process ($V_i' \neq \emptyset$)): a new *LR* is created and pushed in S, with the corresponding entry in the association map.

3.2.1 Step-by-Step Example

Here follows a step-by-step example of the application of Algorithm 1. This example is also depicted in Fig. 6. The

input DPNs are: α (composed of actors A, B and C), β (D, E and C) and γ (A, F, G and C).

1. Firstly the algorithm analyses V'_α, composed with the actors of α.
   - The association map is empty.
   - V'_α is issued as S_1 and a reference from S_1 to α is inserted in the association map.
2. Then V'_β is considered.
   - The association map is not empty anymore and $S = \{S_1\}$.
   - V'_β and S_1 are intersected: they share the actor C.
     - The shared actor C is issued as S_2.
     - The actor C is removed from V'_β and S_1.
     - A reference from S_2 to both α and β is defined.
   - S does not have any other element available for comparison.
   - The remaining V'_β is issued as S_3, referencing in the association map just β.
3. Finally V'_γ is considered.
   - The association map is not empty and $S = \{S_1, S_2, S_3\}$.
   - S_1 shares with V'_γ the actor A.
     - A new logic region, S_4, containing the actor A is issued.
     - A reference from S_4 to α and to γ is pushed in the association map.
     - The actor A is removed from S_1 and V'_γ.
   - S_2 is entirely contained within V'_γ.
     - A reference from S_2 to γ is added in the association map.
     - The actor C is removed from V'_γ.
   - S_3 and V'_γ are disjointed.
   - S does not have any other element available for comparison.
   - The remaining V'_γ, containing the actors F and G, is issued as S_5. It matches just γ in the association map.

Figure 6 shows also the resulting output of the proposed process. Five different sets of computational actors are identified from the three input dataflow specifications. Please note that the SBoxes do not belong to any *LRs*.

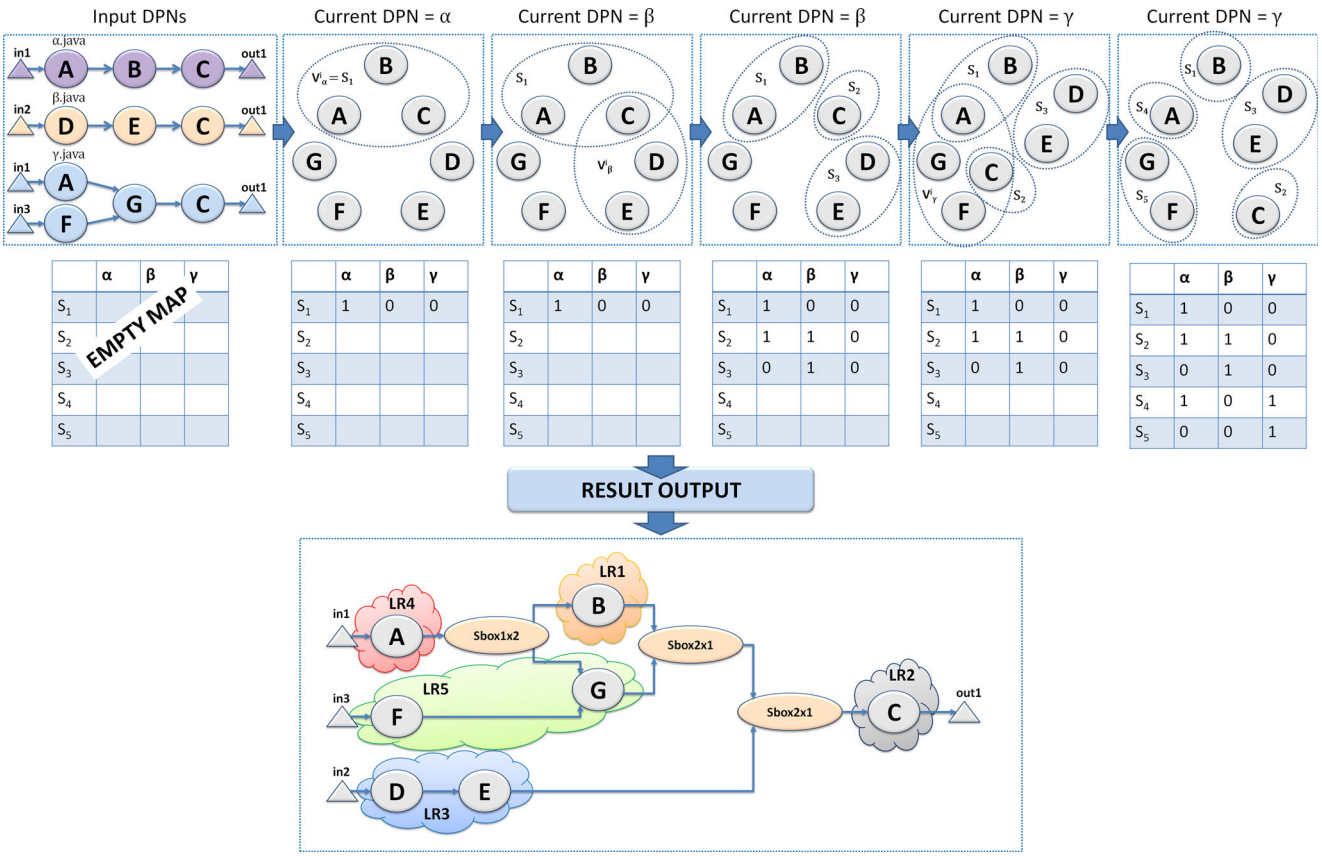


Figure 6 Logic Set Definer: a step-by-step example of Logic Regions identification.

3.2.2 Clock Gating Implementation

At the hardware level, *LRs* are exploited by MDC to implement clock gating. Each identified *LR* potentially corresponds to a different Clock Domain (CD). The objective is to reduce the dynamic part of the power consumption leveraging on the following assumption: the clock of the CDs that are not working, while a DPN is executed, can be turned off to limit the switching activity, and in turn their dissipation, of the inactive FUs. Clock gating implementation is target dependent. MDC provides *AND* gates cells (applied directly on the clock to disable it) for ASIC designs and *BUFG* cells for FPGA implementations on Xilinx boards. Targeting FPGA technologies, the number of allowed *LRs* is limited by the amount of *BUFG* cells on the chosen board/device. To take this constraint into account, MDC provides a way to reduce the number of *LRs*. Sub-optimal *LRs* (where switching activity in unused FUs is present) identification is the output of the *LRs* fusion process performed by the algorithm presented as Algorithm 2. This algorithm merges together two *LRs* at a time, according to a cost function, if the user on the MDC GUI specifically asks for a fixed number (TH) of CDs. Different cost functions are available:

- minimizing the number of units per *LR* - Given c_i, as the cardinality of the i-th *LR*, and P, as the number of networks activating it, we can define wN_i:

$$wN_i = c_i * P \tag{5}$$

  as the *weight* of considered region.
- minimizing the static power consumption per *LR* - Given p_i, as the estimated power consumption (given by Eq. 3) of the i-th *LR*, and P, as the number of networks activating it, we can define wP_i:

$$wP_i = p_i * P \tag{6}$$

  as the *weight* of considered region.

Obviously, in both cases, the fusion process aims at keeping all the weights as minimal as possible. When FPGA technologies are selected the former is applied, whereas the latter (exploiting the back-annotate libraries described in Section 3.1) is allowed also when, for some reasons, adopting an ASIC technology the number of *LRs* has to be limited.

Algorithm 2: Logic Set Definer: Logic Regions merging. ($N = |LR\_MAP|$).

```
while N >= TH do
    sort LR_MAP basing on CF;
    S_lc =lowest cost S_j in LR_MAP;
    S_slc =second lowest cost S_j in LR_MAP;
    put key S_mrg = S_lc ∪ S_slc in LR_MAP;
    add value of key S_lc to value of key S_mrg in LR_MAP;
    add value of key S_slc to value of key S_mrg in LR_MAP;
    remove keys S_lc and S_slc from LR_MAP;
end
```

Starting from the *LRs* identified in Fig. 6, Fig. 7 shows the hardware implementation of the clock gating management:

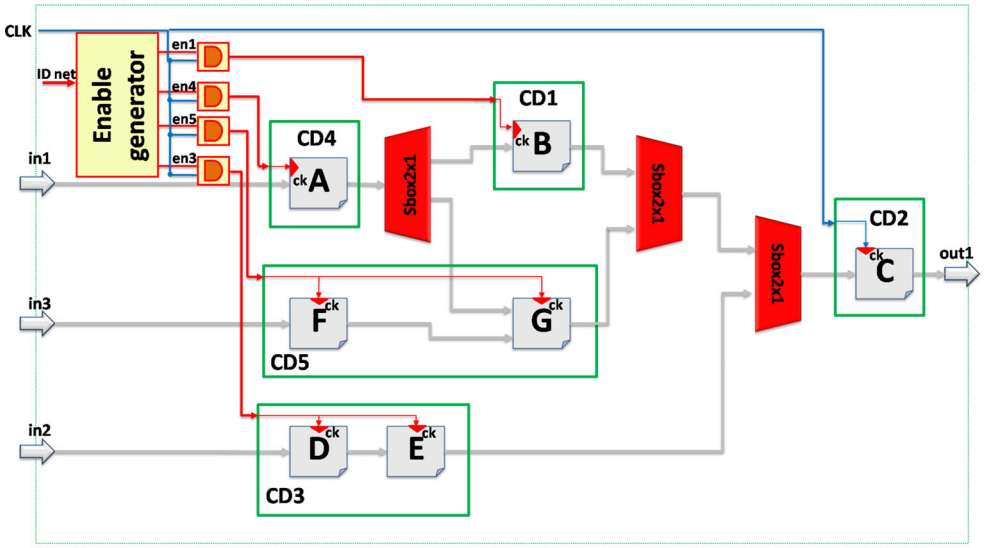


Figure 7 Logic Set Definer: clock gating physical implementation.

Network	Clock Domains
α	CD1, CD2, CD4
β	CD2, CD3
γ	CD2, CD4, CD5,

– CD2, corresponding to the always-on *LR2*, is clocked by the input clock;
– CD1, CD3, CD4 and CD5, corresponding to the *LR1*, *LR3*, *LR4* and *LR5*, are clock gated by means of four clock gating cells;
– the *Enable Generator* generates the enable signals for the clock gating cells on the basis of the requested functionality and it is automatically created and instantiated in the design.

3.2.3 Power Gating Implementation

The main contribution of this paper is the automatic implementation of a power gating strategy, extending the previously described *LRs* identification process. In this case also the asynchronous FUs, such as the SBoxes, have to be included within the *LRs*. They contribute both to the static and dynamic power consumption.

Considering $DFG\langle V, E\rangle$ as the directed graph of the multi-functional specification, when power gating strategies are implemented, the V set includes both the set of computational actors, V', and the set of SBoxes, V'', as $V = V' \cup V''$. At first the basic identification algorithm (Algorithm 1a) is applied and then, starting from the N identified *LRs*, the SBoxes identification algorithm is applied. This latter, hereafter referred as Algorithm 3, exploits the information within the configuration table (C_TAB) to add the SBoxes V'' to the partition S. The algorithm consists of two main steps:

- for each SBox, the algorithm creates a set DPN_{SBi} analysing all the input DPNs. If a DPN activates the considered SBox (the corresponding value within the C_TAB is not a *don't care*) then it has to be active when that functionality is requested. Therefore, the input DPN is added to DPN_{SBi}. When all input DPNs have been cross-checked, the DPN_{SBi} set is added to the SB_MAP. This latter is a map used to keep trace for each SBox, SB_i, of the set of DPNs that need its activation during the execution.
- For each SB_i in the SB_MAP, the algorithm compares the value DPN_{SBi} with all the N *LRs* in the LR_MAP. The following situations may occur:
  - the considered SBox set is activated by the same DPNs triggering the considered *LR* (DPN_{SBi}=$DPNs_j$): SB_i does not represent a new region and it is simply added to S_j;
  - the considered SBox set is activated by different DPN(s) with respect to all the considered *LRs* (SB_i is still unassigned, $!assignedSB = 1$): SB_i represents a new region; therefore, a new set ($S_N = SB_i$), is added to LR_MAP.

Algorithm 3: Logic Set Definer: Logic Regions set extension with the SBoxes. ($N = |LR\_MAP|$).

```
foreach SB_i in C_TAB do
    DPN_SBi =new empty set;
    foreach DPN_k in value of key SB_i in C_TAB do
        if getSBoxValue(SB_i,DPN_k) != don't care then
            add DPN_k to DPN_SBi;
        end
    end
    add value DPN_SBi to key SB_i in SB_MAP
end
foreach SB_i in SB_MAP keys do
    assignedSB=false;
    foreach S_j in LR_MAP keys do
        DPN_SBi =value of key SB_i in SB_MAP;
        DPNs_j =value of key S_j in LR_MAP;
        if DPN_SBi=DPNs_j then
            add SB_i to key S_j in LR_MAP;
            assignedSB=true;
            break;
        end
    end
    if !assignedSB then
        S_N = SB_i;
        put key S_N with value DPN_i in LR_MAP;
    end
end
```

Starting from the output of the step-by-step example in Fig. 8, here follows the application of Algorithm 3. The SBoxes involved are: SB_0, SB_1 and SB_2. At first the algorithm processes C_TAB to create SB_MAP:

1. SB_0 is considered.
   - Create a new empty set DPN_{SB0}
   - Analyze all $DPNs$ of SB_0 in C_TAB
     – $SB_0(DPN_\alpha) = 0$: add α to DPN_{SB0}.
     – $SB_0(DPN_\beta)$ = *don't care*: don't add β to DPN_{SB0}.
     – $SB_0(DPN_\gamma) = 1$: add γ to DPN_{SB0}.
   - DPN_{SB0} is added as value to key SB_0 in SB_MAP.
2. SB_1 is considered.
   - Create a new empty set DPN_{SB1}
   - Analyze all $DPNs$ of SB_1 in C_TAB
     – $SB_1(DPN_\alpha) = 0$: add α to DPN_{SB1}.
     – $SB_1(DPN_\beta)$ = *don't care*: don't add β to DPN_{SB1}.
     – $SB_1(DPN_\gamma) = 1$: add γ to DPN_{SB1}.
   - DPN_{SB1} is added as value to key SB_1 in SB_MAP.
3. Finally SB_2 is considered.
   - Create a new empty set DPN_{SB2}
   - Analyze all $DPNs$ of SB_2 in C_TAB

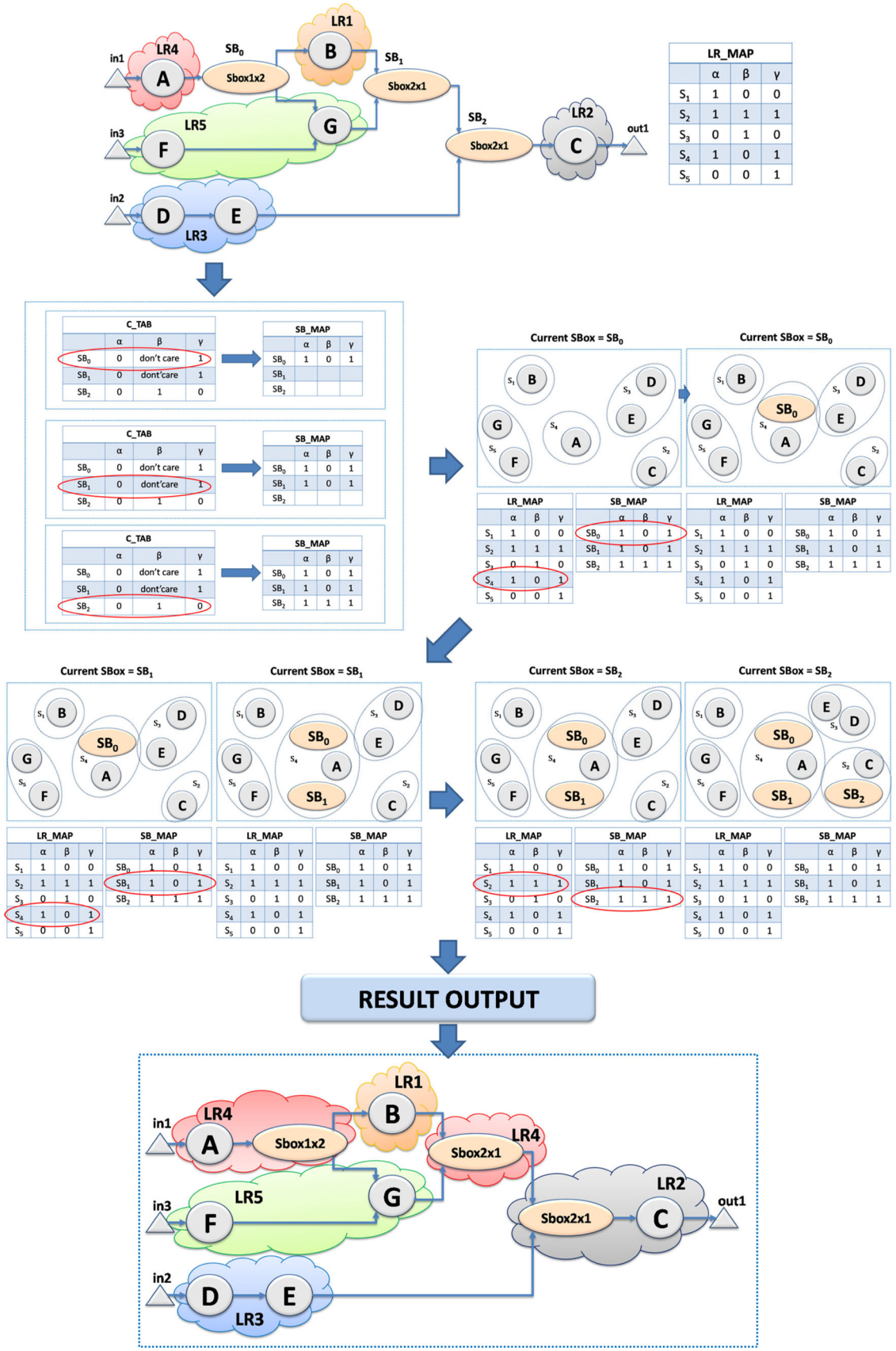


Figure 8 Logic Set Definer: a step-by-step example of the Logic Regions set extension with the SBoxes.

  - $SB_2(DPN_\alpha) = 0$: add α to DPN_{SB2}.
  - $SB_2(DPN_\beta) = 1$: add β to DPN_{SB2}.
  - $SB_2(DPN_\gamma) = 0$: add γ to DPN_{SB2}.
- DPN_{SB2} is added as value to key SB_2 in SB_MAP.

Once the SB_MAP is determined, Algorithm 3 compares each DPN_{SBi} in SB_MAP with all the S_j in LR_MAP.

1. SB_0 is considered.
   - Compare DPN_{SB0} with all sets S_j in LR_MAP
     - $DPN_{SB0} = DPN_{S4}$
     - SB_0 is added to $LR\ S_4$ in LR_MAP
2. Then SB_1 is considered.
   - Compare DPN_{SB1} with all sets S_j in LR_MAP
     - $DPN_{SB1} = DPN_{S4}$
     - SB_1 is added to $LR\ S_4$ in LR_MAP
3. Finally SB_2 is considered.
   - Compare DPN_{SB2} with all sets S_j in LR_MAP
     - $DPN_{SB2} = DPN_{S2}$
     - SB_2 is added to $LR\ S_2$ in LR_MAP

At the end of the process, each *LR* corresponds to a different Power Domain (PD). At runtime, the PDs involving idle resources can be switched off. For each input DPN a different Power Mode (PM) (e.g. a fixed PDs setting) is created. Each PM represents a steady state of the design in which some PDs are active, while others are disabled, according to the *LRs* to be activated by the requested functionality. In order to give to the synthesizer all the information about the power intent, a CPF file is also generated.

MDC adds to the reconfigurable HDL an automatically generated *Power Controller*, which is mainly composed of a different finite state machine for each PD. The *Power Controller* properly drives: *a)* the enable of the *Power Switches* and of the *Isolation* cells, *b)* the save command of the *State Retention* cells (to store registers values before their shutting off), and *c)* the restore command of the *State Retention* cells (to retrieve registers values once switched on).

Depending on the used technology libraries, these signals may change in polarity and timing. For this reason, the CPF file generation requires as input a technology file, where all the above mentioned power gating cells have to be specified and characterized. The CPF generated by MDC is divided in two main parts: technology and power intent.

- The technology part sets the timing libraries and specifies the low-power cells (`define_xxx cell` commands) to be used within the technology specific physical libraries.
- The power intent part depends on the design and manages the PDs, PMs and the respective transitions. Here the power cells, specified on the technology part, are instantiated and associated to the FUs in the design (`create_xxx cells` and `update_xxx cells` commands), accordingly with the PDs. All the logic belonging to the always-on domain is automatically associated to the default PD, which has no associated switching-off logic.

An excerpt of the power intent CPF file corresponding to the five *LRs* identified in Fig. 8 is provided in Appendix A. Figure 9 shows the implementation of the power gating management in the resulting platform:

- PDdef, corresponding to the always-on *LR2*, does not require any power management support;
- PD1, PD3, PD4 and PD5, corresponding to *LR1*, *LR3*, *LR4* and *LR5*, are power gated by means of three different types of cells, *Power Switch*, *Isolation* and *State Retention* cells (SBoxes do not need these latter since they are state-less);
- the *Power Controller* to properly generate the enable/restore signals for the power gating cells.

4 Assessments

This section validates the proposed dataflow-based power management strategy in the real use case scenarios, described in Section 4.1. Section 4.2 briefly introduces the trials setup chosen for the analysis. Whereas, Section 4.3 reports the assessments of the structural level phase of the proposed strategy, while Section 4.4 shows those of the dynamic one. With respect to our previous works [15, 16, 35], a more extensive validation phase over different image processing applications is presented. In particular, beside the already adopted zoom application, we have implemented two new scenarios and we have compared the different MDC-based and commercial tool based power management strategies. Moreover, since the targeted technology impacts on the ratio between static and dynamic contributions of the power consumption, we present also some preliminary explorations targeting a 45 nm CMOS technology.

4.1 Application Scenarios

To validate the proposed power-aware design flow, we targeted three different application scenarios for image processing:

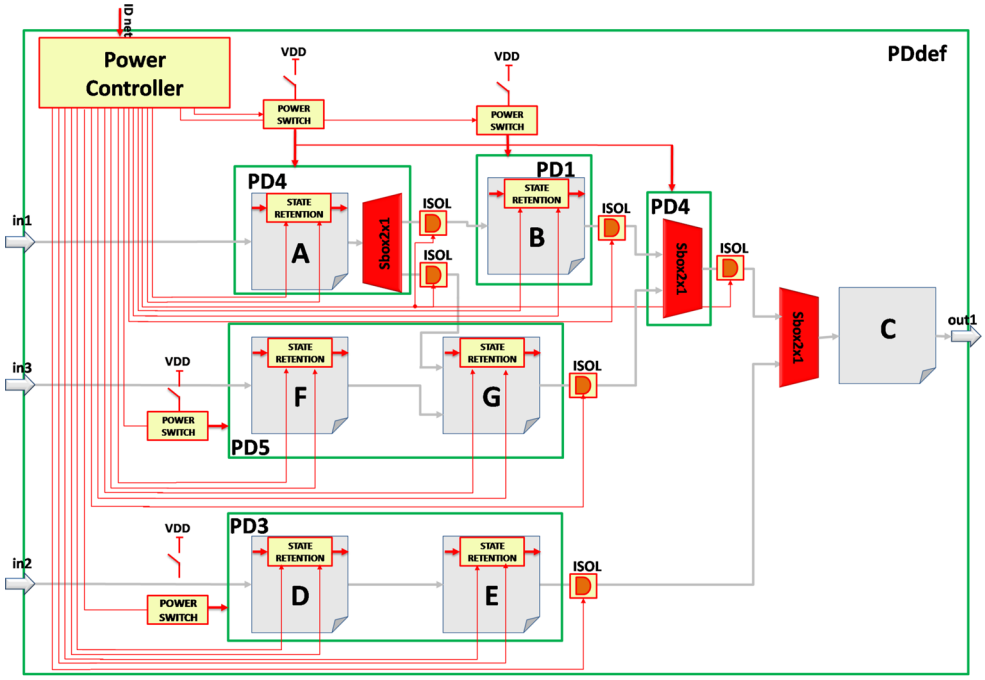


Figure 9 Logic Set Definer: power gating physical implementation.

Network	Power Modes	Power Domains
α	PM1	PDdef, PD1, PD4,
β	PM2	PDdef, PD3
γ	PM3	PDdef, PD4, PD5,

– Spatial Anti-Aliasing (UC1), to correct the distortion effects of an image downsampled violating the Nyquist constraint;
– Zoom (UC2), to scale an image by a given zooming factor interlacing pixels of the original image with other pixels coming from adaptive interpolations of the neighbouring ones;
– Deblocking Filter (UC3), to remove image distortions, like blocking and ringing, coming from compression processes.

The applications have been profiled to identify the most computationally intensive code segments (*computational kernels*). We have identified six, seven and eight kernels respectively for UC1, UC2 and UC3. These kernels have been modelled as DPNs according to the RVC-CAL dataflow model. The final intent is accelerating in hardware these applications by mapping their respective kernels over a different reconfigurable multi-functional architecture, assembled with MDC.

Table 1 depicts an overview of the kernels composition, functionalities and occurrences within the three applications. The number of actors is a raw index of the kernel size, which is important to understand the kernel power consumption results. A more precise insight of the size of each kernel is provided in Table 2 that reports both the overall kernel area (along with its percentage with respect to the corresponding use case) and the area shared with other kernels (along with its percentage with respect to the overall kernel area). Back to Table 1, the data size is referred to the number of tokens that the kernel is able to process for every execution. This metric gives an idea of the kernel complexity and of its execution latency. The occurrences of a given kernel (*#occ* in Table 1) are intended as the number of times the kernel is executed while running its corresponding application. If a kernel is executed several times, its activation frequency is high and, in turn, its switching activity largely impacts on the dynamic power consumption.

4.2 Assessment Setup

In terms of validation the proposed methodology is assessed over different coprocessor architectures, accelerating in hardware the applications presented in the previous section. Each coprocessor has been practically assembled with MDC, enabling/disabling the proposed power-management extensions.

In any case, the ASIC synthesis explorations, to be discussed in the following sections, have been performed using RTL Compiler (from the Cadence SoC Encounter commercial release). With respect to power consumption, all the power estimations have been extracted with RTL Compiler

Table 1 Computational kernels of the adopted use cases.

app	kernel	# actors	# occ	data size	functionality
UC1	*sorter*	2	18659	25	vector sorting
	min_max	1	14864	2	maximum/minimum finding
	rgb2ycc	19	256	$3 \times 4 \times 4$	RGB to YCrCb colour conversion
	ycc2rgb	18	256	$3 \times 4 \times 4$	YCrCb to RGB colour conversion
	abs	1	124366	1	absolute value calculation
	corr	10	2321	25	vector correlation
UC2	*min_max*	1	15142	2	maximum/minimum finding
	abs	1	70045	1	absolute value calculation
	sbwlabel	17	966	16	edge block checking
	chgb	7	3072	4	bilevel/grayscale block checking
	cubic_conv	6	341	16	cubic filter convolution
	median	9	1253	4	median calculation
	cubic	10	1496	4	linear combination calculation
UC3	*min_max*	1	32806	2	maximum/minimum finding
	filter	13	2235	10	vector filtering
	clip	2	346	2	vector comparison
	inner	9	1108	4	vector weighting and sum
	mdiv	7	346	4	vector biasing
	sign	1	346	1	sign calculation
	rgb2yuv	19	256	$3 \times 4 \times 4$	RGB to YUV colour conversion
	yuv2rgb	18	256	$3 \times 4 \times 4$	YUV to RGB colour conversion

taking into account the real switching activity, collected during post-synthesis hardware simulations (performed with Cadence SimVision Debug) within VCD files. Therefore, dynamic power consumption numbers are accurate and representative of the on-off system conditions.

4.3 Structural Level Assessments

The DPNs described above have been passed as input to the *Topology Definer* to perform the design space exploration. UC1 is composed of 1951 design points, UC2 of 13693 and UC3 of 109493 points. These explorations are still feasible, UC3 analysis required just a couple of minutes. Therefore, an exhaustive exploration was performed and it was not necessary to apply any pruning on the design space size.

For all the considered use cases the *Topology Definer* analysis does not detect any degradation of the reconfigurable system maximum operating frequency. This means that, according to Eq. 4, the SBoxes insertion caused combinatorial paths (see Section 3.1) shorter than those of the isolated DPNs: resource sharing preserved the performance of the isolated applications. Since a single data is representative of all the design space, for each considered scenario the frequency graph is not presented. Area and power estimations reported in the Pareto graphs have been derived according to Eqs. 2 and 3.

UC1 area versus power Pareto graph is shown in Fig. 10. It is possible to identify 3 different clusters: *a1*, where the DPNs kept in parallel, if present, are always the smallest ones (*abs*, *min_max*, *sorter*); *a2*, where the largest DPN (*ycc2rgb*) is always merged and, sometimes, the second largest kernel (*rgb2ycc*) is kept in parallel; and, *a3* that involves the remaining design space points, including the non-reconfigurable one (which is the worst point for this UC). In frequency, the estimated maximum value due to the longest SBoxes chain is 1.56 GHz, which is larger than the one fixed by the isolated input DPNs (202.6 MHz). Therefore, *TOP.p* and *TOP.f* are coincident corresponding to an *all-merged* solution that belong to *a1*:*TOP.p/f* is 94513 μm^2 large and consumes 49.1 μW. With respect to the synthesized design the estimation error is 1.5 % in area and 2.3 % in power.

UC2 area versus power Pareto graph is depicted in Figure 11. There are 4 identified clusters: *z1* keeps (again) in parallel the smallest DPNs (*abs*, *min_max*); *z2* and *z3* respectively keep in parallel always one and only one or two and only two of the largest kernels (*sbwlabel*, *chgb*, *cubic*, *median*, *cubic_conv*); *z4* contains the remaining design space points, including the non-reconfigurable one (which is not the worst). In terms of frequency the SBoxes maximum value is 1.51 GHz against the 384.6 MHz of the isolated DPNs. Again, *TOP.p* and *TOP.f* are coincident and correspond to an *all-merged* solution that belong to *z1* with

Table 2 Area occupancy of the kernels within the adopted use cases targeting a 90 nm ASIC technology.

app	kernel	area [μm^2]	%[a]	shared area [μm^2]	%[a]	%[b]
UC1	*sorter*	12010	12.90	3946	4.24	32.86
	min_max	3946	4.24	3946	4.24	100.00
	rgb2ycc	37084	39.82	13955	14.98	37.63
	ycc2rgb	39862	42.80	13955	14.98	35.01
	abs	2089	2.24	2089	2.24	100.00
	corr	31956	34.31	10820	11.62	33.86
UC2	*min_max*	3919	3.33	3919	3.33	100.00
	abs	2089	1.78	2089	1.78	100.00
	sbwlabel	52943	45.01	10831	9.21	20.46
	chgb	19077	16.22	12579	10.69	65.94
	cubic_conv	20613	17.52	12444	10.58	60.37
	median	23782	20.22	16160	13.74	67.95
	cubic	21728	18.47	15478	13.16	71.24
UC3	*min_max*	3943	2.72	3943	2.72	100.00
	filter	55921	38.60	15083	10.41	26.97
	clip	6988	4.82	3943	2.72	56.43
	inner	23302	16.09	15911	10.98	68.28
	mdiv	19346	13.35	11279	7.79	58.30
	sign	842	0.58	0	0.00	0.00
	rgb2yuv	51195	35.34	20816	14.37	40.66
	yuv2rgb	51696	35.69	33815	23.34	65.41

[a]Percentages wrt to the UC total area (baseline row of Table 4)

[b]Percentages wrt the kernel total area.

116619 μm^2 in area and 58.3 μW in power. The estimation error is now 0.9 % for the area and 1.3 % for the power.

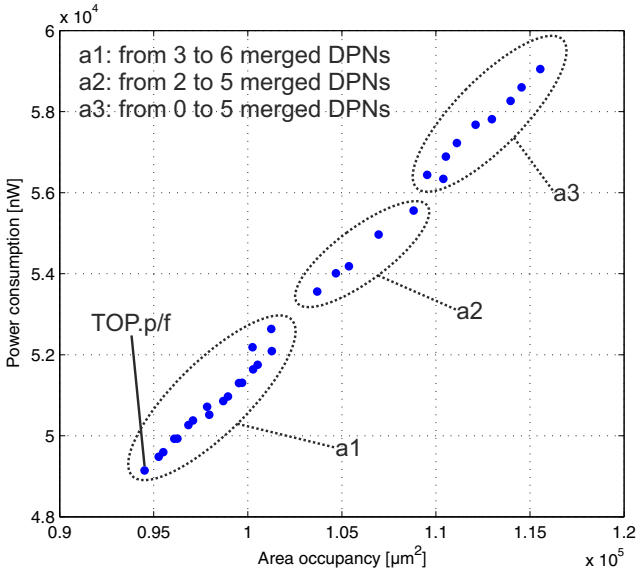


Figure 10 UC1 area vs power Pareto graph.

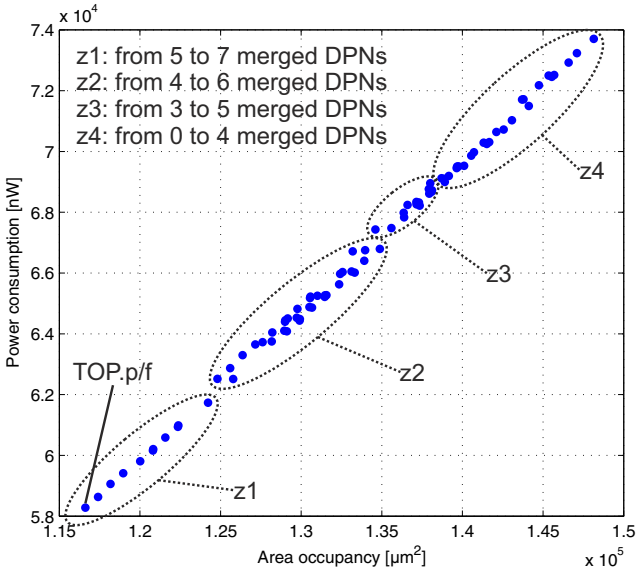


Figure 11 UC2 area vs power Pareto graph.

Figure 12 illustrates the area versus power Pareto graph of UC3. Cluster *d1* involves design points that keep in parallel the smallest DPNs (*clip*, *min_max*, *sign*), while *d2* groups the combinations where also the medium size kernels (*inner* and *mdiv*), but not the biggest ones (*rgb2yuv*, *yuv2rgb*, *filter*), are kept in parallel, one at a time. In cluster *d3* also *rgb2yuv* is among the kernels that may be kept in parallel and in *d4* *yuv2rgb* is added to the list. The last cluster, *d5*, contains the remaining design space points, including the non-reconfigurable one (which is not the worst). In frequency, the maximum achievable value is 202.1 MHz, fixed by the input DPNs. The frequency

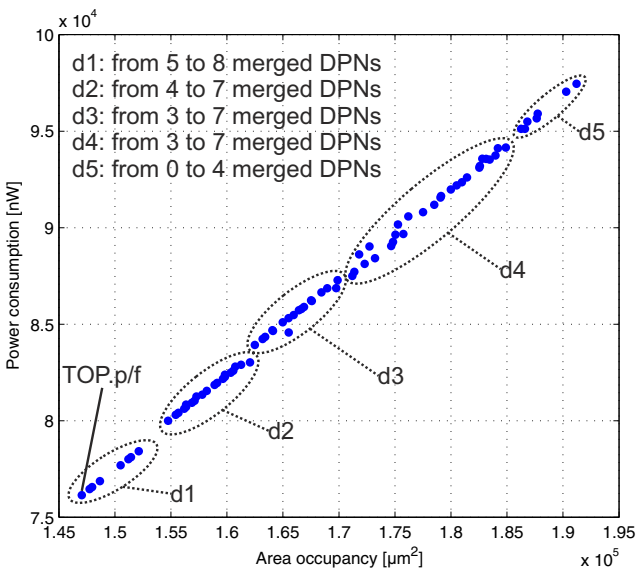


Figure 12 UC3 area vs power Pareto graph.

associated to the longest SBoxes chain is estimated in 1.52 GHz. Again, *TOP.p* and *TOP.f* are coincident and correspond to an *all-merged* solution in *d1* with 147045 μm^2 in area and 76.1 μW in power. The estimation error for this UC is 1.5 % for the area and 2.9 % for the power.

Summarizing, the proposed structural-level methodology, implemented within the *Topology Definer*, allowed the automatic identification of the best topology configurations within a large set of design points. The MDC merging process, leading to the generation of the multi-functional platform, is not decreasing the performances of the isolated input DPN applications, guaranteeing the same maximum achievable frequency. The *Logic Set Definer*, in the considered scenarios, can be then applied to a unique system configuration, *TOP.p/f*. The area and power estimations provided by the profiling phase, thanks to the dataflow modularity, reach very high accuracy rates, with errors below 3 %.

4.4 Dynamic Level Assessments

Given the optimal structural level topology configuration, to validate the effectiveness of the second phase of the proposed power management approach, we assembled four different designs for each UC:

- *UC - TOP.p/f* assembled with the baseline version of MDC, without the adoption of any particular power saving methodology.
- *UC_auto - TOP.p/f* assembled with the baseline version of MDC and implementing fine-grained (register level) clock gating with the automatic support offered by the adopted commercial synthesizer.
- *UC_cg - TOP.p/f* assembled with the clock gating extension of MDC.
- *UC_pg - TOP.p/f* assembled with the power gating extension of MDC.

These designs have been synthesized all at the same frequency, 200 MHz, targeting two different ASIC technology nodes: a 90 nm CMOS one (Sections 4.4.1 and 4.4.2) and a 45 nm one (Section 4.4.3). The frequency choice has been determined by the kernels maximum achievable frequency (202.1 MHz for UC1 and UC3).

Table 3 Use Case composition in terms of *LRs*.

Use Case	UC_cg	UC_pg
UC1	9	14
UC2	13	19
UC3	15	22

Table 4 90 nm ASIC synthesis results.

Design	Area			Power		
	$[\mu m^2]$	%[a]	%[b]	$[\mu W]$	%[a]	%[b]
UC1	93136	–	–	5030.54	–	–
UC1_auto	84925	−8.82	–	3553.15	−29.37	–
UC1_cg	93435	+0.32	+8.82	376.08	−92.52	−89.42
UC1_pg	165229	+77.41	+94.56	490.57	−90.25	−86.19
UC2	117627	–	–	6181.85	–	–
UC2_auto	105972	−9.91	–	4995.84	−19.19	–
UC2_cg	118007	+0.32	+11.35	436.05	−92.95	−91.27
UC2_pg	221589	+88.38	+109.09	619.69	−89.98	−87.60
UC3	144863	–	–	7784.02	–	–
UC3_auto	131052	−9.53	–	6343.41	−18.51	–
UC3_cg	145373	−0.35	+10.93	703.58	−90.96	−88.91
UC3_pg	257495	+77.75	+96.48	871.32	−88.81	−86.26

[a]Percentages wrt to the baseline design without power-management;

[b]Percentages wrt to the baseline design implementing fine-grained clock gating with SoC Encounter

Table 3 shows the composition of the different use cases in terms of *LRs*. The *UC_cg* designs, in all the considered use cases, present a smaller number of *LRs* to be managed. When power gating is required, the *Logic Set Definer* adds also asynchronous sets of SBoxes, resulting then in a larger *LRs* number to be supported and managed.

4.4.1 90 nm CMOS Technology: complete power gating support

This section discusses the synthesis results achieved with a 90 nm CMOS technology. Table 4 summarizes, for each design, area occupancy and power consumption. The reported data are comprehensive of the overhead due to the specific power saving technique, such as the *AND* gates for the MDC-based clock gating implementations and the power controller, the isolation and the retention cells for the MDC-based power gating ones. All the power results are calculated as the sum of single kernels contribution multiplied by the number of their occurrences (reported in Table 1 for each application), divided by the sum of the occurrences of all the kernels.[9] This weighed average is representative of

[9]With kernel contribution we mean the power consumption of the design when the considered kernel is enabled.

the designs power consumption during each use case typical execution.

As expected, both clock gating and power gating methodologies provide higher performance with respect to the baseline designs. For all the use cases the former are always above the 90 % of saving, while the latter reach more than the 88 %. MDC power saving methodologies, acting at the *LRs* level rather than at the register one, can achieve better results with respect to those achievable with fine-grain clock gating automatically applied by Soc Encounter. For all the use cases, with respect to *UC_auto*, the *UC_cg* designs are always above the 88 % of saving, while *UC_pg* ones reach more than the 86 %. In particular, for UC2 (Zoom) coarse-grained clock gating allows the saving of 91.27 % of the total power consumption with respect to the automatic fine-grained clock gating design. With respect to the area, the summary proposed in Table 4 confirms that power gating is an invasive technique, involving a severe area overhead. This latter is between 77.41 and 88.38 % with respect to the baseline designs and more than the 94 % with respect to the automatic fine-grained clock gating ones. This is due to the fact that, on the one hand, more *LRs* have to be handled with respect to a clock gating based methodology (as reported in Table 3) while, on the other, power gating implies adding and managing more components than simple clock gating strategies (as discussed in Section 2.3.2).

Table 5 better focuses on power results analyzing separately, for all the considered use cases and designs, the static power consumption and the dynamic one. As expected, power gating demonstrates higher performance in terms of static power consumption. It allows achieving 70–80 % of saving with respect to both the baseline design and the automatic fine-grained clock gating one. Such a benefit cannot be appreciated in the total power consumption since the static power amount is about the 1–10 % of dynamic one. In fact, 90 nm technologies are far from the physical limit where the static power contribution exceeds the dynamic one. A deeper insight on this aspect is provided in Section 4.4.3. In terms of dynamic power consumption MDC power extensions, the clock gating and the power gating ones, allow good saving percentages (more than the 80 % with respect to both the baseline and the auto clock gated designs). In comparison with the automatic fine-grained clock gating implemented by SoC Encounter, the coarse-grained approach proposed in this work is capable of getting rid of the clock tree power consumption; therefore, reaches considerably higher saving percentages. Clock gating does not act on the static power consumption by definition: *UC_cg* designs, due to the added clock gating modules, present a small overhead in terms of static power consumption. *UC_auto* seems to present a very small saving. This latter is due to the area optimizations, visible on Table 4, that SoC Encounter performs replacing the standard registers with dedicated clock gating cells.

Table 5 90 nm ASIC synthesis results: focus on static and dynamic power consumption.

Design	Static			Dynamic		
	$[\mu W]$	%[a]	%[b]	$[\mu W]$	%[a]	%[b]
UC1	47.96	–	–	4982.58	–	–
UC1_auto	47.51	−0.93	–	3505.64	−29.64	–
UC1_cg	49.07	+2.32	+3.28	327.02	−93.44	−90.67
UC1_pg	8.73	−81.80	−81.63	481.84	−90.33	−86.26
UC2	57.54	–	–	6124.31	–	–
UC2_auto	56.62	−1.61	–	4939.22	−19.35	–
UC2_cg	58.88	+2.33	+3.99	377.17	−93.84	−92.36
UC2_pg	13.24	−76.99	−76.62	606.45	−90.10	−87.72
UC3	73.92	–	–	7710.10	–	–
UC3_auto	72.39	−2.08	–	6271.03	−18.66	–
UC3_cg	75.27	+1.83	+3.99	628.31	−91.85	−89.98
UC3_pg	15.16	−79.50	−79.06	856.17	−88.90	−86.35

[a]Percentages wrt to the baseline design without power-management;

[b]Percentages wrt to the baseline design implementing fine-grained clock gating with SoC Encounter

Let's focus on UC2. Figures 13 and 14 respectively report the static and the dynamic contributes of the kernels involved in the considered UC. In particular, these figures show how the static and the dynamic power consumption of the designs change depending on the enabled kernel. The weighted mean of these data, calculated as described at the beginning of this section, gave the total UC2 numbers reported in Tables 4 and 5.

Focusing on the static consumption (reported in Fig. 13), as expected, power gating is extremely beneficial in reducing the static dissipation for all the kernels. The same does not apply for clock gating as previously discussed. In Fig. 13 it is possible to notice that one kernel, namely *sbwlabel*, is less positively affected by power gating than the others in terms of static consumption. This is the largest kernel (see Table 2) and, when it is active, a considerably high portion of the design is on. Therefore, by construction, it consumes more static power than the others. Moreover, the area overhead to manage power gating increases with the state retention cells number, which are potentially more for large kernels. Therefore, according to all these consid-

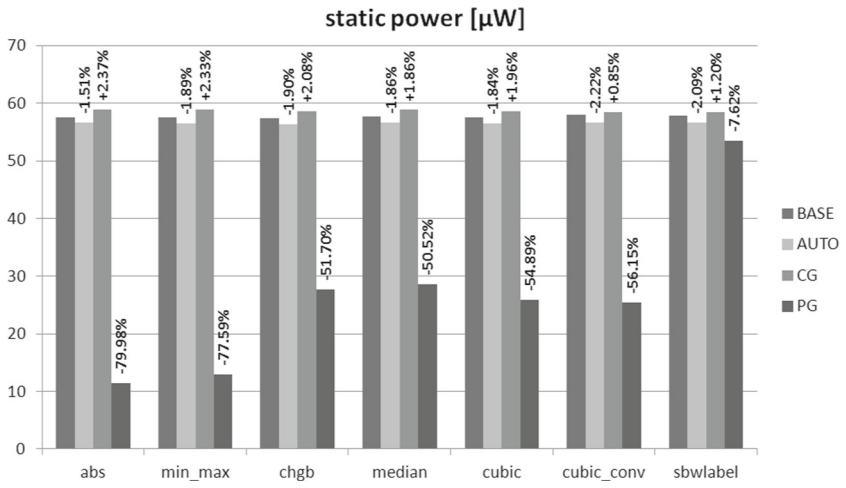


Figure 13 UC2: Static power consumption at 90 nm with state retention cells.

erations, *sbwlabel* benefits necessarily less than the other kernels from the power gating.

With respect to the dynamic power consumption (reported in Fig. 14) it is clearly visible that, within the MDC coarse-grained designs, clock gating achieves higher dynamic power savings than power gating, due to the high resource overhead of the latter technique that causes also additional switching activity on the design. The MDC-based designs where power optimization is enabled are capable of reaching higher performance than the register-level clock gated ones.

With respect to the automatic clock gating, whose saving percentages are always within 10–14 %, the MDC-based designs perform better. The best savings are reached for the smaller kernels (see Table 2), *abs* and *min_max*, where, by shutting down large portions of the design, coarse-grained clock gating and power gating allow saving more than 91 and 89 % respectively. The less positive effect is registered again with *sbwlabel* due to its size (it activates almost the 45 % of the entire design).

Summarizing the coarse-grained clock and power gating approaches proposed in this paper, based on the automatic model-based identification of homogeneous logic regions, are capable of achieving better performance than the classical fine-grained clock gating ones (normally implemented in commercial synthesizers), targeting a 90 nm CMOS technology. Strictly focusing on the MDC power extensions, according to the technology adopted in the discussed analysis, clock gating demonstrated to be more efficient than the proposed coarse-grained power gating implementation; despite this, power gating showed very promising results in terms of static power consumption reduction.

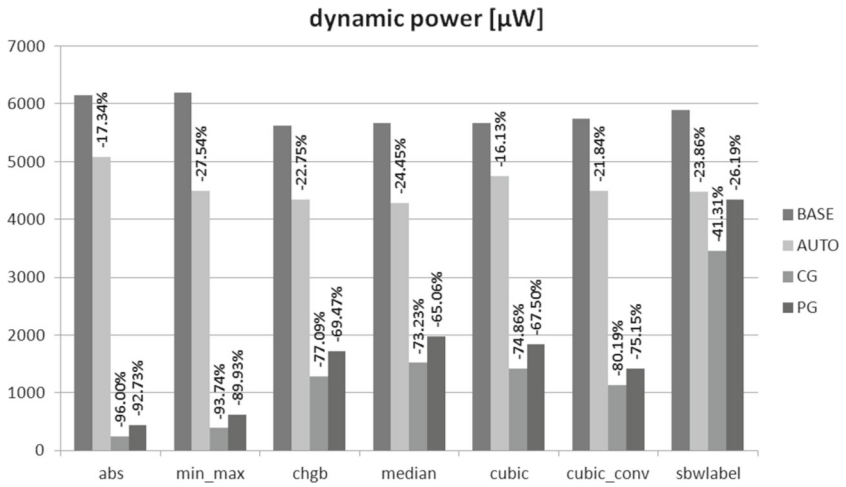


Figure 14 UC2: Dynamic power consumption at 90 nm with state retention cells.

4.4.2 90 nm CMOS Technology: application-specific power gating support

In this section, still targeting a 90 nm CMOS technology and the same use-cases, a different set of designs has been assessed. Given multi-functional scenarios such the targeted ones, it is not necessary to maintain the state of the registers when a certain *LR* is switched-off. Results are stored back to the main memory before a new computation is issued on the coprocessor, so that all the registers are simply overwritten. According to these considerations, it is not necessary to save the status of the registers prior to switch them off and, in turn, it is possible to get rid of the overhead due to the retention registers in the power gated designs. Results are in line with those discussed on the previous section: MDC-based clock gating and power gating methodologies provide higher performance and all the general considerations still hold.

Focusing on power gating, Table 6 summarizes, for each design, the synthesis results in terms of area occupancy and power consumption. The reported data, with respect to Table 4, do not include the contribution of the state retention cells. Getting rid of the state retention cells implies a considerably smaller area overhead (7–9 % in this second scenario versus 70–90 % of the previous one with respect to the baseline designs). Nevertheless, as it can be noticed, the overall power consumption does not change a lot, since with a 90 nm CMOS technology the static power consumption is still far smaller than the dynamic one, which has the largest impact on the power results.

This section demonstrated that, in those applications where the state preservation is not mandatory, it is possible to potentially benefit from the removal of the state retention cells. Nevertheless, despite the huge area saving, targeting a 90 nm CMOS technology this potential benefit does not turn into a real power saving, since the dynamic power consumption is still orders of magnitude larger than the static one.

Table 6 90 nm power gating (without state retention cells) ASIC synthesis results.

Design	Area			Power		
	$[\mu m^2]$	%[a]	%[b]	$[\mu W]$	%[a]	%[b]
UC1_pg	99707	+7.06	+17.40	431.57	−91.42	−87.85
UC2_pg	128383	+9.14	+21.14	549.35	−91.11	−89.00
UC3_pg	156230	+7.85	+19.21	769.71	−90.11	−87.87

[a]Percentages wrt to the baseline design without power-management;

[b]Percentages wrt to the baseline design implementing fine-grained clock gating with SoC Encounter

4.4.3 Preliminary Results Over a 45 nm Technology

The chosen technology influences the results achievable with the proposed strategies. The 90 nm technology is far from the physical point where the two contributes of the power, static and dynamic, are comparable. With the 90 nm technology the dynamic power is about two orders of magnitude larger than the static one. In such a situation clock gating results always beneficial with respect to power gating. It guarantees large dynamic savings with a negligible area overhead.

This section presents some preliminary results over a smaller technology, targeting 45 nm channel length cells, where the dynamic consumption is approximately just one order of magnitude larger than the static one. All the results reported hereafter refer to the application-specific power gating scenario discussed in Section 4.4.2, without the insertion of the state retention cells. Table 7 reports area occupancy and power consumption of the considered designs. The area overhead of the MDC-based coarse-grained solutions is in line to what we have seen in the previous paragraphs for the 90 nm technology: clock gating has a very low overhead (about 0.3 % with respect to the baseline design), while power gating is slightly more invasive (it reaches 10 % of area overhead for UC3). More interesting numbers are related to the power consumption:

Table 7 45 nm ASIC synthesis results.

Design	Area			Power		
	$[\mu m^2]$	%[a]	%[b]	$[\mu W]$	%[a]	%[b]
UC1	25384	–	–	1144.67	–	–
UC1_auto	23360	−7.97	–	804.82	−29.69	–
UC1_cg	25467	+0.33	+9.01	174.07	−84.79	−78.37
UC1_pg	27315	+7.61	+16.93	108.57	−90.52	−86.51
UC2	31564	–	–	1427.69	–	–
UC2_auto	28998	−8.13	–	1163.21	−18.53	–
UC2_cg	31674	+0.35	+9.23	209.39	−85.33	−82.00
UC2_pg	34737	+10.05	+19.79	142.98	−89.99	−87.71
UC3	39040	–	–	1789.00	–	–
UC3_auto	35844	−8.19	–	1456.28	−18.60	–
UC3_cg	39184	−0.37	+9.32	300.08	−83.23	−79.39
UC3_pg	42405	+8.62	+18.30	194.60	−89.12	−86.64

[a]Percentages wrt to the baseline design without power-management;

[b]Percentages wrt to the baseline design implementing fine-grained clock gating with SoC Encounter

Table 8 45 nm ASIC synthesis results: focus on static and dynamic power consumption.

Design	Static			Dynamic		
	$[\mu W]$	%[a]	%[b]	$[\mu W]$	%[a]	%[b]
UC1	102.86	–	–	1041.81	–	–
UC1_auto	102.05	−0.79	–	702.77	−32.54	–
UC1_cg	105.59	+2.65	+3.47	68.48	−93.43	−90.26
UC1_pg	18.35	−82.16	−82.02	90.22	−91.34	−87.16
UC2	126.54	–	–	1301.15	–	–
UC2_auto	121.41	−4.05	–	1041.79	−19.93	–
UC2_cg	129.85	+2.61	+6.95	79.54	−93.83	−92.36
UC2_pg	28.18	−77.73	−78.30	114.80	−91.18	−88.98
UC3	158.99	–	–	1630.02	–	–
UC3_auto	154.08	−3.09	–	1302.21	−20.11	–
UC3_cg	163.10	+2.59	+5.86	136.97	−91.60	−89.48
UC3_pg	30.47	−80.83	−81.32	164.13	−89.93	−87.40

[a]Percentages wrt to the baseline design without power-management;

[b]Percentages wrt to the baseline design implementing fine-grained clock gating with SoC Encounter

coarse-grained power gating designs achieve larger power saving percentages than the corresponding clock gating ones (86–87 % of the baseline design power consumption with respect to the 82 %). Automatic clock gating behavior does not change significantly between the two adopted technologies.

Power gating is more effective than clock gating on a 45 nm technology, despite the two contributes of the overall power consumption still differ for one order of magnitude. As detailed in Table 8: power gating presents again better performance in the static power reduction, while clock gating is again more efficient in dynamic power saving. Note that the savings achievable in terms of dynamic power consumption are extremely closed (91–93 % for clock gating and 87–88 % for power gating). Given this similarity and the fact that only power gating is capable of providing benefits in term of static power consumption (up to 77–82 %), the overall power consumption favours power gating solutions (as demonstrated in Table 7) for the targeted technology node.

Summarizing, this section confirms that the targeted technology may influence the designer in choosing the more appropriate power saving technique. Considering a technology where the dynamic consumption is the main term of the whole power, clock gating should be the better solution. By adopting a smaller technology, where static and dynamic power are comparable, power gating could guarantee better results.

5 Conclusions

In this paper we addressed the problem of providing high-level power management in multi-functional systems, targeting coarse-grained reconfigurable platforms running different applications on shared resources. Heterogeneous runtime reconfigurable systems are deployed. Two are the basic assumptions of this work: *1)* power reduction is of paramount importance in modern battery dependent designs; *2)* systems heterogeneity and complexity is so challenging that manual strategies require too much effort to be considered still viable.

The proposed two-steps approach extends a dataflow-based design framework for coarse-grained reconfigurable designs, the Multi-Dataflow Composer tool, designed within MPEG-RVC research studies. The importance of the proposed power-aware extension of the MDC tool is mainly related to the fact that, within the MPEG-RVC framework, power consumption is still an open issue and, as far as we know, power estimation/management strategies are currently under investigation. Our solution combines structural and dynamic strategies applied automatically at the modeling level.

Given different image processing application scenarios, the proposed flow has been adopted to assemble the reconfigurable computing core of different memory mapped coprocessors. We demonstrated how topology may affect power in coarse-grained architectures. The implemented profiling and exploration strategy has been capable of finding the optimal system topology through very accurate estimations. On the other hand, looking at the dynamic behaviour of the reconfigurable platform, we demonstrated that it is possible to partition, since the first stage of the design flow, the architecture in order to automatically implement different power saving strategies on the hardware platform with very limited designers intervention. MDC power extensions are capable of providing both coarse-grained clock gating and power gating support. For multi-functional reconfigurable system, we demonstrated that: 1) the automatic fine-grained clock gating techniques (as those available on commercial synthesizers) are not sufficient for power saving purposes; 2) according to the adopted technology, the applications characteristics and the constraints to be met (area/power/frequency), it is possible to opt for one of the two supported techniques to achieve optimal performances.

As future evolution of this work, our effort will focus on the development of heuristic methods to speed-up the topology optimization phase, even addressing larger scenarios. Moreover, the smarter combination of the proposed coarse-grained power saving strategies will be imple-

mented, to selectively decide which one between clock and power gating better suites to the considered logic region.

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

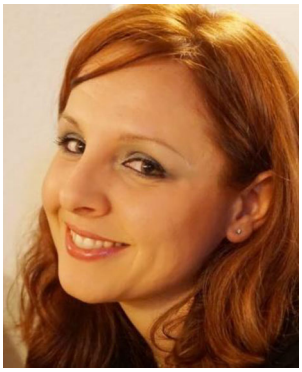
Appendix A: MDC Generated CPF

```
#####################################################
# CPF file automatically generated by:
# Multi-Dataflow Composer tool
#####################################################
set_cpf_version 1.1
set_hierarchy_separator /

#####################################################
# Design part of the CPF (Power Intent part)
#####################################################
#identify the design for which the CPF is created
set_design top_module

# create power domains
create_power_domain -name PDdef -default # (inst_C sbox_2)
create_power_domain -name PD1 -instances {inst_B}\
-shutoff_condition {powerController_0/pw_switch_en1}\
-base_domains {PDdef}
#repeat for all switchable domains


# create nominal conditions
create_nominal_condition -name off -voltage 0
create_nominal_condition -name on -voltage 1.1

# create power modes
create_power_mode -name PMdef\
-domain_conditions {PDdef@on PD1@on PD3@on PD4@on PD5@on}\
-default
create_power_mode -name PM1\
-domain_conditions {PDdef@on PD1@on PD4@on}
#repeat for all power modes

# associate library sets with nominal conditions
update_nominal_condition -name on -library_set set1_wc

# create rules for isolation logic insertion
create_isolation_rule -name iso1 -from PD1\
-isolation_condition {powerController_0/iso_en1}\
-isolation_output high
#repeat for all necessary isolation rules

# create rules for state retention insertion
create_state_retention_rule -name st1 -domain PD1\
-restore_edge {powerController_0/rstr_en1}\
-save_edge {powerController_0/save_en1}

#repeat for all necessary retention rules

# indicate when the power intent for the design ends
end_design
```

References

1. Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., & Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Proceedings of the 38th annual international symposium on computer architecture (ISCA)* (pp. 365–376).
2. Taylor, M.B. (2012). Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *ACM/EDAC/IEEE Design automation conference (DAC)* (pp. 1131–1136).
3. Herbert, S., & Marculescu, D. (2007). Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ACM/IEEE International symposium on low power electronics and design (ISLPED)* (pp. 38–43).
4. Eyerman, S., & Eeckhout, L. (2011). Fine-grained DVFS using on-chip regulators. *ACM Transactions on Architecture and Code Optimization (TACO)*, *8*(1), 1–24.
5. Arora, M., Manne, S., Eckert, Y., Paul, I., Jayasena, N., & Tullsen, D.M. (2014). A comparison of core power gating strategies implemented in modern hardware. In *ACM SIGMETRICS / International conference on measurement and modeling of computer systems (SIGMETRICS)* (pp. 559–560).
6. Jeff, B. (2012). Advances in big.little technology for power and energy savings. In *ARM White paper*.
7. Power Forward Initiative (2009). A Practical Guide to Low Power Design.
8. Wingard, D. (2013). Noc power-management advantages. In *Keynote talks at IP-soc conference and exibition*.
9. Sau, C., Raffo, L., Palumbo, F., Bezati, E., Casale-Brunet, S., & Mattavelli, M. (2014). Automated design flow for coarse-grained reconfigurable platforms: an rvc-cal multi-standard decoder use-case. In *Conference on embedded computer systems: architectures, Modeling, and Simulation (SAMOS XIV), IEEE* (pp. 59–66).
10. Palumbo, F., Carta, N., Pani, D., Meloni, P., & Raffo, L. (2014). The multi-dataflow composer tool: generation of on-the-fly reconfigurable platforms. *Journal of Real-Time Image Processing (JRTIP)*, *9*(1), 233–249.
11. Carta, N., Sau, C., Pani, D., Palumbo, F., & Raffo, L. (2013). A Coarse-Grained reconfigurable approach for Low-Power spike sorting architectures. In *IEEE/EMBS International conference on neural engineering (NER)* (pp. 439–442).
12. Carta, N., Sau, C., Palumbo, F., Pani, D., & Raffo, L. (2013). A Coarse-Grained reconfigurable wavelet denoiser exploiting the Multi-Dataflow composer tool. In *Conference on design and architectures for signal and image processing (DASIP)* (pp. 141–148).
13. ISO/IEC 23001-4 (2009). MPEG-part 4: Codec configuration representation.
14. ISO/IEC 23002-4 (2010). MPEG video tech.—Part 4: Video tool library.
15. Palumbo, F., Sau, C., & Raffo, L. (2014). Coarse-grained reconfiguration: dataflow-based power management. *In IET Computers & Digital Techniques*, *9*(1), 36–48.
16. Palumbo, F., Sau, C., & Raffo, L. (2014). Power-awarness in coarse-grained reconfigurable designs: a dataflow based strategy. In *IEEE Workshop on signal processing systems (siPS), IEEE* (pp. 1–6).
17. Kahn, G. (1974). The semantics of a simple language for parallel programming. In *International Conference on Information Processing* (pp. 471–475).
18. Lee, E.A., & Parks, T. (1995). Dataflow process networks. In *Proceedings of the IEEE* (pp. 773–799).

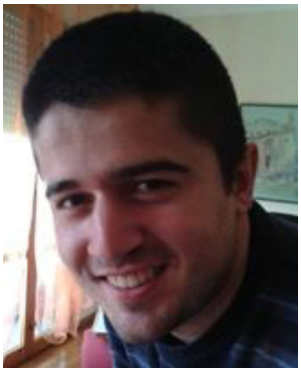
19. Eker, J., & Janneck, J.W. (2003). Cal Language Report Specification of the Cal Actor Language. Technical report EECS Department, University of California, Berkeley.
20. Open RVC-CAL Compiler. http://orcc.sourceforge.net/.
21. Bezati, E., Mattavelli, M., & Janneck, J. (2013). High-Level Synthesis of dataflow programs for signal processing systems. In *International symposium on image and signal processing and analysis (ISPA)* (pp. 750–754).
22. Casale-Brunet, S., Mattavelli, M., & Janneck, J.W. (2013). Turnus: a design exploration framework for dataflow system design. In *International symposium on circuits and systems (ISCAS)* (pp. 654–654).
23. Carta, S.M., Pani, D., & Raffo, L. (2006). Reconfigurable coprocessor for multimedia application domain. *Journal of VLSI Signal Processing Systems*, *44*, 135–152.
24. Kumar, V.V., & Lach, J. (2006). Highly flexible multimode digital signal processing systems using adaptable components and controllers. *EURASIP Journal on Applied Signal Processing*, *14*, 1–9.
25. Palumbo, F., Pani, D., Manca, E., Raffo, L., Mattavelli, M., & RVC, G.Roquier. (2010). A multi-decoder CAL Composer tool. In *Conference on design and architectures for signal and image processing (DASIP)* (pp. 144–151).
26. Palumbo, F., Carta, N., & Raffo, L. (2011). The Multi-Dataflow Composer tool: a runtime reconfigurable HDL platform composer. In *Conference on design and architectures for signal and image processing (DASIP)* (pp. 178–185).
27. Wipliez, M., Siret, N., Carta, N., Palumbo, F., & Raffo, L. (2012). Design IP faster: introducing the C˜high-level language. In *IP-SOC: IP-Embedded System Conference and Exhibition*.
28. Puri, R., Stok, L., & Bhattacharya, S. (2005). Keeping hot chips cool. In *Design automation conference (DAC)* (pp. 285–288).
29. Casale-Brunet, S., Bezati, E., Alberti, C., Mattavelli, M., Amaldi, E., & Janneck, J. (2013). Partitioning and optimization of high level stream applications for multi clock domain architectures. In *IEEE Workshop on Signal Processing Systems (siPS)* (pp. 177–182).
30. Ren, R., Wei, J., Martínez, E.J., González, M.G., Álvaro, C.S., & del Oso, F.P. (2014). A PMC-driven methodology for energy estimation in RVC-CAL video codec specifications. *Signal Processing: Image Communication*, *28*(10), 1303–1314.
31. Schmidt, A.G., Steiner, N., French, M., & Sass, R. (2012). Hwpmi: an extensible performance monitoring infrastructure for improving hardware design and productivity on fpgas. *International Journal of Reconfigurable Computing*, *2012*, 1–12.
32. Lucarz, C., Roquier, G., & Mattavelli, M. (2010). High level design space exploration of RVC codec specifications for multi-core heterogeneous platforms. In *Conference on design and architectures for signal and image processing (DASIP)* (pp. 191–198).
33. Rahman, A.A.A., Thavot, R., Casale-Brunet, S., Bezati, E., & Mattavelli, M. (2012). Design space exploration strategies for FPGA implementation of signal processing systems using CAL dataflow program. In *Conference on design and architectures for signal and image processing (DASIP)* (pp. 1–8).
34. Meloni, P., Pomata, S., Tuveri, G., Secchi, S., Raffo, L., & Lindwer, M. (2012). Enabling fast ASIP design space exploration: an fpga-based runtime reconfigurable prototyper. *VLSI Design*.
35. Palumbo, F., Sau, C., & Raffo, L. (2013). DSE And profiling of Multi-Context Coarse-Grained reconfigurable systems. In *International symposium on image and signal processing and analysis (ISPA)* (pp. 744–749).
36. Zhang, Y., Roivainen, J., & Mammela, A. (2006). Clock-gating in FPGAs: a Novel and Comparative Evaluation. In *EUROMICRO Conference on conference on digital system design: architectures, Methods and Tools (DSD)* (pp. 584–590).
37. Bezati, E., Casale-Brunet, S., Mattavelli, M., & Janneck, J.W. (2014). Coarse grain clock gating of streaming applications in programmable logic implementations. In *Proceedings of the 2014 electronic system level synthesis conference (ESLsyn)* (pp. 1–6).
38. Silicon Integration Initiative (2014). Si2 Common Power Format Specification™ - Version 2.1.



Francesca Palumbo is currently an assistant professor at the University of Sassari, within the Information Engineering unit. She received her "laurea degree" in Electronic Engineering in 2005 at the University of Cagliari, the Master Advanced in Embedded System Design in 2006 at the Advanced Learning and Research Institute of the University of Lugano and the Ph.D. in Electronic at the University of Cagliari. Her research focus is related to reconfigurable system design and development of code generation tools for advanced reconfigurable hardware architectures. She has been involved in the definition of automatic tools to support the MPEG Reconfigurable Video Coding standard. On these topics she closely cooperate with the Institut d'Electronique et de Telecommunications de Rennes and the École polytechnique fédérale de Lausanne.

In the field of dataflow-based programming and dataflow-based tools she has received a Best Paper Award at the Conference on Design and Architectures for Signal and Image Processing 2011 for the work entitled "The Multi-Dataflow Composer tool: A runtime reconfigurable HDL platform composer".

Tiziana Fanni is a Ph.D student at the Department of Electrical and Electronic Engineering - University of Cagliari (Italy). She received her degree in Electronic Engineering in 2014 at the University of Cagliari. In June 2014 she started a 1 year research grant related to power saving methodologies in dataflow-based reconfigurable platforms. Her main research focus is related to reconfigurable systems design and development of code generation tools for low power reconfigurable hardware architectures.



Carlo Sau is currently a Ph.D. student at the University of Cagliari. He received his degree in Electronic Engineering in 2012 at the University of Cagliari. In October 2012, he started a 3 months research grant involving automated methodologies for dataflow-based reconfigurable platforms generation. His main research focus is related to reconfigurable system design and development of code generation tools for advanced reconfigurable hardware architectures.

Paolo Meloni is currently assistant professor at the Department of Electrical and Electronic Engineering (DIEE) in the University of Cagliari. In October 2007 he received a Ph.D. in Electronic Engineering and Computer Science, presenting the thesis "Design and optimization techniques for VLSI network on chip architectures". His research activity is mainly focused on the development of advanced digital systems, with special emphasis on the application-driven design of multi-core on-chip architectures. He is author of a significant record of international research papers and tutor of many bachelor and master students' thesis in Electronic Engineering. He is teaching the course of Embedded Systems at University of Cagliari and is currently part of the technical board and acting as work-package leader in the research projects ASAM (www.asamproject.org) and MADNESS (www.madnessproject.org).