LP-HLS: Automatic Power-Intent Generation for High-Level Synthesis Based Hardware Implementation Flow

Article *in* Microprocessors and Microsystems · February 2017

DOI: 10.1016/j.micpro.2017.02.002

CITATIONS
2

READS
109

4 authors:

Affaq Qamar
Abasyn University
13 PUBLICATIONS **14** CITATIONS
SEE PROFILE

F. B. Muslim
Politecnico di Torino
7 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Javed Iqbal
Politecnico di Torino
8 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Luciano Lavagno
Politecnico di Torino
202 PUBLICATIONS **7,176** CITATIONS
SEE PROFILE

Microprocessors and Microsystems 50 (2017) 26–38

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

LP-HLS: Automatic power-intent generation for high-level synthesis based hardware implementation flow

Affaq Qamar[a,*], Fahad Bin Muslim[b], Javed Iqbal[b], Luciano Lavagno[b]

[a] Department of Electrical Engineering, Abasyn University, Pakistan
[b] Department of Electronics and Telecommunication (DET), Politecnico di Torino, Italy

ARTICLE INFO

Article history:
Received 5 May 2016
Revised 21 November 2016
Accepted 6 February 2017
Available online 10 February 2017

Additional Key Words and Phrases:
High-level synthesis
RTL
Design space exploration
Common power format
Low power designs
Design automation
EDA methodologies

ABSTRACT

The abstraction level for digital designs is rising from Register Transfer Level (RTL) to algorithmic untimed or transaction-based, followed by an automated high-level synthesis (HLS) flow. However, it is still a significant challenge for chip architects and designers to describe low-power design decisions at the system-level. Nowadays, low power design techniques for digital blocks are applied at RTL and there exists no commercial tool or methodology that can automatically derive the power intent from the system-level description. The process requires considerable amount of human intervention and various low-level details that are needed to implement low power schemes at RTL. This research aims to integrate low power techniques, specifically Power Shut-Off (PSO), within a model-based hardware flow and to derive an automated Low Power-High Level Synthesis (LP-HLS) methodology. The methodology aims at minimizing the design effort for low power design by deriving low-level power intent automatically for model-based designs, while using high-level synthesis to achieve a broad set of target system implementations. LP-HLS uses set of pragmas and a directive file to derive power intent information. To illustrate the methodology, three model designs, ranging from simple designs to medium complexity hardware accelerators, are considered. Finally, the power saving results for the design scenarios validate the effectiveness of our LP-HLS methodology.

1. Introduction

System-on-chip (SoC) designs are becoming increasingly heterogeneous as they combine multicore architectures with a variety of hardware accelerators to carry out dedicated computational tasks. These hardware accelerators offer several orders-of-magnitude higher power and timing efficiency than a corresponding software implementation [1]. However, the presence of accelerators aggravates the complexity of SoC design. Moreover, digital designers aim at developing designs that optimize both timing and power, which generally are two conflicting performance objectives. Pareto-optimality can be comfortably achieved only if power specification is considered at the system-level. This is because major architectural decisions related to cost (area), performance and energy utilization can be made only at higher abstraction levels [2]. For example, parallelism versus voltage and clock frequency is much easier to trade off at system-level. Early consideration of effective power management helps relieve the power bottlenecks of today's VLSI design, enabling sustained high-performance operation with desired power consumption levels.

1.1. Model-based designs

As far as behavioral description for the hardware design is concerned, the abstraction level is rising from RTL to algorithmic untimed or transaction-based, followed by an automated high-level synthesis (HLS) flow [3]. HLS takes as input the model-based description of the design, specified in some high-level language such as C, C++, SystemC or Simulink, and synthesizes it to generate RTL. By elaborating different sets of constraints, HLS tools allow designers to evaluate multiple implementation alternatives, a process known as Design Space Exploration (DSE) [26,27]. DSE with HLS is already a major leap from DSE with Logic Synthesis, since the former can be achieved by simply changing HLS tool directives, while the latter usually requires one to manually change a detailed hardware description expressed in the form of Verilog or VHDL. Such hardware descriptions at a lower abstraction level are often tried only for one or two architectural options, due to the much slower design and verification cycle [34]. Also, high design productivity requires that complex SoCs consist of 90% reused components [4]. This in turn requires soft IP components, which are designed once

* Corresponding author.
E-mail address: affaq.qamar@abasyn.edu.pk (A. Qamar).

http://dx.doi.org/10.1016/j.micpro.2017.02.002
0141-9331/

using high-level languages and implemented into various instances to meet the changing design requirements [5].

1.2. Low power design flow

As far as functional description of hardware is concerned, it has evolved to the extent that model-based design approaches using automated HLS flows have gained popularity within the design community [6]. However, it is still a significant challenge for chip architects and designers to describe low-power design decisions at the system-level. This is because; system architects have little or no visibility of the lower-level details that are needed to implement low power schemes at RTL [14]. Similarly, a digital back-end designer needs to interact intensively with the system architect as well as with the verification team to formulate an appropriate hardware platform and a low power scheme in order to meet the requirements. This process still involves a considerable amount of human intervention, because nowadays the low power techniques for a digital system are applied at register transfer level (RTL), and thus the designer needs to intimately know the RTL code in order to successfully apply them. These techniques include supply voltage and clock control technologies, such as power shut-down, clock gating, dynamic voltage and frequency scaling, and adaptive voltage scaling [7]. The power shut-down technology is especially vital to leakage power reduction for battery-driven devices.

1.3. Contribution

This research aims to integrate the aforementioned low power techniques within a model-based hardware flow and to derive an automated Low Power High-Level Synthesis (LP_HLS) methodology. This methodology enables the specification of both the behavioral functionality and the power intent at a level of abstraction higher than RTL and the use of HLS tools at the front-end of the ensuing design flow.

The behavioral description of the design can be captured using SystemC or C/C++, although in this paper we use SystemC for the sake of illustration. In our proposed flow, the *power intent*, i.e. the set of power-saving techniques to be used and all the information that is needed to implement them, is automatically derived from the system-level design using a set of pragmas and a directives file. Combining this information with technology dependent parameters, a tool that we developed derives low power intent written in the Common Power Format (CPF), or Unified Power Format (UPF), which are widely used standards supported by several commercial EDA tool vendors.

In this work, we use CPF for the sake of illustration, and consider power shut-off (PSO) along with clock gating (CG) to achieve power saving [10,35]. To illustrate the methodology, we use three examples ranging from simple designs to medium complexity hardware accelerators. These include a 32 bit Ripple-Carry Adder (RCA), a general purpose ALU performing arithmetic and logical operations, and an IDCT block often used in image and video processing. Our methodology aims at minimizing the design effort for low power design by deriving low-level power intent automatically for model-based designs, while using high-level synthesis to achieve a broad set of target system implementations. In particular, we propose:

- A generic power management module description at the system-level, to specify the design context of a hardware block;
- A tool that, for a given system design context automatically generates the low power design directives needed to implement the PSO technique in the back-end flow.
- A novel LP-HLS methodology which uses an HLS tool as a front-end environment and integrates it with a standard back-end digital low-power flow.

The goal is not to show an improvement in terms of power savings with respect to the standard low-power RTL-to-GDSII flow. The goal is to show that the same improvement can be achieved by starting from system level (e.g. from a SystemC, C or C++ model), with very little effort, and without requiring the designer to understand the details of the automatically generated RTL code.

The rest of the paper is organized as follows. Section 2 briefly presents the related work. Our LP-HLS methodology is explained in detail in Section 3. Section 4 lists our design test cases along with the description of a sample power management module generating the signals necessary to power gate our design test cases. The results are presented in Section 5 while the work is concluded in Section 6.

2. Related work

Most work on low power techniques using CPF and UPF focuses on the RTL, rarely considering system-level implications. An example is [9,35], which focuses on the manual description of the power intent using CPF for RTL implementations generated using HLS. In this work, on the other hand, we propose a methodology to *automatically generate the CPF power specification from a high-level model* just like an HLS tool generates the RTL from a behavioral description.

Specifying power intent at the system-level results in a significant reduction in the design effort. A system-level model is much easier to understand than RTL due to its higher level of abstraction and can result in significant reduction of design effort if power optimization is considered at the system-level. Furthermore, the long simulation and analysis times required while considering power at lower levels of abstraction can be significantly reduced by considering power at the system-level [38]. Moreover, adding the power optimization logic at the system-level rather than at RTL makes functional verification easier, thanks to the design-specific SystemC test bench.

Benini et al. [10] and Lin et al. [11] suggest various energy optimizations starting from software down to circuit-level for electronic systems. Their strategy starts from power optimizations at system-level including energy-aware task scheduling, hardware/software partitioning, power aware architectures using dynamic power management (DPM) policies and code morphing. Similarly, they emphasize adopting low power schemes such as MSV, PSO, CG, and DVFS at the RTL, while using gates with varying transistor widths to achieve additional energy saving at the gate-level.

Schirrmeister [12], and Benini and Micheli [13] provide a review of the different abstraction levels for energy measurement and estimation as well as common techniques to optimize a design for low power above RTL. They use a typical JPEG decoder as a test case. The authors advocate the use of a system-level solution early in the design cycle for data-flow dominated blocks and their associated memories.

Zhang et al. [14] argues that in order to meet strict power requirements, modern day designers may still have to perform manual optimizations on an RTL design described using Verilog or VHDL by applying numerous low power techniques considering functional, structural, temporal and spatial information together. It is extremely difficult to achieve these goals manually in such a complex multi-dimensional space within a limited time. In addition, power scaling requires designers to evaluate and optimize the system architecture as early as possible in the design flow. Certainly, the solution is to raise the level of abstraction beyond RTL to achieve faster power optimization and to use automated RTL synthesis tools.

Thus, there exists a general agreement among the designers regarding the significance of system-level power optimization tech-

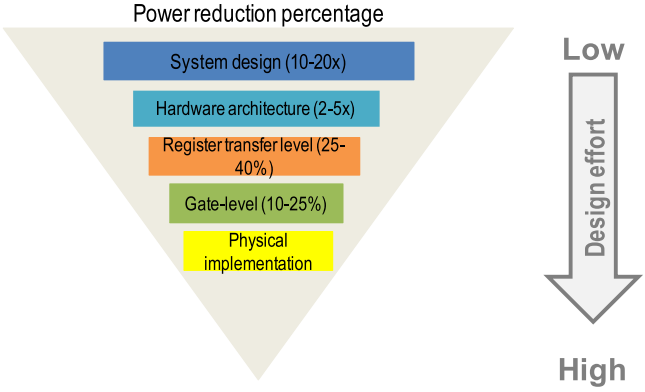

Fig. 1. Power reduction as a function of various stages of design cycle.

niques [29,36]. Commercial efforts in producing pre-RTL power optimization tools have resulted in tools such as Vista Architect from Mentor Graphics and Chip vision's PowerOpt, Two new working groups were formed by IEEE at the end of 2014 to standardize system-level power modeling for SoC devices [15]. The prospect of considering power intent at the system-level and applying it in combination with HLS thus is an important but rather sparsely explored area and hence a theme of this work.

3. LP-HLS methodology

The earlier energy savings are considered in the design cycle, the more savings can be achieved, as shown in Fig. 1. This is because architectural decisions related to cost (area), performance and energy utilization are made at this level [37]. Considering power optimization at a lower level of abstraction, as is the case in a standard ASIC design flow, would require significant iterative changes in the overall design architecture and hence a larger design effort, as depicted in Fig. 1.

Hence, the prospect of a completely automated low power flow, where both the logic and power intent can be incorporated into the design at the system-level, seems very promising. In this section, we present a complete description of our LP-HLS methodology, which comprises an HLS flow, a CPF generator to automatically produce CPF power intent, and the final integration into the back-end flow.

3.1. Model-based hardware design

The overview of the proposed design flow is shown in Fig. 2. The power intent is derived by extracting the design related information from the system-level description. This information is read together with the power rules specification for the PSO, and the technology related parameters to automatically generate the CFP file. The hardware description for low power design is then integrated with the RTL during the later stages of back-end flow.

Luckily, even when the RTL is generated automatically using HLS from the system-level code written in SystemC, the naming convention of the design hierarchy, i.e. instances, signals and ports is preserved, or names for new objects (e.g. fields of SystemC struct-typed I/O signals) can be easily derived automatically from the high-level model. Beginning with the high-level model of a design, the HLS tool executes several tasks, namely allocation, scheduling and binding, to automatically generate an RTL implementation. These tasks can be performed either simultaneously or sequentially. However, since performing them simultaneously would be too time-consuming, the latter option is typically chosen [8].

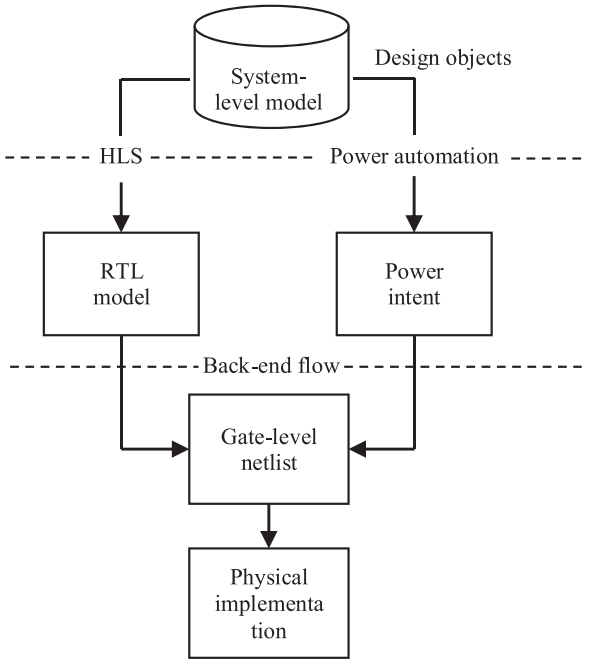

Fig. 2. Overview of the low power design model.

In our flow, the power intent is also derived from annotations in the SystemC code and a design configuration file that refers only to high-level information. This makes it easier for the design architect to select among the underlying power optimization choices, thanks to the higher level of abstraction.

3.2. CPF generation tool

Our automated CPF generation tool relies upon the use of #pragma directives in the system-level design file/s as well as a designer-written intent specification file, to identify power related information. Fig. 3 shows the general structure of the tool.

The technology specification (Tech. spec. in the Fig. 3) file, as the name implies, contains technology related information using directly the corresponding CPF syntax. In particular, it contains; (i) the set of libraries (typically the worst, best and nominal cases), (ii) the information about the power nets that will be created during physical design, and (iii) the nominal conditions (e.g. voltage levels for various power nets) which will be used by the various power modes afterwards.

The Intent Specification file contains pragma directives for the power rules that are needed to successfully implement PSO strategy. These rules depend upon design related parameters. This includes rules such as operating corners for multi-mode-multi-corner (mmmc) analysis, analysis views for power domains and the definition of power rules such as isolation, state-retention and power switch rules.

The CPF generator needs to extract from the system-level model file information such as, the module name of the switchable power domain(s), and the instance name of the signal that defines the shut-off condition, and that will be used to drive the power switches. This is done by inserting a #pragma directive before the respective instance name. The tool searches for those unique pragmas, creates token strings for the instances and stores the information for later processing. The name of the power domain can either be assigned by the designer (e.g. MY_DOMAIN), or be generated by the parser (e.g. switchable_domain_<n>).

Once all the information from the input files is gathered into custom data structures, the tool generates power rules in CPF for-

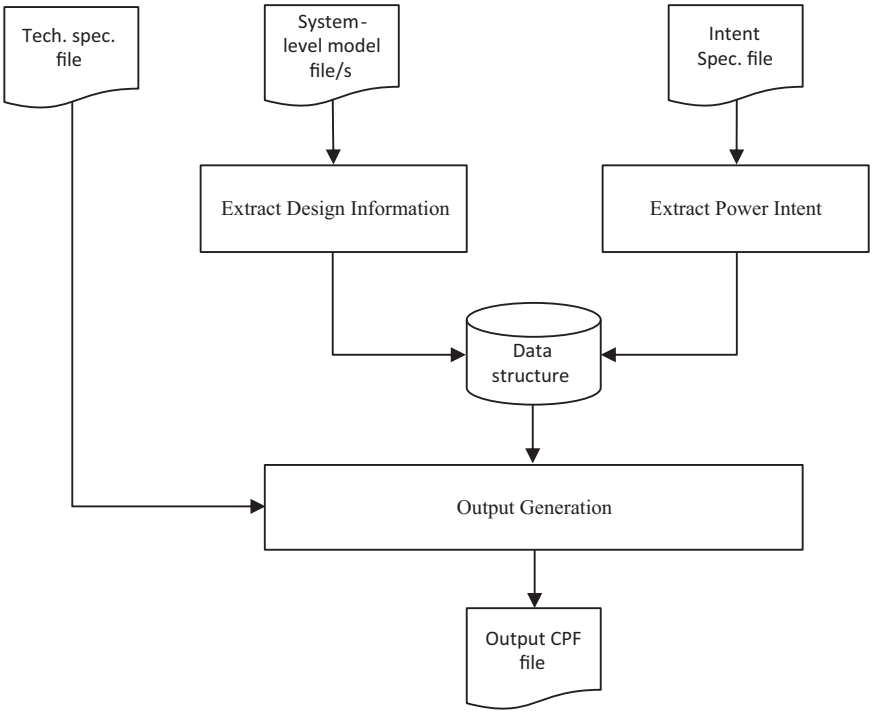

Fig. 3. Basic structure of CPF generation tool.

mat corresponding to each pragma directive, using the information from the configuration files.

The steps executed by our tool to support the CPF-based system-level design flow are depicted as a flow chart in Fig. 4. The tool begins by checking for the presence of a technology library (read from a Technology Specification file) and proceeds with the rest of the flow only if it exists. The system-level code is then read and scanned for the #pragma directives to extract the design-related information, e.g. modules belonging to the switchable power domain, the signal that has to be used to trigger the power gating etc. The power intent (i.e. the directives and rules required to implement PSO) is then extracted by reading the Intent Specification file. These low power directives are then checked against a low power template library for correctness. Low power rules and constraints are then derived by also utilizing the information extracted from the system-level design code. Finally, the output CPF file is generated to implement PSO later during logic synthesis.

Fig. 5 shows an example of how two pragma directives in the SystemC source and some information in the specification file are combined to generate a simple CPF output file.

3.3. LP-HLS flow

After discussing the model-based design approach using HLS and describing our tool that automates the power intent generation, we now discuss the complete LP-HLS flow.

Fig. 6 summarizes the flow of operations that are required to go from a system-level specification to a power-optimized gate-level implementation, using the proposed Low Power High-Level Synthesis (LP-HLS) methodology and a standard RTL-to-gates flow. Most of the blocks in the figure depict the steps involved in a state-of-the-art HLS-based SystemC (or C/C++) to GDSII design flow. Only a few blocks (shaded in blue) have been added to show the steps that were *added by our methodology and tool* to automatically infer the power intent for a specific design. So these boxes depict our contribution to advance beyond the state of the art. Depending on the application, different constraints (e.g., performance, area cost, and power) must be satisfied during the various phases of the flow. Macii et al. [16] suggests that, "when the target is a low-power application, the search for the optimal solution must include, at each level of abstraction, a design improvement loop". In such a loop, a power analyzer/estimator (shown in gray in Fig. 6) ranks the various design, synthesis, and optimization options, and thus helps in selecting the one that is potentially more effective from the power standpoint. However, this methodology requires the availability of power estimators, as well as synthesis and optimization tools, which provide accurate and reliable results at various levels of abstraction.

While adapting a purely algorithmic model to be the input of HLS, the designer must instantiate a Power Management Block (PMB) that decides exactly when the power can be switched ON or OFF, and if necessary keeps the design waiting (e.g. by gating the clock) while the power stabilizes. Of course, this PMB must be part of the always ON domain. Thereafter, the RTL is generated through HLS by performing the necessary steps i.e. by specifying the target technology, the micro-architecture choices and the scheduling constraints. In parallel to HLS, our tool processes the system-level power intent and generates the CPF file, as described above.

The switching activity information for accurate power analysis is extracted by simulating the RTL with the original SystemC test-bench, by using the automatically generated SystemC wrapper. Power aware logic synthesis is then performed by first reading the target libraries, as specified in the CPF file, and by enabling the application of coarse-grained clock gating logic (both in HLS and logic synthesis).

As in the standard power-aware flow, the RTL design is then read and elaborated, followed by technology mapping of the cells to be used as clock-gated integrated cells (CGIC). The power intent is then read, followed by setting the timing constraints and synthesizing the design. The switching activities are annotated and the power intent is applied, followed by the verification of the power structure to check if the low power cells have been correctly inserted in the design according to the rules specified in the

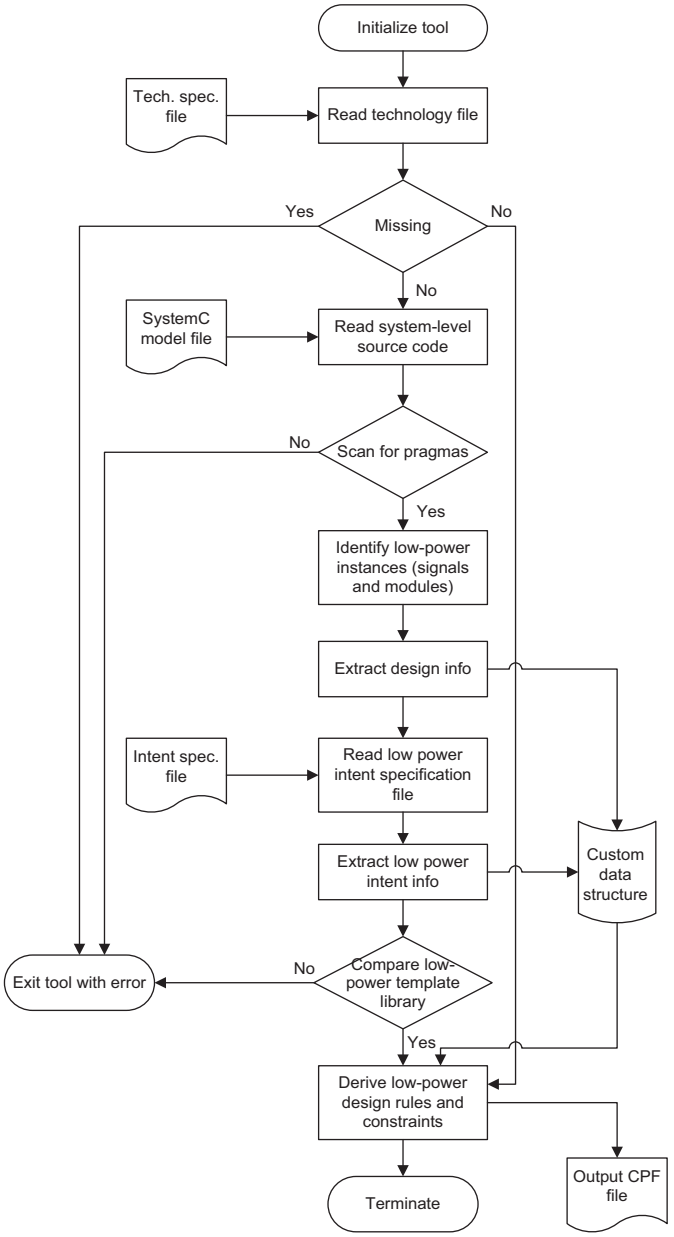

Fig. 4. CPF generation flow.

power intent file. Incremental optimization is performed and finally the gate-level netlist is obtained and checked for logic equivalence against the input RTL.

4. Design scenarios and test cases

The example cases that are adopted are simple, yet interesting enough to illustrate the proposed methodology. We start with a simple design of a hierarchical 32 bit ripple-carry adder (RCA) structure to better understand and apply the investigated methodology. The module processing the 16 most significant bits ($MSB\_RCA_{16\text{-}31}$) is selected to be placed in a switchable power domain, to enable low-power processing of 16-bit numbers. Then we move to a larger design, namely an ALU processor comprising eight modules to perform arithmetic as well as logical operations. Here, we choose to power down the division and multiplication modules when they are not needed.

Memories in a SoC can be as much or even more power hungry than the data path [14]. The dynamic power consumption associated with memory accesses may account for about one-third of the total SoC power, while the remaining two-thirds come from the clock-tree and the data path [17]. Thus, it is a good idea to also test our methodology with a design including some arrays that are mapped to RAMs. Hence, our final design is an inverse discrete cosine transform (IDCT) module, which uses a significant amount of RAM resources and finds its application in JPEG decoders.

Please note that these blocks are not meant to be representative of complex modern SOCs, but rather to be realistic SOC building blocks to which PSO and other power optimization techniques can be meaningfully applied. Our tool and methodology can be used equally well for larger or more numerous blocks, as long as their SystemC models are annotated with our proposed pragmas and our tool is used to derive the power intent.

Since our target is to test and validate the flow, we formulate synthetic application scenarios for our example designs. In this section we discuss very briefly, the test bench setup as well as the PMB design for each.

4.1. Test bench structure

The test bench setup includes a stimulus generator that drives the input control and data signals to the design under test (DUT). A monitor block logs the outputs and checks their validity, as depicted in Fig. 7.

In the case of the RCA and the ALU, the stimulus generator uses pseudo-random number generators (PRNGs) to generate input streams as well as random values of control logic to select between the power ON and OFF conditions for the switchable power domains. Of course, the probabilistic models should reflect the behavior of the real application scenarios. However, this is a well-studied problem that is outside the scope of this paper [31–33].

For the RCA and the ALU, we use both a synthetic switching activity profile in which the switchable domain stays ON for 10% of the time, one in which it is ON for 90% of the time, and one in which it is on for 50% of the time. For the IDCT design, on the other hand, we consider its real-life usage inside a JPEG decoder.

Please note that the PMB could be made more complex, in order to require some minimum number of idle cycles before shutting off the power, but again these considerations are outside the scope of the paper [28,30]. We will see later that each transition between power states requires at least four clock cycles, which would suggest considering a threshold to trigger the transition to be at least four cycles.

4.2. Design under test (DUT)

An important issue worth considering while employing power gating is to prevent floating states from propagating from the Power Switchable Domain (PSD) to the default domain. It is also necessary to save the state of some flip-flops in the design before switching off a part of the design. Isolation cells (ISO) are responsible for isolating the always-on units from the floating values of outputs from the power-gated units. They are typically placed on the outputs of the shut-off power domain during the physical placement stage [18], as shown in Fig. 8.

RET in Fig. 8 represents the state retention cells, which have the ability to retain their states even if the primary power is shut-off. The retention cells are optional and are needed only if the state of some sequential logic in the power switchable domain must be preserved. The header power switches which are inserted during the physical implementation phase provide the ability to cut-off the supply voltage to the switchable domain. The rules for the in-

```
************ INPUT File 1: system_level_model.cc ******************
#PRAGMA default-domain
SC_MODULE (topmodule)
{
    sc_in< bool > clk, reset;
    sc_signal <bool> pse_sig, iso_sig;
    ……
#PRAGMA power-domain-instance MY_DOMAIN
    idct_module XLXI_3;
    ……
    ……
    Power_control XLXI_2; // POWER MANAGEMENT UNIT
    SC_CTOR (topmodule)
    {
        ……
#PRAGMA shutoff-condition
        XLXI_2.pse(pse_sig);
        ……
#PRAGMA isolation-condition
        XLXI_2.iso(iso_sig);
        ……
    }};
```

```
************ INPUT File 2: spec_file ******************************
#PRAGMA isolation-cells
cells ISO_FENCE0N_X* EN
isolation_rule MY_DOMAIN always_ON low
```

```
************* OUTPUT File: automate.cpf ***************************
set_design topmodule

## Specify power domain ##
create_power_domain -name PD_default -default
create_power_domain -name MY_DOMAIN -instances {XLXI_3} -shutoff_condition
{!XLXI_2/pse}

## Isolation Cells ##
define_isolation_cell -cells "ISO_FENCE0N_X*" -enable EN /
-valid_location to

## Isolation rule ##
create_isolation_rule -name iso_1 -from MY_DOMAIN /
-to always_ON -isolation_condition {XLXI_2/iso_en} /
-isolation_output low -isolation_target from
```

Fig. 5. A simple example of input pragmas and corresponding CPF output.

sertion of these low power cells are stated in the power intent file provided to the logic synthesis tool.

In order to ensure no loss of data during the power-up process, we chose to add a first-in-first-out (FIFO) module at the input of the DUT, as depicted in Fig. 8. This buffers the input values while the module is still in the sleep mode and the data input is coming in during the power-up process. Moreover, clock gating is introduced by using the SystemC clock gating capability provided by the HLS tool [22].

4.3. Power management block (PMB)

Fig. 8 also shows the PMB in the default domain. This is added as a SystemC module to produce the control signals in the correct sequence to power gate the module instances of the power switchable domain(s), as indicated in the Fig. 9. The “Power control flag” in Fig. 9 represents the signal in the functional model that activates the power up/down process. This signal must be identified manually by the designer, e.g. by using activity profiling which captures the inactivity intervals of the design and hence identifies when to induce power gating.

Activity profiling plays an extremely important role while choosing a specific power optimization strategy for a design and it can be accomplished fairly accurately for a specific application [19]. The analysis of inactivity periods for a specific application usually relies on using predictive strategies either based on the past history of the idle and active states, or using the application-specific signal activity traces [20,21].

During the power down process, isolation must happen before state retention, followed by power shut-off, while the reverse sequence must be followed during the power up process. The signals for isolation, state retention and power shut-off as provided by the power control module are “iso_en”, “ret_en” and “pse” respectively, as shown in Fig. 9.

In this case the power up/down sequences consume four clock cycles, plus the number of cycles that are required to bring the power rails to the required supply voltage. Many surveys [13,35] suggest combining PSO with CG for maximizing power savings. When the switchable domain is powered down, we gate its clock nets at the time of power down. The SystemC pseudo code of our (purely illustrative) PMB block is presented as Algorithm 1.

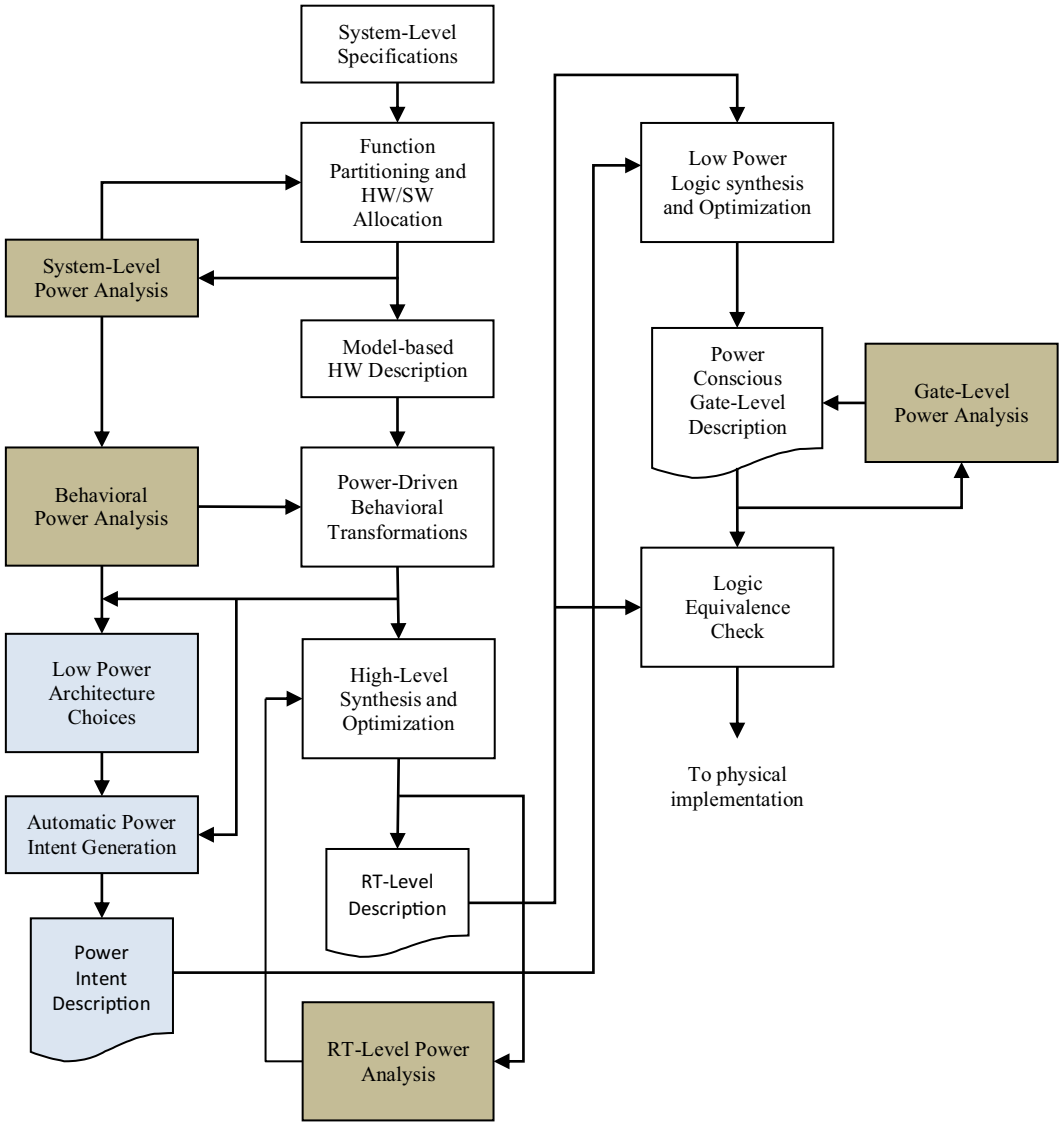

Fig. 6. Low Power High-Level Synthesis LP-HLS methodology flow.

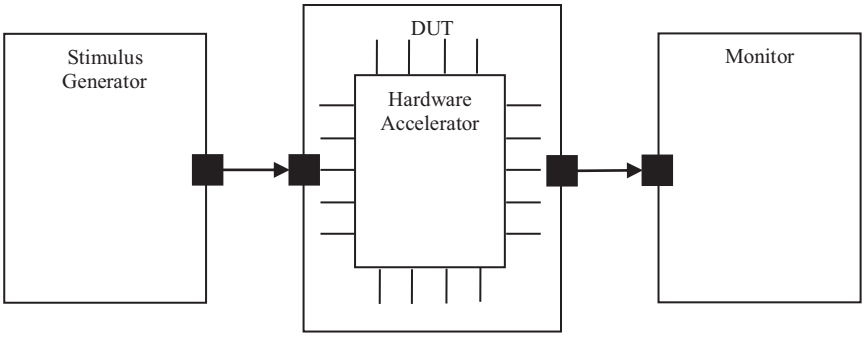

Fig. 7. General structure of test bench for design cases.

4.4. Hardware accelerators – test cases

This section briefly overviews the example cases used to validate the proposed methodology.

4.4.1. 32 bit ripple carry adder (RCA)

In order to apply the PSO technique to the RCA, it is modeled in SystemC as a hierarchical module, comprising of two 16 bit RCAs, called $MSB\_RCA_{16\text{-}31}$ and $LSB\_RCA_{0\text{-}1}$ in Fig. 10. Both modules are functionally identical, but the MSB_RCA is assigned to the switchable power domain, and hence it has a few additional ports to drive power management operations. The output multiplexers also use $P_{shut\text{-}off}$ as a select signal to choose between the valid outputs.

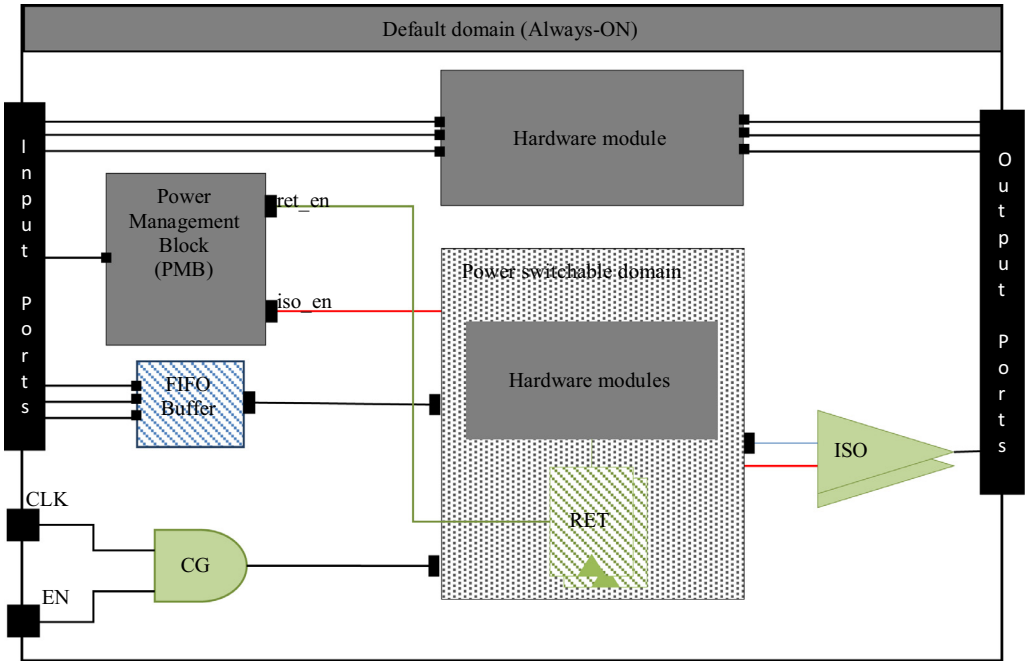

Fig. 8. General structure of DUT.

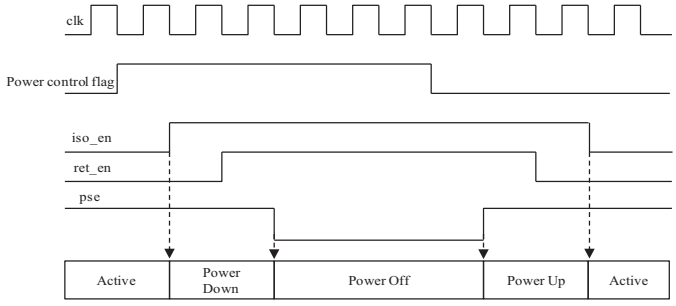

Fig. 9. Low Power High-Level Synthesis LP-HLS methodology flow.

Algorithm 1.
PMB algorithm.

```
// ********** Initialization phase ****************************************
1: Initialize signals (isolation, retention, power switch, clock gating);
2: wait(); // wait for one clock cycle
// ********** PMB logic
**********************************
3: while (true){
4:   if (Power_shut-off == Enable){
5:     isolation = ON;
6:     wait();
7:     retention = ON;
8:     wait();
9:     power_switch = ON;
10:    clk_gating = ON;
11:  }
12:  else{
13:    clk_gating = OFF;
14:    power_switch = OFF;
15:    wait();
16:    retention = OFF;
17:    wait();
18:    isolation = OFF;
19:  }
20:  }
9: end //infinite while loop
```

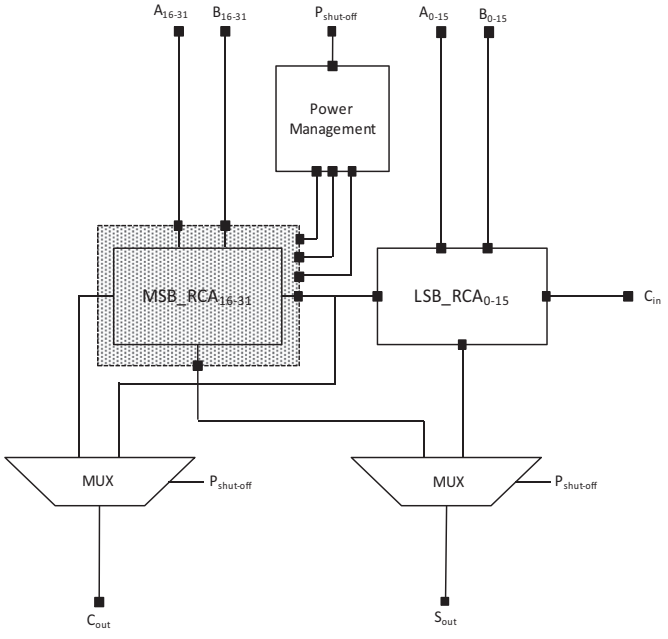

Fig. 10. 32 bit ripple-carry adder (RCA).

4.4.2. ALU processor

The ALU processor, also modeled in SystemC, is capable of performing arithmetic as well as logical operations. The encoder is driven by the control logic (SEL), from the stimulus generator and selects between unique opcodes assigned to each function.

Since the multiplication and division blocks consume most of the hardware resources, they are also the most power hungry blocks [23]. Thus, they become ideal candidates to be assigned to two separate switchable power domains as shown in Fig. 11. The power ON/OFF sequence for the respective domain is triggered based on the SEL signal.

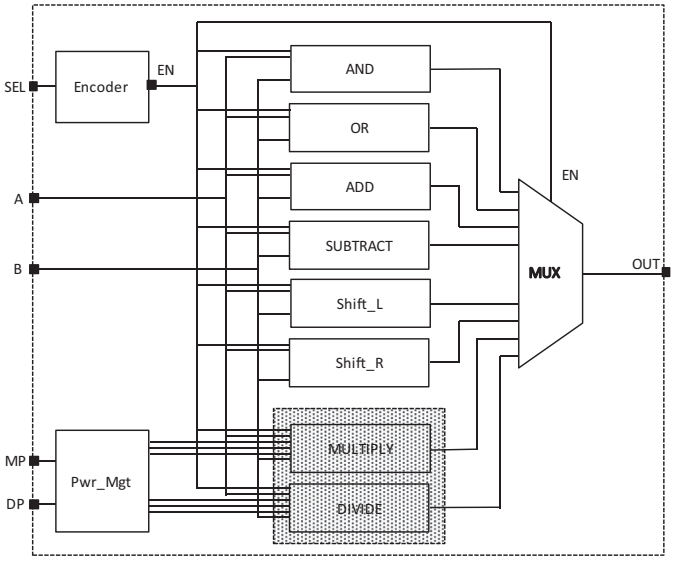

Fig. 11. Arithemetic logic unit (ALU) processor.

4.4.3. IDCT

IDCT is a well-known algorithm used in data compression standards (e.g. JPEG). A JPEG decoder performs various operations like variable length decoding (VLD), zigzag scanning (ZZ), dequantization (DQ), inverse discrete cosine transform (IDCT), color conversion, and reordering on the compressed image. A typical JPEG decoder architecture depicting all these operations is shown in Fig. 12.

This work deals with a synthesizable SystemC implementation of a JPEG IDCT decoder. A 2D-IDCT is performed by first performing 1D-IDCT on each of the columns in the matrix followed by 1D-IDCT on each row. The design architecture consists of concurrent processes with the communication between them taking place at transaction level.

IDCT is the major contributor to the overall complexity of a JPEG decoder [24]. This gives us a strong reason to power gate the IDCT unit of the JPEG decoder which will also serve in validating our LP-HLS methodology. To employ power gating, activity profiling of the IDCT design is first performed manually to identify the idle periods in the design, so as to correctly apply the power optimization strategies [2].

This is done by simulating the RTL with a dedicated SystemC test-bench and analyzing the timing diagram. This enables us to identify the idle periods in the design as well as the signal in the design that can be used to trigger the power control mechanism i.e. “Power control flag” in Fig. 9. Thus, the power control signals in this design, unlike the previous design examples (refer to Sections 4.4.1 and 4.4.2), that use synthetic data to produce the power control signals, come from real-time simulation of a JPEG-IDCT decoder. This provides us with a more realistic scenario to validate the LP-HLS methodology.

The power control signals in the correct sequence are being provided by the PMB, which is added as a separate SystemC module, as depicted in Fig. 12. These signals are used to power gate the IDCT module using the power intent captured in the CPF file. The clock gating of the design is performed by adding a separate SystemC clock gating module. This module performs coarse-grain clock gating and since it is enabled by the same signal used to trigger power gating, it clock gates all those instances that are being power gated as well. Without clock gating, the clock buffer tree continues to propagate the system clock even to the power gated modules, thus consuming dynamic power. Thus, it is a general practice to apply clock gating along with the power gating technique.

It should be noted that the power up/down sequences take four clock cycles to complete. We thus need to prevent a loss of data, which is being fed to the hardware accelerator during the power-up process, by incorporating a first-in-first-out (FIFO) buffer into the design as a separate SystemC module. In addition to the isolation cells and the power switch cells, the state retention cells already explained in previous sections, are also added in the power switchable domain here.

5. Results

The experimental setup consists of a SystemC description of the design-under-test (DUT) i.e. 32 bit RCA, ALU processor, and IDCT, respectively. The entire design flow is carried out using tools provided by Cadence Design Systems. This mainly includes Cadence C-to-Silicon Compiler (Cadence, 2014) for high-level synthesis and RTL Compiler [25] to implement the back-end flow. Several different implementations have been used to validate the results, namely the design without any power optimization techniques, and the design with clock gating and power gating. The general experimental setup uses the LP-HLS flow to automatically derive RTL for design behavior and CPF for power intent description.

The power computation is obtained from the power model included in the standard cell libraries. It is important to note that there are currently no tools/methodologies that can infer the power intent automatically from the high level model. Hence this effort, without our contribution, must be made manually at a lower level of abstraction, and working on code that has been generated automatically, and is thus very hard to understand for the designer. A high amount of design effort had been made in [2], to understand and derive design related information for low power architectures from the automatically generated RTL code, and to understand and then manually write a CPF file conferring to the intended low power implementation. The whole process in that case was extremely cumbersome and it took us several months to complete, especially due to the difficulty of understanding the automatically generated RTL code. On the other hand, with our proposed LP-HLS methodology, which automates the extraction of design-related information and generation of CPF file, the overall design effort is greatly reduced to only a few days. This is consistent with the goals of HLS-based design, which consistently reduces by a factor of 3 or 4 the design and verification time required to get to an RTL model that satisfies all specifications. Our target, therefore, with this research activity is to validate our proposed LP-HLS methodology by performing a thorough power analysis of our test cases. In this work, we used the 45 nm NanGate Open Cell Library, which supports low power cells for ASIC implementation. The power analysis using library power models relies upon the expected state of the signals at the standard cells boundary and their transition activity. As mentioned, the goal is not to show an improvement in terms of power savings with respect to the standard low-power RTL-to-GDSII flow. The goal is to show that the same improvement can be achieved by starting from system-level (e.g. SystemC), with very little effort, and without requiring the designer to understand the details of the automatically generated RTL code.

In order to verify our LP-HLS methodology, we sweep the values of both static probability and toggle rate of the power control pins of the PMB. The static probability, which dictates leakage power, accounts for the total operation workload. The toggle rate, which is associated with the dynamic power, determines the number of transitions per unit time for the power control signals.

For the RCA and ALU examples, where we are relying on synthetic input vectors, we control the utilization (static probability)

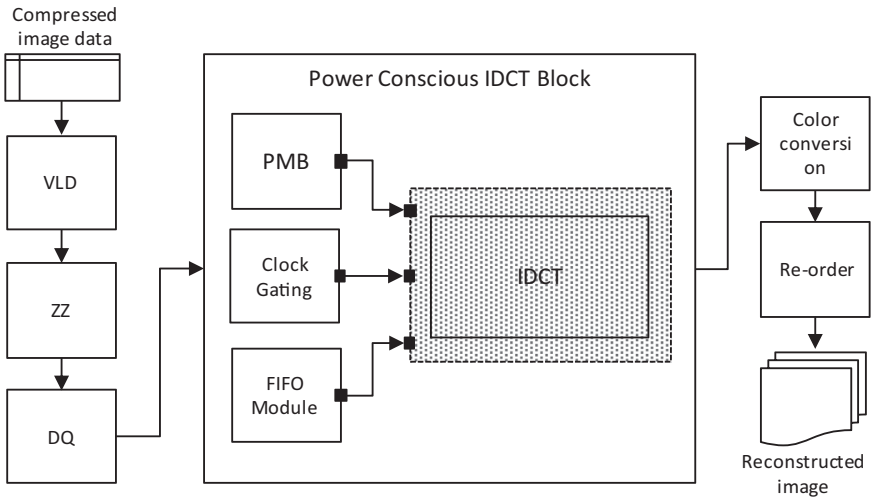

Fig. 12. JPEG decoder using IDCT module.

Table 1
Power optimization of RCA w.r.t. MSB_RCA (power switchable domain).

Operations Workload (MSB_RCA)	P_{static}(μW)	$P_{dynamic}$(μW)	Cell Area (μm^2)
Without power optimization @ 50%	70	255	Total → 3362
90%	35	104	MSB_RCA → 1051 (31%)
70%	31	97	
50%	28	93	
30%	25	90	

Table 2
Power optimization of ALU w.r.t. DIV and MULT (power switchable domains) operations.

Operations Workload (DIV – MULT)	P_{static}(μW)	$P_{dynamic}$(μW)	Cell Area (μm^2)
Without power optimization @ 10%–40%	190	1072	Total → 27,361
30% – 60%	140	341	DIV → 8219 (30%)
20% – 50%	133	283	MULT → 4790 (18%)
10% – 40%	129	237	
1% – 10%	86	160	

of the PSD by specifying a usage percentage w.r.t the rest of the design. The transition rate for the ON/OFF states of the power control signals is based on the utilization factor. On the other hand, the IDCT uses real-time input vectors from the JPEG decoder, so we keep the utilization fixed while we sweep the toggle rate to observe changes in dynamic power.

The power optimization result for the RCA example is presented in Table 1. The MSB_RCA comprises of 31% of the total size of the design, hence it is a good candidate for the power gating. The rest of the design includes the PMB, LSB_RCA and two multiplexers to compute sum, as mentioned in Fig. 10. The power analysis is performed for RCA without power optimization at 50% usage (i.e. the MSB_RCA is used in 50% of the clock cycles). From the graph in Fig. 13a, it is clear that the total power consumption is almost double as compared to the power consumption with power optimization even when MSB_RCA is active for 90% of the time, mainly because of clock gating.

Similarly for the ALU, we assign the division (DIV) and multiplication (MULT) operations to two separate PSDs. Together, they constitute 48% of the total design area as mentioned in Table 2. We then analyze the power consumption by assigning different utilization to DIV and MULT. The choice of 1% usage for DIV and 10% for MULT corresponds to a hypothetical integer workload, while the sweep between 30–10% for DIV and 60–40% of MULT can be regarded as a DSP workload. Fig. 13b shows the leakage as well as dynamic power results for the ALU. Since the cell area of the power switchable logic is almost half of the total logic, the power saving is even greater than in the previous example.

The IDCT design scenario uses vectors from the JPEG decoder to obtain power results with and without power optimization, as indicated by the first two rows of Table 3, respectively. The next three rows were obtained by increasing the toggle rate of the power shut-off signals to 4 times, 8 times and 32 times of the original toggle rates that were obtained by simulating the RTL with a realistic usage scenario. More toggling would result in the IDCT module being powered on and off more frequently and this would result in an increase in the dynamic power consumption of the design while the static power remains the same. The reason for no change in the static power is that the static probabilities of these pins remain the same for all the cases. The study of this behavior with multiple toggle rates is helpful in estimating the extent of switching beyond which any power savings would be overshadowed by the resulting dynamic power consumption. A visual comparison of the effect of power optimization for the IDCT is provided in Fig. 13c.

The total power versus RAM power and the total area versus RAM area comparison for the IDCT test case is depicted in Table 4. The purpose is to demonstrate that the IDCT test case used for validating our methodology includes a significant amount of RAM, and hence the methodology can be applicable to optimize designs where the memory accesses and its static power take a considerable amount of the total power. The dynamic and static power due to the RAM is approximately 36% and 45% of the total module power in the JPEG usage scenario. The RAM area is approximately 49% of the total area. It can be noted that after power optimization, the fraction of total dynamic power due to the RAM increases to 36% with respect to the 11% before optimization. This is because the main power-reduction contribution of power shut-off is actually to perform a very global form of clock gating, which is not supported by the current HLS tools. However, global clock gating is less effective for RAMs, since their carefully managed write enable pin already does almost-perfect clock gating.

6. Conclusions

This work proposes an LP-HLS methodology that derives power intent from the system-level description of a digital design. The

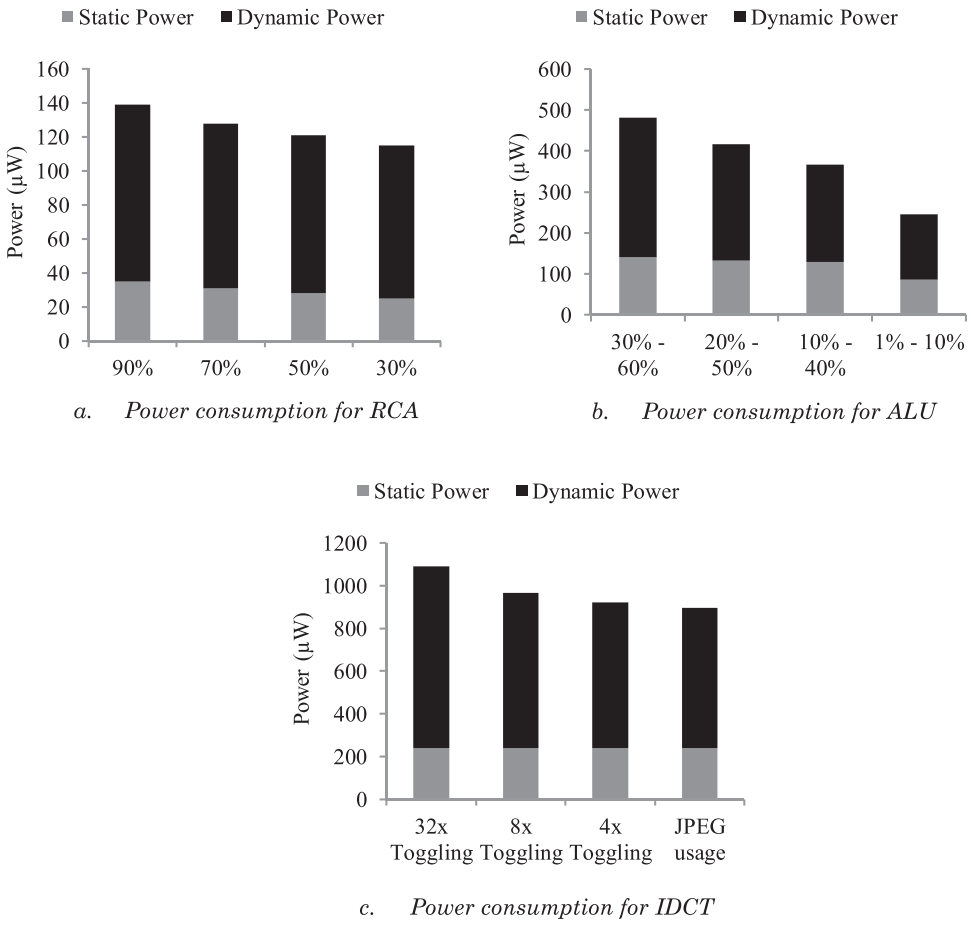

Fig. 13. Static and dynamic power consumptions for design scenarios.

Table 3
Power optimization of IDCT (power switchable domain).

Operations Workload (IDCT)	P_{static}(µW)	$P_{dynamic}$(µW)	Cell Area (µm^2)
Without power optimization	572	12,570	Total → 44,124
32x Toggling of enable	240	849	IDCT → 42,271 (96%)
8x Toggling of enable	240	726	
4x Toggling of enable	240	680	
JPEG usage	240	655	

Table 4
Total vs RAM Power and Area of IDCT (power switchable domain).

Operations Workload (IDCT)	P_{static}(µW)		$P_{dynamic}$(µW)		Cell Area (µm^2)	
	IDCT	RAM	IDCT	RAM	IDCT	RAM
Without power optimization	572	280	12,570	1396	43,919	21,156
32x Toggling of enable	240	108	849	258	44,124	21,490
8x Toggling of enable	240	108	726	250	44,124	21,490
4x Toggling of enable	240	108	680	240	44,124	21,490
JPEG usage	240	108	655	237	44,124	21,490

framework is based on (1) a generic power management module description at the system-level, to specify the design context of a hardware block, and (2) a tool that, for a given design context automatically generates the low power design directives needed to implement the PSO technique in the back-end flow. To illustrate the methodology, three example hardware accelerators ranging from simple designs to medium complexity were developed in SystemC. These include a 32 bit Ripple-Carry Adder (RCA), a general purpose ALU performing arithmetic and logical operations, and an IDCT design often used in image and video processing. The methodology aims at minimizing the design effort for low power design by deriving low-level power intent automatically for model-based designs, while using high level synthesis to achieve a broad set of target system implementations. Power analysis was carried out for the design scenarios by varying the usage of the designs. The power optimization results at the end validate the accurate derivation of power intent by using our LP-HLS methodology.

References

[1] M. Horowitz, Computing's energy problem (and what we can do about it), in: IEEE Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014, pp. 10–14.

[2] F.B. Muslim, A. Qamar, L. Lavagno, Low power methodology for an ASIC design flow based on High-Level Synthesis, IEEE 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2015.

[3] H.Y. Liu, L.P. Carloni, On learning-based methods for design-space exploration with high-level synthesis, in: Proceedings of the 50th Annual Design Automation Conference, ACM, 2013, p. 50.

[4] H.Y. Liu, M. Petracca, L.P. Carloni, Compositional system-level design exploration with planning of high-level synthesis, in: Proceedings of the Conference on Design, Automation and Test in Europe, 2012, pp. 641–646.

[5] W. Cesário, et al., Component-based design approach for multicore SoCs, in: Proceedings of the 39th annual Design Automation Conference DAC, 2002, pp. 789–794.

[6] A. Qamar, F.B. Muslim, L. Lavagno, Analysis and implementation of the Semi–Global Matching 3D vision algorithm using code transformations and High--Level Synthesis, in: Proceedings of the 81st IEEE Vehicular Technology Conference (VTC Spring), 2015, pp. 1–5.

[7] M. Kurimoto, et al., Verification work reduction methodology in low-power chip implementation, in: ACM Transactions on Design Automation of Electronic Systems (TODAES), 18, 2013, p. 12.

[8] P. Coussy, D.D. Gajski, M. Meredith, A. Takach, An introduction to high-level synthesis, IEEE Des. Test Comput. (4) (2009) 8–17.

[9] A. Mathur, Q. Wang, Power reduction techniques and flows at RTL and system-level, in: Proceedings of the 22nd IEEE conference on VLSI Design, 2009, pp. 28–29.

[10] L. Benini, G.D. Micheli, E. Macii, Designing low-power circuits: practical recipes, in: IEEE Circuits and Systems Magazine, 1, 2001, pp. 6–25.

[11] C.Y. Lin, et al., The design and experiments of a SID-based power-aware simulator for embedded multicore systems, ACM Trans. Des. Autom. Electron. Syst. 20 (2) (2015) 22.

[12] F. Schirrmeister, Design for low-power at the electronic system-level, vol1.1, Chip Vision Design Systems, 2009. White paper.

[13] L. Benini, G.D. Micheli, System-level power optimization: techniques and tools, ACM Trans. Des. Autom. Electron. Syst. 5 (2) (2000) 115–192.

[14] Z. Zhang, D. Chen, S. Dai, K. Campbell, High-level synthesis for low-power design, IPSJ Trans. Syst. LSI Des. Methodol. 8 (0) (2015) 12–25.

[15] S. Yu, IEEE Standards Association. [online] 2014, 2015 retrieved on 29 May 2015, http://standards.ieee.org/news/2014/ieee_p2415_p2416_wgs.html.

[16] E. Macii, M. Pedram, F. Somenzi, High-level power modeling, estimation, and optimization, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 17 (11) (1998) 1061–1079.

[17] D. Macko, K. Jelemenska, Managing digital-system power at the system-level, in: AFRICON, 2013, IEEE, 2013, September, pp. 1–5.

[18] R. Chadha, J. Bhasker, Architectural techniques for low power, in: An ASIC Low Power Primer, Springer, New York, 2013, pp. 93–111.

[19] G. Panic, Z. Stamenkovic, Activity profiling and power estimation for embedded wireless sensor node design, in: Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on, 2015, April, pp. 231–236.

[20] M.A. Hoque, M. Siekkinen, J.K. Nurminen, Energy efficient multimedia streaming to mobile devices—a survey, Commun. Surv. Tutor. IEEE 16 (1) (2014) 579–597.

[21] M. Kazandjieva, O. Gnawali, B. Heller, P. Levis, C. Kozyrakis, Identifying Energy Waste Through Dense Power Sensing and Utilization Monitoring, 3, CSTR, 2010 *Computer science technical report*.

[22] Cadence design systems, user manual, Cadence C-to-Silicon Compiler User Guide Product Version 13.20, 2013.

[23] T.T. Hoang, et al., Power gating multiplier of embedded processor datapath, in: IEEE 7th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME), 2011, pp. 41–44.

[24] M. Sonka, V. Hlavac, R. Boyle, Image Processing, Analysis, and Machine Vision, 4th. ed., CL Engineering, 2014.

[25] Cadence design systems, user manual, Cadence Low power in Encounter RTL *Compiler*, Product version 10.1, 2013.

[26] S. Ravi, M. Joseph, High-level test synthesis: a survey from synthesis process flow perspective, ACM Trans. Des. Autom. Electron. Syst. 19 (4) (2014) 38.

[27] J. Cong, From design to design automation, in: Proceedings of the ACM International Symposium on Physical Design, 2014, pp. 121–126.

[28] A. Bartolini, C. Hankendi, A.K. Coskun, L. Benini, Message passing-aware power management on many-core systems, J. Low Power Electron. 10 (4) (2014) 531–549.

[29] R. Ahmed, A. Bsoul, S.J. Wilton, P. Hallschmid, R. Klukas, High-level synthesis-based design methodology for Dynamic Power-Gated FPGAs, in: 24th IEEE International Conference on Field Programmable Logic and Applications (FPL), 2014, pp. 1–4.

[30] R. Ahmed, S.J. Wilton, P. Hallschmid, R. Klukas, Hierarchical dynamic power–gating in FPGAs, in: Applied Reconfigurable Computing, Springer International Publishing, 2015, pp. 27–38.

[31] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, N.P. Jouppi, McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures, in: 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-42, 2009, pp. 469–480.

[32] J.W. Tschanz, S.G. Narendra, Y. Ye, B. Bloechel, S. Borkar, V. De, Dynamic sleep transistor and body bias for active leakage power control of microprocessors, IEEE J. Solid-State Circuits 38 (11) (2003) 1838–1845.

[33] D. Brooks, V. Tiwari, M. Martonosi, Wattch: a framework for architectural-level power analysis and optimizations, ACM 28 (2) (2000) 83–94.

[34] L. Daoud, D. Zydek, H. Selvaraj, A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance computing, in: Advances in Systems Science, Springer International Publishing, 2014, pp. 483–492.

[35] G. Verma, M. Kumar, V. Khare, Low power techniques for digital system design, Indian J. Sci. Technol. 8 (17) (2015).

[36] E. Bezati, S.C. Brunet, M. Mattavelli, J.W. Janneck, Coarse grain clock gating of streaming applications in programmable logic implementations, in: Proceedings of the IEEE Conference of Electronic System-level Synthesis (ESLsyn), 2014, pp. 1–6.

[37] S. Sinha, T. Srikanthan, Dataflow graph partitioning for area-efficient high-level synthesis with systems perspective, ACM Trans. Des. Autom. Electron. Syst. 20 (1) (2014) 5.

[38] Y. Samei, R. Domer, Automated estimation of power consumption for rapid system-level design, in: Performance Computing and Communications Conference (IPCCC), 2014 IEEE International, 2014, December, pp. 1–8.

Affaq Qamar is Assistant Professor at the Department of Electrical Engineering at Abasyn University Peshawar, Pakistan, since 2016. He graduated from the Politecnico di Torino, Italy in Dec 2015 with a Ph.D. in Electronics and Telecommunications. He has received his MS degree in Integrated Electronic System Design from Chalmers University of Technology in 2010. His research interests are design methodologies for electronic system design, high-level synthesis, low power design architectures for SoCs, embedded system design for real-time applications, and green energy.

Fahad Bin Muslim is a Ph.D. student at the Department of Electronics and Telecommunications (DET) at the Politecnico di Torino, Italy working under the supervision of Professor Luciano Lavagno. He received his MS degree in Communication Engineering from Chalmers University of Technology in 2010. His research interests include electronic design automation with emphasis on low power designs.

Javed Iqbal is a Ph.D. student and Research Fellow at Department of Electronics and Telecommunications, Politecnico di Torino, Italy. He got his Masters of Science in Telecommunications Engineering in 2014 from Politecnico di Torino. He is currently working on the design and implementation of low-power wireless sensor nodes for human detection, localization and identification.

Luciano Lavagno received his Ph.D. in EECS from U.C.Berkeley in 1992. He co-authored four books and over 200 scientific papers. He was the architect of the POLIS HW/SW co-design tool. Between 2003 and 2014 he was an architect of the Cadence CtoSilicon high-level synthesis tool. Since 1993 he is a professor with Politecnico di Torino, Italy. His research interests include synthesis of asynchronous circuits, HW/SW co-design, high-level synthesis, and design tools for wireless sensor networks.